
A neurally plausible model learns successor representations in partially observable environments

Eszter Vertes **Maneesh Sahani**
Gatsby Computational Neuroscience Unit
University College London
London, W1T 4JG
{eszter, maneesh}@gatsby.ucl.ac.uk

Abstract

Animals need to devise strategies to maximize returns while interacting with their environment based on incoming noisy sensory observations. Task-relevant states, such as the agent’s location within an environment or the presence of a predator, are often not directly observable but must be inferred using available sensory information. Successor representations (SR) have been proposed as a middle-ground between model-based and model-free reinforcement learning strategies, allowing for fast value computation and rapid adaptation to changes in the reward function or goal locations. Indeed, recent studies suggest that features of neural responses are consistent with the SR framework. However, it is not clear how such representations might be learned and computed in partially observed, noisy environments. Here, we introduce a neurally plausible model using *distributional successor features*, which builds on the distributed distributional code for the representation and computation of uncertainty, and which allows for efficient value function computation in partially observed environments via the successor representation. We show that distributional successor features can support reinforcement learning in noisy environments in which direct learning of successful policies is infeasible.

1 Introduction

Humans and other animals are able to evaluate long-term consequences of their actions and adapt their behaviour to maximize reward across different environments. This behavioural flexibility is often thought to result from interactions between two adaptive systems implementing model-based and model-free reinforcement learning (RL).

Model-based learning allows for flexible goal-directed behaviour, acquiring an internal model of the environment which is used to evaluate the consequences of actions. As a result, an agent can rapidly adjust its policy to localized changes in the environment or in the reward function. But this flexibility comes at a high computational cost, as optimal actions and value functions depend on expensive simulations in the model. Model-free methods, on the other hand, learn cached values for states and actions, enabling rapid action selection. This approach, however, is particularly slow to adapt to changes in the task, as adjusting behaviour even to localized changes, e.g. in the placement of the reward, requires updating cached values at all states in the environment. It has been suggested that the brain makes use both of these complementary approaches, and that they may compete for behavioural control [Daw et al., 2005]; indeed, several behavioural studies suggest that subjects implement a hybrid of model-free and model-based strategies [Daw et al., 2011, Glascher et al., 2010].

Successor representations [SR; Dayan, 1993] augment the internal state used by model-free systems by the expected future occupancy of each world state. SRs can be viewed as a *precompiled* representation of the model under a given policy. Thus, the SRs fall in between model-free and model-based

approaches and can reproduce a range of corresponding behaviours [Russek et al., 2017]. Recent studies have argued for evidence consistent with SRs in rodent hippocampal and human behavioural data [Stachenfeld et al., 2017, Momennejad et al., 2017].

Motivated by both theoretical and experimental work arguing that neural RL systems operate over latent states and need to handle state uncertainty [Dayan and Daw, 2008, Gershman, 2018, Starkweather et al., 2017], our work takes the successor framework further by considering partially observable environments. Adopting the framework of distributed distributional coding [Vértes and Sahani, 2018], we show how learnt latent dynamical models of the environment can be naturally integrated with SRs defined over the latent space. We begin with short overviews of reinforcement learning in the partially observed setting (section 2); the SR (section 3); and distributed distributional codes (DDCs) (section 4). In section 5, we describe how using DDCs in the generative and recognition models leads to a particularly simple algorithm for learning latent state dynamics and the associated SR.

2 Partially observable Markov decision processes

Markov decision processes (MDP) provide a framework for modelling a wide range of sequential decision-making tasks relevant for reinforcement learning. An MDP is defined by a set of states S and actions A , a reward function $R : S \times A \rightarrow \mathbb{R}$, and a probability distribution $\mathcal{T}(s'|s, a)$ that describes the Markovian dynamics of the states conditioned on actions of the agent. For notational convenience we will take the reward function to be independent of action, depending only on state; but the approach we describe is easily extended to the more general case. A partially observable Markov decision process (POMDP) is a generalization of an MDP where the Markovian states $s \in S$ are not directly observable to the agent. Instead, the agent receives observations ($o \in O$) that depend on the current *latent* state via an observation process $\mathcal{Z}(o|s)$. Formally, a POMDP is a tuple: $(S, A, \mathcal{T}, R, O, \mathcal{Z}, \gamma)$, comprising the objects defined above and the discount factor γ . POMDPs can be defined over either discrete or continuous state spaces. Here, we focus on the more general continuous case, although the model we present is applicable to discrete state spaces as well.

3 The successor representation

As an agent explores an environment, the states it visits are ordered by the agent’s policy and the transition structure of the world. State representations that respect this dynamic ordering are likely to be more efficient for value estimation and may promote more effective generalization. This may not be true of the observed state coordinates. For instance, a barrier in a spatial environment might mean that two states with adjacent physical coordinates are associated with very different values.

Dayan [1993] argued that a natural state space for model-free value estimation is one where distances between states reflect the similarity of future paths given the agent’s policy. The successor representation (Dayan, 1993; SR) for state s_i is defined as the expected discounted sum of future occupancies for each state s_j , given the current state s_i :

$$M^\pi(s_i, s_j) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k \mathbb{I}[s_{t+k} = s_j] \mid s_t = s_i \right]. \quad (1)$$

That is, in a discrete state space, the SR is a $N \times N$ matrix where N is the number of states in the environment. The SR depends on the current policy π through the expectation in the right hand side of eq. 1, taken with respect to a (possibly stochastic) policy $p^\pi(a_t|s_t)$ and environment $\mathcal{T}(s_{t+1}|s_t, a_t)$. Importantly, the SR makes it possible to express the value function in a particularly simple form. Following from eq. 1 and the definition of the value function:

$$V^\pi(s_i) = \sum_j M^\pi(s_i, s_j) R(s_j), \quad (2)$$

where $R(s_j)$ is the immediate reward in state s_j .

The successor matrix M^π can be learned by TD learning, in much the same way as TD is used to update value functions. In particular, the SR is updated according to a TD error:

$$\delta_t(s_j) = \mathbb{I}[s_t = s_j] + \gamma M^\pi(s_{t+1}, s_j) - M^\pi(s_t, s_j), \quad (3)$$

which reflects errors in *state predictions* rather than rewards, a learning signal typically associated with model-based RL.

As shown in eq. 2, the value function can be factorized into the SR—i.e., information about expected future states under the policy—and instantaneous reward in each state¹. This modularity enables rapid policy evaluation under changing reward conditions: for a fixed policy only the reward function needs to be relearned to evaluate $V^\pi(s)$. This contrasts with both model-free and model-based algorithms, which require extensive experience or rely on computationally expensive evaluation, respectively, to recompute the value function.

3.1 Successor representation using features

The successor representation can be generalized to continuous states $s \in \mathcal{S}$ by using a set of feature functions $\{\psi_i(s)\}$ defined over \mathcal{S} . In this setting, the successor representation (also referred to as the successor feature representation or SF) encodes expected feature values instead of occupancies of individual states:

$$M^\pi(s_t, i) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \psi_i(s_{t+k}) \mid s_t, \pi\right] \quad (4)$$

Assuming that the reward function can be written (or approximated) as a linear function of the features: $R(s) = w_{rew}^T \psi(s)$ (where the feature values are collected into a vector $\psi(s)$), the value function $V(s_t)$ has a simple form analogous to the discrete case:

$$V^\pi(s_t) = w_{rew}^T M^\pi(s_t) \quad (5)$$

For consistency, we can use linear function approximation with the same set features as in eq. 4 to parametrize the successor features $M^\pi(s_t, i)$.

$$M^\pi(s_t, i) \approx \sum_j U_{ij} \psi_j(s_t) \quad (6)$$

The form of the SFs, embodied by the weights U_{ij} , can be found by temporal difference learning:

$$\Delta U_{ij} = \delta_i \psi_j(s_t) \quad \delta_i = \psi_i(s_t) + \gamma M(s_{t+1}, i) - M(s_t, i) \quad (7)$$

As we have seen in the discrete case, the TD error here signals prediction errors about features of state, rather than about reward.

4 Distributed distributional codes

Distributed distributional codes (DDC) are a candidate for the neural representation of uncertainty [Zemel et al., 1998, Sahani and Dayan, 2003] and recently have been shown to support accurate inference and learning in hierarchical latent variable models [Vértes and Sahani, 2018]. In a DDC, a population of neurons represent distributions in their firing rates implicitly, as a set of expectations:

$$\mu = \mathbb{E}_{p(s)}[\psi(s)] \quad (8)$$

where μ is a vector of firing rates, $p(s)$ is the represented distribution, and $\psi(s)$ is a vector of encoding functions specific to each neuron. DDCs can be thought of as representing exponential family distributions with sufficient statistics $\psi(s)$ using their mean parameters $\mathbb{E}_{p(s)}[\psi(s)]$ [Wainwright and Jordan, 2008].

5 Distributional successor representation

As discussed above, the successor representation can support efficient value computation by incorporating information about the policy and the environment into the state representation. However, in more realistic settings, the states themselves are not directly observable and the agent is limited to state-dependent noisy sensory information.

¹Alternatively, for the more general case of action-dependent reward, the expected instantaneous reward under the policy-dependent action in each state.

Algorithm 1 Wake-sleep algorithm in the DDC state-space model

Initialise T, W
while not converged **do**
 Sleep phase:
 sample: $\{s_t^{sleep}, o_t^{sleep}\}_{t=0\dots N} \sim p(\mathcal{S}_N, \mathcal{O}_N)$
 update W : $\Delta W \propto \sum_t (\psi(s_t^{sleep}) - f_W(\mu_{t-1}(\mathcal{O}_{t-1}^{sleep}), o_t^{sleep})) \nabla_W f_W$
 Wake phase:
 $\mathcal{O}_N \leftarrow$ {collect observations}
 infer posterior $\mu_t(\mathcal{O}_t) = f_W(\mu_{t-1}(\mathcal{O}_{t-1}), o_t)$
 update T : $\Delta T \propto (\mu_{t+1}(\mathcal{O}_{t+1}) - T\mu_t(\mathcal{O}_t))\mu_t(\mathcal{O}_t)^T$
 update observation model parameters
end while

In this section, we lay out how the DDC representation for uncertainty allows for learning and computing with successor representations defined over latent variables. First, we describe an algorithm for learning and inference in dynamical latent variable models using DDCs. We then establish a link between the DDC and successor features (eq. 4) and show how they can be combined to learn what we call the *distributional successor features*. We discuss different algorithmic and implementation-related choices for the proposed scheme and their implications.

5.1 Learning and inference in a state space model using DDCs

Here, we consider POMDPs where the state-space transition model is itself defined by a conditional DDC with means that depend linearly on the preceding state features. That is, the conditional distribution describing the latent dynamics implied by following the policy π can be written in the following form:

$$p^\pi(s_{t+1}|s_t) \Leftrightarrow \mathbb{E}_{s_{t+1}|s_t, \pi}[\psi(s_{t+1})] = T^\pi \psi(s_t) \quad (9)$$

where T^π is a matrix parametrizing the functional relationship between s_t and the expectation of $\psi(s_{t+1})$ with respect to $p^\pi(s_{t+1}|s_t)$.

The agent has access only to sensory observations o_t at each time step, and in order to be able to make use of the underlying latent structure, it has to learn the parameters of generative model $p(s_{t+1}|s_t)$, $p(o_t|s_t)$ as well as learn to perform inference in that model.

We consider online inference (filtering), i.e. at each time step t the recognition model produces an estimate $q(s_t|\mathcal{O}_t)$ of the posterior distribution $p(s_t|\mathcal{O}_t)$ given all observations up to time t : $\mathcal{O}_t = (o_1, o_2, \dots, o_t)$. As in the DDC Helmholtz machine [Vértes and Sahani, 2018], these distributions are represented by a set of expectations—i.e., by a DDC:

$$\mu_t(\mathcal{O}_t) = \mathbb{E}_{q(s_t|\mathcal{O}_t)}[\psi(s_t)] \quad (10)$$

The filtering posterior $\mu_t(\mathcal{O}_t)$ is computed iteratively, using the posterior in the previous time step $\mu_{t-1}(\mathcal{O}_{t-1})$ and the new observation o_t . Due to the Markovian structure of the state space model (see fig. 1), the recognition model can be written as a recursive function:

$$\mu_t(\mathcal{O}_t) = f_W(\mu_{t-1}(\mathcal{O}_{t-1}), o_t) \quad (11)$$

with a set of parameters W .

The recognition and generative models are updated using an adapted version of the wake-sleep algorithm [Hinton et al., 1995, Vértes and Sahani, 2018]. In the following, we describe the two phases of the algorithm in more detail (see Algorithm 1).

Sleep phase

The aim of the sleep phase is to adjust the parameters of the recognition model given the current generative model. Specifically, the recognition model should approximate the expectation of the DDC encoding functions $\psi(s_t)$ under the filtering posterior $p(s_t|\mathcal{O}_t)$. This can be achieved by moment matching, i.e., simulating a sequence of latent and observed states from the current model and

minimizing the Euclidean distance between the output of the recognition model and the sufficient statistic vector $\psi(\cdot)$ evaluated at the latent state from the next time step.

$$W \leftarrow \operatorname{argmin}_W \sum_t \|\psi(s_t^{sleep}) - f_W(\mu_{t-1}(\mathcal{O}_{t-1}^{sleep}), o_t^{sleep})\|^2 \quad (12)$$

where $\{s_t^{sleep}, o_t^{sleep}\}_{t=0 \dots N} \sim p(s_0)p(o_0|s_0) \prod_{t=0}^{N-1} p(s_{t+1}|s_t, T^\pi)p(o_{t+1}|s_{t+1})$.

This update rule can be implemented online, and after a sufficiently long sequence of simulations $\{s_t^{sleep}, o_t^{sleep}\}_t$ the recognition model will learn to approximate expectations of the form: $f_W(\mu_{t-1}(\mathcal{O}_{t-1}^{sleep}), o_t^{sleep}) \approx \mathbb{E}_{p(s_t|o_t)}[\psi(s_t)]$, yielding a DDC representation of the posterior.

Wake phase

In the wake phase, the parameters of the generative model are adapted such that it captures the sensory observations better. Here, we focus on learning the policy-dependent latent dynamics $p^\pi(s_{t+1}|s_t)$; the observation model can be learned by the approach of [Vértes and Sahani, 2018]. Given a sequence of inferred posterior representations $\{\mu_t(\mathcal{O}_t)\}$ computed using wake phase observations, the parameters of the latent dynamics T can be updated by minimizing a simple predictive cost function:

$$T \leftarrow \operatorname{argmin}_T \sum_t \|\mu_{t+1}(\mathcal{O}_{t+1}) - T\mu_t(\mathcal{O}_t)\|^2 \quad (13)$$

The intuition behind eq. 13 is that for the optimal generative model the latent dynamics satisfies the following equality: $T^* \mu_t(\mathcal{O}_t) = \mathbb{E}_{p(o_{t+1}|\mathcal{O}_t)}[\mu_{t+1}(\mathcal{O}_{t+1})]$. That is, the predictions made by combining the posterior at time t and the prior will agree with the average posterior at the next time step—making T^* a stationary point of the optimization in eq. 14. For further details on the nature of the approximation implied by the wake phase update and its relationship to variational learning, see the supplementary material. In practice, the update can be done online, using gradient steps analogous to prediction errors:

$$\Delta T \propto (\mu_{t+1}(\mathcal{O}_{t+1}) - T\mu_t(\mathcal{O}_t))\mu_t(\mathcal{O}_t)^T \quad (14)$$

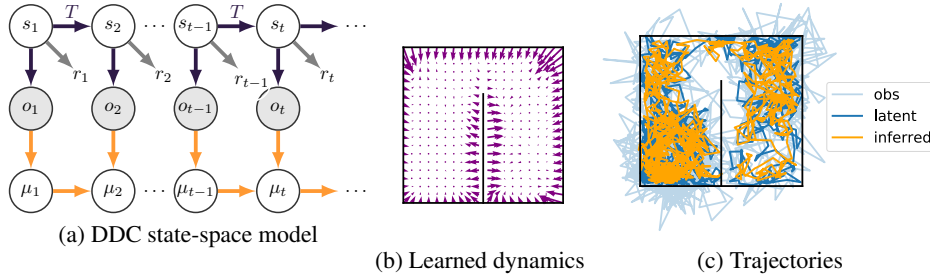


Figure 1: Learning and inference in a state-space model parametrized by a DDC. (a) The structure of the generative and recognition models. (b) Visualization of the dynamics T learned by the wake-sleep (algorithm 1). Arrows show the conditional mean $\mathbb{E}_{p(s_{t+1}|s_t)}$ for each location. (c) Posterior mean trajectories inferred using the recognition model, plotted on top of true latent and observed trajectories.

Figure 1 shows a state-space model corresponding to a random walk policy in the latent space with noisy observations, learned using DDCs (Algorithm 1). For further details of the experiment, see the supplementary material.

5.2 Learning distributional successor features

Next, we show how using a DDC to parametrize the generative model (eq. 9) allows for computing the successor features defined in the latent space in a tractable form, and how this computation can be combined with inference based on sensory observations.

Following the definition of the SFs (eq. 4):

$$M(s_t) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k \psi(s_{t+k}) | s_t, \pi\right] = \sum_{k=0}^{\infty} \gamma^k \mathbb{E}[\psi(s_{t+k}) | s_t, \pi] \quad (15)$$

We can compute the conditional expectations of the feature vector ψ in eq. 15 by applying the dynamics k times to the features $\psi(s_t)$: $\mathbb{E}_{s_{t+k}|s_t}[\psi(s_{t+k})] = T^k \psi(s_t)$. Thus, we have:

$$M(s_t) = \sum_{k=0}^{\infty} \gamma^k T^k \psi(s_t) \quad (16)$$

$$= (I - \gamma T)^{-1} \psi(s_t) \quad (17)$$

Eq. 17 is reminiscent of the result for discrete observed state spaces $M(s_i, s_j) = (I - \gamma P)_{ij}^{-1}$ [Dayan, 1993], where P is a matrix containing Markovian transition probabilities between states. In a continuous state space, however, finding a closed form solution like eq. 17 is non-trivial, as it requires evaluating a set of typically intractable integrals. The solution presented here directly exploits the DDC parametrization of the generative model and the correspondence between the features used in the DDC and the SFs.

In this framework, we can not only compute the successor features in closed form in the latent space, but also evaluate the *distributional successor features*, the posterior expectation of the SFs given a sequence of sensory observations:

$$\mathbb{E}_{s_t|\mathcal{O}_t}[M(s_t)] = (I - \gamma T)^{-1} \mathbb{E}_{s_t|\mathcal{O}_t}[\psi(s_t)] \quad (18)$$

$$= (I - \gamma T)^{-1} \mu_t(\mathcal{O}_t) \quad (19)$$

The results from this section suggest a number of different ways the distributional successor features $\mathbb{E}_{s_t|\mathcal{O}_t}[M(s_t)]$ can be learned or computed.

Learning distributional SFs during *sleep phase*

The matrix $U = (I - \gamma T)^{-1}$ needed to compute distributional SFs in eq. 19 can be learned from temporal differences in feature predictions based on *sleep phase* simulated latent state sequences (section 3.1).

Computing distributional SFs by *dynamics*

Alternatively, eq. 19 can be implemented as a fixed point of a linear dynamical system, with recurrent connections reflecting the model of the latent dynamics:

$$\tau \dot{x}_n = -x_n + \gamma T x_n + \mu_t(\mathcal{O}_t) \quad (20)$$

$$\Rightarrow x_\infty = (I - \gamma T)^{-1} \mu_t(\mathcal{O}_t) \quad (21)$$

In this case, there is no need to learn $(I - \gamma T)^{-1}$ explicitly but it is implicitly computed through dynamics. For this to work, there is an underlying assumption that the dynamical system in eq. 20 reaches equilibrium on a timescale faster than that on which the observations \mathcal{O}_t evolve.

Both of these approaches avoid having to compute the matrix inverse directly and allow for evaluation of policies given by a corresponding dynamics matrix T^π offline.

Learning distributional SFs during *wake phase*

Instead of fully relying on the learned latent dynamics to compute the distributional SFs, we can use posteriors computed by the recognition model during the wake phase, that is, using observed data. We can define the distributional SFs directly on the DDC posteriors: $\widetilde{M}(\mathcal{O}_t) = \mathbb{E}_\pi[\sum_k \gamma^k \mu_{t+k}(\mathcal{O}_{t+k}) | \mu_t(\mathcal{O}_t)]$, treating the posterior representation $\mu_t(\mathcal{O}_t)$ as a feature space over sequences of observations $\mathcal{O}_t = (o_1 \dots o_t)$. Analogously to section 3.1, $\widetilde{M}(\mathcal{O}_t)$ can be acquired by TD learning and assuming linear function approximation: $\widetilde{M}(\mathcal{O}_t) \approx U \mu_t(\mathcal{O}_t)$. The matrix U can be

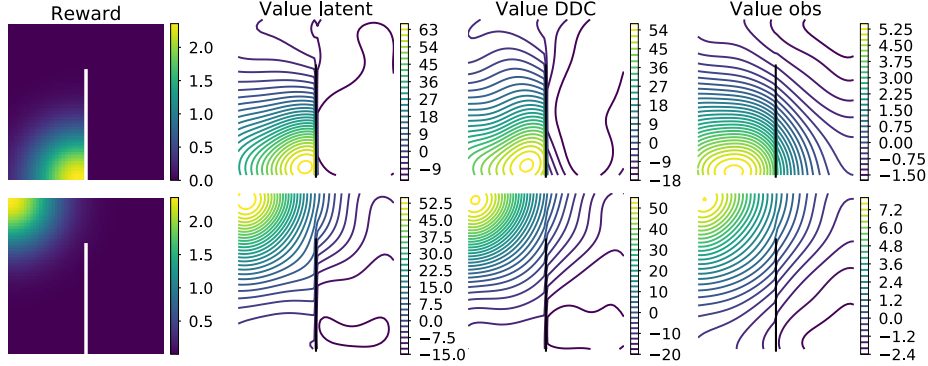


Figure 2: Value functions computed using successor features under a random walk policy

updated online, while executing a given policy and continuously inferring latent state representations using the recognition model:

$$\Delta U \propto \delta_t \mu_t(\mathcal{O}_t)^T \quad (22)$$

$$\delta_t = \mu_t(\mathcal{O}_t) + \gamma M(\mathcal{O}_{t+1}) - M(\mathcal{O}_t) \quad (23)$$

It can be shown that $\widetilde{M}(\mathcal{O}_t)$, as defined here, is equivalent to $\mathbb{E}_{s_t|\mathcal{O}_t}[M(s_t)]$ if the learned generative model is optimal—assuming no model mismatch—and the recognition model correctly infers the corresponding posteriors $\mu_t(\mathcal{O}_t)$ (see supplementary material). In general, however, exchanging the order of TD learning and inference leads to different SFs. The advantage of learning the distributional successor features in the wake phase is that even when the model does not perfectly capture the data (e.g. due to lack of flexibility or early on in learning) the learned SFs will reflect the structure in the observations through the posteriors $\mu_t(\mathcal{O}_t)$.

5.3 Value computation in a noisy 2D environment

We illustrate the importance of being able to consistently handle uncertainty in the SFs by learning value functions in a noisy environment. We use a simple 2-dimensional box environment with continuous state space that includes an internal wall. The agent does not have direct access to its spatial coordinates, but receives observations corrupted by Gaussian noise. Figure 2 shows the value functions computed using the successor features learned in three different settings: assuming direct access to latent states, treating observations as though they were noise-free state measurements, and using latent state estimates inferred from observations. The value functions computed in the latent space and computed from DDC posterior representations both reflect the structure of the environment, while the value function relying on SFs over the observed states fails to learn about the barrier.

To demonstrate that this is not simply due to using the suboptimal random walk policy, but persists through learning, we have learned successor features while adjusting the policy to a given reward function (see figure 3). The policy was learned by generalized policy iteration [Sutton and Barto, 1998], alternating between taking actions following a greedy policy and updating the successor features to estimate the corresponding value function.

The value of each state and action was computed from the value function $V(s)$ by a one-step look-ahead, combining the immediate reward with the expected value function having taken a given action:

$$Q(s_t, a_t) = r(s_t) + \gamma \mathbb{E}_{s_{t+1}|s_t, a_t}[V(s_{t+1})] \quad (24)$$

In our case, as the value function in the latent space is expressed as a linear function of the features $\psi(s)$: $V(s) = w^T U \psi(s)$ (eq. 5-6), the expectation in 24 can be expressed as:

$$\mathbb{E}_{s_{t+1}|s_t, a_t}[V(s_{t+1})] = w_{\text{rew}}^T U \cdot \mathbb{E}_{s'|s, a}[\psi(s_{t+1})] \quad (25)$$

$$= w_{\text{rew}}^T U \cdot P \cdot (\psi(s_t) \otimes \phi(a_t)) \quad (26)$$

Where P is a linear mapping, $P : \Psi \times \Phi \rightarrow \Psi$, that contains information about the distribution $p(s_{t+1}|s_t, a_t)$. More specifically, P is trained to predict $\mathbb{E}_{s_{t+1}|s_t, a_t}[\psi(s_{t+1})]$ as a bilinear function

of state and action features $(\psi(s_t), \phi(a_t))$. Given the state-action value, we can implement a greedy policy by choosing actions that maximize $Q(s, a)$:

$$a^* = \operatorname{argmax}_{a \in \mathcal{A}} Q(s_t, a_t) \quad (27)$$

$$= \operatorname{argmax}_{a \in \mathcal{A}} r(s_t) + \gamma w_{\text{rew}}^T U \cdot P \cdot (\psi(s_t) \times \phi(a_t)) \quad (28)$$

The argmax operation in eq. 28 (possibly over a continuous space of actions) could be biologically implemented by a ring attractor where the neurons receive state-dependent input through feedforward weights reflecting the tuning $(\phi(a))$ of each neuron in the ring.

Just as in figure 2, we compute the value function in the fully observed case, using inferred states or using only the noisy observations. For the latter two, we replace $\psi(s_t)$ in eq. 28 with the inferred state representation $\mu(O_t)$ and the observed features $\psi(o_t)$, respectively. As the agent follows the greedy policy and it receives new observations the corresponding SFs are adapted accordingly. Figure 3 shows the learned value functions $V^\pi(s)$, $V^\pi(\mu)$ and $V^\pi(o)$ for a given reward location and the corresponding dynamics T^π . The agent having access to the true latent state as well as the one using distributional SFs successfully learn policies leading to the rewarded location. As before, the agent learning SFs purely based on observations remains highly sub-optimal.

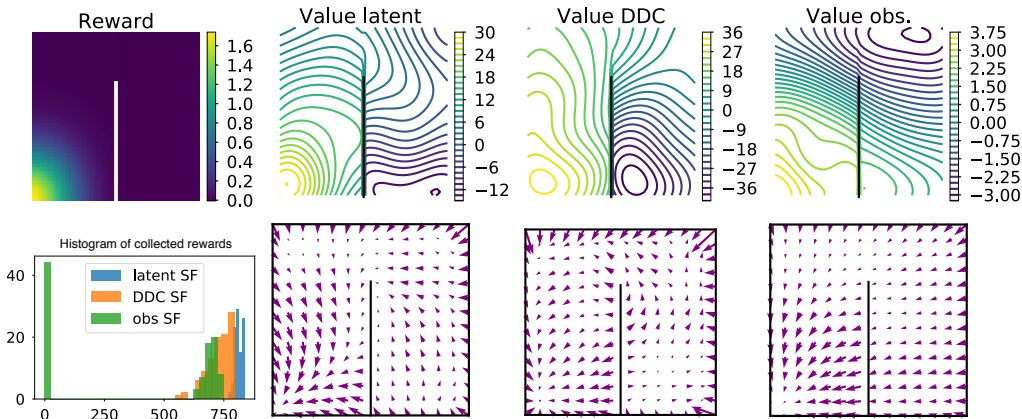


Figure 3: Value functions computed by SFs under the learned policy. Top row shows reward and value functions learned in the three different conditions. Bottom row shows histogram of collected rewards from 100 episodes with random initial states, and the learned dynamics T^π visualized as in fig. 1.

6 Discussion

We have shown that representing uncertainty over latent variables using DDCs can be naturally integrated with representations of uncertainty about future states and therefore can generalize SRs to more realistic environments with partial observability.

In our work, we have defined distributional SFs over states, using single step look-ahead to compute state-action values (eq. 24). Alternatively, SFs could be defined directly over both states and actions [Kulkarni et al., 2016, Barreto et al., 2017] with the distributional development presented here. Barreto et al. [2017, 2019] has shown that successor representations corresponding to previously learned tasks can be used as a basis to construct policies for novel tasks, enabling generalization. Our framework can be extended in a similar way, eliminating the need to adapt the SFs as the policy of the agent changes.

The framework for learning distributional successor features presented here makes a number of connections to experimental observations in the hippocampal literature. While it has been argued that the hippocampus holds an internal model of the environment and thereby supports model-based decision making [Miller et al., 2017], there is little known about how such a model is acquired. Hippocampal replays observed in rodents during periods of immobility and sleep have been interpreted

as mental simulations from an internal model of the environment, and therefore a neural substrate for model-based planning [Pfeiffer and Foster, 2013, Mattar and Daw, 2018]. Here, we propose a complementary function of replays that is to do with learning in the context of partially observed environments. The replayed sequences could serve to refine the recognition model to accurately infer distributions over latent states, just as in the *sleep phase* of our algorithm. Broadly consistent with this idea, Stella et al. [2019] recently observed replays reminiscent of random walk trajectories after an animal freely explored the environment. These paths were not previously experienced by the animal, and could indeed serve as a training signal for the recognition model. Learning to perform inference is itself a prerequisite for learning the dynamics of the latent task-relevant variables, i.e. the internal model.

References

- André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, Hado P. van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.
- André Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Židek, and Rémi Munos. Transfer in Deep Reinforcement Learning Using Successor Features and Generalised Policy Improvement. *arXiv:1901.10964 [cs]*, January 2019. URL <http://arxiv.org/abs/1901.10964>. arXiv: 1901.10964.
- Nathaniel D. Daw, Yael Niv, and Peter Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8(12):1704, December 2005. ISSN 1546-1726. doi: 10.1038/nn1560. URL <https://www.nature.com/articles/nn1560>.
- 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, March 2011. ISSN 0896-6273. doi: 10.1016/j.neuron.2011.02.027. URL <http://www.sciencedirect.com/science/article/pii/S0896627311001255>.
- Peter Dayan. Improving Generalization for Temporal Difference Learning: The Successor Representation. *Neural Computation*, 5(4):613–624, July 1993. ISSN 0899-7667. doi: 10.1162/neco.1993.5.4.613. URL <https://doi.org/10.1162/neco.1993.5.4.613>.
- Peter Dayan and Nathaniel D. Daw. Decision theory, reinforcement learning, and the brain. *Cognitive, Affective, & Behavioral Neuroscience*, 8(4):429–453, December 2008. ISSN 1531-135X. doi: 10.3758/CABN.8.4.429. URL <https://doi.org/10.3758/CABN.8.4.429>.
- Samuel J. Gershman. The Successor Representation: Its Computational Logic and Neural Substrates. *J. Neurosci.*, 38(33):7193–7200, August 2018. ISSN 0270-6474, 1529-2401. doi: 10.1523/JNEUROSCI.0151-18.2018. URL <http://www.jneurosci.org/content/38/33/7193>.
- Jan Gläscher, Nathaniel Daw, Peter Dayan, and John P. O’Doherty. States versus Rewards: Dissociable Neural Prediction Error Signals Underlying Model-Based and Model-Free Reinforcement Learning. *Neuron*, 66(4):585–595, May 2010. ISSN 0896-6273. doi: 10.1016/j.neuron.2010.04.016. URL <http://www.sciencedirect.com/science/article/pii/S0896627310002874>.
- Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A Kernel Two-Sample Test. *Journal of Machine Learning Research*, 13:723–773, March 2012. URL <http://jmlr.csail.mit.edu/papers/v13/gretton12a.html>.
- G E Hinton, P Dayan, B J Frey, and R M Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, May 1995. ISSN 0036-8075.
- Tejas D. Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J. Gershman. Deep Successor Reinforcement Learning. *arXiv:1606.02396 [cs, stat]*, June 2016. URL <http://arxiv.org/abs/1606.02396>. arXiv: 1606.02396.
- Marcelo G. Mattar and Nathaniel D. Daw. Prioritized memory access explains planning and hippocampal replay. *Nature Neuroscience*, 21(11):1609, November 2018. ISSN 1546-1726. doi: 10.1038/s41593-018-0232-z. URL <https://www.nature.com/articles/s41593-018-0232-z>.

- Kevin J Miller, Matthew M Botvinick, and Carlos D Brody. Dorsal hippocampus contributes to model-based planning. *Nature Neuroscience*, 20(9):1269–1276, September 2017. ISSN 1097-6256, 1546-1726. doi: 10.1038/nn.4613. URL <http://www.nature.com/articles/nn.4613>.
- I. Momennejad, E. M. Russek, J. H. Cheong, M. M. Botvinick, N. D. Daw, and S. J. Gershman. The successor representation in human reinforcement learning. *Nature Human Behaviour*, 1(9):680, September 2017. ISSN 2397-3374. doi: 10.1038/s41562-017-0180-8. URL <https://www.nature.com/articles/s41562-017-0180-8>.
- Brad E. Pfeiffer and David J. Foster. Hippocampal place-cell sequences depict future paths to remembered goals. *Nature*, 497(7447):74–79, May 2013. ISSN 1476-4687. doi: 10.1038/nature12112. URL <https://www.nature.com/articles/nature12112>.
- Evan M. Russek, Ida Momennejad, Matthew M. Botvinick, Samuel J. Gershman, and Nathaniel D. Daw. Predictive representations can link model-based reinforcement learning to model-free mechanisms. *PLOS Computational Biology*, 13(9):e1005768, September 2017. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1005768. URL <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005768>.
- Maneesh Sahani and Peter Dayan. Doubly Distributional Population Codes: Simultaneous Representation of Uncertainty and Multiplicity. *Neural Computation*, 15(10):2255–2279, October 2003. ISSN 0899-7667. doi: 10.1162/089976603322362356. URL <http://dx.doi.org/10.1162/089976603322362356>.
- Kimberly L. Stachenfeld, Matthew M. Botvinick, and Samuel J. Gershman. The hippocampus as a predictive map. *Nature Neuroscience*, 20(11):1643–1653, November 2017. ISSN 1546-1726. doi: 10.1038/nn.4650. URL <https://www.nature.com/articles/nn.4650>.
- Clara Kwon Starkweather, Benedicte M. Babayan, Naoshige Uchida, and Samuel J. Gershman. Dopamine reward prediction errors reflect hidden-state inference across time. *Nat. Neurosci.*, 20(4):581–589, April 2017. ISSN 1546-1726. doi: 10.1038/nn.4520.
- Federico Stella, Peter Baracs, Joseph O’Neill, and Jozsef Csicsvari. Hippocampal Reactivation of Random Trajectories Resembling Brownian Diffusion. *Neuron*, February 2019. ISSN 0896-6273. doi: 10.1016/j.neuron.2019.01.052. URL <http://www.sciencedirect.com/science/article/pii/S0896627319300790>.
- Richard S. Sutton and Andrew G. Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.
- Eszter Vértés and Maneesh Sahani. Flexible and accurate inference and learning for deep generative models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 4166–4175. Curran Associates, Inc., 2018. URL <http://papers.nips.cc/paper/7671-flexible-and-accurate-inference-and-learning-for-deep-generative-models.pdf>.
- Martin J. Wainwright and Michael I. Jordan. Graphical Models, Exponential Families, and Variational Inference. *Found. Trends Mach. Learn.*, 1(1-2):1–305, January 2008. ISSN 1935-8237. doi: 10.1561/2200000001. URL <http://dx.doi.org/10.1561/2200000001>.
- Richard S. Zemel, Peter Dayan, and Alexandre Pouget. Probabilistic interpretation of population codes. *Neural computation*, 10(2):403–430, 1998.

Supplementary material

A Approximations in the wake phase update

Here, we give some additional insights into the nature of the approximation implied by the wake phase update for the DDC state-space model and discuss its link to variational methods.

According to the standard M step in variational EM, the model parameters are updated to maximize the expected log-joint of the model under the approximate posterior distributions:

$$\Delta\theta \propto \nabla_{\theta} \sum_t \mathbb{E}_{q(s_t, s_{t+1} | \mathcal{O}_{t+1})} [\log p_{\theta}(s_{t+1} | s_t)] \quad (29)$$

$$= \nabla_{\theta} \sum_t - \int q(s_t, s_{t+1} | \mathcal{O}_{t+1}) (\log p_{\theta}(s_{t+1} | s_t) + \log q(s_t | \mathcal{O}_{t+1})) d(s_t, s_{t+1}) \quad (30)$$

$$= \nabla_{\theta} \sum_t -KL[q(s_t, s_{t+1} | \mathcal{O}_{t+1}) \| p_{\theta}(s_{t+1} | s_t) q(s_t | \mathcal{O}_{t+1})] \quad (31)$$

After projecting the distributions appearing in the KL divergence (eq. 31) into the joint exponential family defined by sufficient statistics $[\psi(s_t), \psi(s_{t+1})]$, they can be represented using the corresponding mean parameters:

$$q(s_t, s_{t+1} | \mathcal{O}_{t+1}) \xrightarrow{\mathcal{P}} \begin{bmatrix} \mathbb{E}_{q(s_t, s_{t+1} | \mathcal{O}_{t+1})} [\psi(s_t)] \\ \mathbb{E}_{q(s_t, s_{t+1} | \mathcal{O}_{t+1})} [\psi(s_{t+1})] \end{bmatrix} = \begin{bmatrix} \mu_t(\mathcal{O}_{t+1}) \\ \mu_{t+1}(\mathcal{O}_{t+1}) \end{bmatrix} \quad (32)$$

$$p_{\theta}(s_{t+1} | s_t) q(s_t | \mathcal{O}_{t+1}) \xrightarrow{\mathcal{P}} \begin{bmatrix} \mathbb{E}_{p_{\theta}(s_{t+1} | s_t) q(s_t | \mathcal{O}_{t+1})} [\psi(s_t)] \\ \mathbb{E}_{p_{\theta}(s_{t+1} | s_t) q(s_t | \mathcal{O}_{t+1})} [\psi(s_{t+1})] \end{bmatrix} = \begin{bmatrix} \mu_t(\mathcal{O}_{t+1}) \\ T\mu_t(\mathcal{O}_{t+1}) \end{bmatrix} \quad (33)$$

To restrict ourselves to online inference, we can make a further approximation: $\mu_t(\mathcal{O}_{t+1}) \approx \mu_t(\mathcal{O}_t)$. Thus, the wake phase update can be thought of as replacing the KL divergence in equation 31 by the Euclidean distance between the (projected) mean parameter representations in eq. 32-33.

$$\sum_t \|\mu_{t+1}(\mathcal{O}_{t+1}) - T\mu_t(\mathcal{O}_t)\|^2 \quad (34)$$

Note that this cost function is directly related to the maximum mean discrepancy (Gretton et al. [2012]; MMD)—a non-parametric distance metric between two distributions—with a finite dimensional RKHS.

B Equivalence of $\mathbb{E}_{p(s_t | \mathcal{O}_t)} [M(s_t)]$ and $\widetilde{M}(\mu_t(\mathcal{O}_t))$

$$\widetilde{M}(\mu_t(\mathcal{O}_t)) = \mathbb{E}_{p(\mathcal{O}_{>t} | \mathcal{O}_t)} \left[\sum_k \gamma^k \mu_{t+k}(\mathcal{O}_{t+k}) \right] \quad (35)$$

where

$$\mathbb{E}_{p(\mathcal{O}_{>t} | \mathcal{O}_t)} [\mu_{t+k}(\mathcal{O}_{t+k})] = \mathbb{E}_{p(\mathcal{O}_{t+1:t+k} | \mathcal{O}_t)} [\mu_{t+k}(\mathcal{O}_{t+k})] \quad (36)$$

$$= \int d\mathcal{O}_{t+1:t+k} p(\mathcal{O}_{t+1:t+k} | \mathcal{O}_t) \int ds_{t+k} p(s_{t+k} | \mathcal{O}_{t+k}) \psi(s_{t+k}) \quad (37)$$

$$= \int d\mathcal{O}_{t+1:t+k} p(\mathcal{O}_{t+1:t+k} | \mathcal{O}_t) \int ds_{t+k} \frac{p(s_{t+k}, \mathcal{O}_{t+1:t+k} | \mathcal{O}_t)}{p(\mathcal{O}_{t+1:t+k} | \mathcal{O}_t)} \psi(s_{t+k}) \quad (38)$$

$$= \int ds_{t+k} \int d\mathcal{O}_{t+1:t+k} p(s_{t+k}, \mathcal{O}_{t+1:t+k} | \mathcal{O}_t) \psi(s_{t+k}) \quad (39)$$

$$= T^k \mu_t \quad (40)$$

$$(41)$$

Thus we have:

$$\widetilde{M}(\mu_t(\mathcal{O}_t)) = \sum_k \gamma^k T^k \mu_t \quad (42)$$

$$= (I - \gamma T)^{-1} \mu_t \quad (43)$$

$$= \mathbb{E}_{p(s_t|\mathcal{O}_t)}[M(s_t)] \quad (44)$$

C Further experimental details

Figure 1: Learning and inference in the DDC state-space model

The generative model corresponding to a random walk policy:

$$p(s_{t+1}|s_t) = [s_t + \tilde{\eta}]_{\text{WALLS}}, \quad (45)$$

$$p(o_t|s_t) = s_t + \xi \quad (46)$$

Where $[\cdot]_{\text{WALLS}}$ indicates the constraints introduced by the walls in the environment (outer walls are of unit length). $\eta \sim \mathcal{N}(0, \sigma_s = 1.)$, $\tilde{\eta} = 0.06 * \eta / \|\eta\|$, $\xi \sim \mathcal{N}(0, \sigma_o = 0.1)$, $s_t, o_t \in \mathbb{R}^2$

We used $K=100$ Gaussian features with width $\sigma_\psi = 0.3$ for both the latent and observed states. A small subset of features were truncated along the internal wall, to limit the artifacts from the function approximation. Alternatively, a features with various spatial scales can also be used. The recursive recognition model was parametrized linearly using the features:

$$f_W(\mu_{t-1}, o_t) = W[T\mu_{t-1}; \psi(o_t)] \quad (47)$$

As sampling from the DDC parametrized latent dynamics is not tractable in general, in the sleep phase, we generated approximate samples from a Gaussian distribution with consistent mean. The generative and recognition models were trained through 50 wake-sleep cycles, with $3 \cdot 10^4$ sleep samples, and $5 \cdot 10^4$ wake phase observations.

The latent dynamics in Fig.1b is visualized by approximating the mean dynamics as a linear readout from the DDC: $\mathbb{E}_{s_{t+1}|s_t}[s_{t+1}] \approx \alpha T \psi(s_t)$ where $s \approx \alpha \psi(s)$.

Figure 2 To compute the value functions under the random walk policy we computed the SFs based on the latent ($\psi(s)$), inferred (μ) or observed ($\psi(o)$) features, with discount factor $\gamma = 0.99$. In each case, we estimated the reward vector w_{rew} using the available state information.

Figure 3 To construct the state-action value function, we used 10 features over actions $\phi(a)$, von Mises functions ($\kappa = 2.$) arranged evenly on $[0, 2\pi]$. The policy iteration was run for 500 cycles, and in each cycle an episode of 500 steps was collected according to the greedy policy. The visited latent, inferred or observed state sequences were used to update the corresponding SFs to re-evaluate the policy. To facilitate faster learning, only episodes with positive returns were used to update the SFs.