Automatic Discovery of Cognitive Strategies with Tiny Recurrent Neural Networks

Li Ji-An[1], Marcus K. Benna[2], and Marcelo G. Mattar[3, 4, @]

[1]Neurosciences Graduate Program, University of California San Diego, La Jolla, CA 92093
[2]Department of Neurobiology, School of Biological Sciences, University of California San Diego, La Jolla, CA 92093
[3]Department of Cognitive Science, University of California San Diego, La Jolla, CA 92093
[4]Department of Psychology, New York University, New York, NY 10012
[@]corresponding author: marcelo.mattar@nyu.edu

Abstract

Normative modeling frameworks such as Bayesian inference and reward-based learning provide valuable insights into the fundamental principles of adaptive behavior. However, their ability to describe realistic animal behavior is limited by the typically small number of fitted parameters, leading to a cycle of handcrafted adjustments and model comparisons that are prone to research subjectivity. Here, we present a novel modeling approach leveraging recurrent neural networks to automatically discover the cognitive algorithms governing animal decision-making. We show that neural networks with only one or two units can predict choices of individual animals more accurately than classical cognitive models, and as accurately as larger neural networks, in three well-studied reward learning tasks. We then interpret the trained networks using dynamical systems concepts such as state-space and fixed-point attractors, leading to a unified comparison of different cognitive models and a detailed characterization of the cognitive mechanisms underlying the animal's choices. Our approach also estimates behavior dimensionality and provides insights into the algorithms emerging in meta-reinforcement learning agents. Overall, we present a systematic approach for discovering interpretable cognitive strategies in decision-making, offering insights into neural mechanisms and a foundation for examining both healthy and dysfunctional cognition.

Introduction

Understanding the neural basis of adaptive behavior is a central objective in neuroscience and cognitive science. Researchers have long strived to develop computational models that encapsulate the complexities of learning and decision-making, from early symbolic models[1] to connectionist approaches[2]. Recently, normative frameworks such as Bayesian inference[3,4] and reward-based learning[5–8] have gained prominence for their ability to elucidate the fundamental principles governing adaptive behavior. These cognitive models formalize how agents accumulate and apply knowledge from environmental interactions for decision-making, a process carried out by neural circuits in the prefrontal cortex and striatum[9–13]. A key advantage of these models is their simplicity, as they typically have few parameters and can be easily augmented by additional assumptions such as forgetting, choice biases, choice perseveration, exploration, and capacity limitations[12,14]. However, this simplicity and extensibility also make these models prone to bias and researcher subjectivity, potentially leading to incorrect or incomplete characterizations of behavior[15].

An alternative modeling framework employs artificial neural networks, a class of computational models consisting of interconnected neuron-like units that can express a wide range of functions. In comparison to classical cognitive models, neural networks impose fewer structural assumptions, require less handcrafting, and provide a more flexible framework for modeling behavior and neural activity[16,17]. A common way of using neural networks is to adjust the parameters to produce optimal behavior in a given task. This approach has been used in neuroscience to explain the neural activity associated with vision,[18,19], spatial navigation[20–22], learning, decision-making, and planning[23–27]. An alternative way of using neural networks is to fit the parameters directly to observable human or animal behavior. This approach can lead to models capable of highly accurate predictions of future behavior, due to the large number of fitted parameters[28–31]. However, this increased flexibility presents challenges such as difficulty in interpreting the fitted networks, hindering the identification of cognitive and neural mechanisms underlying observable behavior.

In this study, we present a novel modeling framework combining the flexibility of neural networks with the interpretability of classical cognitive models. Our approach involves fitting recurrent neural networks (RNNs) to the behavior of individual subject's in reward learning tasks. RNNs are a type of neural network with temporal dynamic behavior due to cyclical connections between nodes. The fitted networks describe how the individual accumulates knowledge through

environmental interactions and applies it in decision-making. In contrast to previous work, however, we use very small networks, often composed of only one or two units, which facilitates interpretation. We then develop an interpretive framework based on concepts from discrete dynamical systems theory, a branch of mathematics that describes how systems (e.g., decision-making agents) evolve over time in response to external inputs (e.g., task states, rewards). In a dynamical system, inputs affect the current state of the system (e.g., the agent's beliefs about the task), which in turn determine the system's subsequent behavior (e.g., the agent's decisions). In this framework, the state-space is a collection of all possible system states, and attractors are specific points or regions in the state-space to which the system tends to converge, representing stable configurations or patterns of behavior. We show that this framework not only enables a unified comparison of different model classes but also facilitates a systematic investigation of the dynamics of the fitted RNNs.

Our results show that, across five datasets involving three animal species, RNNs with just one or two units provide a superior fit to animal behavior than classical cognitive models of equal dimensionality. The minimum number of units required to optimally fit behavior provides an estimate of the dimensionality of behavior in the tasks studied. We then illustrate how the framework of dynamical systems can be used to interpret the fitted networks, leading to direct insights into the cognitive mechanisms underlying decision-making. In particular we identify several novel behavioral patterns overlooked by classical cognitive models, including variable learning rates, a novel confirmation effect, and previously unrecognized forms of value drifting, choice perseveration, and choice biases. Overall, our findings suggest that behavioral modeling based on tiny RNNs yield not only better predictions of behavior than classical cognitive models, but also deeper insights into the underlying cognitive mechanisms, addressing both the interpretability challenges of large neural networks and the subjectivity of classical cognitive models. These results hold significant implications for the fields of computational neuroscience, cognitive science, and computational psychiatry, with the potential to transform our understanding of adaptive decision-making mechanisms and to guide future research directions.

Results

Reward learning tasks

We studied animal behavior in three reward learning tasks featuring increasing levels of complexity: a reversal learning task[11], a two-stage task[13], and a transition-reversal two-stage task[12]. These tasks have been widely utilized in neuroscience and psychology studies to examine how subjects adapt their behavior to changing reward and state contingencies (Fig. 1a). The three tasks share a common structure: in each trial, the subject chooses between action A_1 and A_2, resulting in second-stage states S_1 or S_2 that are each associated with a distinct probability of receiving a unit reward. In neutral trials, both second-stage states offer the same reward probability (e.g., 0.5); in non-neutral trials, one state offers a higher reward probability than the other state (e.g., 0.8 for S_1 and 0.2 for S_2). Reward probabilities remain constant for several trials before abruptly switching their assignments to second-stage states, a moment termed "reversal". In all tasks, the subject's objective is to choose the action currently offering the highest likelihood of yielding a reward.

The three studied tasks also have unique characteristics. In the reversal learning task, each action leads deterministically to one state (e.g., A_1 to S_1 and A_2 to S_2). In the two-stage task, each action leads to one state with high probability (e.g., A_1 leads to S_1 with probability 0.8, a "common" transition) and to another state with low probability (e.g., A_1 leads to S_2 with probability 0.2, a "rare" transition). In the transition-reversal two-stage task, actions similarly lead to one state with high probability and to another with low probability, but the state associated with high/low probability also switches at other random times (e.g., A_1 leads to S_1/S_2 with high/low probability for a few trials, and after a switch, A_1 leads to S_1/S_2 with low/high probability).

We analyzed data from two monkeys (Bartolo dataset[11]) and ten mice (Akam dataset[12]) performing the reversal learning task; from four rats (Miller dataset[13]) and ten mice (Akam dataset[12]) performing the two-stage task; and from seventeen mice (Akam dataset[12]) performing the transition-reversal two-stage task.

Computational models of decision-making

Our approach involves using RNN models to analyze the cognitive mechanisms underlying animals' choices. As a benchmark, we also implemented 30 classical cognitive models previously proposed to describe behavior in reward learning tasks, including Bayesian learning models, RL models, and variants of these models (see Methods). All cognitive models and RNNs share the same input-output structure (Fig. 1b-d). The inputs consist of the previous action a_{t-1}, the previous second-stage state s_{t-1}, and the previous reward r_{t-1} (note that the current state is not specified as an input because it is always the same state, called "choice state"). These inputs update the agent's internal state, which in turn determines the agent's output — a probability distribution over actions, or "policy". The agent's internal state is described by a set of d internal *dynamical variables* that summarize the agent's prior experience (e.g., action values, belief states, etc.) to guide future decisions. Although all models share the same input-output structure, they differ from one another with respect to the rule specifying how each dynamical variable is updated given inputs, which in turn leads to different policies. We note that the

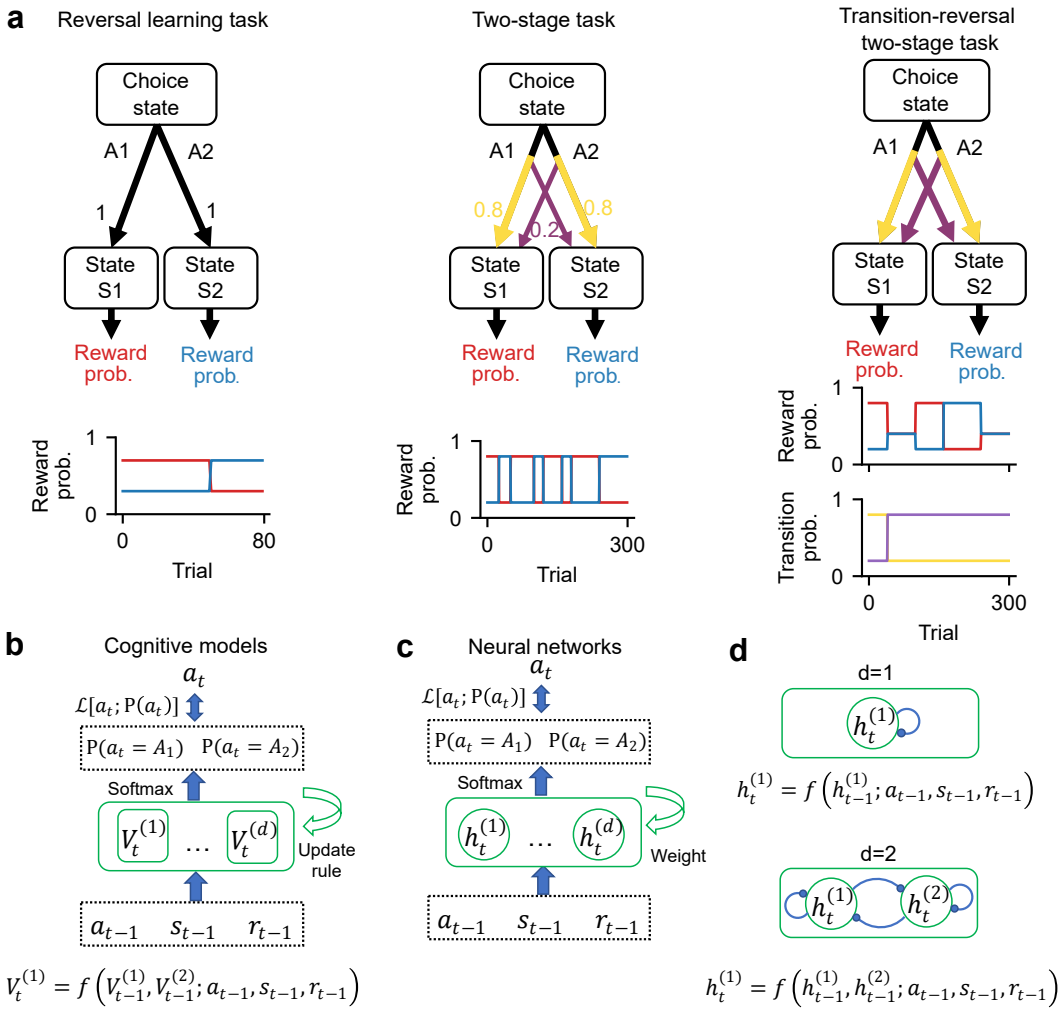

Fig. 1. Task structures and model architectures. **(a)** In each trial of the three tasks, subjects choose action A_1 or A_2 at the choice state, transitioning into one of two second-stage states, S_1 or S_2, which probabilistically yield a reward. Reward probabilities change over time. In the transition-reversal two-stage task, the action-state transition probabilities also change over time. **(b-d)** Cognitive models and neural networks have similar architectures. Inputs consist of the previous action a_{t-1}, the previous second-stage state s_{t-1}, and the previous reward r_{t-1}. The model's internal d dynamical variables $(V_t^{(1)}, \cdots, V_t^{(d)})$ or $(h_t^{(1)}, \cdots, h_t^{(d)})$ are updated based on the inputs and on their previous values. The model's outputs are action probabilities computed from the dynamical variables via the softmax function. The parameters of each model are optimized with the goal of maximizing the probability $\Pr(a_t)$ of the observed action a_t. **(b)** In cognitive models, dynamical variables are designed to represent state values, action values, state-transition probability, latent-state probability, choice perseveration, among others. Each model specifies a different rule for updating the dynamical variables. In the example equation, corresponding to a cognitive model with $d = 2$ dynamical variables, $V_t^{(1)}$ (e.g., an action value) is a function of previous values $V_{t-1}^{(1)}$ and $V_{t-1}^{(2)}$, and of the input $(a_{t-1}, s_{t-1}, r_{t-1})$. **(c)** In neural networks, the activity of each recurrent unit corresponds to a dynamical variable. These variables are not constrained *a priori* to represent any given quantity. **(d)** *Top:* Example of recurrent layer comprised of $d = 1$ unit. The unit activation $h_t^{(1)}$ is a function of previous unit activation $h_{t-1}^{(1)}$ and of the input $(a_{t-1}, s_{t-1}, r_{t-1})$. *Bottom:* Example of recurrent layer comprised of $d = 2$ units. The unit activation $h_t^{(1)}$ is a function of previous unit activations $h_{t-1}^{(1)}$ and $h_{t-1}^{(2)}$, and of the input $(a_{t-1}, s_{t-1}, r_{t-1})$.

dynamical variables hypothesized in a model are distinct from the model parameters: dynamical variables evolve over time, representing the agent's current beliefs about the task and dictating their actions; model parameters, in turn, are stable and specify the rules by which the dynamical variables evolve over time.

The classical cognitive models considered here can be classified into three model families[32]: model-free, model-based, and latent-state (Fig. 1b). The model-free family uses reward input to directly update the model's dynamical variables — state or action values. The model-based family uses a state-transition model (either learned or assumed) to update its dynamical variables — state or action values. In both model-free and model-based families, dynamical variables are updated via conventional RL algorithms. The latent-state model family, in contrast, uses Bayesian inference to update the model's dynamical variable — a belief state specifying which second-stage state the agent currently believes has a higher reward probability. We also consider various common augmentations of these classical cognitive models, where additional dynamical variables are used to capture multi-trial choice perseveration, learned state-transition probabilities, among others (see Methods). As mentioned previously, each family and variant of a cognitive model instantiates a different rule for updating its dynamical variables (e.g., learning rules in RL, or inference rules in Bayesian models), and these rules are typically described by a few, interpretable parameters.

The RNN models we consider utilize the same inputs as the cognitive models (Fig. 1c-d). These inputs influence the activity of the recurrent units through a set of input weights; recurrent units, in turn, determine the network's output (a policy) via a set of output weights. Each recurrent unit represents one dynamical variable, and the total number of units, d, determines the dimensionality of the network's state-space. In our models, we use a type of network unit called the gated recurrent unit (GRU). Like in any recurrent network, the activity of a unit in the GRU is a function of both the network inputs and the previous activity of all recurrent units. However, due to a pre-determined architecture, these functions in GRUs use gates to easily select which past information to retain or discard when incorporating inputs[33]. The function implemented by each unit is specified by a set of adjustable parameters, allowing RNN models to learn a wide range of functions mapping past inputs (observations) into outputs (policies). Importantly, the use of GRUs means that the set of d unit activations fully specifies the network's internal state, rendering the system Markovian. This is in contrast to alternative RNN architectures such as the long short-term memory, where the use of a cell state renders the system non-Markovian[34]. We note, however, that other flexible recurrent architectures could have been used and are worthy of exploration (see Discussion).

Tiny RNNs outperform classical cognitive models in predicting choices

To analyze the experimental data and study the mechanisms underlying animals' choices, we optimized the parameters in each cognitive model and RNN to predict the animal's recorded choices with maximum likelihood (equivalent to minimizing cross-entropy). Note that the RNN models have substantially more free parameters than the classical cognitive models — roughly 40-80 parameters for 1-unit or 2-unit RNNs, vs. 2-10 parameters for cognitive models (see Fig. S1). To ensure a fair comparison between models with such wide differences in number of parameters, we used a nested cross-validation procedure where models are optimized on a training dataset, selected on a validation dataset, and evaluated on an unseen test dataset (note that conventional methods such as AIC or BIC are not appropriate in this setting; see Methods). We then averaged the predictive performance of each model across subjects, yielding a predictive performance for each model in each dataset: monkeys (Bartolo dataset) and mice (Akam dataset) performing the reversal learning task; rats (Miller dataset) and mice (Akam dataset) performing the two-stage task; and mice (Akam dataset) performing the transition-reversal two-stage task (Fig. 2a-c). Our primary focus was on comparing models with an equal number of dynamical variables d, as these models use an equal number of scalar variables to summarize the agent's past and to specify the agent's policy. As we will soon show, models of equal d can also be interpreted using identical methods, and can be straightforwardly compared with one another.

We found that very small RNNs were better at predicting animals' choices than all of the tested classical cognitive models from model-free, model-based, and Bayesian latent-state families, including all of their commonly adopted variants (see Methods). At the group level (i.e., model's predictive performance averaged across subjects), RNNs (GRUs) with 2 units were best at predicting choices in the reversal learning and two-stage tasks, and RNNs with 4 units were best at predicting choices in the transition-reversal two-stage task (Fig. 2a-c; note that the lowest negative log-likelihood is achieved by the GRU). These networks also achieved comparable predictive performance to larger RNNs (e.g., RNNs with 10 or 20 units, as used in[29]), suggesting that animal behavior in these tasks was low-dimensional (Fig. 2a-c; note the tendency of the blue line to flatten out or curve upwards as d increases). Furthermore, RNNs with d units consistently outperformed all classical cognitive models with d variables (Fig. 2a-c; note that colored dots are always above the blue line). At the individual level, the behavior of most animals was best explained by RNNs with 1-2 units in the reversal learning task, 2 units in the two-stage task, and 2-4 units in the transition-reversal two-stage task (see Fig. S2).

The fitted RNNs reproduced several metrics typically used to analyze each task: choice probabilities before and after the reversal in the reversal learning task[11] (Fig. 2d), stay probability the in two-stage task[13] (Fig. 2e), and stay probability in the transition-reversal two-stage task[12] (Fig. 2f). These findings demonstrate that tiny RNNs are versatile models of behavior, not only reproducing well-known behavioral patterns in reward learning tasks, but also capturing more variance in animal

behavior than classical cognitive models and their variants (e.g., models augmented with forgetting, perseveration, etc.).

A noteworthy observation is that incorporating more dynamical variables did not always yield superior predictive performance, and in some cases performance even declined. The number of dynamical variables for which predictive performance is maximized provides an estimation of the dimensionality of behavior d_*, defined as the minimal number of functions of the past required to accurately predict future behavior as well as possible[35,36]. In RNN models, if the number of dynamical variables is lower than d_*, including more variables in the model should increase predictive performance up to d_* dynamical variables. Using more variables than d_* should not improve performance significantly and may even hurt it due to the lack of training data (see "Nested cross-validation procedure" in Methods). Reinterpreting the previous results in terms of dimensionality of behavior, we find that the behavior of most animals in the reversal learning and two-stage tasks has dimensionality d_* of 1 or 2, while the behavior of most animals in the transition-reversal two-stage task has dimensionality d_* ranging between 2 and 4. As we will show, low-dimensional tasks — such as the three tasks studied here, as well as the majority of laboratory tasks used in neuroscience and psychology studies — are particularly well-suited for RNN models with few dynamical variables. However, our tiny RNNs can nonetheless yield useful insights into cognitive strategies even if the behavior is higher dimensional.

Tiny RNNs excel with commonly-sized datasets and improve as more data are used

The superior performance of tiny RNNs compared to classical cognitive models can be attributed to the increased flexibility afforded by a greater number of free parameters. This flexibility allows a single RNN to be molded into a wider range of behaviors than possible with classical cognitive models. However, this flexibility also raises important concerns such as the risk of overfitting and the requirement for large amounts of training data. While nested cross-validation and early stopping eliminate the risk of overfitting (see also next section for empirical evidence), we aimed to determine the amount of data required for training RNNs compared to classical cognitive models.

Our analysis of representative animals from each dataset showed that, as expected, both RNNs and cognitive models generally improved their predictive performance with increasing amounts of available data (Fig. 3a-c). Notably, cognitive models achieved higher performance for small datasets. RNNs only outperformed cognitive models of equal dimensionality when a sufficient number of trials were available for training and validation: approximately 1,000 trials for the reversal learning task (Fig. 3a), 500 trials for the two-stage task (Fig. 3b), and 3,000 trials for the transition-reversal two-stage task (Fig. 3c). However, these analyses also revealed another important trend: while cognitive models reach their peak performance for relatively small datasets, RNNs continue to improve as more data are provided. For instance, in the reversal learning task, the performance of the 2-unit RNNs improved when the number of training trials increased from 1,000 to 4,000, and the absence of an asymptotic behavior suggests further improvement would be possible with even larger datasets. Similar trends were observed for the 2-unit RNNs in the two-stage task and for the 4-unit RNNs in the transition-reversal two-stage task. These results suggest that the increased flexibility of RNNs enables them to capitalize on additional data to achieve higher performance.

The requirement of $500 - 3,000$ trials per animal before RNNs start outperforming cognitive models may appear restrictive. Even though datasets of this size are common in animal experiments, human experiments typically involve much less data per participant, drawing on data from multiple participants to achieve statistical power. Importantly, we discovered that tiny RNNs can also improve individual-level fits by leveraging data from multiple participants, through a knowledge distillation framework[37], leading these models to outperform cognitive models with much less data per participant. This approach involves first training a single, large, but less interpretable RNN (the "teacher network") to predict choice data from all subjects, using a subject embedding layer to capture subject-specific information (Fig. 3d, *left*). We then train a small and more interpretable RNN (the "student network") to predict the policies for a single subject from the teacher network (Fig. 3d, *right*). The rationale is that the probabilities given by the teacher policy offer a richer training target for the student network than the binary subject choices, allowing subject-specific RNNs to achieve higher predictive performance for a given dataset size.

To evaluate this approach, we applied it to a representative mouse performing the transition-reversal two-stage task in the Akam dataset, which contained the largest number of subjects in our study (Fig. 3e). While this mouse in this dataset performed over 18,000 trials, we assessed the performance of the networks when using only a subset of these data. We discovered that the teacher RNN outperformed all cognitive models on unseen test data of this mouse, even when as few as 350 trials from the mouse were used (the smallest amount of training data tested). More importantly, we also found that the student RNN, trained exclusively on trials from this mouse, achieved a performance close to that of the teacher RNN, outperforming all cognitive models as well as the original tiny RNN trained on the very same trials (solo GRU). This suggests that subject-specific tiny RNNs can outperform cognitive models even when as few as 350 trials are available per subject (compared to the requirement of 3,000 trials for solo GRUs), provided that enough trials from other subjects are available.

In summary, our findings demonstrate that tiny RNNs, when trained using a knowledge-distillation approach, surpass classical cognitive models even with dataset sizes commonly found in human experiments. We note that the knowledge-distillation method highlighted in this section is not utilized in the following sections.

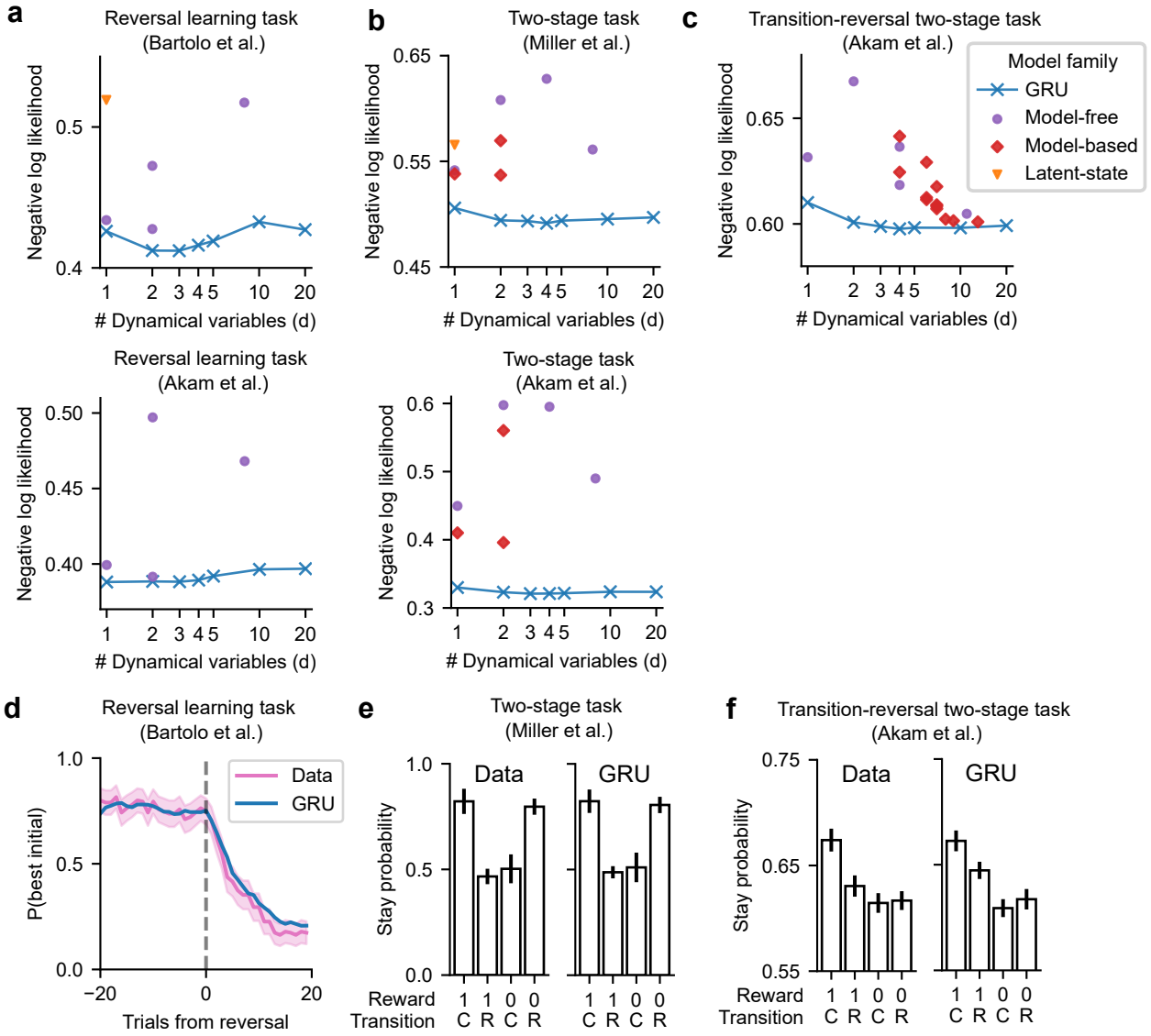

Fig. 2. Tiny RNNs outperform classical cognitive models in predicting animals' choices. (a-c) Predictive performances (trial-averaged negative log-likelihood; lower is better) in three tasks (one per column) and five different datasets (one per panel). Performances are displayed as a function of the number of dynamical variables d. The reported test performance for each model is the average over 10 outer folds in the nested cross-validation. Each dynamical variables in the GRU model family corresponds to one network unit; in cognitive models, dynamical variables can represent action values, state values, multi-trial choice perseveration, learned state-transition probabilities, among others. Each model family may include multiple model variants, leading to two or more identical markers for a given d. **(a)** *Top:* In the reversal learning task with monkeys, the GRU model with $d = 2$ outperforms all other models. *Bottom:* In the reversal learning task with mice, the GRU models with $d = 1$ or $d = 2$ outperform all other models. **(b)** *Top:* In the two-stage task with rats, the GRU model with $d = 2$ outperforms all other models. *Bottom:* In the two-stage task with mice, the GRU model with $d = 2$ outperforms all other models. **(c)** In the transition-reversal two-stage task with mice, the GRU model with $d = 4$ outperforms all other models. **(d-f)** Tiny RNNs reproduced several behavioral metrics typically used to analyze each task. **(d)** Probability of selecting the action with higher reward probability before the reversal (best initial) in the reversal learning task. GRU model uses $d = 2$. Shaded region is the 95% confidence interval for monkeys across blocks. **(e)** Probability of taking the same action following each trial type (stay probability) in the two-stage task. Trial types are 'C-1' (Rewarded-Common transition), 'R-1' (Rewarded-Rare transition), 'C-0' (Unrewarded-Common transition), 'R-0' (Unrewarded-Rare transition). GRU model uses $d = 2$. Error bars show cross-subject SEM. **(f)** Stay probabilities in the transition-reversal two-stage task. GRU model uses $d = d_*$, the dimensionality of the behavior of each mouse. Error bars show cross-subject SEM.

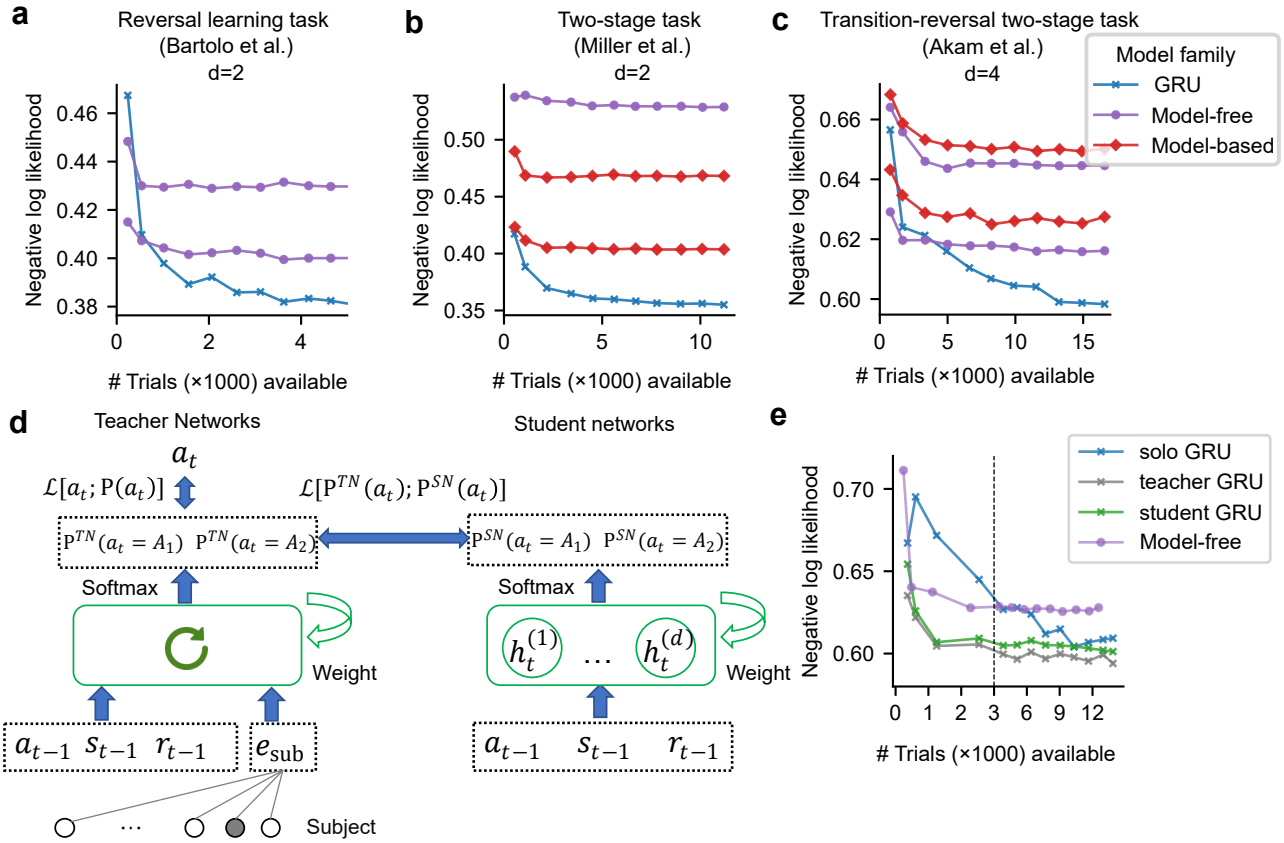

Fig. 3. The predictive performance of tiny RNNs improves as the amount of available data increases. (a-c) Performance of each model (negative log-likelihood) when trained with varying amounts of data (number of trials for training and validation) to predict the choices of a representative animal in each dataset. The reported test performance for each model is the average over 10 outer folds in the nested cross-validation. Each model family may include multiple model variants, leading to two or more identical markers for a given d. **(a)** Two-dimensional models (GRU: two units; model-free: two action values) applied to reversal learning task data from a representative monkey (Bartolo et al.). **(b)** Two-dimensional models (GRU: two units; model-free: two action values; model-based: two state values) applied to two-stage task data from a representative rat (Miller et al.). **(c)** Four-dimensional models (GRU: four units; model-free: two action values and two state values; model-based: two state values and two state-transition probabilities) applied to transition-reversal two-stage task data from a representative mouse (Akam et al.). **(d)** Knowledge distillation framework featuring a large teacher network (TN) and a tiny student network (SN). The teacher network is trained on choice data from multiple subjects. Each subject (shown as circles) is represented by a subject-embedding vector e_{sub} that provides an additional input to the recurrent layer. The student networks are trained to predict the output probabilities generated by the teacher network, using data from a single subject. **(e)** The performance of each model (including the models using the knowledge distillation framework) in predicting the choices from a representative mouse in the transition-reversal two-stage task. Each model is trained with varying amounts of data (number of trials for training and validation) from the representative mouse. The teacher GRU is additionally trained (and validated) on all trials from all other animals. Note that the student GRU outperforms the best model-free model for all tested dataset sizes. The x-axis has a different scale for the low-data (fewer than 3000 trials) and high-data (more than 3000 trials) regions to enhance visualization and clarity. The number of available trials and performance curves here are different from that in (c), due to the training-validation-test procedure for accommodating knowledge distillation being different from the nested cross-validation (see Methods).

Tiny RNNs can identify ground-truth strategies in simulated behavior

The results thus far suggest that animals exhibit behavioral patterns that are better captured by flexible RNN models than by cognitive models designed from normative considerations. What if animals exhibited a different behavioral pattern other than observed in these datasets, such as those exhibited by an RL or Bayesian inference agent? One possibility is that RNN models can only approximate, but not fully account for, the behavior of such agents. In this scenario, a cognitive model well-matched to the animal's behavior would outperform the RNN models. Alternatively, the RNN models might be flexible enough to fully account for the behavior of an RL or Bayesian inference agent. In this scenario, the RNN models may be viewed as a superset of the cognitive models, and their predictive performance would be no worse than that of any cognitive model.

To determine how well can an RNN model predict the behavior of RL and Bayesian inference agents, we simulated the behavior of such agents over 10,000 choices in the two-stage task, with parameters fitted to one example rat from the Miller dataset. By design, thus, these choices embodied the behavior of a normative RL or Bayesian inference agent with one or two dynamical variables. We then fitted all cognitive models and RNN models to optimize predictive performance for this simulated behavior. We found that the best RNN models achieved identical predictive performance as the ground-truth model of equal dimensionality that generated the behavior (Fig. 4). Furthermore, these RNNs accurately learned the ground-truth model's dynamics (see next section and Fig.S3). This demonstrates the remarkable flexibility of RNN models, which can describe diverse behavioral strategies and serve as a superset of classical cognitive models, despite using a consistent architecture that requires virtually no manual engineering.

Incidentally, these results also demonstrate that the nested cross-validation procedure employed in all of our results successfully prevented overfitting, which would have been diagnosed if the more flexible RNNs had achieved higher performance on the training dataset (and lower on the test dataset) than the data-generating model.

Leveraging dynamical systems theory for model interpretation and comparison

Our results thus far demonstrate that tiny RNNs can predict animals' choices better than classical cognitive models, that these models can be trained with typically sized datasets, and that they can be molded to exhibit RL or Bayesian inference behavior when needed. Despite the impressive performance of neural networks, an often-cited barrier to their widespread use in cognitive modeling is their limited interpretability. In the next sections, however, we show that the small size of our RNNs facilitates the interpretation of their dynamics in terms of cognitive processes. Our interpretative framework is grounded in the theory of discrete dynamical systems[38], which uses difference equations to describe how the state of a system changes over time as a function of the inputs. We begin by describing the fundamentals of this framework and illustrating how it can be used to analyze and compare the behavior of cognitive models and RNNs.

To illustrate the dynamical systems framework, we first use it to interpret classical cognitive models. Any cognitive model can be viewed as a dynamical system, where the model's equations describe the temporal evolution of the system when presented with different inputs. The state of the system consists of the agent's beliefs, and is represented by dynamical variables (e.g., action values in RL; posterior probabilities in the latent-state model). For a one-dimensional model, an equivalent, unifying way of describing the state of the system is in terms of the logit (log-odds) of the policy — i.e., the logit at trial t is defined as $L(t) = \log\left(\Pr_t(A_1)/\Pr_t(A_2)\right)$, representing the agent's preference for one action over the other. The evolution of the system, in turn, can be described by the logit-change, the difference between logits in consecutive trials — i.e., the logit-change at trial t is defined as $\delta L(t) = L(t+1) - L(t)$. Together, logit and logit-change represent the state and the evolution of the system, akin to the "position" and "velocity" of a physical system (Fig. 5a).

We computed logit and logit-change for a model-free RL and a latent-state model with $d = 1$, with parameters fitted to the choices of one monkey in the reversal learning task. We then plotted logit-change against logit over trials, grouped according to the four input possibilities I (combinations of actions and rewards) — understood as the "force" acting on the system. These plots, called phase portraits, describe how the system, in any possible state, evolves when receiving an input. As such, they provide a complete characterization of the system and enable the visual identification of several important characteristics. For example, the fixed points for input I are the states L^*_I characterized by a zero logit-change (i.e., points in the phase-portrait that cross the x-axis). When the system is in one such state and receives input I, the system remain in that state. Attractors are states the system tends to converge to. The learning rate of the system, defined as the magnitude of state change (logit-change) caused by a prediction error, is related to the slopes at each position L, or $d\delta L/dL$ (see Methods).

By comparing the phase portraits of the fitted model-free RL and latent-state models, we can identify similarities and differences in how these models process input and handle uncertainty (Fig. 5b). For example, consider the predictions of each model for how the system should evolve when the same input I (i.e., actions and rewards) is given repeatedly (see Fig. 5a for a schematic). In the phase portrait, choose any initial logit state (a value along the x-axis) and identify the logit-change (y-axis) corresponding to input (color) I. The state resulting from input I can be identified by adding the logit-change to the initial logit value, leading to an updated logit value (x-axis). From this new logit value (x-axis), identify the new logit-change (y-axis) corresponding to input (color) I, and repeat this process until convergence. This approach reveals that, in both models, repeatedly selecting a given action and receiving a reward, the system drifts towards a fixed point corresponding to

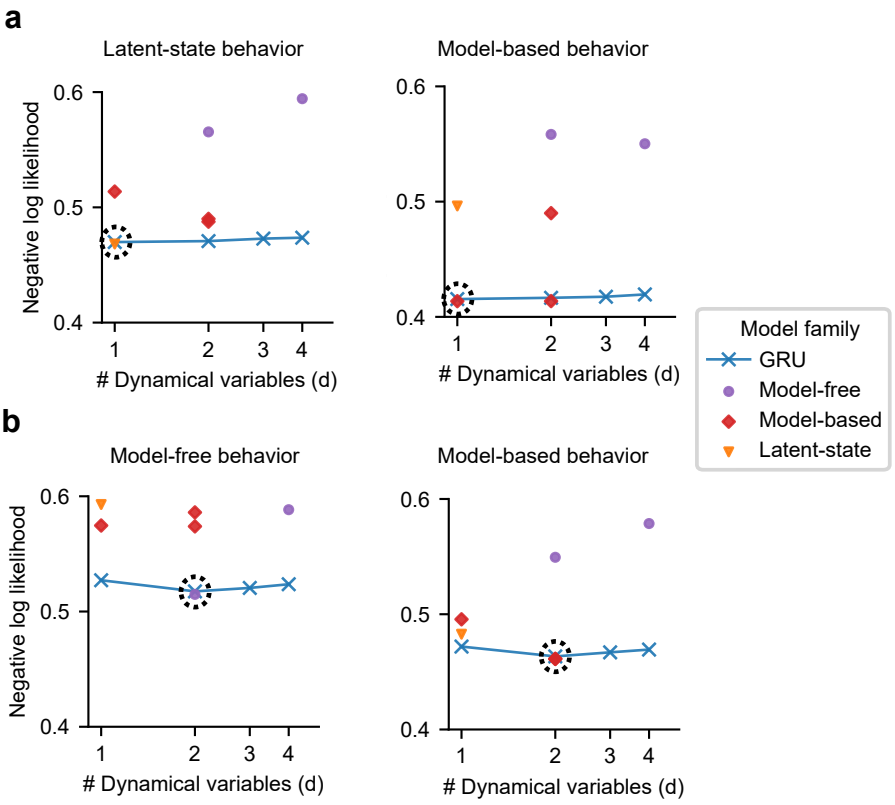

Fig. 4. Tiny RNNs can predict choices as well as the ground-truth model that generated the choices. (a-b) The performance of each model (negative log-likelihood) in predicting the choices generated by a cognitive model in the two-stage task, with choice stochasticity arising due to sampling from the policy. Each panel corresponds to choices generated by a different cognitive model. Dashed circles indicate the predictive performance of the ground-truth cognitive model — i.e., a model fitted to the choice data generated by the same model. Note that the RNNs have the same predictive performance as the ground-truth model, suggesting that it can accurately identify the ground-truth strategy. **(a)** Simulated behavior generated by a latent-state agent with $d = 1$ and a model-based RL agent with $d = 1$. **(b)** Simulated behavior generated by a model-free (Q(1)) agent with $d = 2$ and a model-based RL agent with $d = 2$ (without value forgetting). The reported test performance for each model is the average over 10 outer folds in the nested cross-validation. Each model family may include multiple model variants, leading to two or more identical markers for a given d.

a strong preference for that action (note how the dark blue and dark red lines in Fig. 5b lead to a fixed point, represented by a black cross). In addition to this asymptotic behavior, the straight vs. curved shape of the logit-change plots highlight the distinct ways in which the models process input that disagrees with current beliefs: if an agent selects a strongly disfavored action and receives a reward (e.g., one of the leftmost dark blue dots), a model-free RL agent will experience a large state change (because of a large prediction error), while a latent-state agent will experience only a small state change (because of a strong prior at extreme logit values). The models also behave differently when receiving no reward: while a model-free agent moves towards an indifference state (note the fixed point at $L^* = 0$), the latent-state model moves towards a preference for the unchosen action, suggesting that an unrewarded action is interpreted equivalently to reward for the other action. Indeed, an unrewarded action moves a latent-state the system towards the same fixed point as if a reward had been received there. Together, these examples illustrate how the phase portrait can yield insights into how each model process input.

Crucially, the dynamical systems approach can also be applied to RNNs to provide a detailed comparison with cognitive models. Analyzing a one-unit GRU fitted to data from the reversal learning task (Fig. 5c), we found multiple signatures of a model-free RL strategy. For example, we found that unrewarded trials moved the system towards an indifference state (note attractor at $L = 0$ for light blue and light red lines), and that receiving a reward for a strongly disfavored action resulted in a large change in preferences (note large logit-change for dark blue dots in the left region and for dark red dots in the right region). Additionally, we found no evidence that the animals interpreted an unrewarded action as indicating a reward for the unchosen action (e.g., the light red and dark blue curves overlap in the latent-state model, but not in the model-free RL or GRU models). Since the trained GRUs could have produced either model-free RL and latent-state behavior (Fig. 4), the findings above suggest that the monkey's behavior aligns more closely with that of a model-free RL agent. While these conclusions could have been achieved via direct model comparison (e.g., Fig. 2a), the phase portraits can additionally reveal *how* an animal's behavior relates to one model or the other.

Identifying novel behavioral patterns with one-unit RNNs

The previous results illustrate how the dynamical systems framework can be used to interpret the dynamics of simple cognitive models, and to recognize such patterns in the fitted RNNs. In this section, we illustrate how to identify patterns in the dynamics of one-dimensional RNNs that are absent from cognitive models, and how to interpret such patterns to uncover novel insights about the animal's cognitive processes. Re-examining the phase portrait of the GRU, we note that despite a number of similarities with the portrait of a model-free RL agent, those phase portraits also had visible differences (Fig. 5b,c). For instance, some of the lines in the GRU plot are not straight but curved, suggesting a non-constant learning rate that depends on the animals' internal state (note the curved red lines in Fig. 5c). Additionally, the two lines for $R = 0$ are not overlapping (light red and light blue lines in Fig. 5c). The horizontal decoupling of the $R = 0$ lines is a sign of choice perseveration, but since in the GRU plots they decouple only for extreme values of L, this suggests a peculiar pattern of preference-dependent choice perseveration. Crucially, any insight gathered from the phase portraits can be subsequently verified via a targeted hypothesis testing. Indeed, we use this approach to confirm the existence of a such as state-dependent perseveration (see Supplementary Results and Fig. S4).

Another distinct aspect of the GRU diagram is that the fixed point for $A_1, R = 1$ has a larger magnitude than the fixed point for $A_2, R = 1$ (Fig. 5c). This suggests that the monkey never exhibits a preference for A_2 as high as it does for A_1, akin to a choice bias. To better understand this and other related phenomena, we defined the (normalized) asymptotic preference u_I as the agent's normalized preference for one action over the other after repeatedly receiving input I, i.e., $u_I = L_I^* / \max_I L_I^*$. For example, $|u_I| = 1$ means that input I moves the system towards a state of high maximum preference, and $u_I = 0$ means that that input I moves the system towards a state of zero preference, or indifference. While u_I specifies the asymptotic behavior of the system, we can also view each input I as nudging the agent's current normalized preferences $\hat{Q}(t)$ towards u_I (i.e., a "prediction error" $[u_I - \hat{Q}(t)]$), where $\hat{Q}(t) = L(t) / \max_I L_I^*$. By comparing u_I for different inputs, we identified important differences in each model's behavior. For instance, in line with our earlier observations, we found that unrewarded inputs shifts a model-free RL agent towards a state of indifference ($u_I = 0$), and a latent-state agent towards a state of strong preference for the alternative action ($|u_I| = 1$; compare light blue and light red dots in Fig. 5d). In the fitted GRU, unrewarded inputs lead to a state of indifference as in model-free RL, but rewards following A_2 lead to a lower asymptotic preference than rewards following A_1, a pattern not found in cognitive models (notice that $|u_I| < 1$ for the dark red dot in the rightmost of Fig. 5d).

The asymptotic preference u_I effectively summarizes the effect of inputs on the state of the system. This can facilitate insights from tasks with many inputs by extracting patterns from visually complex phase portraits. To illustrate this, we computed u_I for various models fitted to one rat's behavior (Miller dataset) in the two-stage task. In this task, u_I describes how a model's internal state is affected by each of the eight possible inputs (combinations of two actions, two second-stage states, and two rewards; Fig. 5e). Analyzing u_I for the GRU model, we found an intriguing pattern absent in all cognitive models, a type of "confirmation effect": when the animal experiences a rare transition followed by a reward (i.e., A_1, S_2 and A_2, S_1), the animal moves towards a state of indifference ($|u_I| \approx 0$). This contrasts with the ubiquitous assumption in cognitive models, where a reward always increases one's preference for one action over the other (compare the dark blue

marker for A_1, S_2 and the dark red marker for A_2, S_1 in different plots). We found that this reward-induced indifference was present in either rats (Miller et al.) and mice (Akam et al.), with some other animals showing slightly different patterns in rare transitions (see Fig. S5).

Lastly, we computed asymptotic preferences to analyze the behavior of mice on the transition-reversal two-stage task (see Fig. 5f for an example mouse). We found that the fitted GRUs exhibited patterns of asymptotic preference that generally resembled that of a model-free RL agent, where the presence of reward usually leads to higher asymptotic preference than no reward ($|u_{R=1}| > 0.5 > |u_{R=0}|$). However, the u_I plots also revealed substantial individual differences in the asymptotic preference of mice (see Fig. S6). To examine whether such observations can lead to novel testable predictions, we compared an individual's tendency to gravitate towards indifference after a reward (calculated as $1 - |u_I|$ averaged over four Is when the reward is present; notice that $1 - |u_I|$ is always zero for rewarded trials in the model-free RL model in Fig. 5f) with the individual's performance (the average number of trials in a block before reaching a performance threshold). The strong correlation suggests that mice with a lower asymptotic preference for rewarded actions tend to exhibit poorer task performance ($\rho = 0.62$, $p = 0.008$, Fig. 5g). More broadly, these analyses demonstrate how the dynamics of tiny RNNs can reveal numerous insights that would be difficult with traditional neuroscience methods, as exemplified by the discovery of a state-dependent perseveration and a confirmation effect (reward-induced indifference).

RNNs with two units reveal additional behavioral patterns

Our findings thus far demonstrate the potential of one-unit RNNs to uncover new insights into the cognitive processes underlying decision-making. However, we previously showed that RNNs with 2-4 units perform even better than one-unit RNNs in predicting animal choices, often surpassing more complex models. To interpret such models, one could employ the same logit analyses as before, acknowledging that they do not provide a complete characterization of the dynamics for models with $d > 1$ (see Fig. S7 for an example). Another method is to fit a symbolic regression[39] on the GRU's dynamical variables, which can enable the identification of simple equations describing the dynamics of the system (see Methods). In this section, however, we present an alternative geometric method for interpreting two-dimensional models, using a monkey's behavior on the reversal learning task as a case study.

Our approach for analyzing two-dimensional models consists of examining their vector fields, which provides a complete description of dynamics. In a vector field, axes represent dynamical variables, and arrows indicate state changes during each trial (Fig.6, each panel corresponds to one input type). To illustrate this approach, we generated and compared the vector fields for a fitted two-dimensional model-free RL model, and a fitted two-unit GRU model. We note that these visualizations contain rich information about the dynamics of each model, enabling numerous insights into the cognitive mechanisms underlying choices. For brevity, this section illustrates one such insight out of many possible ones.

To interpret vector fields, one can examine both the direction of arrows, as well as regions they converge to. For example, the vector fields of the fitted model-free RL indicate that receiving a reward after selecting A_i increases $Q(A_i)$, whereas experiencing no reward decreases $Q(A_i)$ (i.e., a pattern of "drift-to-zero"), with no impact on the other Q-value. These dynamics correspond to a line attractor for each input condition (see white crosses in Fig.6a). If the model is augmented with value forgetting, the vector fields instead reveal point attractor dynamics (Fig.S8a). As in any dynamical system, the dynamics around the fixed points can be fully described by the singular value decomposition of linear approximations.

We then analyzed the dynamics of the fitted two-unit RNN model, using the activation of each unit as axes (Fig.6b). We found that these variables learned a quantity analogous to an action value even when not constrained to do so (see Fig.S8b for the constrained case). The network dynamics correspond to an approximate line attractor, but with different attractive states than found in model-free RL model. In rewarded trials (i.e., A_1 $R = 1$ and A_2 $R = 1$), the value of the chosen action gravitates towards a maximum (notice the white crosses near the right and top borders), and the value of the unchosen action decays slightly (notice the arrows pointing slightly downward for A_1 $R = 1$ and leftward for A_2 $R = 1$). In unrewarded trials (i.e., A1 R=0 and A2 R=0), conversely, the value of the chosen action gravitates towards the value of the unchosen action (notice the white crosses near the diagonal line, representing states of indifference). This suggests a rather peculiar drift-to-the-other pattern (see Fig. 6c for a schematic).

We validated this surprising prediction through a targeted analysis comparing two variants of a model-free RL agent, including (i) conventional drift-to-zero rule; and (ii) drift-to-the-other rule. The model-free RL model with the drift-to-the-other rule outperformed the one with the drift-to-zero rule and achieved a predictive performance close to the two-unit GRU model (Fig.S9). Finally, by analyzing phase portrait diagrams of two-unit RNNs (including variants of linear RNNs) in the two-stage and transition-reversal two-stage tasks, we found that these networks were also able to discover strategies strikingly different from typical two-action-value formalizations (Supplementary Results, Fig.S10, S11, S12). These results demonstrate how insights can be obtained from two-unit RNNs, as exemplified by the discovery of a drift-to-the-other pattern.

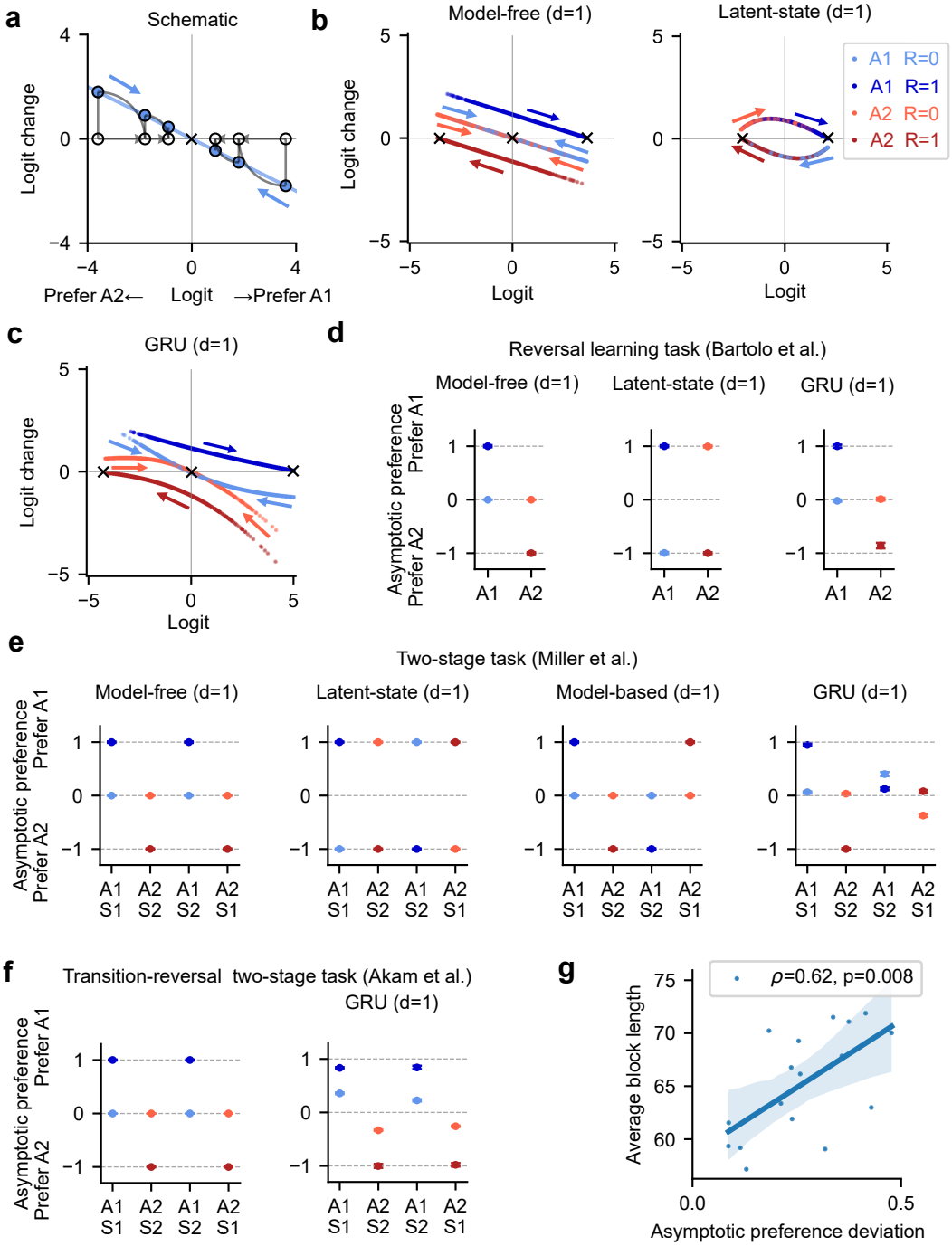

Fig. 5. Leveraging dynamical systems theory for model interpretation and comparison. **(a-c)** Logit analysis, illustrating how an agent's preference for one action over the other changes on each trial (logit change), as a function of the animal's current preferences (logit), of the action taken (A_1, blue; A_2, red), and of the reward received ($R = 0$, light; $R = 1$, dark). **(a)** Schematic showing how the agent's preference (logit) evolves over consecutive trials, represented by empty circles. Starting from a state of a strong preference for A_1 (rightmost empty circle), the input A_1 $R = 0$ causes a negative logit-change (rightmost blue circle), resulting in a reduced preference for A_1 (a leftward shift along the logit axis). Multiple consecutive trials with the same input result in progressively lower preferences for A_1, towards a state of indifference towards either action (black cross at logit=0). Similarly, starting from a state of a strong preference for A_2, and repeatedly experiencing A_1 $R = 0$, shifts the state rightwards towards indifference. **(b)** Logit analysis for two one-dimensional cognitive models fitted to one monkey in the reversal learning task. Each point, colored according to the corresponding input, represents a trial. Colored arrows indicate the flow direction of the model state (logit) after receiving the corresponding input. **(c)** Logit analysis for the one-unit GRU fitted to one monkey in the reversal learning task. **(d-f)** Asymptotic preferences u_I, illustrating the agent's preference for one action over the other after repeatedly experiencing input I. This quantity analogously represents the instantaneous effect of I on the agent's normalized preferences. Colors represent input types, and error bars indicate the standard deviations of asymptotic preference over different model instantiations in nested cross-validation. **(d)** Asymptotic preferences for one-dimensional models fitted to one monkey in the reversal learning task. **(e)** Asymptotic preferences for one-dimensional models fitted to one rat in the two-stage task. **(f)** Asymptotic preferences for one-dimensional models fitted to one mouse in the transition-reversal two-stage task. **(g)** Relationship between an individual's tendency to gravitate towards indifference after a reward (asymptotic preference deviation) and the individual's task performance (average block length, lower is better). Each point is an animal. The shaded region is the 95% confidence interval of the linear regression.

Analyzing behavior in task-optimized neural networks

In this study, our primary focus has been on analyzing neural networks whose parameters were fitted to observable behavior. We note that this approach is distinct from another common use of neural networks in neuroscience, where parameters are adjusted to achieve optimal performance on a specific task. In such cases, researchers compare the behavior and neural representations of task-optimized networks to those of biological organisms, aiming to gain insights into the latter. However, without a rigorous framework guiding these comparisons, the insights generated by this approach are often limited. In this final section, we aim to determine if our interpretative dynamical systems framework can identify emergent algorithms learned in task-optimized RNNs, allowing for a more comprehensive comparison between biological and artificial systems.

To investigate the potential of our approach for analyzing task-optimized networks, we trained an RNN agent within a meta-reinforcement learning (meta-RL) framework to maximize total rewards in a two-stage task (Fig.7a; Fig.S13). Previous work demonstrated that task-optimized meta-RL agents are capable of learning a standalone learning algorithm, akin to how interactions between the prefrontal cortex and striatum might learn network dynamics implementing different algorithms according to task demands[24]. In particular, prior work suggested that task-optimized meta-RL agents use a model-based RL strategy in the two-stage task, based on analyses of "stay probability" and "prediction errors"[24]. To test this hypothesis more directly, a common approach is to use model comparison, asking which of several models is most consistent with the data (e.g., Fig. S14a). However, such approaches cannot rule out other untested mechanisms generating the data.

To gain mechanistic insights into the strategy learned by the meta-RL agent, we compared the phase portrait of the meta-RL agent with the phase portraits of different cognitive models fitted to the agent's behavior. We found that the dynamics of the trained meta-RL agent (Fig. 7b) resembled the dynamics of a one-dimensional latent-state agent (Fig. 7c), not of a one-dimensional model-based RL strategy (Fig. 7d), not of a two-dimensional model-based RL strategy as previously suggested (Fig. 7d)[24], and not of any animals performing the same task (Fig. S5). Our logit analysis further revealed that, during training, the intermediate representations before convergence were substantially different from any cognitive models one could possibly imagine (see Fig. S14b), a situation where traditional model comparisons are non-informative and even strongly misleading. Subtle differences nonetheless existed between the meta-RL agent and the latent-state agent (comparing Fig. 7b-c), with the logit-change lying on several adjacent curves of the logit for a given input condition, deviating slightly from the latent-state's prediction of an exact function of logit. We determined that these adjacent curves directly reflect information about historical input sequences (see Fig. S14c), suggesting that such a "history effect" slightly disrupts the acquisition of a perfect Bayesian inference algorithm. These analyses demonstrate that meta-RL agents learned a strategy akin to Bayesian inference in the two-stage task, markedly different from the strategies employed by animals. Overall, it illustrates how our approach contributes to our understanding of computational processes in both biological and artificial systems.

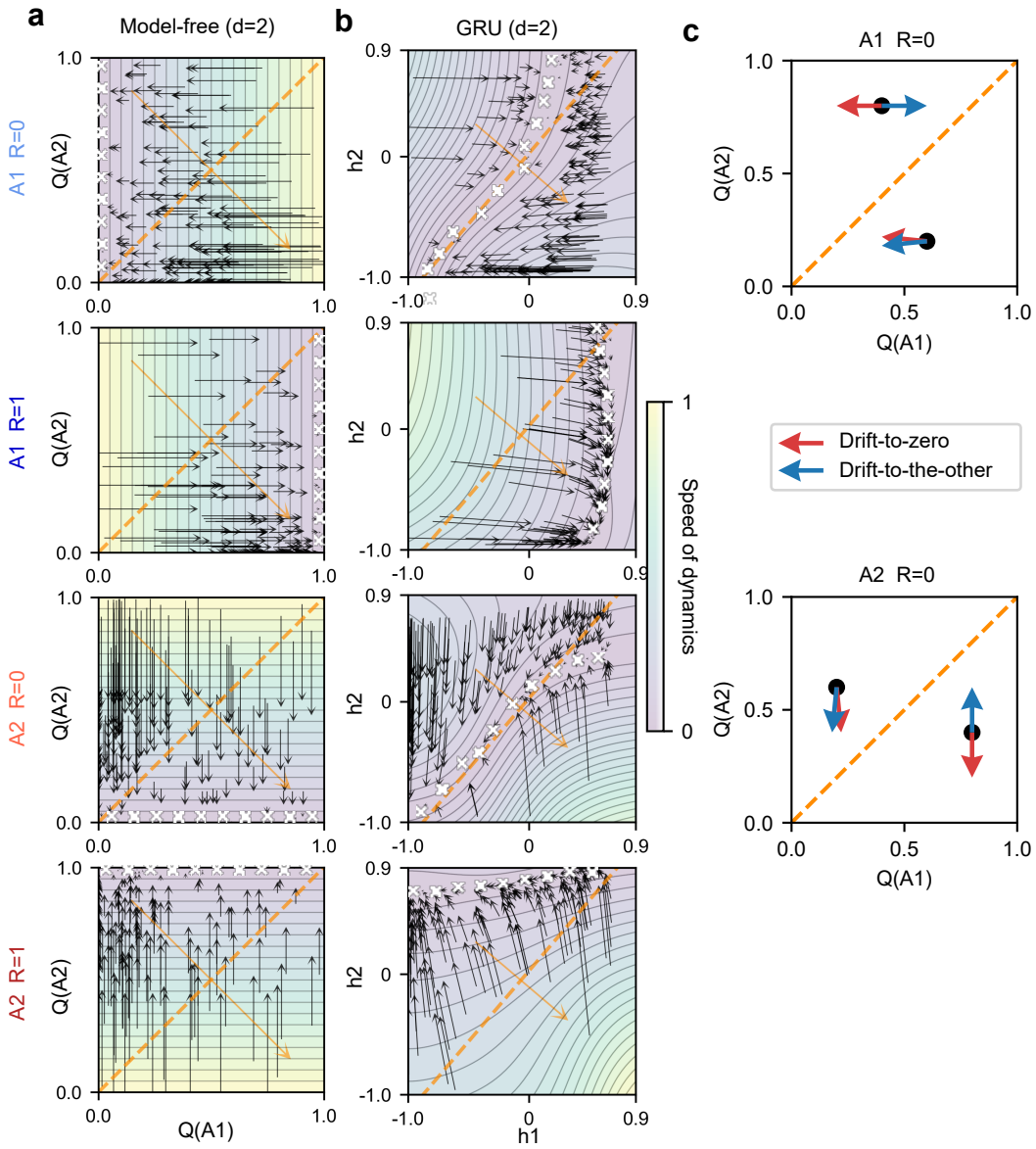

Fig. 6. Interpreting RNNs with two units reveal a novel drift-to-the-other pattern in behavior. **(a-b)** Vector field analysis for two-dimensional models fitted to a monkey's behavior on the reversal learning task. Each panel illustrates the effect of one input on the state variables (axes). Black arrows (flow lines) indicate the state changes in individual trials, with the irregularly distributed endpoints reflecting the actually experienced states. White crosses represent the attractor states, where the model dynamics may settle after repeated trials with the corresponding input. Dashed lines over the diagonal are decision boundaries indicating indifference states. The agent's action preference is read out by the distance between the decision boundary and the orthogonal projection of the model state onto the readout vectors (orange arrows). Background colors indicate the speed of the dynamics, defined as the norm of the velocity vector at each position. **(a)** The model-free RL model. **(b)** The GRU model. **(c)** Schematic contrasting a drift-to-zero and a drift-to-the-other rule. *Top:* If an agent currently prefers A_2 over A_1 (region above diagonal), selects the disfavored action A_1, and receives no reward, the traditional drift-to-zero rule predicts a decrease in $Q(A_1)$ (red arrow pointing leftward), while a drift-to-the-other rule predicts an increase in $Q(A_1)$ (blue arrow pointing rightward). *Bottom:* If an agent currently prefers A_1 over A_2 (region below diagonal), selects the disfavored action A_2, and receives no reward, the traditional drift-to-zero rule predicts a decrease in $Q(A_2)$ (red arrow pointing downward), while a drift-to-the-other rule predicts an increase in $Q(A_2)$ (blue arrow pointing upward).

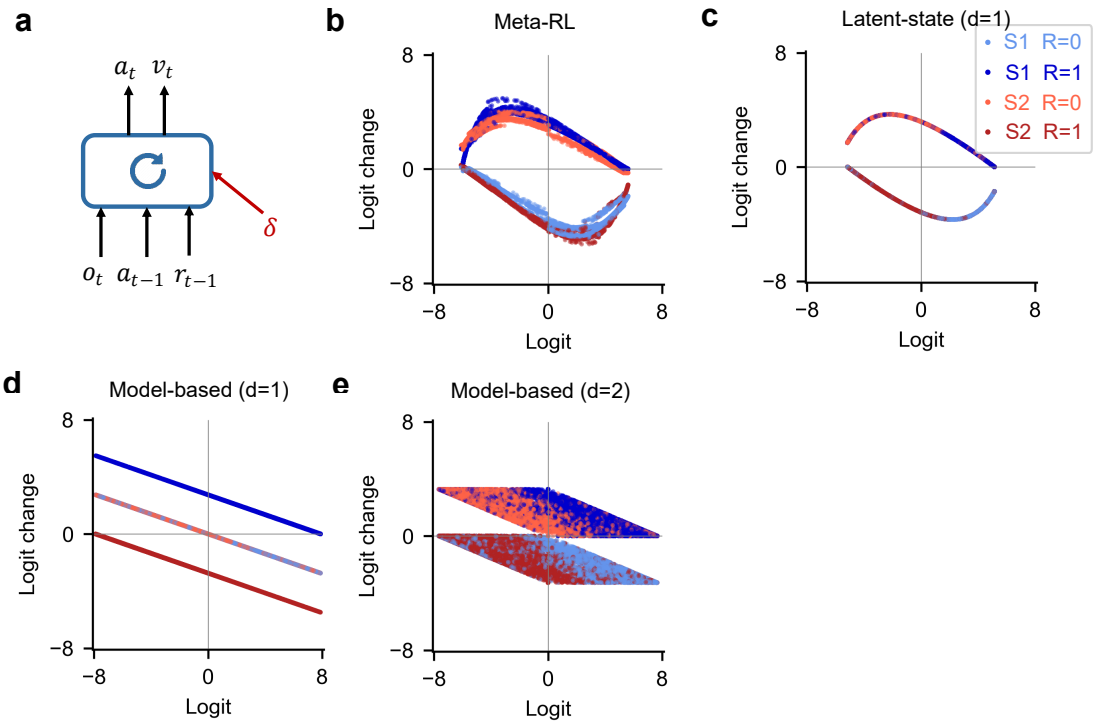

Fig. 7. A meta-reinforcement learning agent learns a strategy similar to the one-dimensional latent-state strategy, different from other RL strategies or any animals' strategies. (a) Architecture of the meta-RL agent, adapted from[24]. The agent receives the observation, the past action, and the past reward as input, and generates a new action and an estimate of the current state value. It adjusts the synaptic weights to optimize the performance in the two-stage task using gradient descent based on the reward prediction error signals δ. **(b-e)** The logit analysis of the meta-RL agent and fitted cognitive models. **(b)** The meta-RL agent. **(c)** The one-dimensional latent-state model fitted to the meta-RL agent's behavior (the task-optimal Bayesian model is included in this latent-state family). **(d)** The one-dimensional model-based RL model fitted to the meta-RL agent's behavior. **(e)** The previously suggested[24] two-dimensional model-based RL model fitted to the meta-RL agent's behavior. The logit-change pattern computed from the output of the meta-RL agent is similar to the logit-change pattern of the one-dimensional latent-state model, strikingly different from the one- or two-dimensional model-based RL models and all animals performing the two-stage task (e.g., strategies in Fig. S5).

Discussion

We introduced a novel modeling approach utilizing tiny RNNs to decipher the computations underpinning animal decision-making. Across three well-studied reward learning tasks performed by animals of three species, we found that RNNs with merely one or two units outperform tested classical cognitive models of equal dimensionality in predicting the behavior of individual animals. Most importantly, we can interpret the trained networks using a dynamical systems framework, allowing us to uncover valuable insights into animals' cognitive strategies during decision-making without extensive handcrafted model comparisons. For instance, our approach demonstrated that animal behavior aligns more closely with model-free RL algorithms than alternative cognitive models. We also identified novel behavioral features not predicted by RL or Bayesian models, such as variable learning rates, a unique confirmation effect, and previously unobserved forms of value drifting, choice perseveration, and choice biases. We argue that these insights would have been difficult to obtain with conventional modeling approaches. Additionally, our method estimates behavior dimensionality and offers mechanistic insights into the emergent algorithms in task-optimized RL agents. Overall, the proposed approach holds promise for unveiling new cognitive strategies in animal decision-making and enhancing our comprehension of the neural mechanisms driving adaptive behavior.

Modeling decision-making with tiny RNNs

Our finding that neural networks outperform classical cognitive models (Fig. 2, 3) in capturing variance in behavior is consistent with previous studies demonstrating the superior performance of feedforward [28] and recurrent neural networks [29,30] over traditional models. The superior performance of neural networks can be attributed to the increased flexibility enabled by the larger number of parameters typically used by these models. The uniqueness of our work lies in our use of RNNs at the smallest scale, which facilitates the interpretation of network dynamics and the identification of novel cognitive strategies. Our tiny networks stand in contrast to the more typical architectures employing tens to hundreds of units [24,26,29,30] (but see small RNNs for knowledge distillation [40]). This is a distinct advantage over analyzing larger models that require simulations with non-representative input sequences [29,41] or dimensionality reduction of dynamics that may have behaviorally irrelevant dimensions [42].

Our approach of fitting tiny RNNs to animal choices may lead to concerns that these models are either too simple or too complex. For some researchers, such as those accustomed to large machine learning models trained on large datasets, an expected concern is that networks with only one or two units are too simple to fully capture the behavior of animals in laboratory tasks. We found that this was not the case in the three studied tasks, where tiny RNNs outperformed all cognitive models of equal dimensionality and all neural network models of greater complexity (Fig. 2). While better fits are typically possible with larger networks, this was not the case in the three tasks studied, possibly due to the simplicity of the behavior expressed by animals on those tasks. We speculate that behavior will be similarly low-dimensional in many other experimental paradigms used in psychology and cognitive science, and that tiny RNNs will be a useful tool for understanding the mechanisms underlying those behaviors. Crucially, tiny RNNs can also be trained to exhibit behavior consistent with either RL and Bayesian inference (Fig. 4), suggesting that they are capable of describing both normative and realistic, suboptimal behavior.

For other researchers, such as those accustomed to simple statistical models having few interpretable parameters, an expected concern is that neural networks are too complex to model the simple behavior expressed by animals in laboratory tasks. Indeed, a typical cognitive model uses a handful of easy-to-interpret parameters, in contrast to 40-80 less transparent parameters in one- or two-unit RNNs. Using concepts from discrete dynamical systems, however, we found that we could interpret trained RNNs, as well as cognitive models, using phase portraits encapsulating and embodying all parameters in either type of model. A related concern is that neural network models require large amounts of training data. We have found this was also not a serious concern, as datasets of standard size in neuroscience appear to be sufficient to take advantage of the network's superior flexibility (Fig. 3a-c). We further showed that, using a knowledge distillation framework, we could achieve impressive fits with very few trials per subject by leveraging data from other subjects, extending the applicability of our approach to human datasets composed of multiple subjects with a few hundred trials each (Fig. 3d-e). These results are consistent with prior studies using group choice data for training large RNNs in reward-based learning tasks [29,30,41].

In summary, our tiny RNNs achieve a balance between simplicity and complexity, combining the flexibility of neural networks with the interpretability of classical cognitive models.

Interpreting models with a dynamical systems approach

The application of discrete dynamical systems theory to cognitive models and RNNs enabled a rich understanding of their inner workings, akin to how continuous dynamical systems theory has been used to explain the coordinated activity of interacting neural populations [43] implementing functions such as motor control [44], working memory [45], perceptual decision-making [23] and value prediction [46]. We have analyzed how the one- or two-dimensional models' internal states (formally,

the logits and the phase portraits of the policy) change over time as a function of reward, observations, and past actions (Fig. 5, 6), which can be directly generalized to models with more dynamical variables. By describing all models using the same framework, we could draw direct comparisons between the strategies learned by the RNNs and those assumed in the cognitive models.

Our results showed that, in all tasks, animals employed RL-like heuristics in updating their internal variables, in contrast with the nonlinear dynamics predicted by the latent-state model using Bayesian inference and by meta-RL agents optimized to perform the same tasks (see Supplementary Discussion). This discrepancy suggests that either animals might not fully grasp the task reward rule or that animals indeed resort to RL as a general approach for solving reward learning tasks. Our results also suggest that animals treated the presence and absence of reward differently in all three tasks (e.g., a confirmation effect in the two-stage task), and there exists a novel pattern of state-dependent choice perseveration in the reversal learning task (see Supplementary Discussion).

Dimensionality of cognitive dynamics

The dimensionality of an agent's behavior can be defined as the number of functions of the agent's past one needs to measure in order to accurately predict its future as well as possible[35,36]. Our analysis of the predictive performance of RNNs with varying sizes revealed that merely 1-4 dynamical variables were adequate for an optimal explanation of behavior in reward learning tasks, suggesting that the behavior of animals in these tasks is low-dimensional (Fig. 2, S2). This observation aligns with other studies describing low-dimensional dynamics in cortical regions[23] and the surprising effectiveness of low-rank RNNs in accounting for intricate neural dynamics[47]. We note that the identification of dynamical variables in experimental data is an ongoing research topic in the field of neuroscience, complex systems, and physics, including efforts to extract a small set of variables from neural data[42,47], from video recordings of physical dynamical systems[48] and to learn effective dynamics for multiscale complex systems[49].

Our results revealed a clear differentiation between the dimensionality of dynamics (i.e., the number of dynamical variables) and their complexity (e.g., the number of fixed points in phase portraits). This dichotomy is also evident in the framework of low-rank RNNs[47,50–52], which distinguishes between the rank of recurrent matrices (dimensionality of dynamics) and the number of neuronal populations (complexity of dynamics). Low-rank RNNs have been successful in characterizing the dimensionality and complexity of various perceptual decision-making tasks[52]. Therefore, it is imperative to extend our framework to more complex, naturalistic cognitive tasks. In the following section, we outline our future directions for expanding our current approach.

Studying human behavior

In this study, we found that monkeys, rats, and mice all exhibited signatures of RL algorithms in their behavior. In the two-stage task, often used to elicit distinguishable model-free and model-based RL behavioral patterns, we found that the behavior of rats and mice was consistent with both RL models in common transitions, and with neither RL models in rare transitions. In contrast, human behavior on the same task is often viewed as a combination of model-free and model-based RL, with different subjects exhibiting different balances of either strategy. Our results raise the alternative possibility that, like rats and mice, human behavior in this task is also characterized by idiosyncratic strategies absent in either RL model, a hypothesis raised previously[15]. This hypothesis could be tested straightforwardly by applying our proposed approach to human data.

More generally, our approach can be used to study human behavior in most reward-learning tasks, with the caveat that such experiments typically involve many subjects, each contributing no more than a few hundred trials. In such settings, the knowledge distillation framework can be used to fit interpretable subject-specific (student) networks to richer training targets provided by a group-trained (teacher) network. Knowledge distillation is an active research area in machine learning[37]. The approach used in our study can be extended in various ways, such as exploring diverse teacher-student architectures, distillation algorithms, and the definitions of knowledge (e.g., internal representations other than the action probability of the teacher network can also serve as the "knowledge" for the student to learn).

By extending our approach to studying human behavior, we speculate that it might also contribute to the development of a formal account of mental illness[53,54]. By improving our understanding of the cognitive dynamics in both healthy and clinical populations, we can achieve a more formal and precise description of the computations leading to behavioral disorders.

Interpreting behavior in more complex tasks

Our study focused on three reward-learning tasks commonly studied in psychology and neuroscience. Future work should test the ability of our models to describe behavior in more complex reward-learning tasks, including tasks with more than 2 actions (e.g., multi-armed bandits) or more than two stages of decisions (e.g., a 3-stage task). Such tasks are likely to require more sophisticated cognitive strategies (e.g., planning), and may also tax our cognitive resources (e.g., working memory).

Our framework is also applicable to tasks other than reward learning, including perceptual or economic decision-making, spatial navigation, and memory tasks, where RNNs can be used to model dependencies between successive actions. While behavior in some of these settings will likely have higher dimensionality than in our studied tasks, we hope that tiny RNNs will remain a valuable tool for identifying the most important cognitive strategies. Our logit analysis, in particular, is readily transferable to models with more than one dynamical variable (e.g., Fig. S7) or more than two actions (e.g., using the one-vs-rest logit definition). Likewise, our two-dimensional vector field analysis can be generalized directly to the visualization of three-dimensional models in three-dimensional spaces.

Eventually, novel techniques will be needed to interpret larger models. For more complex tasks, we suggest that analyzing tiny RNNs with one, two, and gradually increasing numbers of neurons is essential for identifying the correct dimensionality and understanding the additional patterns learned by incorporating each dynamical variable. In models with more than three dynamical variables, one can hold a set of dynamical variables and examine the remaining two or three variables. Alternative approaches for interpreting larger networks include model simulations with pre-designed input sequences[29,41], dimensionality reduction techniques[55], or other methods for analyzing multidimensional nonlinear discrete dynamical systems[38].

Fitting and interpreting neural data

In our study, each model fitted to animal behavior implements a function that maps the animal's past interactions with the environment to a policy. Given that this policy closely tracks the animal's actions, this function can be thought of as approximating an equivalent function implemented by a biological network in the animal's brain. Indeed, latent variables in fitted cognitive models are predictive of neural activity in the prefrontal cortex and striatum, even though these models have not been trained on neural activity[11–13,56]. In light of our results showing a superior predictive performance for the trained RNNs, one might expect that the activity in the network units will not only be predictive of prefrontal and striatal activity, but that it might provide deeper insights than cognitive models into the neural mechanisms of adaptive decisions.

If neural recordings are available, one might additionally consider fitting tiny RNNs to predict neural activity, or to jointly predict behavior (e.g., choices) and neural activity. Similar approaches have been used in latent variable models[57], such as low-rank RNNs[47] and latent linear dynamical systems[58], to study low-dimensional structures in high-dimensional neural population activity. Our approach can be viewed as a bridge between neural activity and behavior.

GRUs and extensions

Our networks utilized Gated Recurrent Units (GRUs), a recurrent architecture whose hidden state is updated by learnable functions of its inputs. This architecture features Markovian hidden variables, in contrast to the more widespread Long Short-Term Memory networks. We also introduced a few switching RNN architectures to accommodate discrete inputs (see Methods). These GRUs, as well as other existing recurrent architectures[59], can be directly applied in contexts with continuous inputs and a larger number of actions.

The updating equation of GRUs limits the complexity of low-dimensional dynamics that these tiny RNNs can capture (see Fig. S15 for the case of one-dimensional dynamics). Although the simplicity of the phase portraits learned (Fig. 5, S5, S6) suggests that a more complex set of variable-updating equations might not explain more variance in behavior with fewer dynamical variables in these three tasks, it is possible that some tasks may require more intricate variable-updating dynamics than tiny GRUs can provide.

To extend the capacity of these tiny RNNs, one possibility is to substitute the one-step dynamics in GRUs with a feedforward architecture (i.e., a multi-layer feedforward network that takes d dynamical variables as inputs and outputs d dynamical variables at the next time step). Another option is the low-rank RNNs[51,52], where the dimensionality of the dynamics is constrained by the rank of recurrent matrices. For non-constant inputs, the dynamics of low-rank RNNs exist in a $(d + N_{\text{in}})$-dimensional space, where N_{in} denotes the number of input dimensions. This is in contrast with our RNN models, in which the dimensionality of hidden dynamics is independent of the number of input dimensions. Further investigation is required to evaluate the advantages and disadvantages of these two options.

Conclusion

In conclusion, we have developed a framework to discover the cognitive algorithms underlying decision-making using low-dimensional RNNs. Our approach not only facilitates model interpretation and selection but also enables the discovery of new aspects of behavior inaccessible by previous models. This work paves the way for more sophisticated analyses of decision-making and other cognitive processes, ultimately contributing to a deeper understanding of the neural mechanisms underlying both adaptive and pathological behavior.

Acknowledgments

We thank the support from Swarma Club and Causal Emergence Reading Group supported by the Save 2050 Programme jointly sponsored by Swarma Club and X-Order. We thank K. T. Jensen, H. Xiong, Z. Jin, and X. Li for their inspirational discussions. This work was supported by the Kavli Institute for Brain and Mind (KIBM) Innovative Research Grant #2022-2209 and in part by National Science Foundation (NSF) awards CNS-1730158, ACI-1540112, ACI-1541349, OAC-1826967, OAC-2112167, CNS-2100237, CNS-2120019, the University of California Office of the President, and the University of California San Diego's California Institute for Telecommunications and Information Technology/Qualcomm Institute. Thanks to CENIC for the 100Gbps networks.

Author Contributions

All authors conceived the study, participated in the discussions, wrote the paper, and acquired funding.

Declaration of Interests

All authors declare no competing interests.

Methods

Tasks and datasets

Reversal learning task

The reversal learning task is a paradigm designed to assess subjects' ability to adapt their behavior in response to changing reward contingencies. In each trial, subjects are presented with two actions, A_1 and A_2, yielding a unit reward with probability p_1^{reward} and p_2^{reward}, respectively. These reward probabilities remain constant for several trials before switching unpredictably and abruptly, without explicit cues. When this occurs, the action associated with the higher reward probability becomes linked to the lower reward probability, and vice versa. The task necessitates continuous exploration of which action currently has a higher reward probability in order to maximize total rewards. For consistency with the other tasks, we assume that actions (A_1 and A_2) are made at the choice state, and A_i deterministically leads to state S_i, where the reward is delivered.

In the Bartolo dataset[11], two monkeys completed a total of 15,500 trials of the reversal learning task with two state-reward types: (1) $p_1^{\text{reward}} = 0.7$ and $p_2^{\text{reward}} = 0.3$; (2) $p_1^{\text{reward}} = 0.3$ and $p_2^{\text{reward}} = 0.7$. Blocks were 80 trials long, and the switch happened at a "reversal trial" between 30 and 50. We predicted the behavior from trials 10 to 70, similar to the original preprocessing procedure[11] because the monkeys were inferring the current block type ("what" block: choosing from two objects; "where" block: choosing from two locations) in the first few trials.

In the Akam dataset[12], ten mice completed a total of 67,009 trials of the reversal learning task with three state-reward types: (1) $p_1^{\text{reward}} = 0.75$ and $p_2^{\text{reward}} = 0.25$; (2) $p_1^{\text{reward}} = 0.25$ and $p_2^{\text{reward}} = 0.75$; (3) $p_1^{\text{reward}} = 0.5$ and $p_2^{\text{reward}} = 0.5$ (neutral trials). Block transitions from non-neutral blocks were triggered 10 trials after an exponential moving average (tau = 8 trials) crossed a 75% correct threshold. Block transitions from neutral blocks occurred with a probability 10% on each trial after the 15th of the block to give an average neutral block length of 25 trials.

Two-stage task

The two-stage task is a paradigm commonly used to distinguish between the influences of model-free and model-based reinforcement learning on animal behavior[32,56]. In each trial, subjects are presented with two actions, A_1 and A_2, while at the choice state. Action A_1 leads with a high probability to state S_1 and a low probability to state S_2, while action A_2 leads with a high probability to state S_2 and a low probability to state S_1. From second-stage states S_1 and S_2, the animal can execute an action for a chance of receiving a unit reward. Second-stage states are distinguishable by visual cues and have different probabilities of yielding a unit reward: p_1^{reward} for S_1 and p_2^{reward} for S_2. These reward probabilities remain constant for several trials before switching unpredictably and abruptly. When this occurs, the second-stage state associated with the higher reward probability becomes linked to the lower reward probability, and vice versa.

In the Miller dataset[13], four rats completed a total of 33,957 trials of the two-stage task with two state-reward types: (1) $p_1^{\text{reward}} = 0.8$ and $p_2^{\text{reward}} = 0.2$; (2) $p_1^{\text{reward}} = 0.2$ and $p_2^{\text{reward}} = 0.8$. Block switches occurred with a 2% probability on each trial after a minimum block length of 10 trials.

In the Akam dataset[12], ten mice completed a total of 133,974 trials of the two-stage task with three state-reward types: (1) $p_1^{\text{reward}} = 0.8$ and $p_2^{\text{reward}} = 0.2$; (2) $p_1^{\text{reward}} = 0.2$ and $p_2^{\text{reward}} = 0.8$; (3) $p_1^{\text{reward}} = 0.4$ and $p_2^{\text{reward}} = 0.4$ (neutral trials). Block transitions occur 20 trials after an exponential moving average (tau = 8 trials) of subject's choices crossed a 75% correct threshold. In neutral blocks, block transitions occurred with 10% probability on each trial after the 40th. Transitions from non-neutral blocks occurred with equal probability either to another non-neutral block or to the neutral block. Transitions from neutral blocks occurred with equal probability to one of the non-neutral blocks.

Transition-reversal two-stage task

The transition-reversal two-stage task is a modified version of the original two-stage task, with the introduction of occasional reversals in action-state transition probabilities[12]. This modification was proposed to facilitate the dissociation of state prediction and reward prediction in neural activity and to prevent habit-like strategies that may produce model-based control-like behavior without forward planning. In each trial, subjects are presented with two actions, A_1 and A_2, at the choice state. One action commonly leads to state S_1 and rarely to state S_2, while the other action commonly leads to state S_2 and rarely to state S_1. These action-state transition probabilities remain constant for several trials before switching unpredictably and abruptly, without explicit cues. In second-stage states S_1 and S_2, subjects execute an action for a chance of receiving a unit reward. The second-stage states are visually distinguishable and have different reward probabilities that also switch unpredictably and abruptly, without explicit cues, similar to the other two tasks.

In the Akam dataset[12], seventeen mice completed a total of 230,237 trials of the transition-reversal two-stage task with two action-state types: (1) $\Pr(S_1|A_1) = \Pr(S_2|A_2) = 0.8$ and $\Pr(S_2|A_1) = \Pr(S_1|A_2) = 0.2$; (2)$\Pr(S_1|A_1) =$

$\Pr(S_2|A_2) = 0.2$ and $\Pr(S_2|A_1) = \Pr(S_1|A_2) = 0.8$. There were also three state-reward types: (1) $p_1^{\text{reward}} = 0.8$ and $p_2^{\text{reward}} = 0.2$; (2) $p_1^{\text{reward}} = 0.2$ and $p_2^{\text{reward}} = 0.8$; (3) $p_1^{\text{reward}} = 0.4$ and $p_2^{\text{reward}} = 0.4$ (neutral trials). Block transitions occur 20 trials after an exponential moving average (tau = 8 trials) of subject's choices crossed a 75% correct threshold. In neutral blocks, block transitions occurred with 10% probability on each trial after the 40th. Transitions from non-neutral blocks occurred with equal probability (25%) either to another non-neutral block via reversal in the reward or transition probabilities, or to one of the two neutral blocks. Transitions from neutral blocks occurred via a change in the reward probabilities only to one of the non-neutral blocks with the same transition probabilities.

Recurrent neural networks

Network architectures

We investigated several architectures, as described below. Our primary goal is to capture the maximum possible behavioral variance with d dynamical variables. While we generally prefer more flexible models due to their reduced bias, such models typically require more data for training, and insufficient data can result in underfitting and poorer performance in comparison to less flexible (simpler) models. Therefore, we aimed to balance data efficiency and model capacity through cross-validation.

After finding the best performing model class, we performed an investigation of the network properties that contributed the most to the successfully explained variance. Analogous to ablation studies, our approach consisted of gradually removing components or adding constraints to the architectures, such as eliminating nonlinearity or introducing symmetric weight constraints. The unaffected predictive performance suggests that the examined components are not essential for the successfully explained variance. If affected, this indicates that these components can contribute to explaining additional behavioral patterns. Following this approach, we can establish connections between architectural components and their corresponding underlying behavioral patterns. The primary objective of this approach is to capture maximum variance with minimal components in the models, resulting in highly interpretable models.

Recurrent layer The neural network models in this paper used vanilla gated recurrent units (GRU) in their hidden layers[33]. The hidden state h_t at the beginning of trial t consists of d elements (dynamical variables). The initial hidden state h_1 is set to 0 and h_t ($t > 1$) is updated as follows:

$$\begin{aligned} r_t &= \sigma(W_{ir}x_{t-1} + b_{ir} + W_{hr}h_{t-1} + b_{hr}) \\ z_t &= \sigma(W_{iz}x_{t-1} + b_{iz} + W_{hz}h_{t-1} + b_{hz}) \\ n_t &= \tanh(W_{in}x_{t-1} + b_{in} + r_t \odot (W_{hn}h_{t-1} + b_{hn})) \\ h_t &= (1 - z_t) \odot n_t + z_t \odot h_{t-1} \end{aligned} \tag{1}$$

where σ is the sigmoid function, $\odot$ is the Hadamard (element-wise) product, x_{t-1} and h_{t-1} are the input and hidden state from the last trial $t-1$, and r_t, z_t, n_t are the reset, update, new gates (intermediate variables) at trial t, respectively. The weight matrices $W_{..}$ and biases $b_{..}$ are trainable parameters. The hidden state of the network, h_t, represents a summary of past inputs and is the only information used to generate outputs.

To accommodate discrete inputs, we also introduce a modified architecture called switching GRU, where recurrent weights and biases are input-dependent, similar to discrete-latent-variable-dependent switching linear dynamical systems[60]. In this architecture, the hidden state h_t ($t > 1$) is updated as follows:

$$\begin{aligned} r_t &= \sigma(b_{ir}^{(x_{t-1})} + W_{hr}^{(x_{t-1})}h_{t-1} + b_{hr}^{(x_{t-1})}) \\ z_t &= \sigma(b_{iz}^{(x_{t-1})} + W_{hz}^{(x_{t-1})}h_{t-1} + b_{hz}^{(x_{t-1})}) \\ n_t &= \tanh(b_{in}^{(x_{t-1})} + r_t \odot (W_{hn}^{(x_{t-1})}h_{t-1} + b_{hn}^{(x_{t-1})})) \\ h_t &= (1 - z_t) \odot n_t + z_t \odot h_{t-1} \end{aligned} \tag{2}$$

where $W_{h\cdot}^{(x_{t-1})}$ and $b_{\cdot\cdot}^{(x_{t-1})}$ are the weight matrices and biases selected by the input x_{t-1} (i.e., each input x_{t-1} induces an independent set of weights $W_{h\cdot}$ and biases $b_{..}$).

Because the inputs in the three tasks we studied are discrete, switching GRUs are a generalization of vanilla GRUs (i.e., a vanilla GRU can be viewed as a switching GRU whose recurrent weights do not vary with the input). Generalizations of switching GRUs from discrete to continuous inputs are closely related to multiplicative integration GRUs[61].

Across three tasks and five datasets, we found that the switching GRU models performed similarly to the vanilla GRU models for $d \geq 2$, but consistently outperformed the vanilla GRU models for $d = 1$. Therefore, for the results in the main text, we reported the performance of the switching GRU models for $d = 1$ and the performance of the vanilla GRU

models for $d \geq 2$. Mathematically, these vanilla GRU models can be directly transformed into corresponding switching GRU models:

$$\begin{aligned} b_{i.}^{(x_{t-1})} &\longleftarrow W_{i.}x_{t-1} + b_{i.} \\ b_{h.}^{(x_{t-1})} &\longleftarrow b_{h.} \\ W_{h.}^{(x_{t-1})} &\longleftarrow W_{h.} \end{aligned} \tag{3}$$

We also proposed the switching linear neural networks (SLIN), where the hidden state h_t ($t > 1$) is updated as follows:

$$h_t = W^{(x_{t-1})}h_{t-1} + b^{(x_{t-1})} \tag{4}$$

where $W^{(x_{t-1})}$ and $b^{(x_{t-1})}$ are the weight matrices and biases selected by the input x_{t-1}. In some variants, we constrained $W^{(x_{t-1})}$ to be symmetric.

Input layer The network's input x_t consists of the previous action a_{t-1}, the previous second-stage state s_{t-1}, and the previous reward r_{t-1} (but $a_t = s_t$ in the reversal learning task). In the vanilla GRU networks, the input x_t is three-dimensional and projects with linear weights to the recurrent layer. In the switching GRU networks, the input x_t is used as a selector variable where the network's recurrent weights and biases depend on the network's inputs. Thus, switching GRUs trained on the reversal learning task have four sets of recurrent weights and biases corresponding to all combinations of a_{t-1} and r_{t-1}, and switching GRUs trained on the two-stage and transition-reversal two-stage tasks have eight sets of recurrent weights and biases corresponding to all combinations of a_{t-1}, s_{t-1}, and r_{t-1}.

Output layer The network's output consists of two units whose activities are linear functions of the hidden state h_t. A softmax function (a generalization of the logistic function) is used to convert these activities into a probability distribution (a policy). In the first trial, the network's output is read out from the initial hidden state h_1, which has not yet been updated on the basis of any input. For two-unit networks, the network's output was computed either as a linear function of the hidden state (i.e., $o_t^{(i)} = \beta_{1,i} \cdot h_t^{(1)} + \beta_{2,i} \cdot h_t^{(2)}, i = 1, 2$) or as a direct readout of the hidden state (i.e., $o_t^{(i)} = \beta \cdot h_t^{(i)}, i = 1, 2$).

Network training

Networks were trained using the Adam optimizer (learning rate of 0.005) on batched training data with cross-entropy loss, recurrent weight L1-regularization loss (coefficient drawn between 1e-5 and 1e-1, depending on experiments), and early-stop (if the validation loss does not improve for 200 iteration steps). All networks were implemented with PyTorch.

Classical cognitive models

Models for the reversal learning task

In this task, we implemented one model from the latent-state family and five models from the model-free family (adopted from[32] and[13], or constructed from GRU phase portraits).

Latent-state strategy (d=1). This model assumes the existence of the latent state h, with $h = i$ representing a higher reward probability following action A_i (state S_i). The probability $\Pr_t(h = 1)$, as the dynamical variable, is first updated via Bayesian inference:

$$\hat{\Pr}_t(h = 1) = \frac{\Pr(r_{t-1}|h = 1, s_{t-1}) \Pr_{t-1}(h = 1)}{\Pr(r_{t-1}|h = 1, s_{t-1}) \Pr_{t-1}(h = 1) + \Pr(r_{t-1}|h = 2, s_{t-1}) \Pr_{t-1}(h = 2)}, \tag{5}$$

where the left-hand side is the posterior probability (we omit the conditions for simplicity). The agent also incorporates the knowledge that, in each trial, the latent-state h can switch (e.g., from $h = 1$ to $h = 2$) with a small probability p_r. Thus the probability $\Pr_t(h)$ reads,

$$\Pr_t(h = 1) = (1 - p_r)\hat{\Pr}_t(h = 1) + p_r(1 - \hat{\Pr}_t(h = 1)). \tag{6}$$

The action probability is then derived from softmax$(\beta \Pr_t(h = 1), \beta \Pr_t(h = 2))$ with inverse temperature β ($\beta \geq 0$).

Model-free strategy (d=1). This model hypothesizes that the two action values $Q_t(A_i)$ are fully anti-correlated ($Q_t(A_1) = -Q_t(A_2)$) as follows:

$$\begin{aligned} Q_t(a_{t-1}) &= Q_{t-1}(a_{t-1}) + \alpha(r_{t-1} - Q_{t-1}(a_{t-1})) \\ Q_t(\bar{a}_{t-1}) &= Q_{t-1}(\bar{a}_{t-1}) - \alpha(r_{t-1} + Q_{t-1}(\bar{a}_{t-1})), \end{aligned} \tag{7}$$

where $\bar{a}_{t-1}$ is the unchosen action, and α is the learning rate ($0 \leq \alpha \leq 1$). We specify the $Q_t(A_1)$ as the dynamical variable.

Model-free strategy (d=2). This model hypothesizes that the two action values $Q_t(A_i)$, as two dynamical variables, are updated independently:

$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \alpha(r_{t-1} - Q_{t-1}(a_{t-1})). \tag{8}$$

The unchosen action value $Q_t(\bar{a}_{t-1})$ is unaffected.

Model-free strategy with value forgetting (d=2). The chosen action value is updated as in the previous model. The unchosen action value $Q_t(\bar{a}_{t-1})$, instead, is gradually forgotten:

$$Q_t(\bar{a}_{t-1}) = DQ_{t-1}(\bar{a}_{t-1}), \tag{9}$$

where D is the value forgetting rate ($0 \leq D \leq 1$).

Model-free strategy with the drift-to-the-other rule (d=2). This strategy is constructed from the phase diagram of the two-unit GRU. When there is a reward, the chosen action value is updated as follows,

$$Q_t(a_{t-1}) = D_1 Q_{t-1}(a_{t-1}) + 1, \tag{10}$$

where D_1 is the value drifting rate ($0 \leq D_1 \leq 1$). The unchosen action value is slightly decreased:

$$Q_t(\bar{a}_{t-1}) = Q_{t-1}(\bar{a}_{t-1}) - b, \tag{11}$$

where b is the decaying bias ($0 \leq b \leq 1$, usually small). When there is no reward, the unchosen action value is unchanged, and the chosen action value drifts to the other:

$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \alpha_0(Q_{t-1}(\bar{a}_{t-1}) - Q_{t-1}(a_{t-1})), \tag{12}$$

where α_0 is the drifting rate ($0 \leq \alpha_0 \leq 1$).

For all model-free RL models with $d = 2$, the action probability is determined by $\text{softmax}(\beta Q_t(A_1), \beta Q_t(A_2))$.

Model-free reward-as-cue strategy (d=8). This model assumes that the animal considers the combination of the second-stage state s_{t-1} and the reward r_{t-1} from the trial $t-1$ as the augmented state $\mathcal{S}_t$ for trial t. The eight dynamical variables are the values for the two actions at the four augmented states. The action values are updated as follows:

$$Q_t(\mathcal{S}_{t-1}, a_{t-1}) = Q_{t-1}(\mathcal{S}_{t-1}, a_{t-1}) + \alpha(r_{t-1} - Q_{t-1}(\mathcal{S}_{t-1}, a_{t-1})). \tag{13}$$

The action probability at trial t is determined by $\text{softmax}(\beta Q_t(\mathcal{S}_t, A_1), \beta Q_t(\mathcal{S}_t, A_2))$.

Models for the two-stage task

We implemented one model from the latent-state family, four models from the model-free family, and three from the model-based family (adopted from[32] and[13]).

Latent-state strategy (d=1). Same as latent-state strategy (d=1) in the reversal learning task, except that $h = i$ represents a higher reward probability following state S_i (not action A_i).

Model-free strategy (d=1). Same as the model-free strategy (d=1) in the reversal learning task by ignoring the second-stage states s_{t-1}.

Model-free Q(1) strategy (d=2). Same as the model-free strategy (d=2) in the reversal learning task by ignoring the second-stage states s_{t-1}.

Model-free Q(0) strategy (d=4). This model first updates the first-stage action values $Q_t(a_{t-1})$ with the second-stage state values $V_{t-1}(s_{t-1})$:

$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \alpha(V_{t-1}(s_{t-1}) - Q_{t-1}(a_{t-1})), \tag{14}$$

while the unchosen action value $Q_t(\bar{a}_{t-1})$ is unaffected. Then the second-stage state value $V_t(s_{t-1})$ is updated by the observed reward:

$$V_t(s_{t-1}) = V_{t-1}(s_{t-1}) + \alpha(r_{t-1} - V_{t-1}(s_{t-1})). \tag{15}$$

The four dynamical variables are the two action values and two state values.

Model-free reward-as-cue strategy (d=8). Same as model-free reward-as-cue strategy (d=8) in the reversal learning task.

Model-based strategy (d=1). In this model, the two state values $V_t(S_i)$ are fully anti-correlated ($V_t(S_1) = -V_t(S_2)$):

$$\begin{aligned} V_t(s_{t-1}) &= V_{t-1}(s_{t-1}) + \alpha(r_{t-1} - V_{t-1}(s_{t-1})) \\ V_t(\bar{s}_{t-1}) &= V_{t-1}(\bar{s}_{t-1}) - \alpha(r_{t-1} + V_{t-1}(\bar{s}_{t-1})), \end{aligned} \tag{16}$$

where $\bar{s}_{t-1}$ is the unvisited state. The dynamical variable is the state value $V_t(S_1)$.

Model-based strategy (d=2). The visited state value is updated:

$$V_t(s_{t-1}) = V_{t-1}(s_{t-1}) + \alpha(r_{t-1} - V_{t-1}(s_{t-1})). \tag{17}$$

The unvisited state value is unchanged. The two dynamical variables are the two state values.

Model-based strategy with value forgetting (d=2). The visited state value is updated as in the previous model. The unvisited state value is gradually forgotten:

$$V_t(\bar{s}_{t-1}) = DV_{t-1}(\bar{s}_{t-1}), \tag{18}$$

where D is the value forgetting rate ($0 \leq D \leq 1$).

For all model-based RL models, the action values at the first stage are directly computed using the state transition model:

$$Q_t(A_i) = \sum_j \Pr(S_j|A_i)V_t(S_j), \tag{19}$$

where $\Pr(S_j|A_i)$ is known. The action probability is determined by softmax$(\beta Q_t(A_1), \beta Q_t(A_2))$.

Models for the transition-reversal two-stage task

For this task, we further include cognitive models proposed in [12]. We first describe different model components (ingredients) and corresponding numbers of dynamical variables, and then specify the components employed in each model.

Second-stage state value component. The visited state value is updated:

$$V_t(s_{t-1}) = V_{t-1}(s_{t-1}) + \alpha_Q(r_{t-1} - V_{t-1}(s_{t-1})). \tag{20}$$

The unvisited state value $V_t(\bar{s}_{t-1})$ is either unchanged or gradually forgotten with f_Q as the value forgetting rate. This component requires two dynamical variables.

Model-free action value component. The first-stage action values $Q_t^{mf}(a_{t-1})$ are updated by the second-stage state values $V_{t-1}(s_{t-1})$ and the observed reward:

$$Q_t^{mf}(a_{t-1}) = Q_{t-1}^{mf}(a_{t-1}) + \alpha(\lambda r_{t-1} + (1-\lambda)V_{t-1}(s_{t-1}) - Q_{t-1}^{mf}(a_{t-1})), \tag{21}$$

where λ is the eligibility trace. The unchosen action value $Q_t^{mf}(\bar{a}_{t-1})$ is unaffected or gradually forgotten with f_Q as the value forgetting rate. This component requires two dynamical variables.

Model-based component. The action-state transition probabilities are updated as:

$$\begin{aligned} P_t(s_{t-1}|a_{t-1}) &= P_{t-1}(s_{t-1}|a_{t-1}) + \alpha_T(1 - P_{t-1}(s_{t-1}|a_{t-1})) \\ P_t(\bar{s}_{t-1}|a_{t-1}) &= P_{t-1}(\bar{s}_{t-1}|a_{t-1}) + \alpha_T(0 - P_{t-1}(\bar{s}_{t-1}|a_{t-1})), \end{aligned} \tag{22}$$

where α_T is the transition probability learning rate. For the unchosen action, the action-state transition probabilities are either unchanged or forgotten:

$$\begin{aligned} P_t(s_{t-1}|\bar{a}_{t-1}) &= P_{t-1}(s_{t-1}|\bar{a}_{t-1}) + f_T(0.5 - P_{t-1}(s_{t-1}|\bar{a}_{t-1})) \\ P_t(\bar{s}_{t-1}|\bar{a}_{t-1}) &= P_{t-1}(\bar{s}_{t-1}|\bar{a}_{t-1}) + f_T(0.5 - P_{t-1}(\bar{s}_{t-1}|\bar{a}_{t-1})), \end{aligned} \tag{23}$$

where f_T is the transition probability forgetting rate.

The model-based action values at the first stage are directly computed using the learned state transition model:

$$Q_t^{mb}(A_i) = \sum_j P_t(S_j|A_i)V_t(S_j). \tag{24}$$

This component requires two dynamical variables ($P_t(S_1|A_1)$ and $P_t(S_1|A_2)$), since other variables can be directly inferred.

Motor-level model-free action component. Due to the apparatus design in this task[12], it is proposed that the mice consider the motor-level actions a_{t-1}^{mo}, defined as the combination of the last-trial action a_{t-1} and the second-stage state s_{t-2} before it. The motor-level action values $Q_t^{mo}(a_{t-1}^{mo})$ are updated as:

$$Q_t^{mo}(a_{t-1}^{mo}) = Q_{t-1}^{mo}(a_{t-1}^{mo}) + \alpha(\lambda r_{t-1} + (1-\lambda)V_{t-1}(s_{t-2}) - Q_{t-1}^{mo}(a_{t-1}^{mo})), \tag{25}$$

where λ is the eligibility trace. The unchosen motor-level action value Q_t^{mo} is unaffected or gradually forgotten with f_Q as the value forgetting rate. This component requires four dynamical variables (four motor-level actions).

Choice perseveration component. The single-trial perseveration $\tilde{X}_{t-1}^{cp}$ is set to -0.5 for $a_{t-1} = A_1$ and 0.5 for $a_{t-1} = A_2$. The multi-trial perseveration Q_{t-1}^{cp} (exponential moving average of choices) is updated as:

$$X_t^{cp} = X_{t-1}^{cp} + \alpha_c(\tilde{X}_{t-1}^{cp} - X_{t-1}^{cp}), \tag{26}$$

where α_c is the choice perseveration learning rate. In some models, the α_c is less than 1, so one dynamical variable is required; while in some other models, the α_c is fixed to 1, suggesting that it is reduced to the single-trial perseveration and no dynamical variable is required.

Motor-level choice perseveration component. The multi-trial motor-level perseveration $X_{t-1}^{mocp}(s_{t-2})$ is updated as:

$$X_t^{mocp}(s_{t-2}) = X_{t-1}^{mocp}(s_{t-2}) + \alpha_m(\tilde{X}_{t-1}^{cp} - X_{t-1}^{mocp}(s_{t-2})), \tag{27}$$

where α_m is the motor-level choice perseveration learning rate. This component requires two dynamical variables.

Action selection component. The net action values are computed as follows:

$$Q_t^{net}(A_i) = G^{mf}Q_t^{mf}(A_i) + G^{mo}Q_t^{mo}(A_i, s_{t-1}) + G^{mb}Q_t^{mb}(A_i) + X_t(A_i), \tag{28}$$

where G^{mf}, G^{mo}, G^{mb} are model-free, motor-level model-free, model-based inverse temperatures, and $X_t(A_i)$ is:

$$\begin{aligned} X_t(A_1) &= 0 \\ X_t(A_2) &= B_c + B_r\tilde{X}_{t-1}^{s} + P_cX_t^{cp} + P_mX_t^{mocp}(s_{t-1}), \end{aligned} \tag{29}$$

where B_c (bias), B_r (rotation bias), P_c, P_m are weights controlling each component, and $\tilde{X}_{t-1}^{s}$ is -0.5 for $s_{t-1} = S_1$ and 0.5 for $s_{t-1} = S_2$.

The action probabilities are generated via softmax$(Q_t^{net}(A_1), Q_t^{net}(A_2))$.

Model-free strategies. We include five model-free RL models:

- the model-free strategy (d=1) same as the two-stage task;
- the model-free Q(1) strategy (d=2) same as the two-stage task;
- state value [2] + model-free action value [2] + bias [0] + rotation bias [0] + single-trial choice perseveration [0];
- state value [2] + model-free action value with forgetting [2] + bias [0] + rotation bias [0] + single-trial choice perseveration [0];
- state value [2] + model-free action value with forgetting [2] + motor-level model-free action value with forgetting [4] + bias [0] + rotation bias [0] + multi-trial choice perseveration [1] + multi-trial motor-level choice perseveration [2].

Here, we use the format of "model component [required number of dynamical variables]" (more details in [12]).

Model-based strategies. We include twelve model-based RL models:

- state value [2] + model-based [2] + bias [0] + rotation bias [0] + single-trial choice perseveration [0];
- state value [2] + model-free action value [2] + model-based [2] + bias [0] + rotation bias [0] + single-trial choice perseveration [0];
- state value [2] + model-based with forgetting [2] + bias [0] + rotation bias [0] + single-trial choice perseveration [0];
- state value [2] + model-free action value with forgetting [2] + model-based with forgetting [2] + bias [0] + rotation bias [0] + single-trial choice perseveration [0];
- state value [2] + model-free action value with forgetting [2] + model-based [2] + bias [0] + rotation bias [0] + single-trial choice perseveration [0];
- state value [2] + model-free action value [2] + model-based [2] + bias [0] + rotation bias [0] + multi-trial choice perseveration [1];
- state value [2] + model-free action value with forgetting [2] + model-based with forgetting [2] + bias [0] + rotation bias [0] + multi-trial choice perseveration [1];
- state value [2] + model-free action value with forgetting [2] + model-based [2] + bias [0] + rotation bias [0] + multi-trial choice perseveration [1];
- state value [2] + model-free action value with forgetting [2] + model-based with forgetting [2] + bias [0] + rotation bias [0] + multi-trial motor-level choice perseveration [2];
- state value [2] + model-based with forgetting [2] + bias [0] + rotation bias [0] + multi-trial choice perseveration [1] + multi-trial motor-level choice perseveration [2];
- state value [2] + model-free action value with forgetting [2] + model-based with forgetting [2] + bias [0] + rotation bias [0] + multi-trial choice perseveration [1] + multi-trial motor-level choice perseveration [2];
- state value [2] + model-free action value with forgetting [2] + model-based with forgetting [2] + motor-level model-free action value with forgetting [4] + bias [0] + rotation bias [0] + multi-trial choice perseveration [1] + multi-trial motor-level choice perseveration [2].

Here, we use the format of model component [required number of dynamical variables] (more details in [12]).

Model fitting

Maximum likelihood estimation

The parameters in all models were optimized on the training dataset to maximize the log-likelihood (i.e., minimize the negative log-likelihood, or cross-entropy) for the next-action prediction. The loss function is defined as follows:

$$\begin{aligned}\mathcal{L} &= -\log \Pr[\text{action sequences from one subject given one model}] \\ &= -\sum_{n=1}^{N_{\text{session}}} \sum_{t=1}^{T_n} \log \Pr[\text{observing } a_t \text{ given past observations and the model}],\end{aligned} \tag{30}$$

where N_{session} is the number of sessions and T_n is the number of trials in session n.

Nested cross-validation

To avoid overfitting and ensure a fair comparison between models with varying numbers of parameters, we implemented nested cross-validation. For each animal, we first divided sessions into non-overlapping shorter blocks (approximately 150 trials per block) and allocated these blocks into ten folds. In the outer loop, nine folds were designated for training and validation, while the remaining fold was reserved for testing. In the inner loop, eight of the nine folds were assigned for training (optimizing a model's parameters for a given set of hyperparameters), and the remaining fold of the nine was allocated for validation (selecting the best-performing model across all hyperparameter sets). Notice that this procedure allows different hyperparameters for each test set.

RNNs' hyperparameters encompassed the L1-regularization coefficient on recurrent weights (drawn from 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, depending on the experiments), the number of training epochs (i.e., early stopping), and the random seed (three seeds). For cognitive models, the only hyperparameter was the random seed (used for parameter initialization). The inner loop produced nine models, with the best-performing model, based on average performance in the training and validation datasets, being selected and evaluated on the unseen testing fold. The final testing performance was computed as the average across all ten testing folds, weighted by the number of trials per block. This approach ensures that test data is exclusively used for evaluation and is never encountered during training or selection.

During RNN training, we employed early-stopping if the validation performance failed to improve after 200 training epochs. This method effectively prevents RNN overfitting on the training data. According to this criterion, a more flexible model may demonstrate worse performance than a less flexible one, as the training for the former could be halted early due to insufficient training data. However, it is expected that the more flexible model would continue to improve with additional training data (e.g., see Fig. 3).

We note that, in the rich-data situation, this training-validation-test split in (nested) cross-validation is better than the typical usage of Akaike information criteria (AIC)[62], corrected AIC (AICc)[63], or Bayesian information criteria (BIC)[64] in cognitive modeling, due to the following reasons[65]: the (nested) cross-validation provides a direct and unbiased estimate of the expected extra-sample test error, which reflects the generalization performance on new data points with inputs not necessarily appearing in the training dataset; in contrast, AIC, AICc, and BIC can only provide asymptotically unbiased estimates of in-sample test error under some conditions (e.g., models are linear in their parameters), measuring the generalization performance on new data points with inputs always appearing in the training dataset (the labels could be different from those in the training dataset due to noise). In addition, the calculation of model complexity (the number of effective parameters) required by AIC/AICc/BIC is difficult to obtain in neural networks due to their explicit regularizations (e.g., L1 regularization in our study) and implicit regularizations[66].

Knowledge distillation

We employ the knowledge distillation framework[37] to fit models to individual subjects, while simultaneously leveraging group data: first fitting a teacher network to data from multiple subjects, and then fitting a student network to the outputs of the teacher network corresponding to an individual subject.

Teacher network

In the teacher network (TN), each subject is represented by a one-hot vector. This vector projects through a fully-connected linear layer into a subject-embedding vector e_{sub}, which is provided as an additional input to the RNN. The TN uses 20 units in its hidden layer, and uses the same output layer and loss (cross-entropy between the next-trial action and the predicted next-trial action probability) as in previous GRU models.

Student network

The student network (SN) has the same architecture as previous tiny GRUs. The only difference is that, during training and validation, the loss is defined as cross-entropy between the next-trial action probability provided by the teacher and the next-trial action probability predicted by the student:

$$\mathcal{L} = -\sum_{n=1}^{N_{\text{session}}}\sum_{t=1}^{T_n}\sum_{a=1}^{N_a} \Pr^{TN}[a_t = a|\text{past observations}] \log \Pr^{SN}[a_t = a|\text{past observations}], \tag{31}$$

where N_{session} is the number of sessions, T_n is the number of trials in session n, and N_a is the number of actions ($N_a = 2$ in our tasks).

Training, validation, and test data

To study the influence of the number of training trials from one representative mouse on the performance of knowledge distillation, we employed a procedure different from nested cross-validation. This procedure splits the data from animal M into two sets. The first set consisted of 25% of the trials and was used as a hold-out M-test dataset. The second set consisted of the remaining 75% trials, from where smaller datasets of different sizes were sampled. From each sampled dataset, 90% of the trials were used for training (M-training dataset) and 10% for validation (M-validation dataset). Next, we split the data from all other animals, with 90% of the data used for training (O-training dataset) and 10% for validation (O-validation dataset).

After dividing the datasets as described above, we trained the models. The solo GRUs were trained to predict choices on the M-training dataset and selected on the M-validation dataset. The teacher GRUs were trained to predict choices on the M- and O-training datasets and selected on the M- and O-validation datasets. The number of embedding units in the teacher GRUs was selected based on the M-validation dataset. The student GRUs were trained on the M-training dataset and selected on the M-validation dataset, but with the training target of action probabilities provided by the teacher GRUs. Here the student GRUs and the corresponding teacher GRUs were trained on the same M-training dataset. Finally, all models were evaluated on the unseen M-test data.

When training the student GRUs, due to symmetry in the task, we augment the M-training datasets by flipping the action and second-stage states, resulting in an augmented dataset that is four times the size of the original one, similar to [30]. One key difference between our augmentation procedure and that of [30] is that the authors augmented the data for training the group RNNs, where the potential action bias presented in the original dataset (and other related biases) becomes invisible to the RNNs. In contrast, our teacher RNNs are trained only on the original dataset, where any potential action biases can be learned. Even if we augment the training data later for the student networks, the biases learned by the teacher network can still be transferred into the student networks. In addition to direct augmentation, simulating the teacher network can be another method to generate pseudo-data. The benefit of these pseudo-data was discussed in model compression [67].

Phase portraits

Models with $d = 1$

Logit. In each trial t, a model predicts the action probabilities $\Pr(a_t = A_1)$ and $\Pr(a_t = A_2)$. We define the logit $L(t)$ (log-odds) at trial t as $L(t) = \log\left(\Pr(a_t = A_1)/\Pr(a_t = A_2)\right)$. When applied to probabilities computed via softmax, the logit yields $L(t) = \log\left(e^{\beta o_t^{(1)}}/e^{\beta o_t^{(2)}}\right) = \beta(o_t^{(1)} - o_t^{(2)})$, where $o_t^{(i)}$ is the model's output for action $a_t = A_i$ before softmax. Thus, the logit can be viewed as reflecting the preference for action A_1 over A_2: in RNNs, the logit corresponds to the score difference $o_t^{(1)} - o_t^{(2)}$; in model-free and model-based RL models, the logit is proportional to the difference in first-stage action values $Q_t(A_1) - Q_t(A_2)$; in latent-state models, the logit is proportional to the difference in latent-state probabilities $\Pr_t(h = 1) - \Pr_t(h = 2) = 2\Pr_t(h = 1) - 1$.

Logit change. We define the logit change, $\delta L(t)$, in trial t as the difference between $L(t+1)$ and $L(t)$. In one-dimensional models, $\delta L(t)$ is a function of the input and $L(t)$, forming a vector field.

Stability of fixed points. Here we derive the stability of a fixed point in one-dimensional discrete dynamical systems. The system's dynamics update according to:

$$L_{\text{next}} = f_I(L), \tag{32}$$

where L is the current-trial logit, L_{next} is the next-trial logit, and f_I is a function determined by input I (omitted for simplicity). At a fixed point, denoted by $L = L^*$, we have

$$L^* = f(L^*). \tag{33}$$

Next, we consider a small perturbation ΔL around the fixed point:

$$\begin{aligned} L_{\text{next}} &= f(L^* + \Delta L) \\ &\approx f(L^*) + f'(L^*)\Delta L \\ &= L^* + f'(L^*)\Delta L. \end{aligned} \tag{34}$$

The fixed point is stable only when $-1 < f'(L^*) < 1$. Because the logit change δL is defined as $\delta L = g(L) = f(L) - L$, we have the stability condition $-2 < g'(L^*) < 0$.

Effective learning rate and slope. In the one-dimensional RL models with prediction error updates and constant learning rate α, we have

$$g(L) = \alpha(L^* - L), \tag{35}$$

where g(L) is the logit change at L. In general, to obtain a generalized form of $g(L) = \alpha(L)(L^* - L)$ with a non-constant learning rate, we define the effective learning rate $\alpha(L)$ at L relative to a stable fixed point L^* as:

$$\alpha(L) = -\frac{g(L) - g(L^*)}{L - L^*} = -\frac{g(L)}{L - L^*}. \tag{36}$$

At L^*, $\alpha(L^*)$ is the negative slope $-g'(L^*)$ of the tangent at L^*. However, for general $L \neq L^*$, $\alpha(L)$ is the negative slope of the secant connecting $(L, g(L))$ and $(L^*, 0)$, which is different from $-g'(L)$.

We have

$$\begin{aligned} \alpha'(L)\Delta L &\approx \alpha(L + \Delta L) - \alpha(L) \\ &= \frac{g(L)}{L - L^*} - \frac{g(L + \Delta L)}{L + \Delta L - L^*} \\ &\approx \frac{-\alpha(L) - g'(L)}{L - L^*}\Delta L. \end{aligned} \tag{37}$$

Letting ΔL go to zero, we have:

$$\alpha(L) = -g'(L) - \alpha'(L)(L - L^*), \tag{38}$$

which provides the relationship between the effective learning rate $\alpha(L)$ and the slope of the tangent $g'(L)$.

Models with $d > 1$

In models with more dynamical variables, $\delta L(t)$ is no longer solely a function of the input and $L(t)$ due to added degrees of freedom. In these models, the state-space is spanned by a set of dynamical variables, collected by the vector $F(t)$. For example, the action value vector is the $F(t) = (Q_t(A_1), Q_t(A_2))^T$ in the two-dimensional RL models. The vector field $\delta F(t)$ can be defined as $\delta F(t) = F(t+1) - F(t) = (Q_{t+1}(A_1) - Q_t(A_1), Q_{t+1}(A_2) - Q_t(A_2))^T$, a function of $F(t)$ and the input in trial t.

Symbolic regression

Apart from the two-dimensional vector field analysis, symbolic regression is another method for discovering concise equations that summarize the dynamics learned by GRUs. To accomplish this, we used PySR[68] to search for simple symbolic expressions of the updated dynamical variables as functions of the current dynamical variables for each possible input I (for the GRU with $d = 2$ and a diagonal readout matrix). Ultimately, this process revealed a model-free strategy featuring the "drift-to-the-other" rule.

Meta-reinforcement learning models

We trained meta-RL agents on the two-stage task (common transition: $\Pr(S_1|A_1) = \Pr(S_2|A_2) = 0.8$, rare transition: $\Pr(S_2|A_1) = \Pr(S_1|A_2) = 0.2$; see Fig. S13) implemented in NeuroGym[69]. Each second-stage state leads to a different probability of a unit reward, with the most valuable state switching stochastically ($\Pr(r = 1|S_1) = 1 - \Pr(r = 1|S_2) = 0.8$ or 0.2 with probability of 0.025 on each trial). There are three periods (discrete time steps) on one trial: Delay 1, Go, and Delay 2. During Delay 1, the agent receives the observation (choice state $S0$ and a fixation signal), and the reward (1 or 0) from second-stage states on the last trial. During Go, the agent receives the observation of the choice state and a go signal. During Delay 2, the agent receives the observation of state S1/S2 and a fixation signal. If the agent does not select action A1 or A2 during Go or select action F (Fixate) during Delay periods, a small negative reward (-0.1) is given. The contributions of second-stage states, rewards, and actions on networks are thus separated in time.

The agent architecture is a fully connected, gated recurrent neural network (long short-term memory[34]) with 48 units[24]. The input to the network consists of the current observation (state $S_0/S_1/S_2$ and a scalar fixation/go signal), a scalar reward signal of the previous time step, and a one-hot action vector of the previous time step. The network outputs a scalar baseline (value function for the current state) serving as the critic and a real-valued action vector (passed through a softmax layer to sample one action from A1/A2/F) serving as the actor. The agents are trained using the Advantage Actor-Critic RL algorithm[70] with the policy gradient loss, value estimate loss, and entropy regularization. We trained and analyzed agents for five seeds. Our agents obtained 0.64 rewards on average on each trial (0.5 rewards for chance level), close to optimal performance (0.68 rewards obtained by an oracle agent knowing the correct action).

References

[1] Allen Newell and Herbert A Simon. Computer science as empirical inquiry: Symbols and search. In *ACM Turing award lectures*, page 1975. 2007.

[2] James L McClelland, David E Rumelhart, PDP Research Group, et al. *Parallel Distributed Processing, Volume 2: Explorations in the Microstructure of Cognition: Psychological and Biological Models*, volume 2. MIT press, 1987.

[3] Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285, 2011.

[4] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017.

[5] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[6] Yael Niv. Reinforcement learning in the brain. *Journal of Mathematical Psychology*, 53(3):139–154, 2009.

[7] Matthew Botvinick, Sam Ritter, Jane X Wang, Zeb Kurth-Nelson, Charles Blundell, and Demis Hassabis. Reinforcement learning, fast and slow. *Trends in cognitive sciences*, 23(5):408–422, 2019.

[8] Anne Gabrielle Eva Collins. Reinforcement learning: Bringing together computation and cognition. *Current Opinion in Behavioral Sciences*, 29:63–68, 2019.

[9] Paul Cisek and John F Kalaska. Neural mechanisms for interacting with a world full of action choices. *Annual review of neuroscience*, 33:269–298, 2010.

[10] Marcelo G Mattar and Máté Lengyel. Planning in the brain. *Neuron*, 2022.

[11] Ramon Bartolo and Bruno B Averbeck. Prefrontal cortex predicts state switches during reversal learning. *Neuron*, 106 (6):1044–1054, 2020.

[12] Thomas Akam, Ines Rodrigues-Vaz, Ivo Marcelo, Xiangyu Zhang, Michael Pereira, Rodrigo Freire Oliveira, Peter Dayan, and Rui M Costa. The anterior cingulate cortex predicts future states to mediate model-based action selection. *Neuron*, 109(1):149–163, 2021.

[13] Kevin J Miller, Matthew M Botvinick, and Carlos D Brody. Value representations in the rodent orbitofrontal cortex drive learning, not choice. *Elife*, 11:e64575, 2022.

[14] Robert C Wilson and Anne GE Collins. Ten simple rules for the computational modeling of behavioral data. *Elife*, 8: e49547, 2019.

[15] Carolina Feher da Silva and Todd A Hare. Humans primarily use model-based inference in the two-stage task. *Nature Human Behaviour*, 4(10):1053–1066, 2020.

[16] Blake A Richards, Timothy P Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, et al. A deep learning framework for neuroscience. *Nature neuroscience*, 22(11):1761–1770, 2019.

[17] Guangyu Robert Yang and Xiao-Jing Wang. Artificial neural networks for neuroscientists: a primer. *Neuron*, 107(6): 1048–1070, 2020.

[18] Daniel LK Yamins, Ha Hong, Charles F Cadieu, Ethan A Solomon, Darren Seibert, and James J DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the national academy of sciences*, 111(23):8619–8624, 2014.

[19] Daniel LK Yamins and James J DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356–365, 2016.

[20] Christopher J Cueva and Xue-Xin Wei. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *arXiv preprint arXiv:1803.07770*, 2018.

[21] Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J Chadwick, Thomas Degris, Joseph Modayil, et al. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429–433, 2018.

[22] James CR Whittington, Timothy H Muller, Shirley Mark, Guifen Chen, Caswell Barry, Neil Burgess, and Timothy EJ Behrens. The tolman-eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 183(5):1249–1263, 2020.

[23] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *nature*, 503(7474):78–84, 2013.

[24] Jane X Wang, Zeb Kurth-Nelson, Dharshan Kumaran, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Demis Hassabis, and Matthew Botvinick. Prefrontal cortex as a meta-reinforcement learning system. *Nature neuroscience*, 21(6):860–868, 2018.

[25] Valeria Fascianelli, Fabio Stefanini, Satoshi Tsujimoto, Aldo Genovesio, and Stefano Fusi. Neural representational geometry correlates with behavioral differences between monkeys. *bioRxiv*, pages 2022–10, 2022.

[26] Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nature neuroscience*, 22(2):297–306, 2019.

[27] Kristopher T Jensen, Guillaume Hennequin, and Marcelo G Mattar. A recurrent network model of planning explains hippocampal replay and human behavior. *bioRxiv*, pages 2023–01, 2023.

[28] Joshua C Peterson, David D Bourgin, Mayank Agrawal, Daniel Reichman, and Thomas L Griffiths. Using large-scale experiments and machine learning to discover theories of human decision-making. *Science*, 372(6547):1209–1214, 2021.

[29] Amir Dezfouli, Kristi Griffiths, Fabio Ramos, Peter Dayan, and Bernard W Balleine. Models that learn how humans learn: The case of decision-making and its disorders. *PLoS computational biology*, 15(6):e1006903, 2019.

[30] Mingyu Song, Yael Niv, and Mingbo Cai. Using recurrent neural networks to understand human reward learning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 43, 2021.

[31] Paul I Jaffe, Russell A Poldrack, Robert J Schafer, and Patrick G Bissett. Modelling human behaviour in cognitive tasks with latent dynamical systems. *Nature Human Behaviour*, pages 1–15, 2023.

[32] Thomas Akam, Rui Costa, and Peter Dayan. Simple plans or sophisticated habits? state, transition and learning interactions in the two-step task. *PLoS computational biology*, 11(12):e1004648, 2015.

[33] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[34] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[35] William Bialek. On the dimensionality of behavior. *Proceedings of the National Academy of Sciences*, 119(18):e2021860119, 2022.

[36] Surya Ganguli. Measuring the dimensionality of behavior. *Proceedings of the National Academy of Sciences*, 119(43): e2205791119, 2022.

[37] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021.

[38] Oded Galor. *Discrete dynamical systems*. Springer Science & Business Media, 2007.

[39] Miles Cranmer, Alvaro Sanchez Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *Advances in Neural Information Processing Systems*, 33:17429–17442, 2020.

[40] Rylan Schaeffer, Mikail Khona, Leenoy Meshulam, International Brain Laboratory, and Ila Rani Fiete. Reverse-engineering recurrent neural network solutions to a hierarchical inference task for mice. *bioRxiv*, pages 2020–06, 2020.

[41] Matan Fintz, Margarita Osadchy, and Uri Hertz. Using deep learning to predict human decisions and using cognitive models to explain deep learning models. *Scientific reports*, 12(1):4736, 2022.

[42] Christopher Langdon and Tatiana A Engel. Latent circuit inference from heterogeneous neural responses during cognitive tasks. *bioRxiv*, pages 2022–01, 2022.

[43] Saurabh Vyas, Matthew D Golub, David Sussillo, and Krishna V Shenoy. Computation through neural population dynamics. *Annual review of neuroscience*, 43:249–275, 2020.

[44] Mark M Churchland, John P Cunningham, Matthew T Kaufman, Justin D Foster, Paul Nuyujukian, Stephen I Ryu, and Krishna V Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012.

[45] Yang Xie, Peiyao Hu, Junru Li, Jingwen Chen, Weibin Song, Xiao-Jing Wang, Tianming Yang, Stanislas Dehaene, Shiming Tang, Bin Min, et al. Geometry of sequence working memory in macaque prefrontal cortex. *Science*, 375(6581): 632–639, 2022.

[46] Jay Hennig, Sandra A. Romero Pinto, Takahiro Yamaguchi, Scott W. Linderman, Naoshige Uchida, and Samuel J. Gershman. Emergence of belief-like representations through reinforcement learning. *bioRxiv*, 2023. doi: 10.1101/2023.04.04.535512. URL https://www.biorxiv.org/content/early/2023/04/07/2023.04.04.535512.

[47] Adrian Valente, Jonathan W Pillow, and Srdjan Ostojic. Extracting computational mechanisms from neural data using low-rank rnns. In *Advances in Neural Information Processing Systems*.

[48] Boyuan Chen, Kuang Huang, Sunand Raghupathi, Ishaan Chandratreya, Qiang Du, and Hod Lipson. Automated discovery of fundamental variables hidden in experimental data. *Nature Computational Science*, 2(7):433–442, 2022.

[49] Pantelis R Vlachas, Georgios Arampatzis, Caroline Uhler, and Petros Koumoutsakos. Multiscale simulations of complex systems by learning their effective dynamics. *Nature Machine Intelligence*, 4(4):359–366, 2022.

[50] Francesca Mastrogiuseppe and Srdjan Ostojic. Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron*, 99(3):609–623, 2018.

[51] Manuel Beiran, Alexis Dubreuil, Adrian Valente, Francesca Mastrogiuseppe, and Srdjan Ostojic. Shaping dynamics with multiple populations in low-rank recurrent networks. *Neural Computation*, 33(6):1572–1615, 2021.

[52] Alexis Dubreuil, Adrian Valente, Manuel Beiran, Francesca Mastrogiuseppe, and Srdjan Ostojic. The role of population structure in computations through neural dynamics. *Nature neuroscience*, 25(6):783–794, 2022.

[53] P Read Montague, Raymond J Dolan, Karl J Friston, and Peter Dayan. Computational psychiatry. *Trends in cognitive sciences*, 16(1):72–80, 2012.

[54] Quentin JM Huys, Michael Browning, Martin P Paulus, and Michael J Frank. Advances in the computational understanding of mental illness. *Neuropsychopharmacology*, 46(1):3–19, 2021.

[55] John P Cunningham and Byron M Yu. Dimensionality reduction for large-scale neural recordings. *Nature neuroscience*, 17(11):1500–1509, 2014.

[56] Nathaniel D Daw, Samuel J Gershman, Ben Seymour, Peter Dayan, and Raymond J Dolan. Model-based influences on humans' choices and striatal prediction errors. *Neuron*, 69(6):1204–1215, 2011.

[57] Cole Hurwitz, Nina Kudryashova, Arno Onken, and Matthias H Hennig. Building population models for large-scale neural recordings: Opportunities and pitfalls. *Current opinion in neurobiology*, 70:64–73, 2021.

[58] Joana Soldado Magraner, Valerio Mante, and Maneesh Sahani. Inferring context-dependent computations through linear approximations of prefrontal cortex dynamics. *bioRxiv*, pages 2023–02, 2023.

[59] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[60] Scott W Linderman, Andrew C Miller, Ryan P Adams, David M Blei, Liam Paninski, and Matthew J Johnson. Recurrent switching linear dynamical systems. *arXiv preprint arXiv:1610.08466*, 2016.

[61] Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Russ R Salakhutdinov. On multiplicative integration with recurrent neural networks. *Advances in neural information processing systems*, 29, 2016.

[62] Hirotugu Akaike. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6): 716–723, 1974.

[63] Joseph E Cavanaugh. Unifying the derivations for the akaike and corrected akaike information criteria. *Statistics & Probability Letters*, 33(2):201–208, 1997.

[64] Gideon Schwarz. Estimating the dimension of a model. *The annals of statistics*, pages 461–464, 1978.

[65] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[66] Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. Implicit regularization of discrete gradient dynamics in linear neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[67] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[68] Miles Cranmer. Pysr: Fast & parallelized symbolic regression in python/julia, September 2020. URL `http://doi.org/10.5281/zenodo.4041459`.

[69] Manuel Molano-Mazon, Joao Barbosa, Jordi Pastor-Ciurana, Marta Fradera, Ru-Yuan Zhang, Jeremy Forest, Jorge del Pozo Lerida, Li Ji-An, Christopher J Cueva, Jaime de la Rocha, et al. Neurogym: An open resource for developing and sharing neuroscience tasks. 2022.

[70] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.