RL Tutorial 1

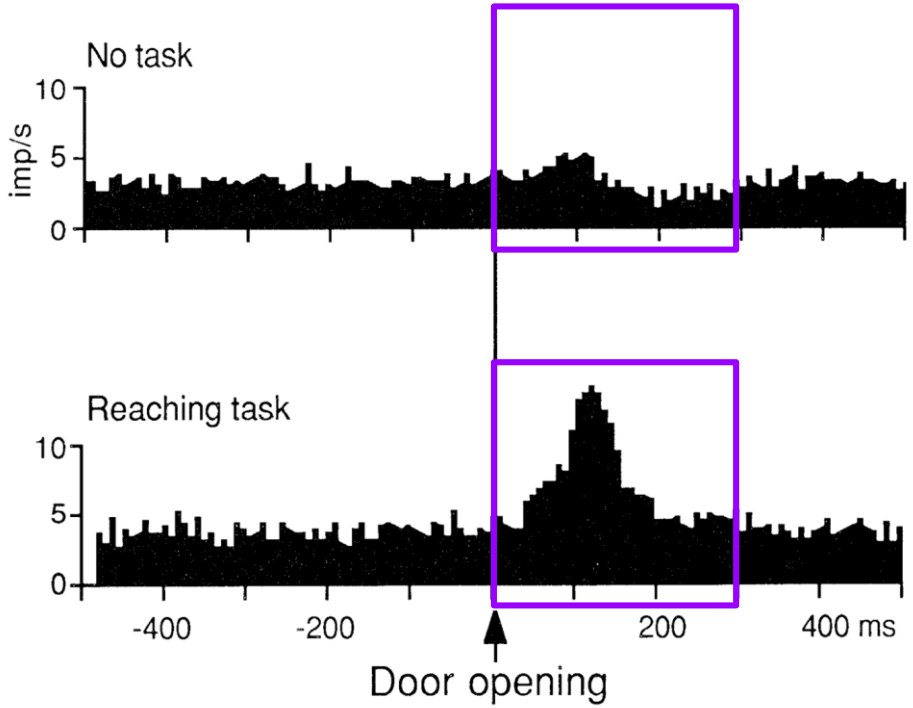
Eric DeWitt

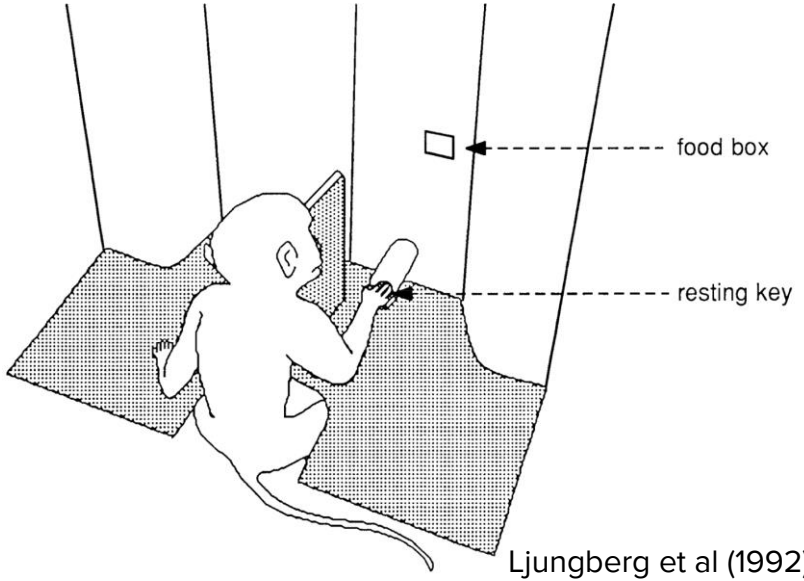
Why is reinforcement learning interesting?

There are very few examples of neural activity that almost exactly matchs a theoretical prediction. (A bit of history...)

- Reinforcement learning has its roots in 1890s with Pavlov's work in physiology and animal learning research of the early 1900s (Thorndike, Hull, Tolman, ...)
- In the late 80's and early 90's Richard Sutton, Andrew Barto and colleagues formalized reinforcement learning, while keeping the close links to neurobiology and behavior. (And they wrote ***the*** book on it...)
- At the same time, Wolfram Schultz laboratory was studying dopamine's role in action and its effects on the motor system, when they noticed dopamine also responded to motivations (rewards) and surprise!

Why is reinforcement learning interesting?



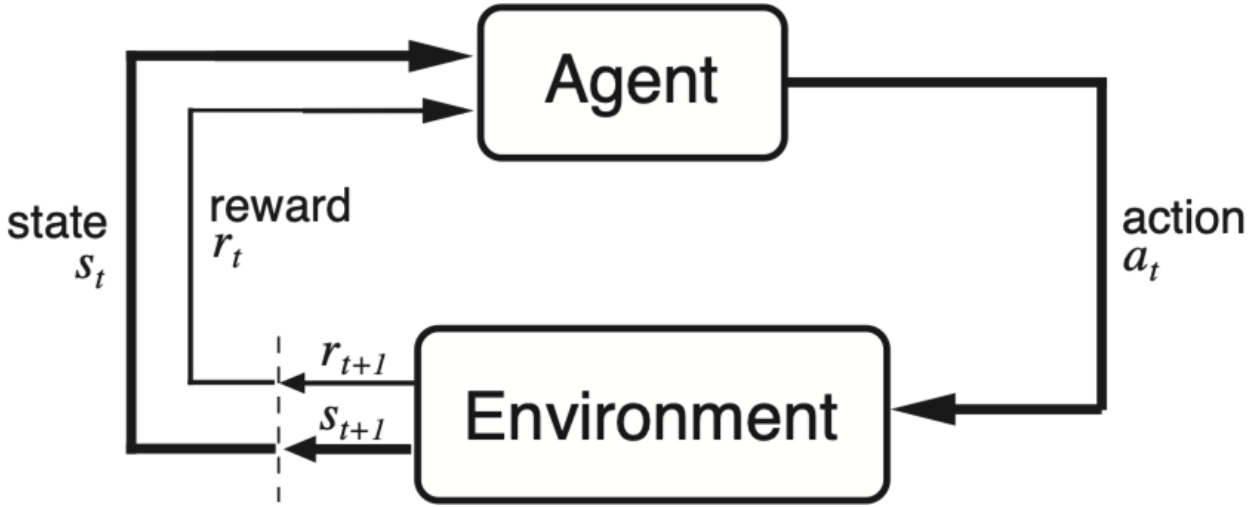


Ljungberg et al (1992)

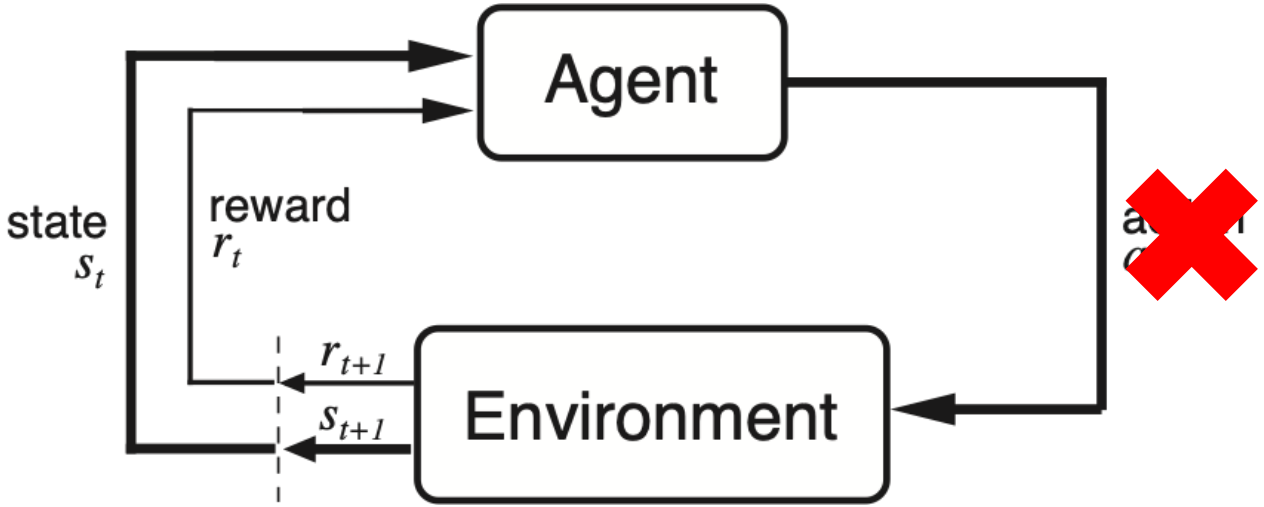
Why is reinforcement learning interesting?

- Also in the late 90's, Peter Dayan and P Read Montague first successfully modeled the honey bee's octopamine neuron during classical conditioning as a reward prediction error
- They realized that the dopamine activity observed by Schultz was also consistent with a reward prediction error
- Since then, thousands of papers have confirmed the (basic) relationship between reward prediction error and dopamine activity
- Recent work, of course, has provided evidence that the story is more complicated, but this long connection between between behavior, neural activity and reinforcement learning has proved immensely powerful

Dopamine and classical conditioning



Dopamine and classical conditioning

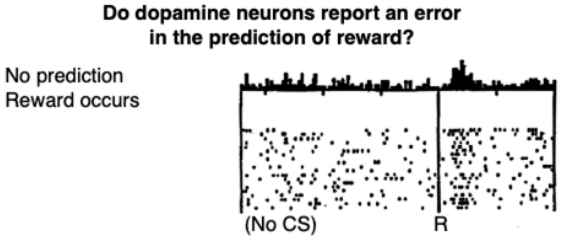


Dopamine and classical conditioning

$$V(S_t) \leftarrow V(S_t) + \alpha\left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right]$$

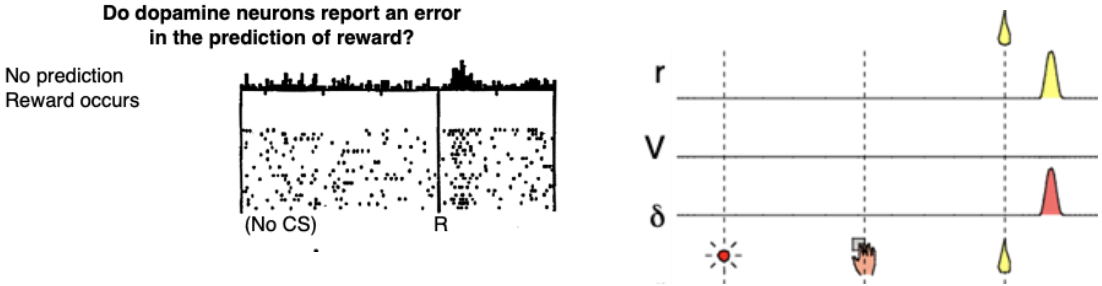
Reward Prediction Error

Dopamine and classical conditioning



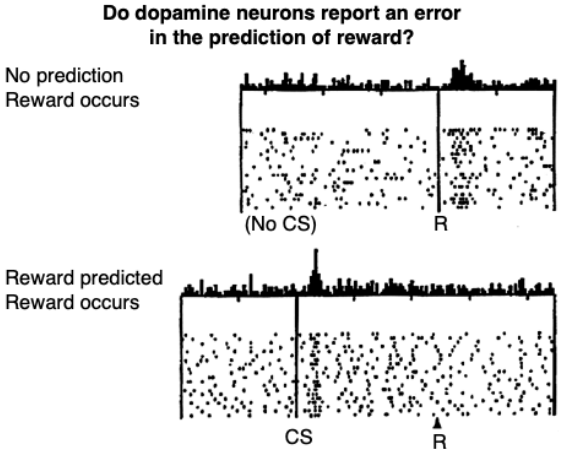
Schultz et al (1997)

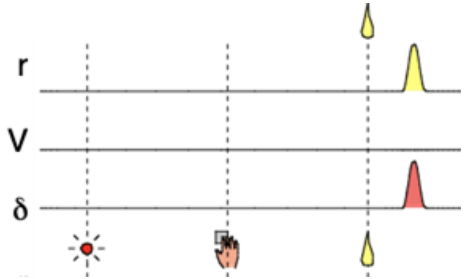
Dopamine and classical conditioning



Schultz et al (1997)

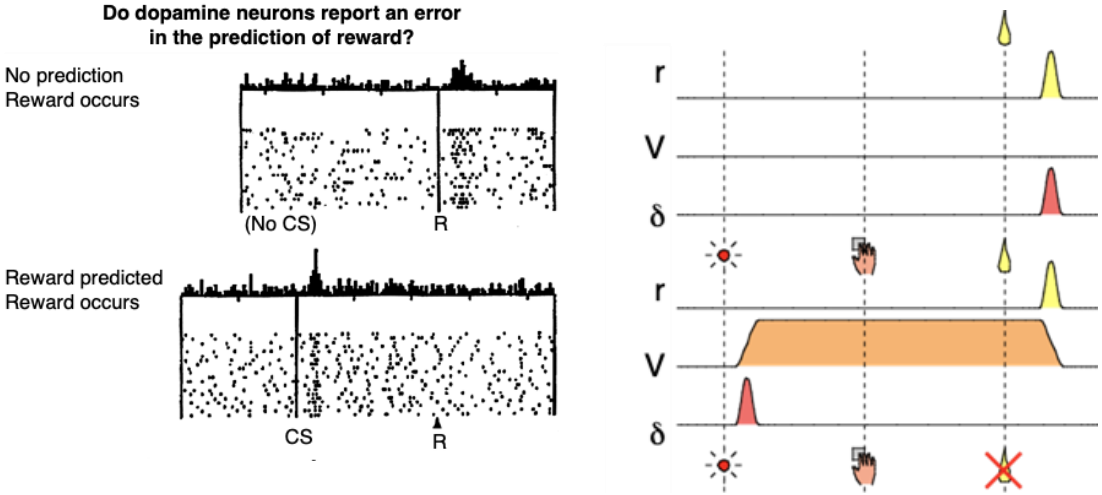
Dopamine and classical conditioning





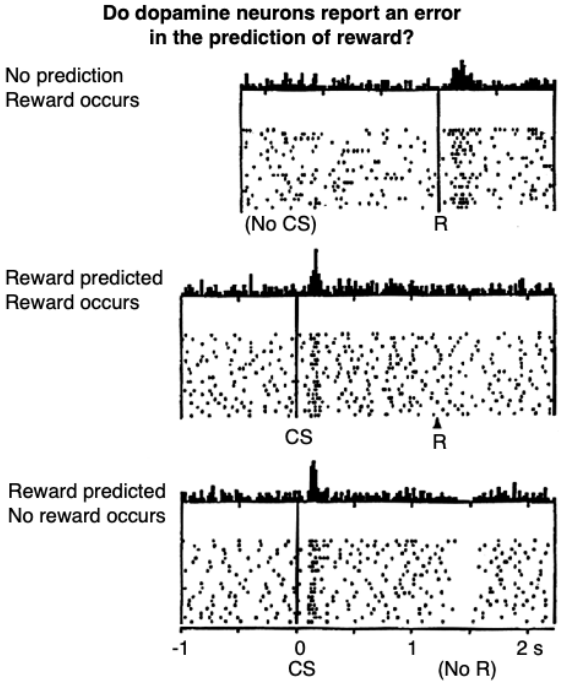
Schultz et al (1997)

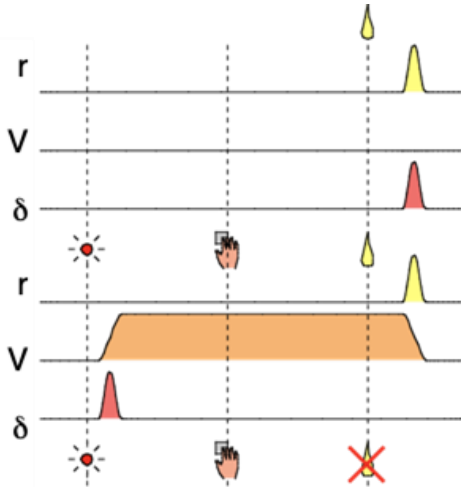
Dopamine and classical conditioning



Schultz et al (1997)

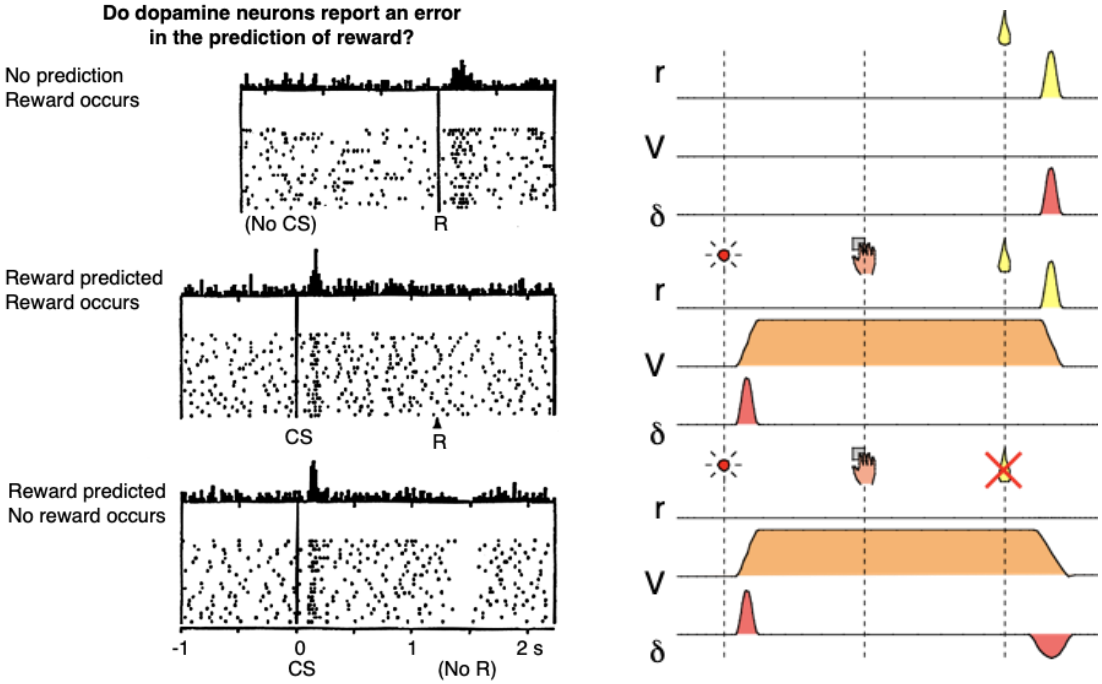
Dopamine and classical conditioning





Schultz et al (1997)

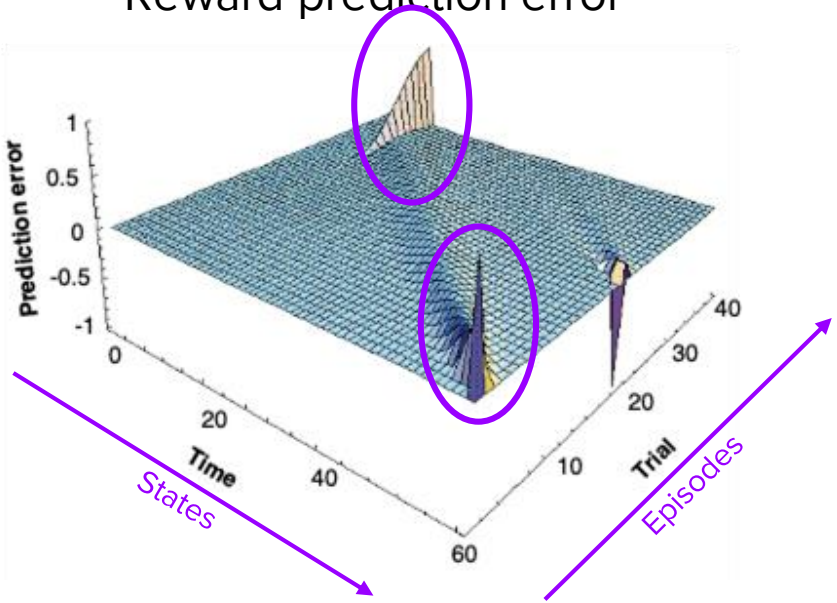
Dopamine and classical conditioning



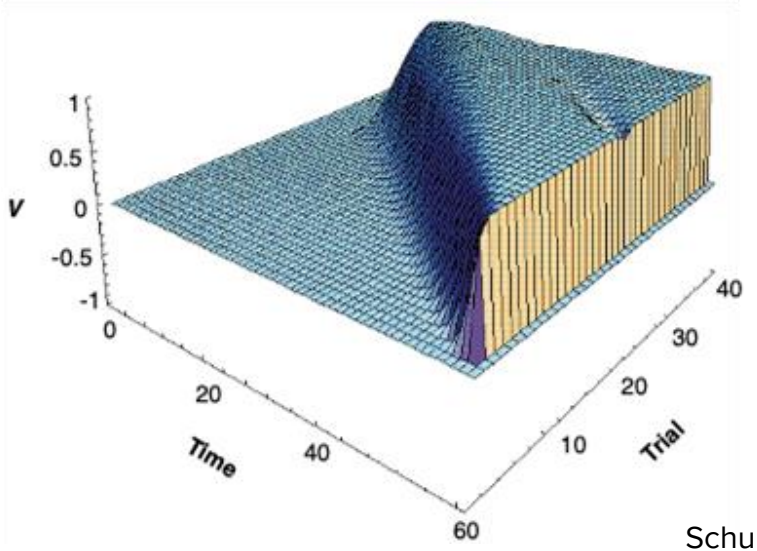
Schultz et al (1997)

Dopamine and classical conditioning

Reward prediction error

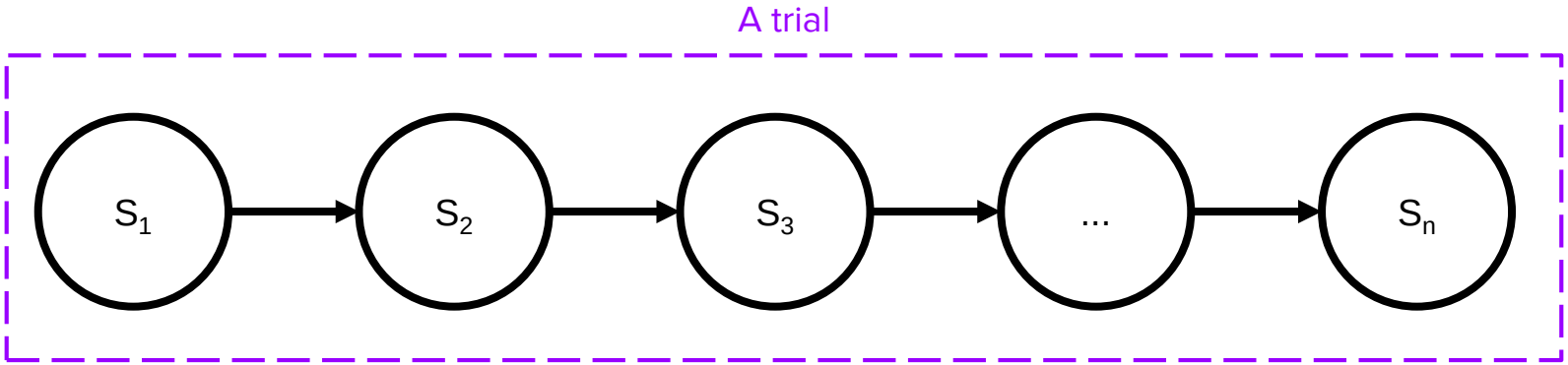


Value function



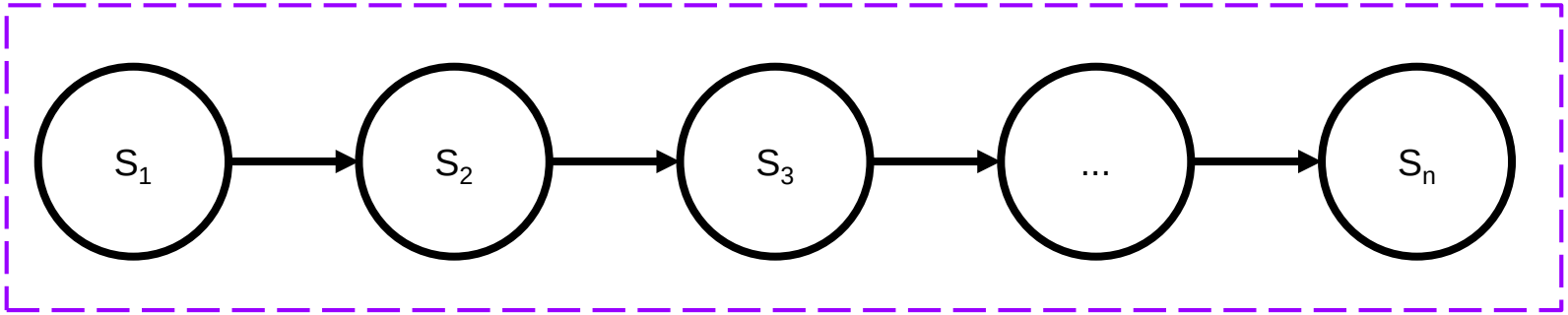
Schultz et al (1997)

Tapped delay line model

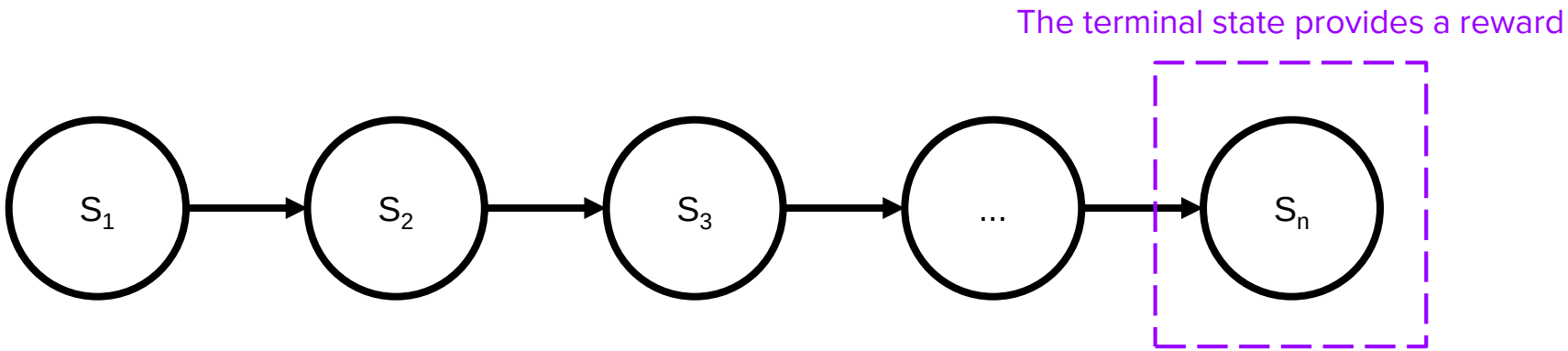


Tapped delay line model

Cue and time are combined into a series of states



Tapped delay line model



Tapped delay line model (python class)

Run the following code for your implementation:

```
class classical_conditioning():

    def __init__(self, n_steps):

        # Task variables
        self.n_steps = n_steps
        self.n_actions = 0

        # Reward variables
        self.reward_state = [0,0]
        self.reward_magnitude = reward_magnitude
        self.reward_probability = reward_probability
        self.reward_time = reward_time

        # Time step at which the conditioned stimulus is presented
        self.cs_time = int(n_steps/4) - 1

        # Create a state dictionary
        self.create_state_dictionary()

    def define_reward(self, reward_magnitude, reward_time):

        """
```

Exercise 1

You will use the classical conditioning class to:

- Understand the tapped delay line representation
- Observe changes in the value function
- Observe the predicted reward prediction error
- Replicate and understand the basic biological observations of dopamine

RL Tutorial 1 - Part 2

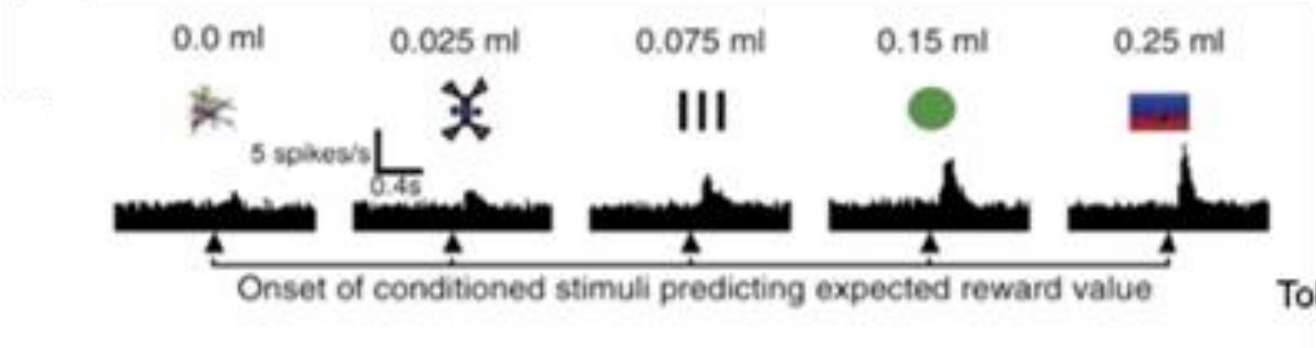
Eric DeWitt

What happens when rewards vary?

You have just seen what happens when we model the dopamine responses when a reward always occurs at the same time with the same magnitude. What happens when this is not true?

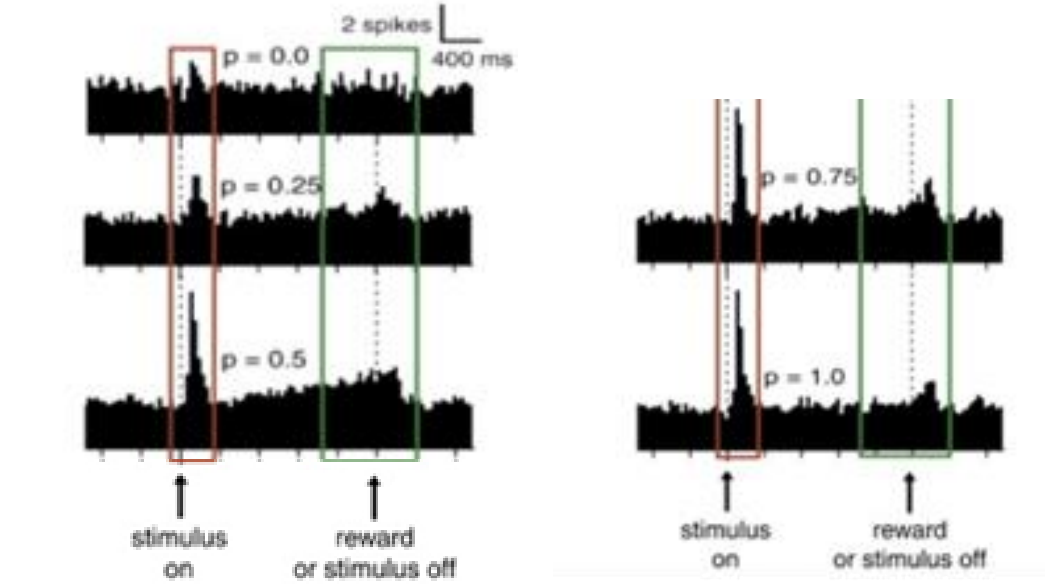
- Rewards can vary in magnitude
- Rewards can vary in probability
- Rewards can vary in the time they arrive following a cue

Dopamine activity to consider (magnitude)



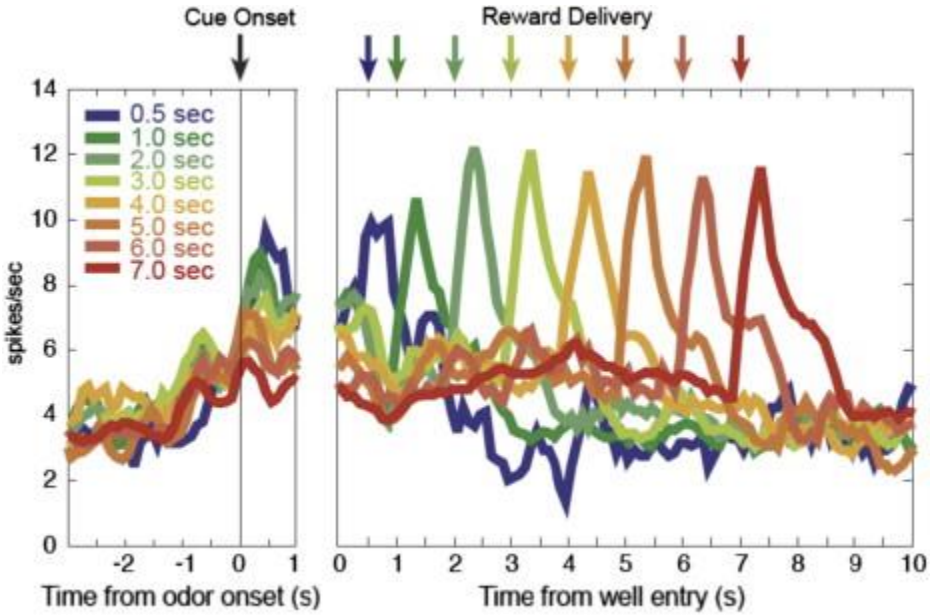
Tobler et al., 2005

Dopamine activity to consider (probability)



Fiorillo et al., 2003

Dopamine activity to consider (timing)



Roesch et al., 2007

Exercise 2 and 3

You will now explore some of these biological responses in the classical conditioning model.

- Exercise 2 explores changing the magnitude of reward
- Exercise 3 explores changing the probability of reward

There are some advanced exercises if you want to explore more!

RL Tutorial 2

Eric DeWitt

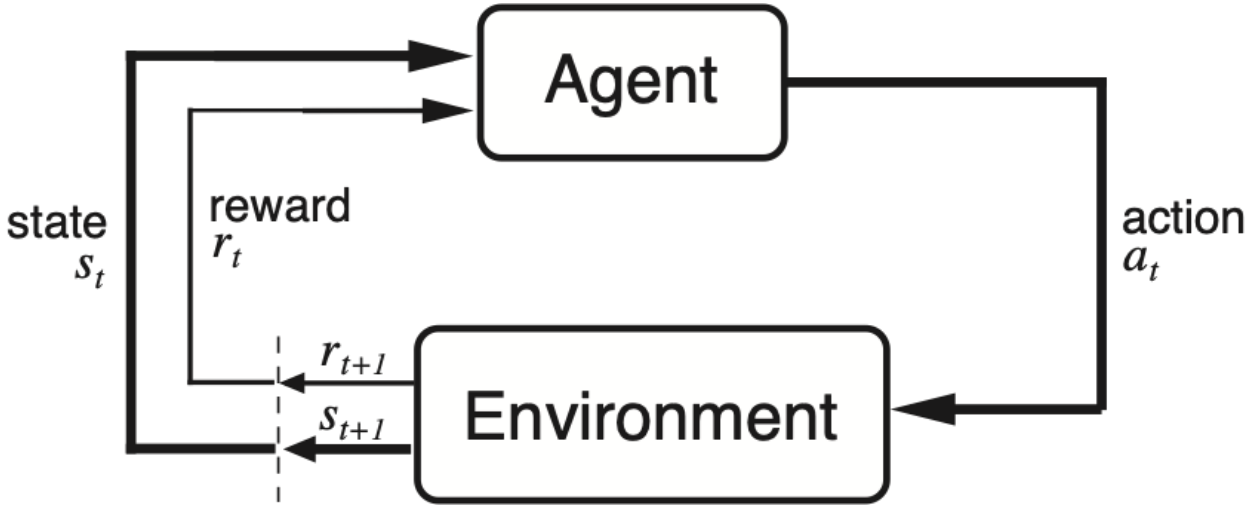
How do RL agents learn to act?

You should have an intuition from Tutorial 1 for how dopamine might implement a reward prediction error to learn future expected value.

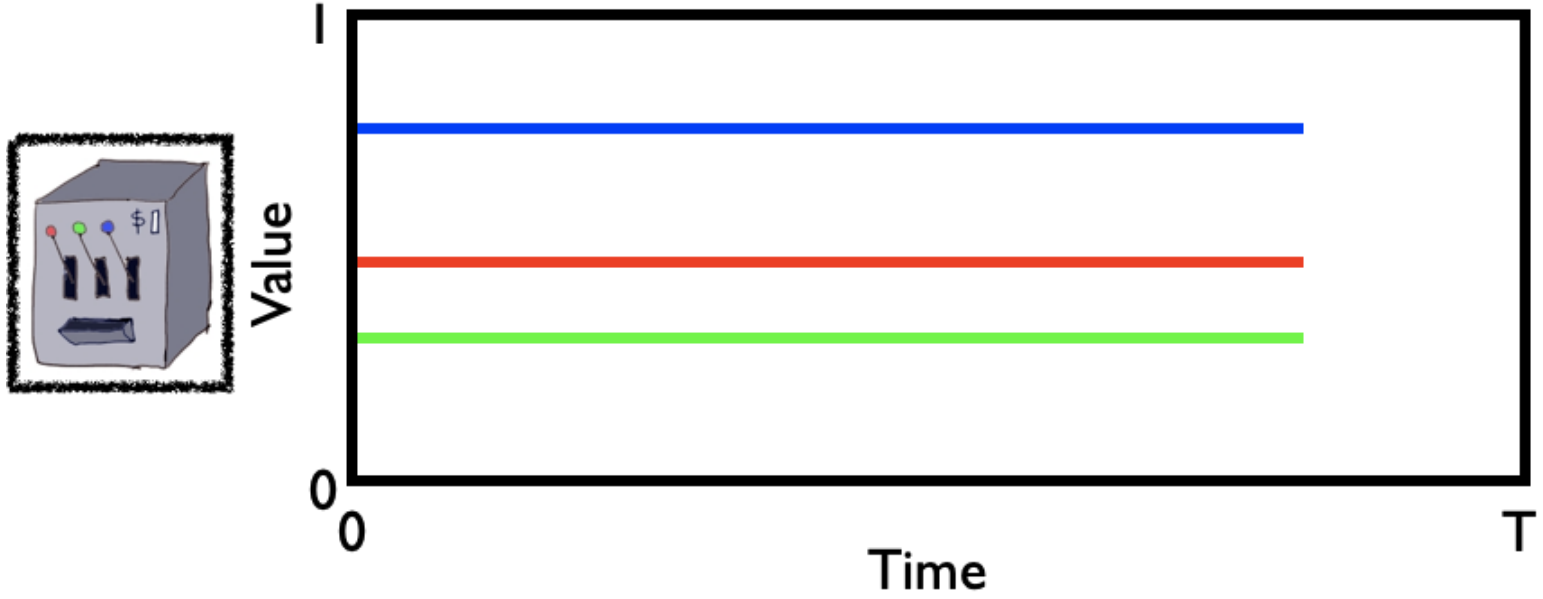
But how does the brain use expected values to learn ***how*** to act?

- A policy is the formal description of how an agent chooses an action based on the observed state of the world
- A fundamental problem with learning how to act is exploration vs exploitation
- When you have states and actions, you need to use a representation of value that informs the policy—or an action-value representation—if you want both the value function and the actions to produce the optimal behavior

Why is reinforcement learning interesting?



The n-armed bandit: explore or exploit?

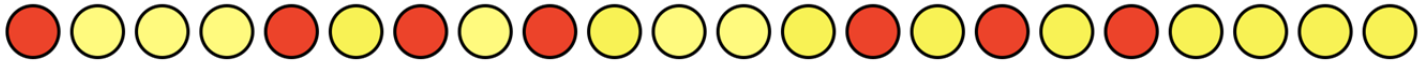


The n-armed bandit: explore or exploit?

Explore then exploit

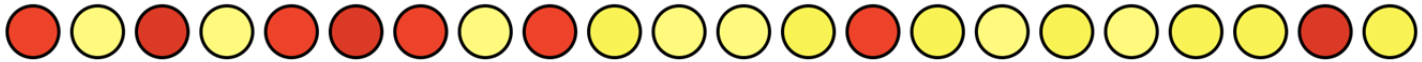
...

ε-Greedy



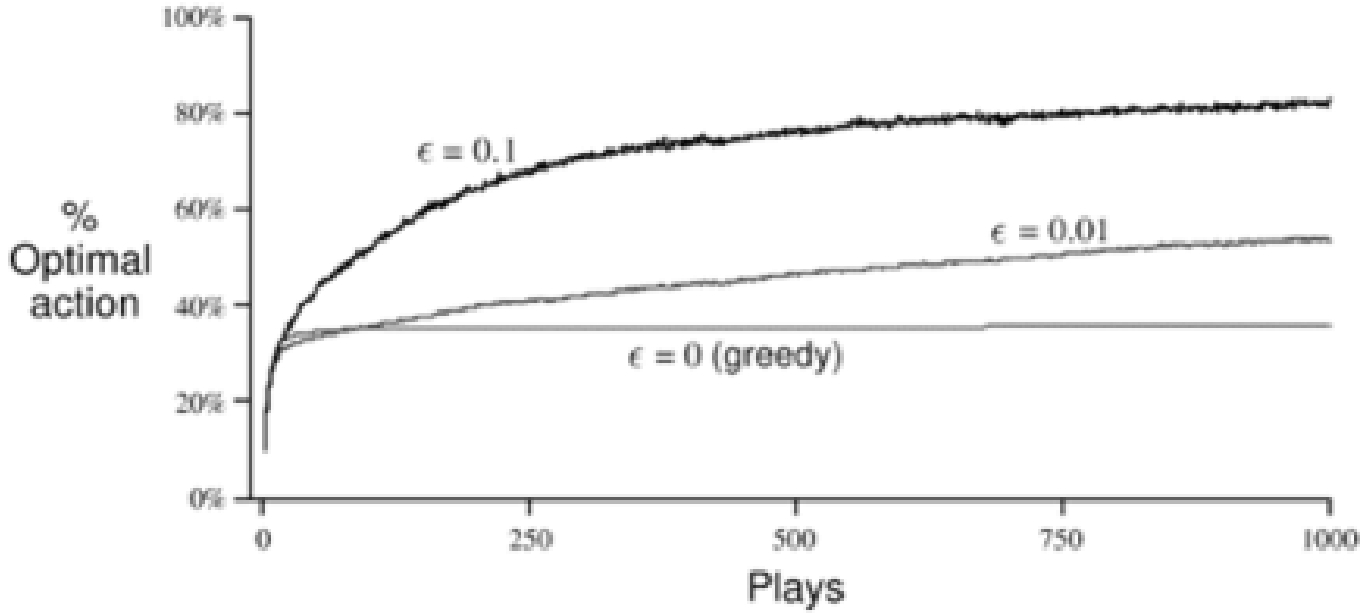
...

ε-Greedy with decaying **ε**

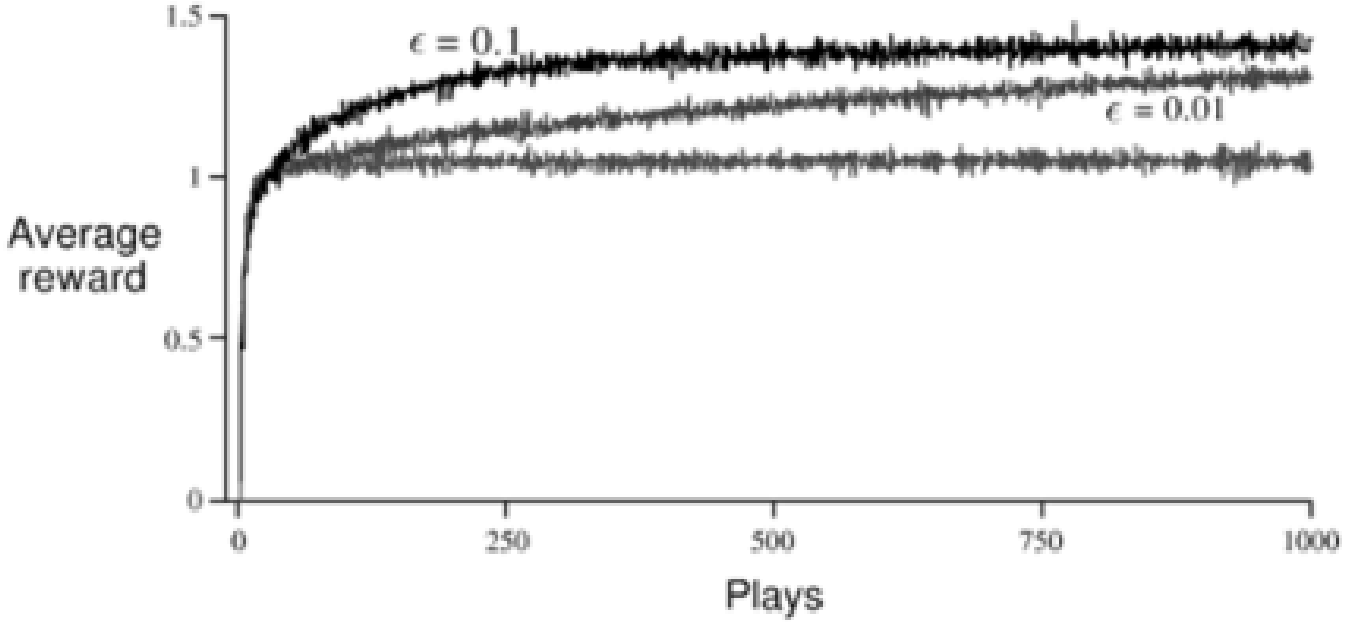


...

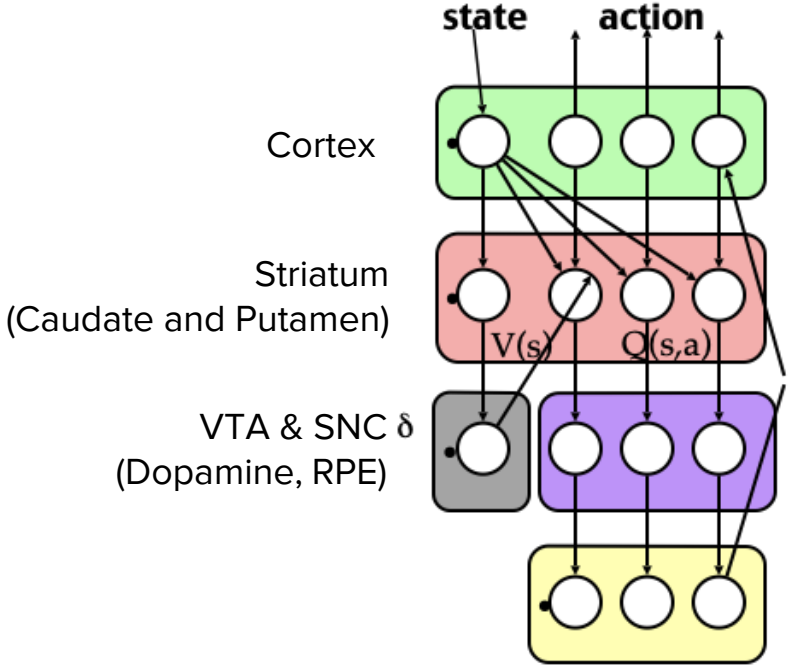
The n-armed bandit: explore or exploit?



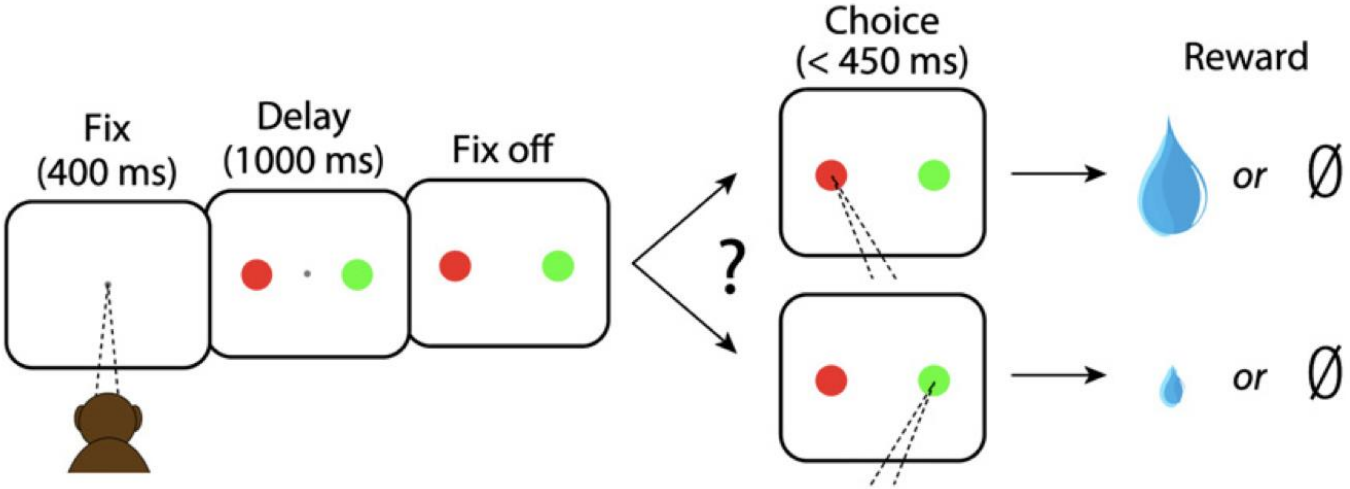
The n-armed bandit: explore or exploit?



Action values in biology

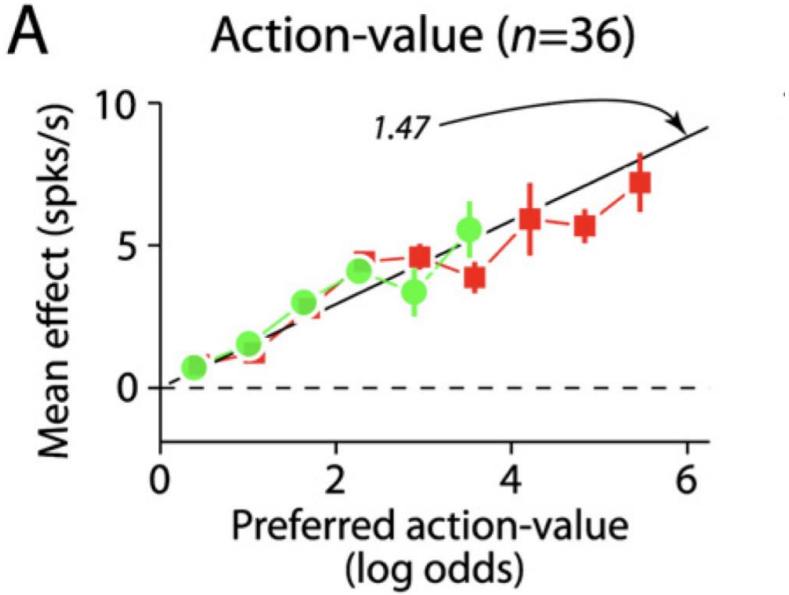


Action values in biology



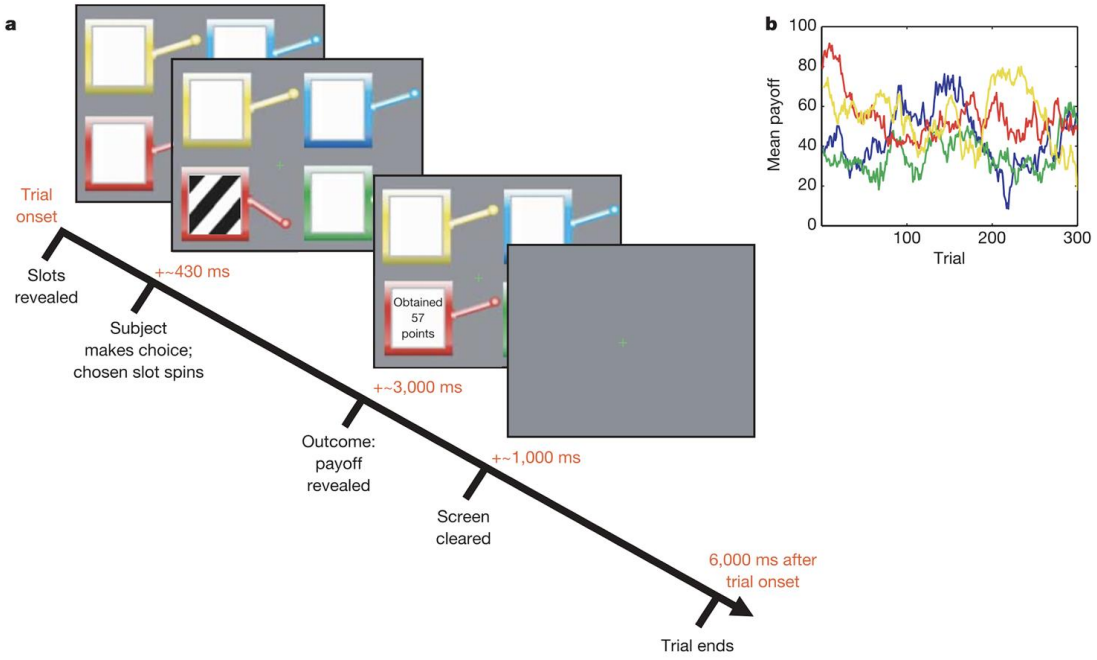
Lau & Glimcher (2008)

Action values in biology



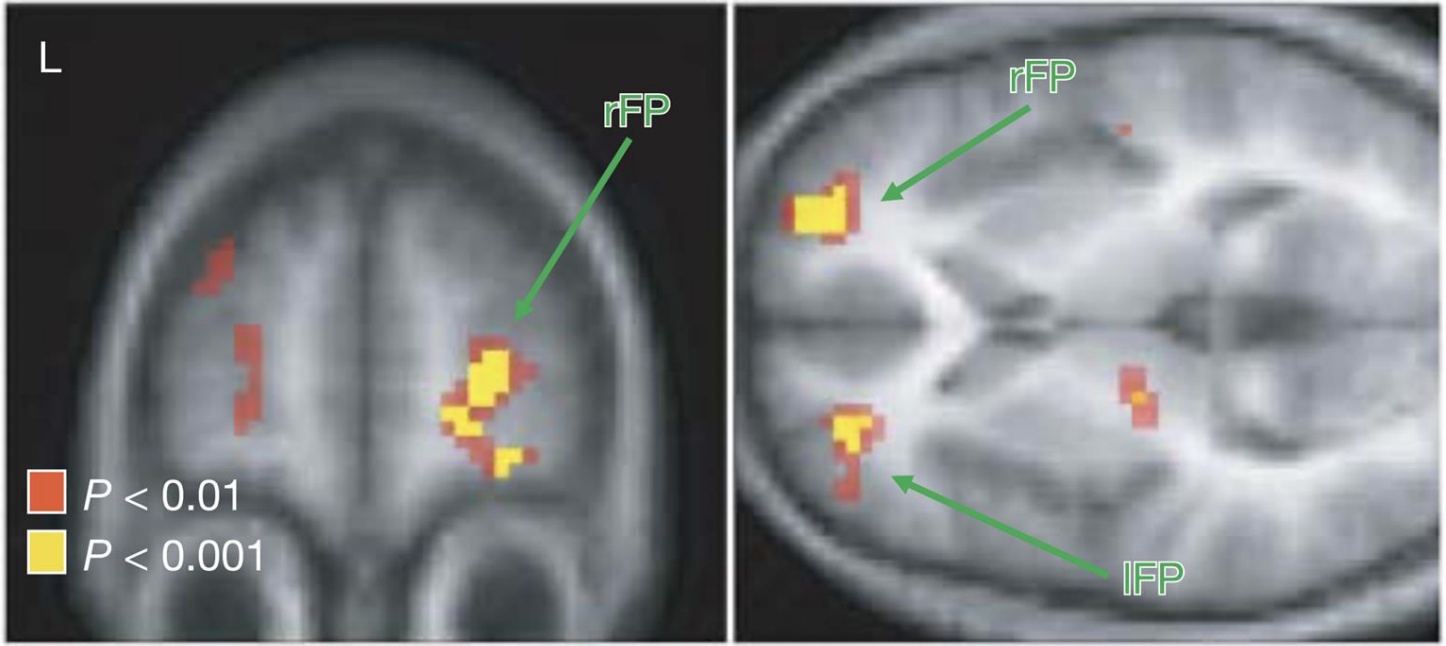
Lau & Glimcher (2008)

The n-armed bandit: explore or exploit?



Daw et al (2006)

The n-armed bandit: explore or exploit?



Daw et al (2006)

Exercises 1, 2 and 3

You will use n-armed bandits study action values representations and to build an intuition for the exploration exploitation trade-off.

- In Exercise 1 you will implement $\boldsymbol{\varepsilon}$-Greedy
- In Exercise 2 you will implement an action-value update
- In Exercise 3 you will explore how different values of the ‘exploration’ parameter ($\boldsymbol{\varepsilon}$) and the learning rate ($\boldsymbol{\alpha}$) effect how well the agents learns

RL Tutorial 3

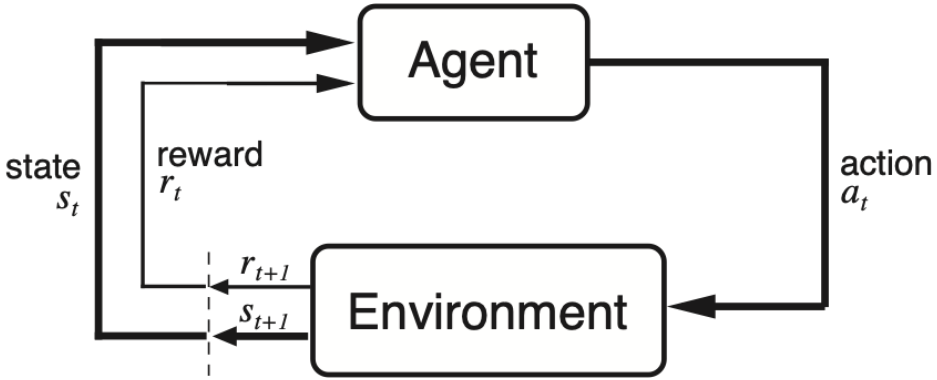
Marcelo Mattar

How do reinforcement learning agents act?

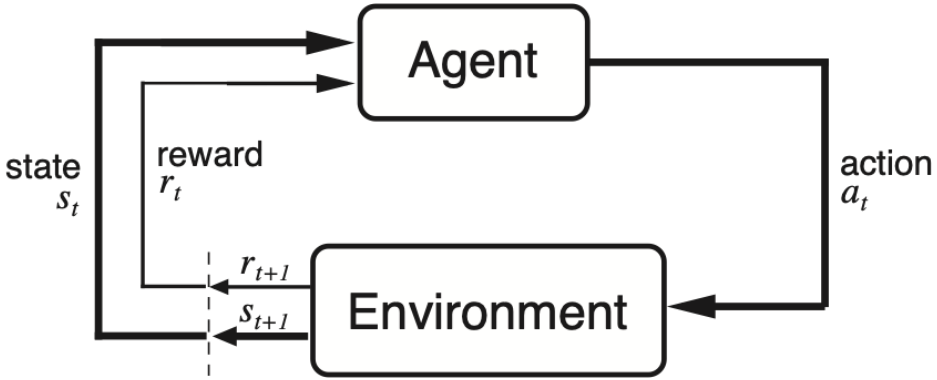
In the previous tutorial, we saw how agents learn to act in the bandit setting. In this tutorial, we will see how agents learn to act in the more general setting of Markov Decision Processes (MDPs).

- In bandits, the agent has no control over the state of the world.
- In MDPs, the agent's actions may influence the future states of the world

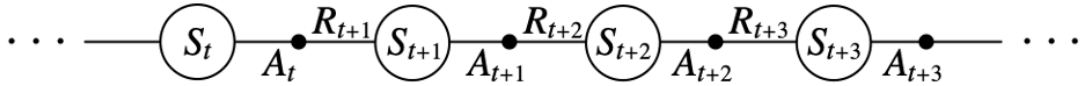
Why is reinforcement learning interesting?



Why is reinforcement learning interesting?



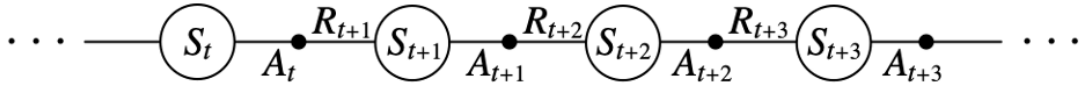
Choosing actions in MDPs



REMINDER: How are action values computed in bandit (one state) setting?

$$q(a) = \mathbb{E}\left[R_t \mid A_t = a\right]$$

Choosing actions in MDPs



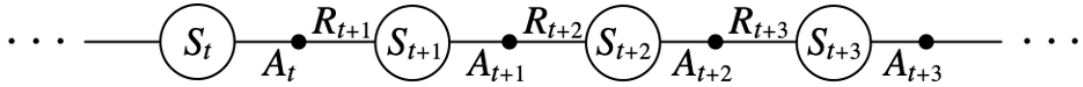
REMINDER: How are action values computed in bandit (one state) setting?

$$q(a) = \mathbb{E}\left[R_t \mid A_t = a\right]$$

NOW: How are action values computed in MDPs?

$$q(s, a) = \mathbb{E}\left[\sum_{k=1}^{\infty} R_{t+k} \mid S_t = s, A_t = a\right]$$

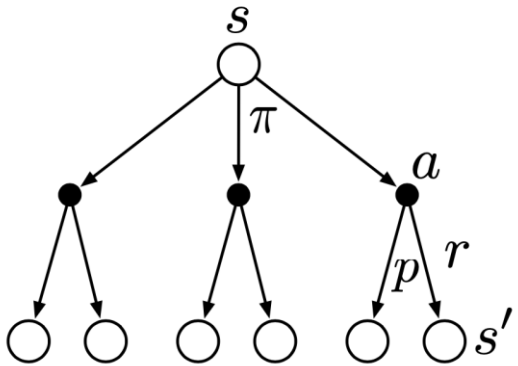
Computing action values



PROBLEM: Computing future reward can be difficult!

SOLUTION: Bellman optimality equation:

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \;\middle|\; S_t = s, A_t = a\right]$$



(i.e., use the value of the next state/action as a stand-in for future rewards)

Learning (optimal) action values: Q-learning

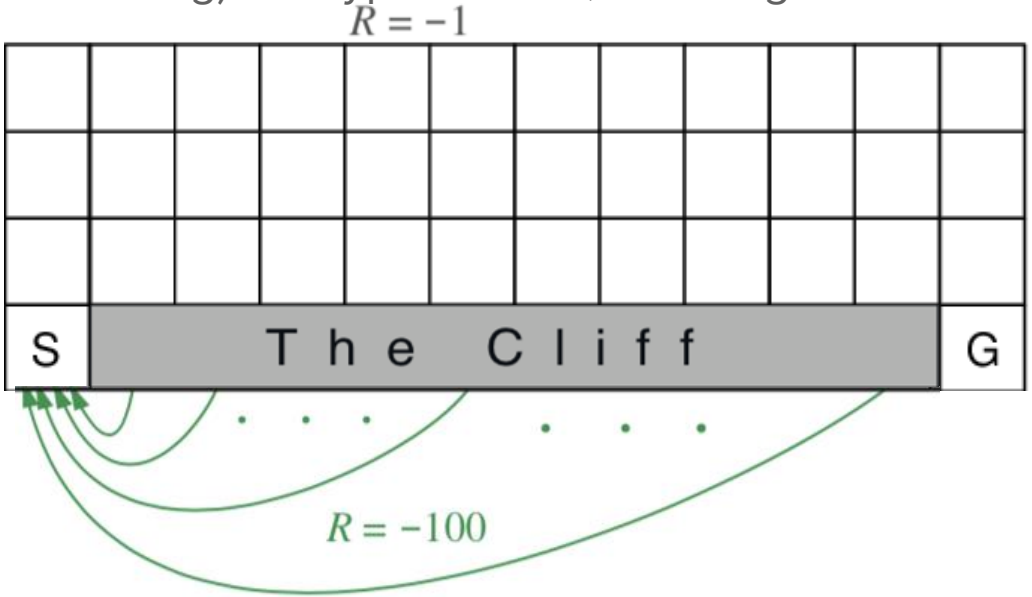
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma \max_a Q(S', a) - Q(S, A) \right]$
    $S \leftarrow S'$
  until S is terminal

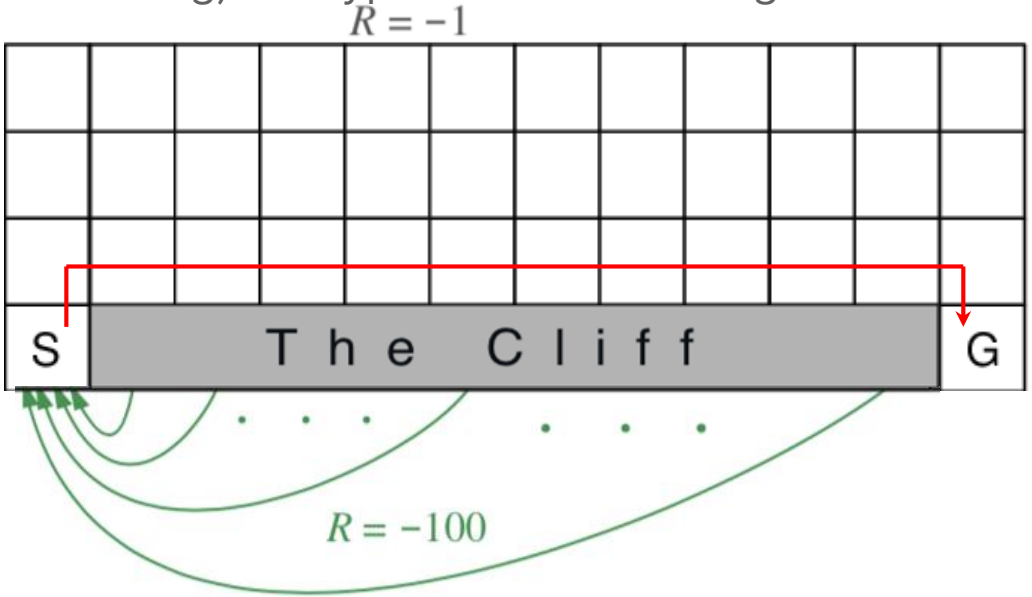
Exercise

In this exercise, you will implement an RL agent that learns how to act (using Q-learning) in a type of MDP, called "grid-world"



Exercise

In this exercise, you will implement an RL agent that learns how to act (using Q-learning) in a type of MDP called "grid-world"



On-policy vs. off-policy learning

When estimating the value of future states/actions:

- Off-policy methods estimate values that are ***not*** based on the agent's current policy. Most often, values are estimated under the assumption of optimal future behavior (i.e. optimal policy).
- Q-learning is an example of off-policy learning algorithm.

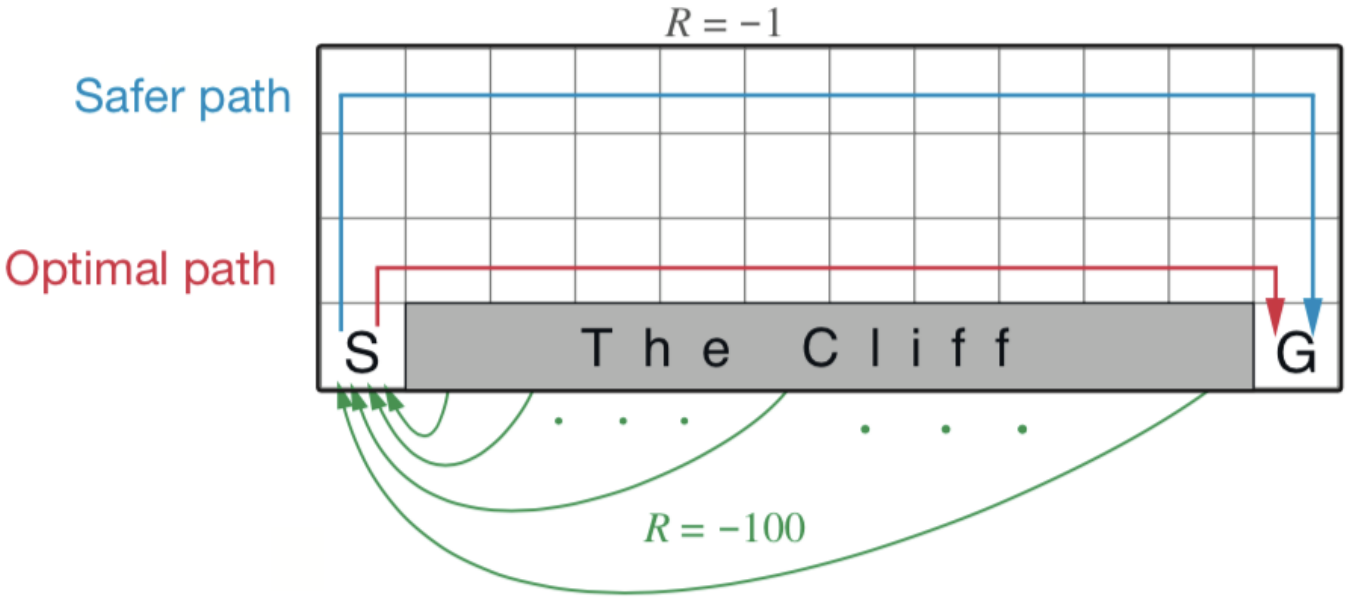
- On-policy methods estimate values under the agent's current policy, including possible randomness/stochasticity in future behavior (e.g. ε-greedy)
- SARSA is an example of on-policy learning algorithm

Learning (practical) action values: SARSA

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Initialize $Q(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(\textit{terminal-state}, \cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q(S',A') - Q(S,A)\right]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

Case study: accounting for future exploration



RL Tutorial 4

Marcelo Mattar

Learning vs. planning

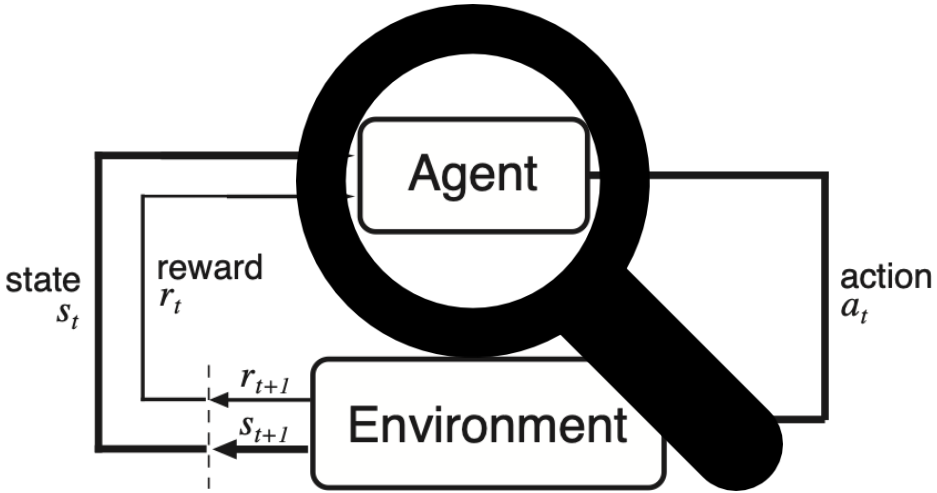
- So far, we've seen how agents learn values by acting in the world and *experiencing* the outcome of their actions
- Methods such as Q-learning are ***model-free*** as they do not require a model

- In this tutorial, we will learn about ***model-based*** methods, which compute action values via planning.
- Instead of learning values from experience, planning is the process of computing action values from a model.

But what is a model?

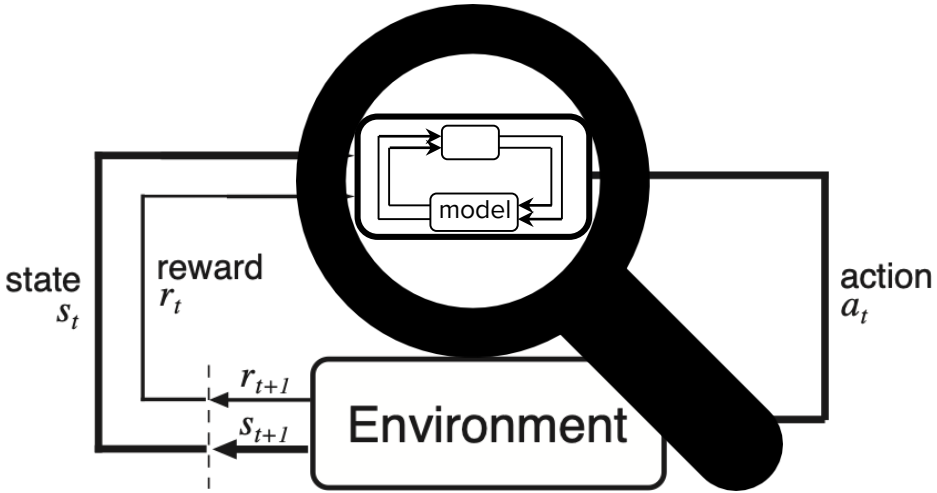
A representation of how the world will respond to the agent's actions

But what is a model?



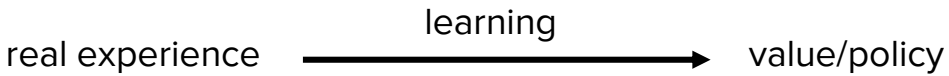
But what is a model?

Given a state and an action, a model returns the next state and next reward

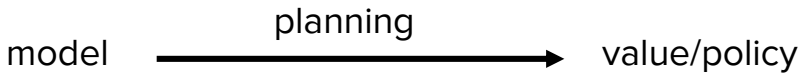


Model-free and Model-based RL

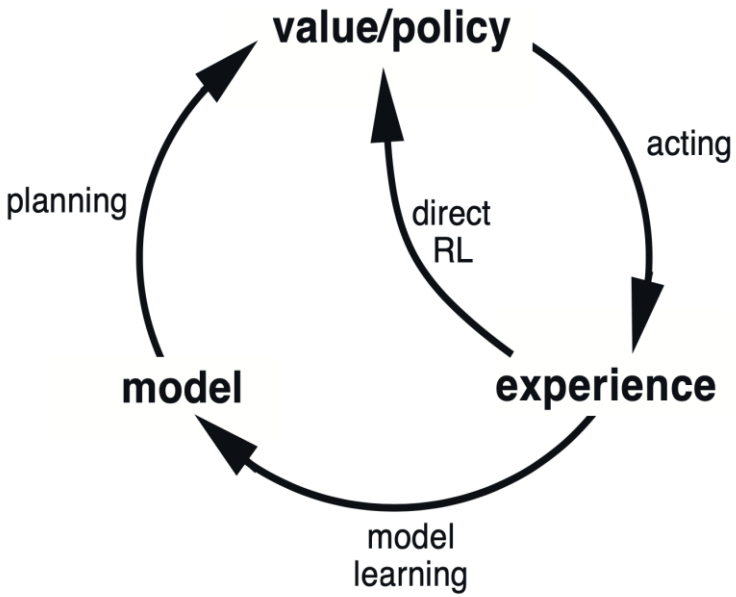
- Model-free RL:



- Model-based RL:



Integrating planning and learning



Dyna-Q architecture

Tabular Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

Loop forever:

(a) $S \leftarrow$ current (nonterminal) state

(b) $A \leftarrow \varepsilon$-greedy(S,Q)

(c) Take action A; observe resultant reward, R, and state, S'

(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)

(f) Loop repeat n times:

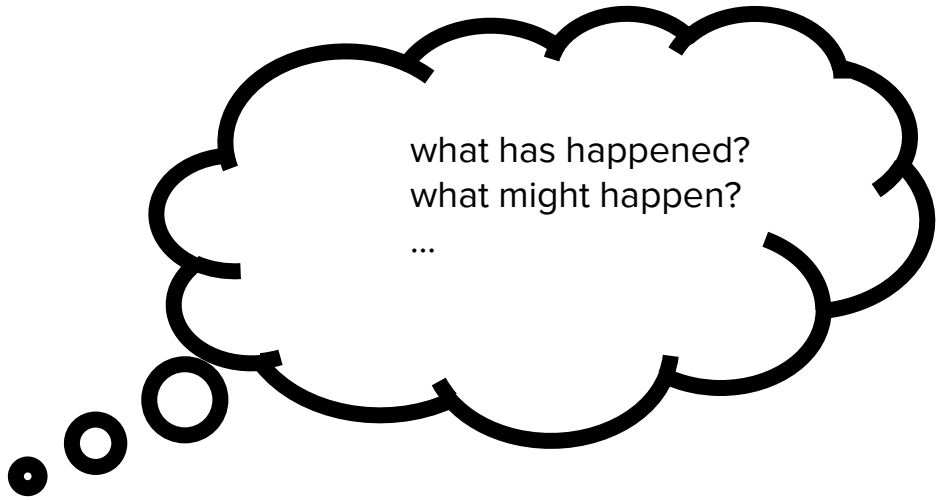
$S \leftarrow$ random previously observed state

$A \leftarrow$ random action previously taken in S

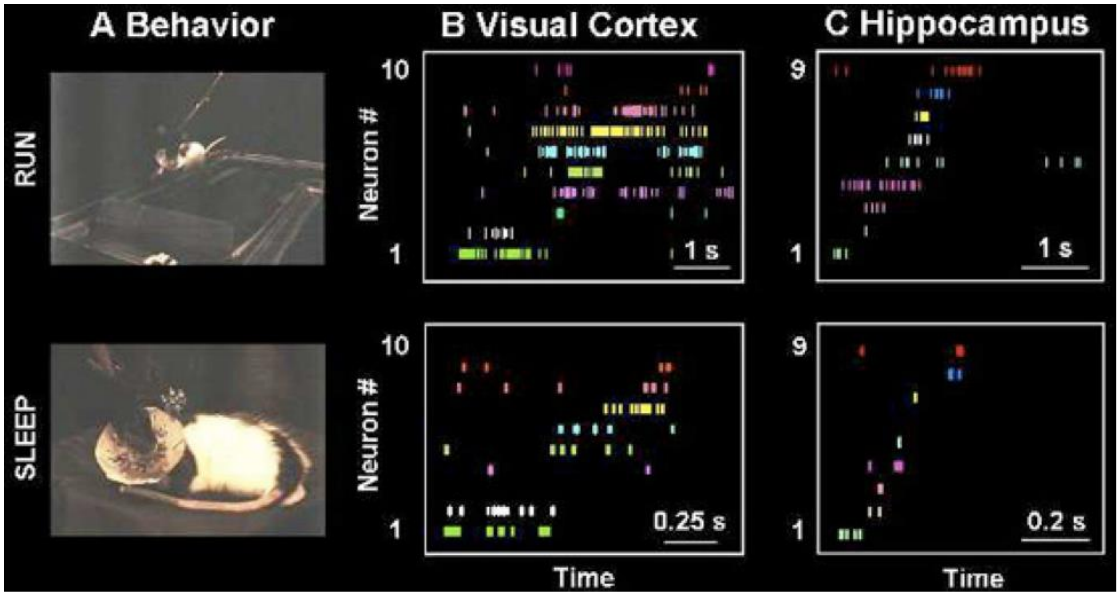
$R, S' \leftarrow Model(S,A)$

$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

A mathematical framework for cognition?

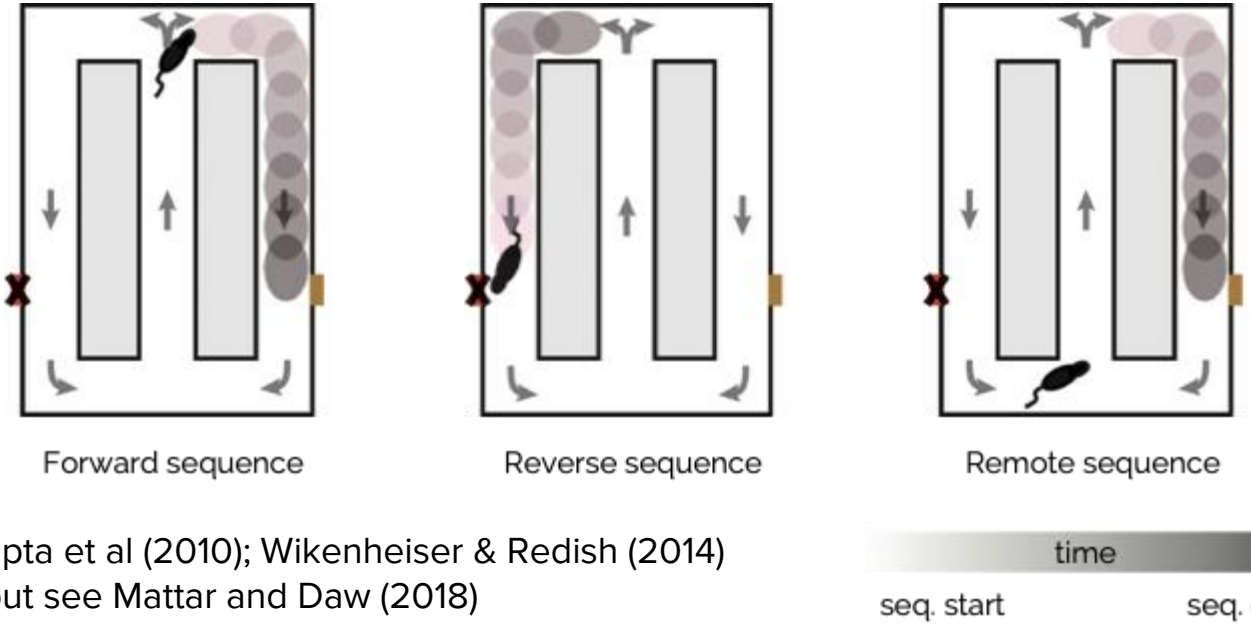


Replay for consolidation



Ji, D. & Wilson, M.A. (2007)

Replay for planning



Gupta et al (2010); Wikenheiser & Redish (2014)
...but see Mattar and Daw (2018)