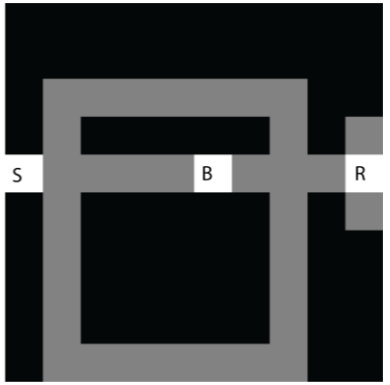
# Predictive representations can link model-based reinforcement learning to model-free mechanisms

Russek, Momennejad, Botvinick, Gershman and Daw

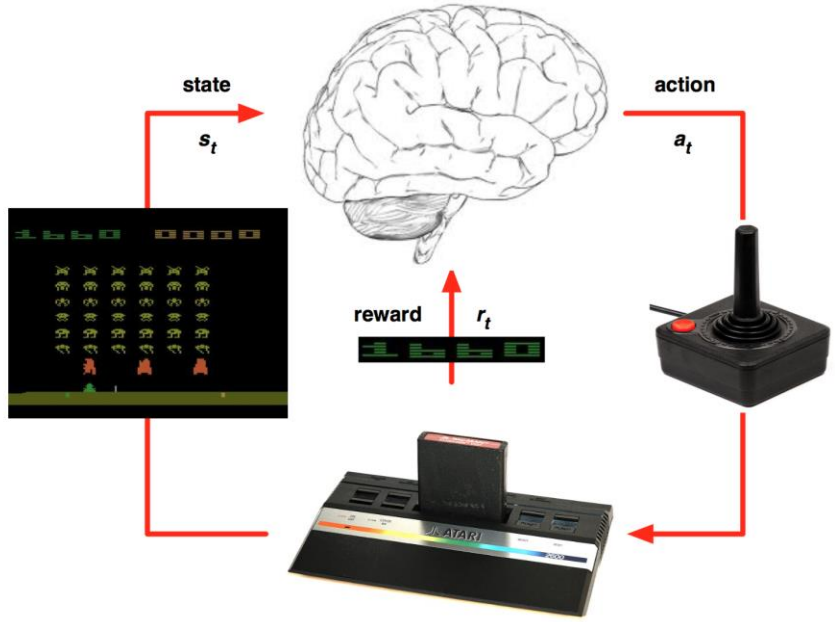
PLOS Computational Biology

# Background: General RL Framework

- Set of "states" that you can move between by taking "actions"
- Reward is distributed in the environment
- Objective is to choose actions that maximise future reward
- Value of a state is total future expected reward
- $V(S_t) = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \Sigma^{\infty}_{n=1} \gamma^n R_{t+n}$

  $= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \ldots) = R_{t+1} + \gamma V(S_{t+1})$
- $Q(S_t, A_t) = R(S_t, A_t) + \gamma V(S_{t+1})$

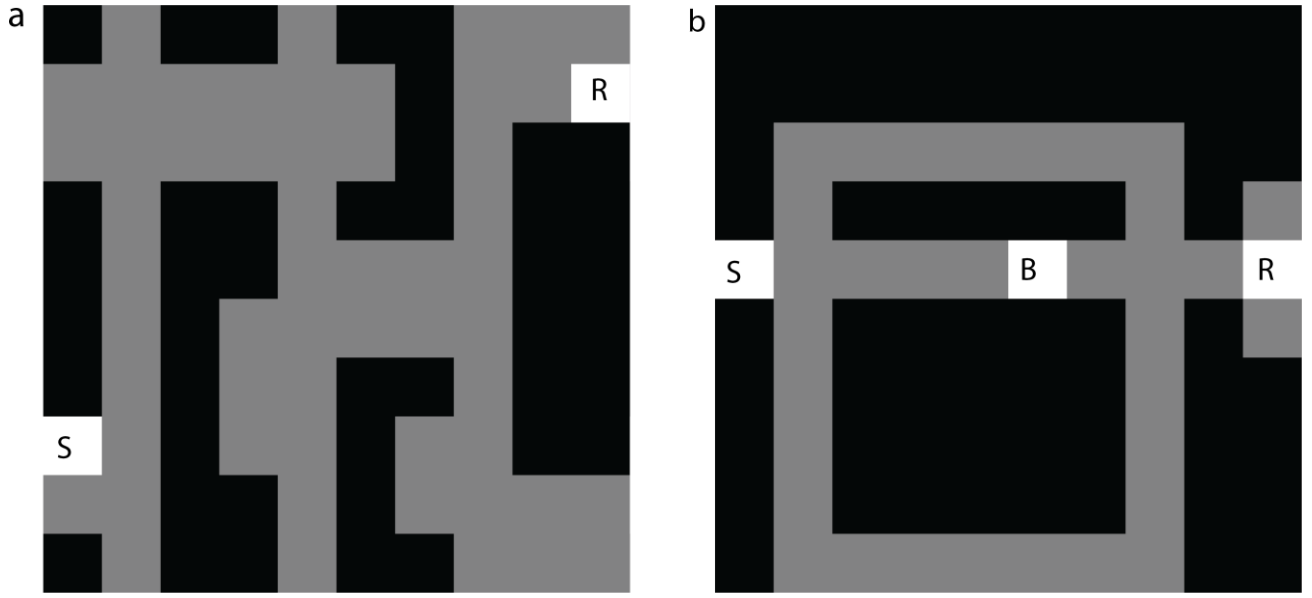



$\gamma$ descibes how much we care about future rewards
$0 \leq \gamma < 1$
0 = maximally short-sighted (only care about next reward)
1 = maximally long-sighted (care equally about future rwd)

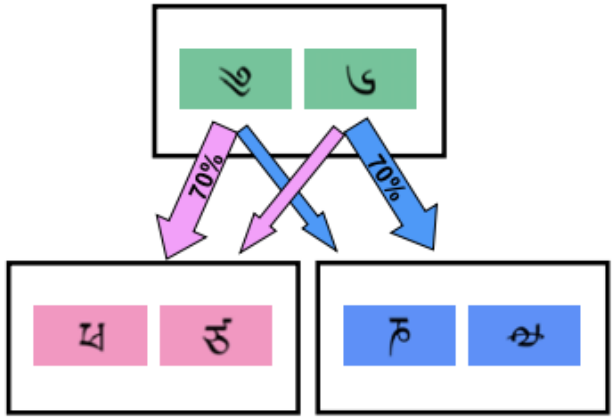
# Two categories of changes requiring adaptation

- Changes in reward structure e.g.
  - Latent learning tasks
  - Reward revaluation tasks

- Contingency change e.g.
  - Contingency degradation
  - Shortcut tasks
  - Detour task

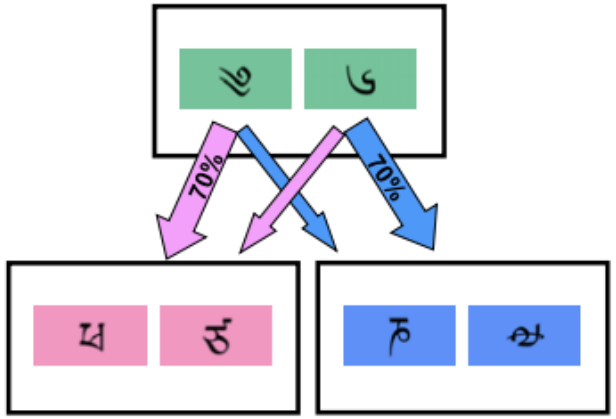


# Model-free RL




- "Stimulus-Response" learner
- Directly learns the value of actions from experience (trial and error)
- Can be learned via a temporal-difference update
- Q-learning update rule after witnessing $(S_t, A_t) \rightarrow (R_{t+1}, S_{t+1})$:
  - $$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[ R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$
- Typically viewed as efficient but inflexible to environmental changes

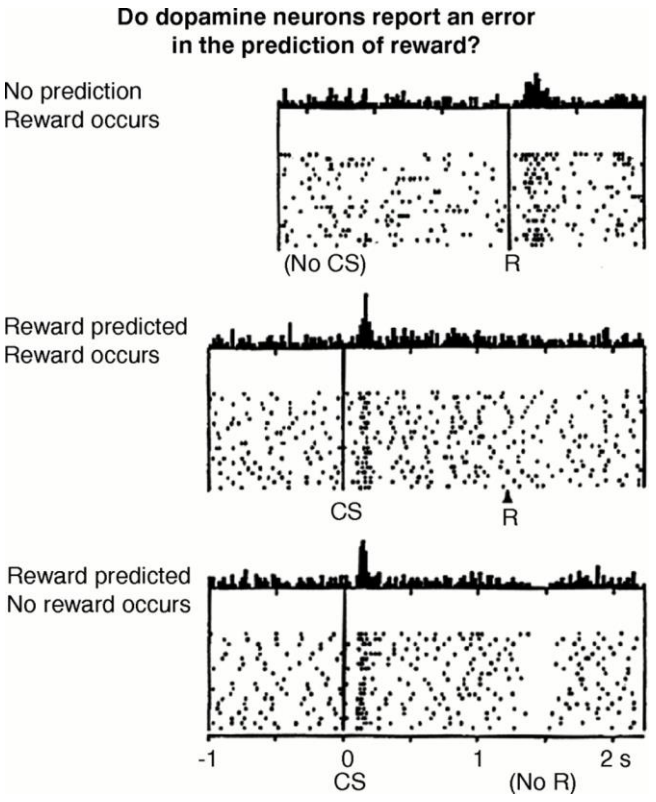
# Model-based RL




- Learns a model of the environment to help guide decision making
- The model might be transition structure between states
- Can then simulate trajectories "off-line" during planning
- Refine the model "online" through experience

# MF and MB RL circuits in the brain?

- Been proposed that model-free and model-based mechanisms might work/compete in parallel in the brain
- Generally midbrain dopamine neurons viewed as coding value prediction error but is typically associated with model-free learning as a way to directly learn long-run value
- Dopamine is not typically associated with model-based learning models because they string together term predictions of the learned model to estimate the best action i.e. if your model-based prediction was bad, you should adjust your model of small-scale transitions rather than long-run expected value
- Essentially they try to push that current model-based RL methods don't have a role for dopamine but rodent lesion data and 2-step task data implicate dopamine with model-based behaviours.



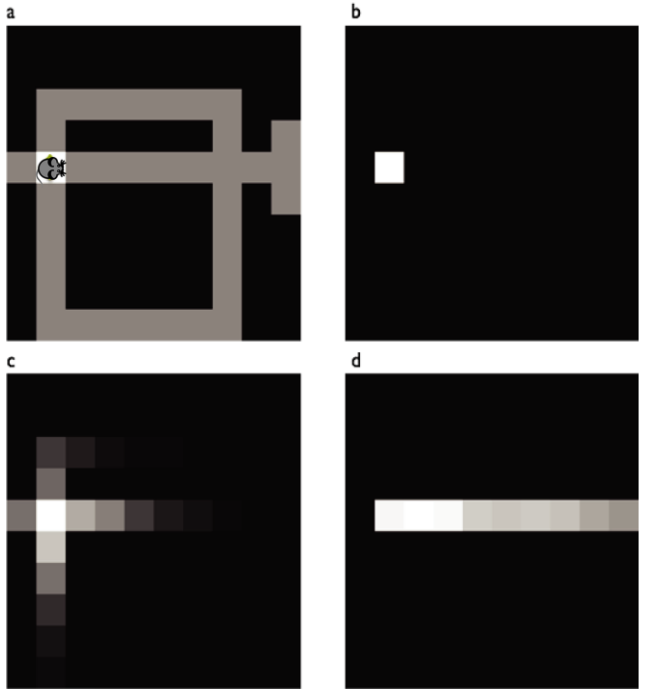
# A Successor Representation (SR)

- Recall that value is expected accumulated reward, discounted over time

- Therefore, if we could learn:

1. expected accumulated occupancy, discounted over time
2. the reward structure of the environment

And combine them together at decision time to estimate value.

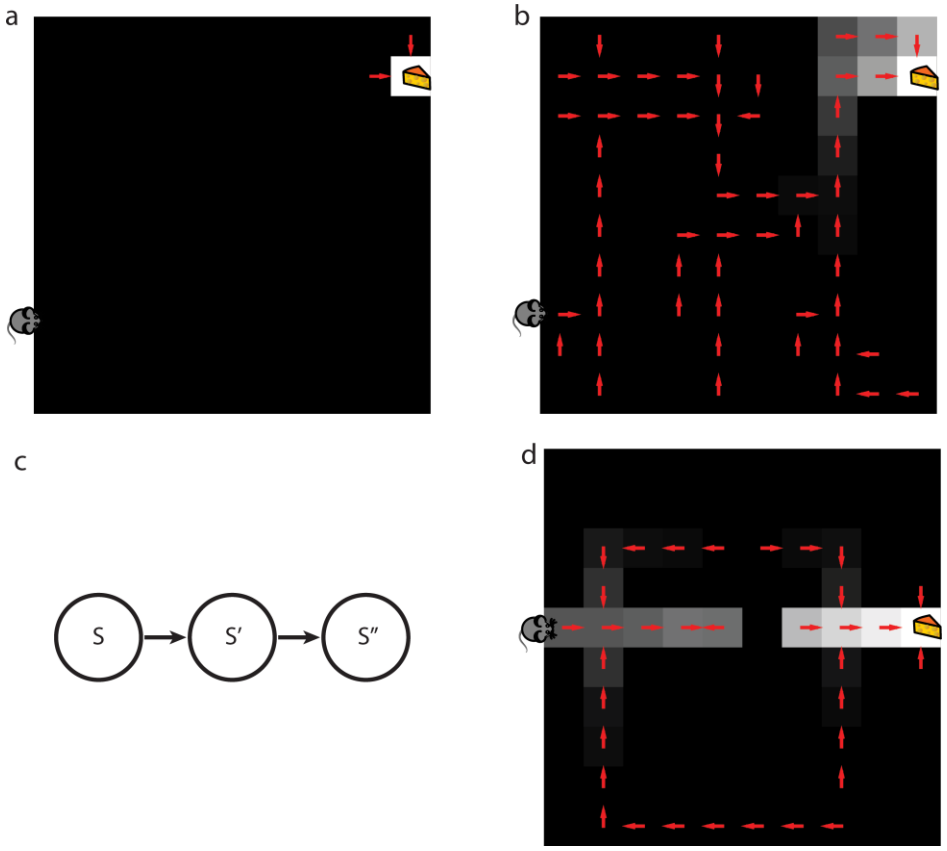



- Expected future occupancy is known as a successor representation

- It is affected by previous behaviour (the policy for how actions are chosen)

# Algorithm 1: SR-TD (temporal-difference)

- Directly estimates the successor representation from experience
- Passes latent-learning task
- Cannot solve detour task
- Evidence* that rodents can solve this detour task




*Tolman EC, Honzik CH. Introduction and removal of reward, and maze performance in rats. Univ Calif Publ Psychol. 1930

# Algorithm 2: SR-MB (model-based)

- Directly estimates the state transition structure "T" and uses this to calculate successor representation "M". Calculation plausible be RNN

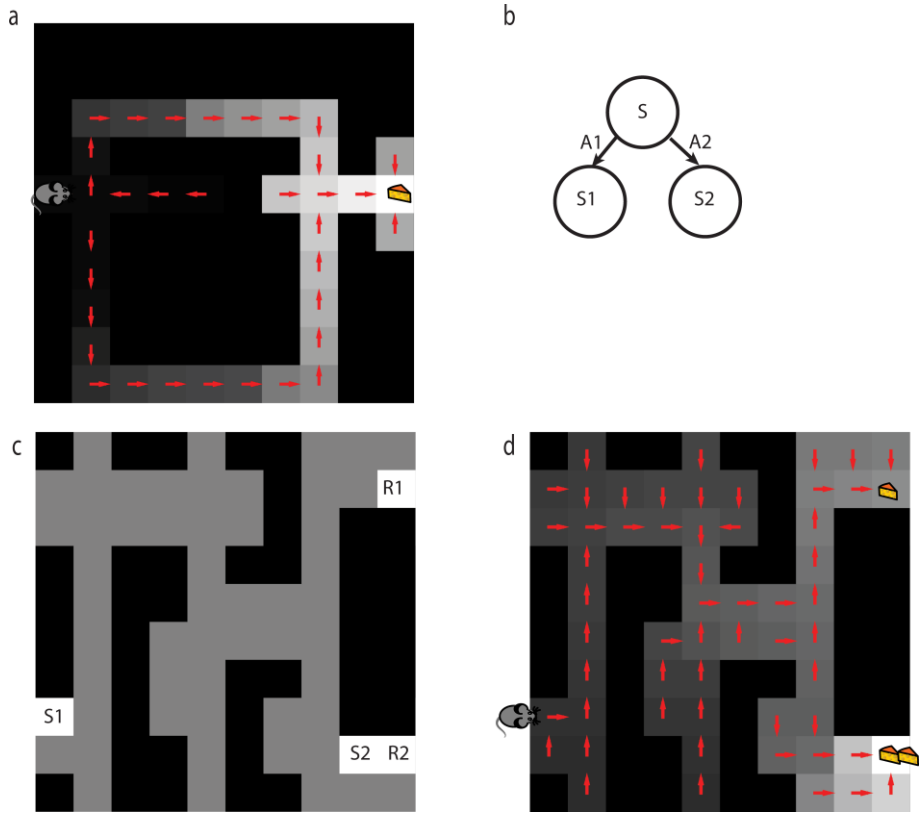
$$M^{\pi}(s,:) = \mathbf{1}_s^{\mathrm{T}} + \gamma T^{\pi}(s,:) + \gamma^2 T^{\pi^2}(s,:) + \gamma^3 T^{\pi^3}(s,:) \ldots = \sum_{n=0}^{\infty} \gamma^n T^{\pi^n}(s,:)$$

- Passes latent-learning task

- Passes detour task

- Is there a task that separates it from "pure" model based?

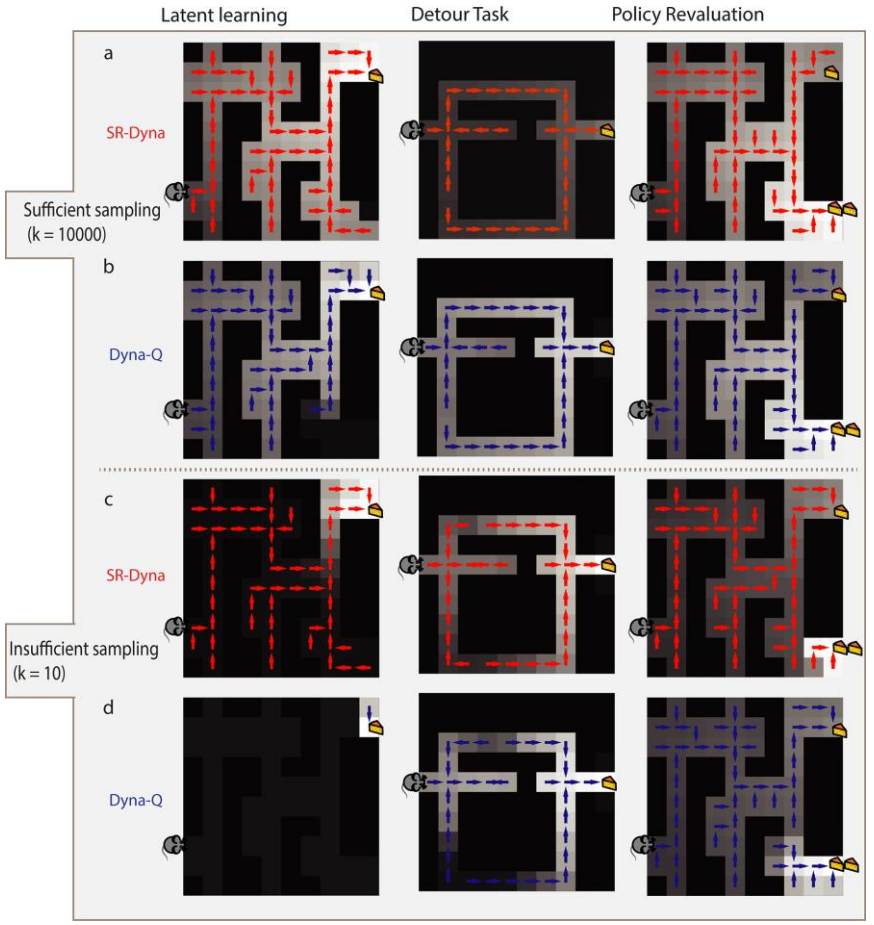
- But this is because it is a "on-policy" method




Which means new learning about changes to the environment can have little to no relevance to the previous learned policy, thus restricting adaptation.

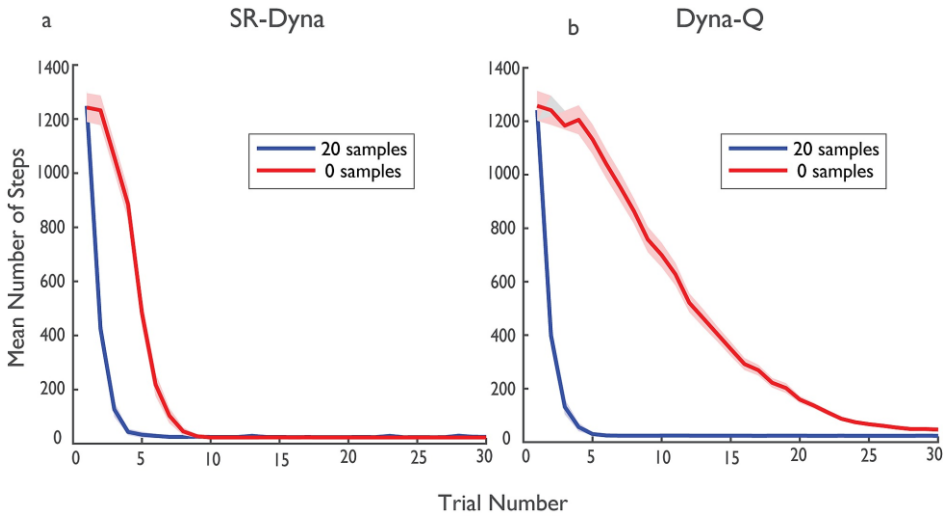
# Algorithm 3: Dyna-SR (uses ”off-policy” replay)

- Previous success has been had by incorporating “replay” of past experiences between actions.
- Could incorporate this into SR
- Experiences are sampled with recency weighted bias
- Naturally performance depends on how many samples are permitted between actions



# Is there a behavioural distinction between Dyna-Q and Dyna-SR?

- Dyna-SR without sufficient replay would reduce to SR-TD (algorithm 1)
- Dyna-Q without sufficient replay would reduce to basic Q-learning (model-free)
- They suggest this could be tested by preventing replay by manipulating rest periods or distractor tasks…



# What does this actually tell us?..

- Gives insight into mechanisms required to model solving real-life tasks
- Interesting in the context of Stachenfeld et al. 2017 ("Hippocampus as Predictive Map", Nature Neuroscience)
- General framework – not restricted to spatial navigation, can be conceptual
- Potentially useful concept for experimenters to know..

# Specific task procedures

Latent learning task.

The latent learning task was simulated in the grid-world environment shown in Fig 2A. Starting from position S, the agent first took 25000 steps exploring the maze. After exploration, the reward in position R1 was raised to 10. To learn about the reward, the agent completed a single step, starting from position R1, 20 times. We then recorded the state value function.

Detour task.

The detour task was simulated using the grid-world environment shown in Fig 2B. Starting from position S, the agent first took 10000 steps exploring the maze. The reward in position R was then increased to 10. The agent then completed 5 trials, starting from position S that ended when the reward was reached. A wall was then added to in position B. To learn about the wall, the agent completed a single step, starting from the position immediately left of the wall, 40 times. We then recorded the state value function.

Novel revaluation task.

The novel revaluation task was simulated using the environment in Fig 5C. The agent first completed the entire latent learning task. After successfully reaching position R1 from position S, the agent then completed 20 trials. Each trial alternately started at S1 or S2 and ended when the agent reached position R1. We then set the reward in position R2 to 20. To learn about the new reward, the agent completed one step, starting from position R2, 20 times. We then recorded the state value function.