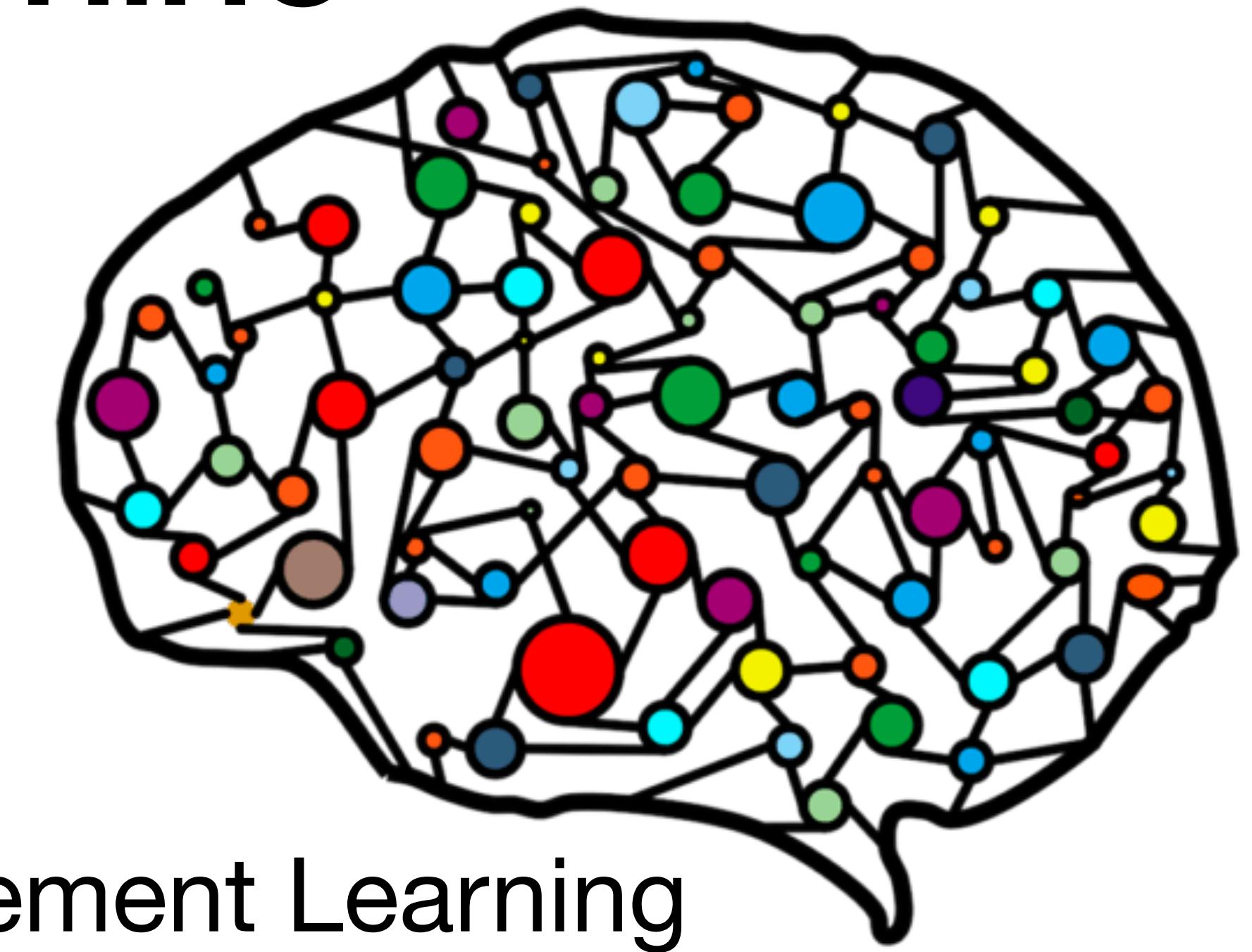


General Principles of Human and Machine Learning



Lecture 4: Introduction to Reinforcement Learning

Dr. Charley Wu

<https://hmc-lab.com/GPHML.html>

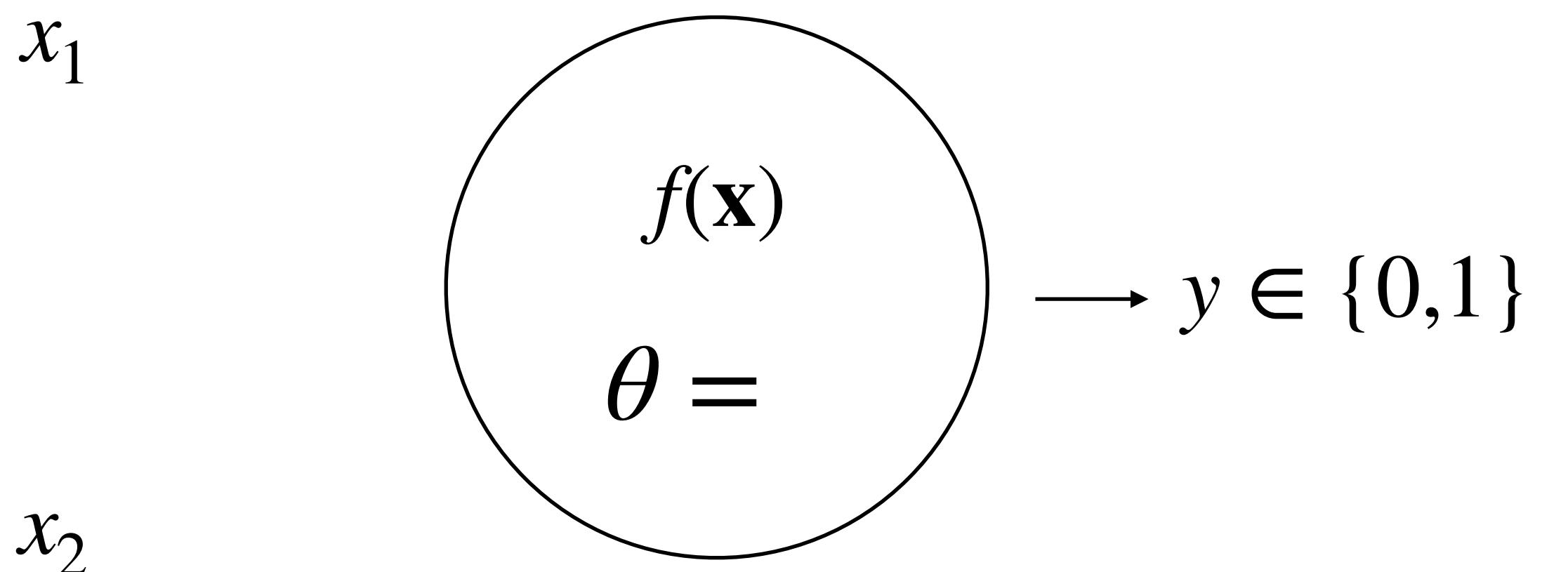
Quiz results

- Average grade 81%
- If you did well, please keep up the good work!
- If you wish you had done better, use this as a learning experience and remember that 1 pop quiz is a freebie (best 3 out of 4)
- If you missed the quiz but had a documented absence (email to me + TA 24hrs in advance), then we can work something out for further documented absences
- Quiz questions may reappear on the exam

Clarifications

NAND

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \sum w_i x_i \geq \theta \\ 0 & \text{else} \end{cases}$$

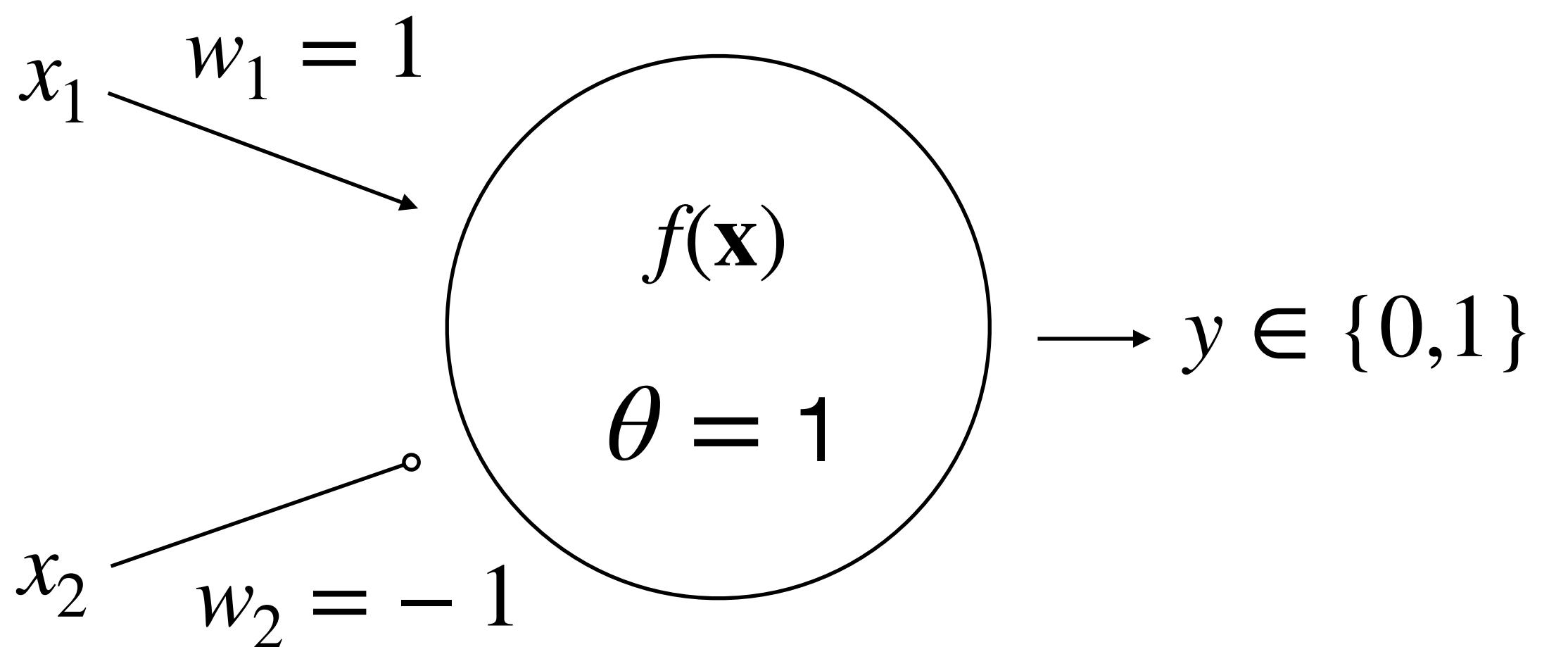


Neuron fires when x_1 is on AND x_2 not on

Clarifications

NAND

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \sum w_i x_i \geq \theta \\ 0 & \text{else} \end{cases}$$

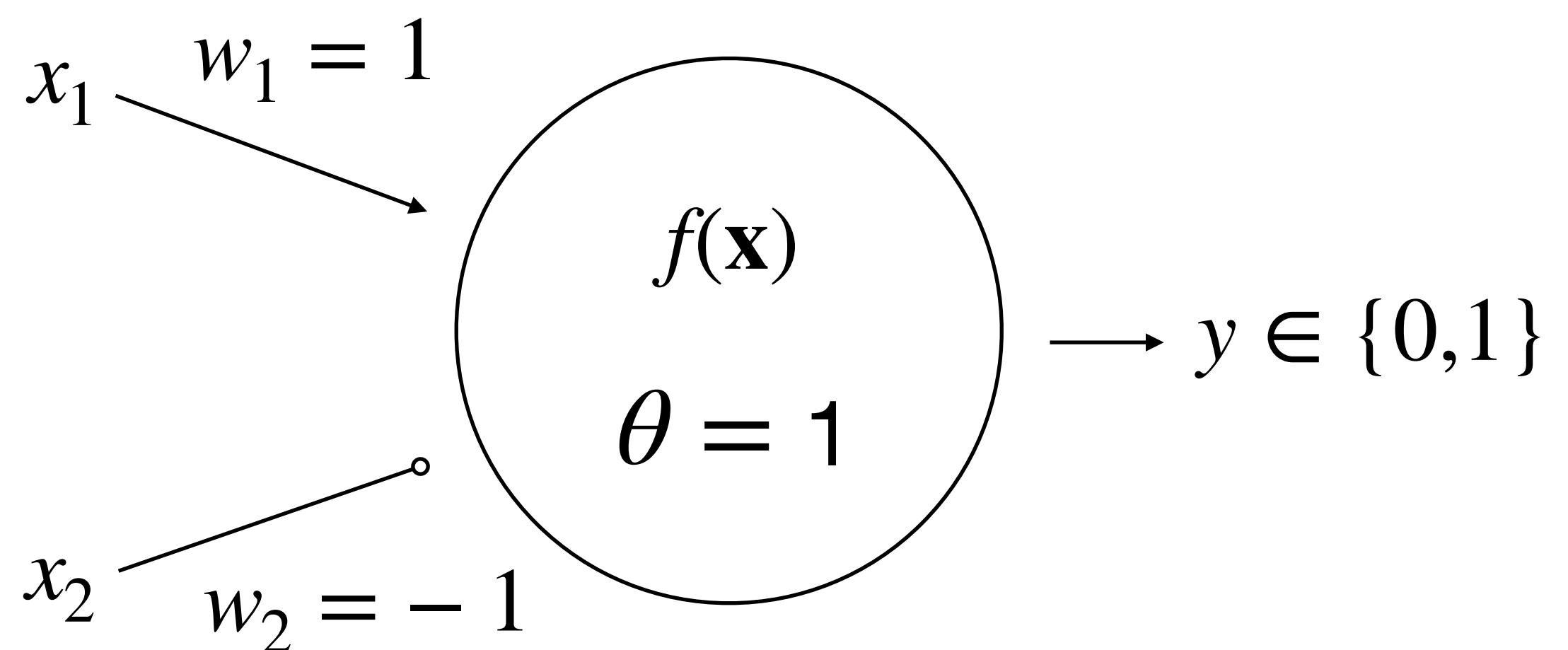


Neuron fires when x_1 is on AND x_2 not on

Clarifications

NAND

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \sum w_i x_i \geq \theta \\ 0 & \text{else} \end{cases}$$



Neuron fires when x_1 is on AND x_2 not on

Rescorla-Wagner

Reward prediction

$$\hat{r}_t = \sum_i \text{CS}_i^t w_i$$

Weight update

For i where $CS_i = 1$:

$$w_i \leftarrow w_i + \eta(r_t - \hat{r}_t)$$

Learning rate

Observed outcome

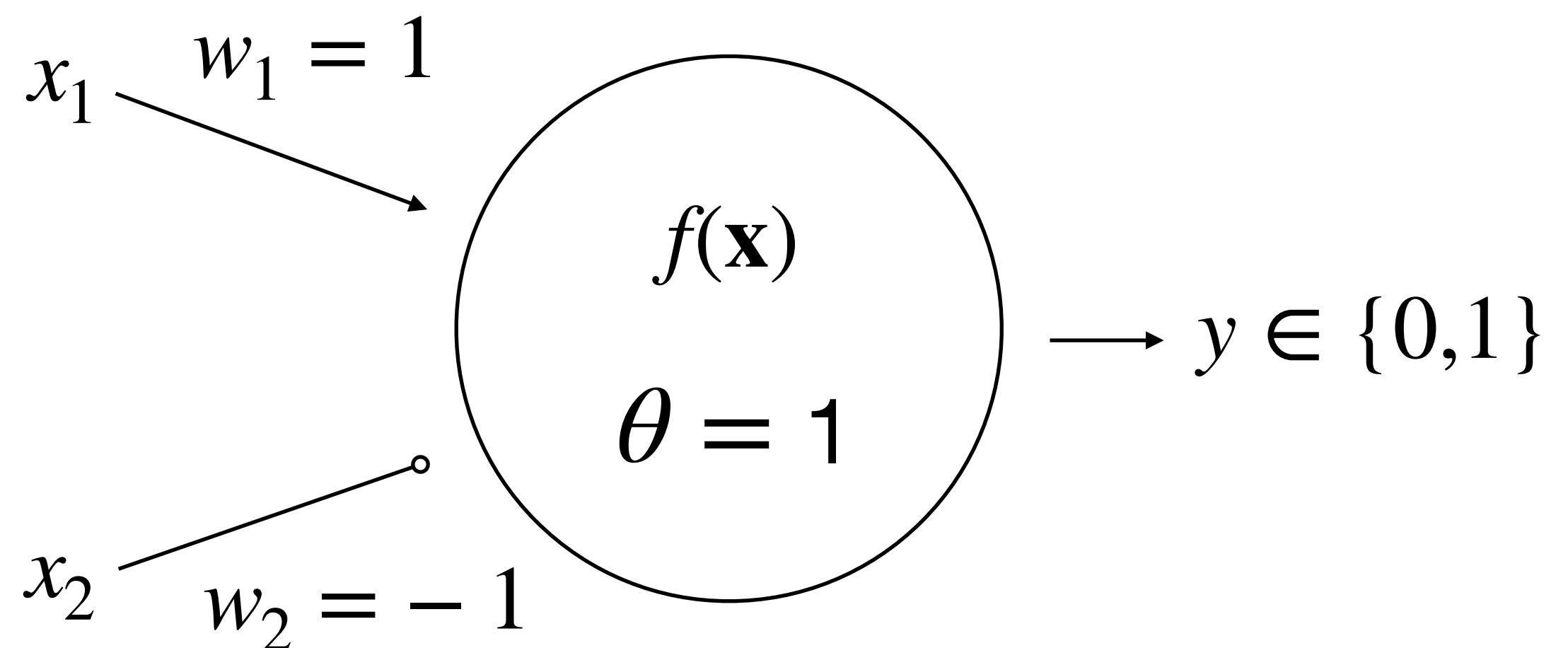
Predicted outcome

δ Reward prediction error (RPE) 3

Clarifications

NAND

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \sum w_i x_i \geq \theta \\ 0 & \text{else} \end{cases}$$



Neuron fires when x_1 is on AND x_2 not on

Rescorla-Wagner

Reward prediction

$$\hat{r}_t = \sum_i \text{CS}_i^t w_i$$

Weight update

For i where $\text{CS}_i = 1$:

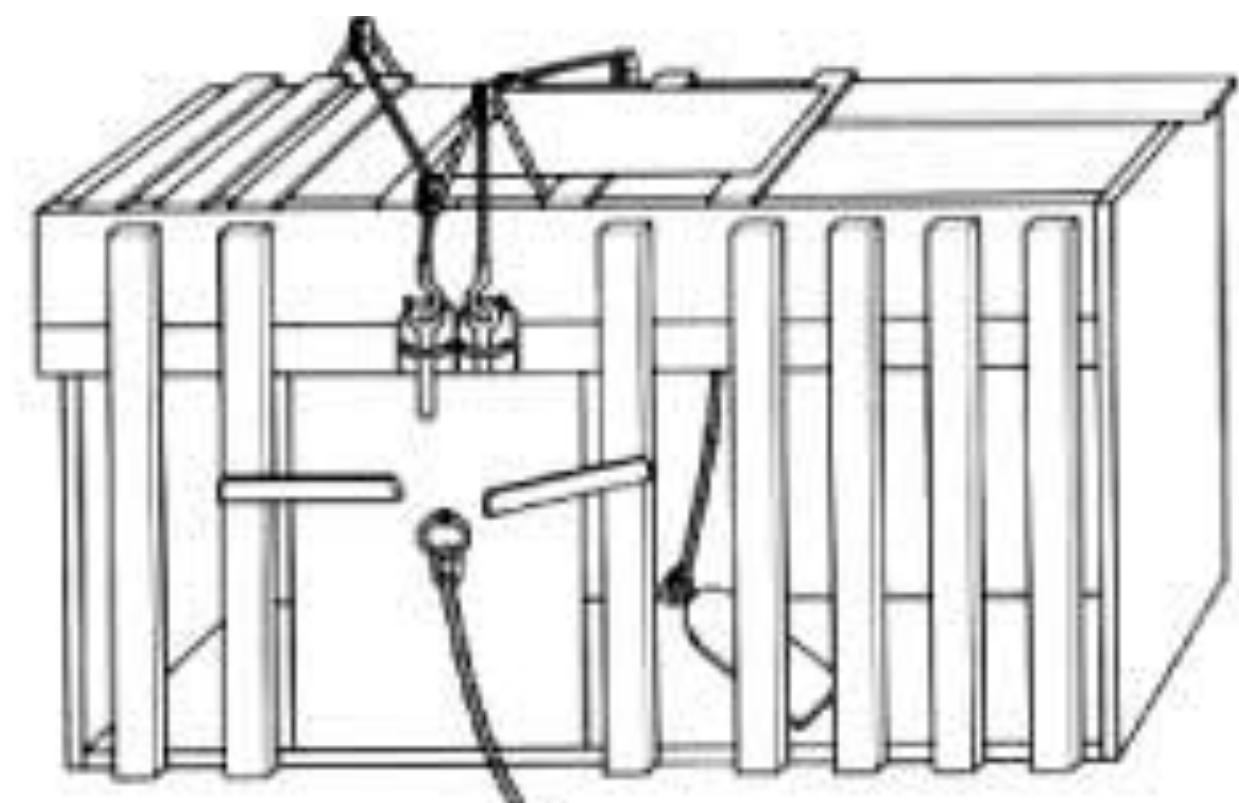
$$w_i \leftarrow w_i + \eta(r_t - \hat{r}_t)$$

Larger when better reward than expected!

Learning rate Observed outcome Predicted outcome
 δ Reward prediction error (RPE)

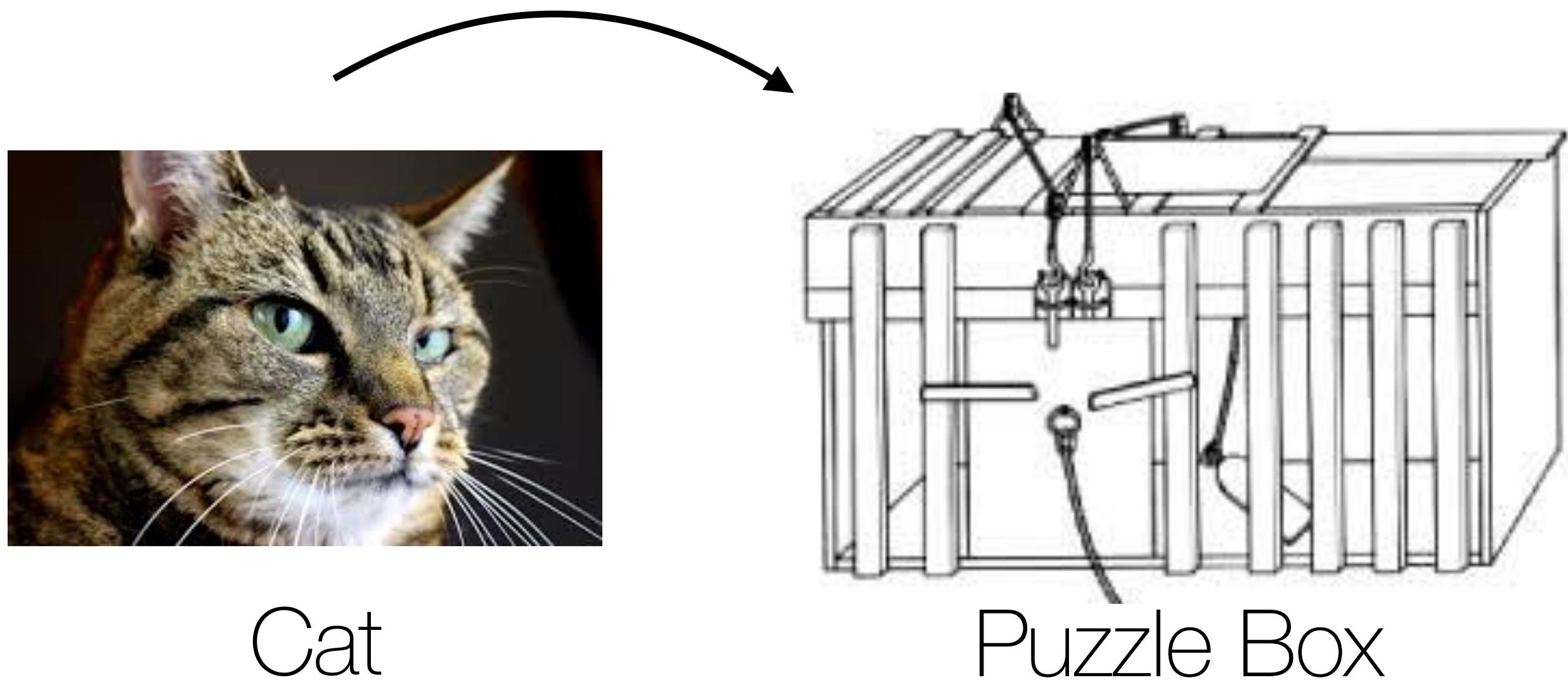
The story so far ...

Thorndike's (1898) Law of Effect

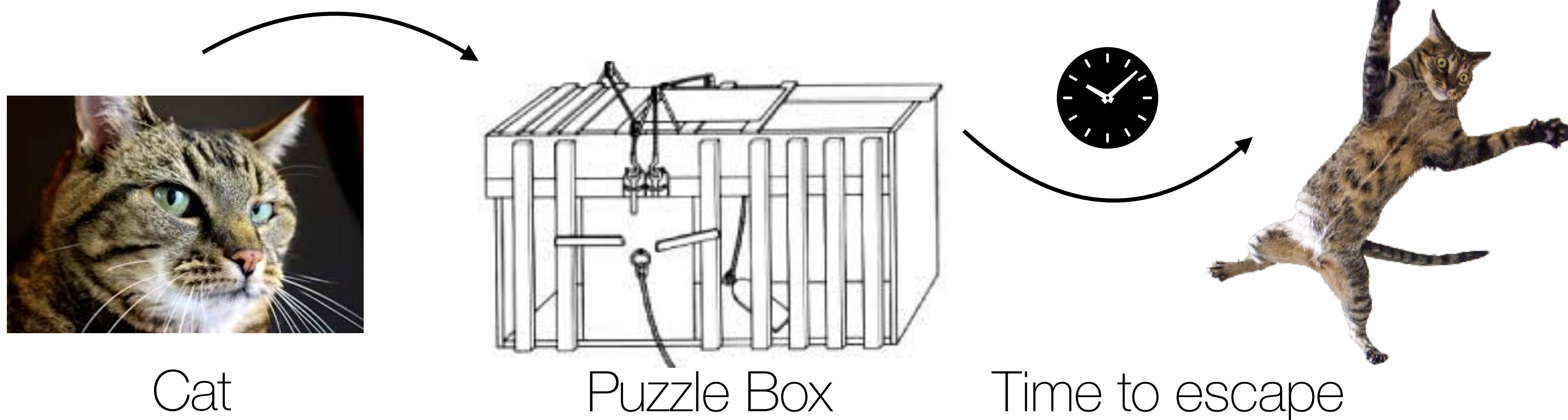


Puzzle Box

Thorndike's (1898) Law of Effect



Thorndike's (1898) Law of Effect

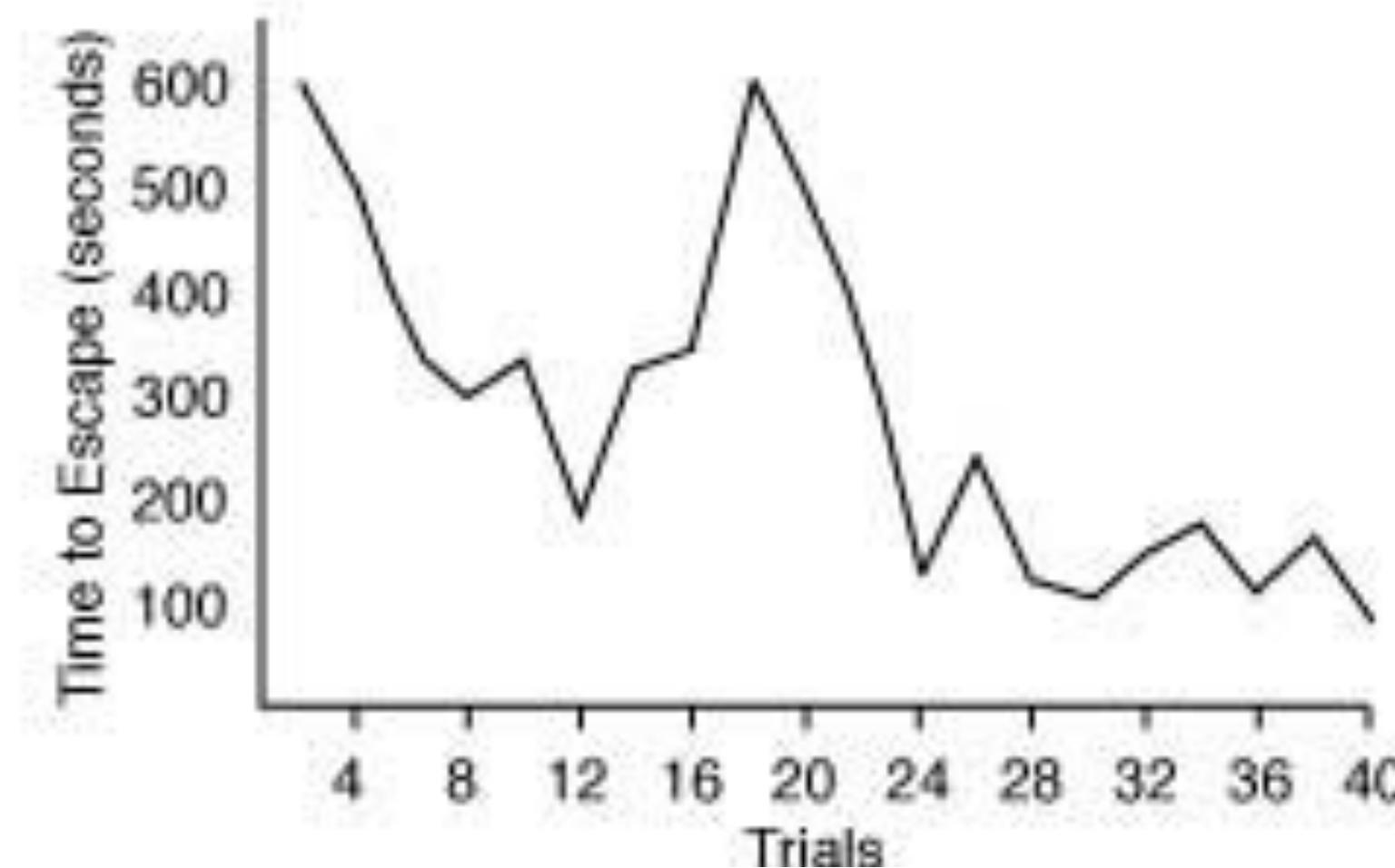
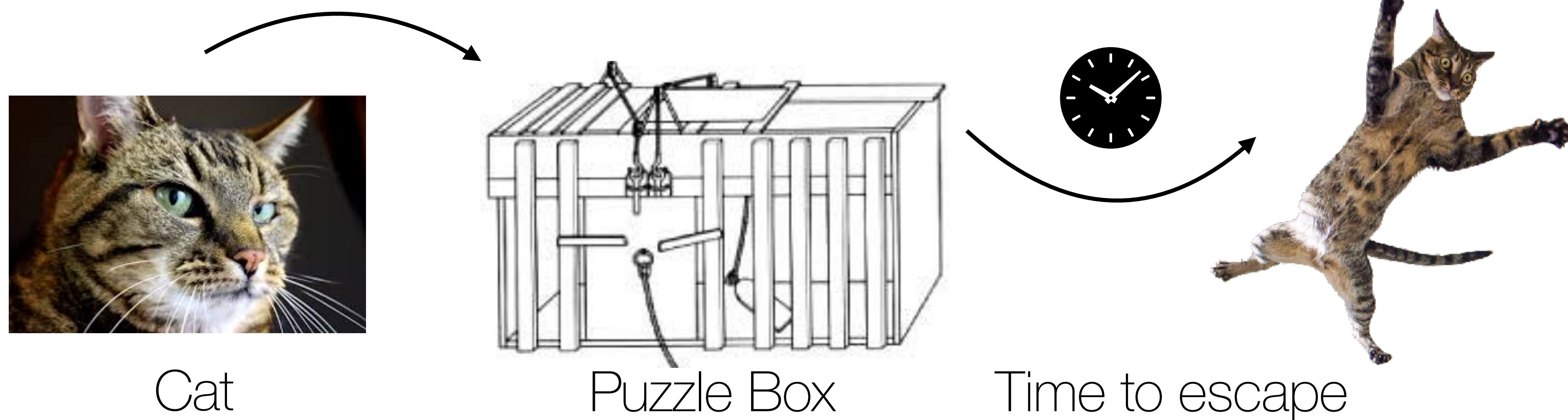


Cat

Puzzle Box

Time to escape

Thorndike's (1898) Law of Effect

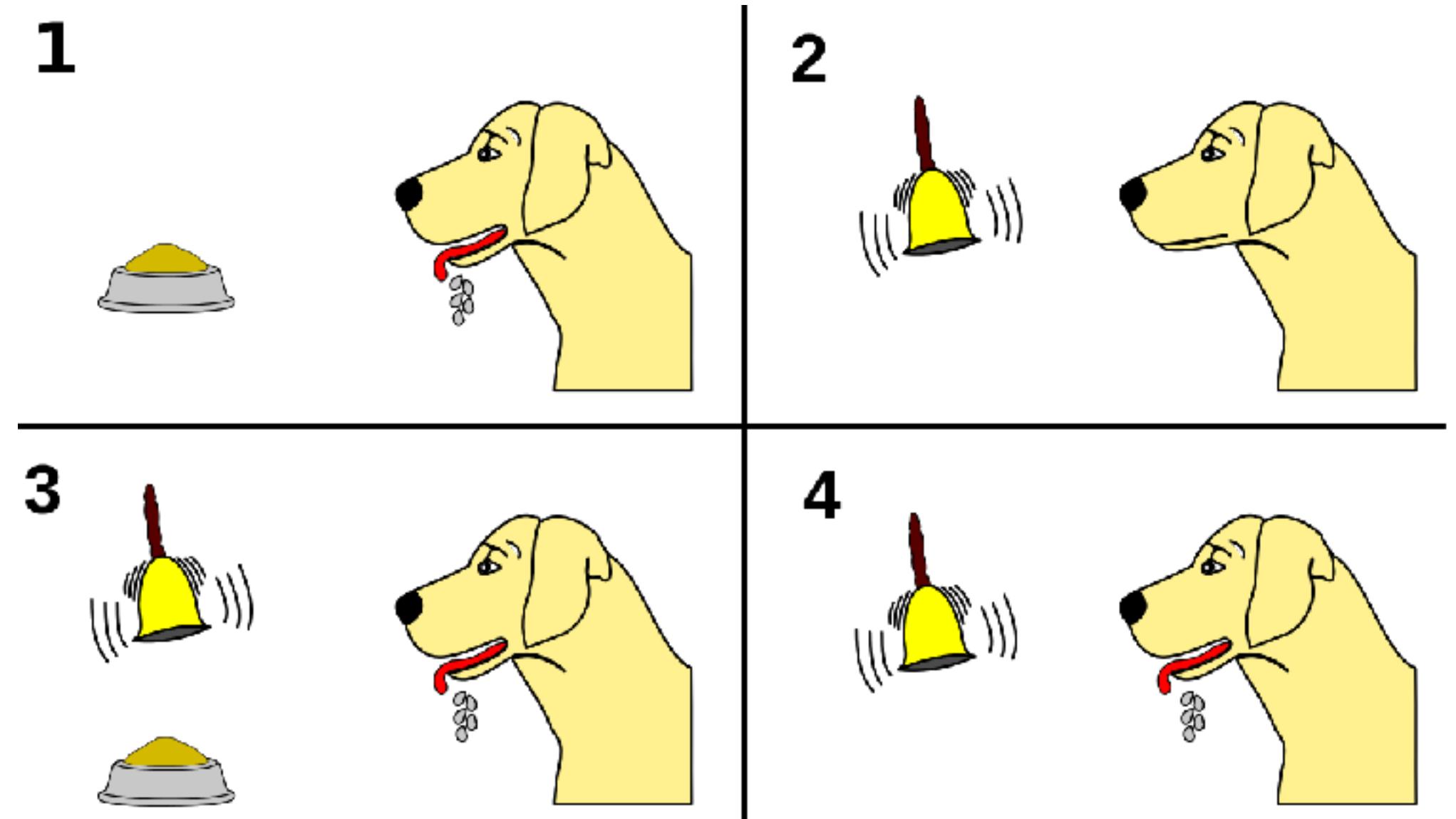


Actions associated with satisfaction are strengthened, while those associated with discomfort become weakened.

Classical and Operant Conditioning

Classical Condition (Pavlov, 1927)

Learning as the passive coupling of stimulus (bell ringing) and response (salivation), anticipating future rewards



Operant Condition (Skinner, 1938)

Skinner (1938): Learning as the active shaping of behavior in response to rewards or punishments



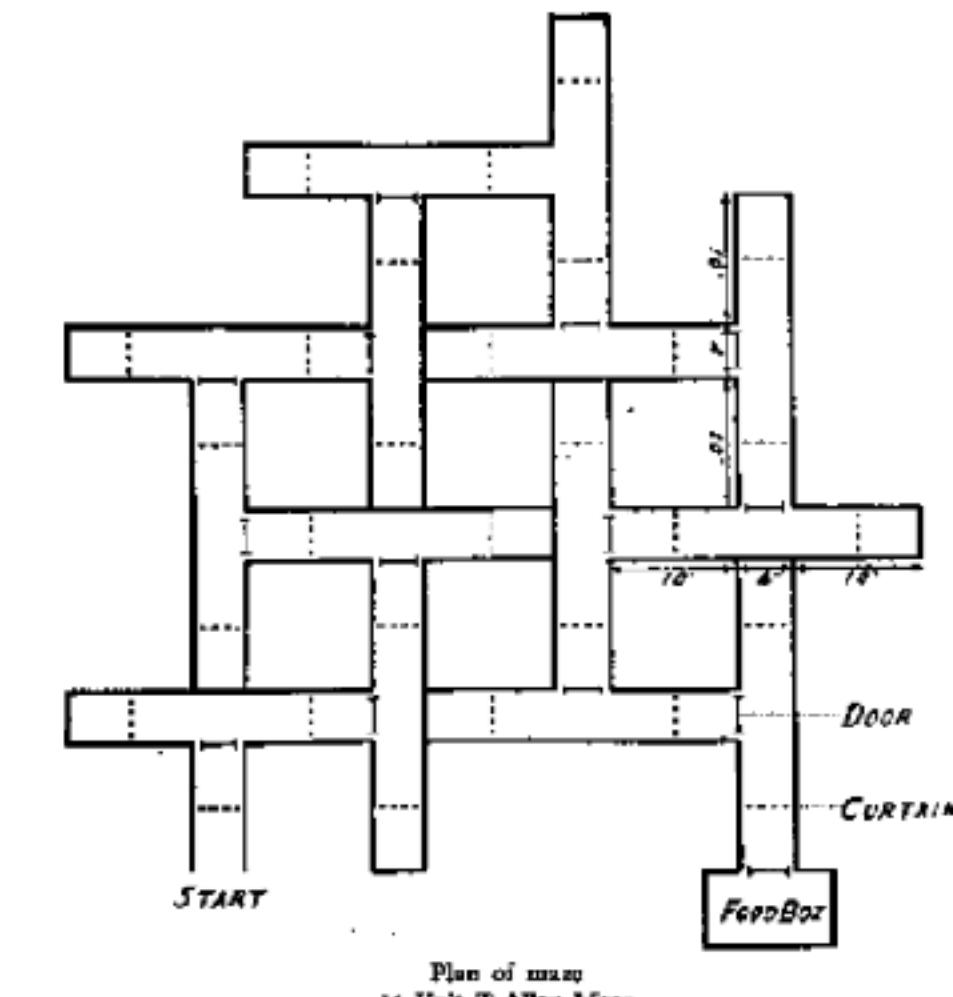
Tolman and Cognitive maps

- Learning is not just a telephone switchboard connecting incoming sensory signals to outgoing responses (S-R Learning)
- Rather, “latent learning” establishes something like a “field map of the environment” gets established (S-S learning)

Stimulus-Response (S-R) Learning



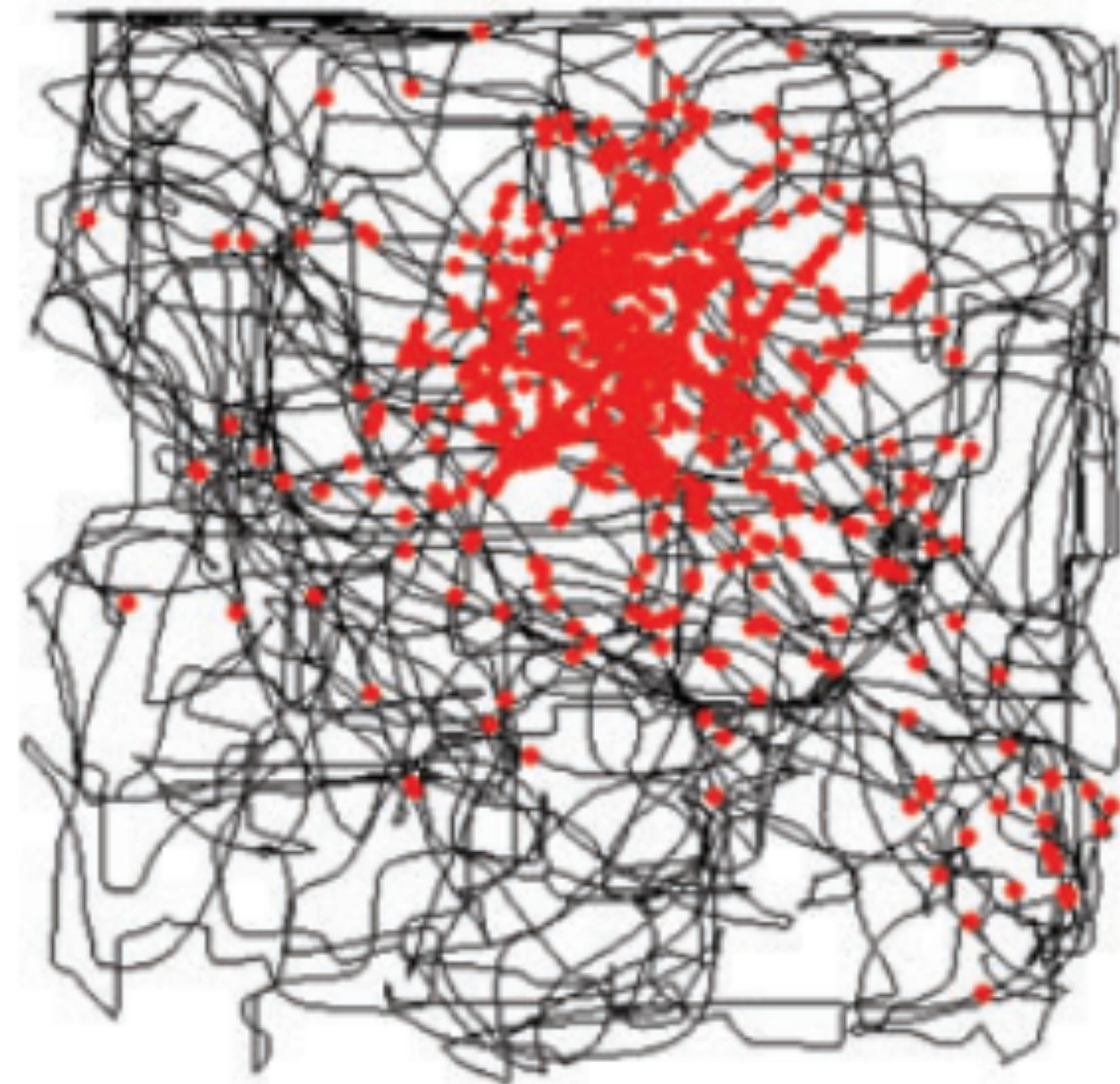
Stimulus-Stimulus (S-S) Learning



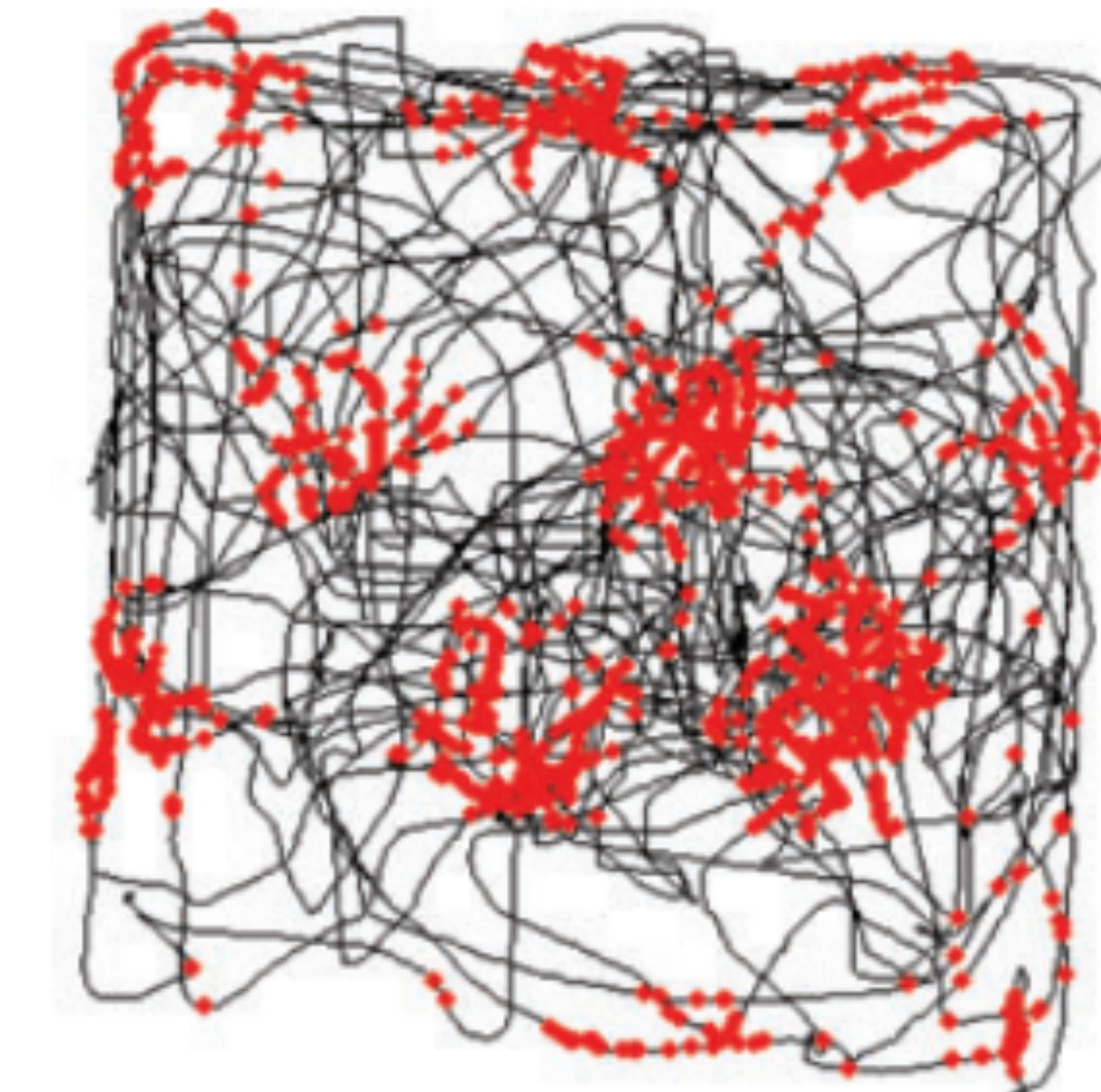
Plan of maze
in Unit T Alley Maze
FIG. 1
(From M. H. Elliott, The effect of change of reward on the maze performance of rats. *Univ. Calif. Publ. Psychol.*, 1928, 4, p. 20.)

Cognitive maps in biological brains

Place cells in the hippocampus

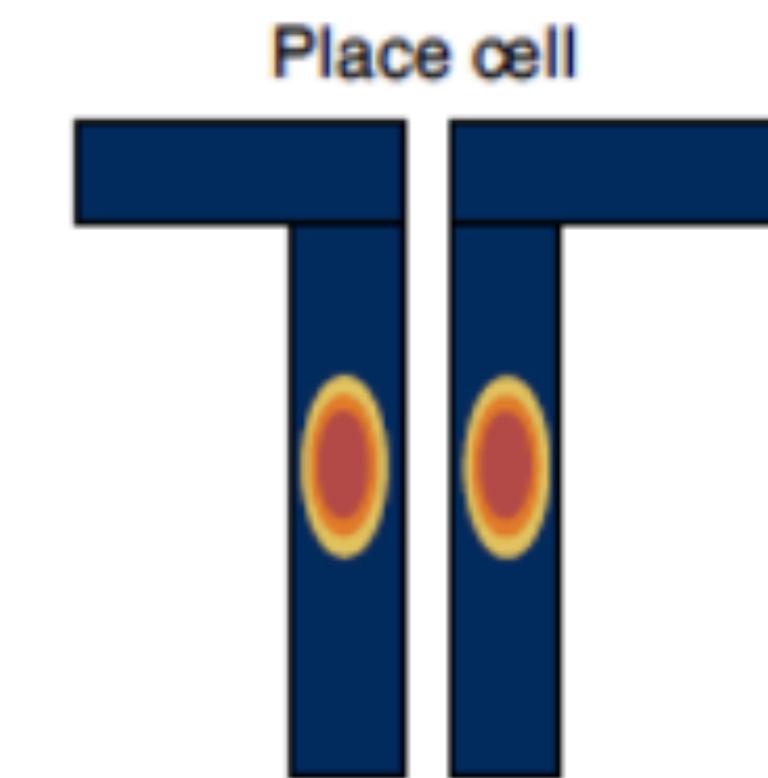
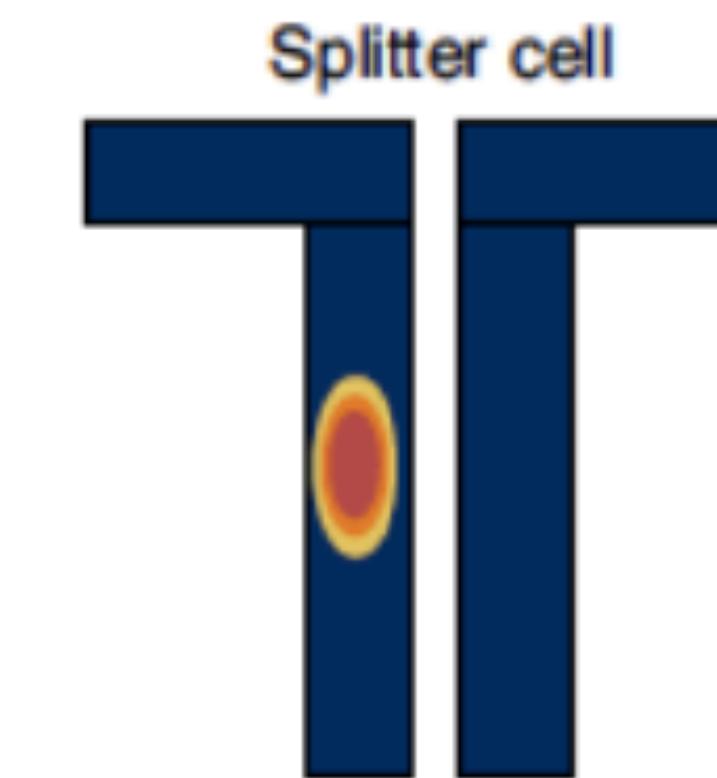
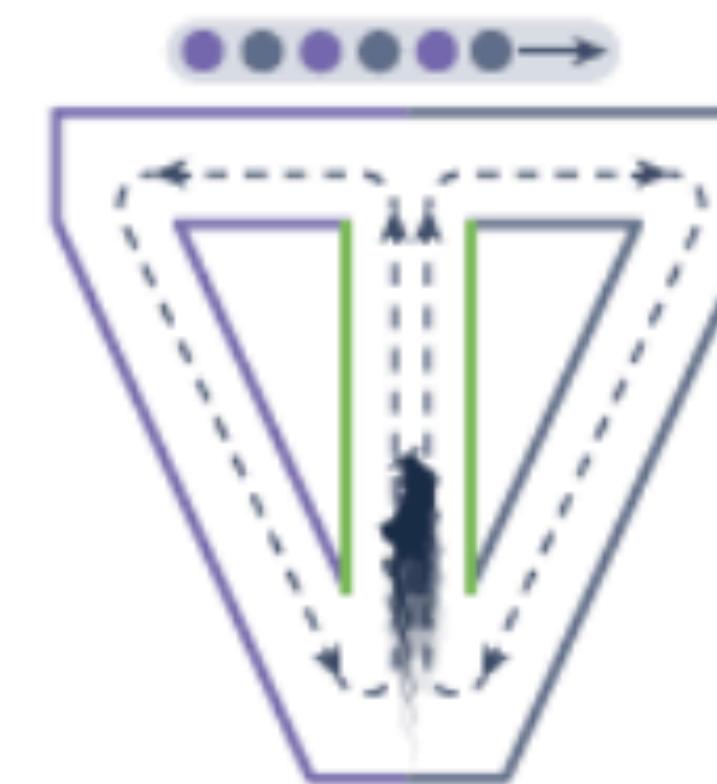
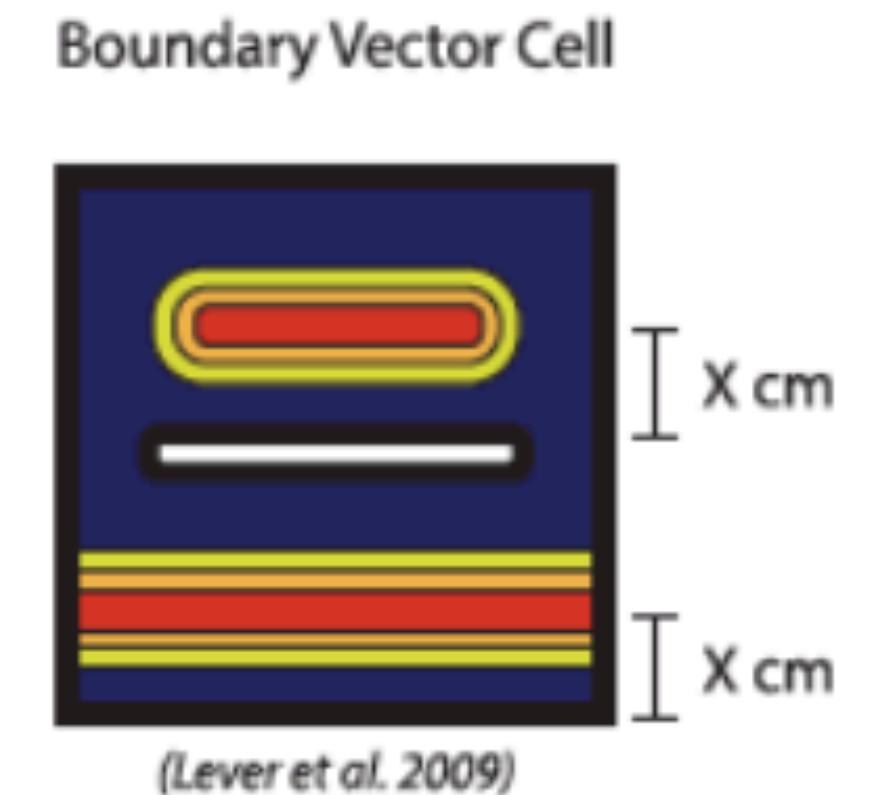
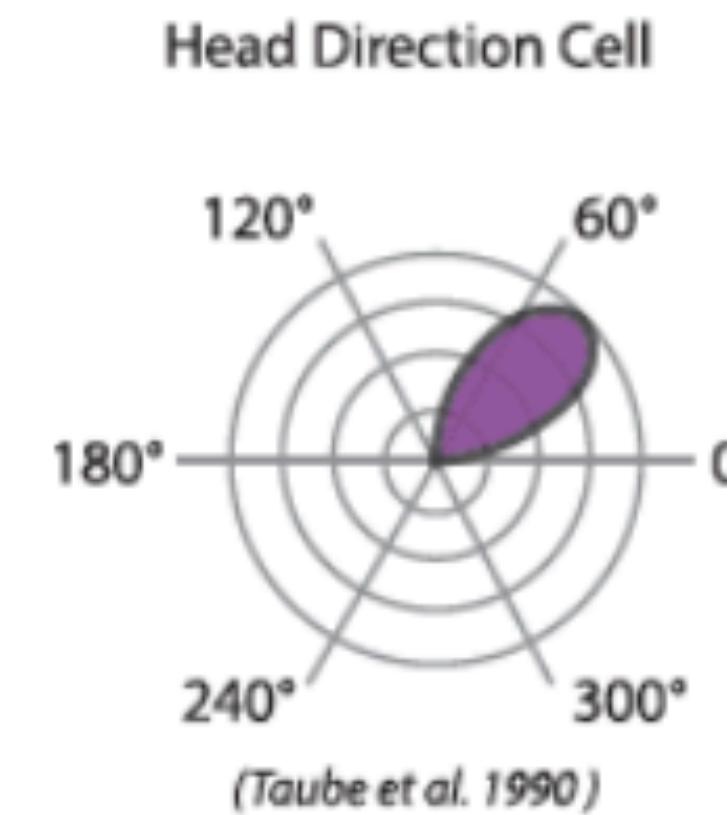
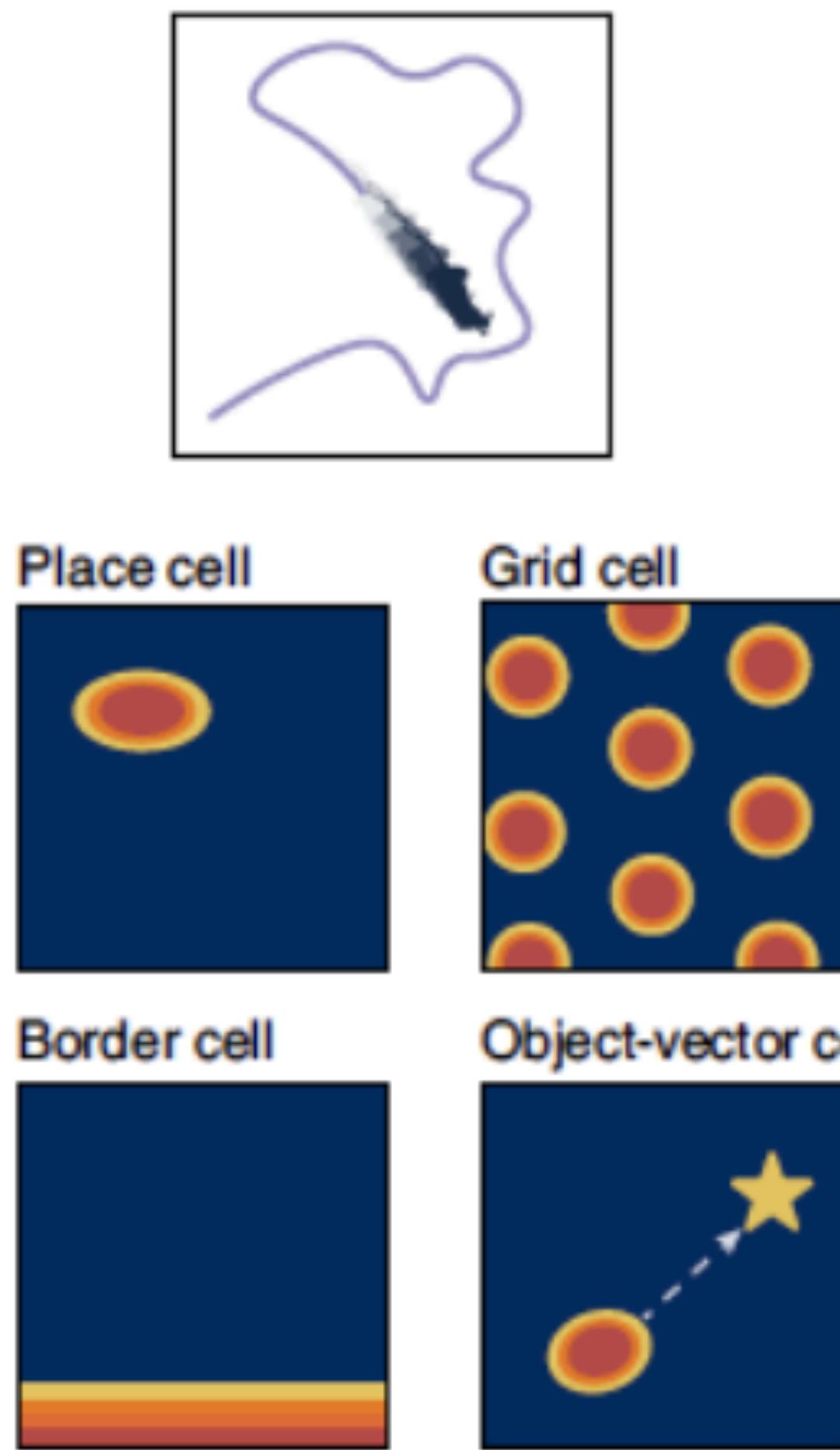


Grid cells in the medial entorhinal cortex



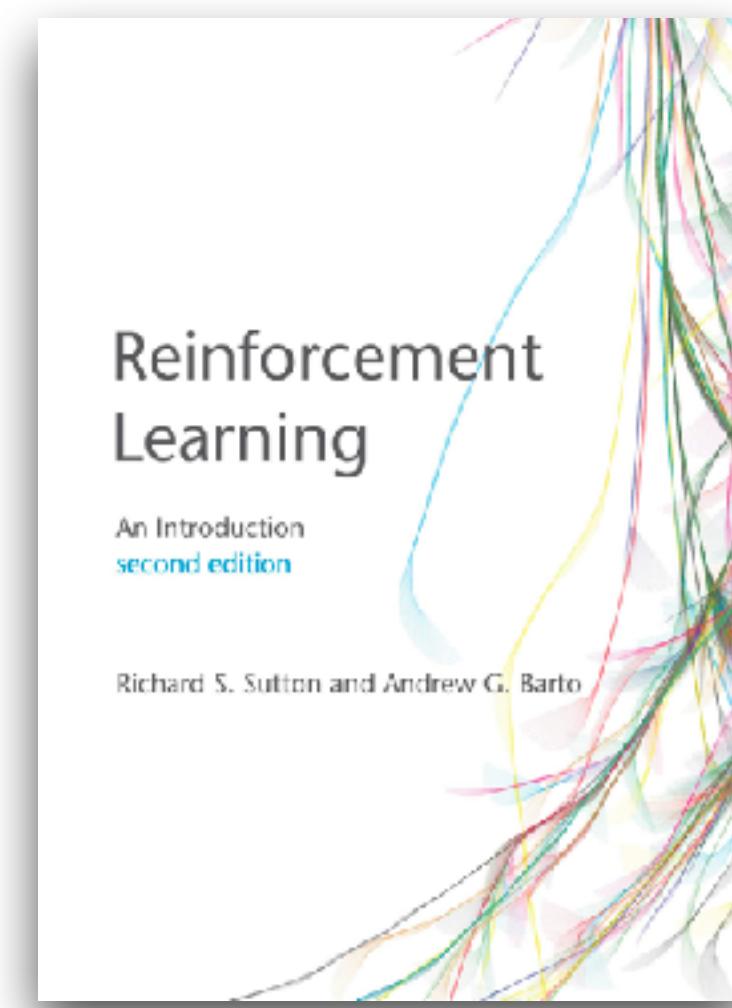
Moser et al., (*Ann Rev Neuro* 2008)

“Hippocampal zoo”



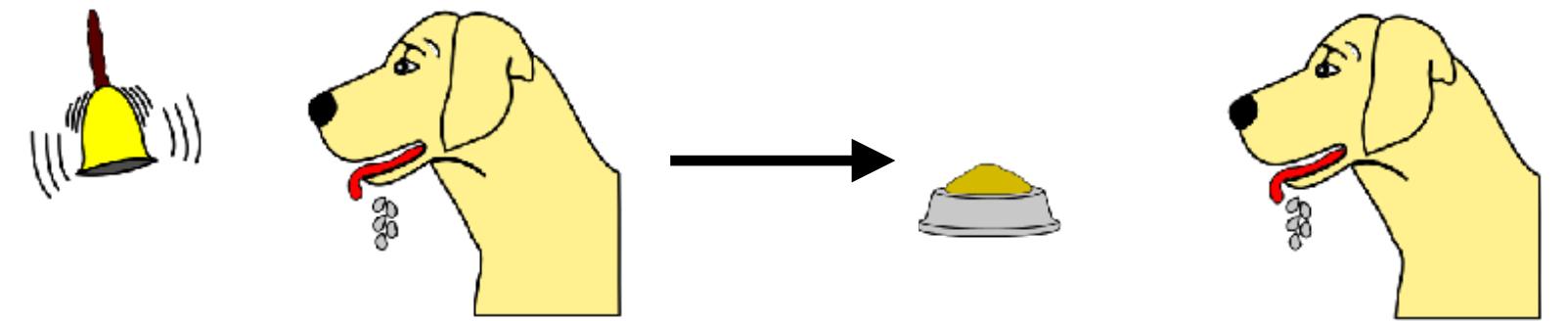
Agenda for today: From Tolman to Reinforcement Learning

- **Part 1:** Introduce RL framework, origins, and terminology (Sutton & Barto)
- **Part 2:** Model-free vs. model-based RL

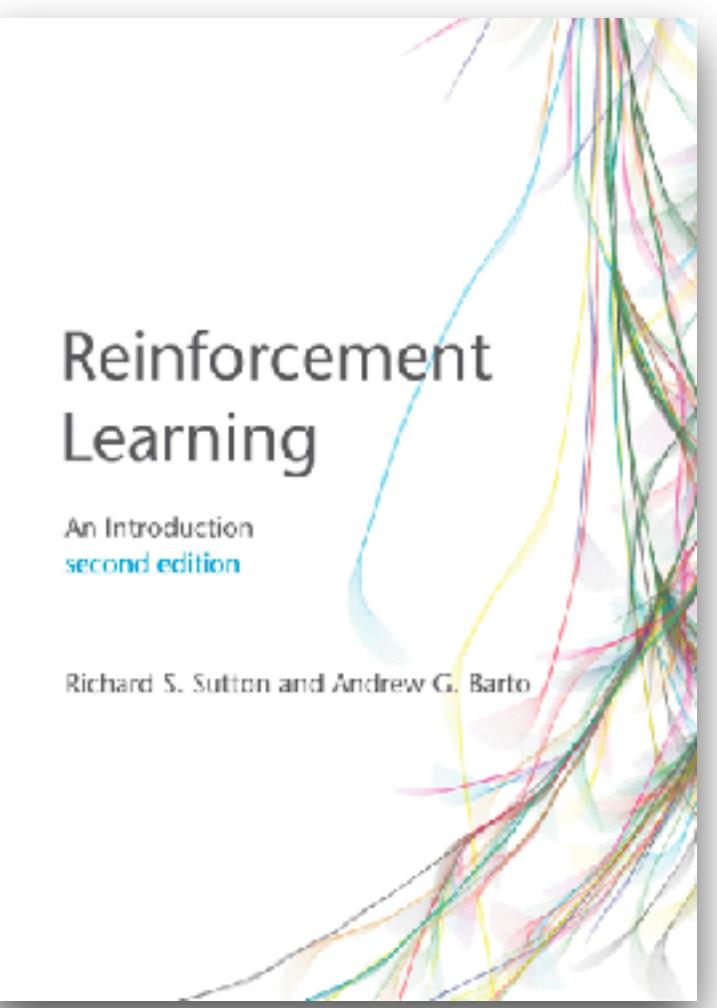


Reinforcement Learning

Pavlovian (classical) conditioning

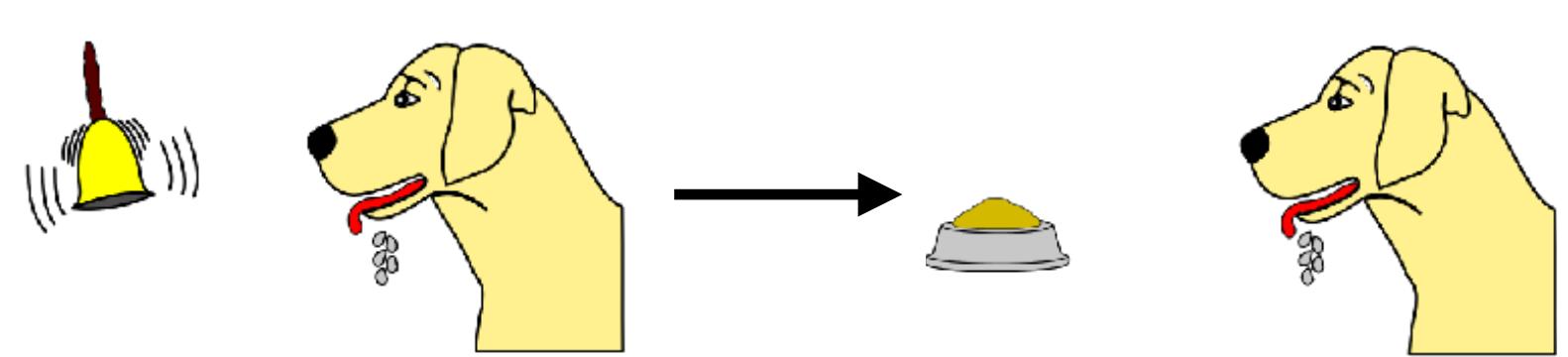


Learn which environmental cues *predict* reward



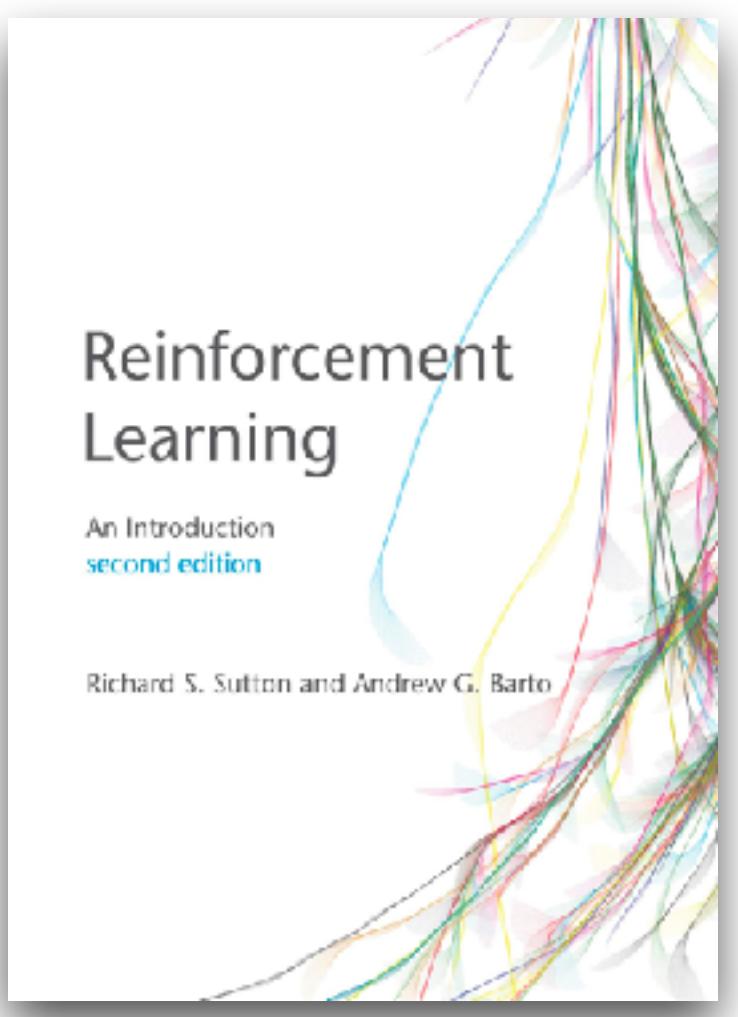
Reinforcement Learning

Pavlovian (classical) conditioning

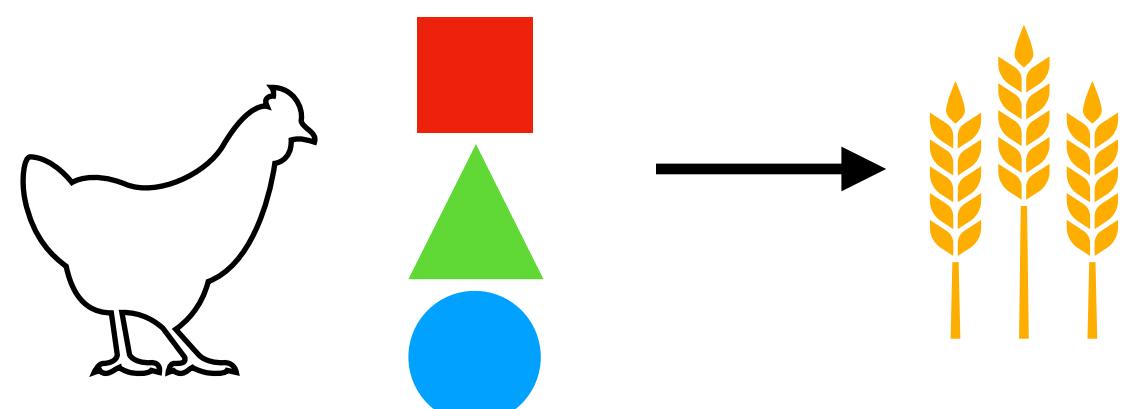


Learn which environmental cues *predict* reward

Reinforcement Learning

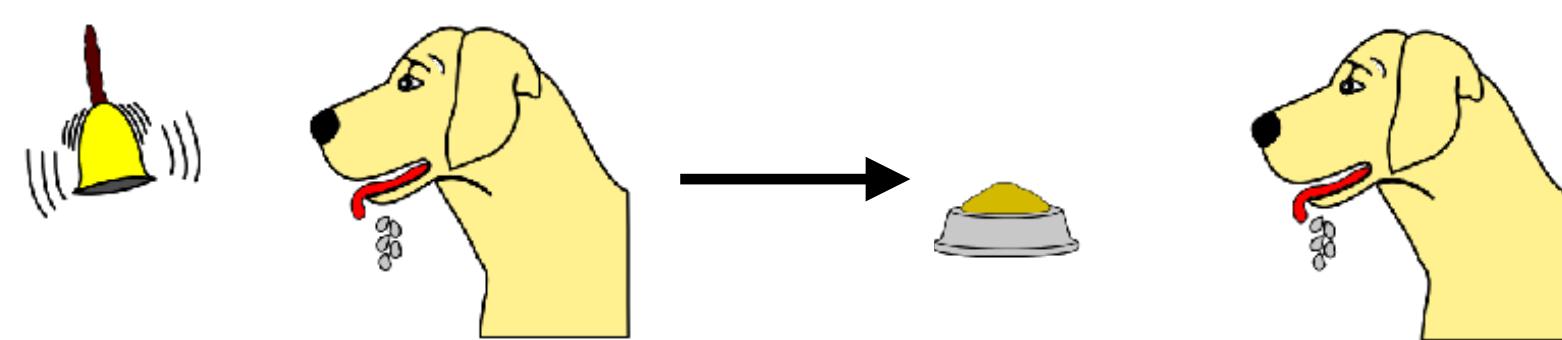


Operant (instrumental) conditioning

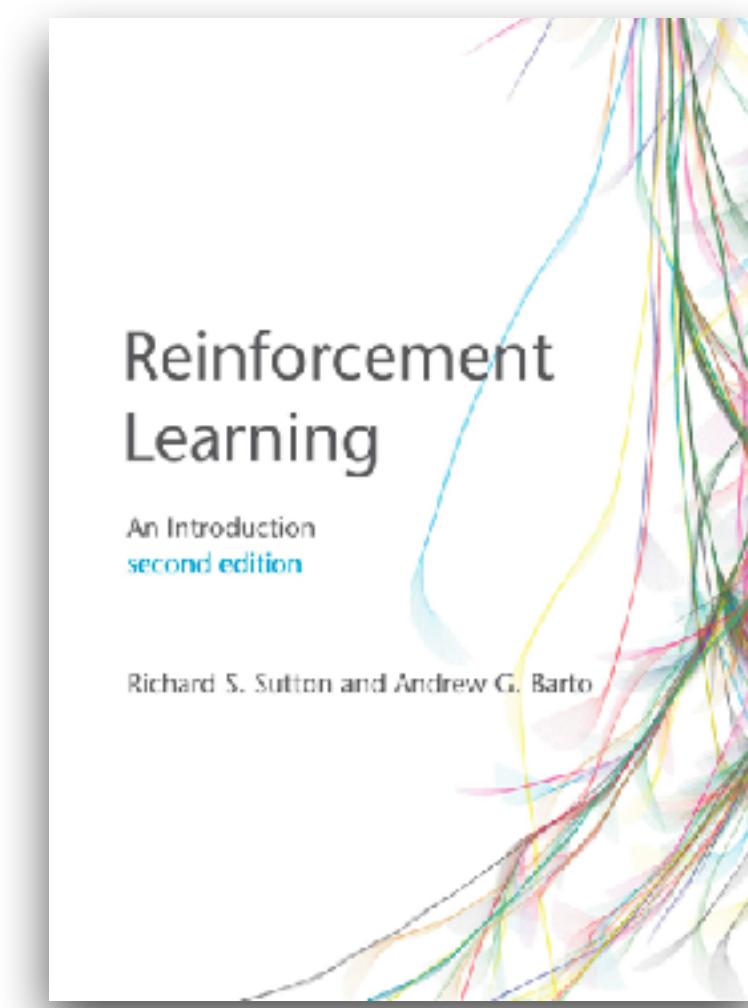


Learn which actions *predict* reward

Pavlovian (classical) conditioning

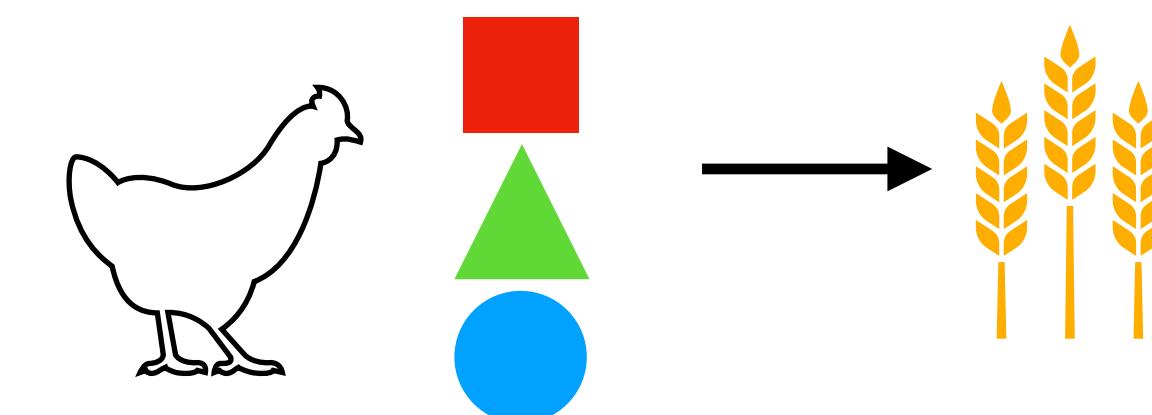


Learn which environmental cues *predict* reward



Reinforcement Learning

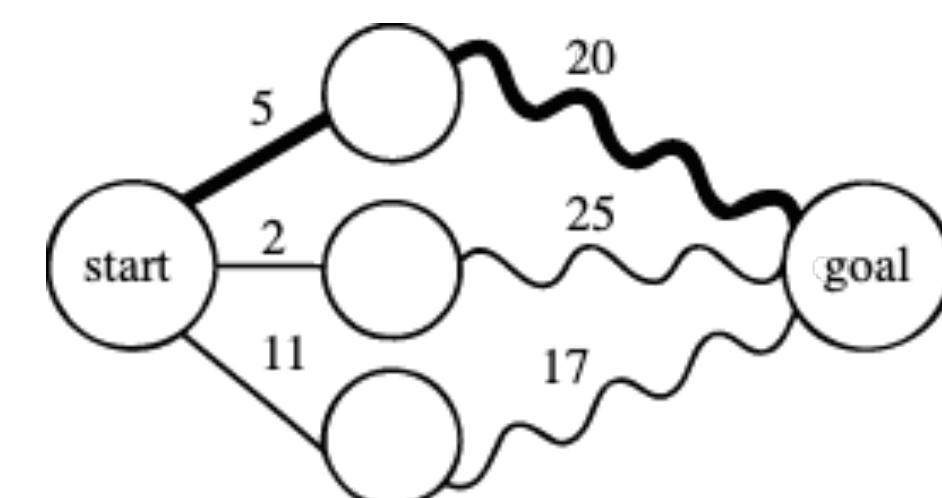
Operant (instrumental) conditioning



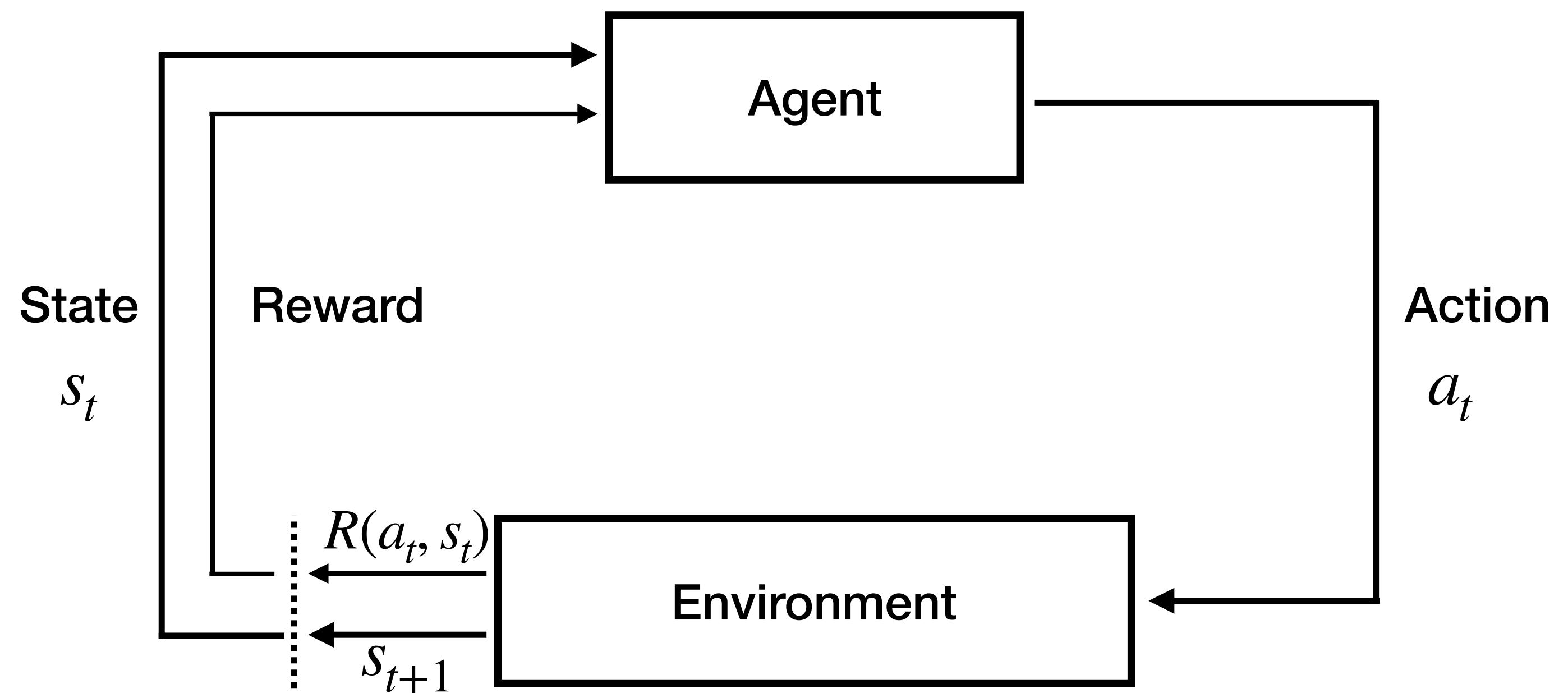
Learn which actions *predict* reward

Neuro-dynamic programming Bertsekas & Tsitsiklis (1996)

Stochastic approximations to dynamic programming problems

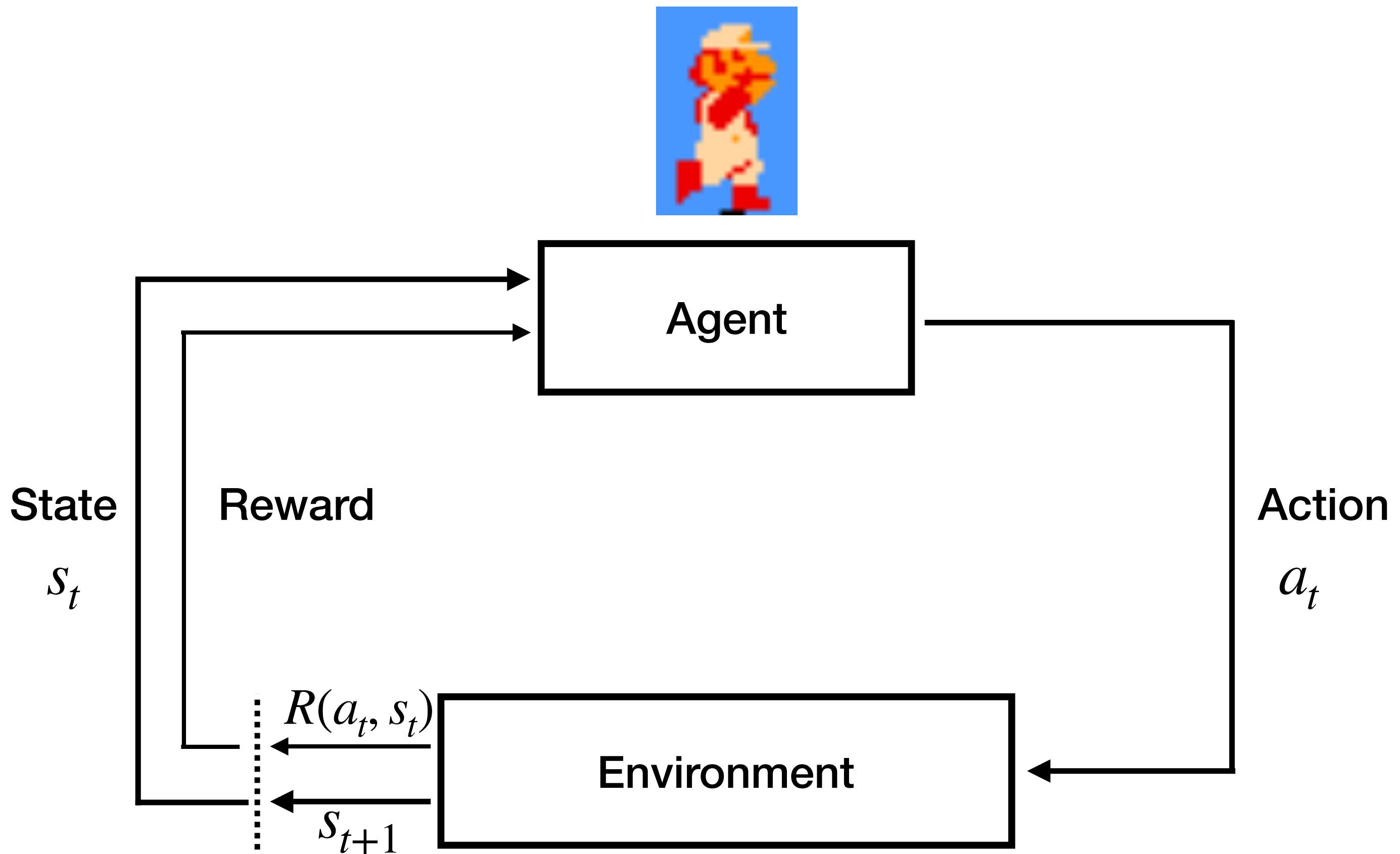


Reinforcement Learning



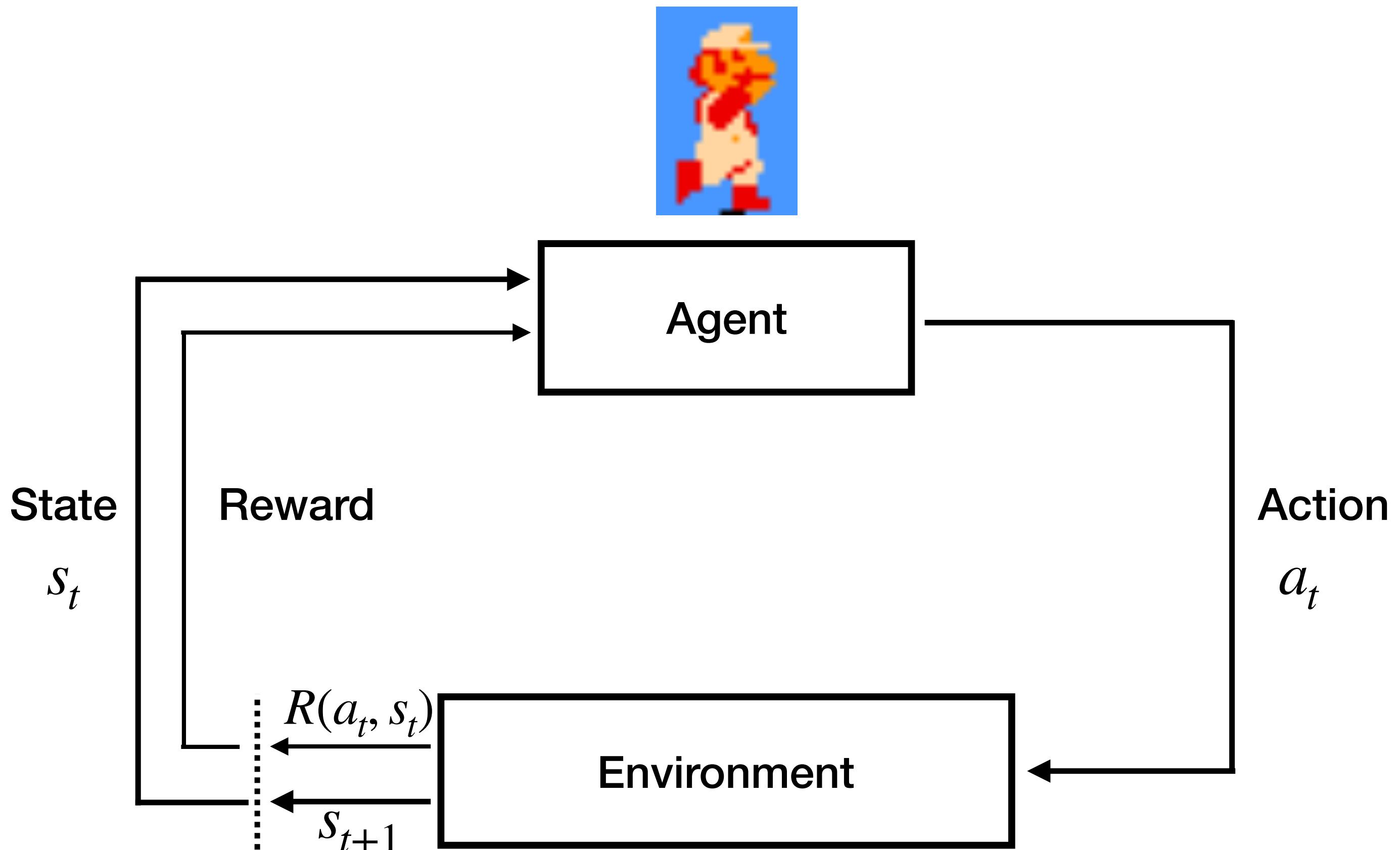
Reinforcement Learning

The Agent:



Reinforcement Learning

The Agent:



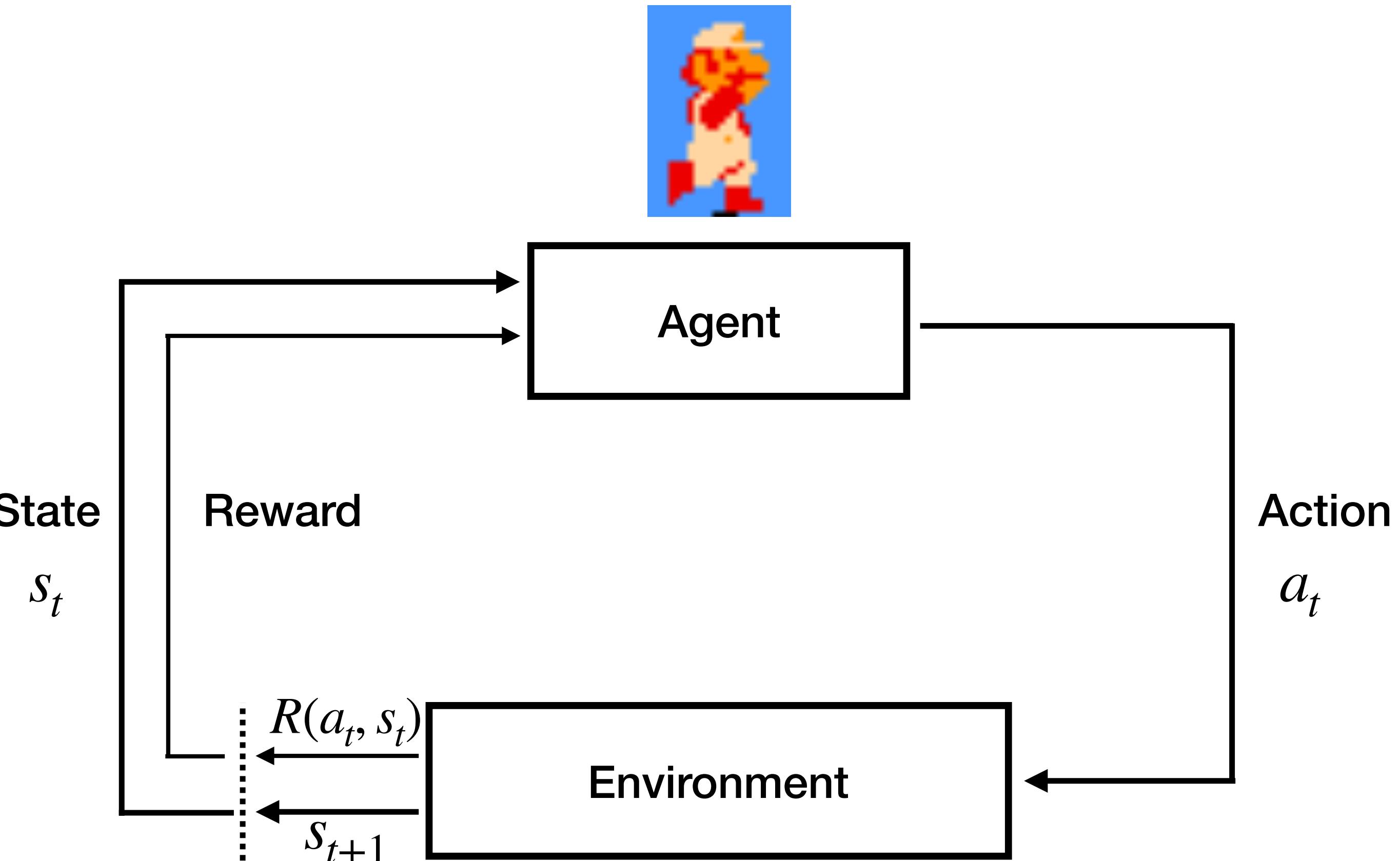
The Environment:



Reinforcement Learning

The Agent:

- Selects actions a_t
- Receives feedback from the environment in terms of new states s_{t+1} and rewards $R(a_t, s_t)$



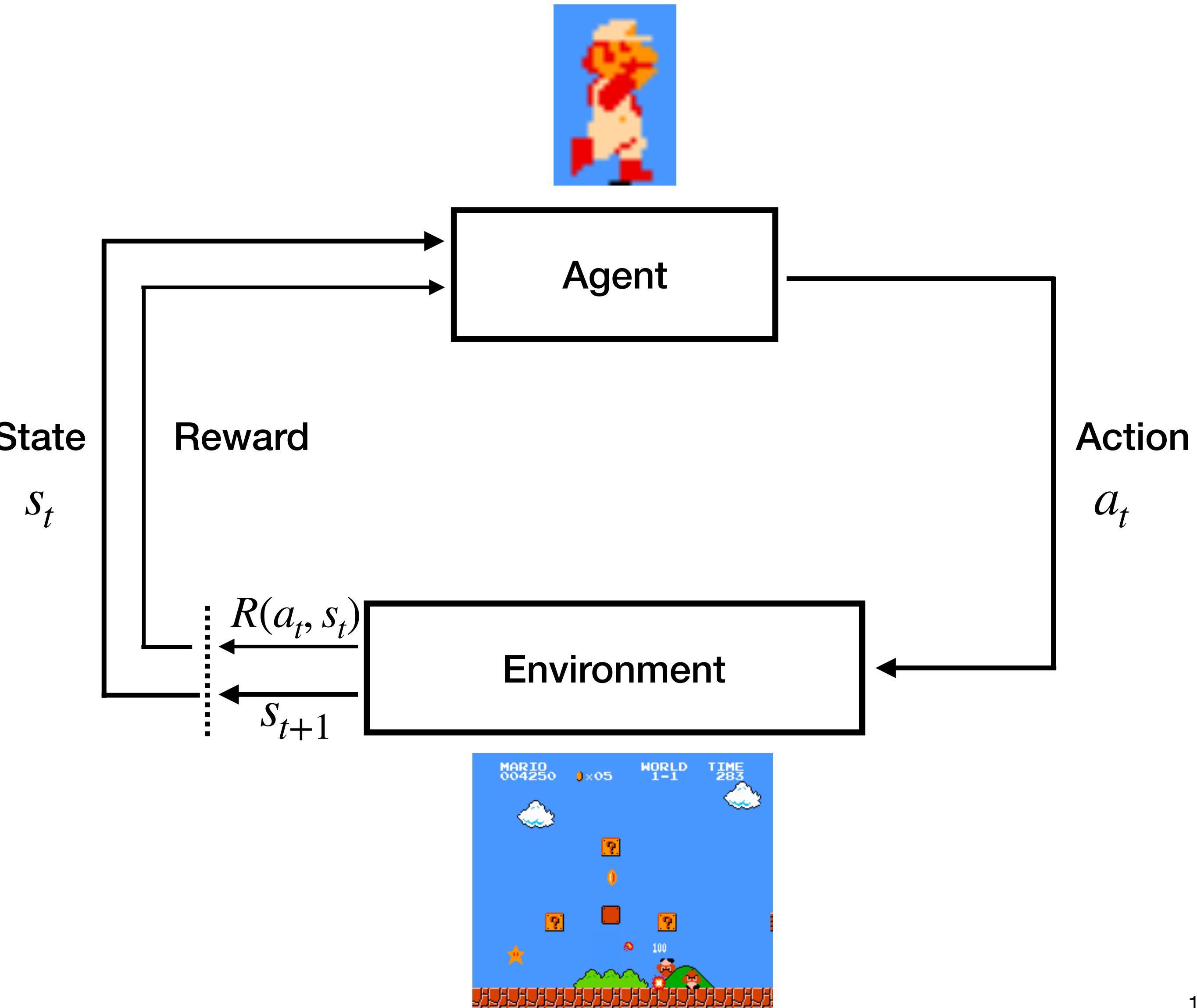
The Environment:



Reinforcement Learning

The Agent:

- Selects actions a_t
- Receives feedback from the environment in terms of new states s_{t+1} and rewards $R(a_t, s_t)$



The Environment:

- Governs the transition between states $s_t \rightarrow s_{t+1}$
- Provides rewards $R(a_t, s_t)$

Environment

actions

states

reward

Environment

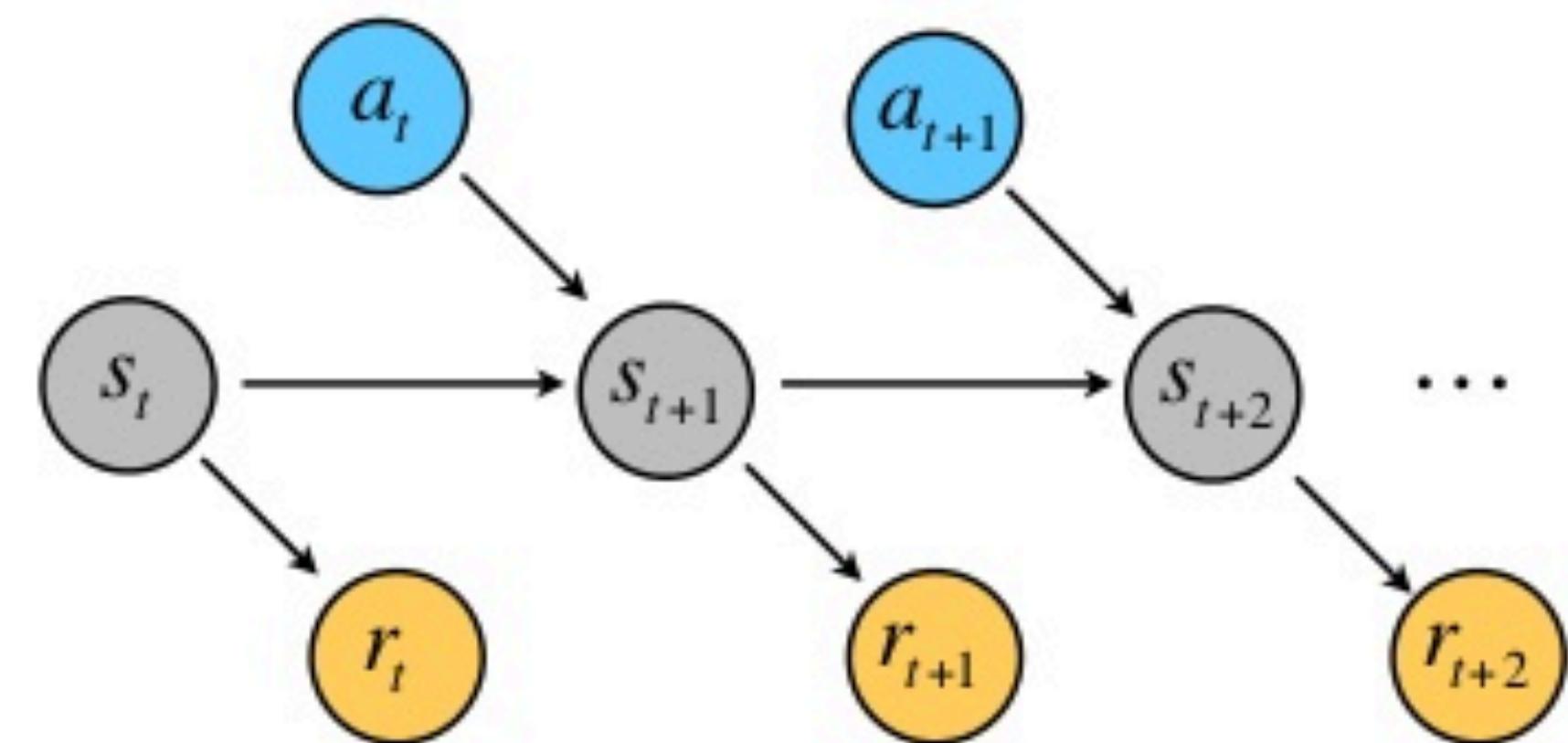
Markov Decision Process (MDP)

- Markov Principle: simplifying assumption that the system is fully defined by only the previous state $P(s_{t+1} | s_t, a_t)$

actions

states

reward



Environment

Markov Decision Process (MDP)

- Markov Principle: simplifying assumption that the system is fully defined by only the previous state $P(s_{t+1} | s_t, a_t)$

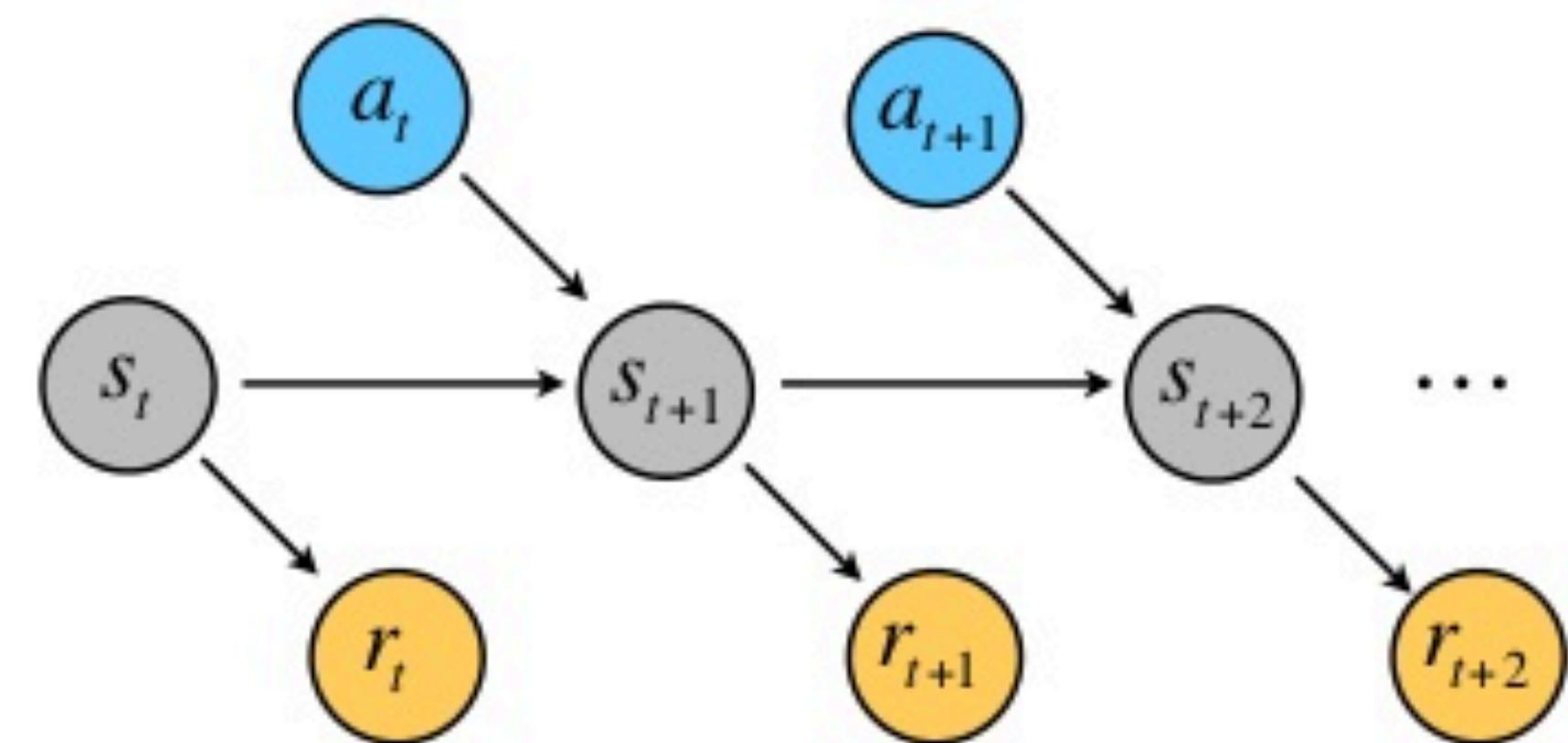
What are states?

- Locations on a grid, pixels on a screen, feature values, etc...

actions

states

reward



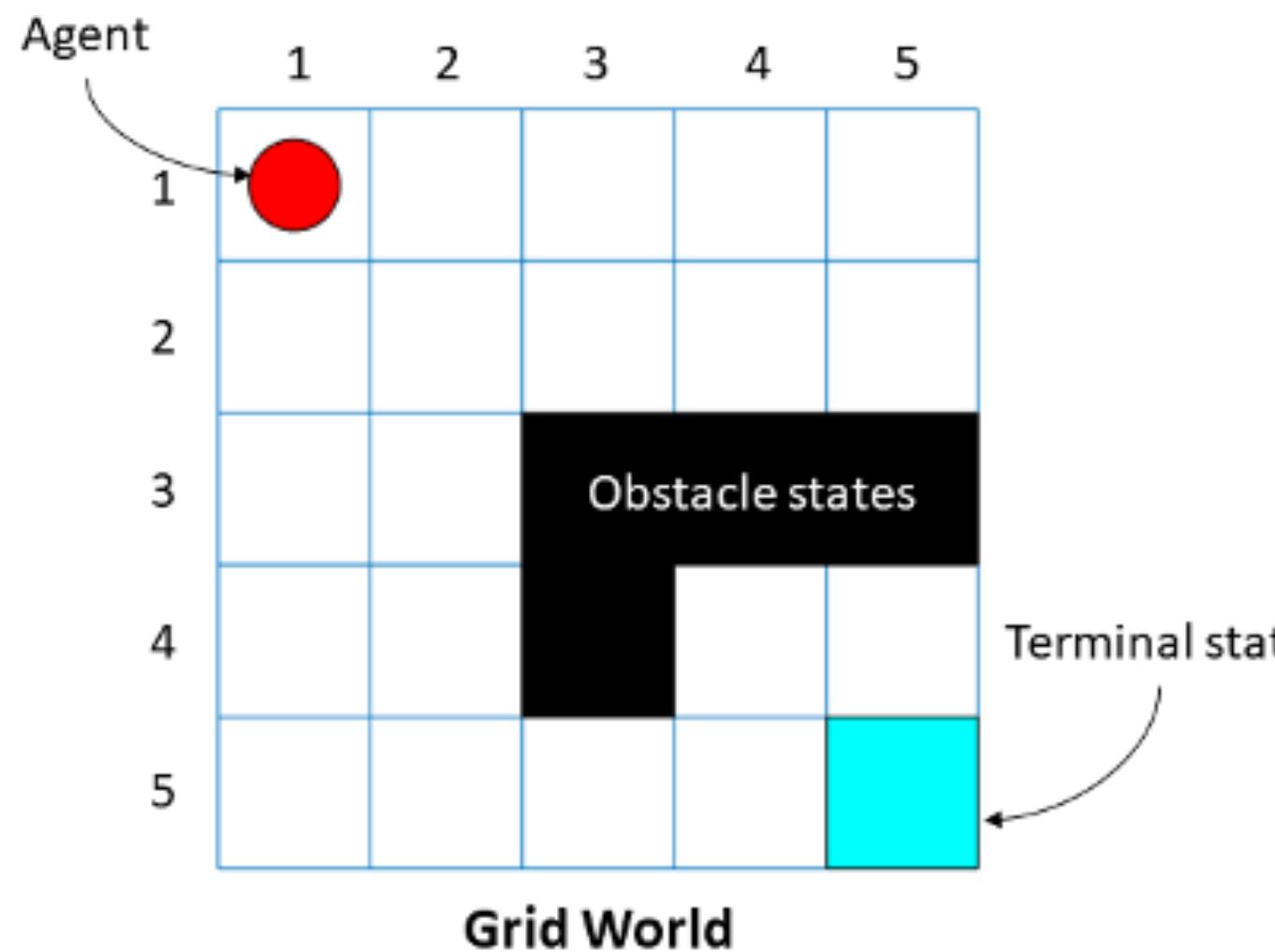
Environment

Markov Decision Process (MDP)

- Markov Principle: simplifying assumption that the system is fully defined by only the previous state $P(s_{t+1} | s_t, a_t)$

What are states?

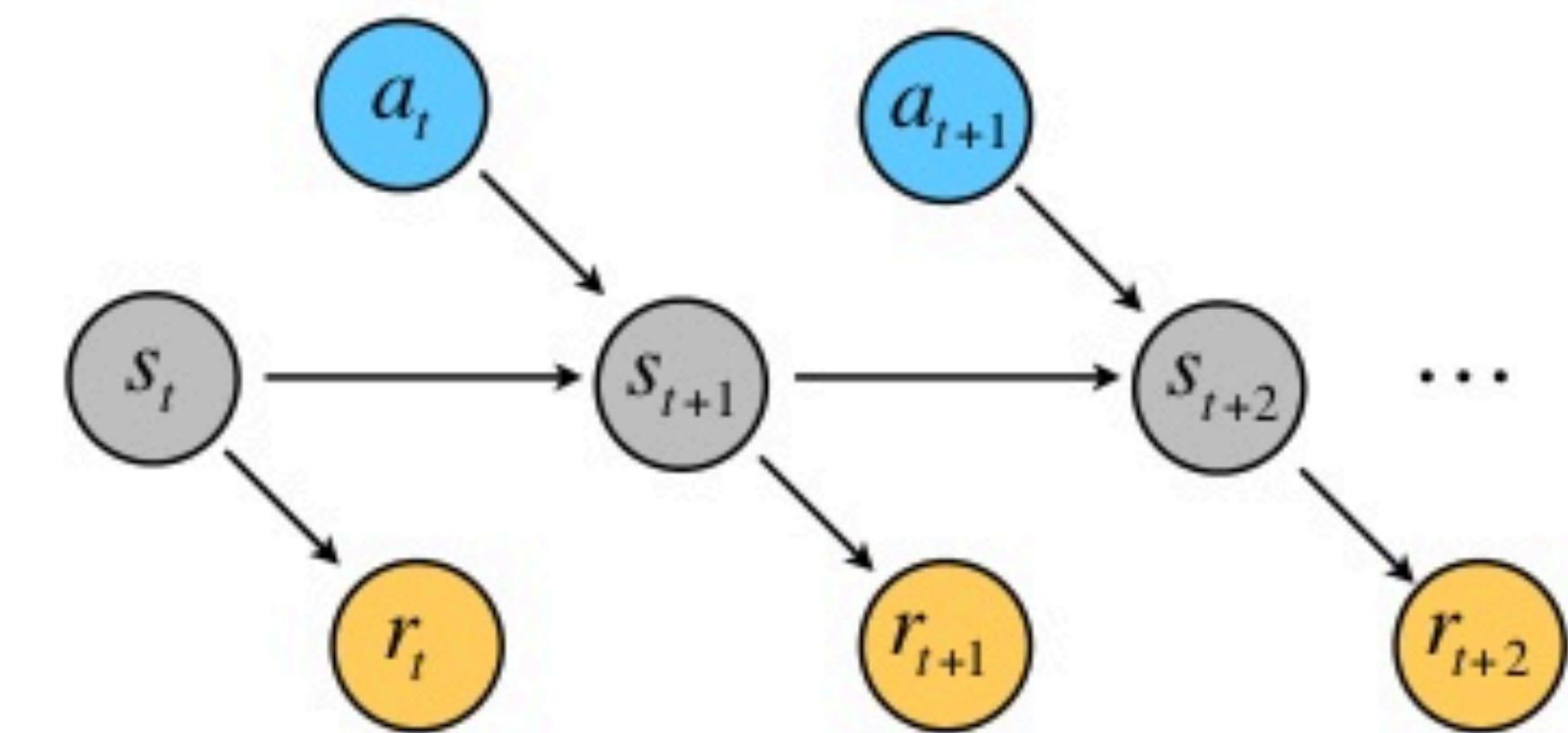
- Locations on a grid, pixels on a screen, feature values, etc...



actions

states

reward



Environment

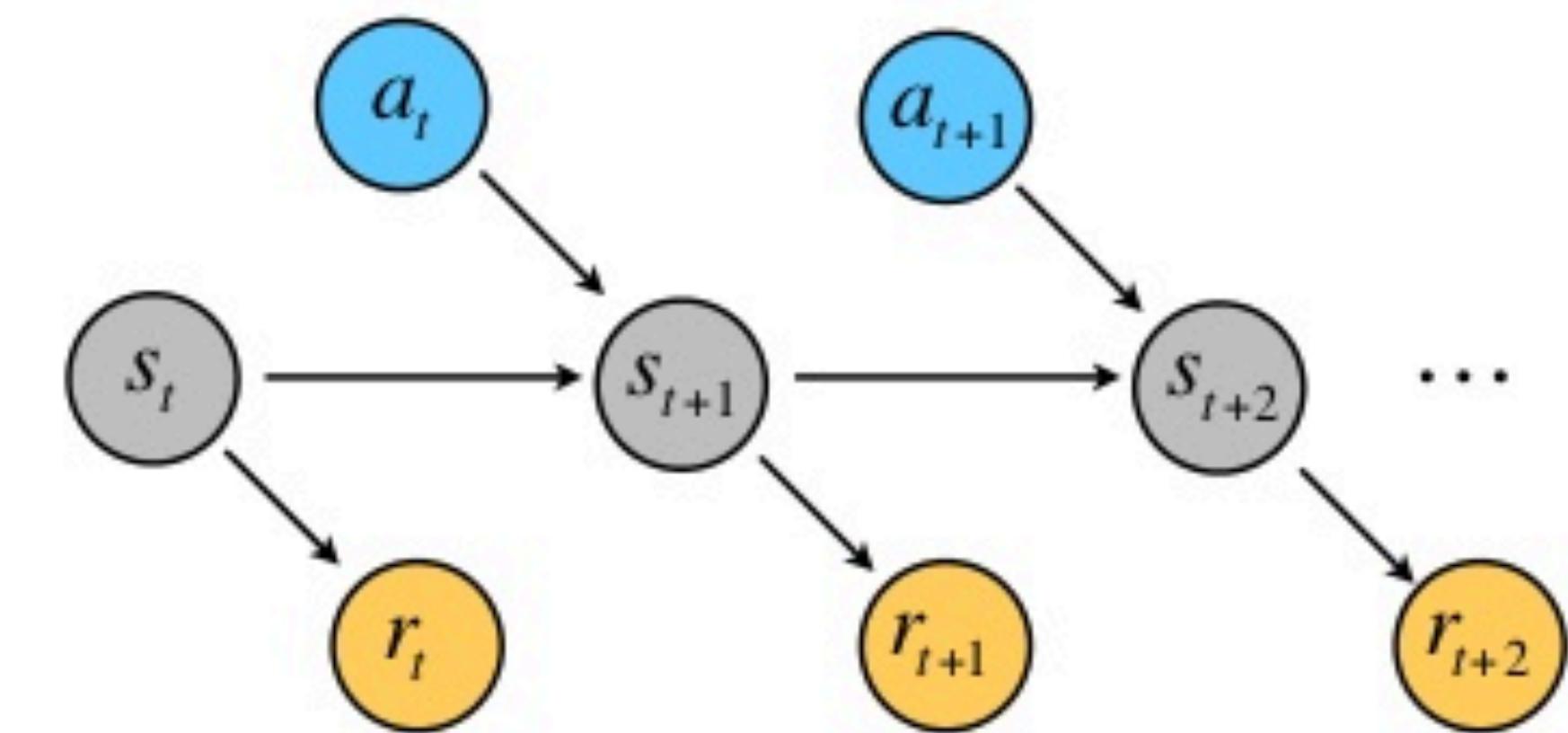
actions

states

reward

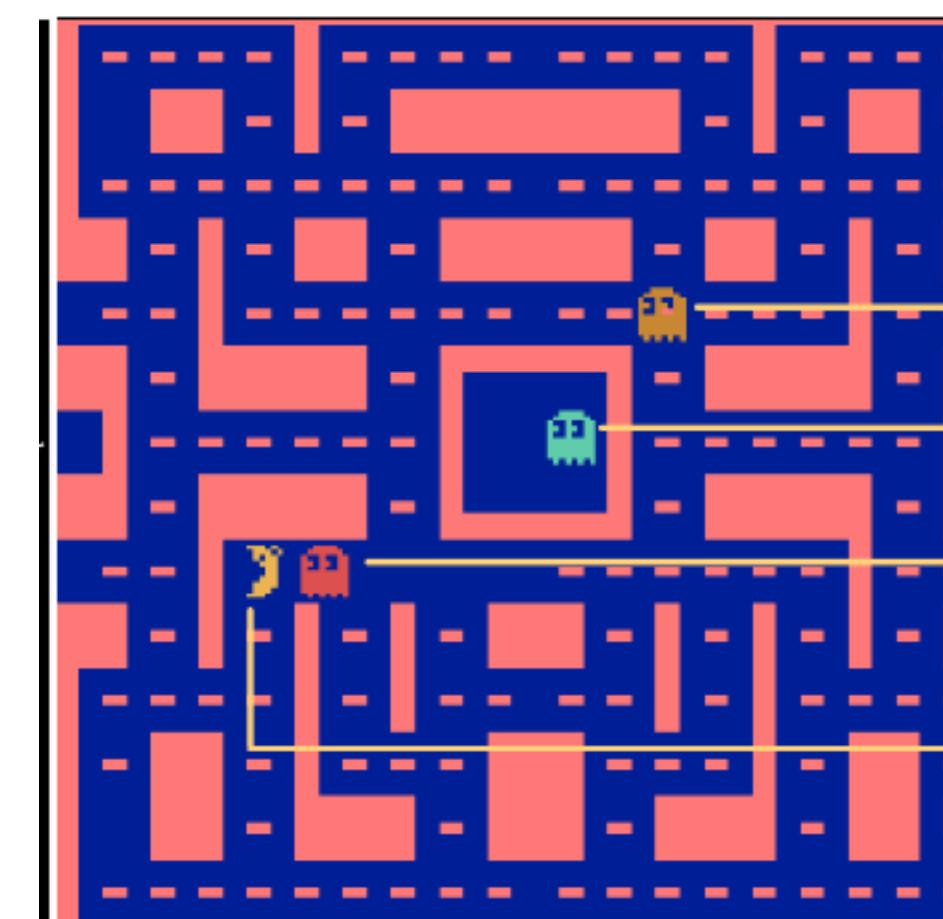
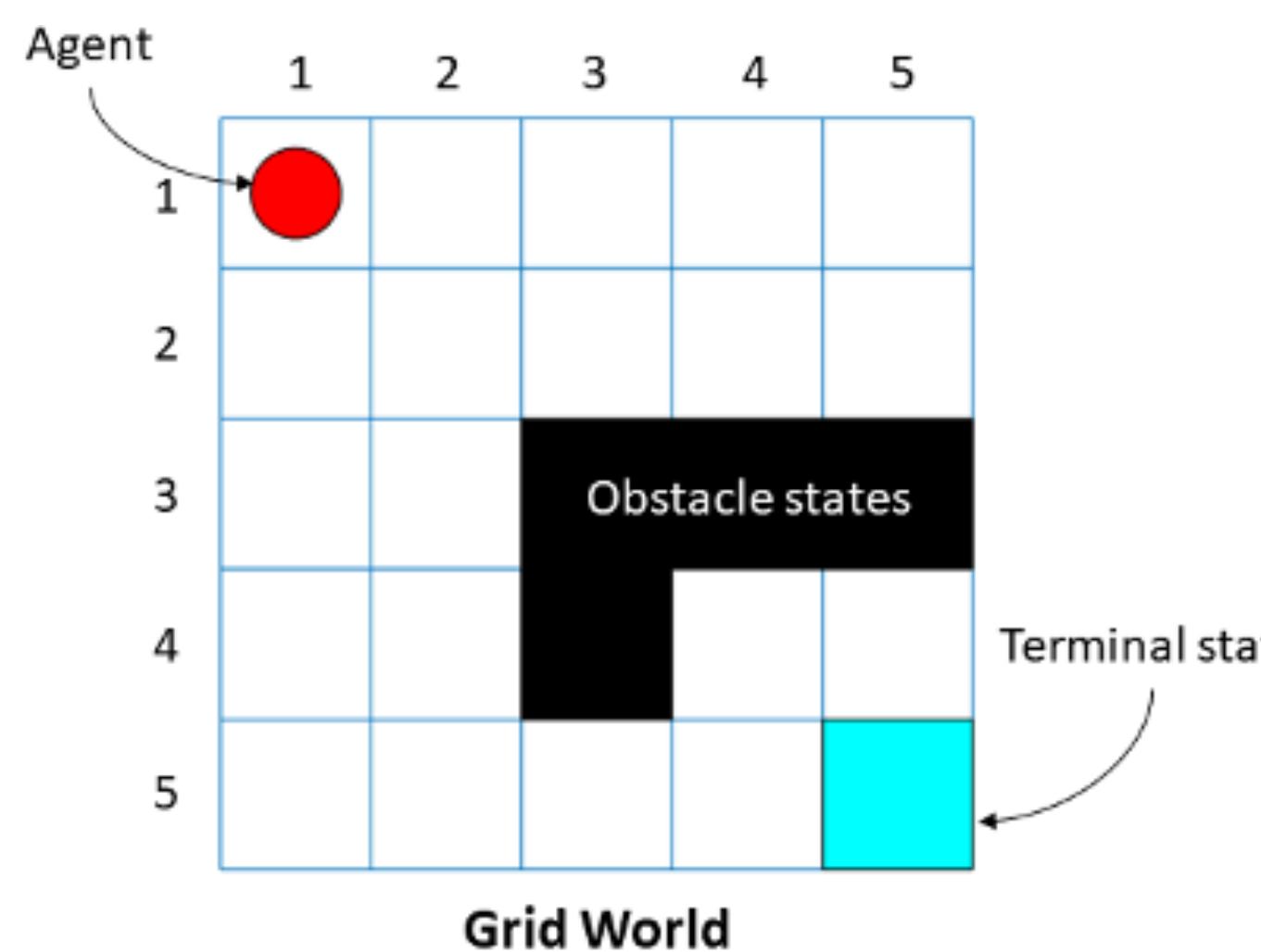
Markov Decision Process (MDP)

- Markov Principle: simplifying assumption that the system is fully defined by only the previous state $P(s_{t+1} | s_t, a_t)$



What are states?

- Locations on a grid, pixels on a screen, feature values, etc...



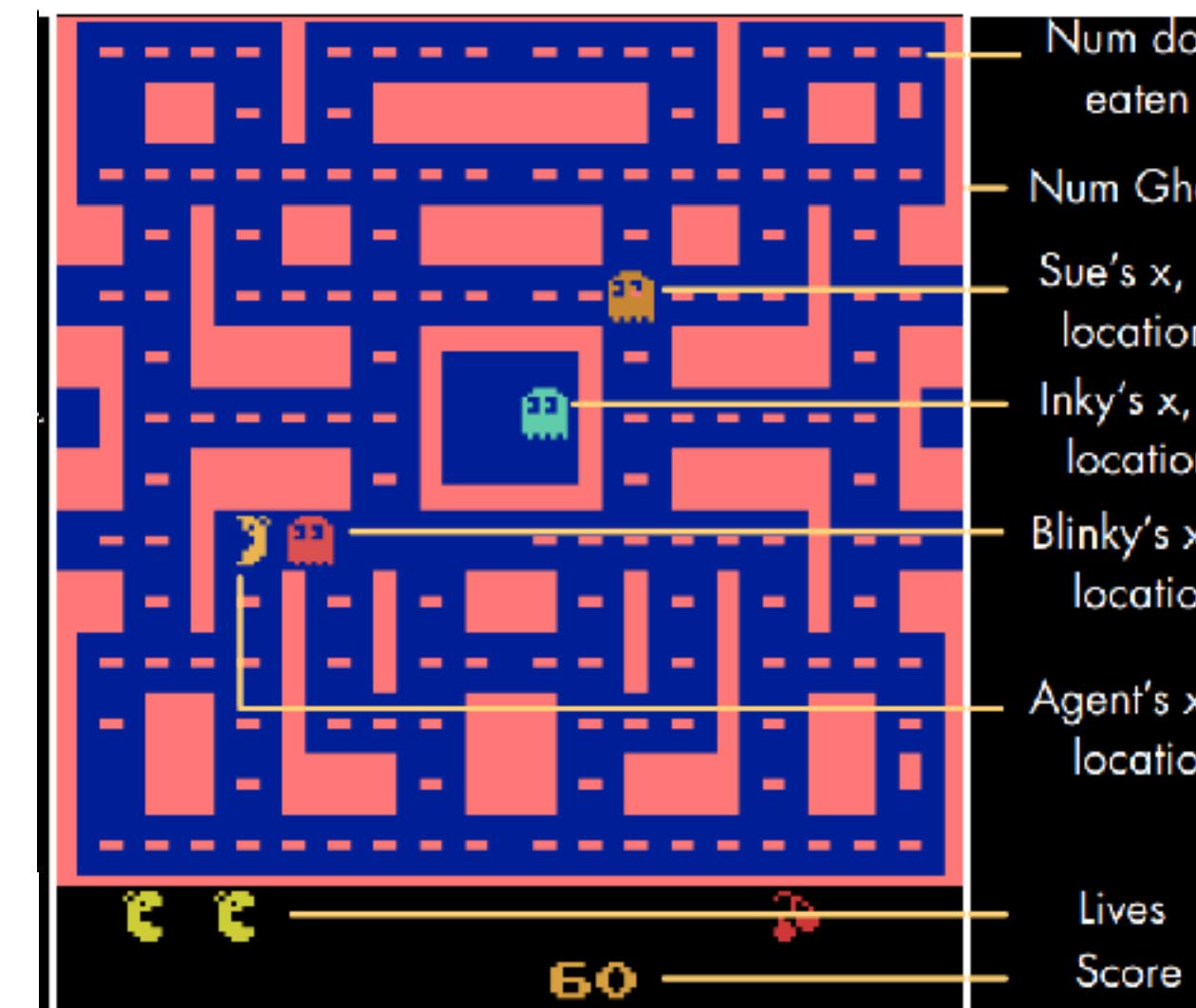
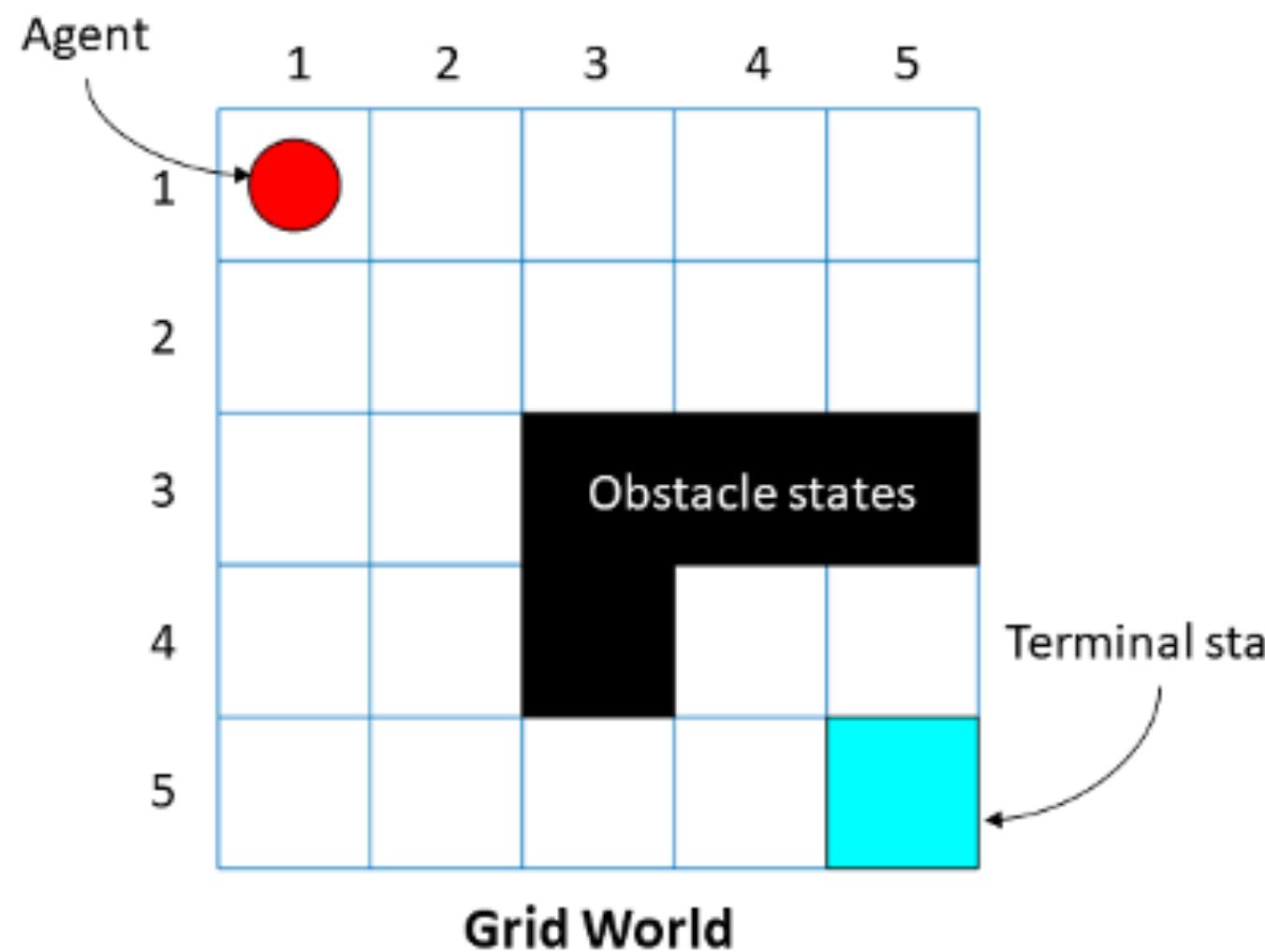
Environment

Markov Decision Process (MDP)

- Markov Principle: simplifying assumption that the system is fully defined by only the previous state $P(s_{t+1} | s_t, a_t)$

What are states?

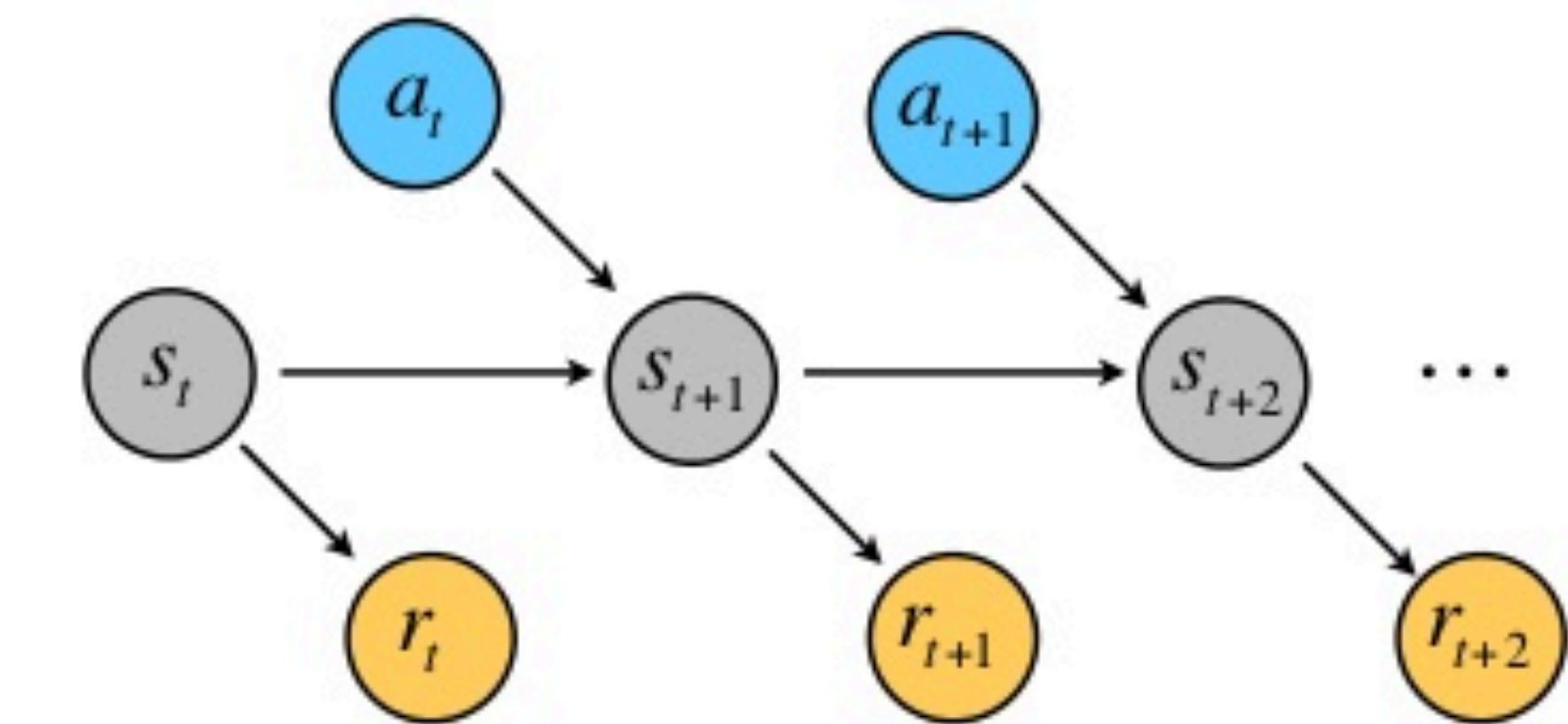
- Locations on a grid, pixels on a screen, feature values, etc...



actions

states

reward



Environment

actions

states

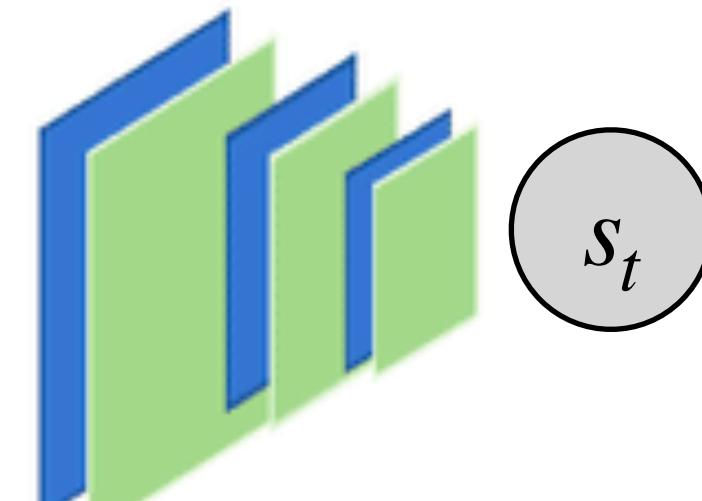
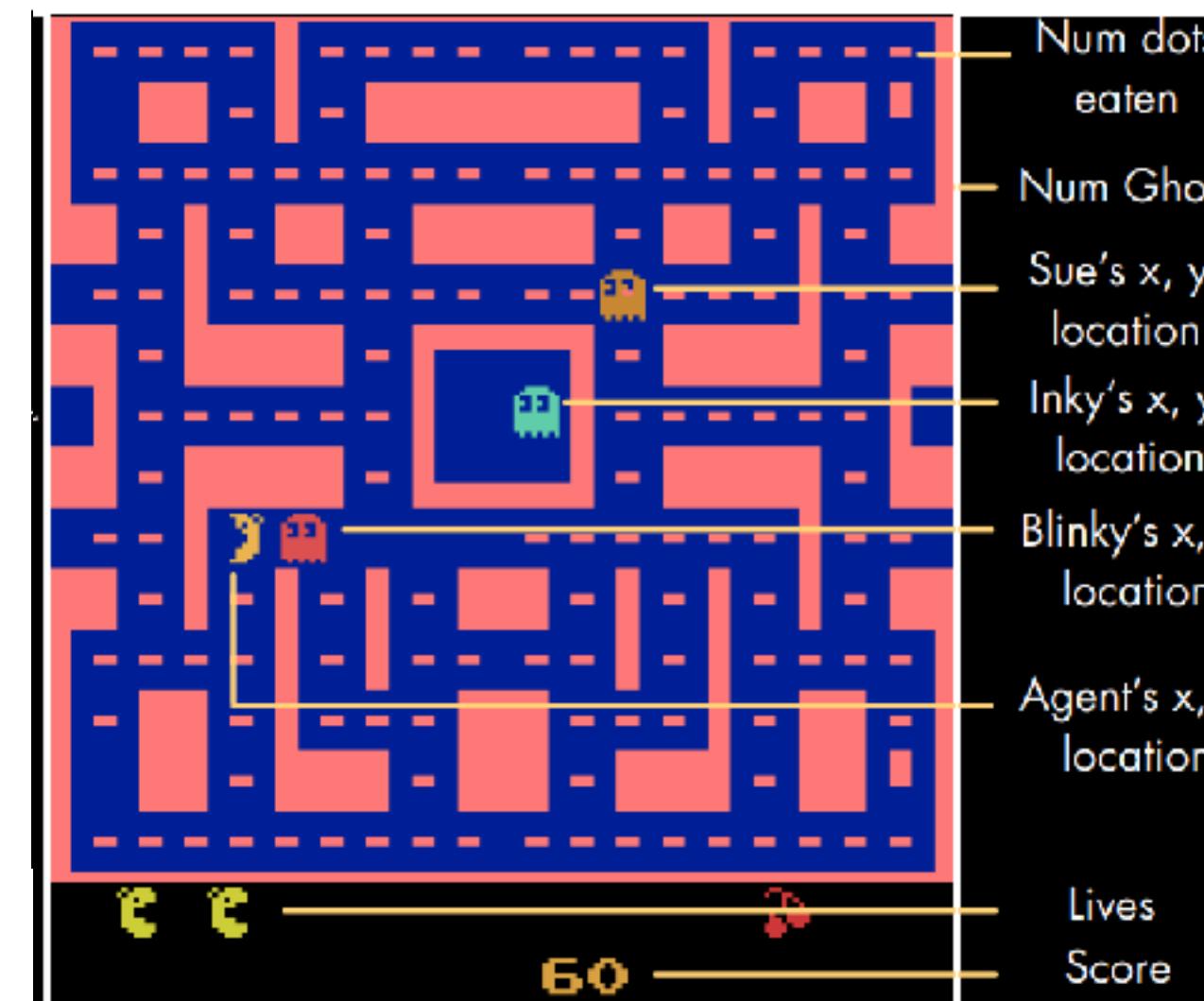
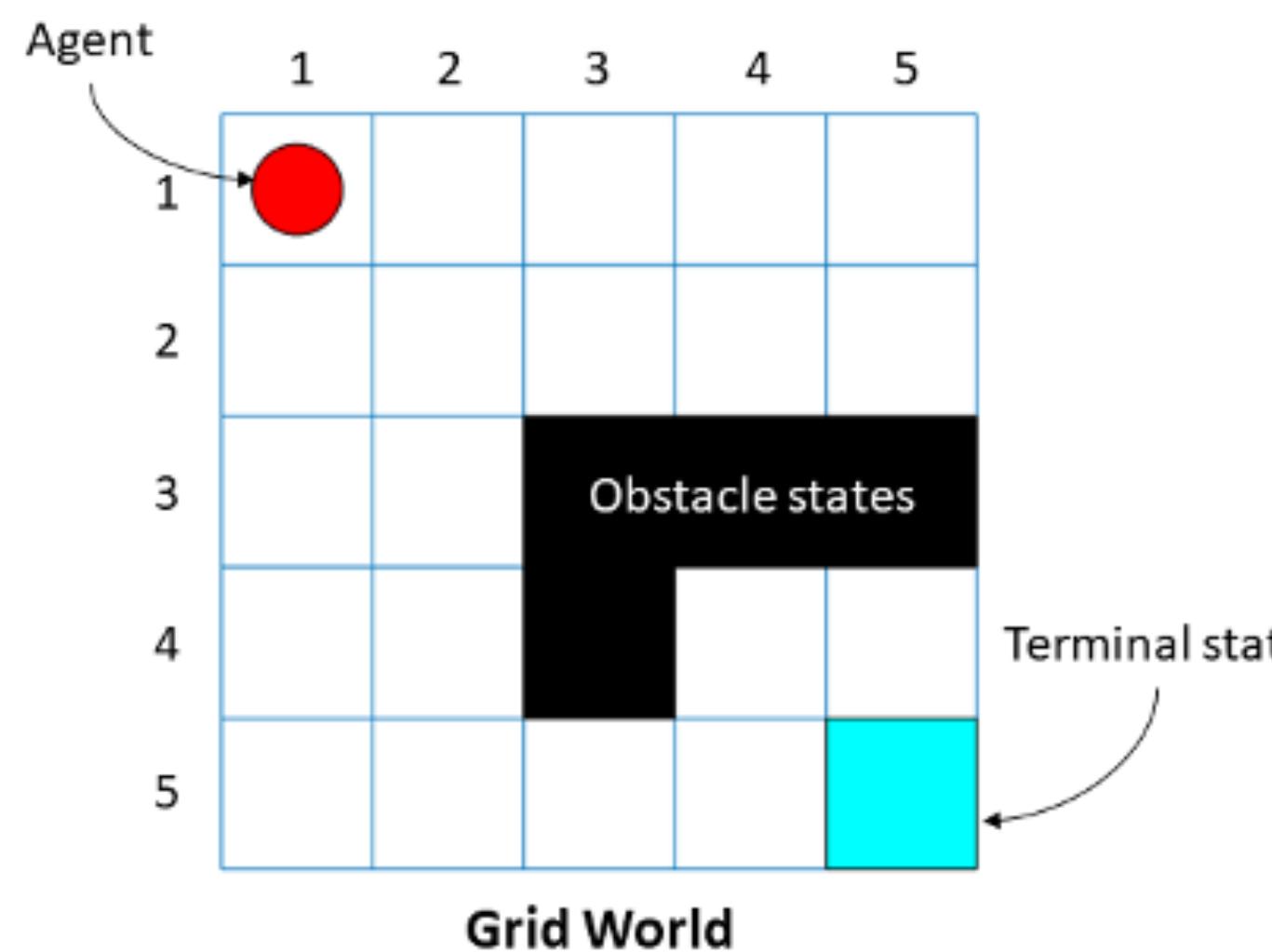
reward

Markov Decision Process (MDP)

- Markov Principle: simplifying assumption that the system is fully defined by only the previous state $P(s_{t+1} | s_t, a_t)$

What are states?

- Locations on a grid, pixels on a screen, feature values, etc...



Agent

Agent

- **Represent Past Experiences**
- **Implement a Policy that Maximizes Reward**

Agent

- **Represent Past Experiences**

- How good is a state? $V(s_t)$



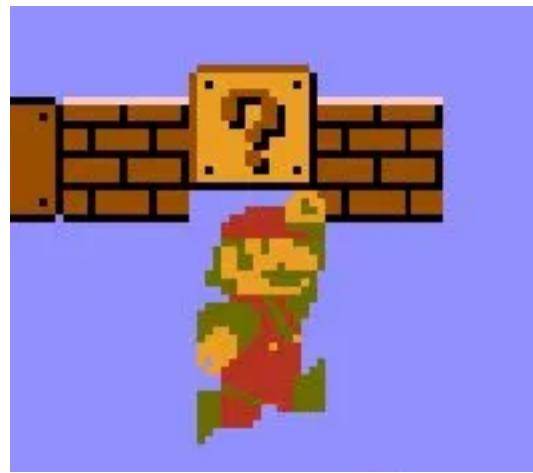
- **Implement a Policy that Maximizes Reward**

Agent

- **Represent Past Experiences**

- How good is a state? $V(s_t)$

- How good is a state-action pair? $Q(s_t, a_t)$

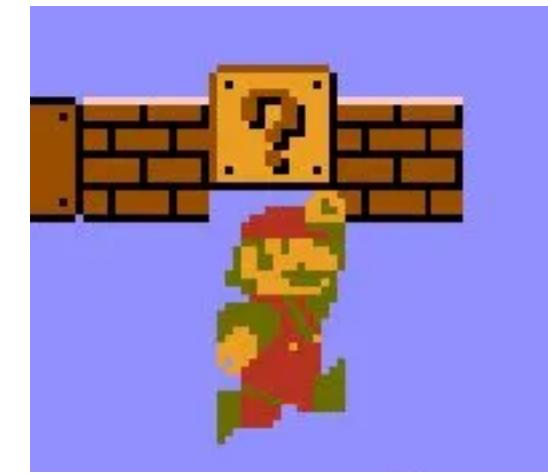


- **Implement a Policy that Maximizes Reward**

Agent

- **Represent Past Experiences**

- How good is a state? $V(s_t)$
- How good is a state-action pair? $Q(s_t, a_t)$
- How good is a *trajectory* $\tau = (s_0, a_0, s_1, a_1, \dots)$?

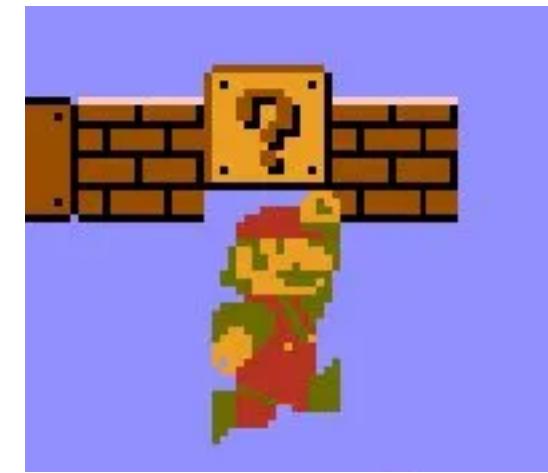


- **Implement a Policy that Maximizes Reward**

Agent

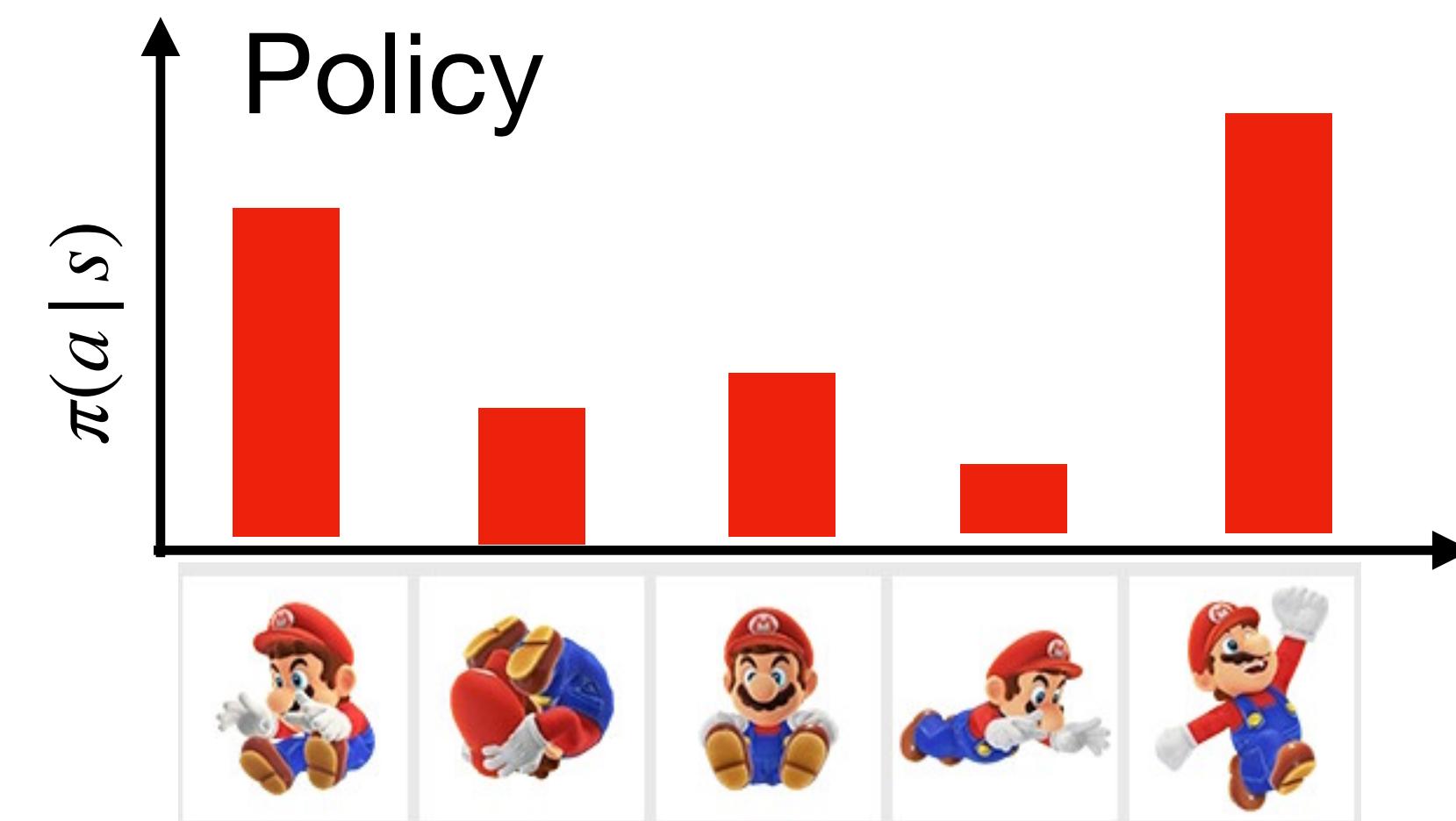
- **Represent Past Experiences**

- How good is a state? $V(s_t)$
- How good is a state-action pair? $Q(s_t, a_t)$
- How good is a *trajectory* $\tau = (s_0, a_0, s_1, a_1, \dots)$?



- **Implement a Policy that Maximizes Reward**

- π defines how to act, where $\pi(a | s)$ is the probability of selecting action a in state s
- sample actions from the policy $a_t \sim \pi$



Normative vs. Descriptive

RL as a **normative** framework:

- How *should* a rational agent behave when learning from the environment?
- Which learning mechanisms and which policies lead to better outcomes?

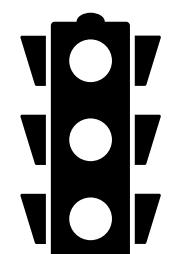
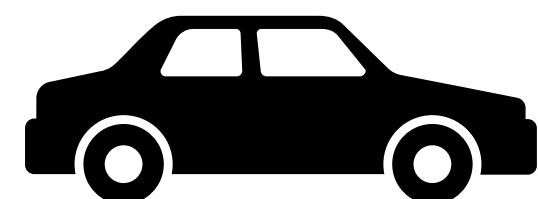
RL as a **descriptive** framework:

- How *does* an agent update beliefs and select actions when learning from the environment?
- Which learning mechanisms and which policies provide better descriptions of behavior

Normative vs. Descriptive

RL as a **normative** framework:

- How *should* a rational agent behave when learning from the environment?
- Which learning mechanisms and which policies lead to better outcomes?



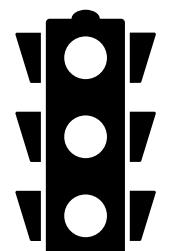
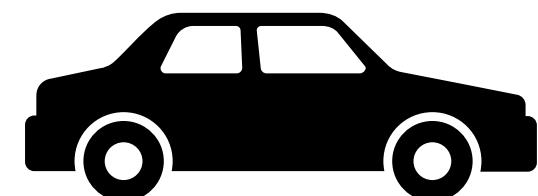
RL as a **descriptive** framework:

- How *does* an agent update beliefs and select actions when learning from the environment?
- Which learning mechanisms and which policies provide better descriptions of behavior

Normative vs. Descriptive

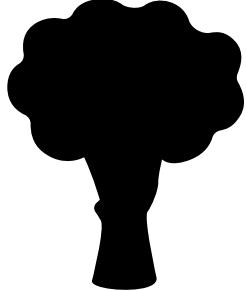
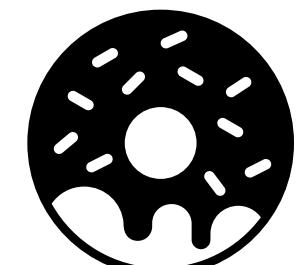
RL as a **normative** framework:

- How *should* a rational agent behave when learning from the environment?
- Which learning mechanisms and which policies lead to better outcomes?



RL as a **descriptive** framework:

- How *does* an agent update beliefs and select actions when learning from the environment?
- Which learning mechanisms and which policies provide better descriptions of behavior

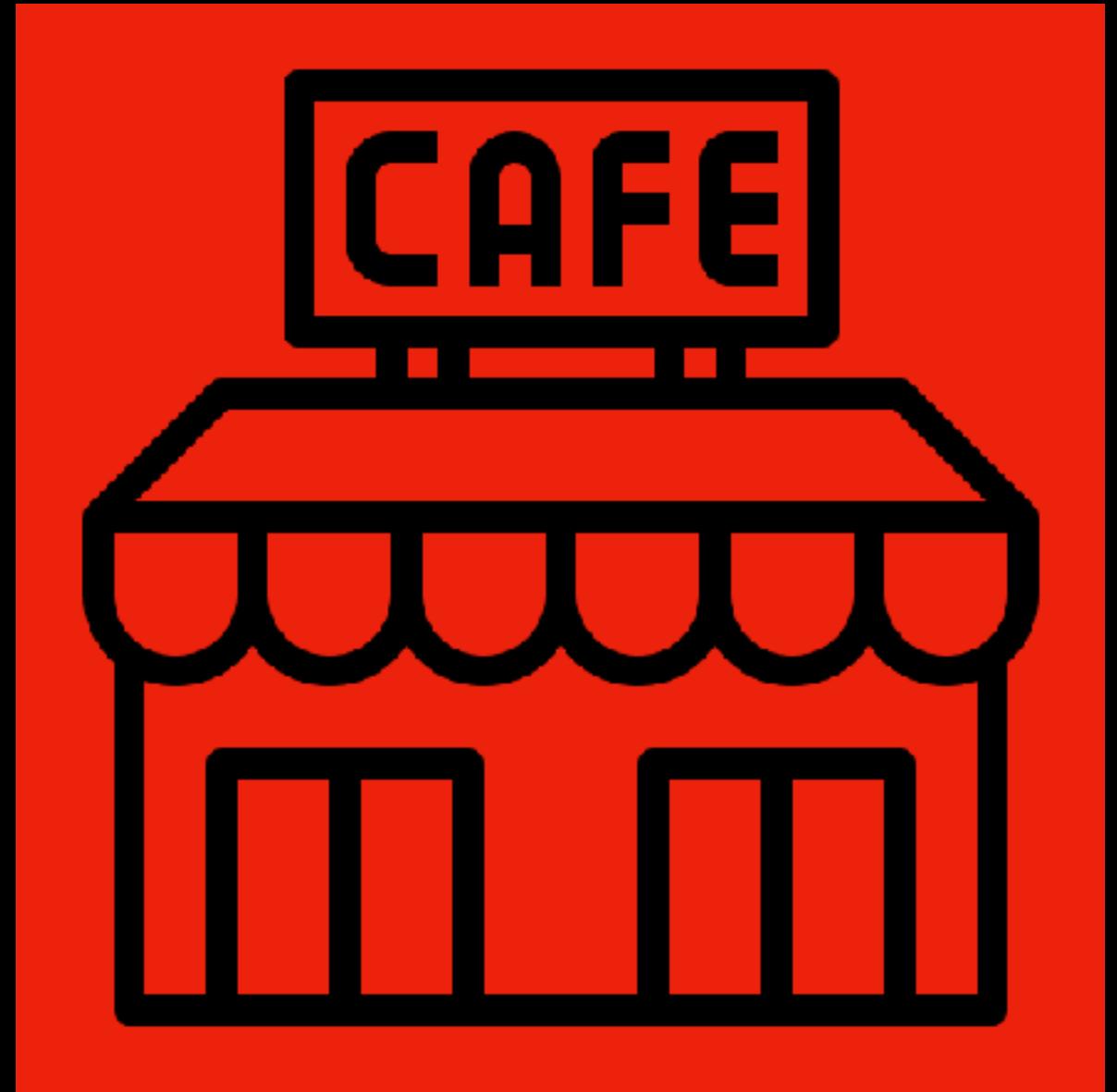


Simplest RL problem & simplest RL model









Options





Options

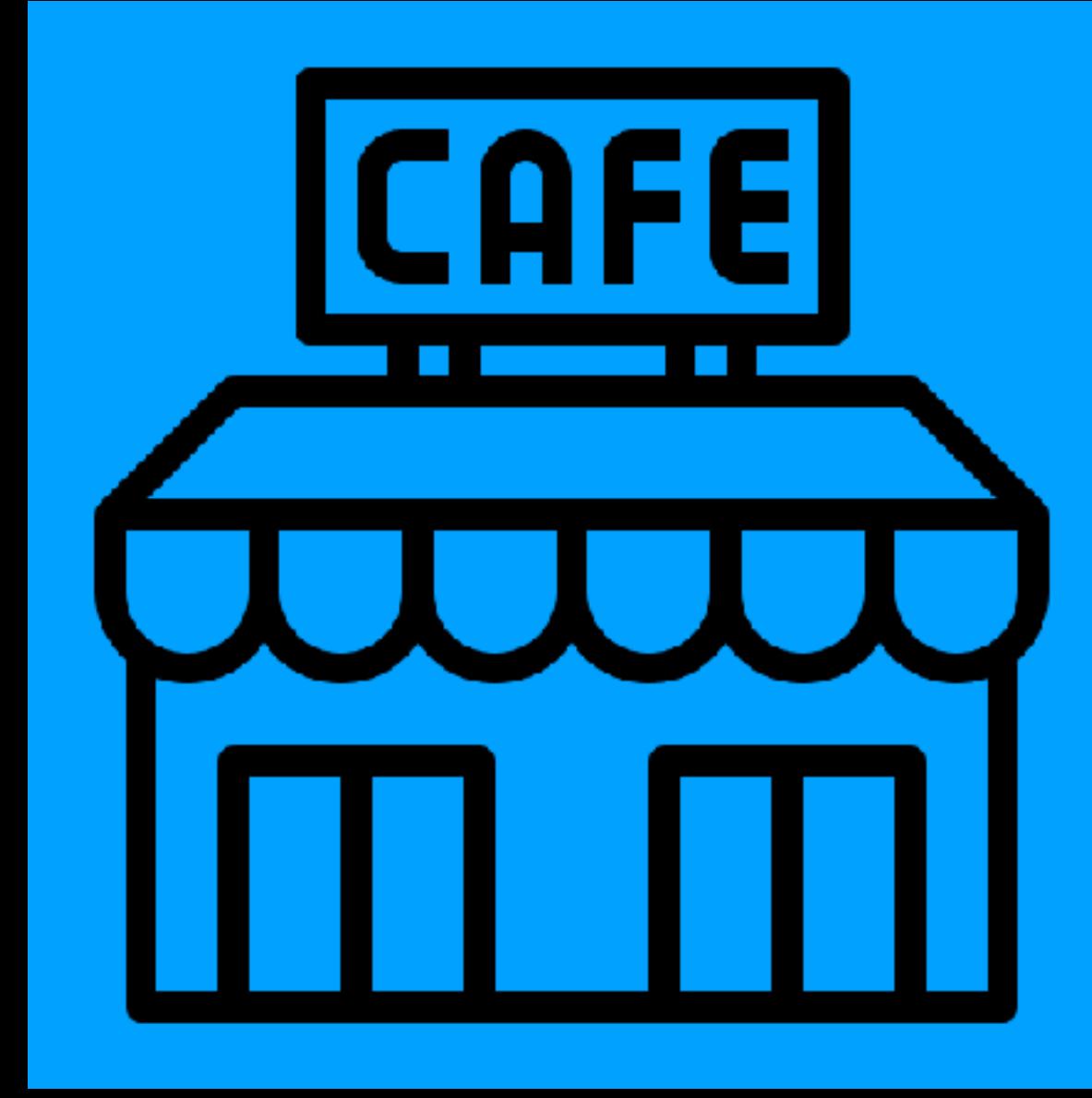


Outcomes





Options



Outcomes





Options



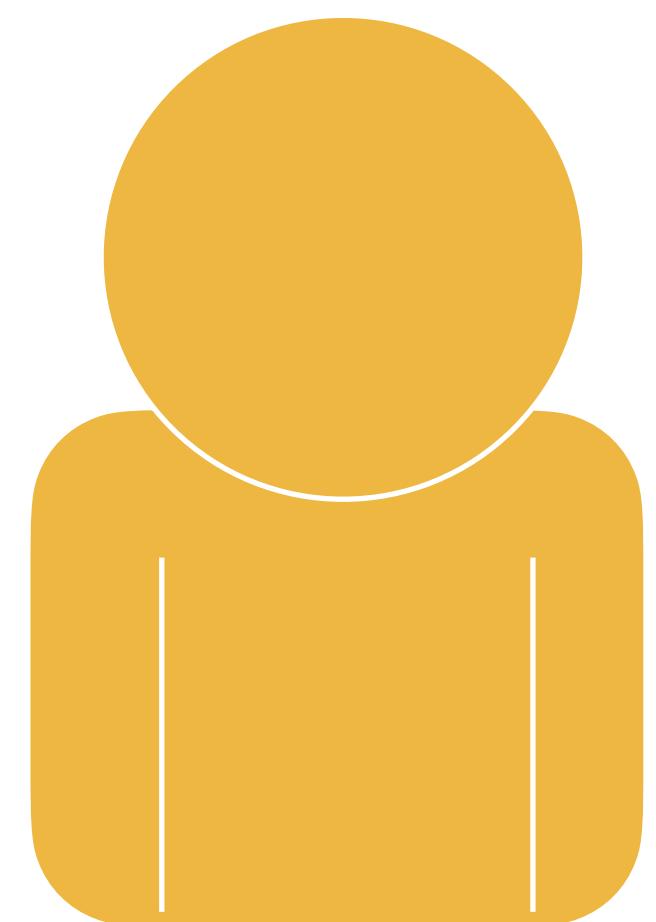
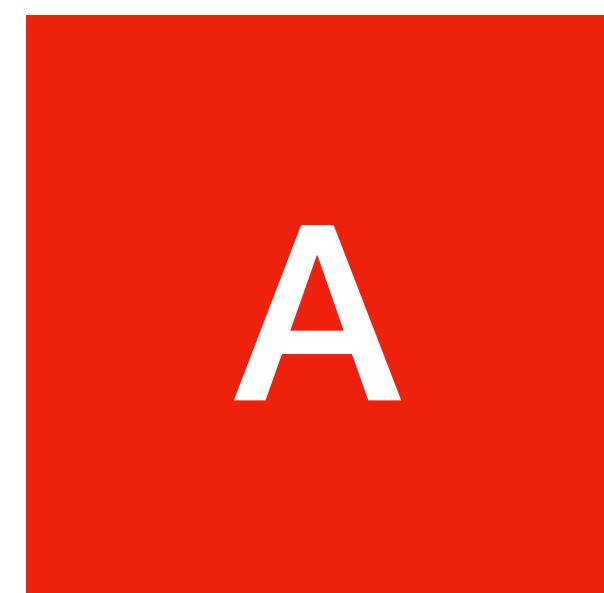
Outcomes



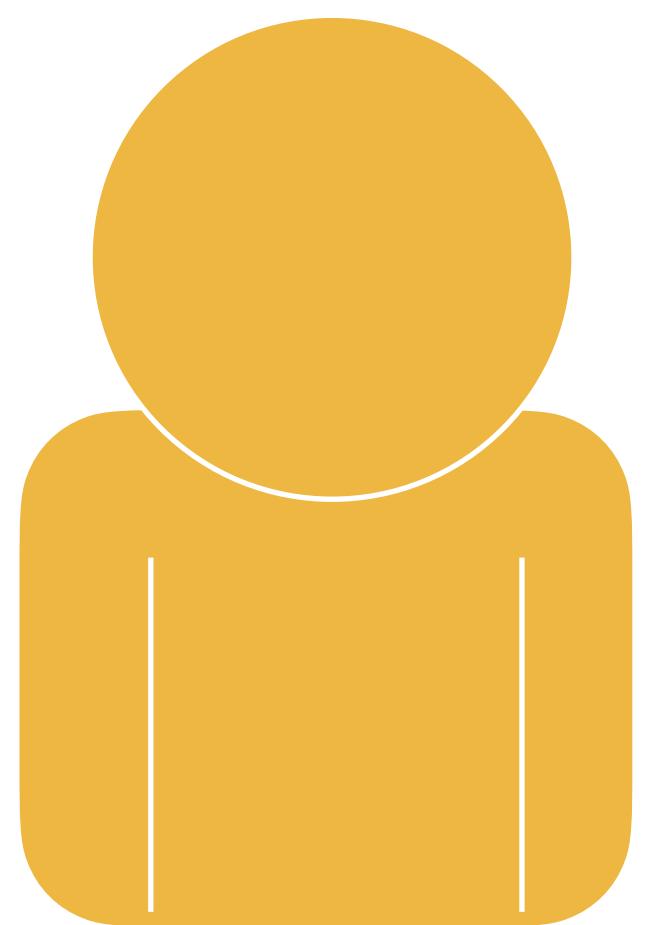
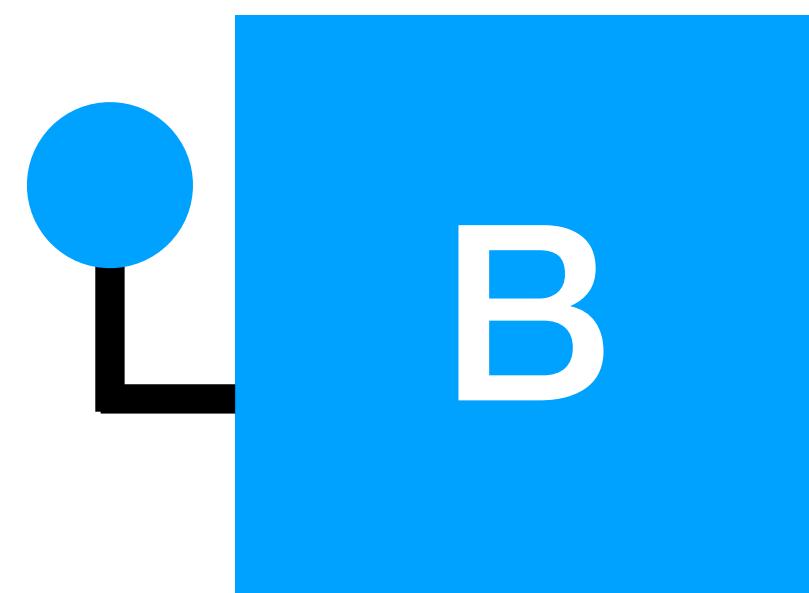
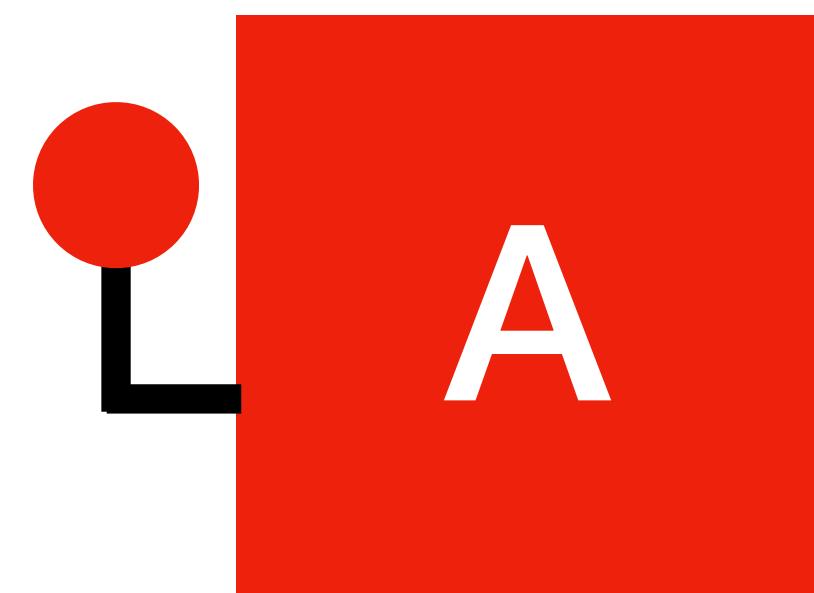




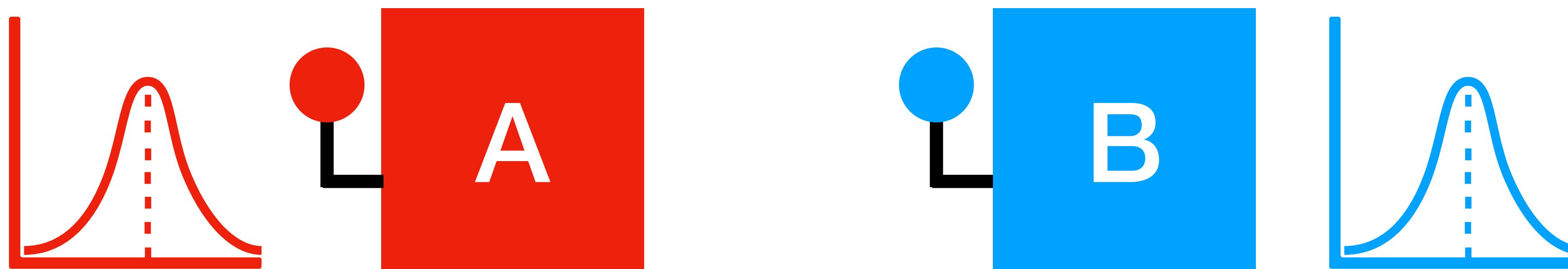
2-Armed Bandit Problem



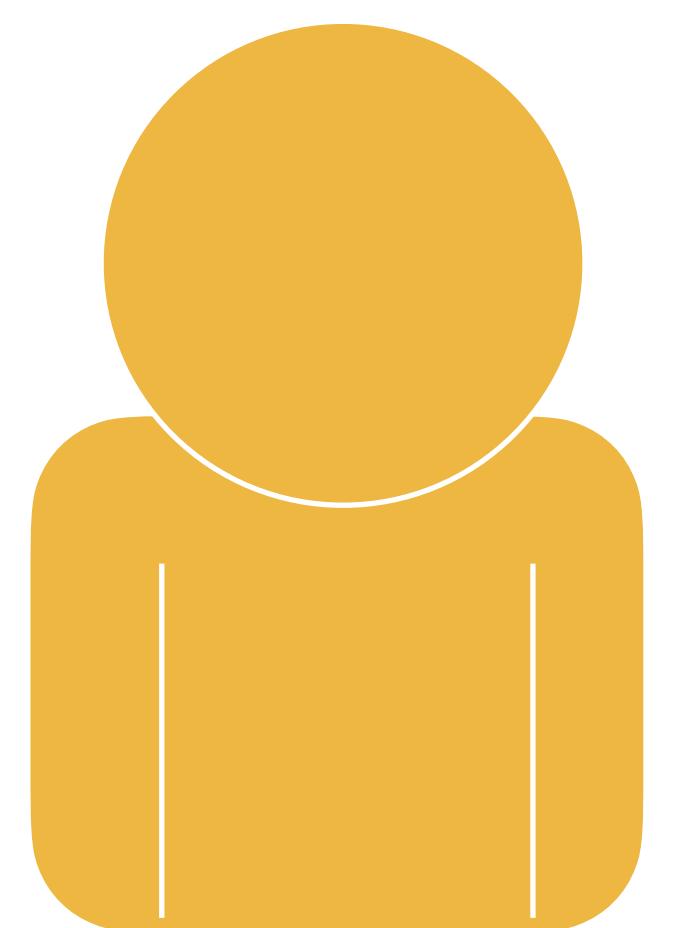
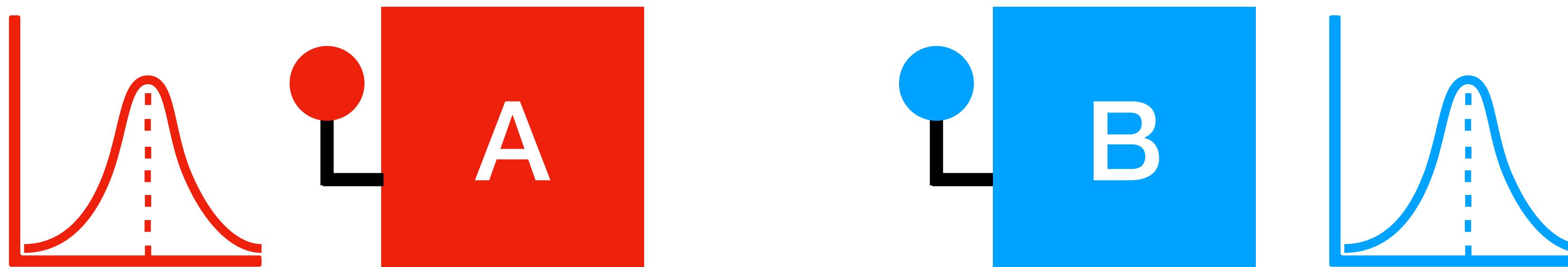
2-Armed Bandit Problem



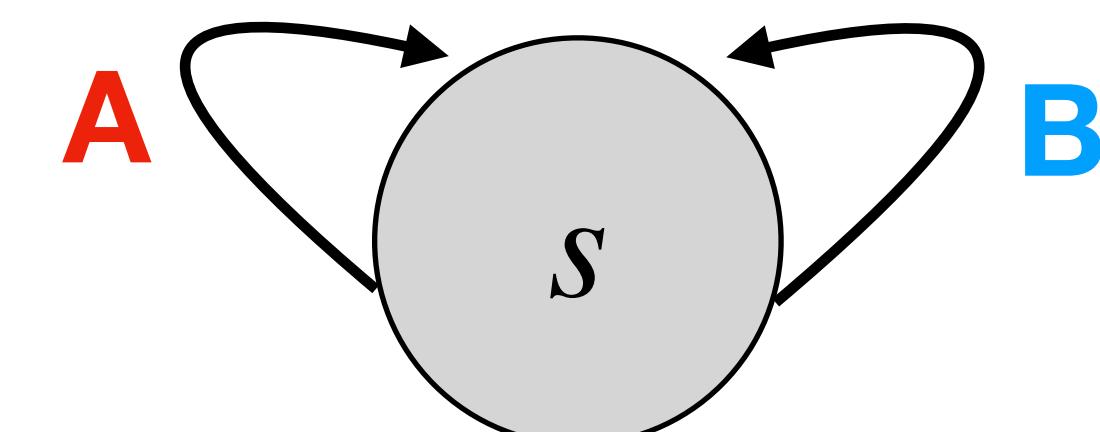
2-Armed Bandit Problem



2-Armed Bandit Problem



Single state problem



Q-Learning (Watkins, 1989)

Value learning

Q-Learning (Watkins, 1989)

Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta [r - Q_t(a)]$$

Q-Learning (Watkins, 1989)

Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta [r - Q_t(a)]$$

↑ ↑
Observed Predicted
reward reward



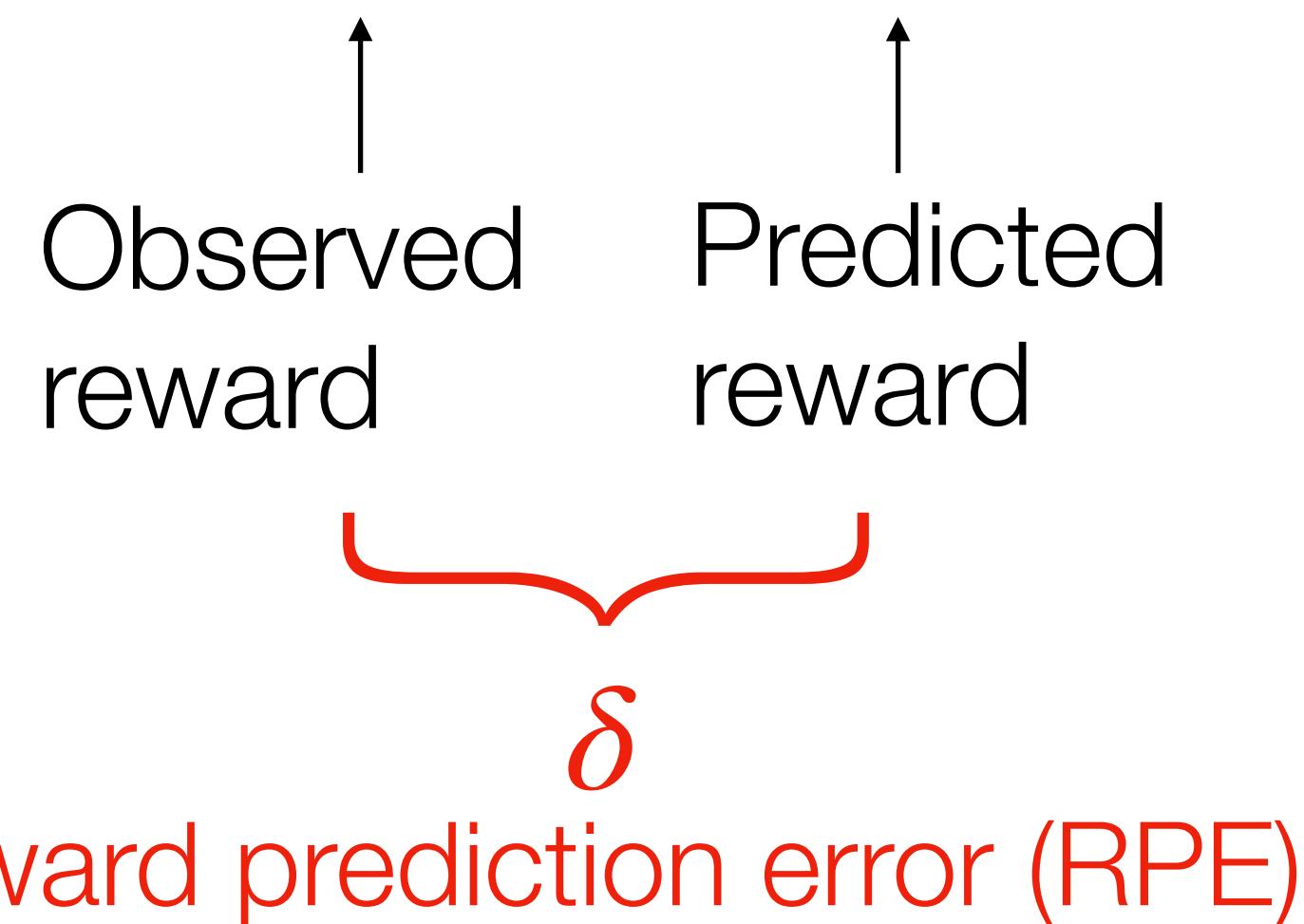
δ

Reward prediction error (RPE)

Q-Learning (Watkins, 1989)

Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta [r - Q_t(a)]$$



The delta-rule of learning:

- Learning occurs only when events violate expectations ($\delta \neq 0$)
- The magnitude of the error corresponds to how much we update our beliefs

Q-Learning (Watkins, 1989)

Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta [r - Q_t(a)]$$

↑ ↑
learning rate Observed reward Predicted reward

δ
Reward prediction error (RPE)

The delta-rule of learning:

- Learning occurs only when events violate expectations ($\delta \neq 0$)
- The magnitude of the error corresponds to how much we update our beliefs

Q-Learning (Watkins, 1989)

Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta [r - Q_t(a)]$$

learning rate Observed reward Predicted reward

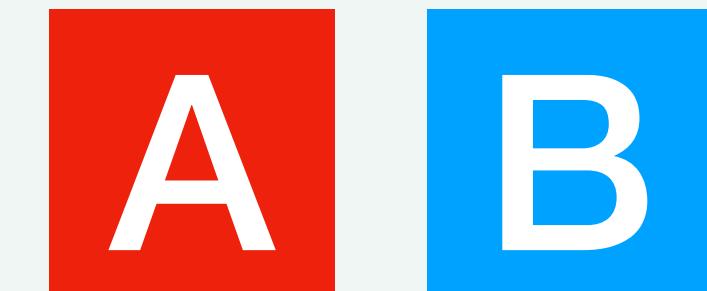
δ
Reward prediction error (RPE)

The diagram illustrates the Q-Learning update rule. It shows the formula $Q_{t+1}(a) \leftarrow Q_t(a) + \eta [r - Q_t(a)]$. Above the formula, three arrows point upwards from below to each term: the first arrow points to η and is labeled "learning rate"; the second arrow points to r and is labeled "Observed reward"; the third arrow points to $Q_t(a)$ and is labeled "Predicted reward". Below these three terms, a red bracket groups them together and points to the variable δ , which is labeled "Reward prediction error (RPE)".

The delta-rule of learning:

- Learning occurs only when events violate expectations ($\delta \neq 0$)
- The magnitude of the error corresponds to how much we update our beliefs

Exercise 1: Compute Q-values



assume $\eta = .9$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	5	
t=2			B	12	
t=3			B	4	
t=4			A	8	

Q-Learning (Watkins, 1989)

Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta [r - Q_t(a)]$$

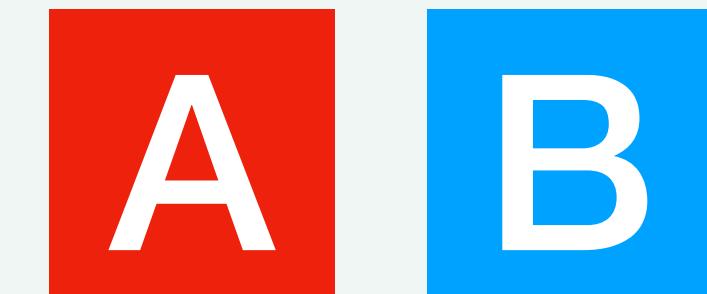
learning rate Observed reward Predicted reward

δ
Reward prediction error (RPE)

The delta-rule of learning:

- Learning occurs only when events violate expectations ($\delta \neq 0$)
- The magnitude of the error corresponds to how much we update our beliefs

Exercise 1: Compute Q-values



assume $\eta = .9$

	$Q(A)$	$Q(B)$	a	r	δ
t=1	0	0	A	5	5
t=2	4.5	0	B	12	12
t=3	4.5	10.8	B	4	-6.8
t=4	4.5	4.68	A	8	3.5

Q-Learning (Watkins, 1989)

Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta [r - Q_t(a)]$$

Q-Learning (Watkins, 1989)

Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta [r - Q_t(a)]$$

Policy

$$P(a) \propto \exp(Q_t(a)/\tau)$$

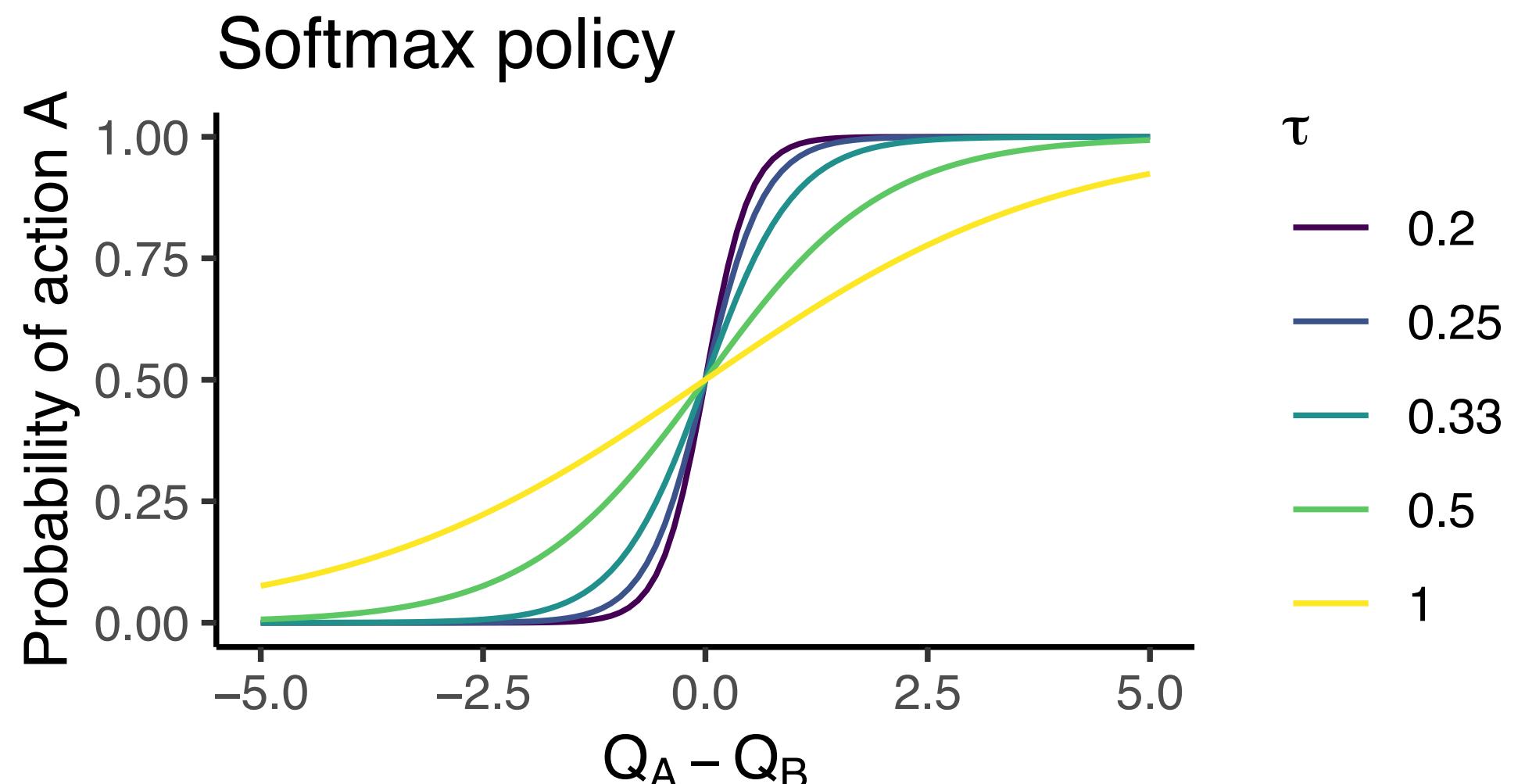
Q-Learning (Watkins, 1989)

Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta [r - Q_t(a)]$$

Policy

$$P(a) \propto \exp(Q_t(a)/\tau)$$



Q-Learning (Watkins, 1989)

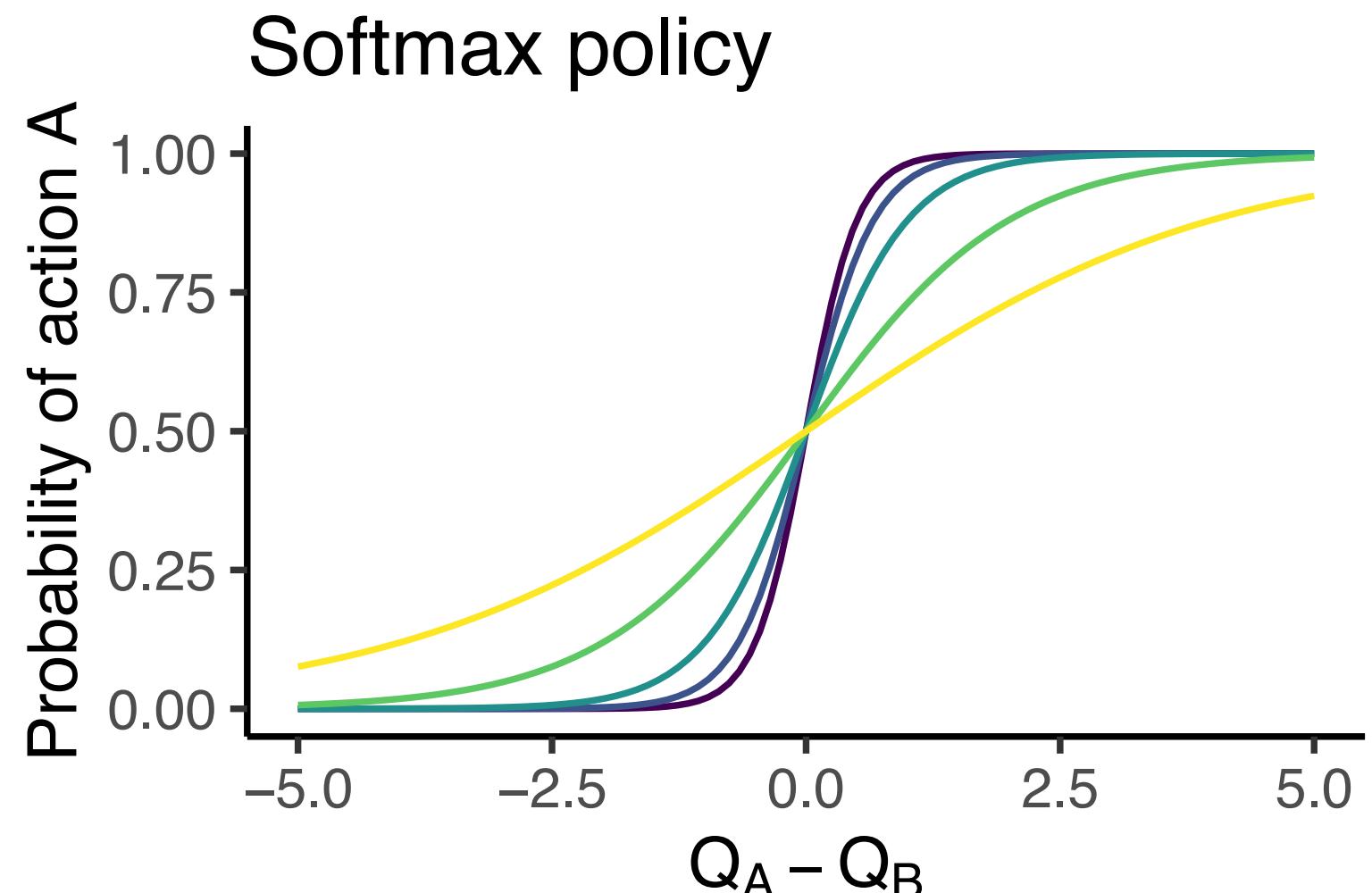
Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta [r - Q_t(a)]$$

Policy

temperature

$$P(a) \propto \exp(Q_t(a)/\tau) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$



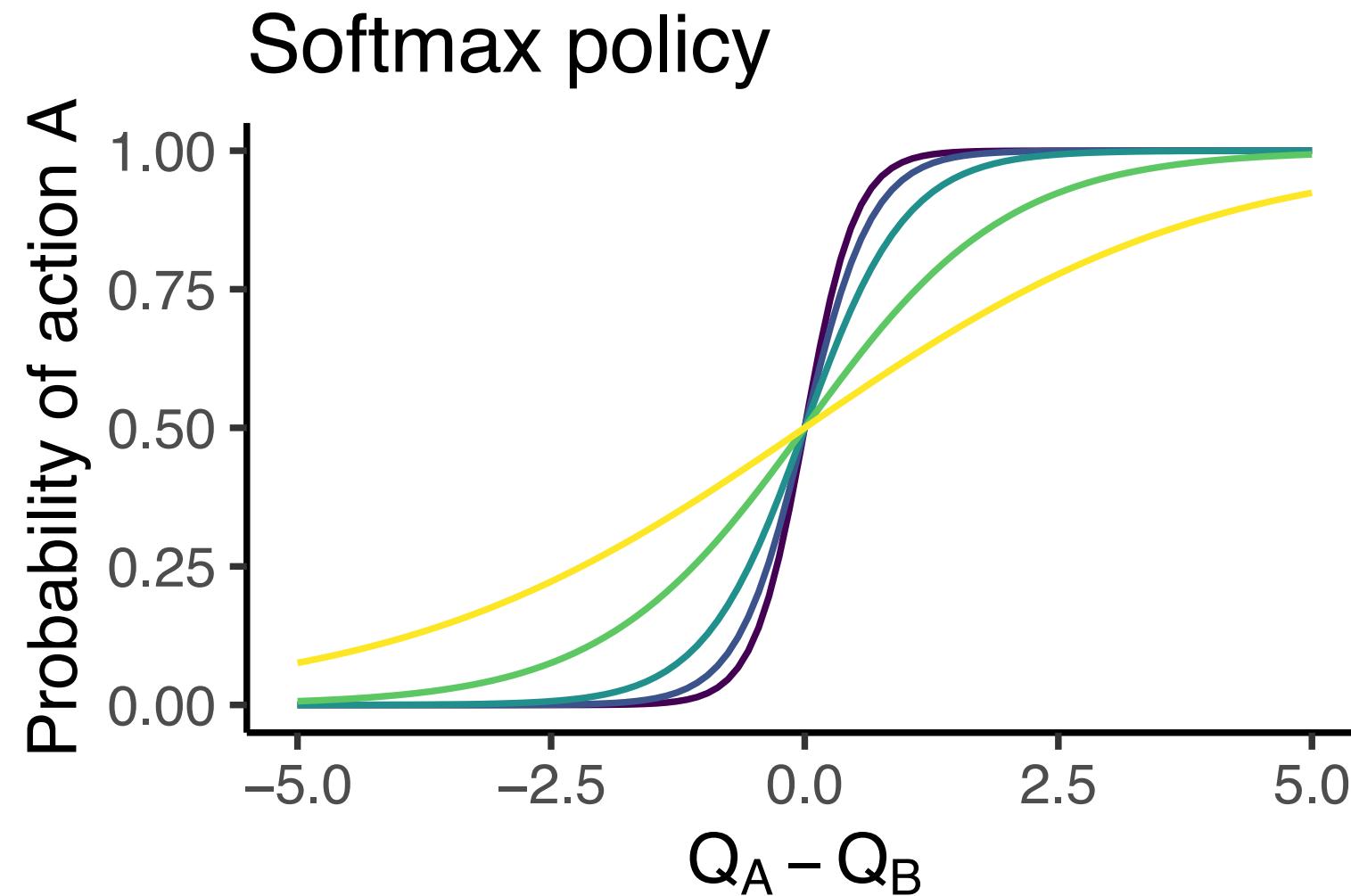
Q-Learning (Watkins, 1989)

Value learning

$$Q_{t+1}(a) \leftarrow Q_t(a) + \eta [r - Q_t(a)]$$

Policy

$$P(a) \propto \exp(Q_t(a)/\tau) = \frac{\exp(Q_t(a)/\tau)}{\sum_i \exp(Q_t(a_i)/\tau)}$$

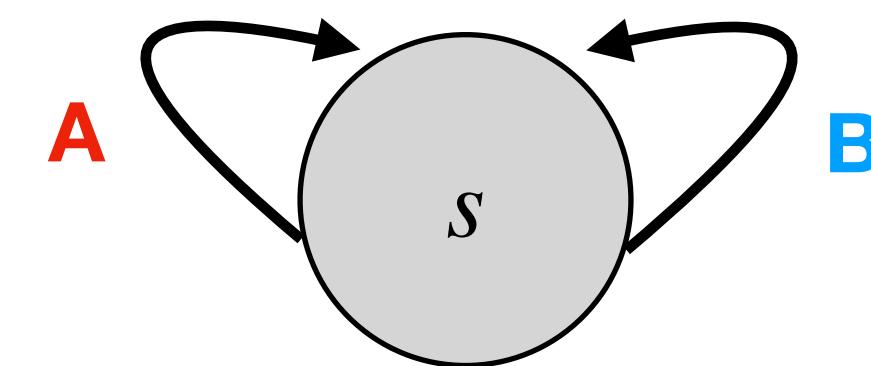
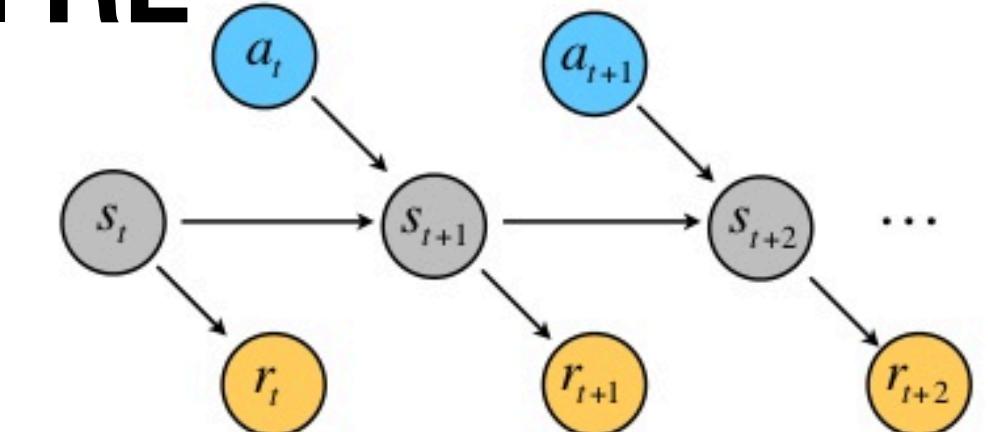


Exercise 2: Sample actions from policy

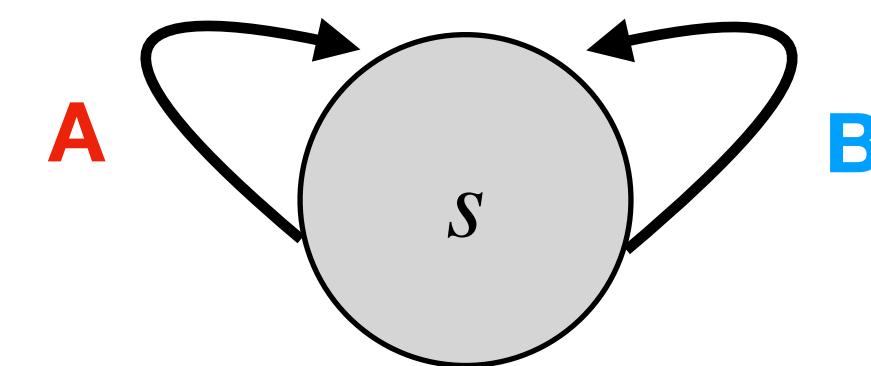
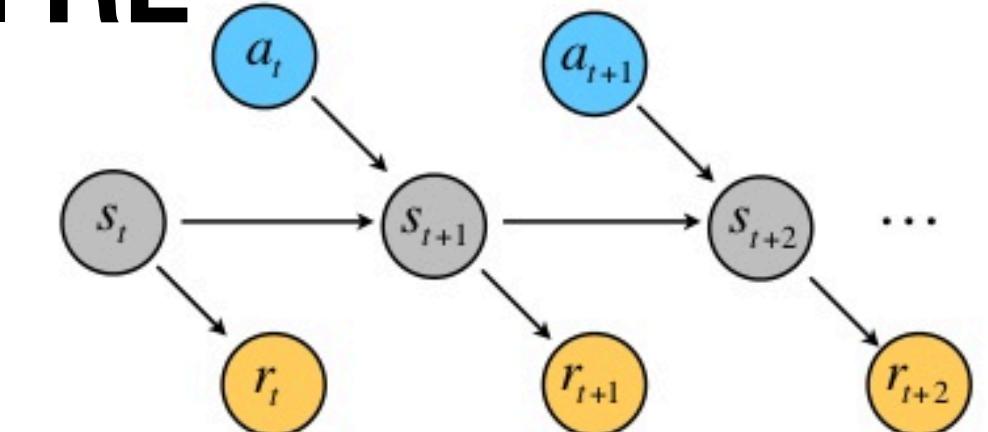
assume:
 $\eta = .9$
 $\tau = .25$

	A	B	$Q(A)$	$Q(B)$	a	r	δ
t=1			0	0			
t=2							
t=3							
t=4							

Moving back to the general RL problem

	Bandit problem 	General RL 
Examples	choosing restaurants, buying a phone, funding research, A/B testing of advertisements,	robot bartender, playing games, self-driving car, chatbots, etc...
Value representations	$Q(a)$	$Q(a, s)$ or $V(s)$
Policy	$\pi(a)$	$\pi(a s)$
Planning?	Not needed	Important!
Long-term Value?	$V(a) = Q(a)$	

Moving back to the general RL problem

	Bandit problem 	General RL 
Examples	choosing restaurants, buying a phone, funding research, A/B testing of advertisements,	robot bartender, playing games, self-driving car, chatbots, etc...
Value representations	$Q(a)$	$Q(a, s)$ or $V(s)$
Policy	$\pi(a)$	$\pi(a s)$
Planning?	Not needed	Important!
Long-term Value?	$V(a) = Q(a)$?

Challenge 1: Credit assignment

- How do we assign credit to actions that are responsible for future reward?
(Minsky, 1961)



Challenge 1: Credit assignment

- How do we assign credit to actions that are responsible for future reward?
(Minsky, 1961)
- **Temporal Difference (TD) Learning** defines a value function that augments reward expectations with the **discounted value** of the next state

$$V(s) \leftarrow V(s) + \eta \left(r + \underline{\gamma V(s')} - V(s) \right)$$

discounted future value



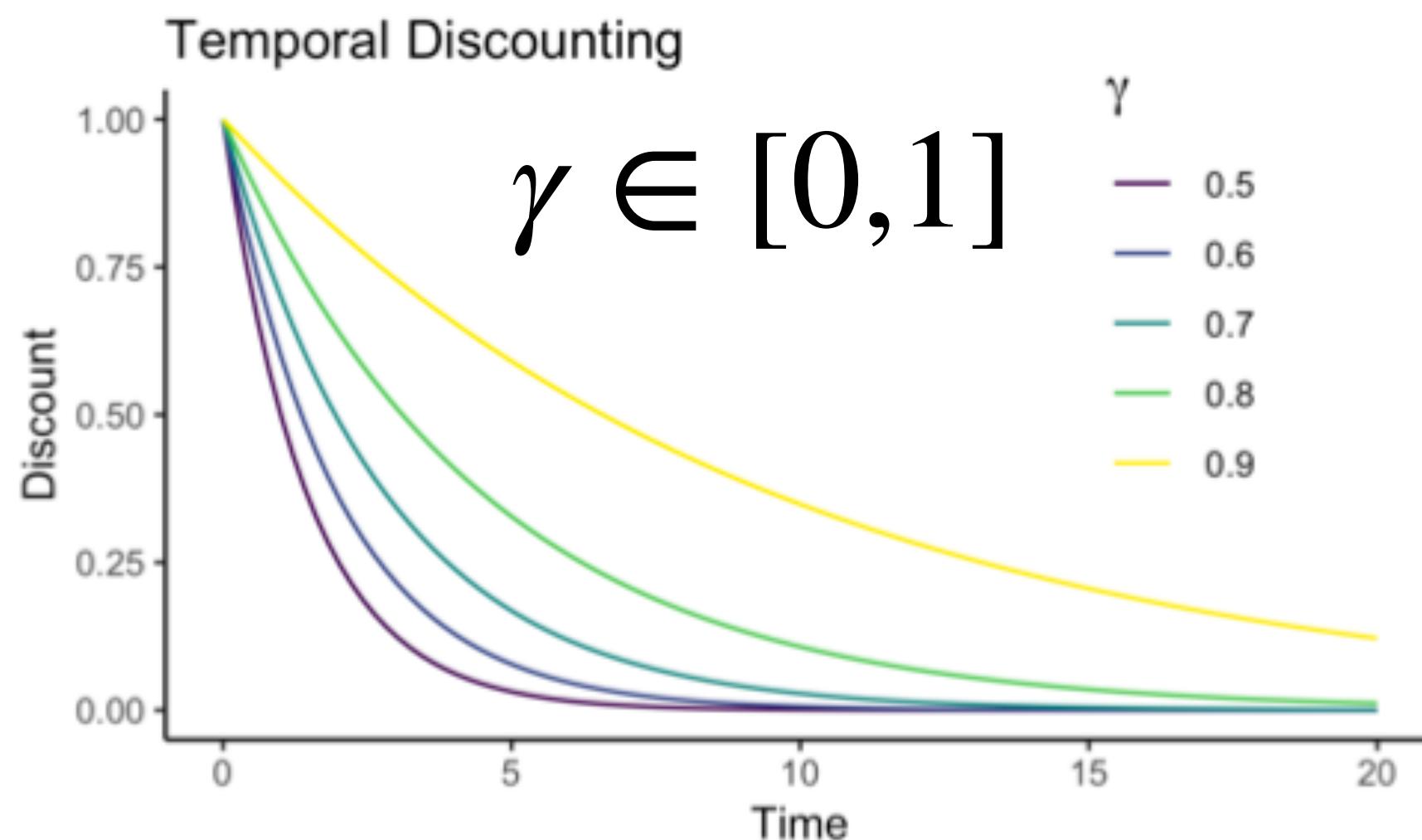
Challenge 1: Credit assignment

- How do we assign credit to actions that are responsible for future reward?
(Minsky, 1961)
- **Temporal Difference (TD) Learning** defines a value function that augments reward expectations with the **discounted value** of the next state



$$V(s) \leftarrow V(s) + \eta \left(r + \underline{\gamma V(s')} - V(s) \right)$$

discounted future value

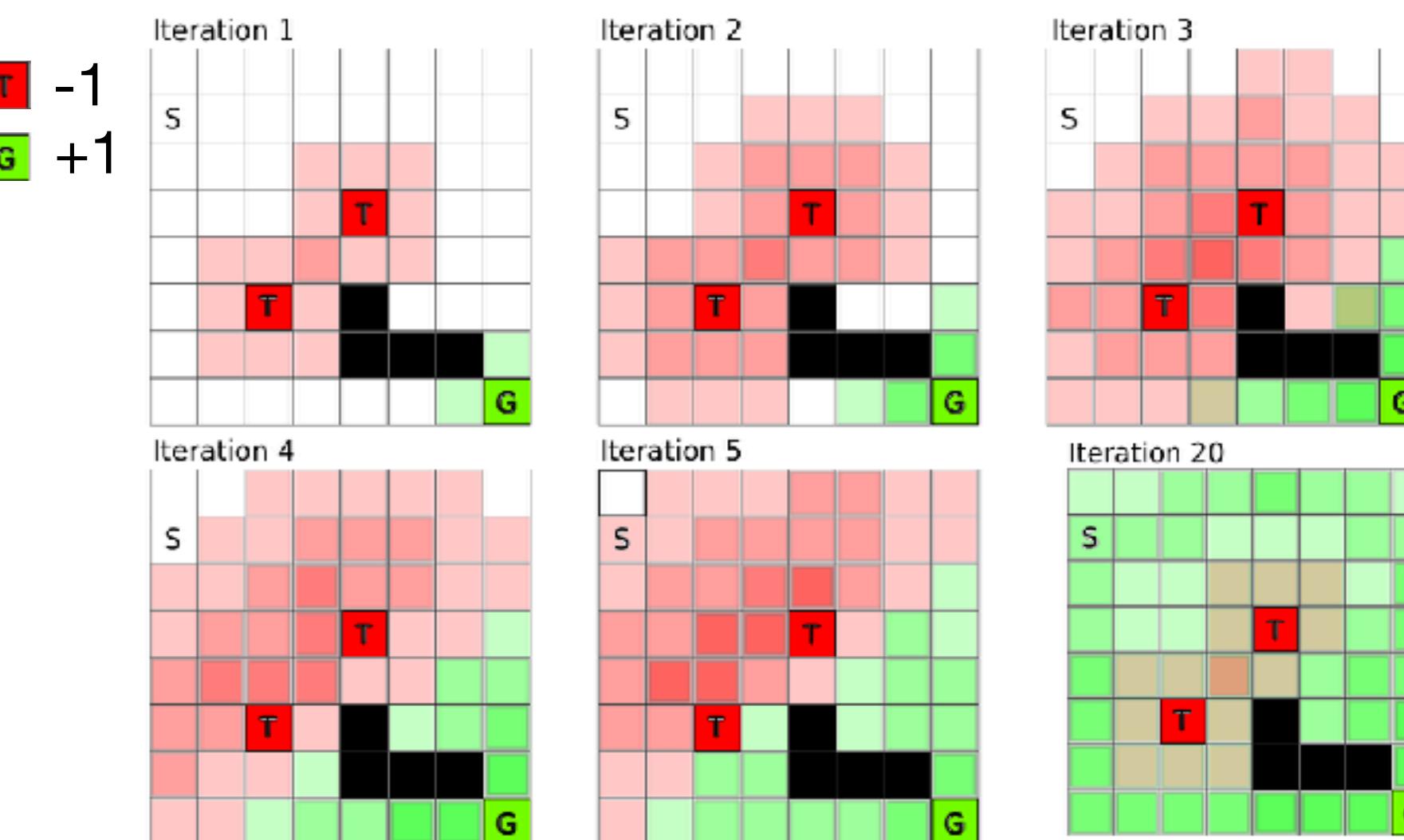
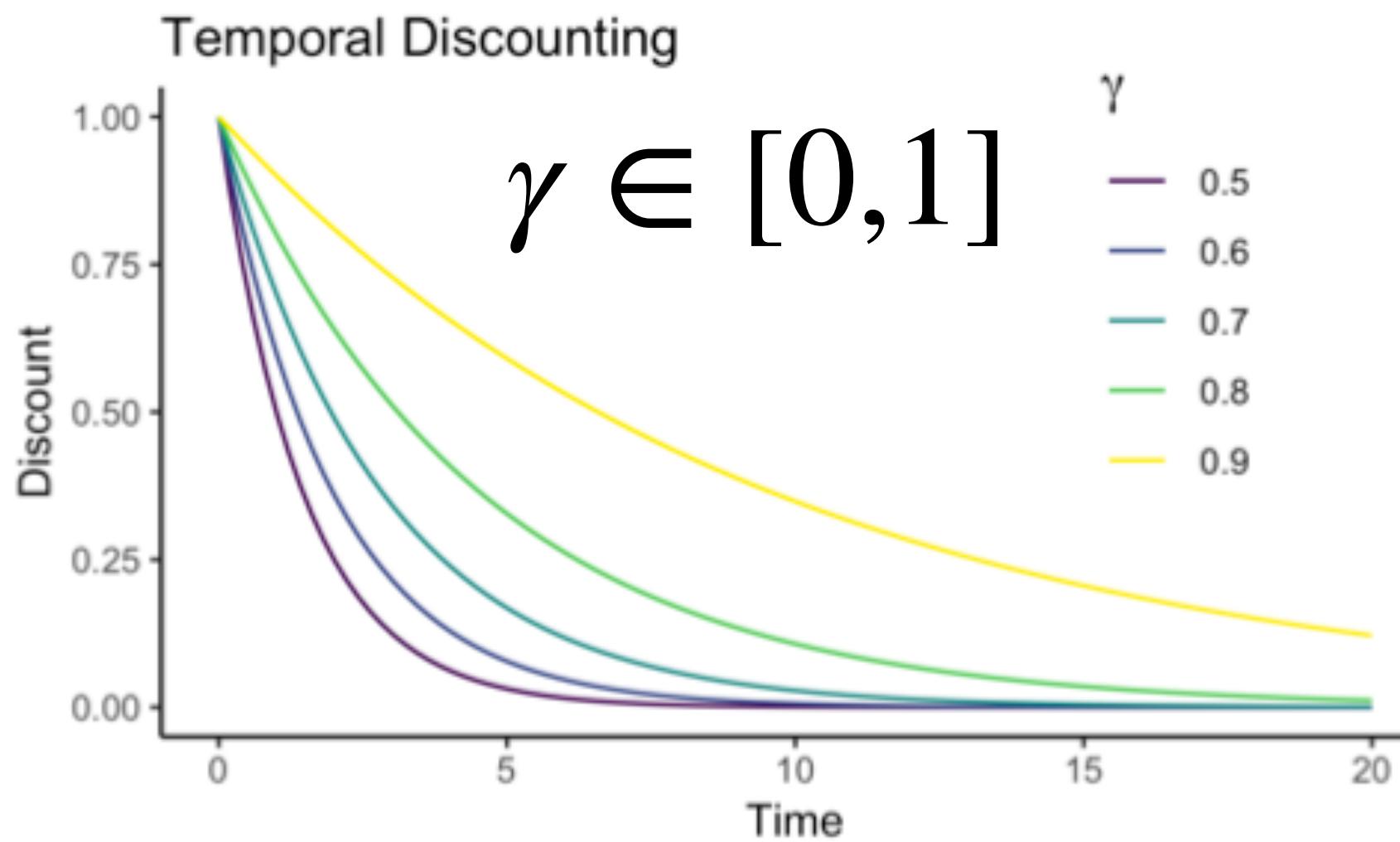


Challenge 1: Credit assignment

- How do we assign credit to actions that are responsible for future reward?
(Minsky, 1961)
- **Temporal Difference (TD) Learning** defines a value function that augments reward expectations with the **discounted value** of the next state



$$V(s) \leftarrow V(s) + \eta \underbrace{\left(r + \gamma V(s') - V(s) \right)}_{\text{discounted future value}}$$



Challenge 1: Credit assignment

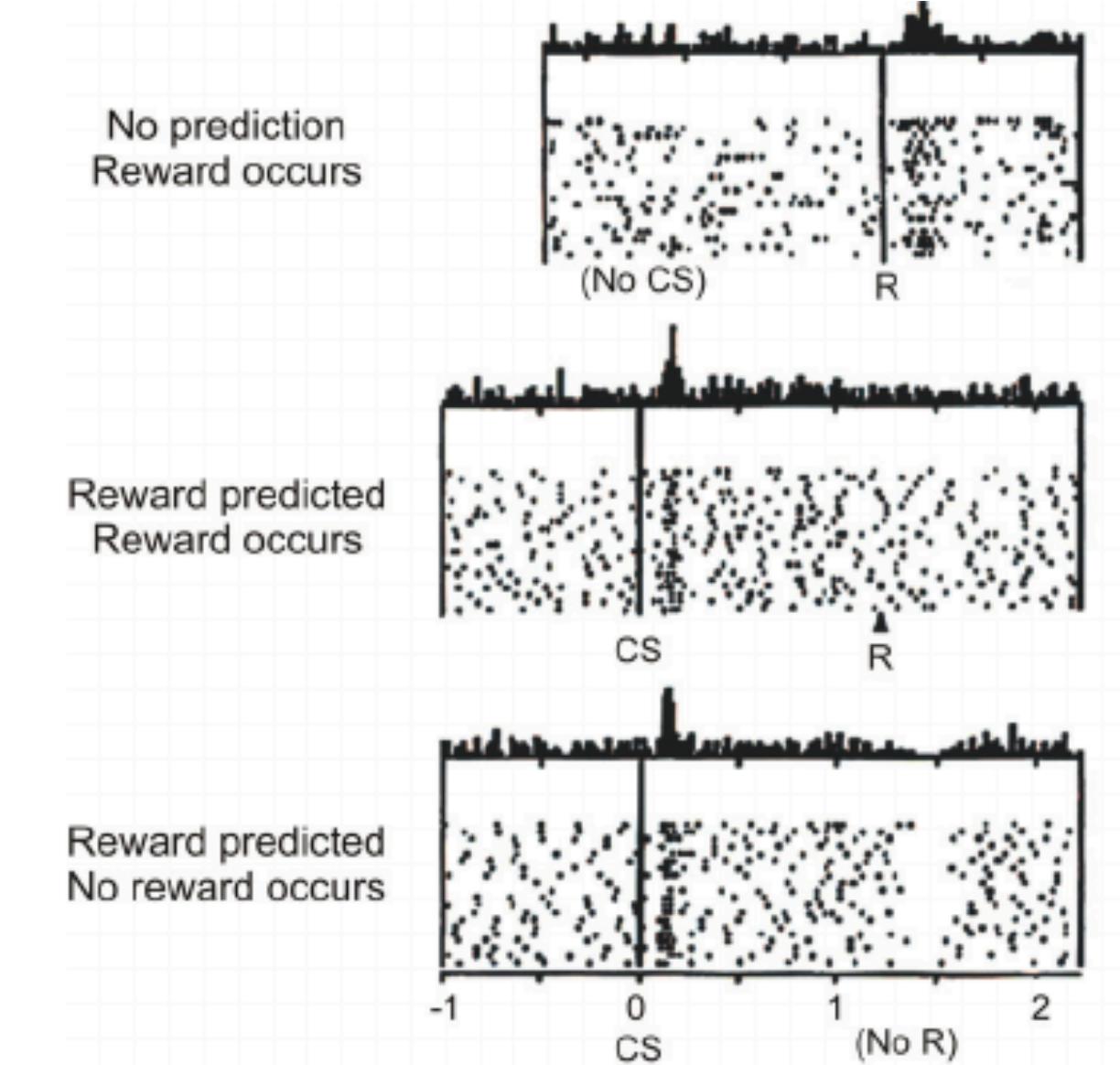
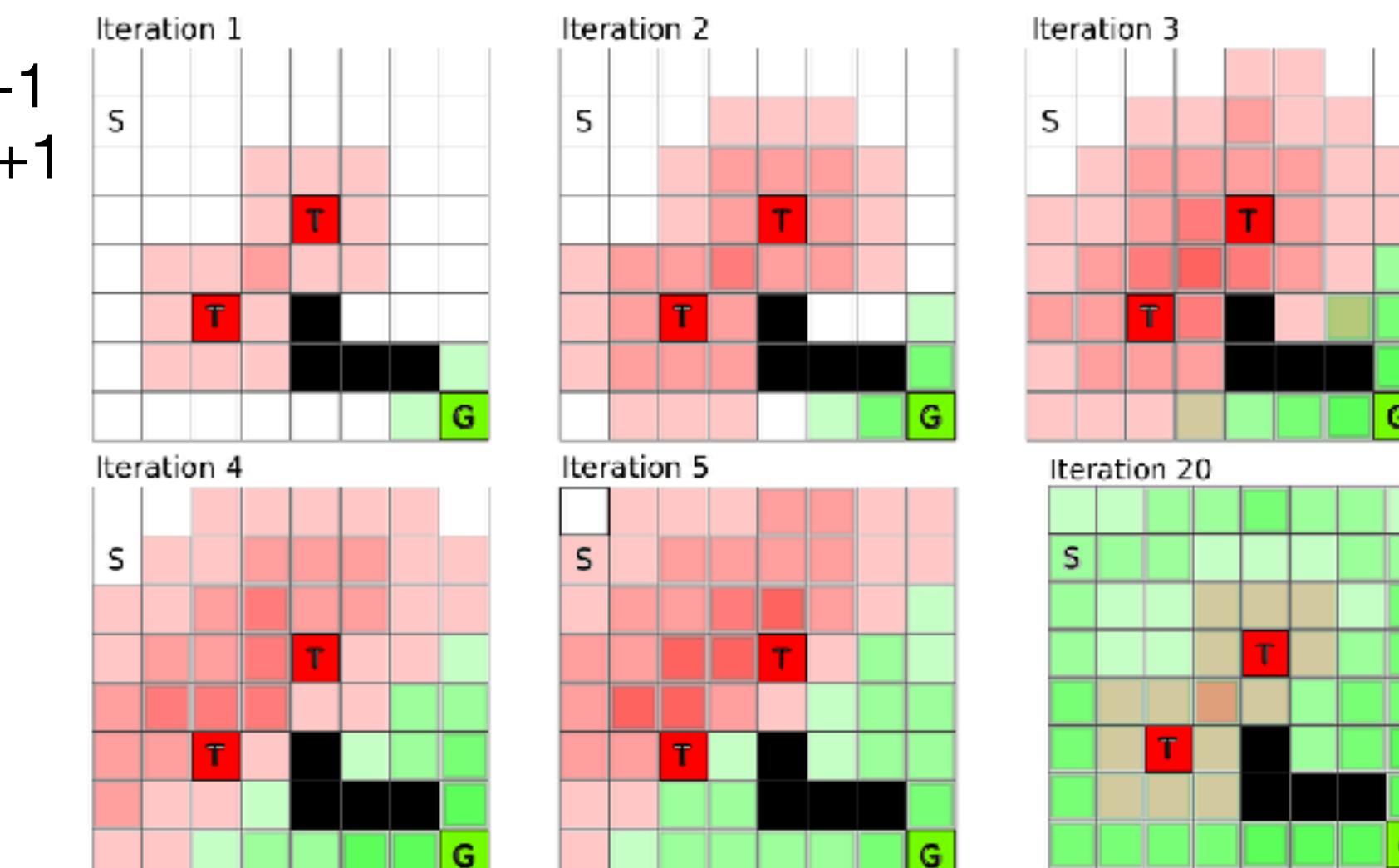
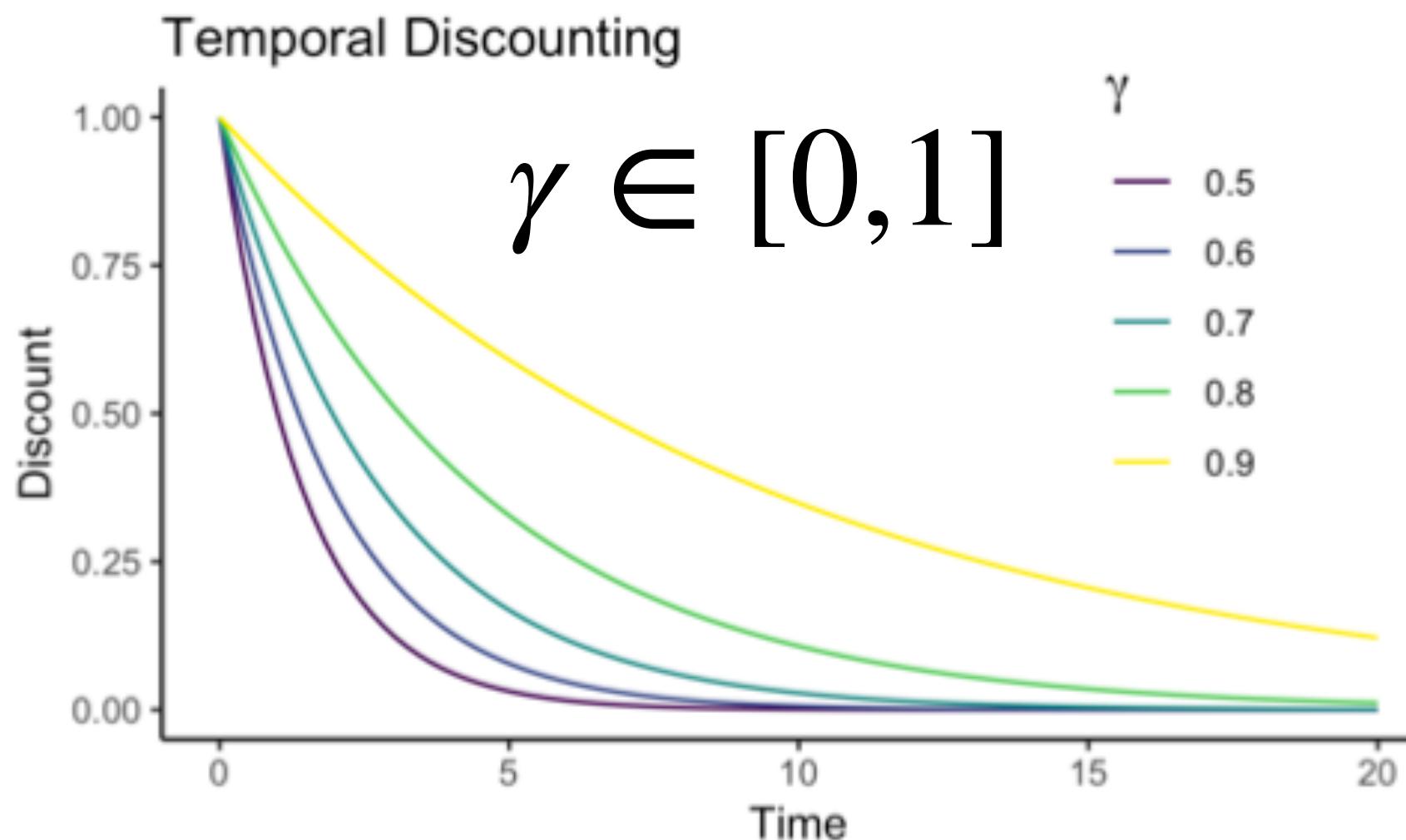
- How do we assign credit to actions that are responsible for future reward?
(Minsky, 1961)
- **Temporal Difference (TD) Learning** defines a value function that augments reward expectations with the **discounted value** of the next state



$$V(s) \leftarrow V(s) + \eta \left(r + \underbrace{\gamma V(s')}_{\text{discounted future value}} - V(s) \right)$$

TD Prediction Error

↳ Dopaminergic Neurons



Challenge 1: Credit assignment

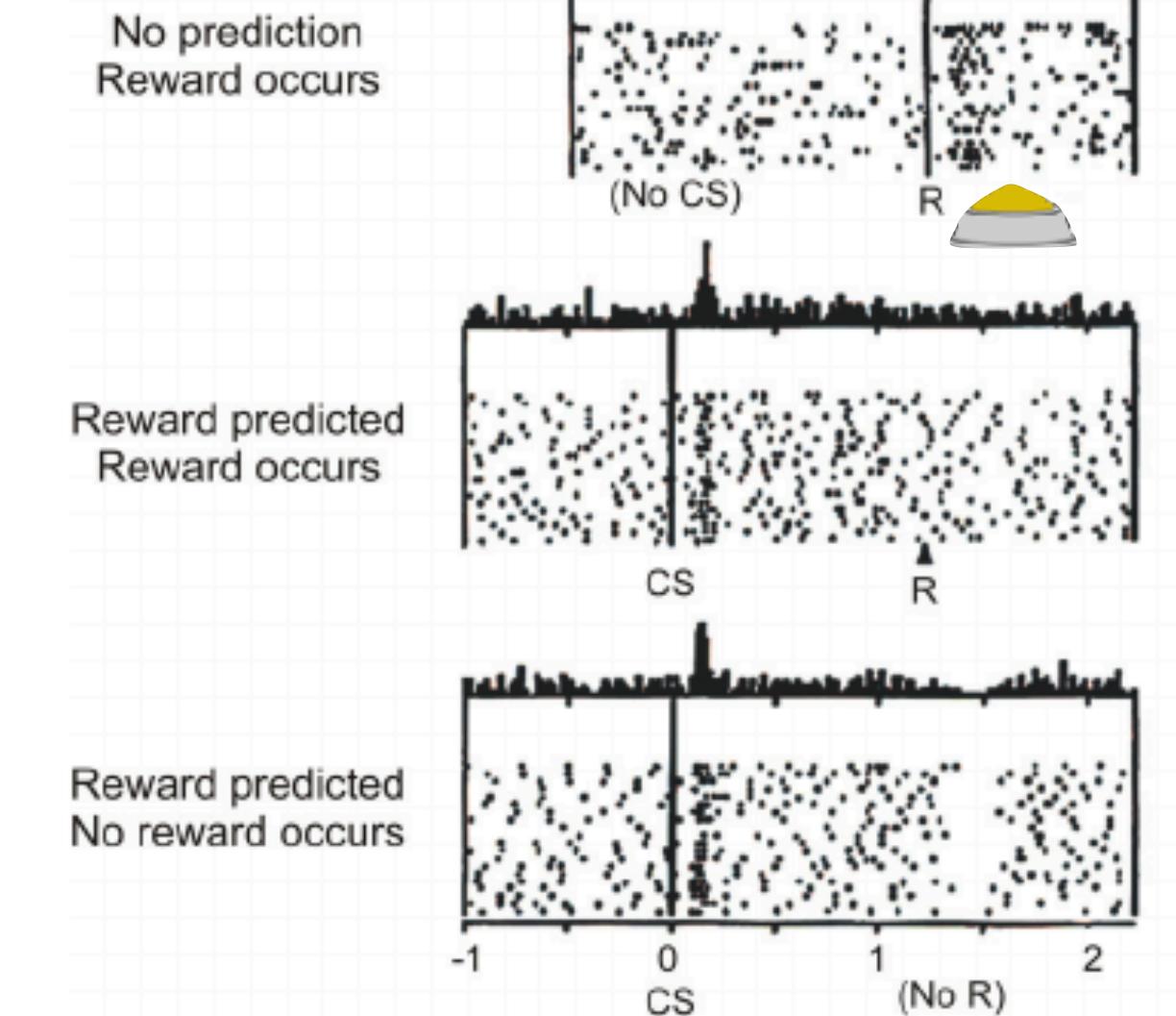
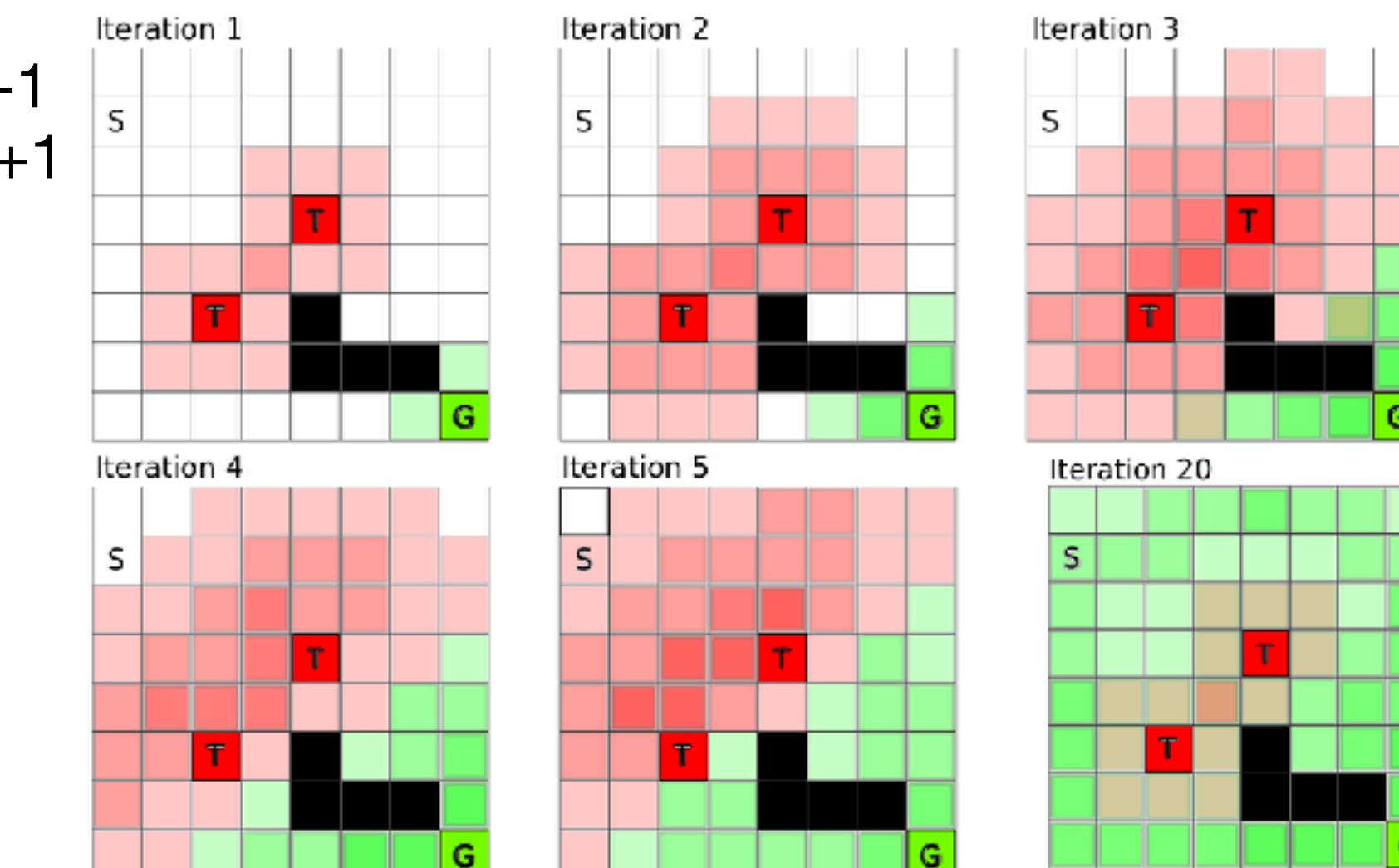
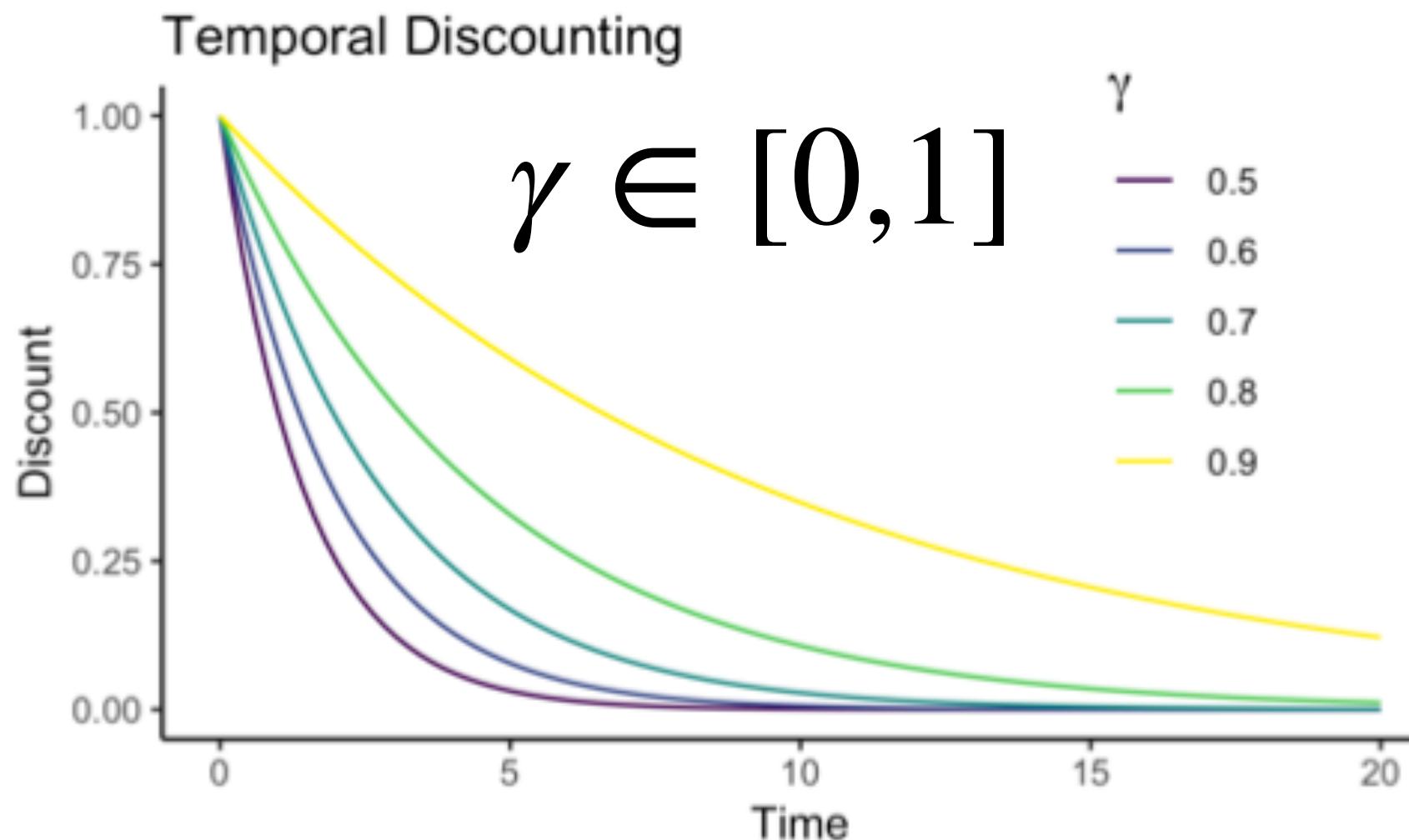
- How do we assign credit to actions that are responsible for future reward?
(Minsky, 1961)
- **Temporal Difference (TD) Learning** defines a value function that augments reward expectations with the **discounted value** of the next state



$$V(s) \leftarrow V(s) + \eta \left(r + \underbrace{\gamma V(s')}_{\text{discounted future value}} - V(s) \right)$$

TD Prediction Error

↳ Dopaminergic Neurons



Challenge 1: Credit assignment

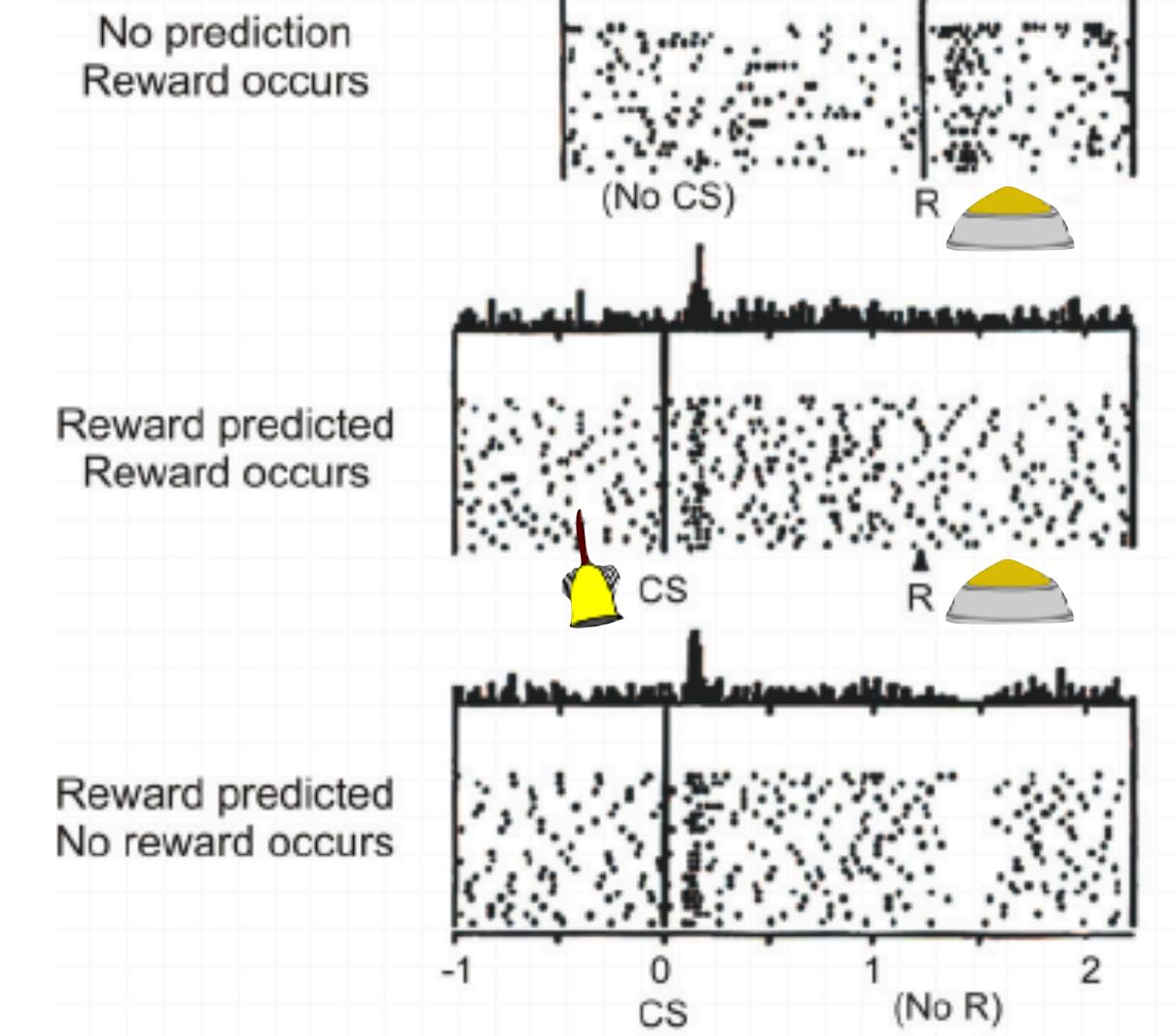
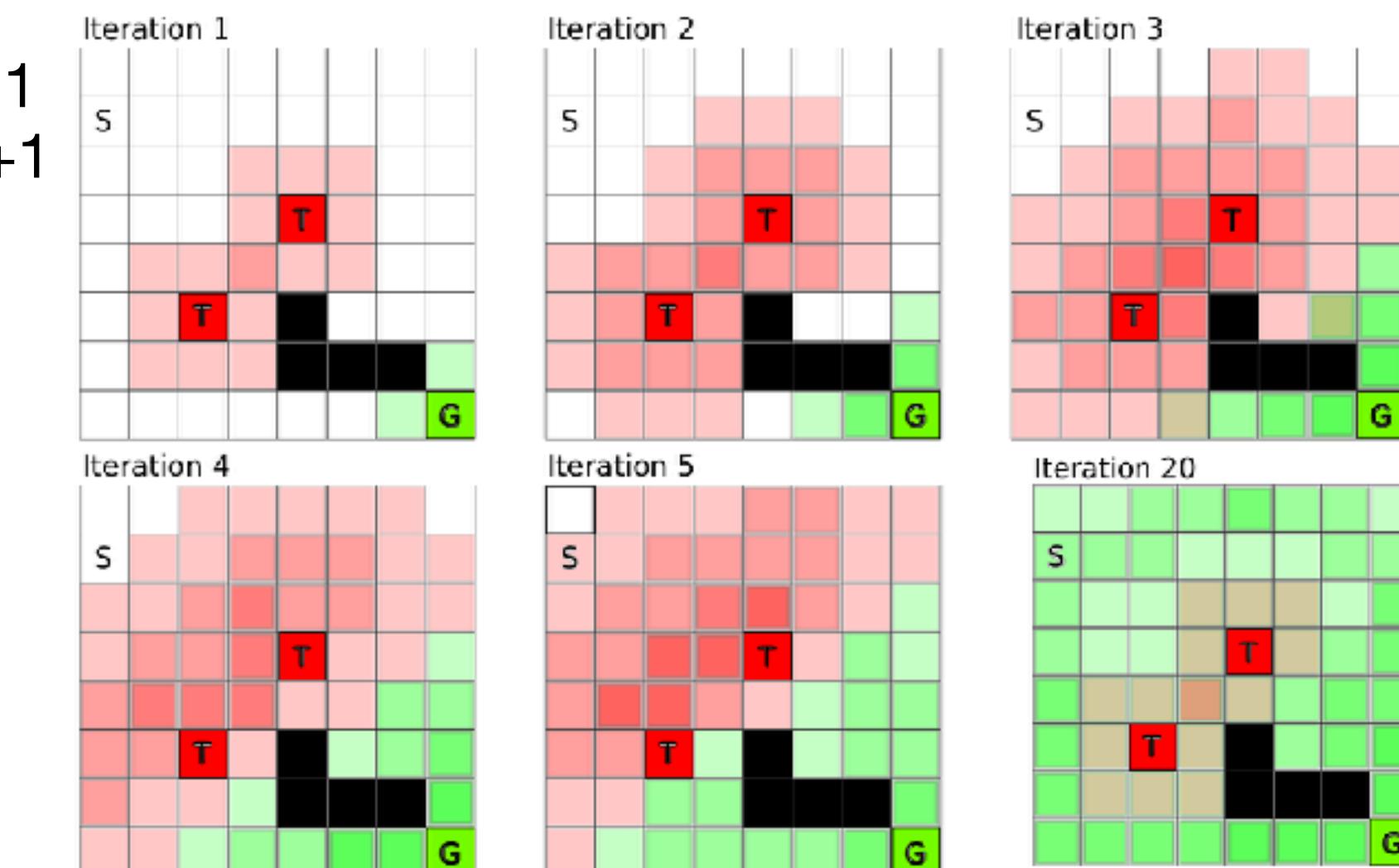
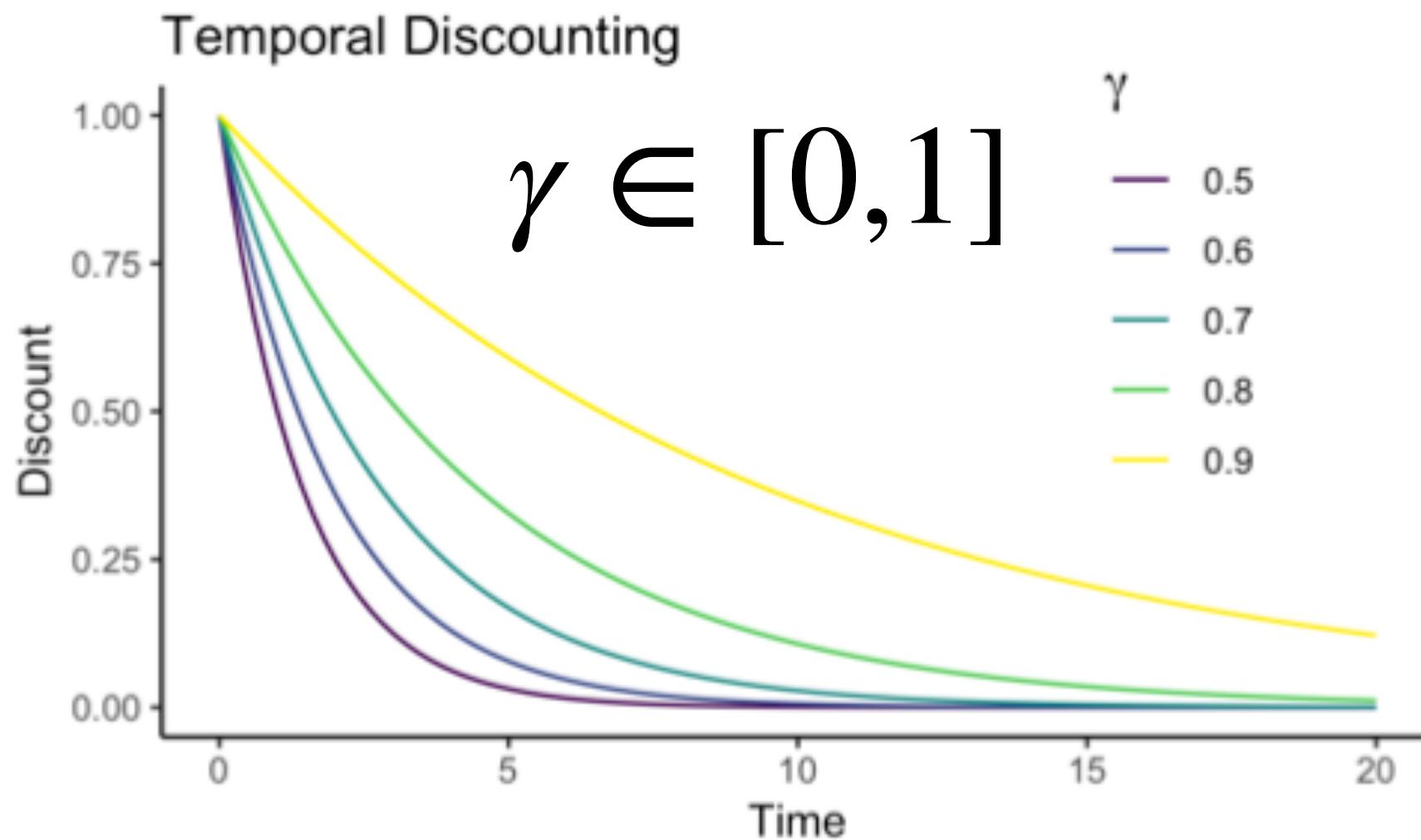
- How do we assign credit to actions that are responsible for future reward?
(Minsky, 1961)
- **Temporal Difference (TD) Learning** defines a value function that augments reward expectations with the **discounted value** of the next state



$$V(s) \leftarrow V(s) + \eta \left(r + \underbrace{\gamma V(s')}_{\text{discounted future value}} - V(s) \right)$$

TD Prediction Error

↳ Dopaminergic Neurons



Challenge 1: Credit assignment

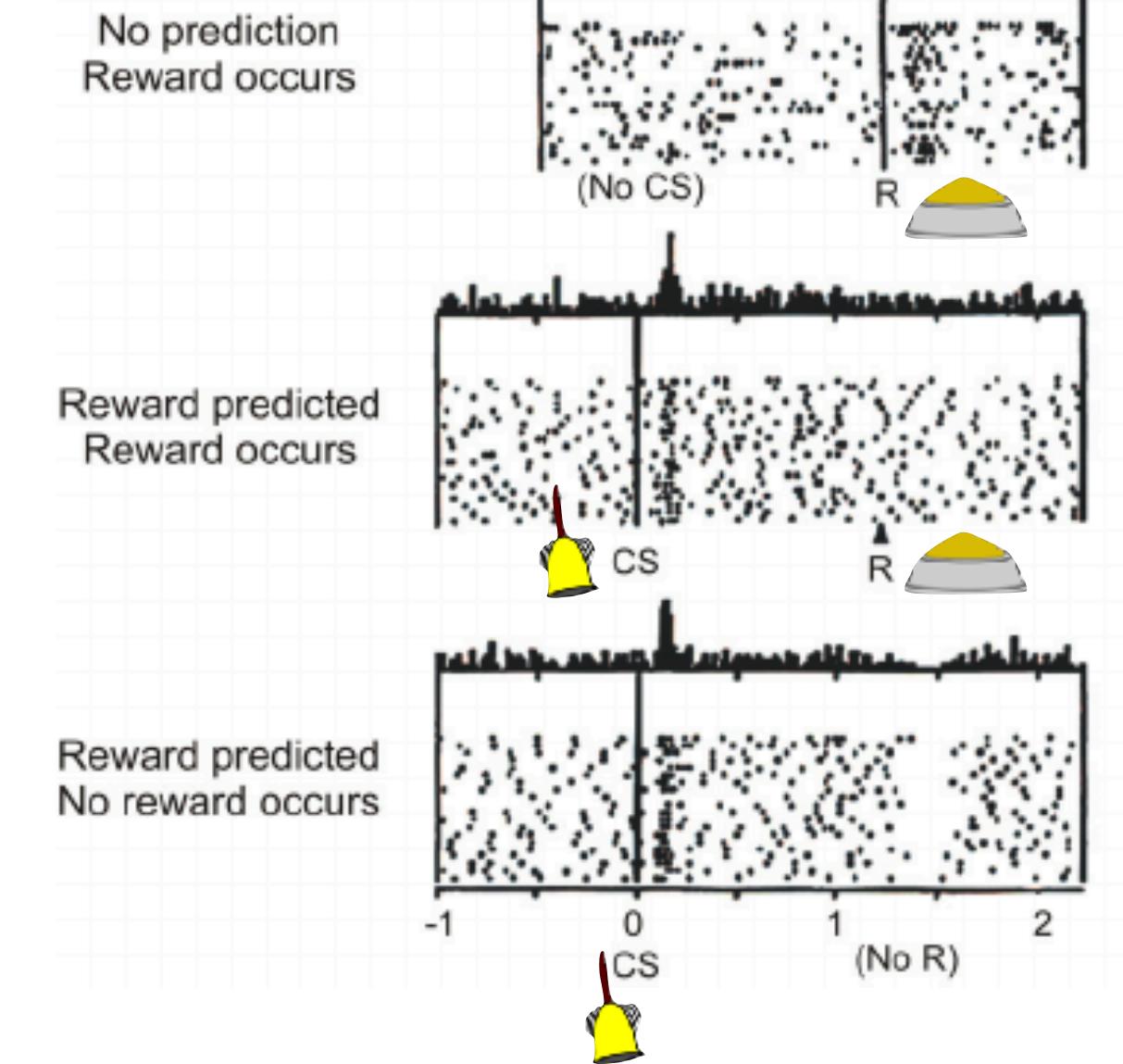
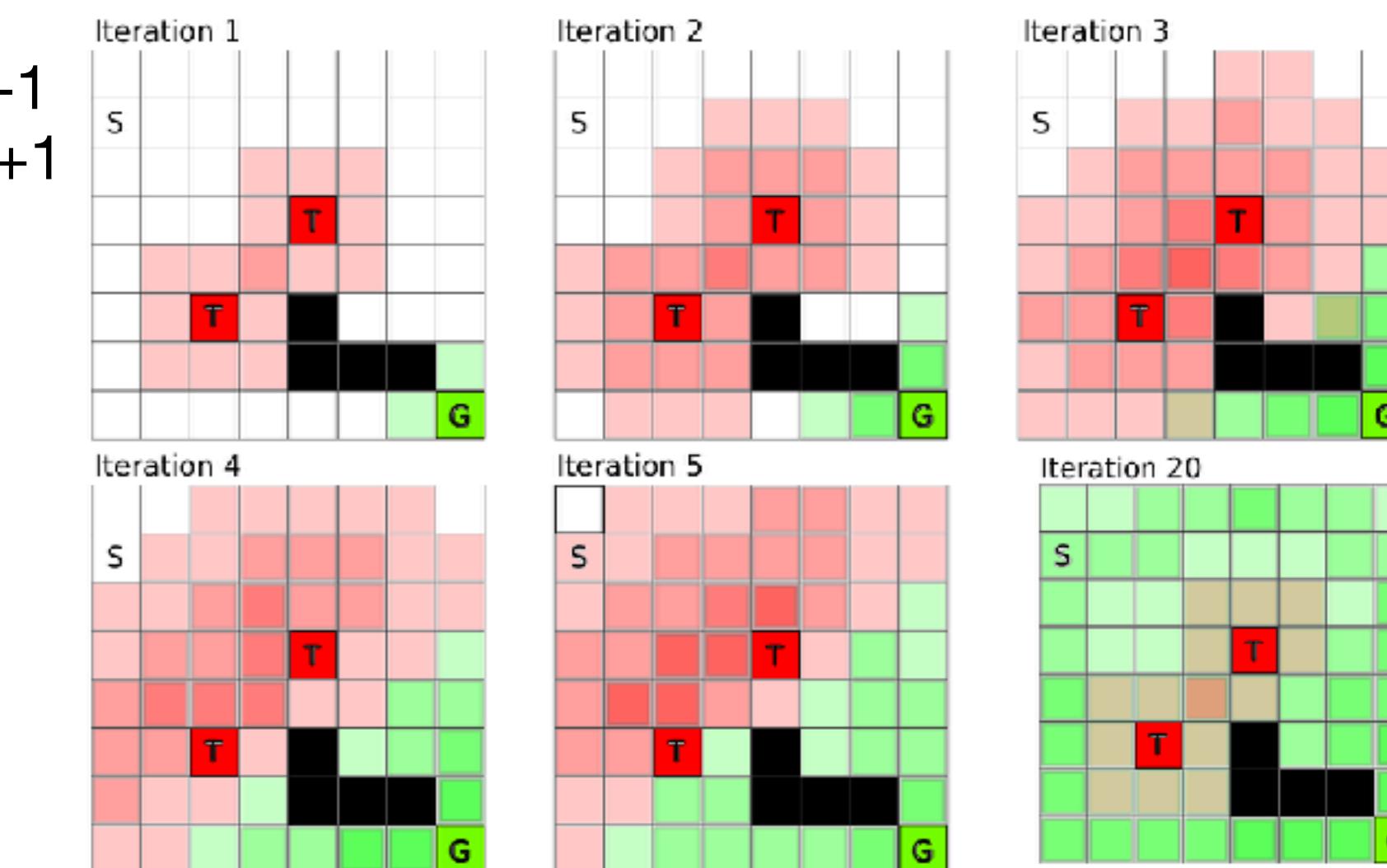
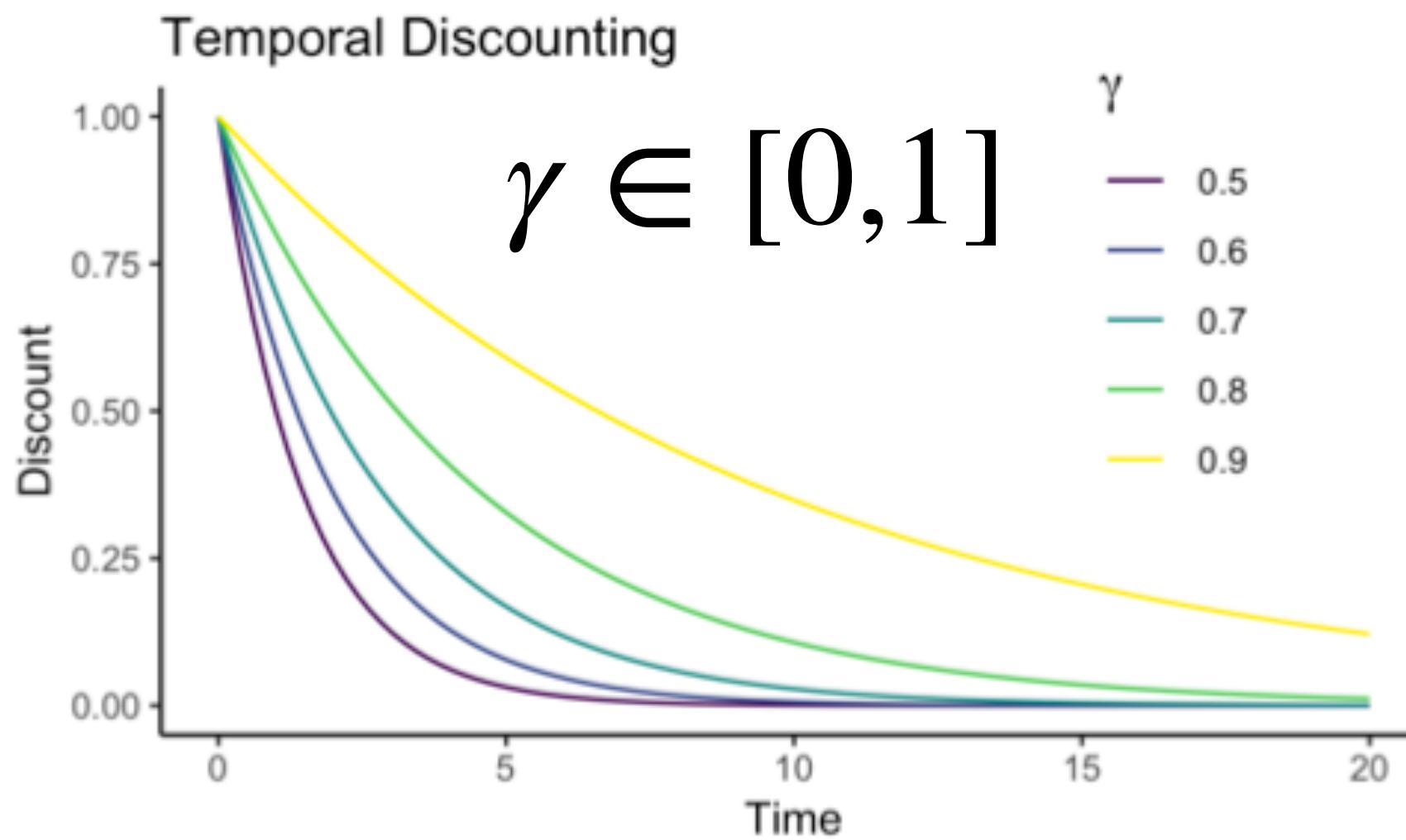
- How do we assign credit to actions that are responsible for future reward?
(Minsky, 1961)
- **Temporal Difference (TD) Learning** defines a value function that augments reward expectations with the **discounted value** of the next state



$$V(s) \leftarrow V(s) + \eta \left(r + \underbrace{\gamma V(s')}_{\text{discounted future value}} - V(s) \right)$$

TD Prediction Error

↳ Dopaminergic Neurons



Difference between $V(s)$ and $Q(s,a)$

- $V(s)$ defines how good is the state
 - Actions become implicit under policy π
- $Q(s, a)$ defines how good it is to take action a in state s
 - Actions are made explicit
- We will use $V(s)$ and $Q(s, a)$ somewhat interchangeably, depending on what the situation calls for
- It's ok to be somewhat confused at times, and I promise not to ask any "gotcha" questions purposefully trying to trick you into confusing the two

Delta-rule update with TD error

$$V(s) \leftarrow V(s) + \eta (r + \gamma V(s') - V(s))$$

$$Q(s, a) \leftarrow Q(s, a) + \eta [r, \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

Difference between $V(s)$ and $Q(s,a)$

- $V(s)$ defines how good is the state
 - Actions become implicit under policy π
- $Q(s, a)$ defines how good it is to take action a in state s
 - Actions are made explicit
- We will use $V(s)$ and $Q(s, a)$ somewhat interchangeably, depending on what the situation calls for
- It's ok to be somewhat confused at times, and I promise not to ask any "gotcha" questions purposefully trying to trick you into confusing the two

$$V_\pi(s) = \sum_a \pi(a | s) Q_\pi(s, a)$$

Delta-rule update with TD error

$$V(s) \leftarrow V(s) + \eta (r + \gamma V(s') - V(s))$$

$$Q(s, a) \leftarrow Q(s, a) + \eta [r, \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

The (formal) RL Problem

The (formal) RL Problem

Select a policy π^* that maximizes expected rewards

The (formal) RL Problem

Select a policy π^* that maximizes expected rewards

Not just immediate rewards, but discounted future returns

- Value function under some policy π :

$$V_\pi(s) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t \in \tau} \gamma^t R_{t+1} \mid s_0 = s \right]$$

The (formal) RL Problem

Select a policy π^* that maximizes expected rewards

Not just immediate rewards, but discounted future returns

- Value function under some policy π :

$$V_\pi(s) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t \in \tau} \gamma^t R_{t+1} \mid s_0 = s \right]$$

- We can rewrite the expectation $\mathbb{E}_{\tau \sim \pi}$ in terms of the **policy** and **state transitions**

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a) [R(s', a) + \gamma V_\pi(s')]$$

The (formal) RL Problem

Select a policy π^* that maximizes expected rewards

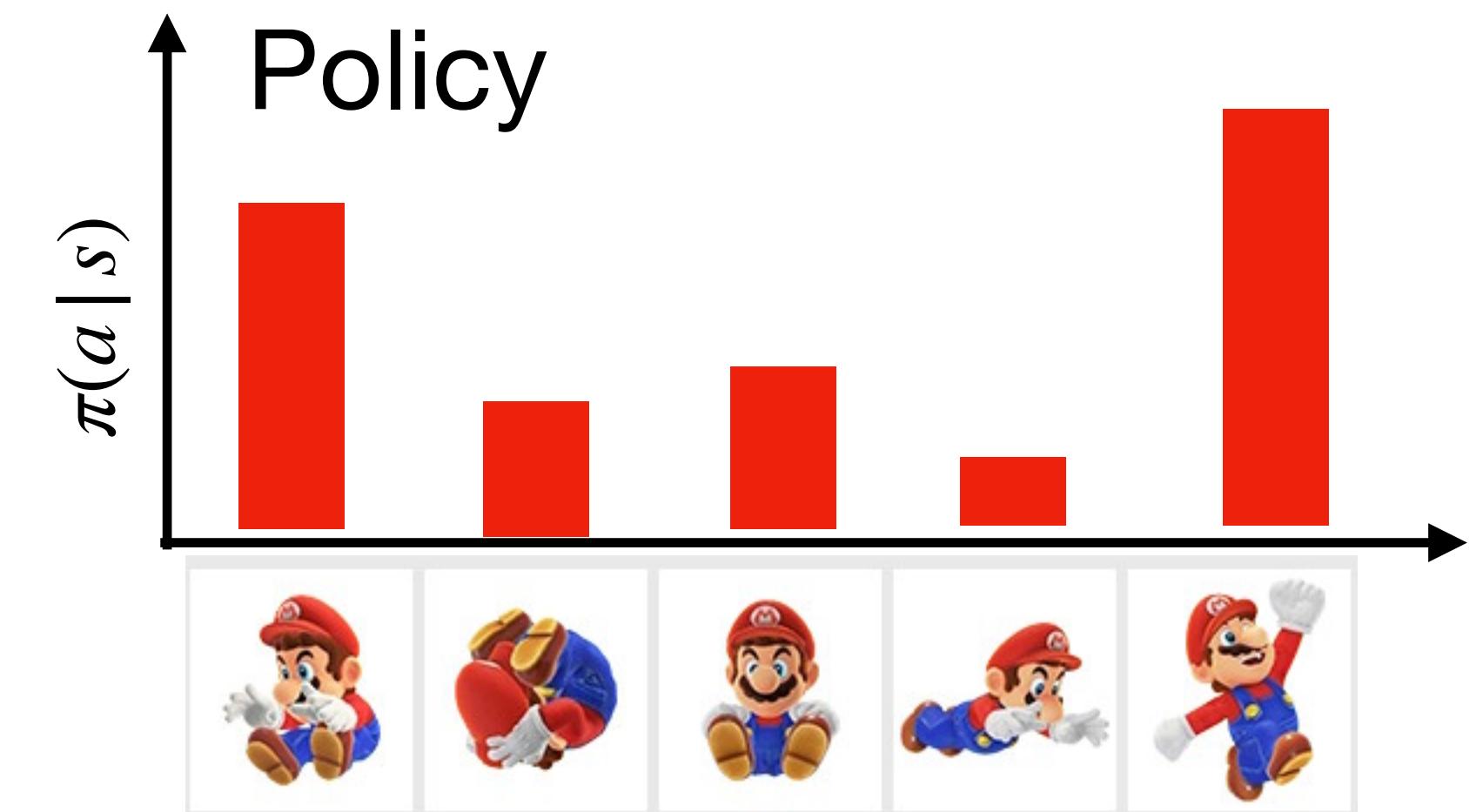
Not just immediate rewards, but discounted future returns

- Value function under some policy π :

$$V_\pi(s) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t \in \tau} \gamma^t R_{t+1} \mid s_0 = s \right]$$

- We can rewrite the expectation $\mathbb{E}_{\tau \sim \pi}$ in terms of the **policy** and **state transitions**

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a) [R(s', a) + \gamma V_\pi(s')]$$



The (formal) RL Problem

Select a policy π^* that maximizes expected rewards

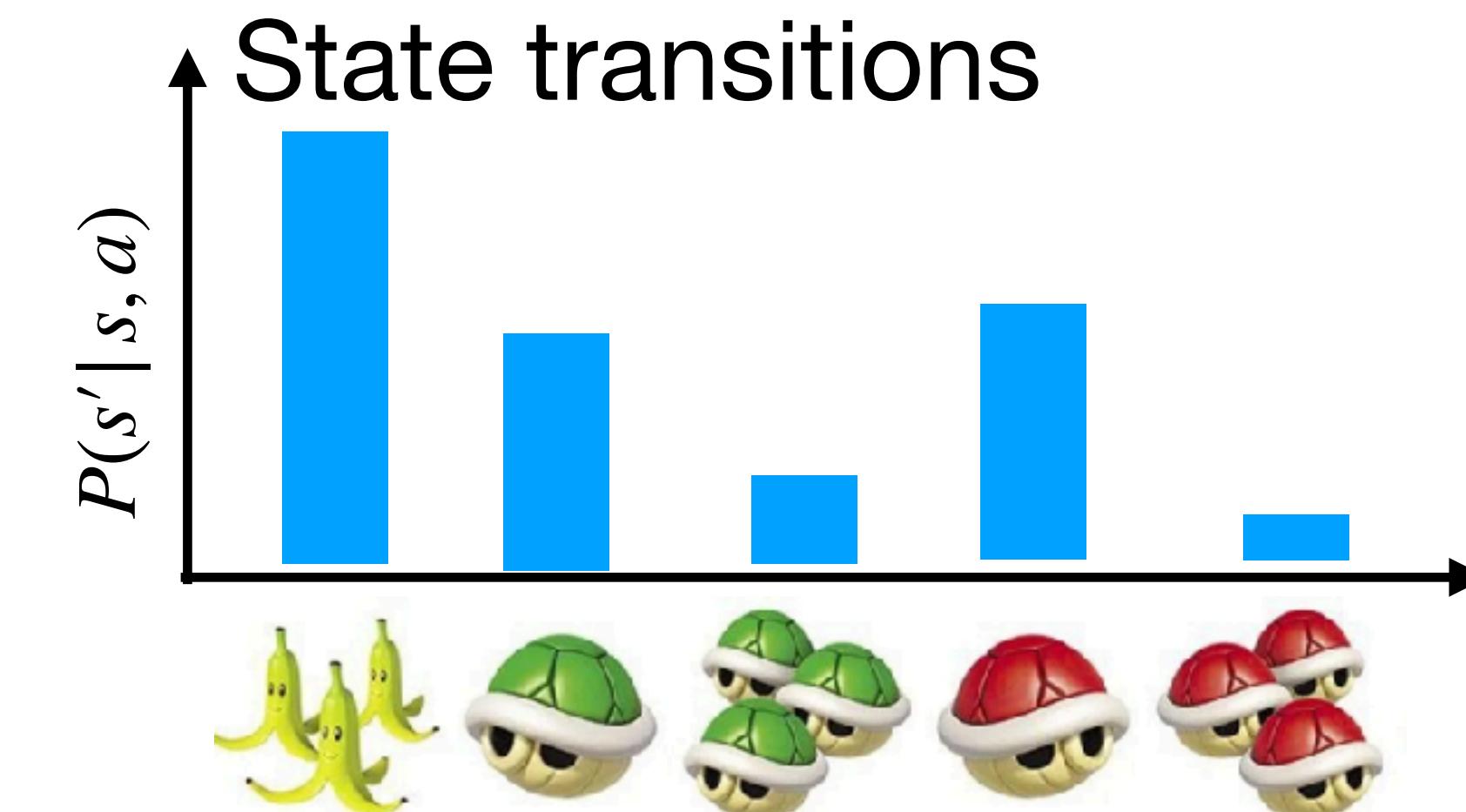
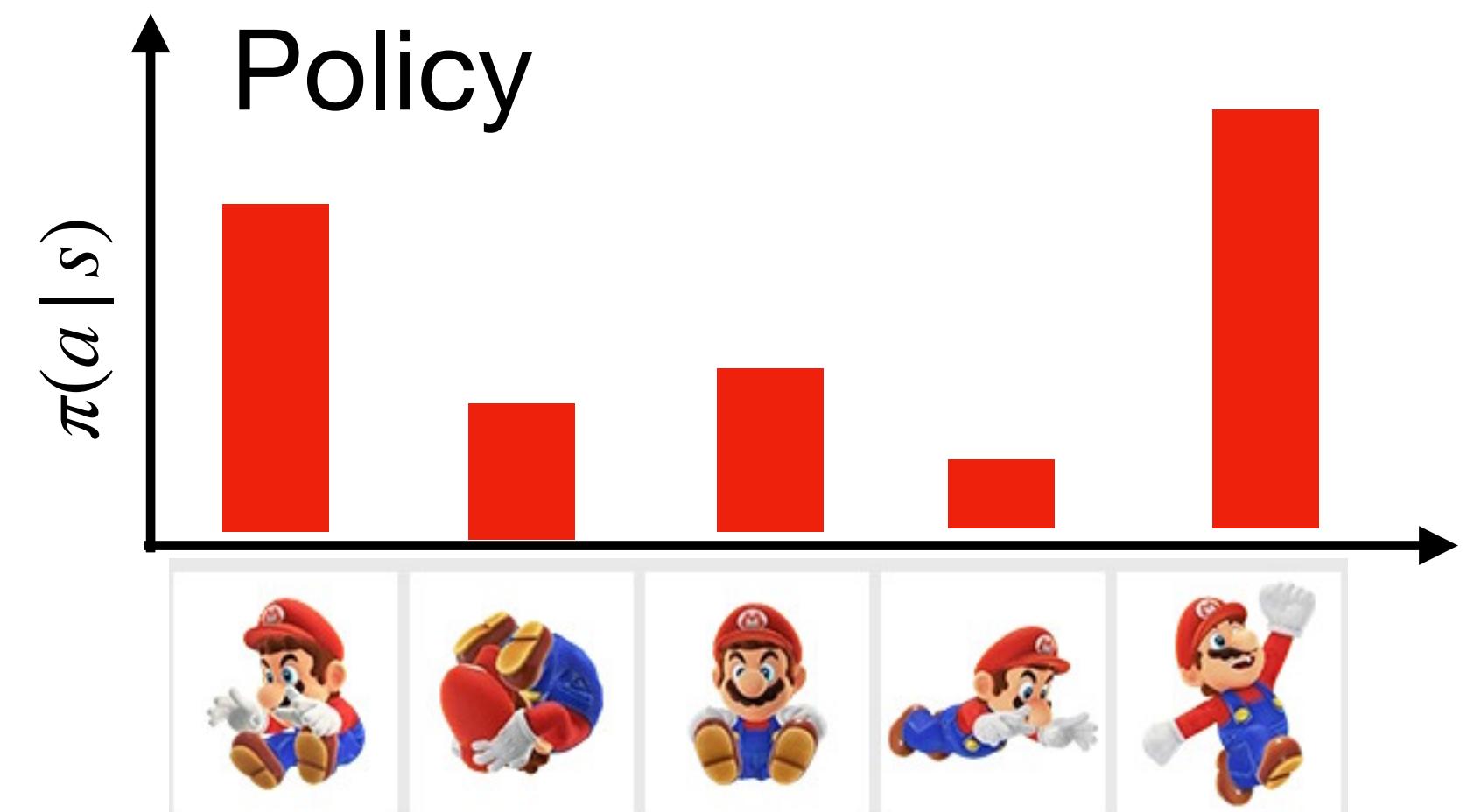
Not just immediate rewards, but discounted future returns

- Value function under some policy π :

$$V_\pi(s) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t \in \tau} \gamma^t R_{t+1} \mid s_0 = s \right]$$

- We can rewrite the expectation $\mathbb{E}_{\tau \sim \pi}$ in terms of the **policy** and **state transitions**

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a) [R(s', a) + \gamma V_\pi(s')]$$



The (formal) RL Problem

Select a policy π^* that maximizes expected rewards

Not just immediate rewards, but discounted future returns

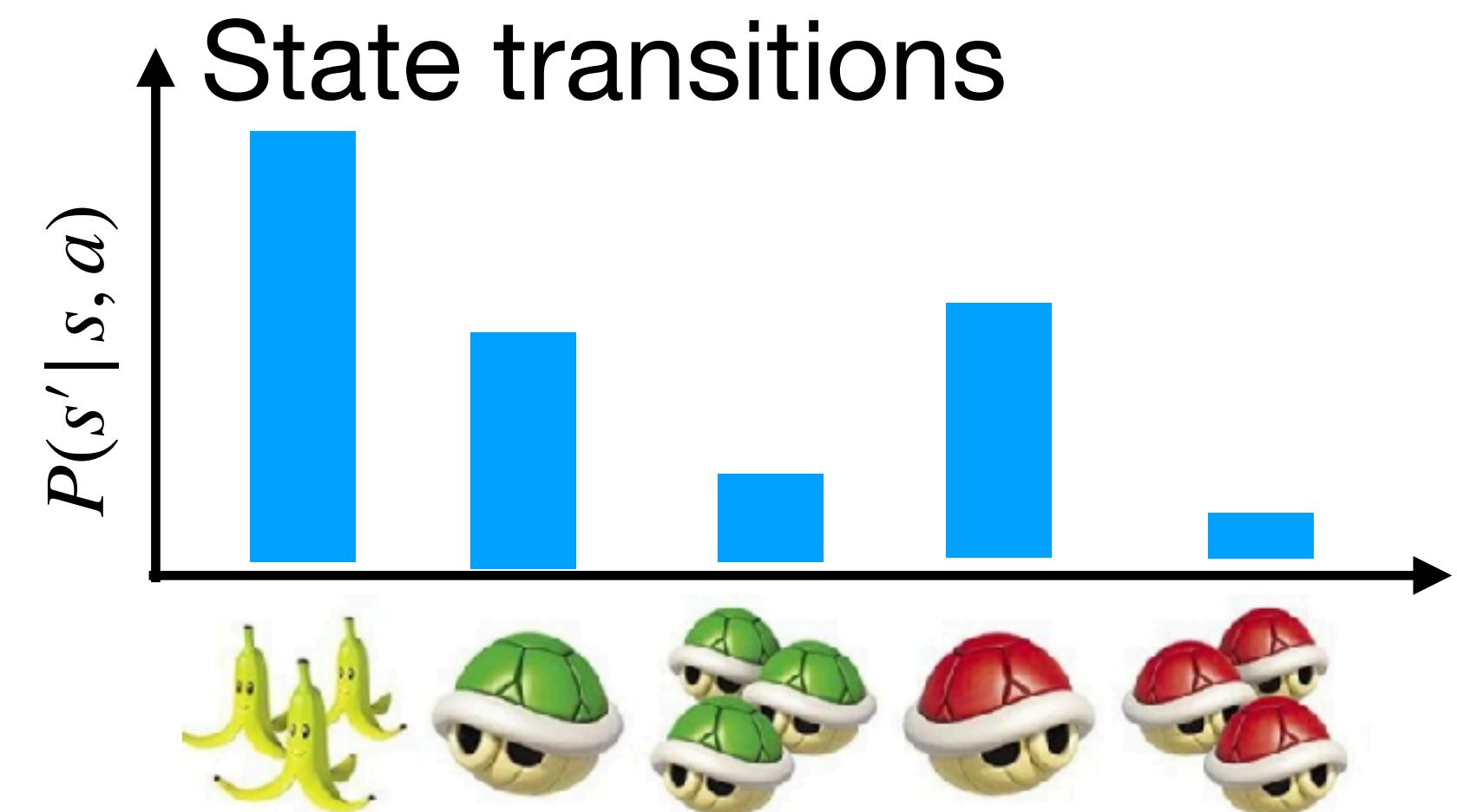
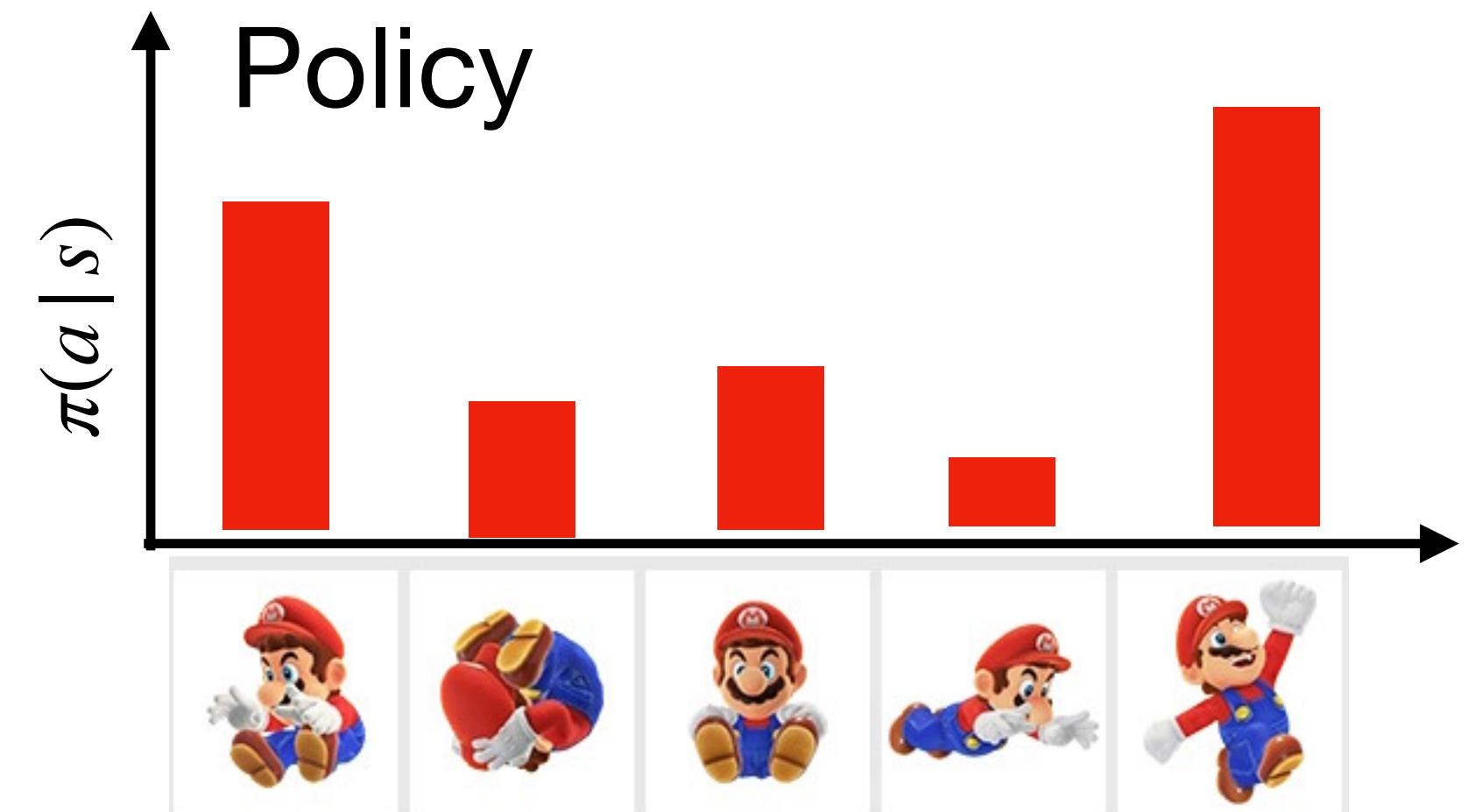
- Value function under some policy π :

$$V_\pi(s) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t \in \tau} \gamma^t R_{t+1} \mid s_0 = s \right]$$

- We can rewrite the expectation $\mathbb{E}_{\tau \sim \pi}$ in terms of the **policy** and **state transitions**

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a) [R(s', a) + \gamma V_\pi(s')]$$

- The sum can be written recursively as **immediate reward** + **discounted future reward**



The (formal) RL Problem

Select a policy π^* that maximizes expected rewards

Not just immediate rewards, but discounted future returns

- Value function under some policy π :

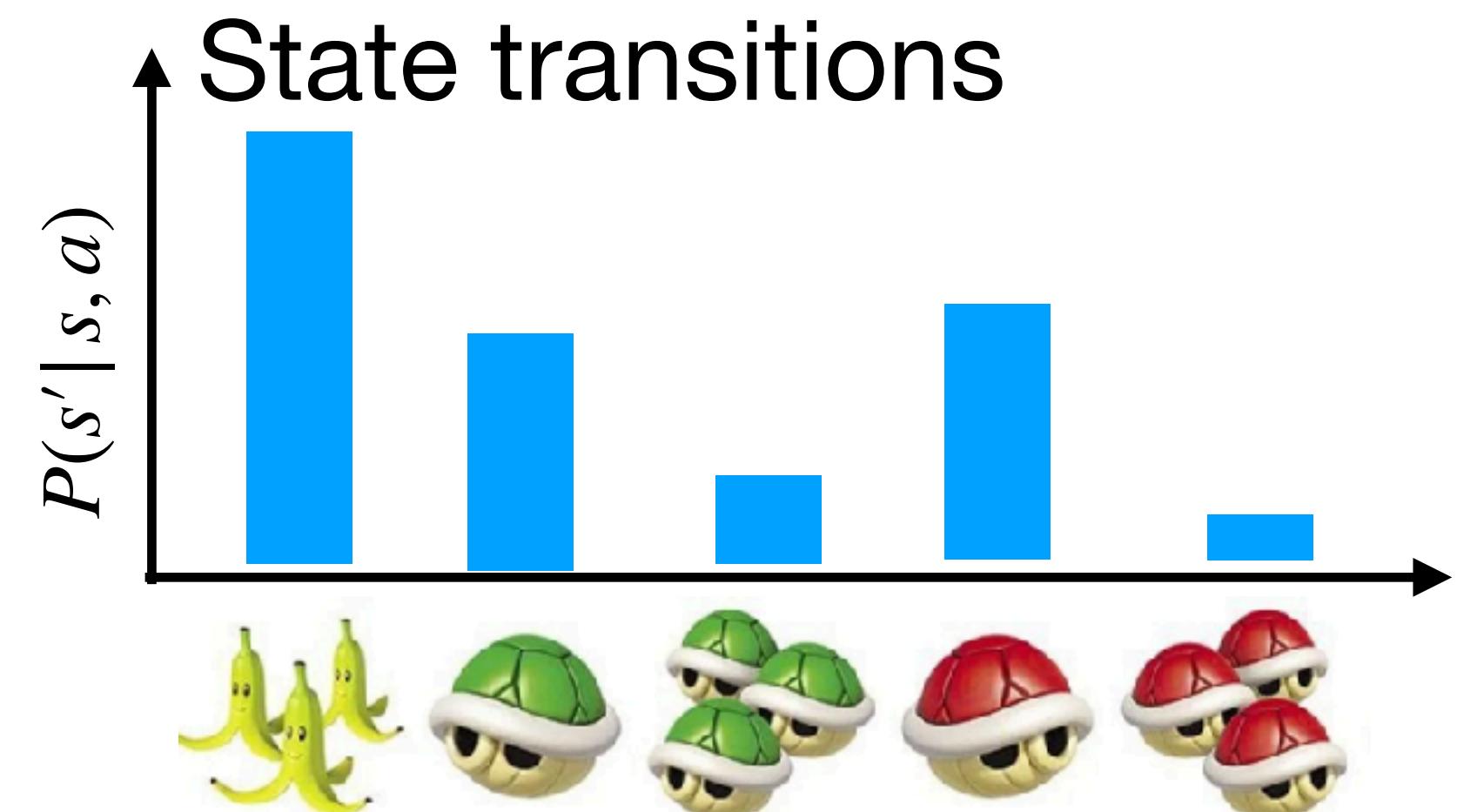
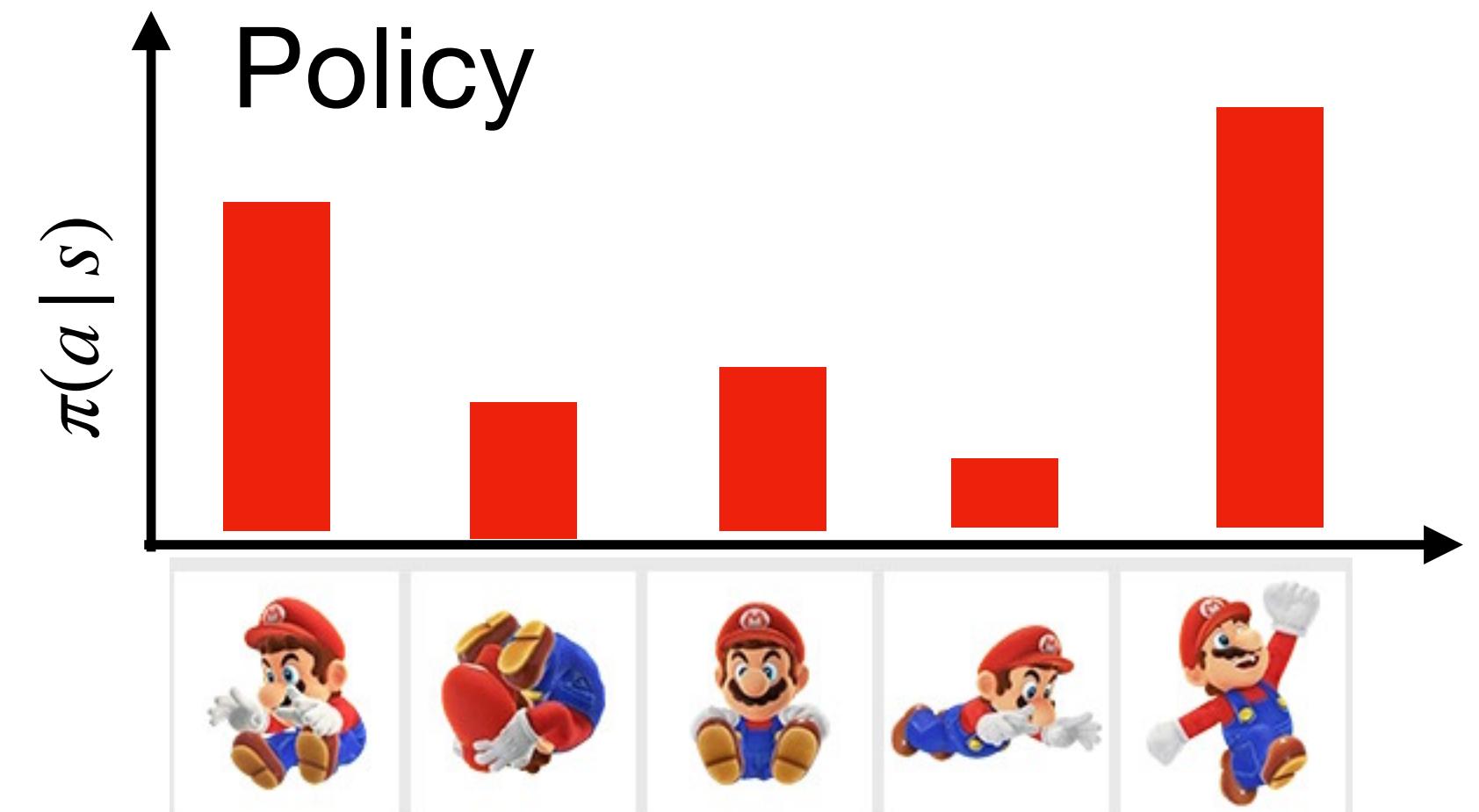
$$V_\pi(s) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t \in \tau} \gamma^t R_{t+1} \mid s_0 = s \right]$$

- We can rewrite the expectation $\mathbb{E}_{\tau \sim \pi}$ in terms of the **policy** and **state transitions**

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a) [R(s', a) + \gamma V_\pi(s')]$$

- The sum can be written recursively as **immediate reward** + **discounted future reward**

$$\gamma^0 = 1$$



The (formal) RL Problem

Select a policy π^* that maximizes expected rewards

Not just immediate rewards, but discounted future returns

- Value function under some policy π :

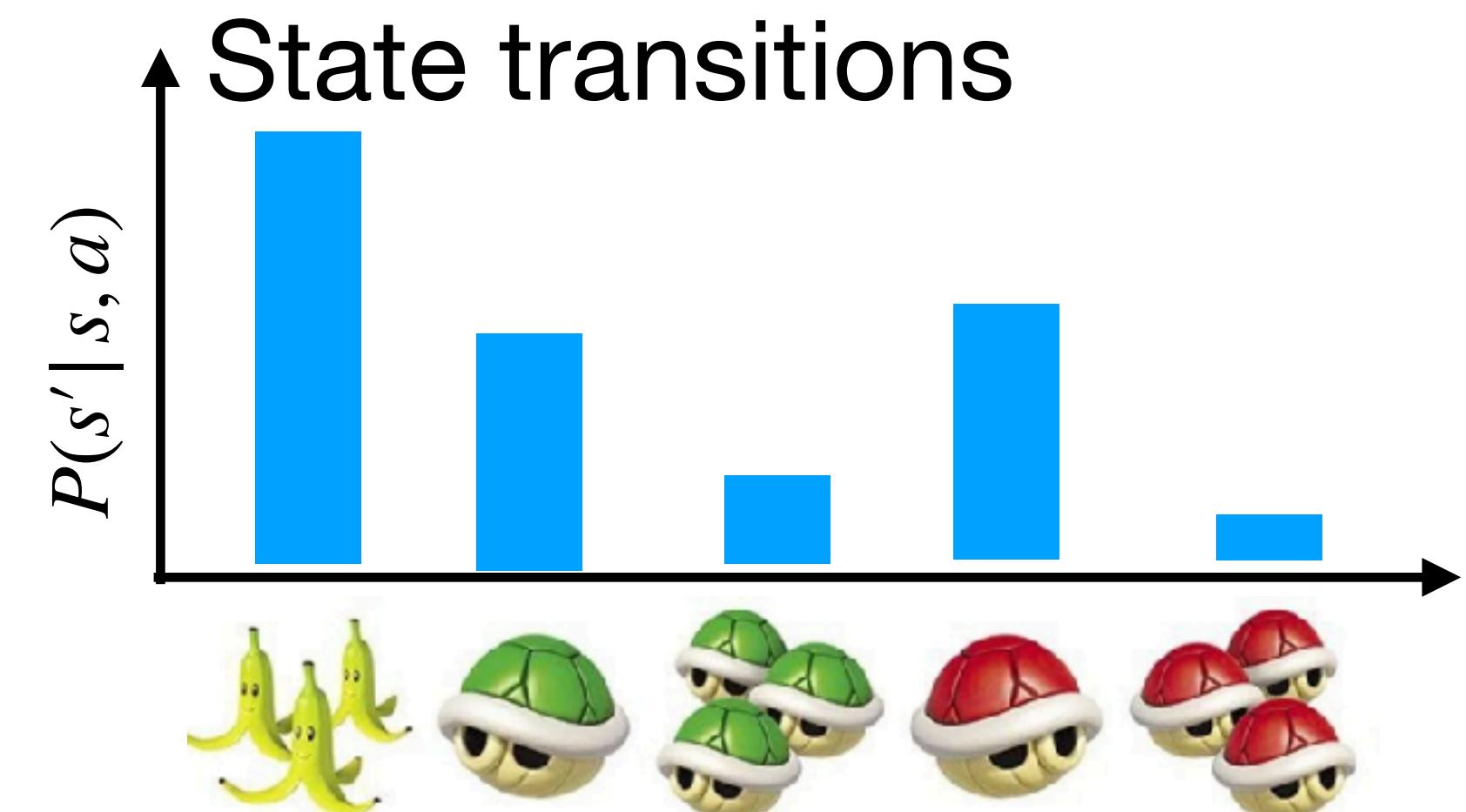
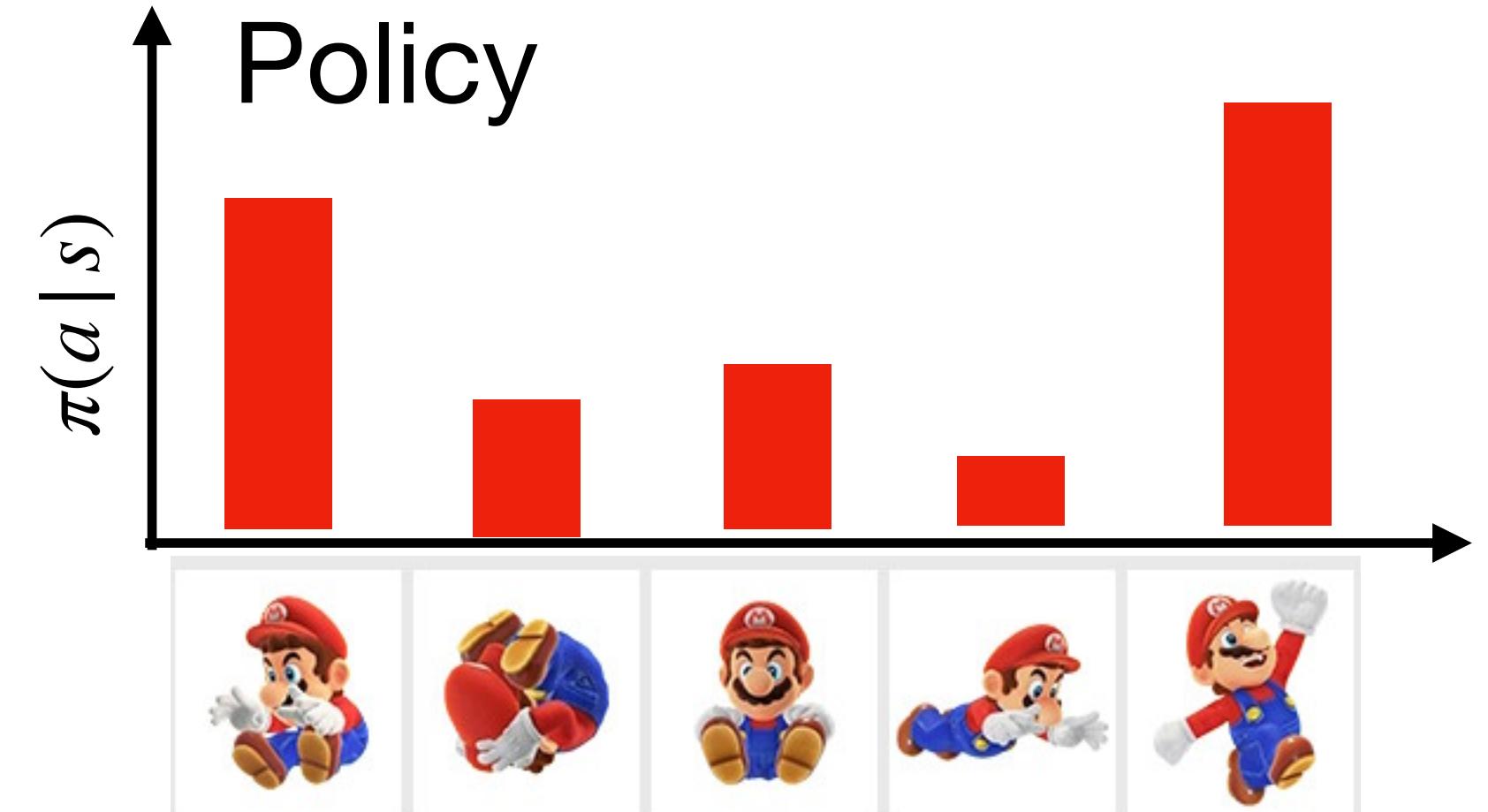
$$V_\pi(s) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t \in \tau} \gamma^t R_{t+1} \mid s_0 = s \right]$$

- We can rewrite the expectation $\mathbb{E}_{\tau \sim \pi}$ in terms of the **policy** and **state transitions**

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a) [R(s', a) + \gamma V_\pi(s')]$$

- The sum can be written recursively as **immediate reward** + **discounted future reward**

$$\gamma^0 = 1$$



The (formal) RL Problem

Select a policy π^* that maximizes expected rewards

Not just immediate rewards, but discounted future returns

- Value function under some policy π :

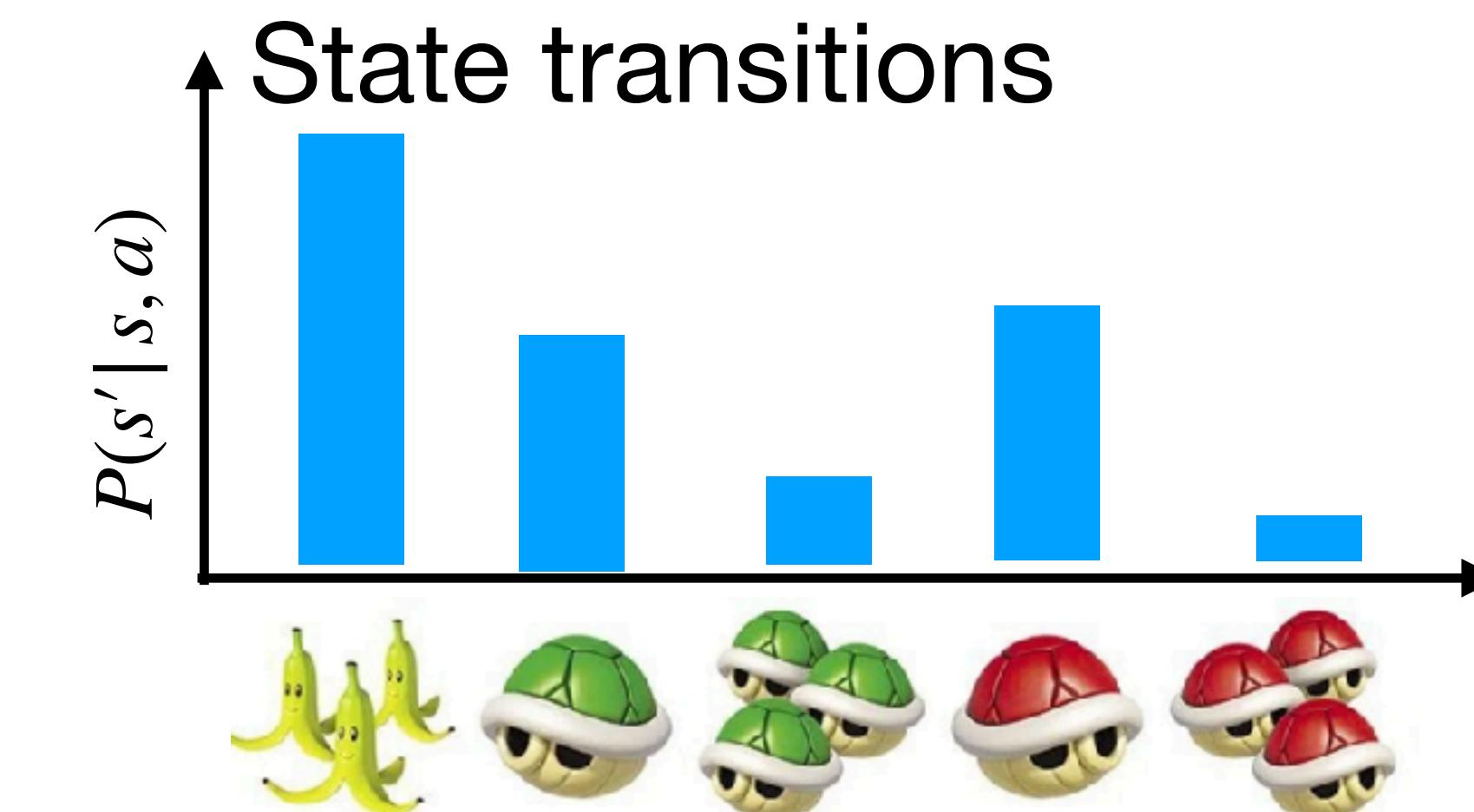
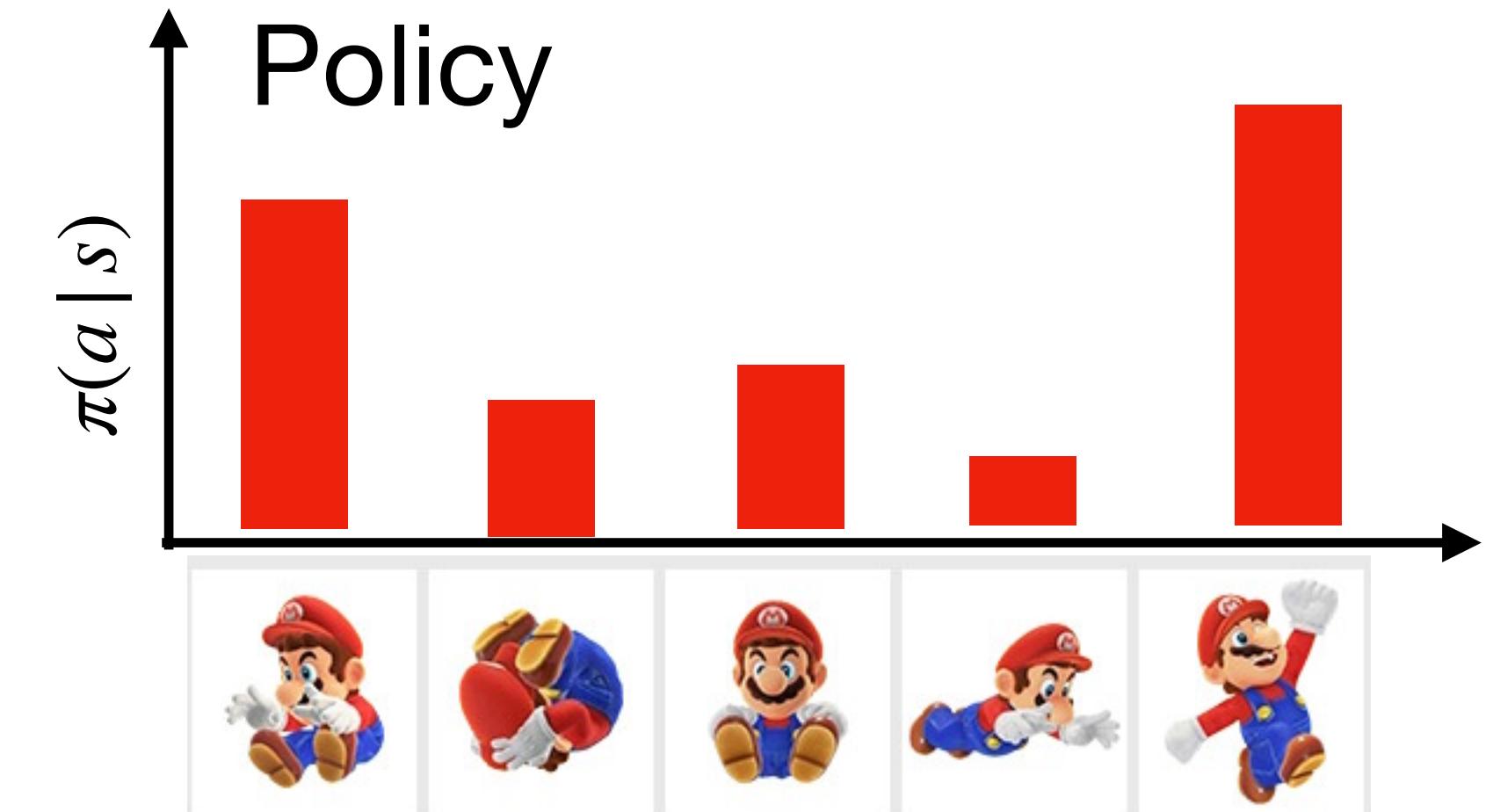
$$V_\pi(s) = \mathbb{E}_{\tau \sim \pi} \left[\sum_{t \in \tau} \gamma^t R_{t+1} \mid s_0 = s \right]$$

- We can rewrite the expectation $\mathbb{E}_{\tau \sim \pi}$ in terms of the **policy** and **state transitions**

$$V_\pi(s) = \sum_a \pi(a \mid s) \sum_{s'} P(s' \mid s, a) [R(s', a) + \gamma V_\pi(s')]$$

- The sum can be written recursively as **immediate reward** + **discounted future reward**

$$\gamma^0 = 1$$



Optimal policies via Bellman Equations

Memorizing these equations not necessary for exam

- This recursive formulation of the value function is known as **the Bellman equation**

$$V_{\pi}(s) = \sum_a \pi(a | s) \sum_{s'} P(s' | s, a) [R(s', a) + \gamma V_{\pi}(s')]$$

- This allows us to break the optimization problem into series of simpler sub-problems
 - if each sub-problem is solved optimally, the overall problem will also be optimal
- Note that there is no longer any **reward prediction error** updating
- Rather, we want to describe a **theoretically optimal solution**:
 - We first define an optimal value function by assuming value-maximizing actions:

$$V_*(s) = \arg \max_a \sum_{s'} P(s' | s, a) [R(s, a) + \gamma V_*(s')]$$

- We then (theoretically) arrive at an optimal policy by selecting actions that maximize value:

$$\pi_* = \arg \max_a V_*(s)$$

Optimal policies via Bellman Equations

Memorizing these equations not necessary for exam

- This recursive formulation of the value function is known as **the Bellman equation**

$$V_{\pi}(s) = \sum_a \pi(a | s) \sum_{s'} P(s' | s, a) [R(s', a) + \gamma V_{\pi}(s')]$$

- This allows us to break the optimization problem into series of simpler sub-problems
 - if each sub-problem is solved optimally, the overall problem will also be optimal
- Note that there is no longer any **reward prediction error** updating
- Rather, we want to describe a **theoretically optimal solution**:
 - We first define an optimal value function by assuming value-maximizing actions:

$$V_*(s) = \arg \max_a \sum_{s'} P(s' | s, a) [R(s, a) + \gamma V_*(s')]$$

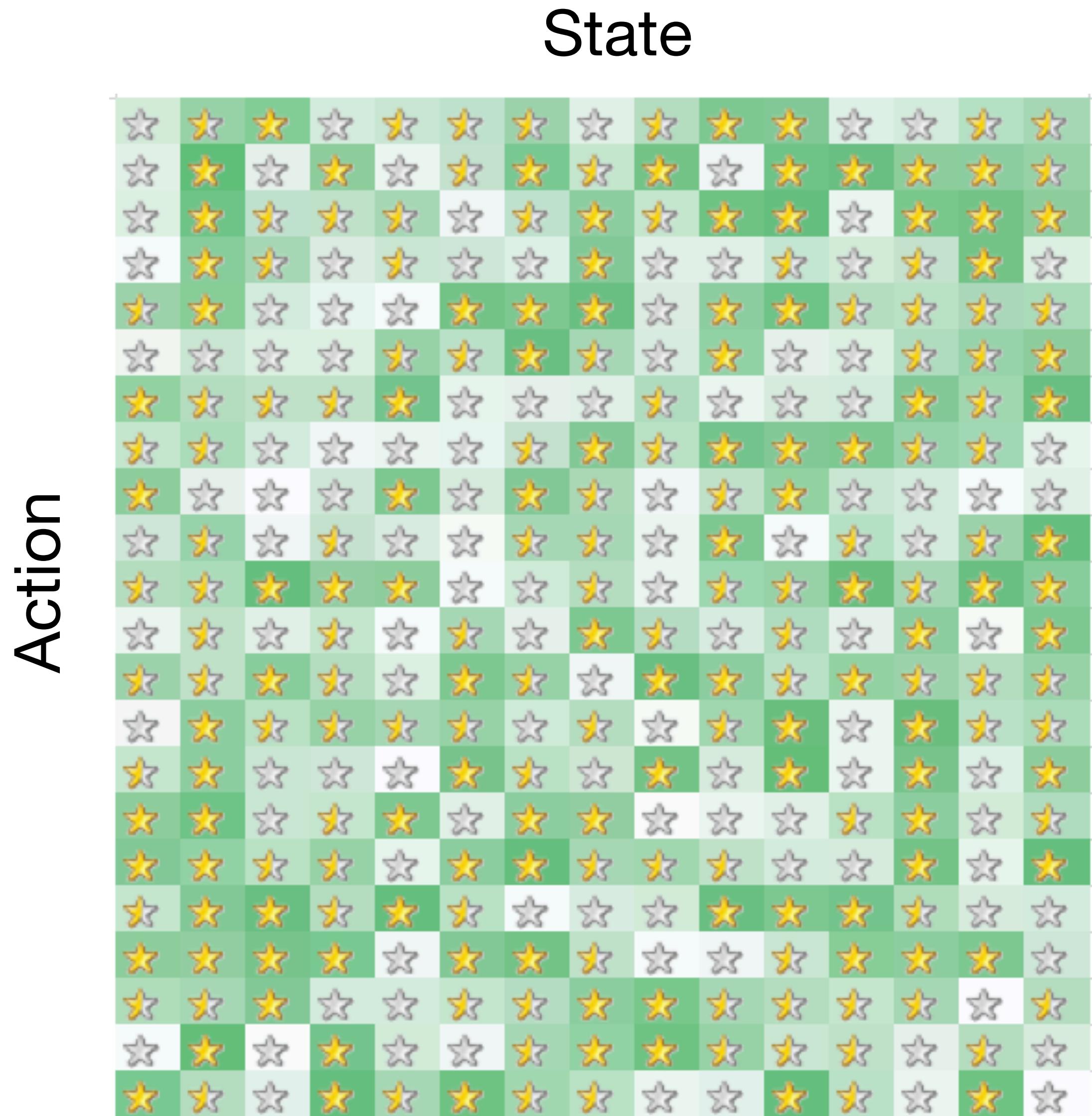
- We then (theoretically) arrive at an optimal policy by selecting actions that maximize value:

$$\pi_* = \arg \max_a V_*(s)$$

* In practice, optimal solutions are usually unobtainable

Tabular methods

- Based on methods from Dynamic programming (Bellman, 1957), Tabular methods were first proposed as solutions for RL problems by Minsky (1961)
- Think of a giant lookup table, where we store a value representation for each combination of state+action
- **Value iteration** and **policy iteration** are examples of tabular methods
- Caveat: solutions require repeat visits to each state, which is infeasible in most real-world problems



Value iteration

Iteratively visit all states and update the value function until a “good enough” solution has been reached.

1. Initialize the value function as $V_{k=0}(s) = 0$ for all states
2. For k in $(1, 2, \dots)$ update all s in \mathcal{S} :

$$V_{k+1}(s) = \max_{a \in A} \sum_{s'} P(s' | s, a)[R(s, a) + \gamma V_k(s')]$$

until $\max_{s \in \mathcal{S}} |V_k(s) - V_{k-1}(s)| < \theta$ Bellman residual

V_k converges on V_* as $k \rightarrow \infty$, and perhaps sooner, but with many costly sweeps through the state space

Value iteration

Pieter Abbeel

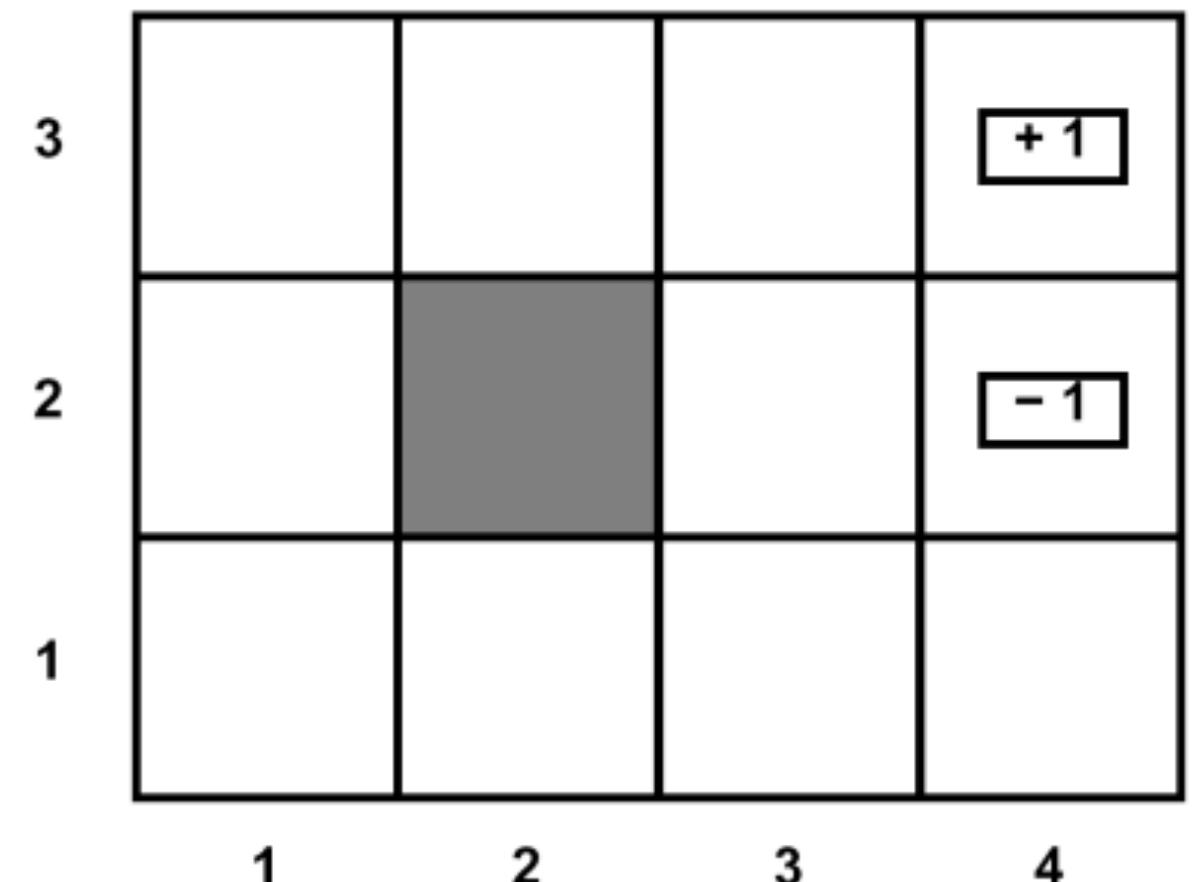
Iteratively visit all states and update the value function until a “good enough” solution has been reached.

1. Initialize the value function as $V_{k=0}(s) = 0$ for all states
2. For k in $(1, 2, \dots)$ update all s in \mathcal{S} :

$$V_{k+1}(s) = \max_{a \in A} \sum_{s'} P(s' | s, a)[R(s, a) + \gamma V_k(s')]$$

until $\max_{s \in \mathcal{S}} |V_k(s) - V_{k-1}(s)| < \theta$ Bellman residual

V_k converges on V_* as $k \rightarrow \infty$, and perhaps sooner, but with many costly sweeps through the state space



Value iteration

Pieter Abbeel

Iteratively visit all states and update the value function until a “good enough” solution has been reached.

1. Initialize the value function as $V_{k=0}(s) = 0$ for all states
2. For k in $(1, 2, \dots)$ update all s in \mathcal{S} :

$$V_{k+1}(s) = \max_{a \in A} \sum_{s'} P(s' | s, a)[R(s, a) + \gamma V_k(s')]$$

until $\max_{s \in \mathcal{S}} |V_k(s) - V_{k-1}(s)| < \theta$ Bellman residual

V_k converges on V_* as $k \rightarrow \infty$, and perhaps sooner, but with many costly sweeps through the state space

3				+ 1
2		■		- 1
1				
	1	2	3	4

0.00	0.00	0.00	1.00
0.00		0.00	-1.00
0.00	0.00	0.00	0.00
VALUES AFTER 1 ITERATIONS			

Value iteration

Pieter Abbeel

Iteratively visit all states and update the value function until a “good enough” solution has been reached.

1. Initialize the value function as $V_{k=0}(s) = 0$ for all states
2. For k in $(1, 2, \dots)$ update all s in \mathcal{S} :

$$V_{k+1}(s) = \max_{a \in A} \sum_{s'} P(s' | s, a)[R(s, a) + \gamma V_k(s')]$$

until $\max_{s \in \mathcal{S}} |V_k(s) - V_{k-1}(s)| < \theta$ Bellman residual

V_k converges on V_* as $k \rightarrow \infty$, and perhaps sooner, but with many costly sweeps through the state space

3				+ 1
2		■		- 1
1				
	1	2	3	4

0.00	0.00	0.72	1.00
0.00		0.00	-1.00
0.00	0.00	0.00	0.00
VALUES AFTER 2 ITERATIONS			

Value iteration

Pieter Abbeel

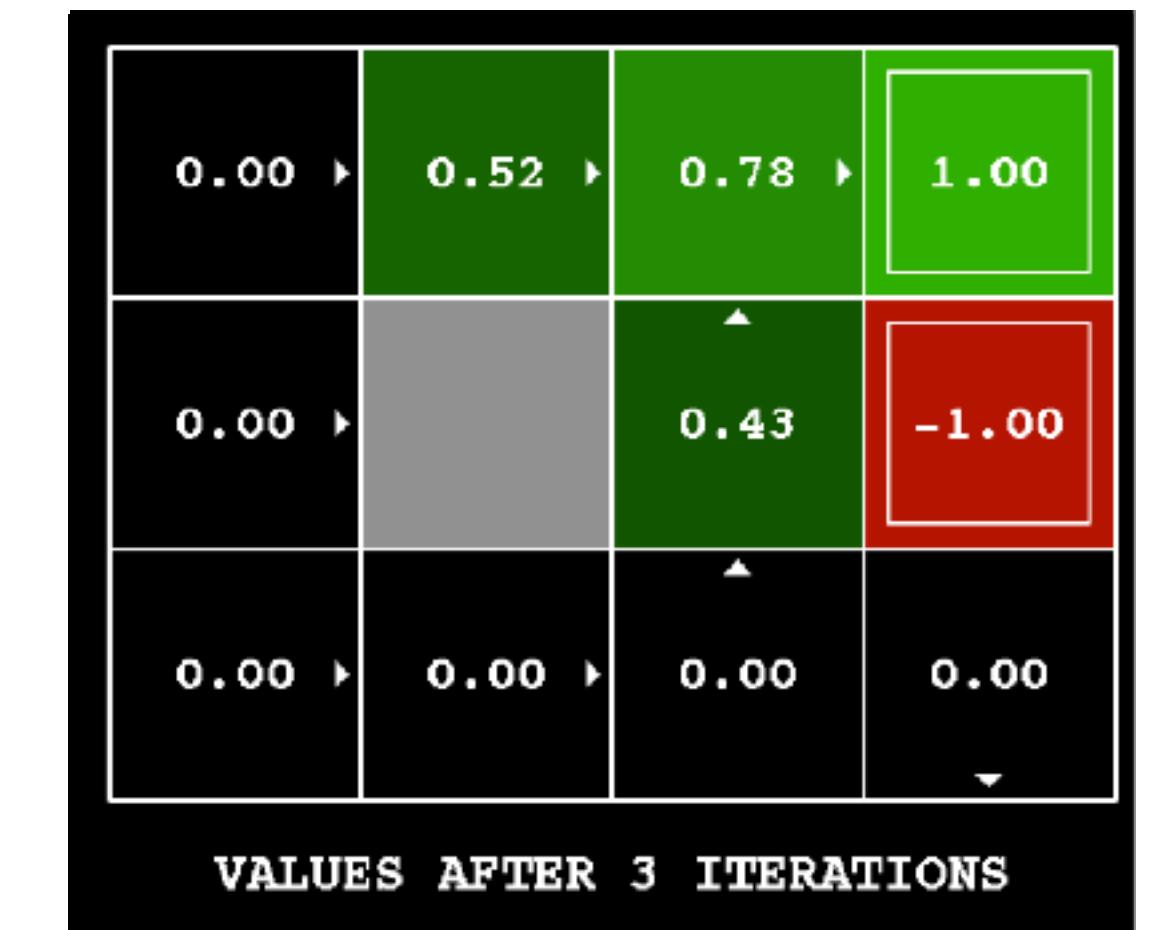
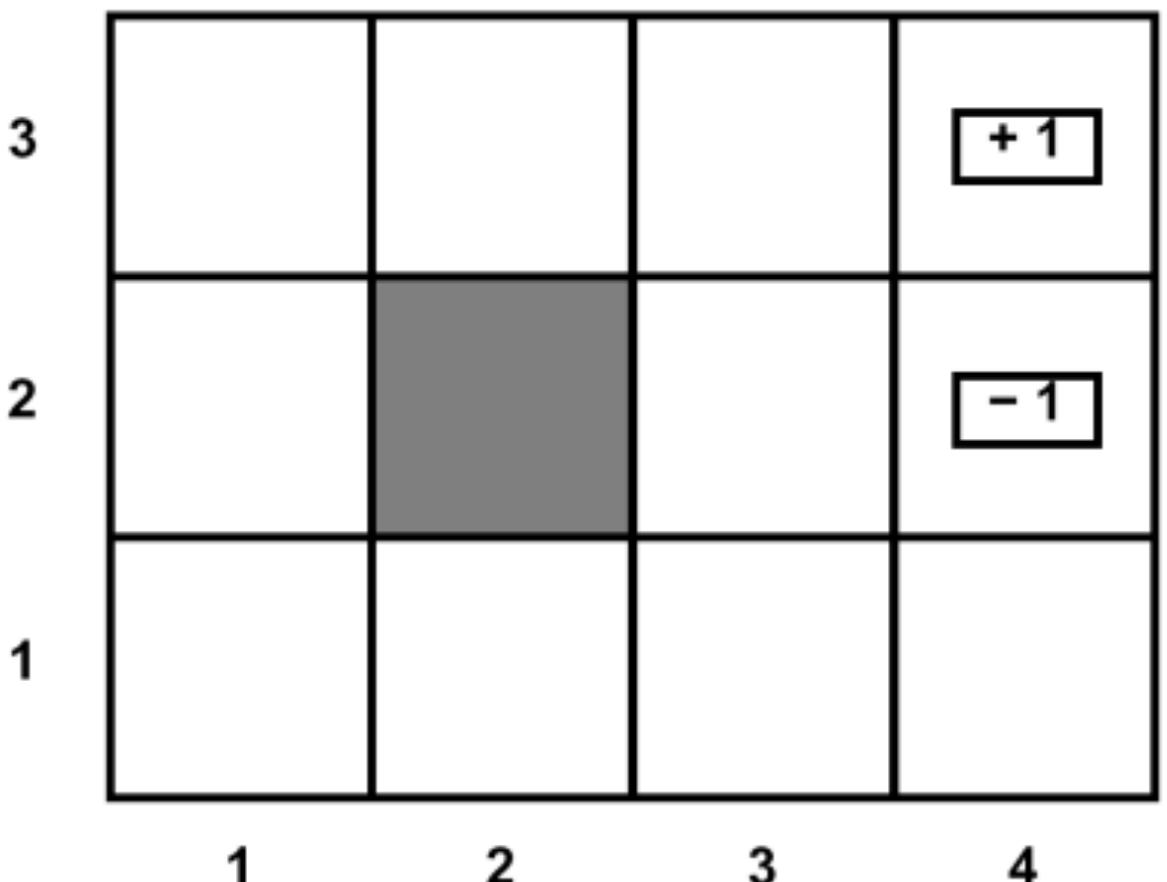
Iteratively visit all states and update the value function until a “good enough” solution has been reached.

1. Initialize the value function as $V_{k=0}(s) = 0$ for all states
2. For k in $(1, 2, \dots)$ update all s in \mathcal{S} :

$$V_{k+1}(s) = \max_{a \in A} \sum_{s'} P(s' | s, a)[R(s, a) + \gamma V_k(s')]$$

until $\max_{s \in \mathcal{S}} |V_k(s) - V_{k-1}(s)| < \theta$ Bellman residual

V_k converges on V_* as $k \rightarrow \infty$, and perhaps sooner, but with many costly sweeps through the state space



Value iteration

Pieter Abbeel

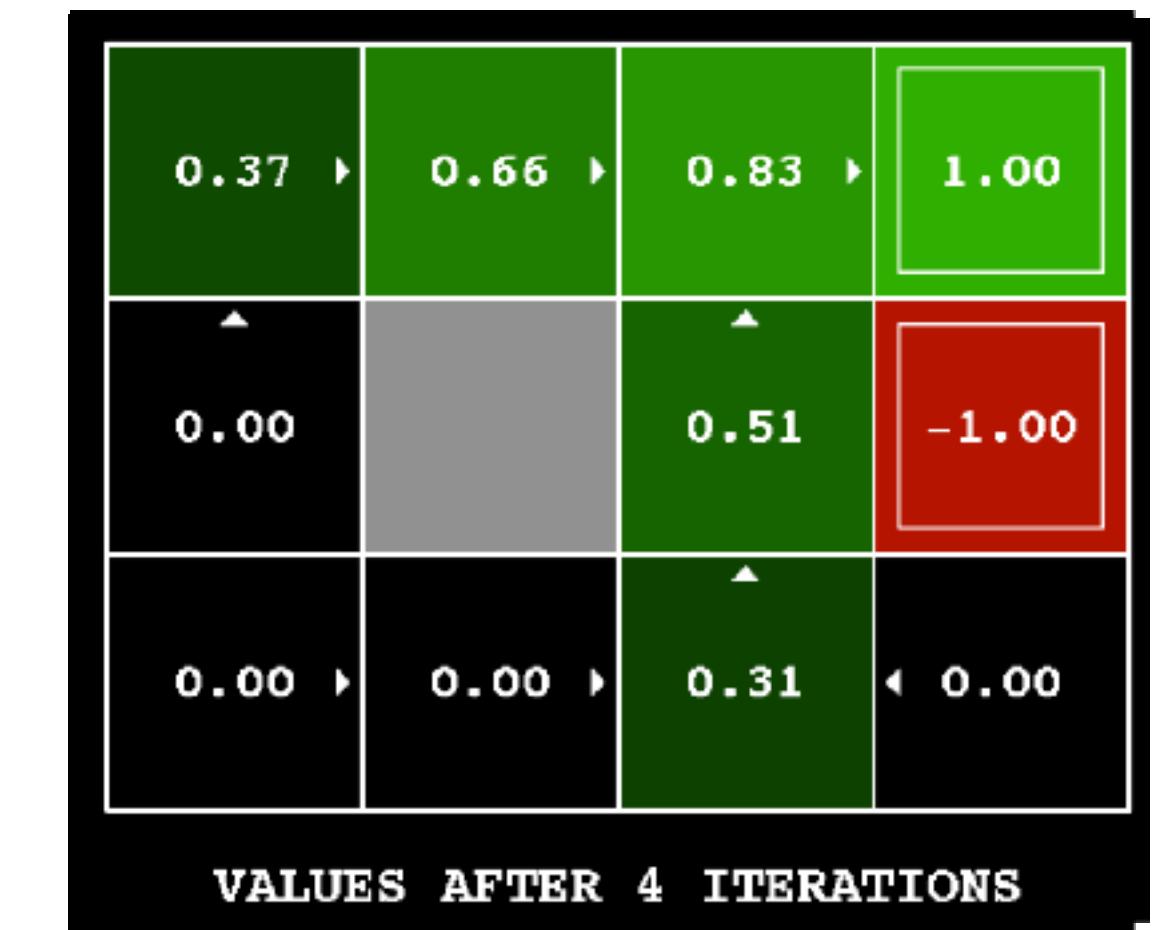
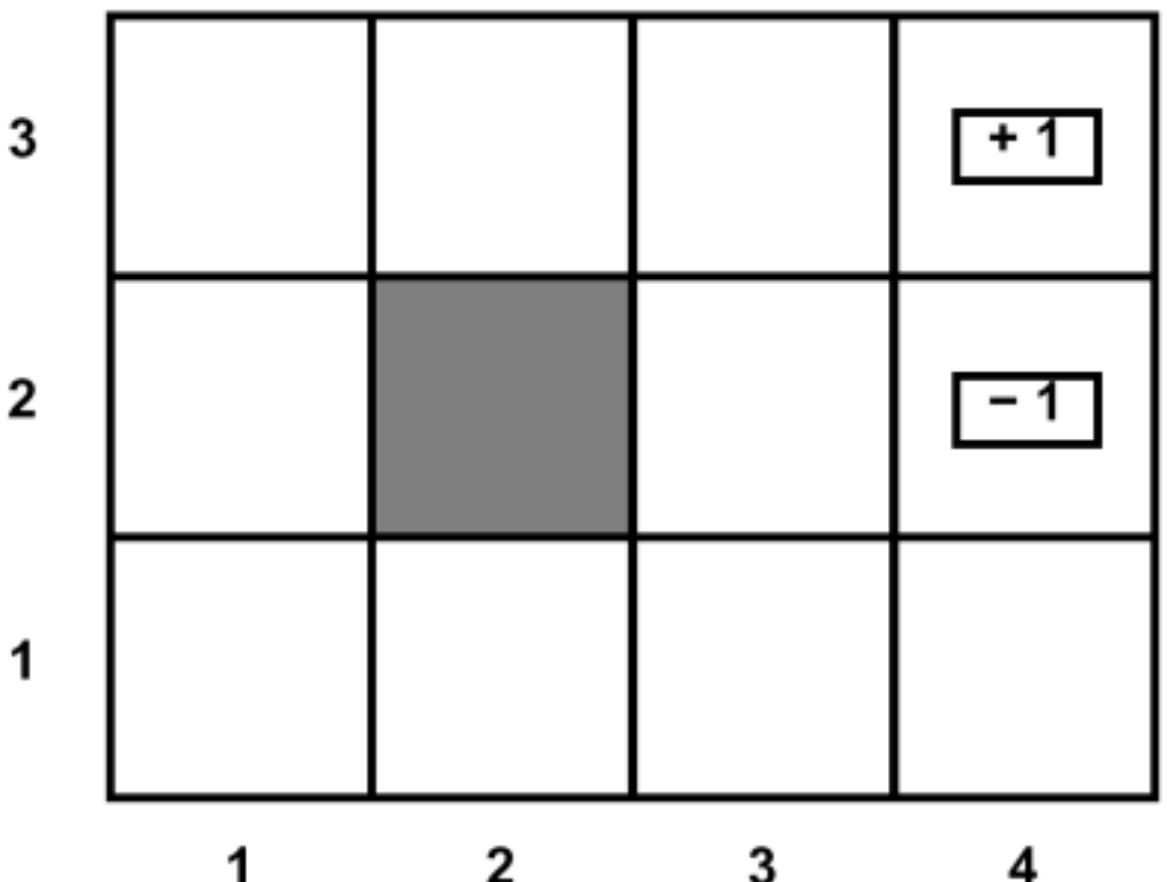
Iteratively visit all states and update the value function until a “good enough” solution has been reached.

1. Initialize the value function as $V_{k=0}(s) = 0$ for all states
2. For k in $(1, 2, \dots)$ update all s in \mathcal{S} :

$$V_{k+1}(s) = \max_{a \in A} \sum_{s'} P(s' | s, a)[R(s, a) + \gamma V_k(s')]$$

until $\max_{s \in \mathcal{S}} |V_k(s) - V_{k-1}(s)| < \theta$ Bellman residual

V_k converges on V_* as $k \rightarrow \infty$, and perhaps sooner, but with many costly sweeps through the state space



Value iteration

Pieter Abbeel

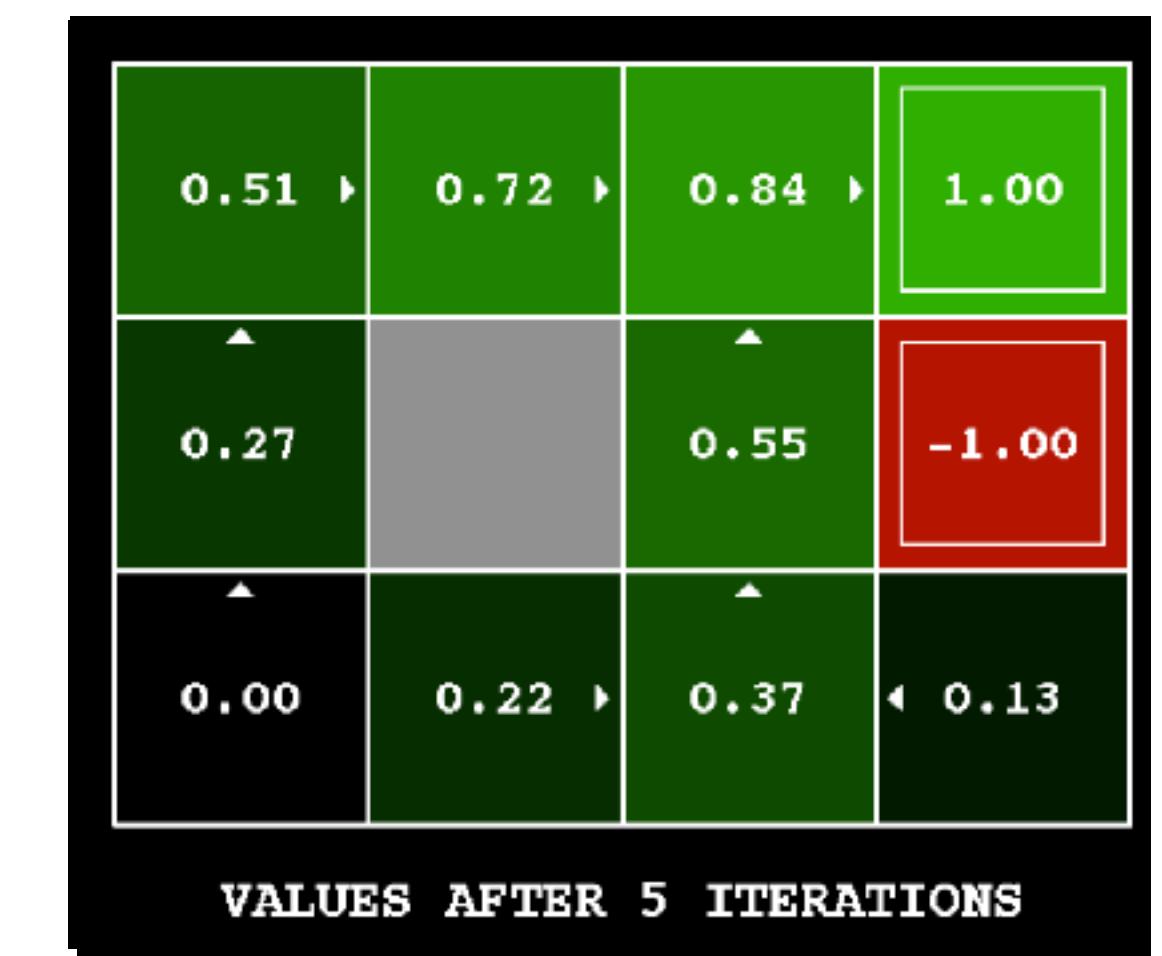
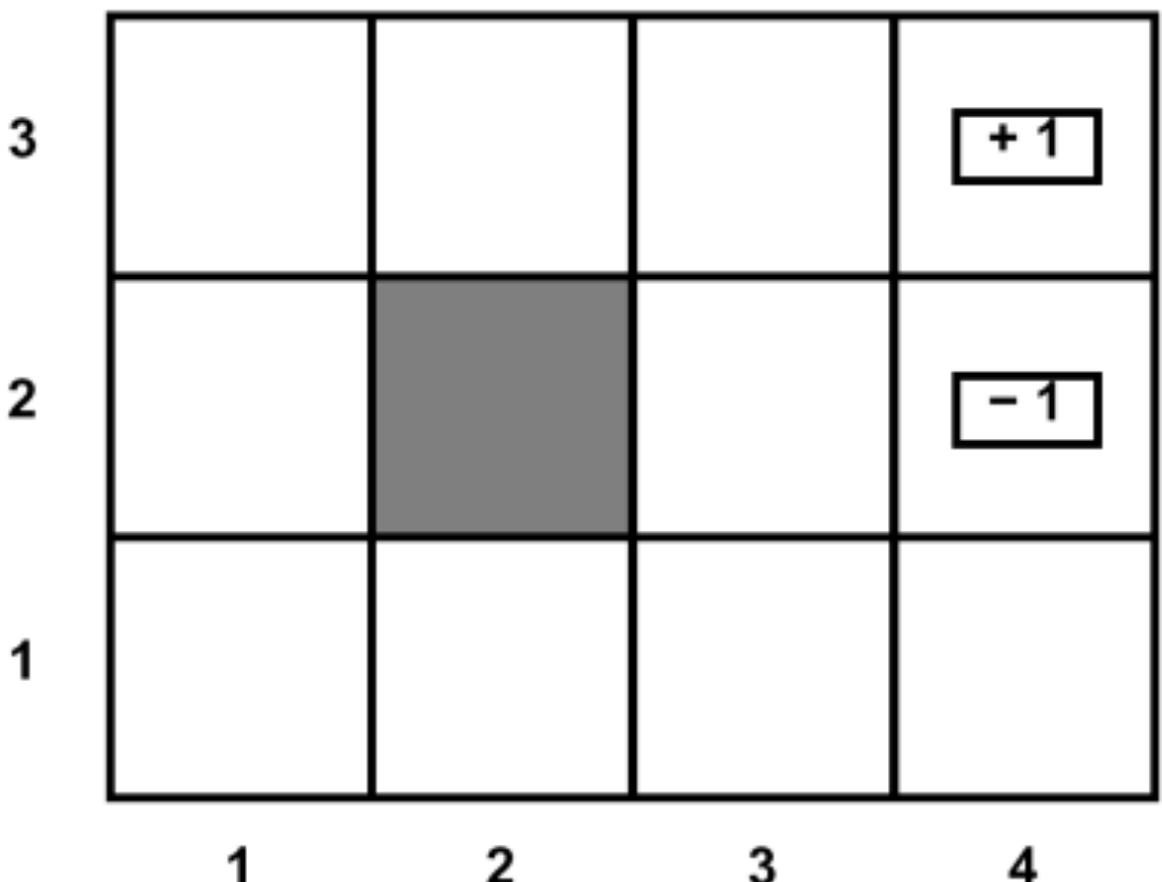
Iteratively visit all states and update the value function until a “good enough” solution has been reached.

1. Initialize the value function as $V_{k=0}(s) = 0$ for all states
2. For k in $(1, 2, \dots)$ update all s in \mathcal{S} :

$$V_{k+1}(s) = \max_{a \in A} \sum_{s'} P(s' | s, a)[R(s, a) + \gamma V_k(s')]$$

until $\max_{s \in \mathcal{S}} |V_k(s) - V_{k-1}(s)| < \theta$ Bellman residual

V_k converges on V_* as $k \rightarrow \infty$, and perhaps sooner, but with many costly sweeps through the state space



Value iteration

Pieter Abbeel

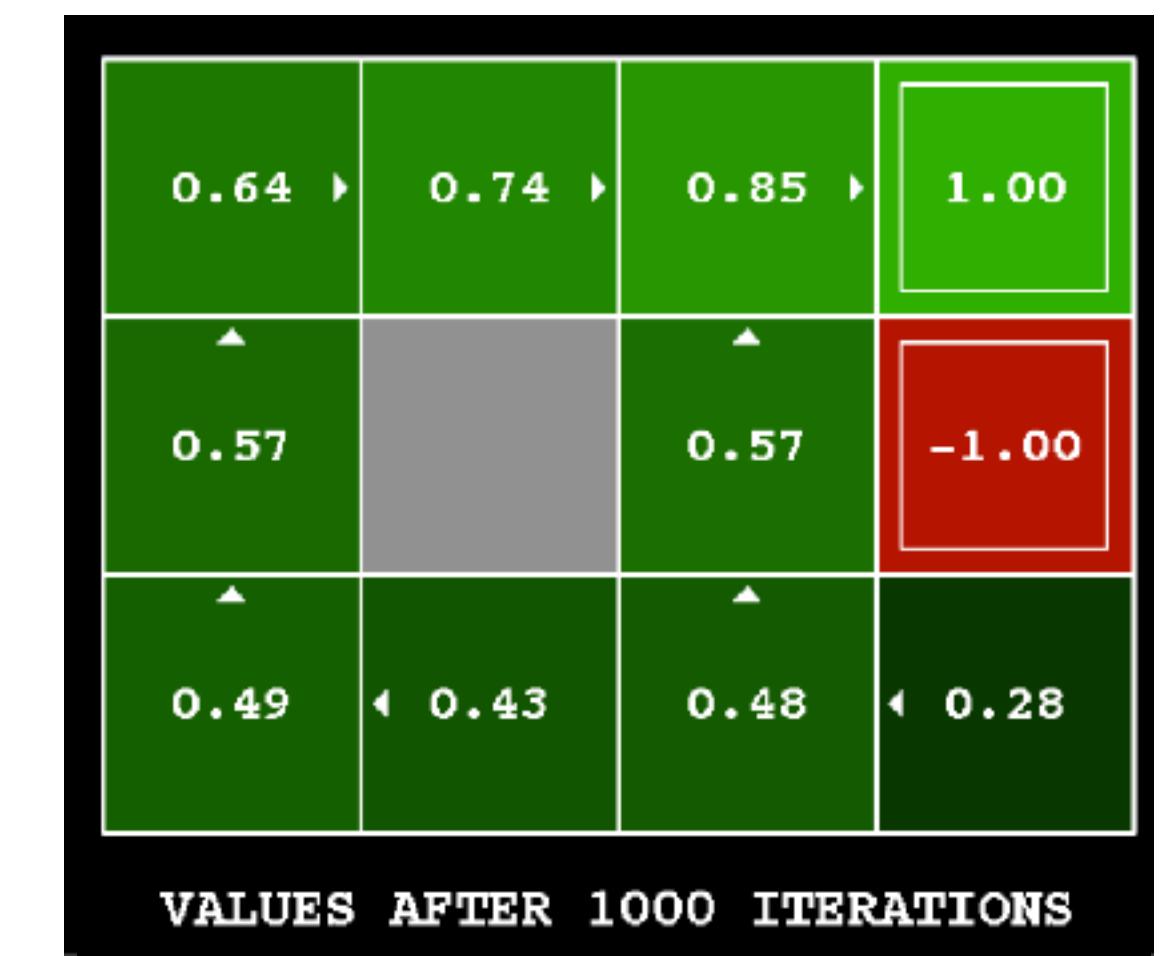
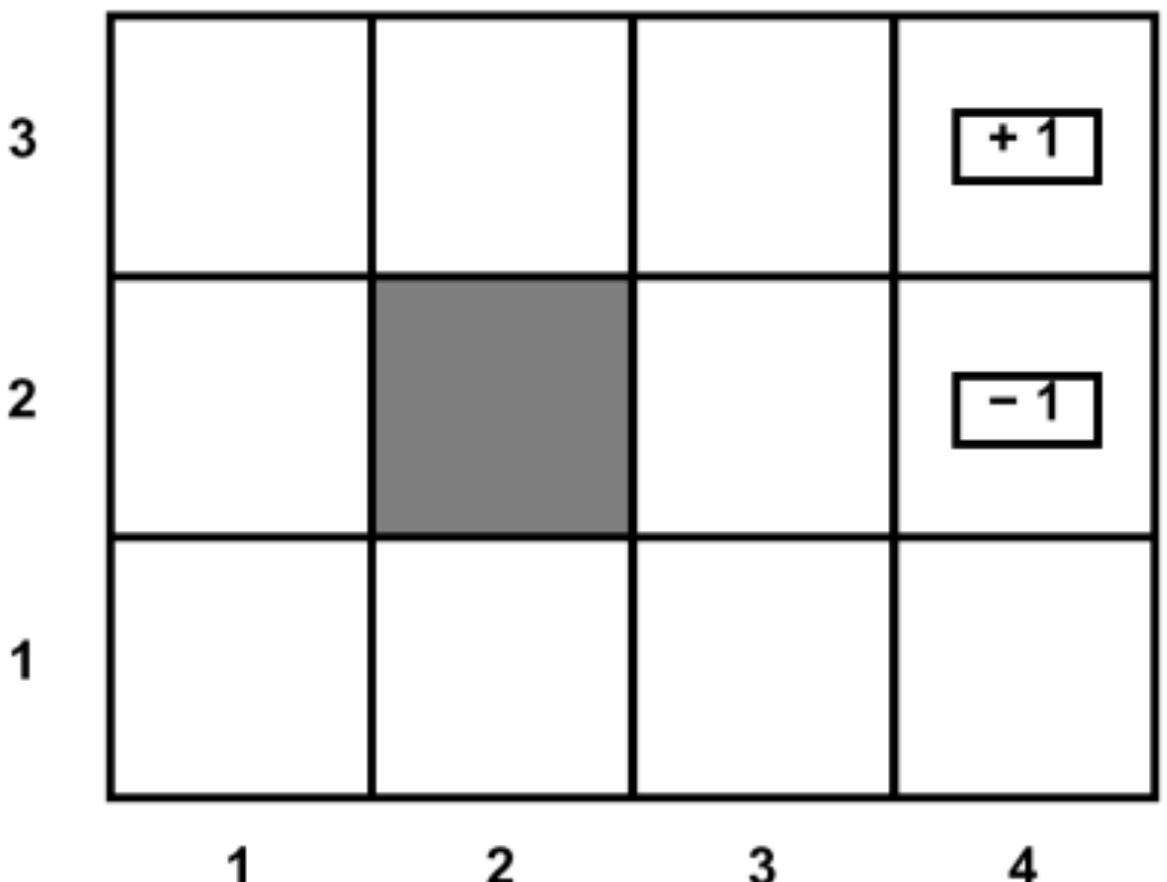
Iteratively visit all states and update the value function until a “good enough” solution has been reached.

1. Initialize the value function as $V_{k=0}(s) = 0$ for all states
2. For k in $(1, 2, \dots)$ update all s in \mathcal{S} :

$$V_{k+1}(s) = \max_{a \in A} \sum_{s'} P(s' | s, a)[R(s, a) + \gamma V_k(s')]$$

until $\max_{s \in \mathcal{S}} |V_k(s) - V_{k-1}(s)| < \theta$ Bellman residual

V_k converges on V_* as $k \rightarrow \infty$, and perhaps sooner, but with many costly sweeps through the state space



Policy iteration

Alternate between evaluating a policy and then improving the policy.

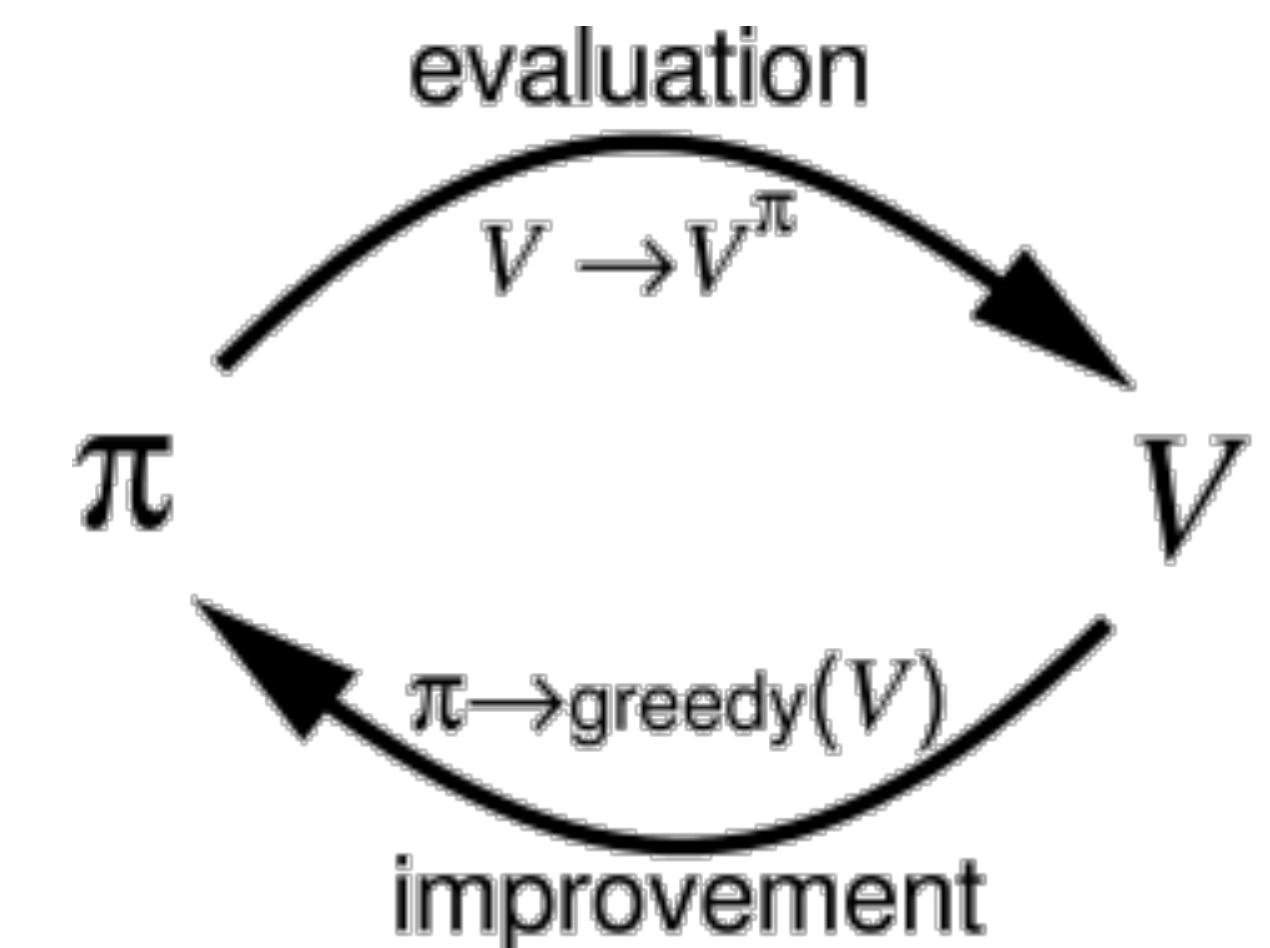
Start with π_0 (typically a random policy), and then iterate for all $s \in \mathcal{S}$ in each step

- **Policy Evaluation**

$$V_{\pi_k}(s) = \mathbb{E}_{\pi_k} \left[R(s', a) + \gamma V_{\pi_k}(s') \right]$$

- **Policy Improvement**

$$\pi_{k+1} = \arg \max_a \sum_{s'} P(s' | s, a) \left[R(s, a) + \gamma V_{\pi_k} \right]$$



Policy iteration

Alternate between evaluating a policy and then improving the policy.

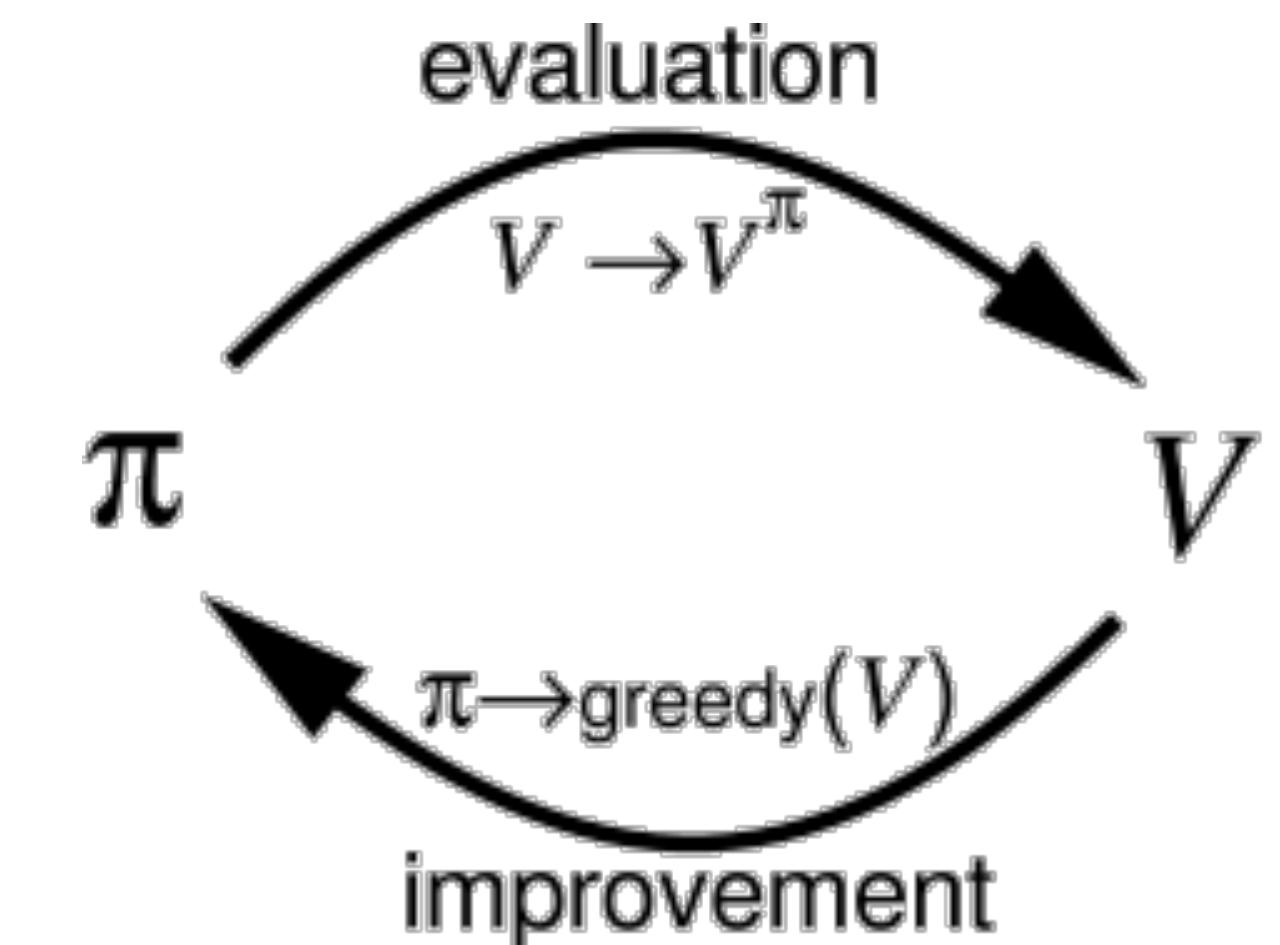
Start with π_0 (typically a random policy), and then iterate for all $s \in \mathcal{S}$ in each step

- **Policy Evaluation**

$$V_{\pi_k}(s) = \mathbb{E}_{\pi_k} \left[R(s', a) + \gamma V_{\pi_k}(s') \right]$$

- **Policy Improvement**

$$\pi_{k+1} = \arg \max_a \sum_{s'} P(s' | s, a) \left[R(s, a) + \gamma V_{\pi_k} \right]$$



Policy can converge faster than value function, but still requires visiting all states multiple times and lacks convergence guarantees

Challenge 2: Generalization in large action space

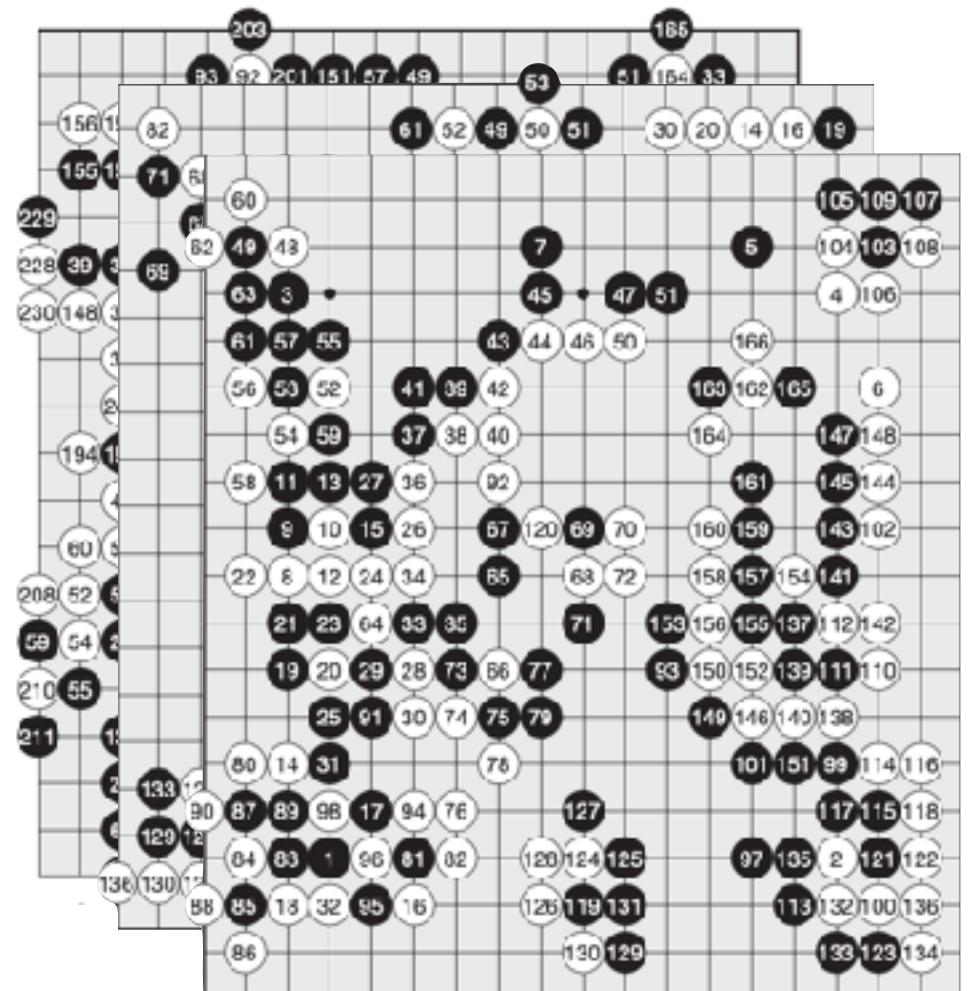
- What do you do when the number of states and actions are too large to visit?

Challenge 2: Generalization in large action space

- What do you do when the number of states and actions are too large to visit?

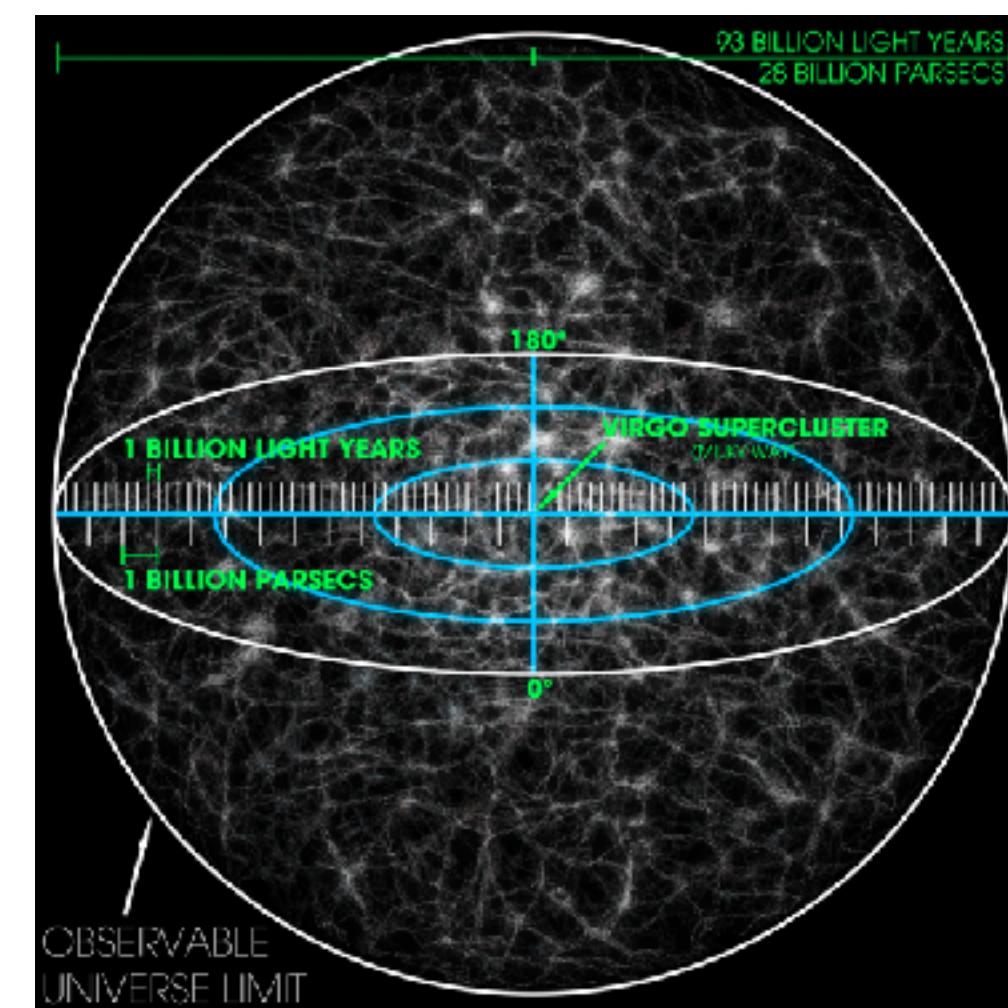
Game states

2.1×10^{170}



Atoms in observable
Universe

10^{80}

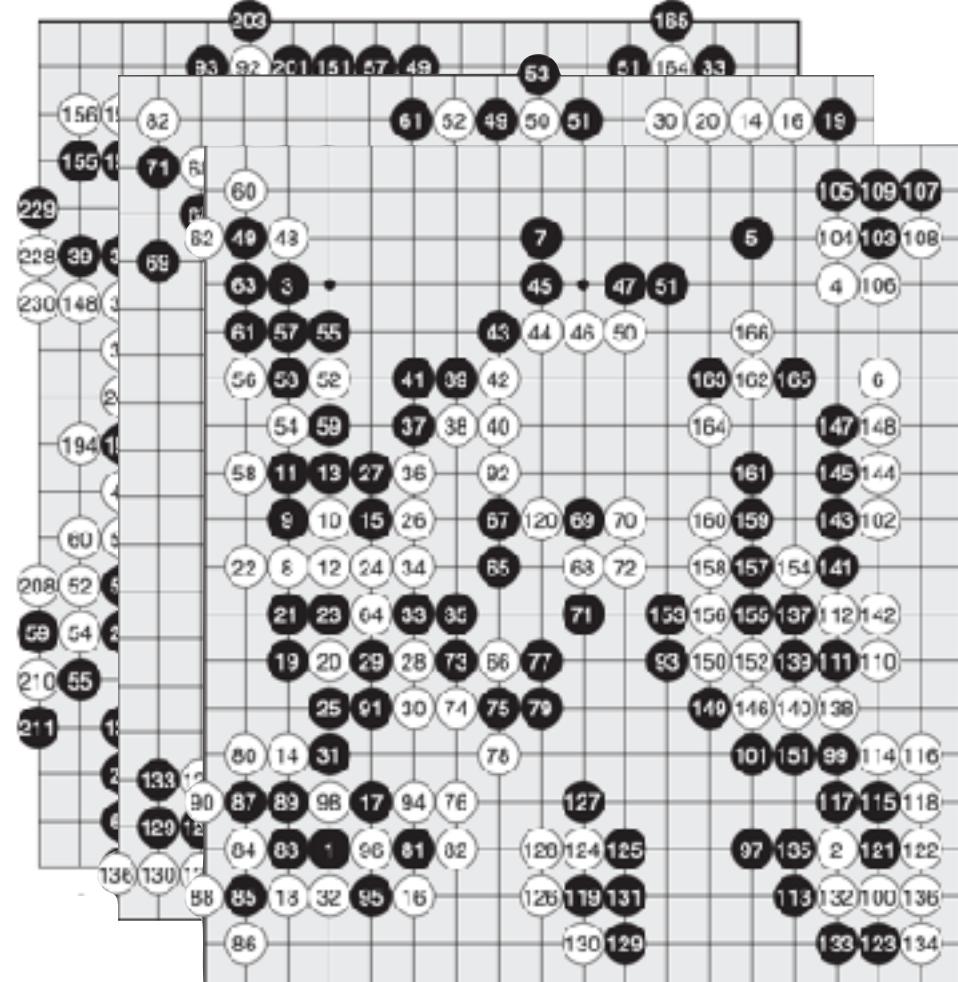


Challenge 2: Generalization in large action space

- What do you do when the number of states and actions are too large to visit?
- **Function approximation:** learn a function mapping states/actions to value, and generalize via interpolation/extrapolation

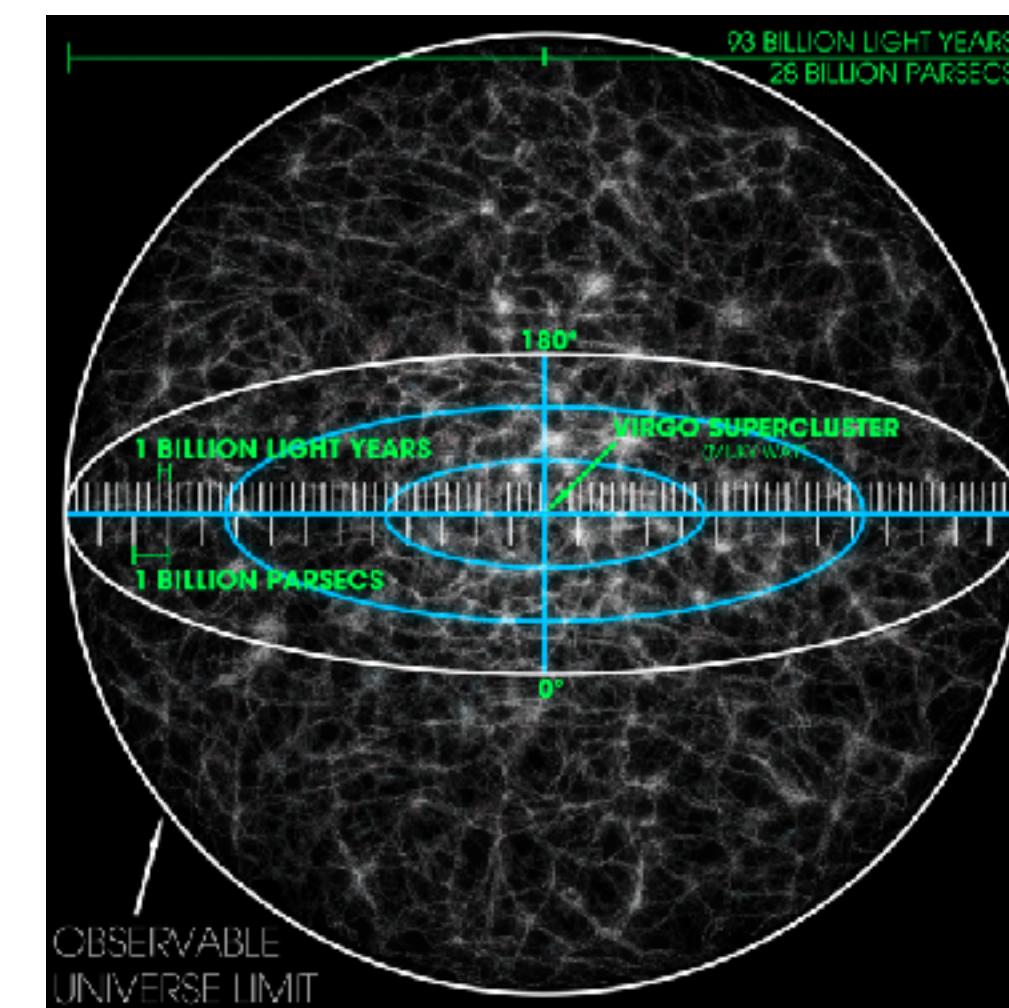
Game states

$$2.1 \times 10^{170}$$



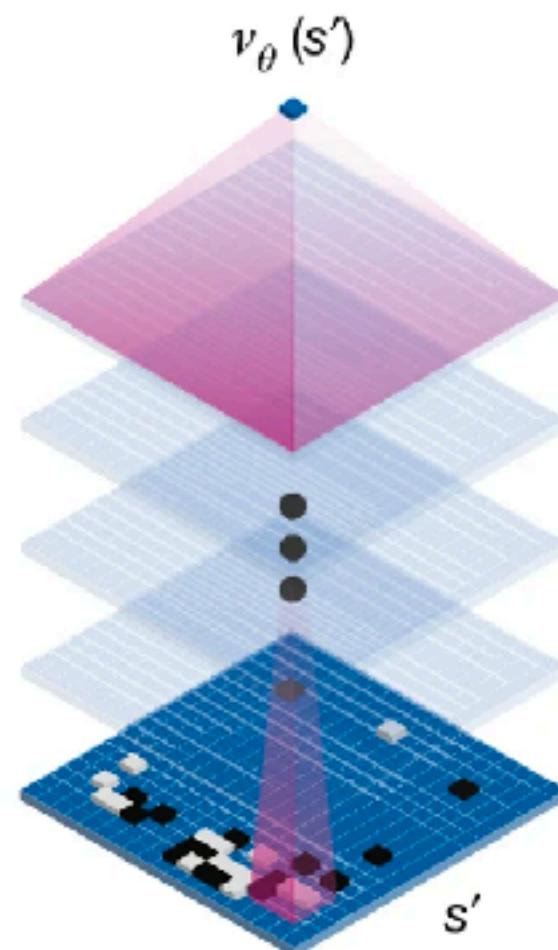
Atoms in observable Universe

$$10^{80}$$

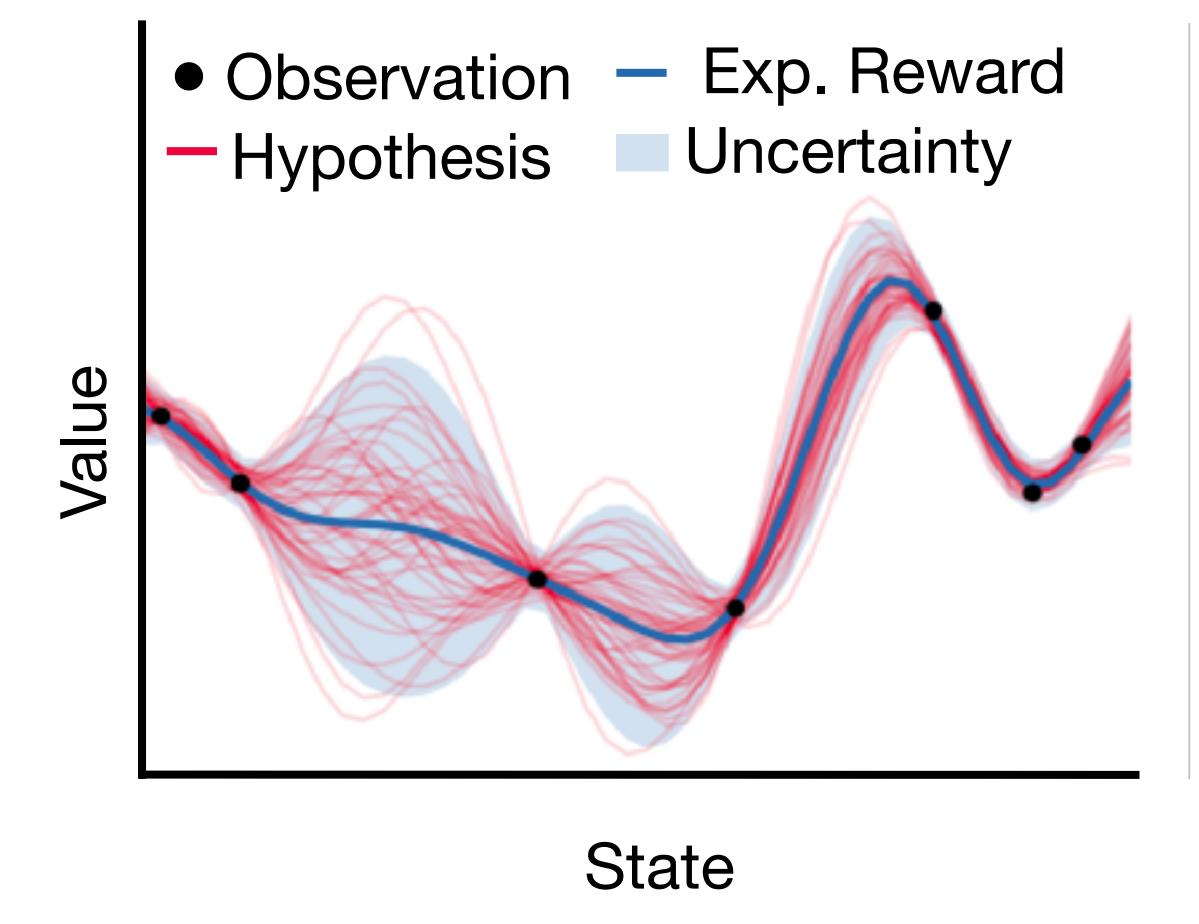


Function Approximation (Weeks 5 & 10)

$$V_{\theta}(s) := f(s, \theta)$$



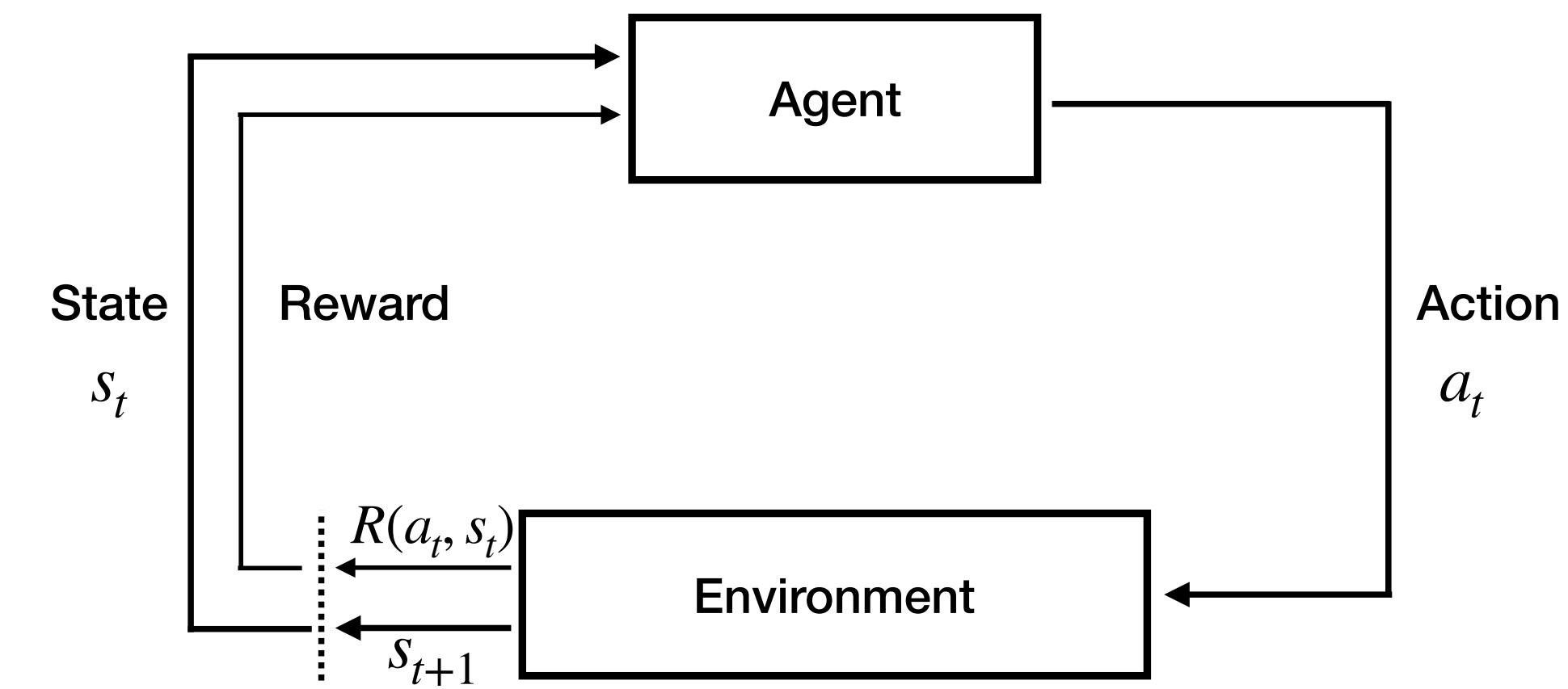
Silver et al., (Nature 2016)



Wu et al., (AnnRevPsych 2024)

RL summary

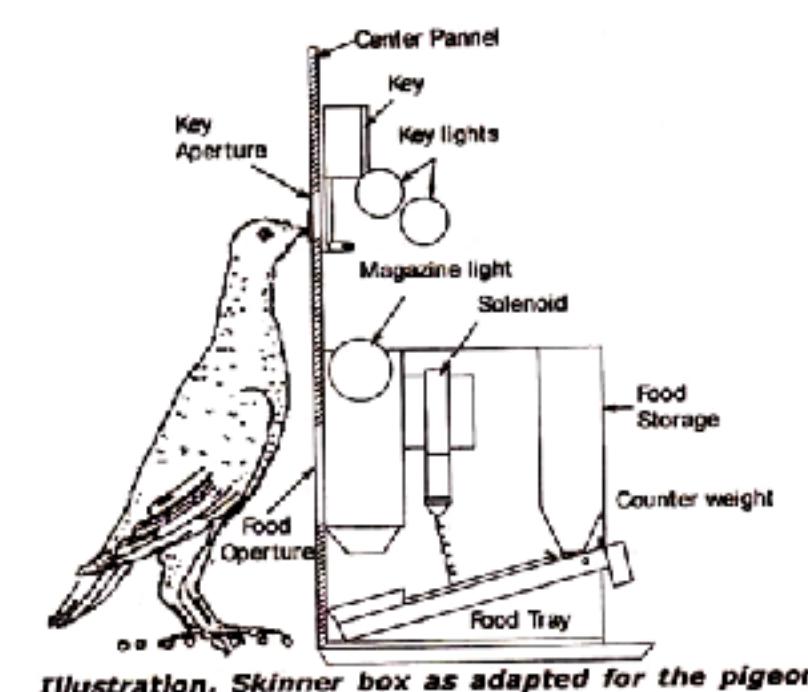
- **Normative** framework for learning an optimal policy π^* in arbitrarily complex environments
- Simplest setting is a 2-armed bandit problem, where Q-learning is equivalent to Rescorla-Wagner
- More complex settings require *credit assignment* and *generalization*
- RL also provides a **descriptive** model of human learning
- TD-learning prediction error tracks dopamine signals in the brain and widely used to study human behavior (more on this next week)



5 minute break

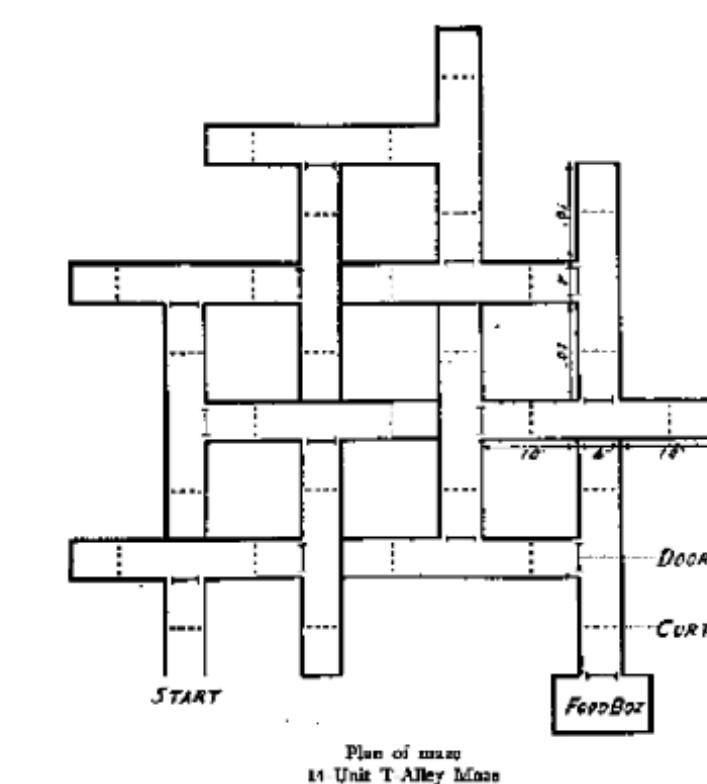
Model-free RL

- Habit
- Cheap
- $Q(s, a)$
- Myopically selecting actions that have been associated with reward



Model-based RL

- Goal-directed
- Computationally costly
- $P(s', r | s, a)$
- Planning and seeking of long term outcomes



Monte carlo tree search

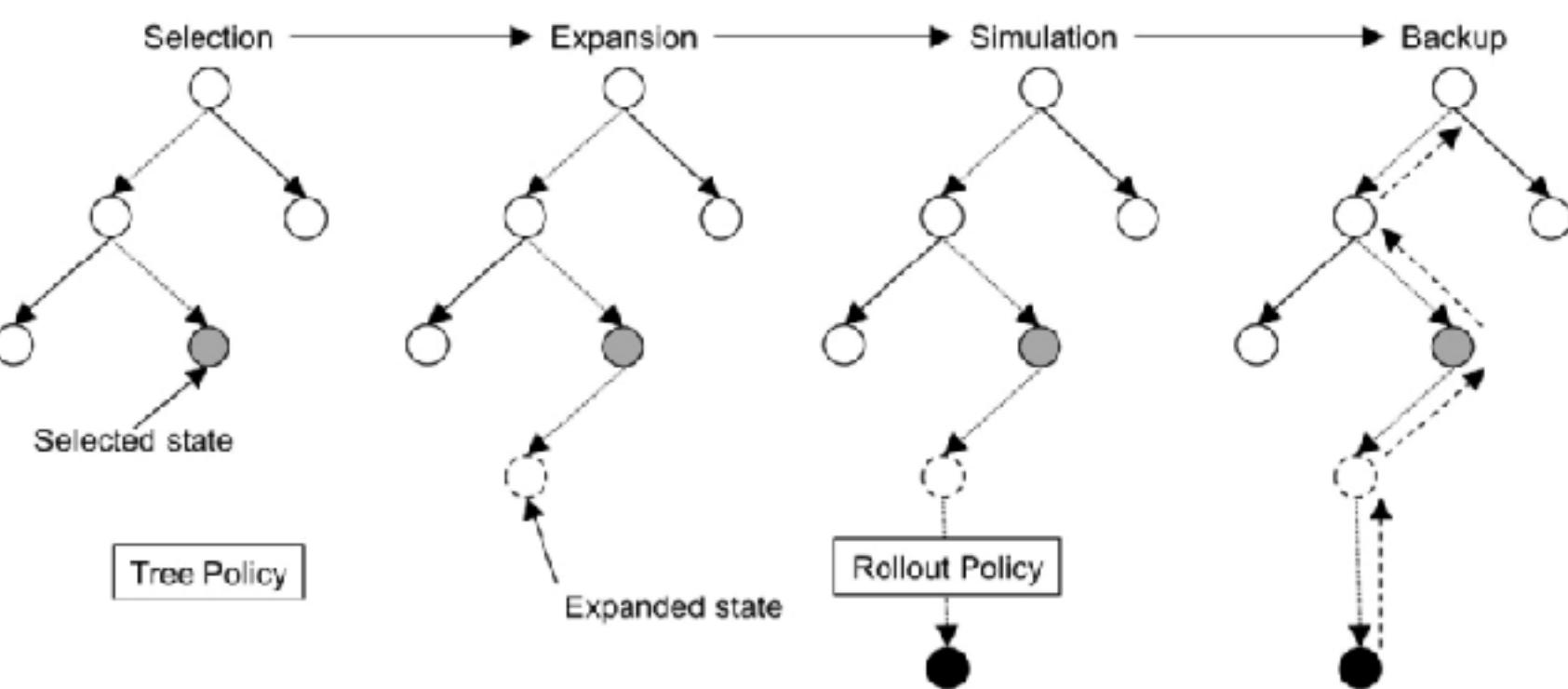


image credit Alyssa Dayan (from Dolan & Dayan, 2013)

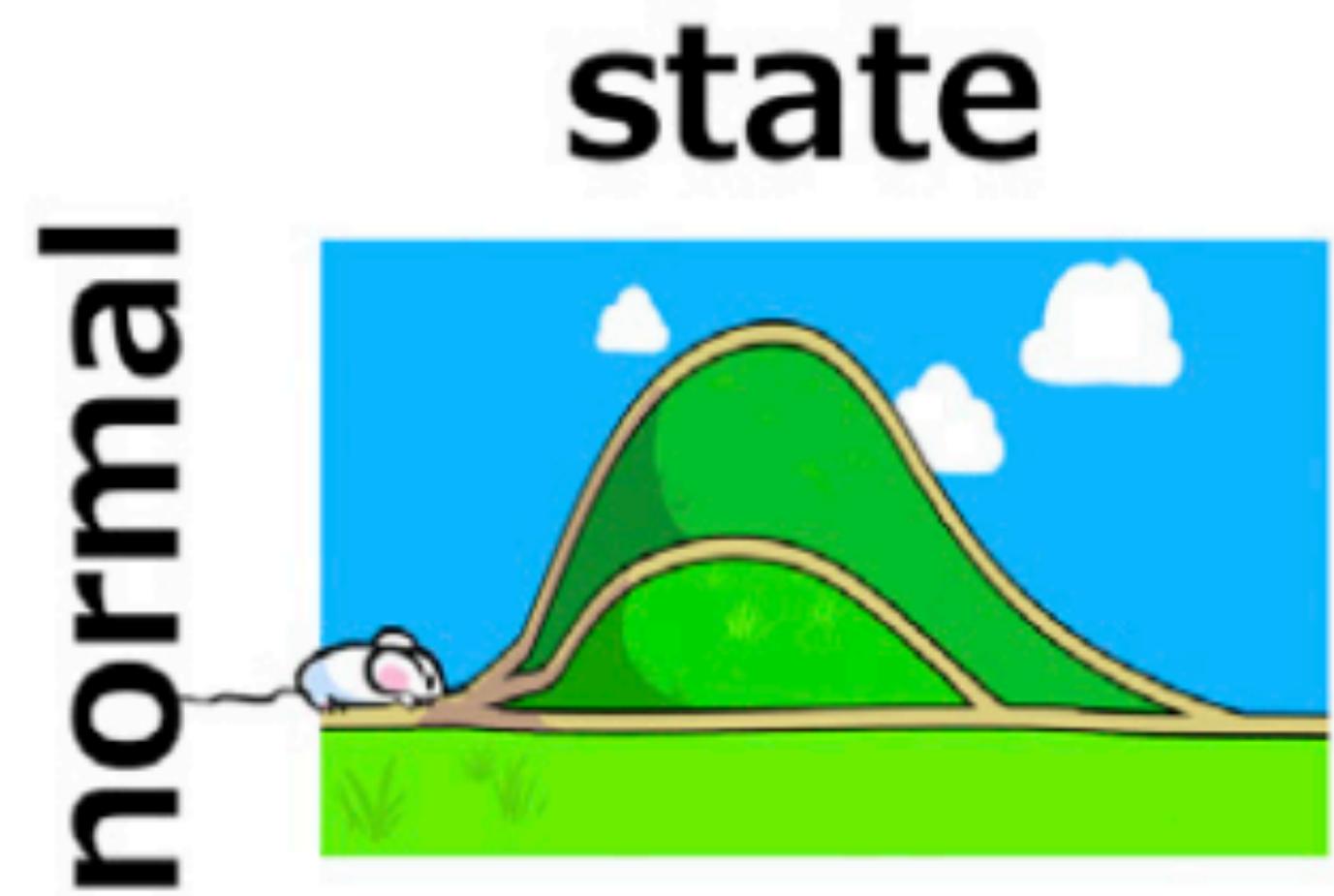


image credit Alyssa Dayan (from Dolan & Dayan, 2013)

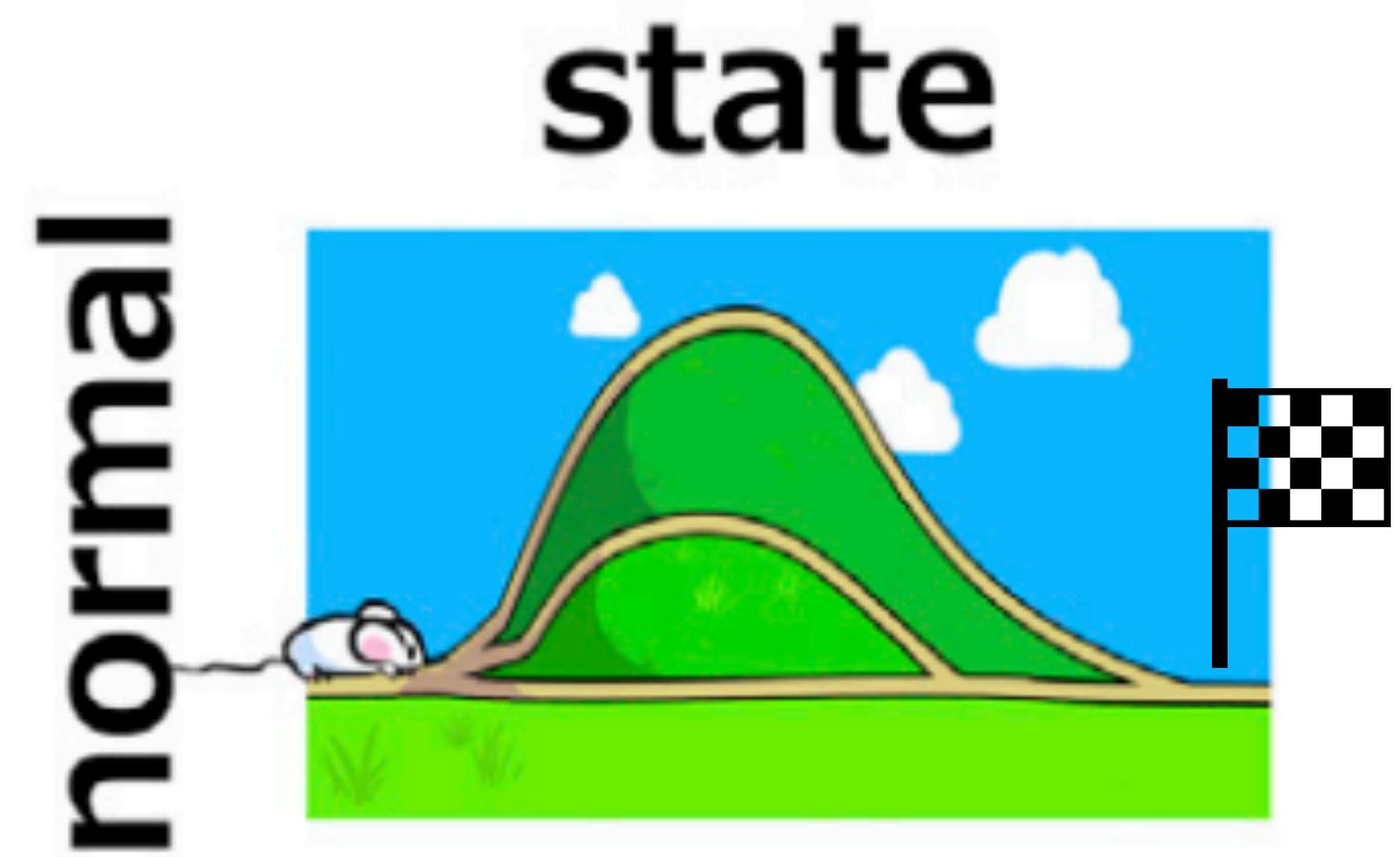
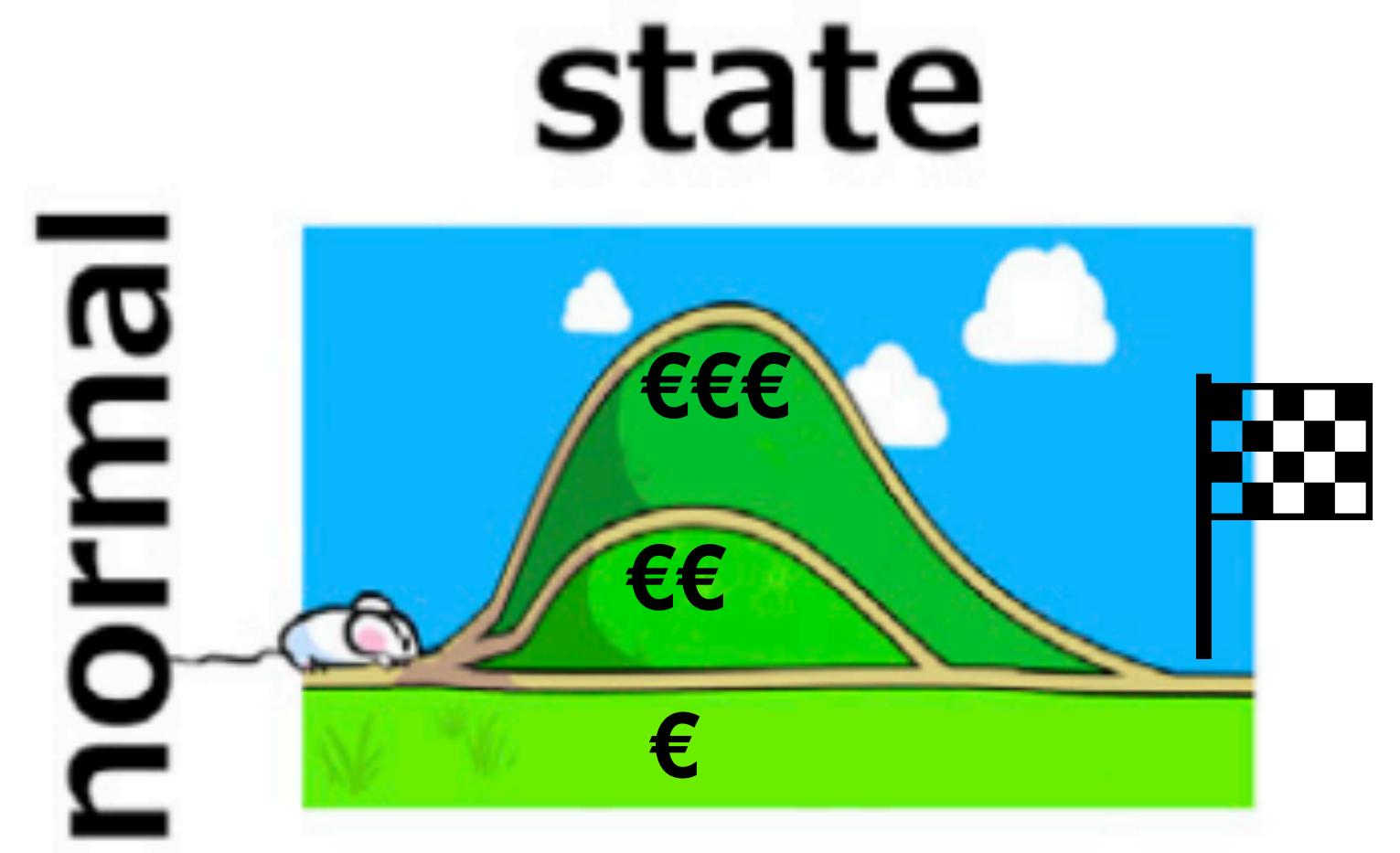
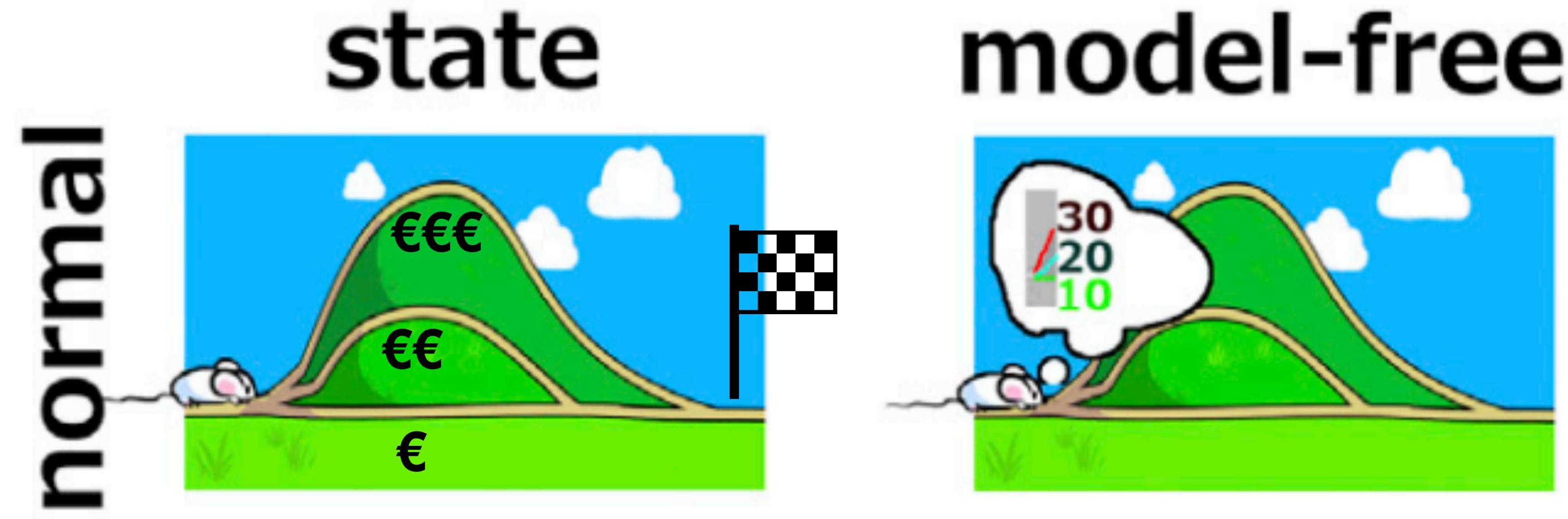
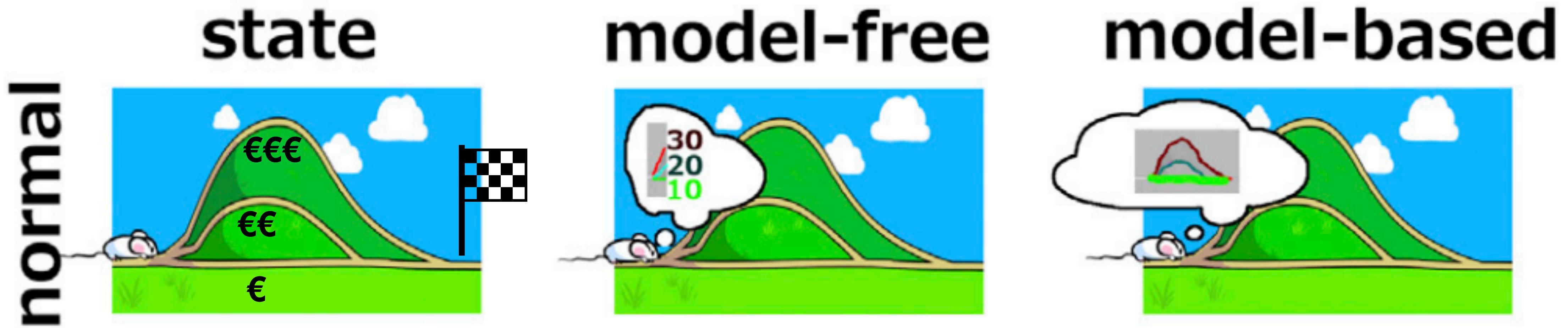


image credit Alyssa Dayan (from Dolan & Dayan, 2013)

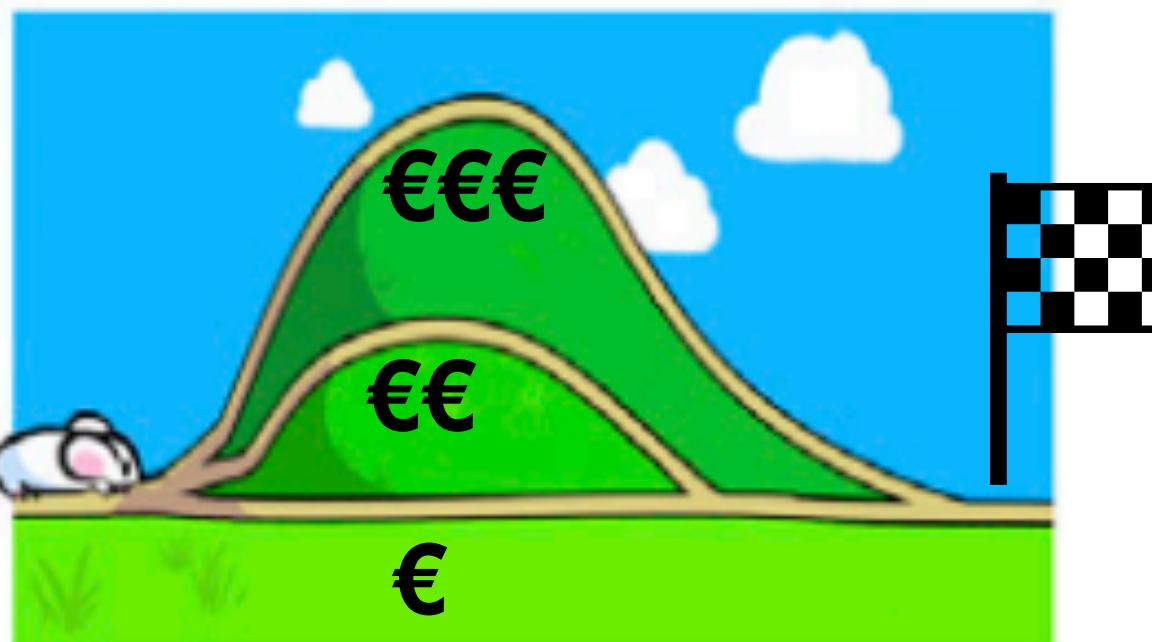






blocked normal

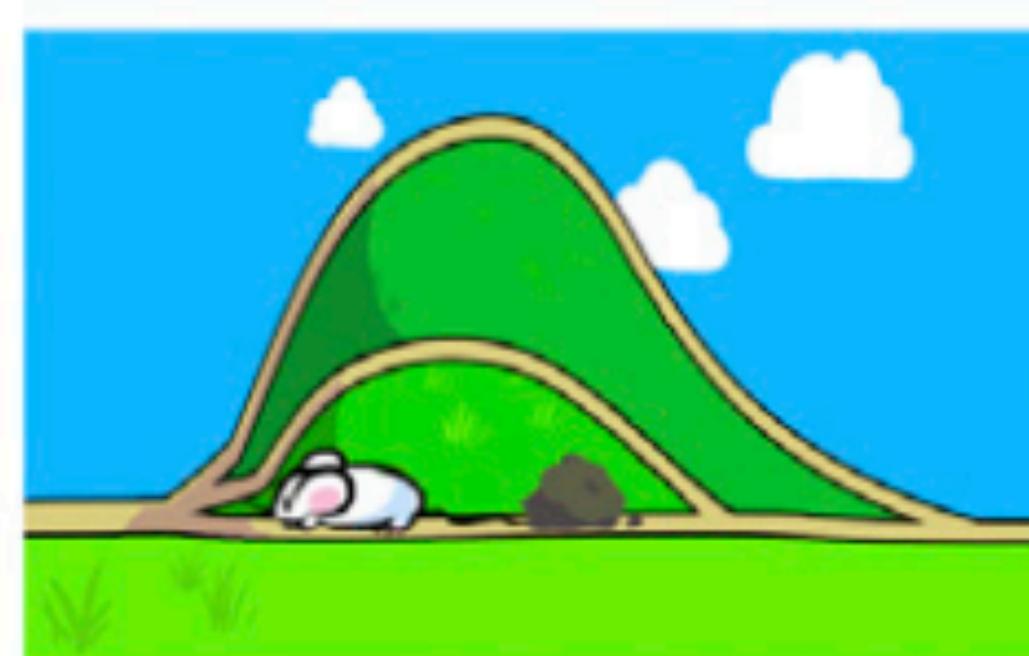
state



model-free

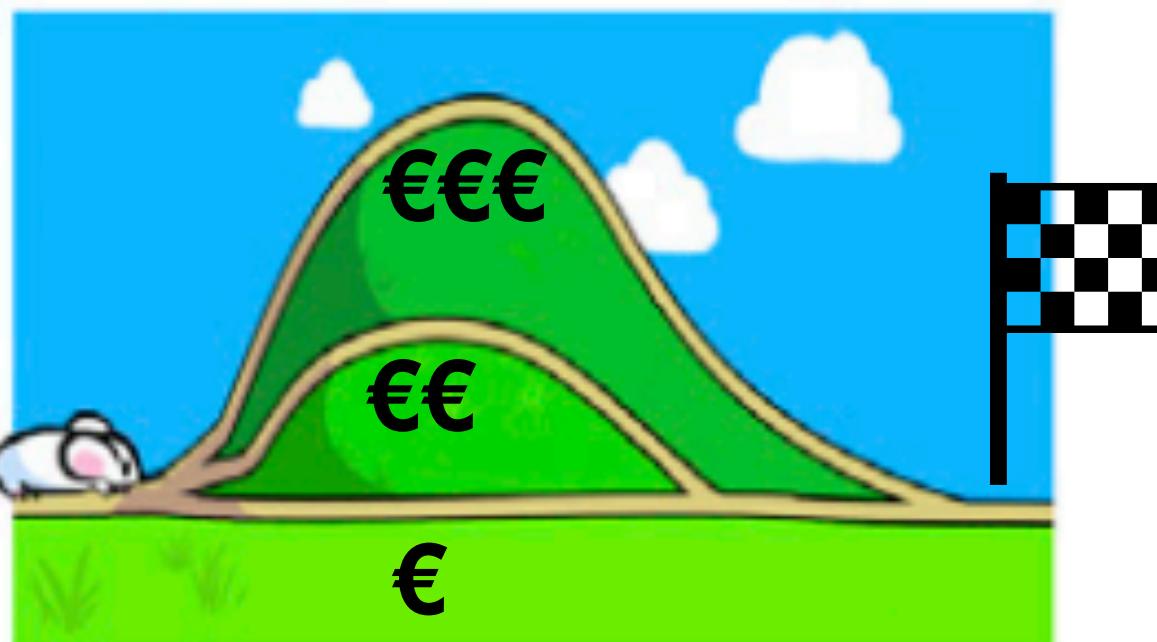


model-based



blocked normal

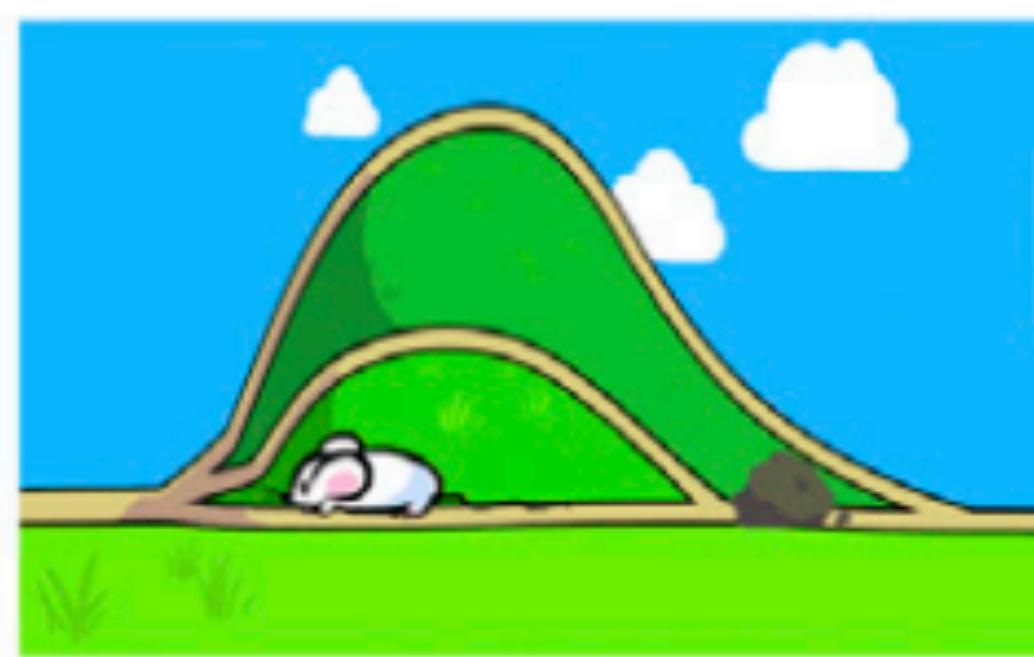
state



model-free

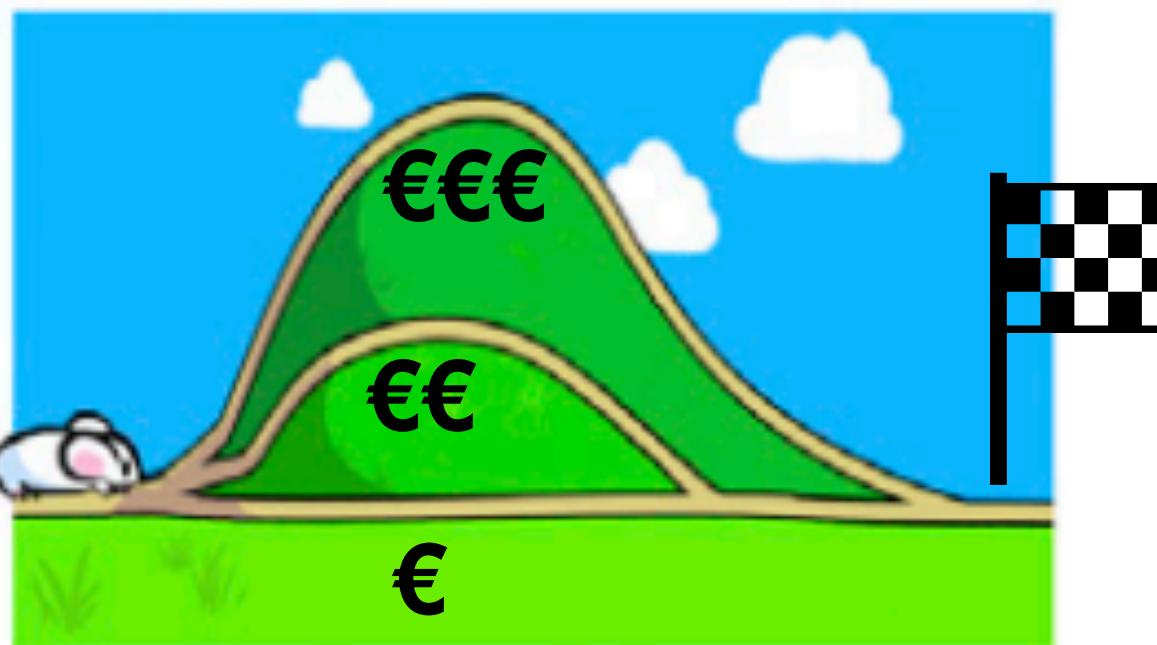


model-based



blocked normal

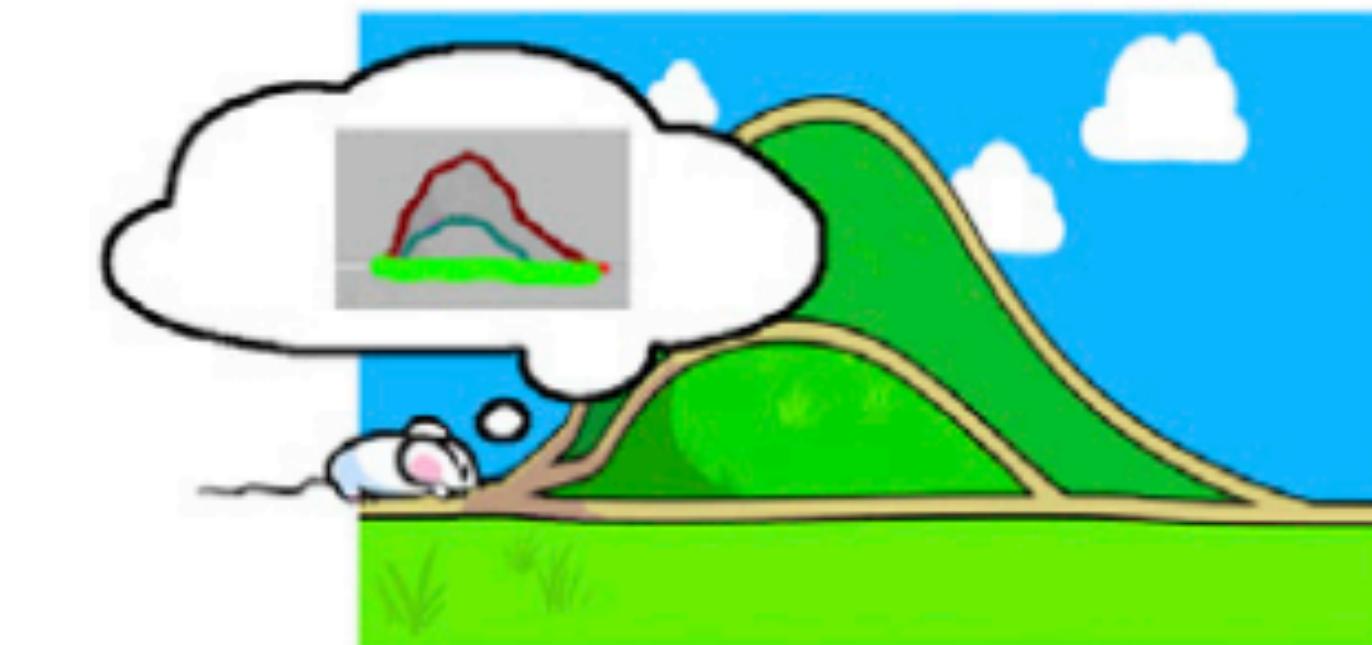
state



model-free

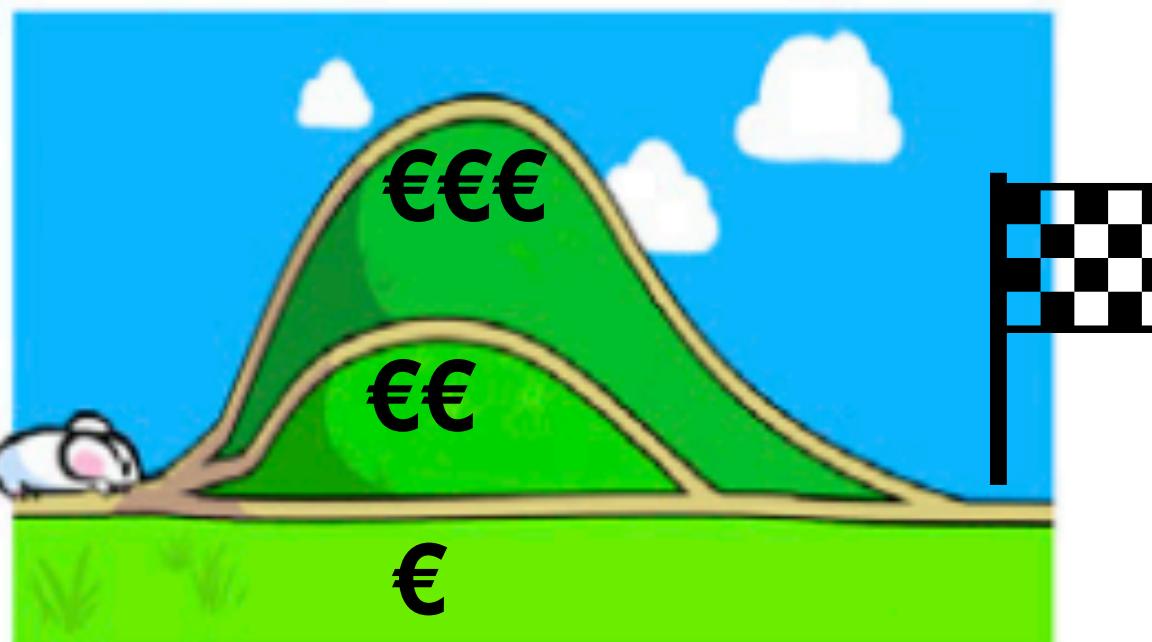


model-based



blocked normal

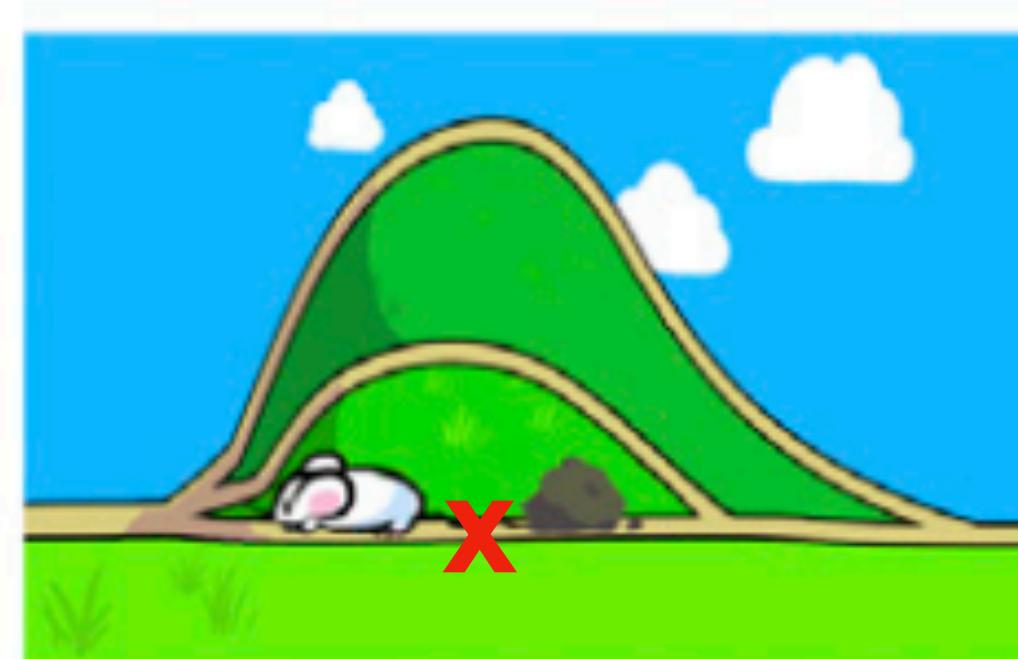
state



model-free

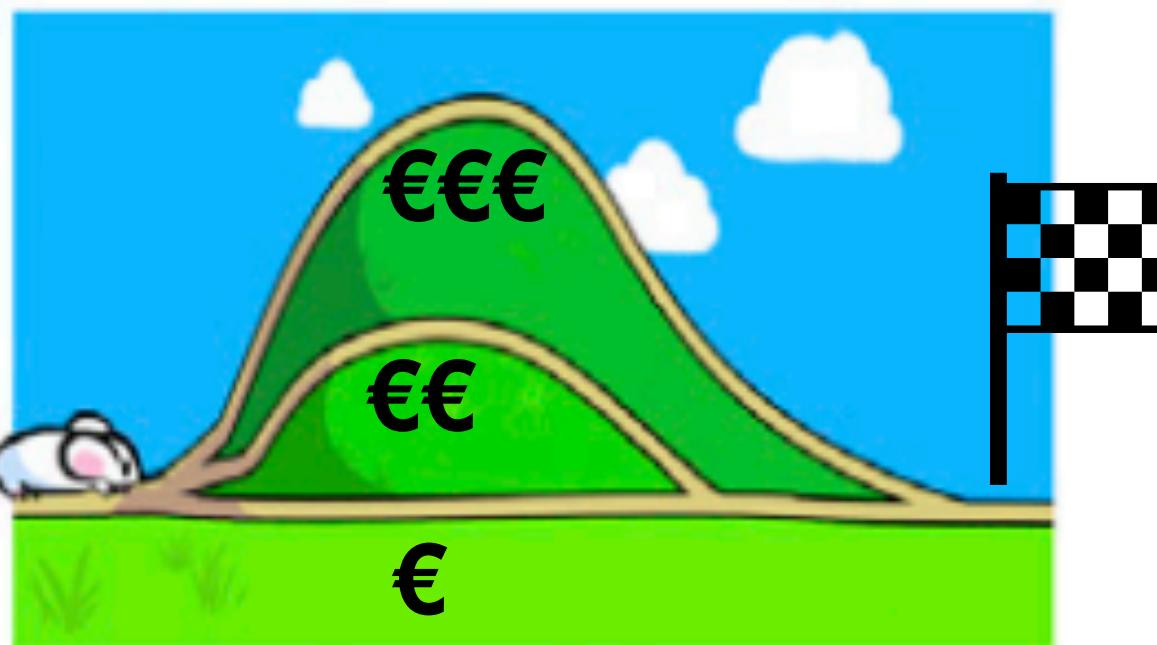


model-based



blocked normal

state



model-free



model-based

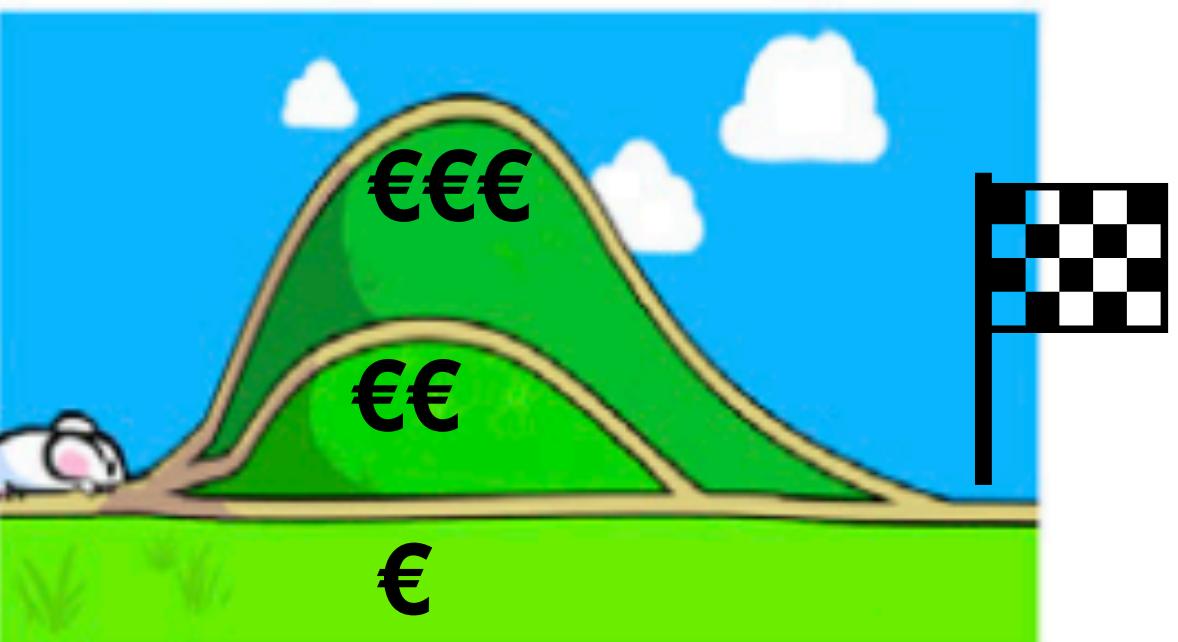


↑ no difference



blocked normal

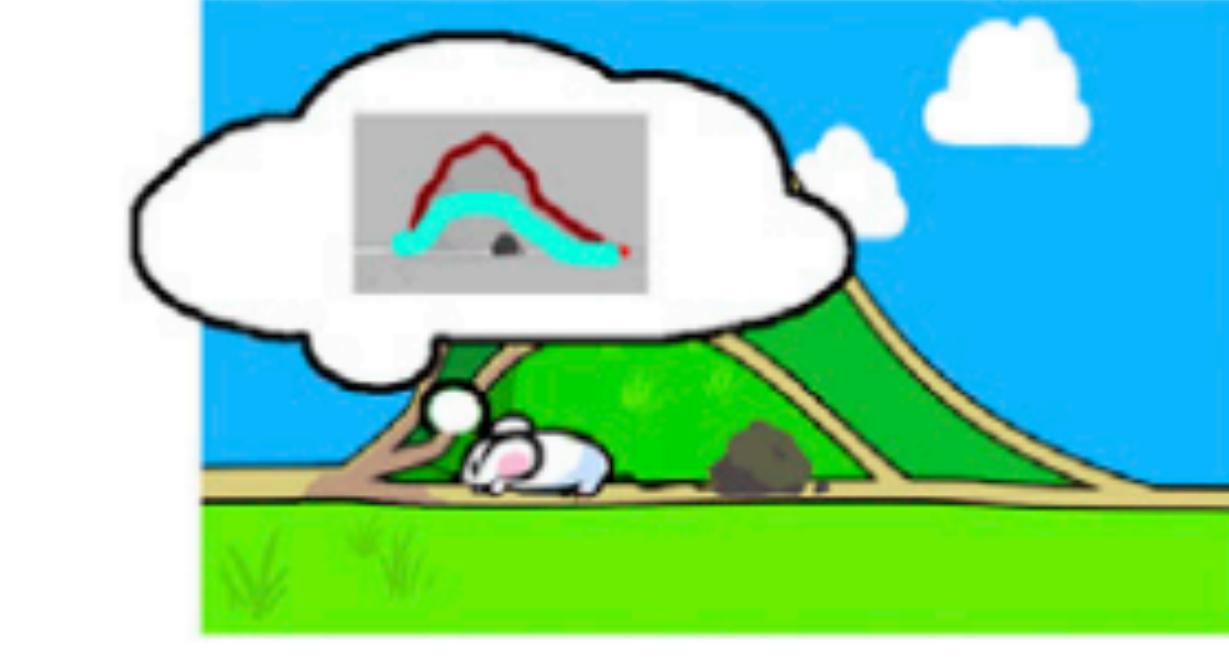
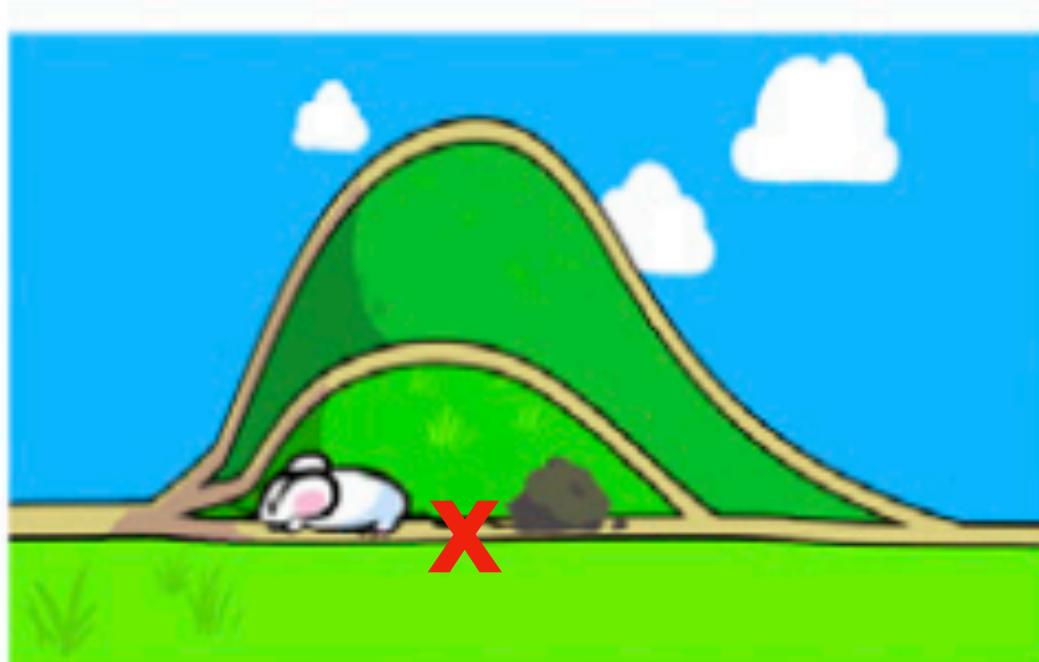
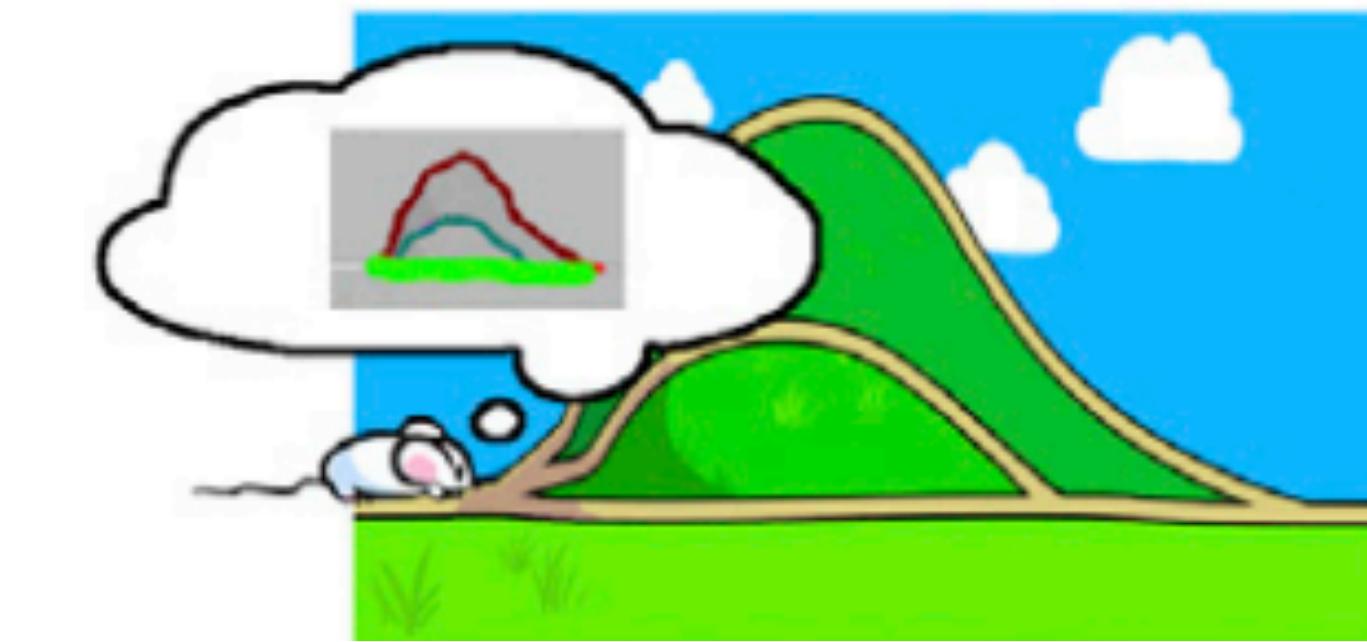
state



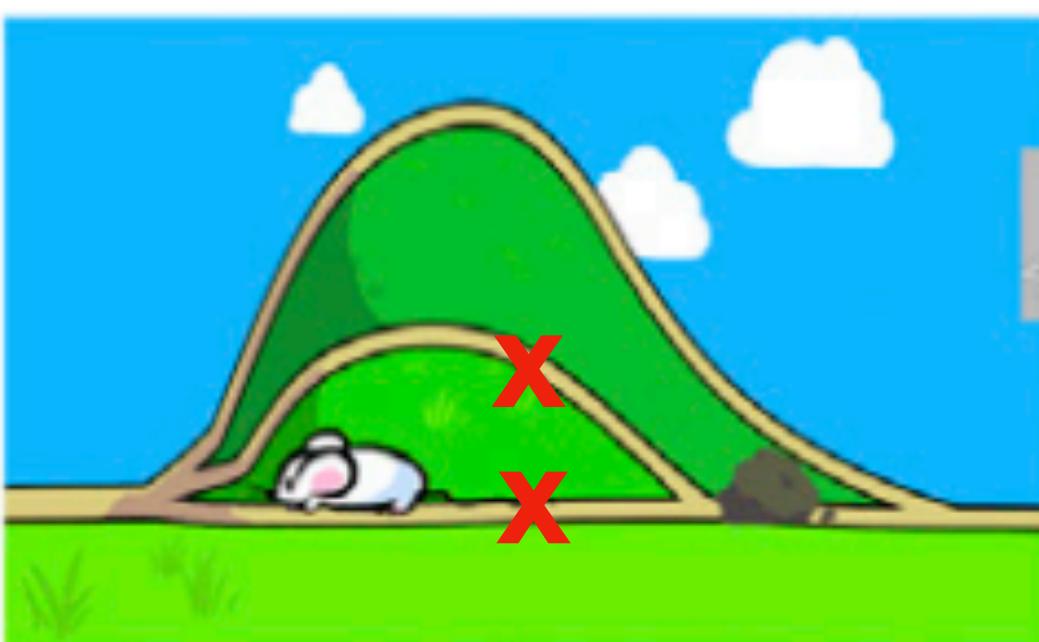
model-free



model-based

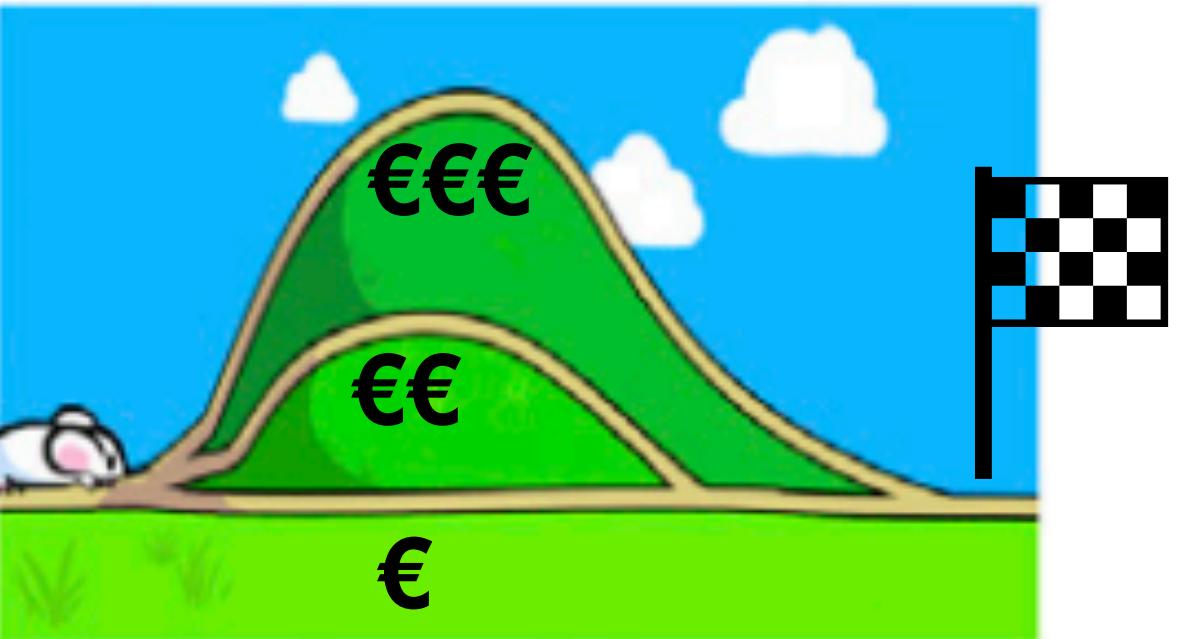


↑ no difference



blocked normal

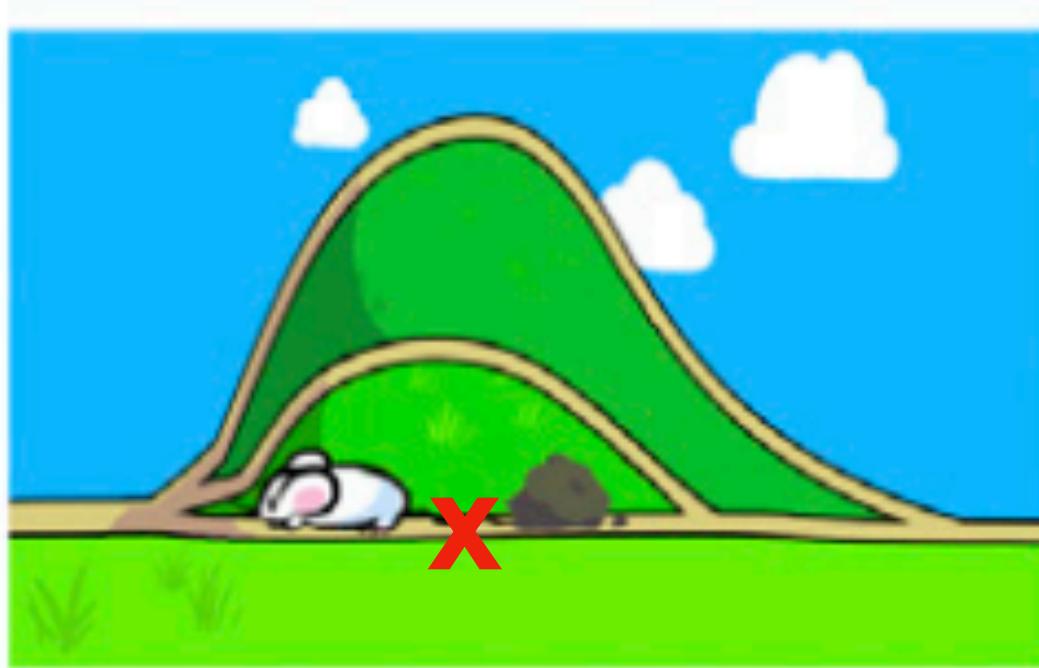
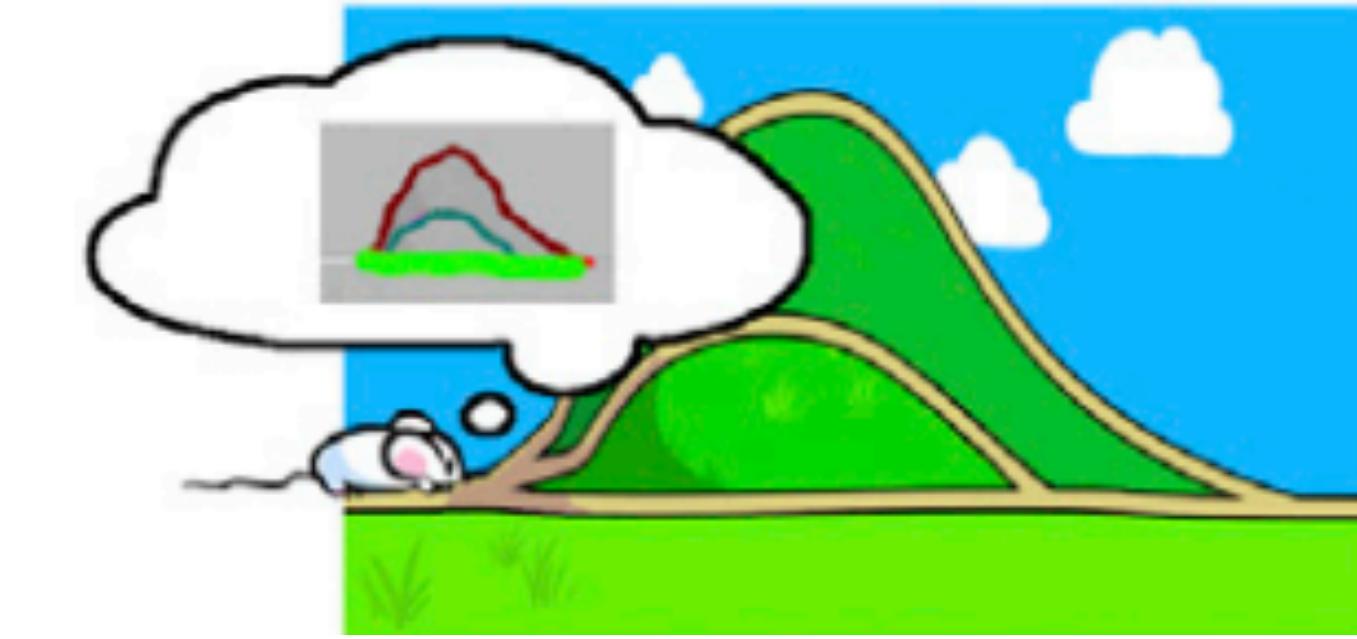
state



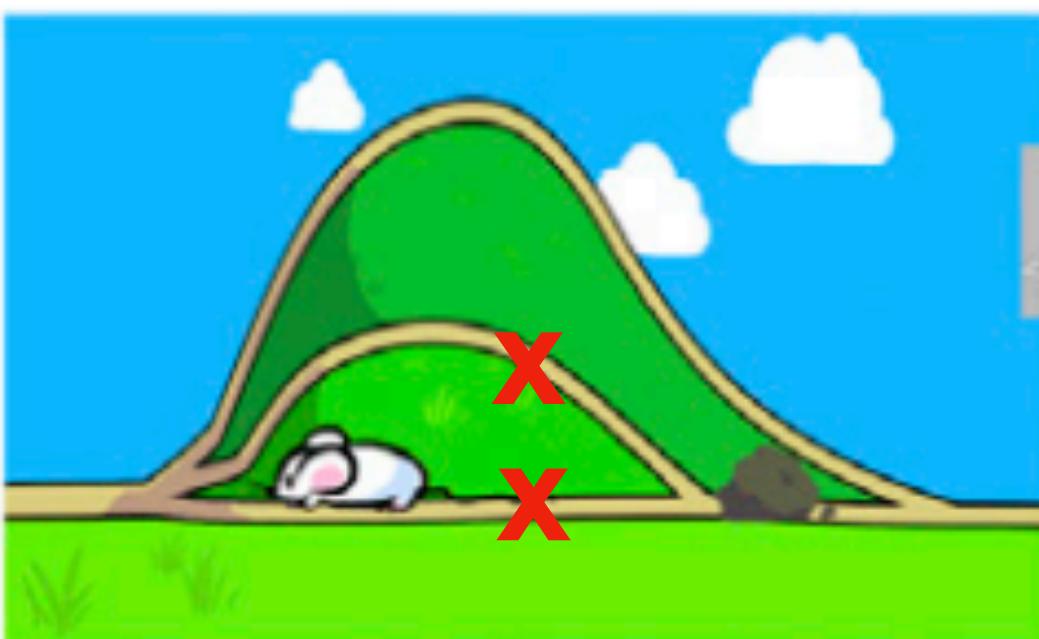
model-free



model-based



↔ no difference

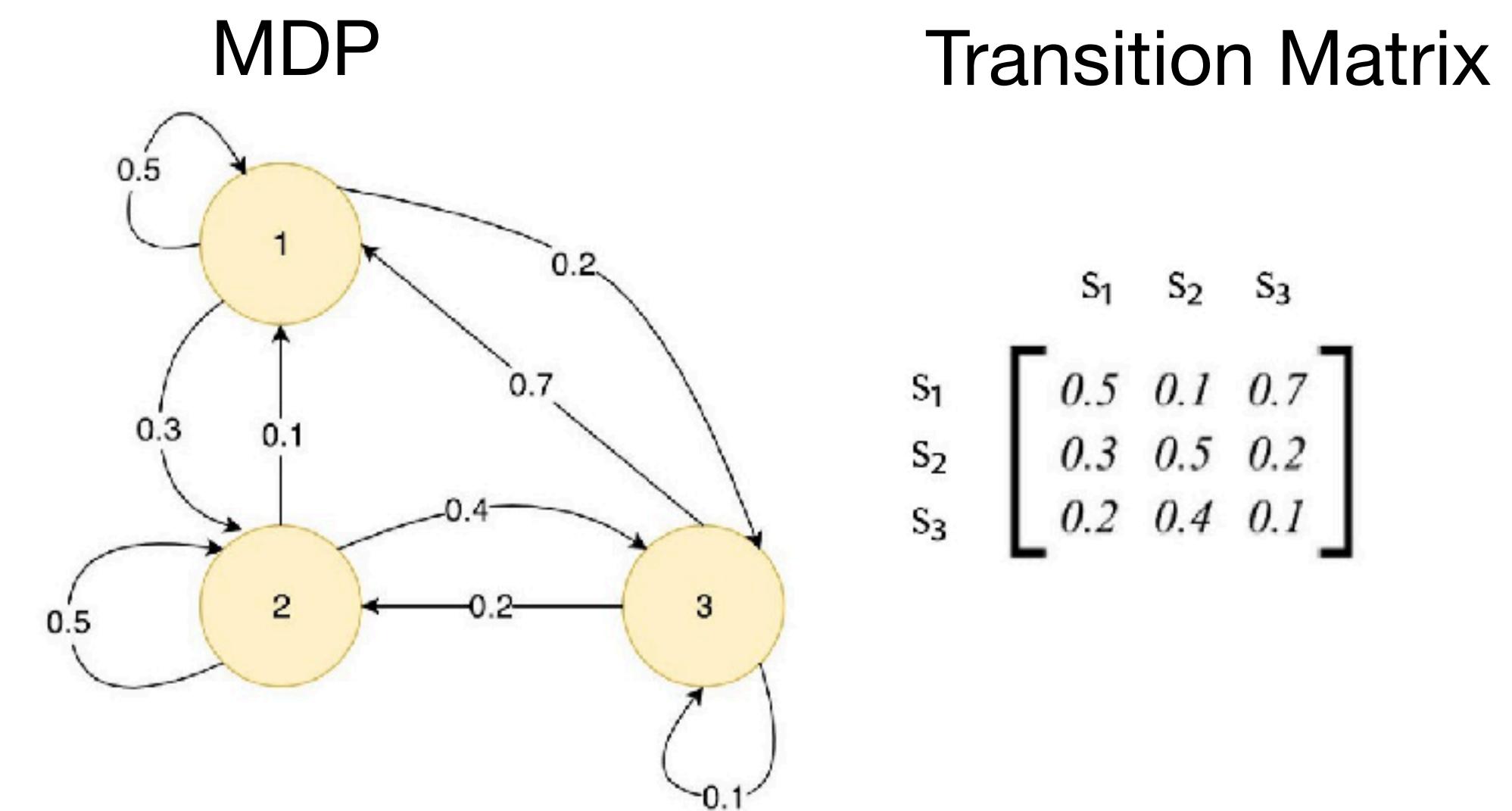
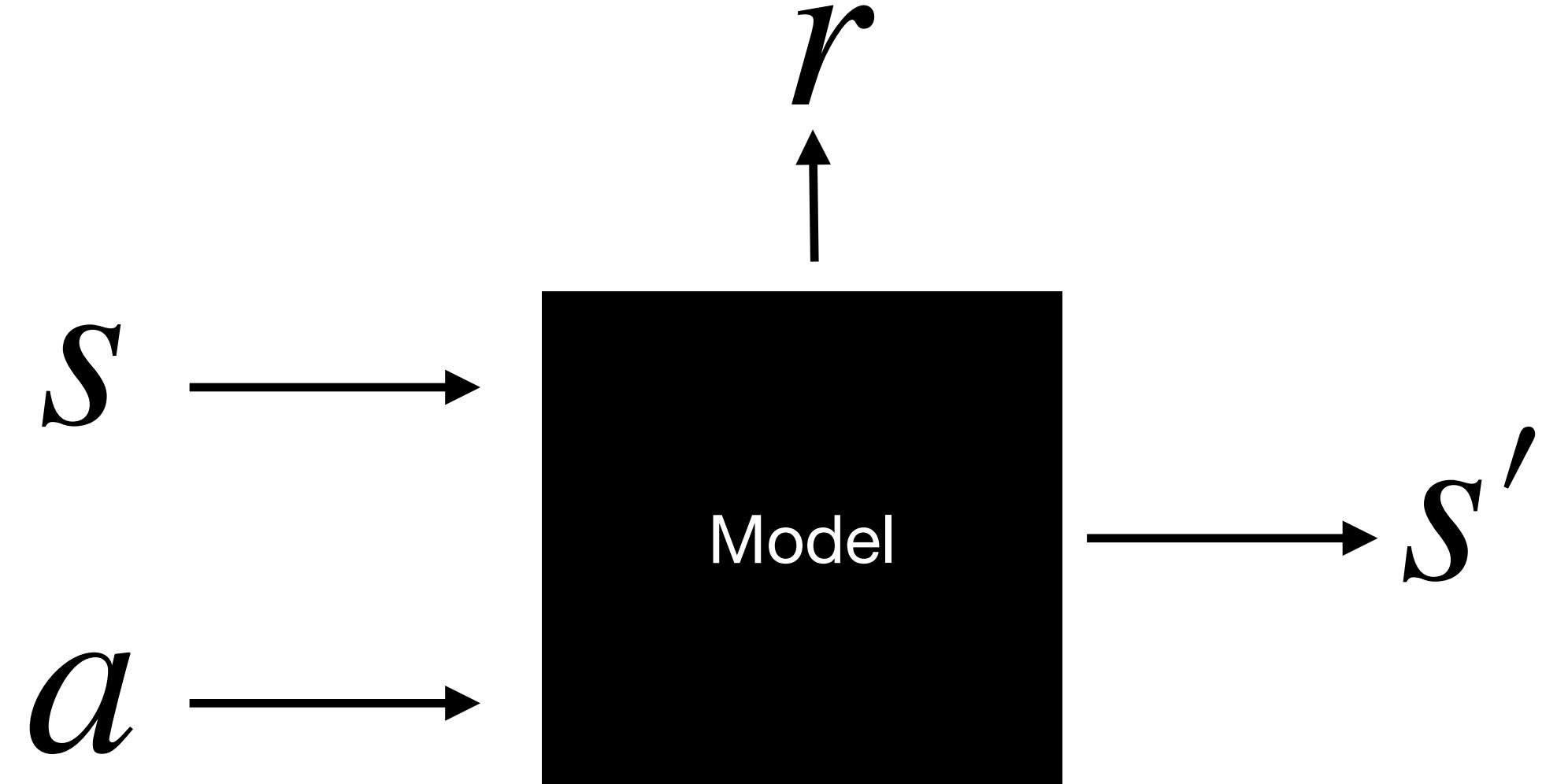


jumps to correct solution



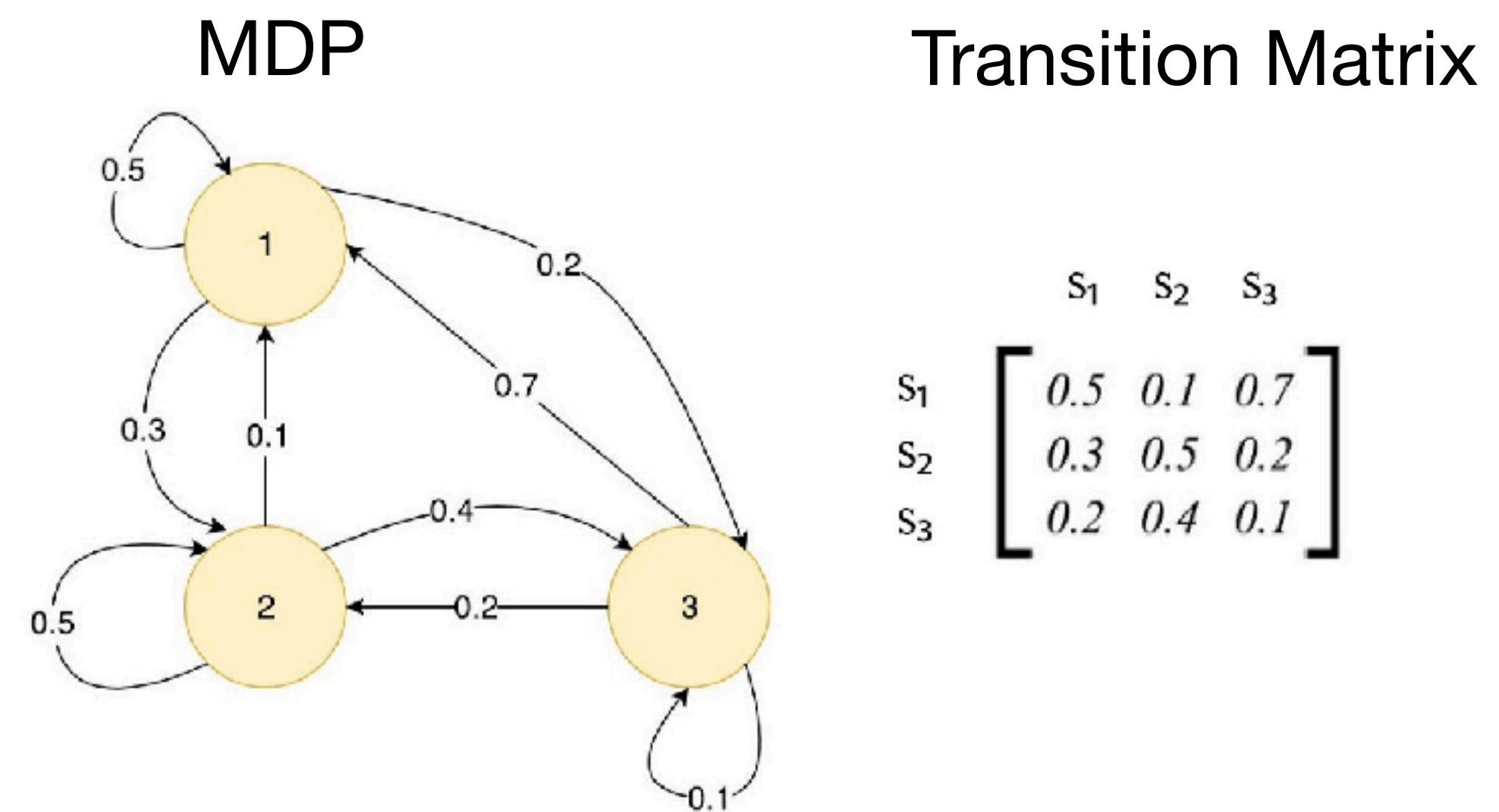
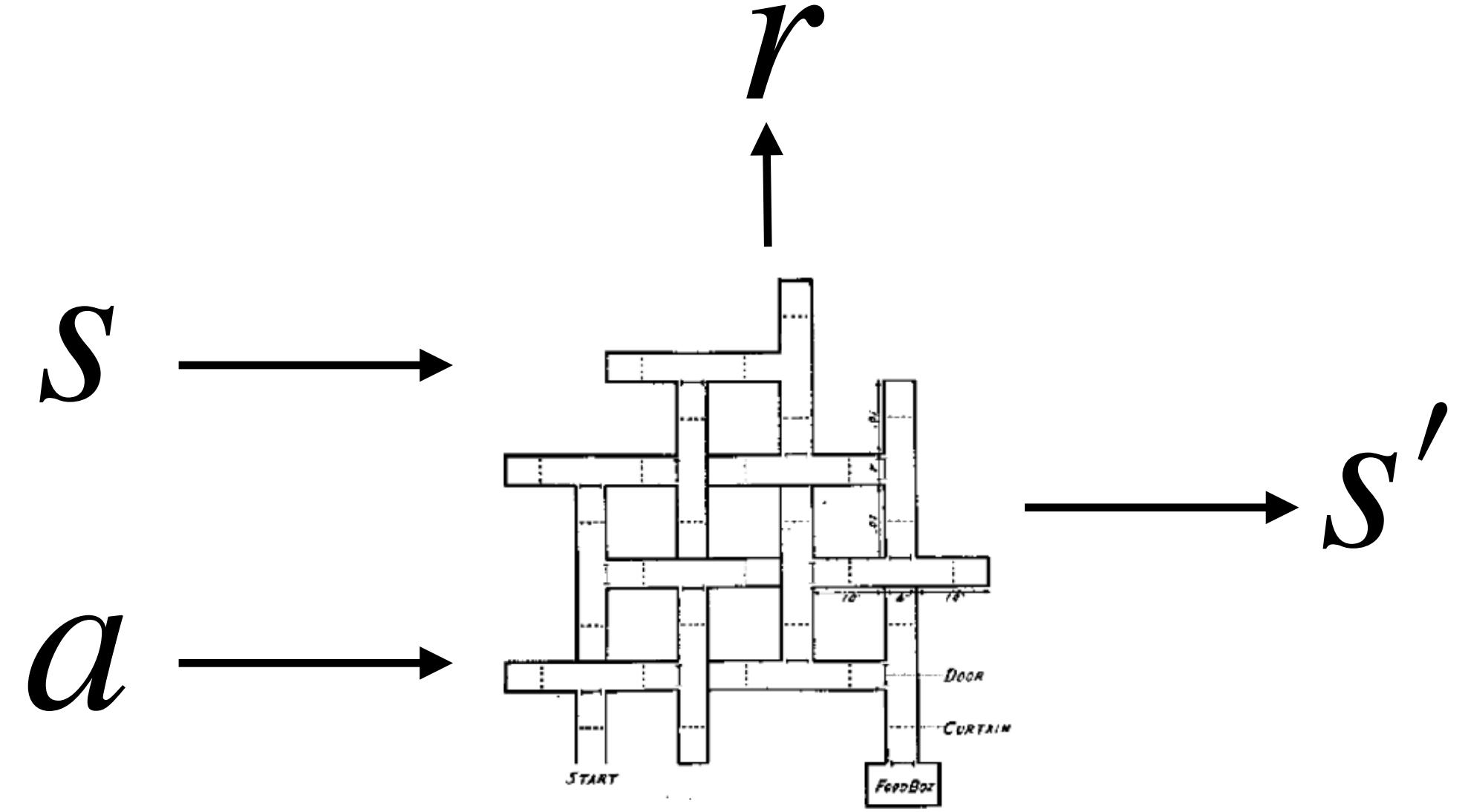
What is model-based RL?

- An internal representation of the environment
- Ingredients:
 - Transition matrix $T(s' | s, a)$
 - Reward function $R(s, a)$
 - State space $s \in \mathcal{S}$
 - Action space $a \in \mathcal{A}$
 - How is it learned? (find out next week!)



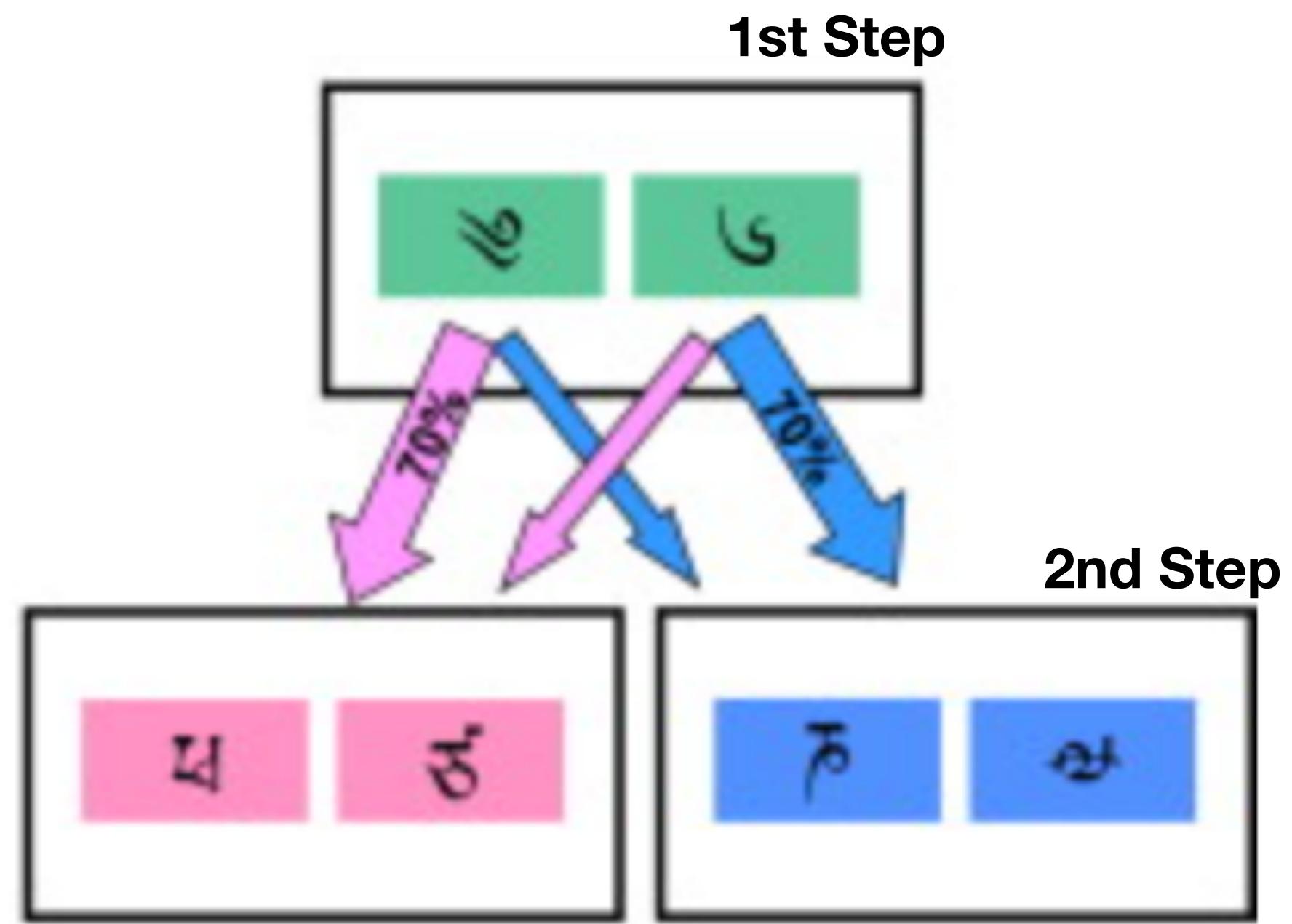
What is model-based RL?

- An internal representation of the environment
- Ingredients:
 - Transition matrix $T(s' | s, a)$
 - Reward function $R(s, a)$
 - State space $s \in \mathcal{S}$
 - Action space $a \in \mathcal{A}$
 - How is it learned? (find out next week!)



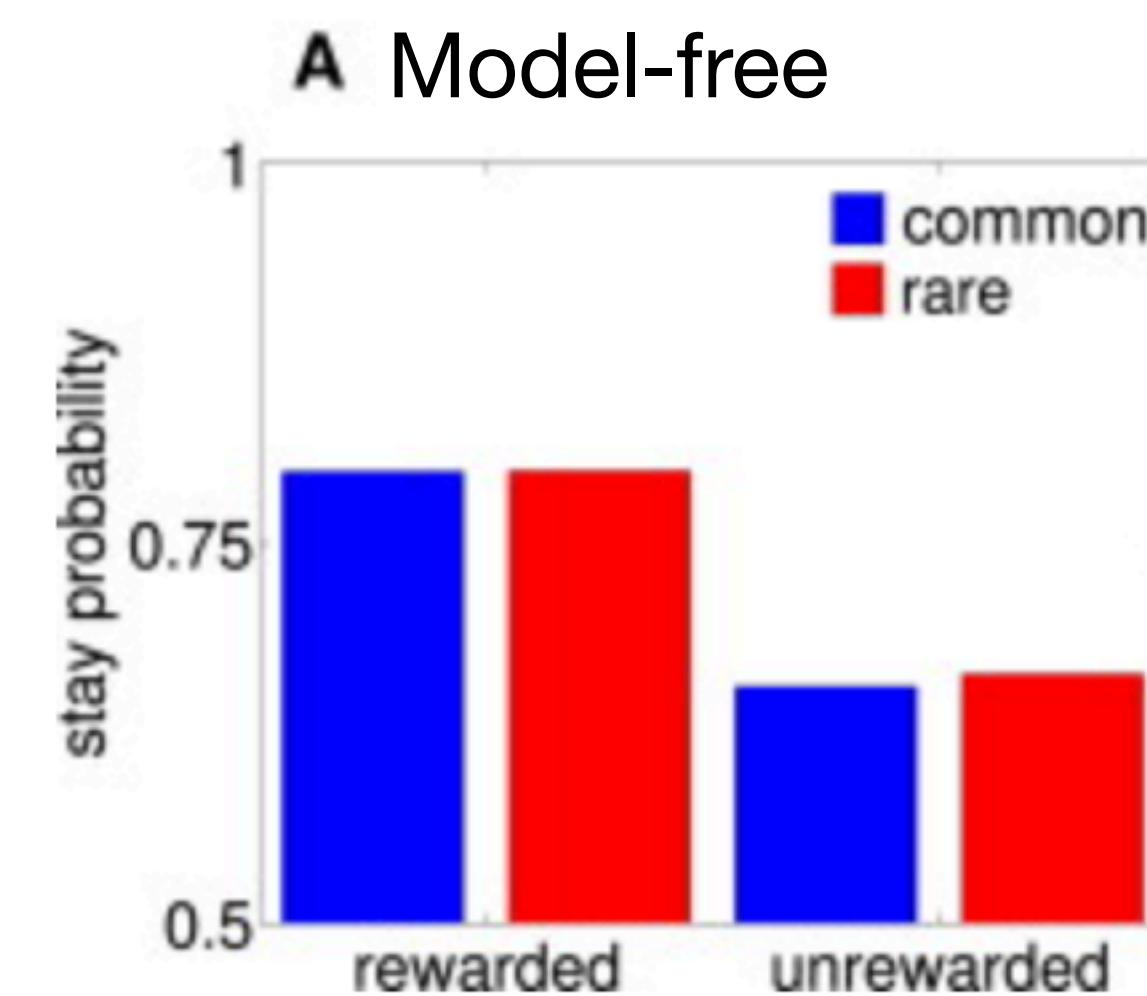
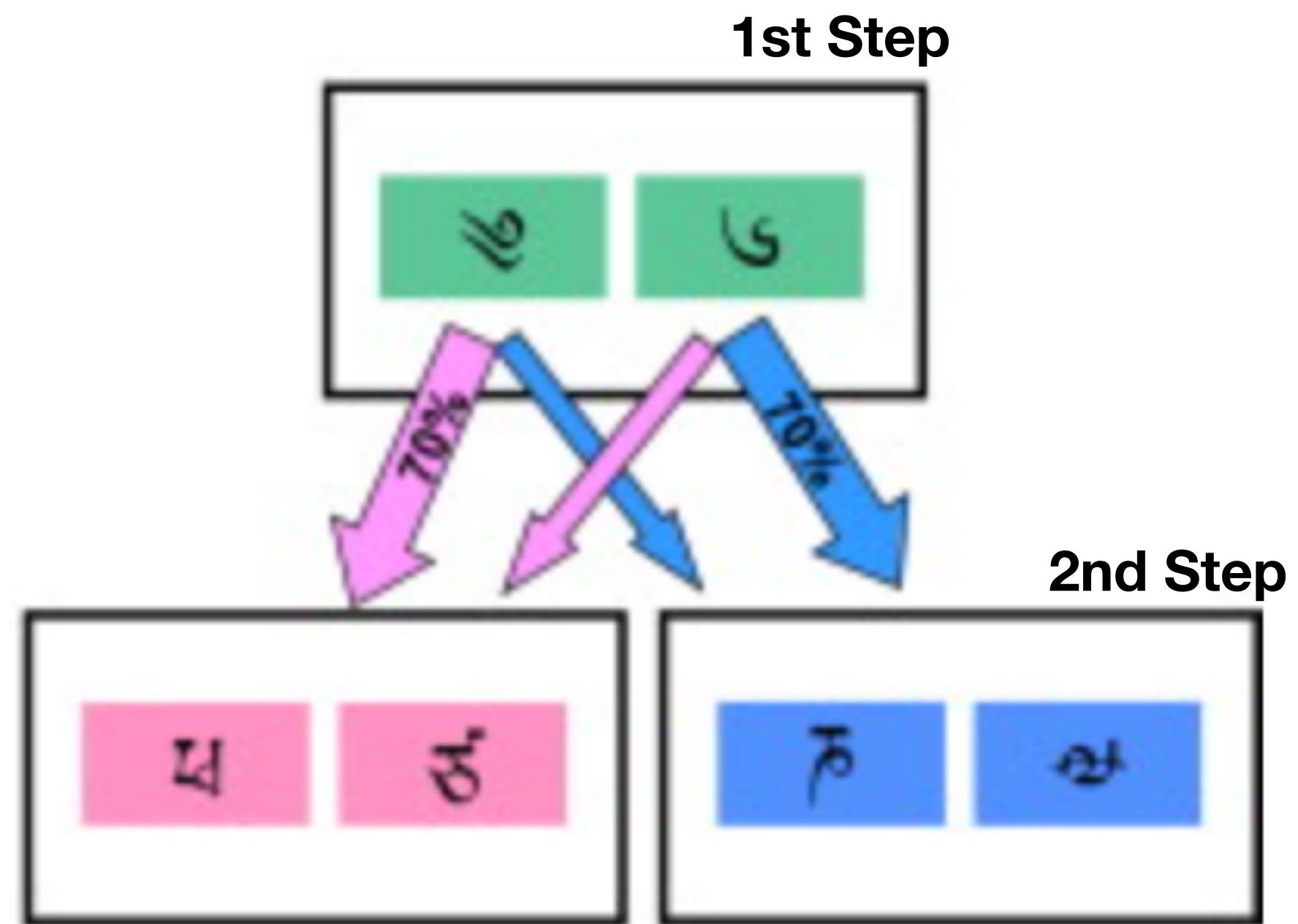
Two-step task

- Two-stage decision-making task used to distinguish model-free vs. model-based learning
- 1st step choices have common (70%) and rare (30%) transitions to different sets of 2nd step options
- 2nd step options have different P(reward)



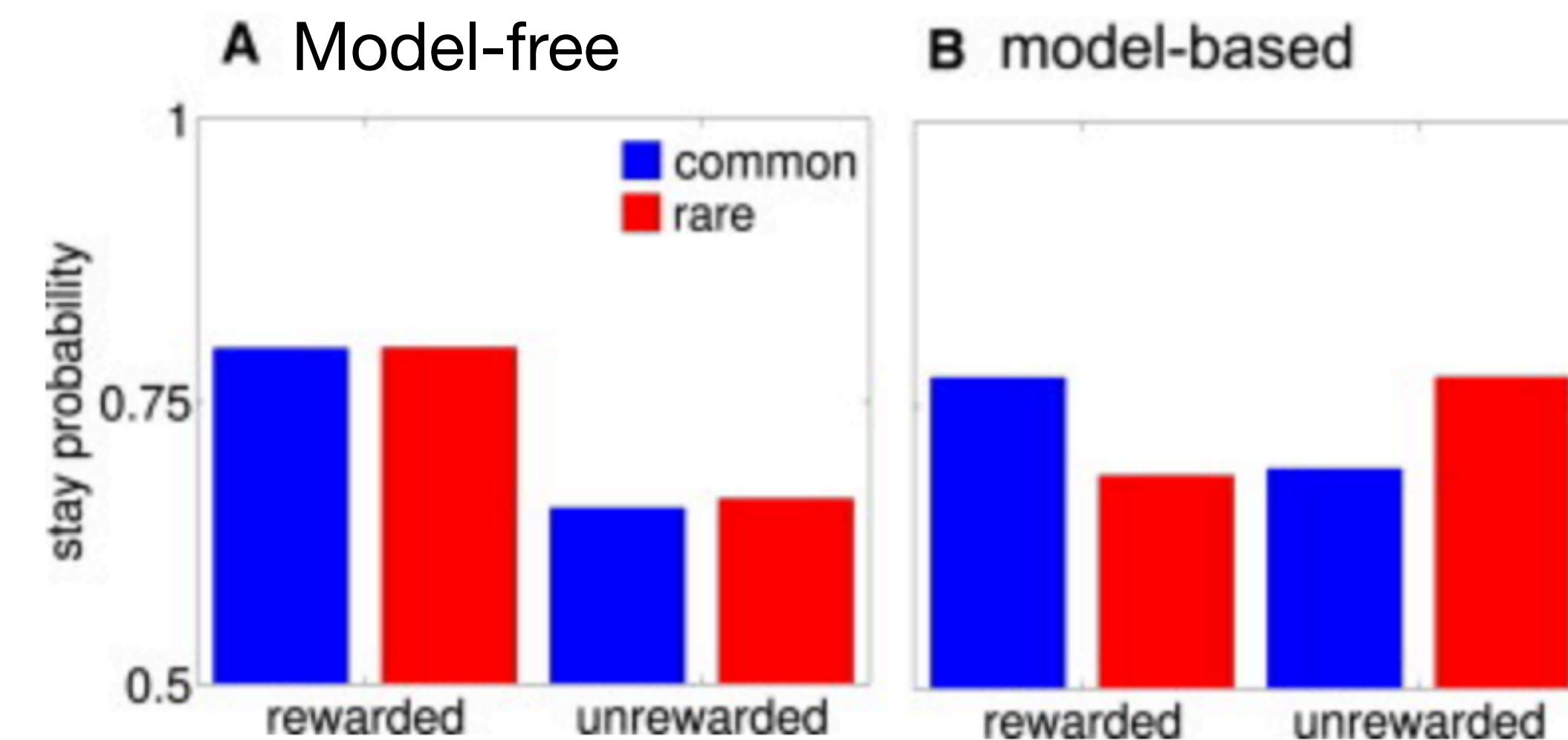
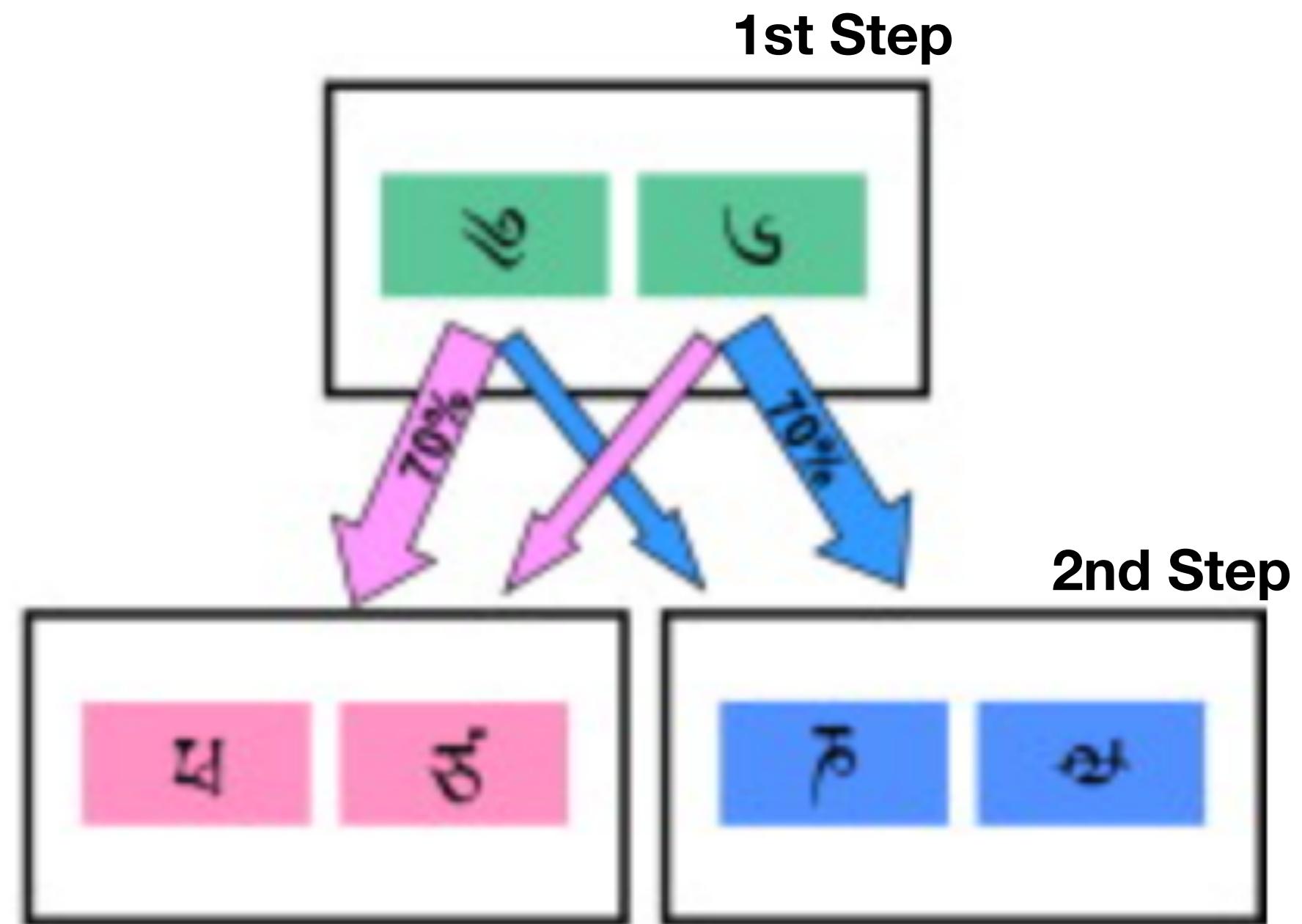
Two-step task

- Two-stage decision-making task used to distinguish model-free vs. model-based learning
- 1st step choices have common (70%) and rare (30%) transitions to different sets of 2nd step options
- 2nd step options have different P(reward)
- Model-free predictions depend solely on reward



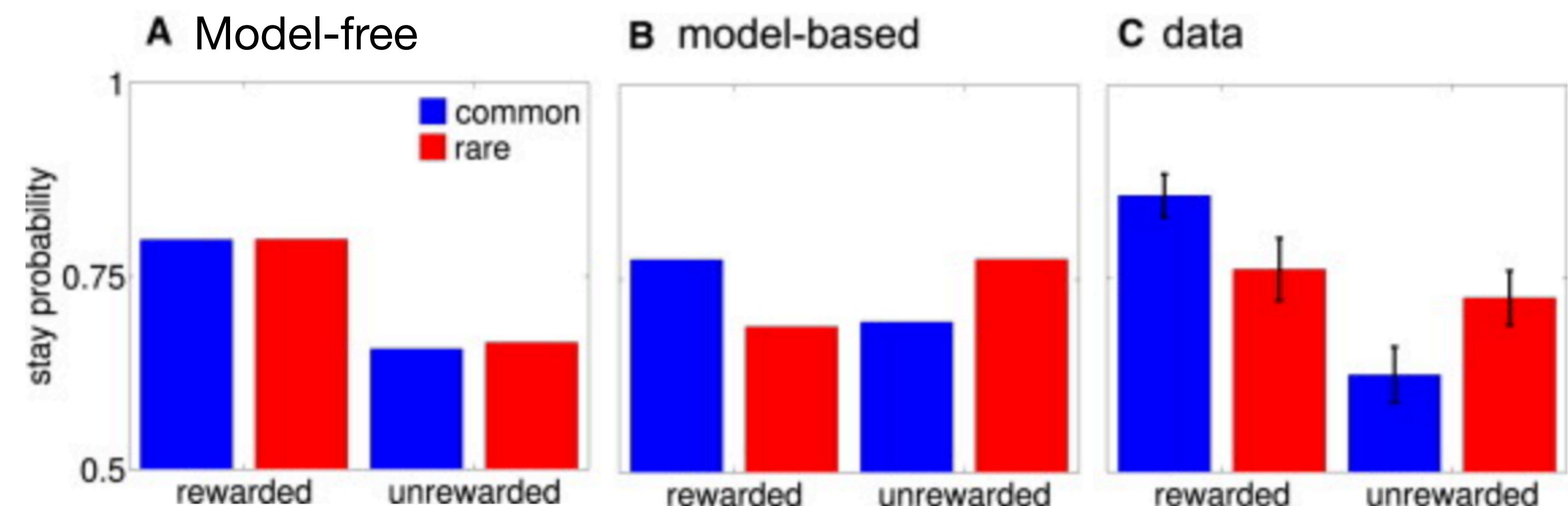
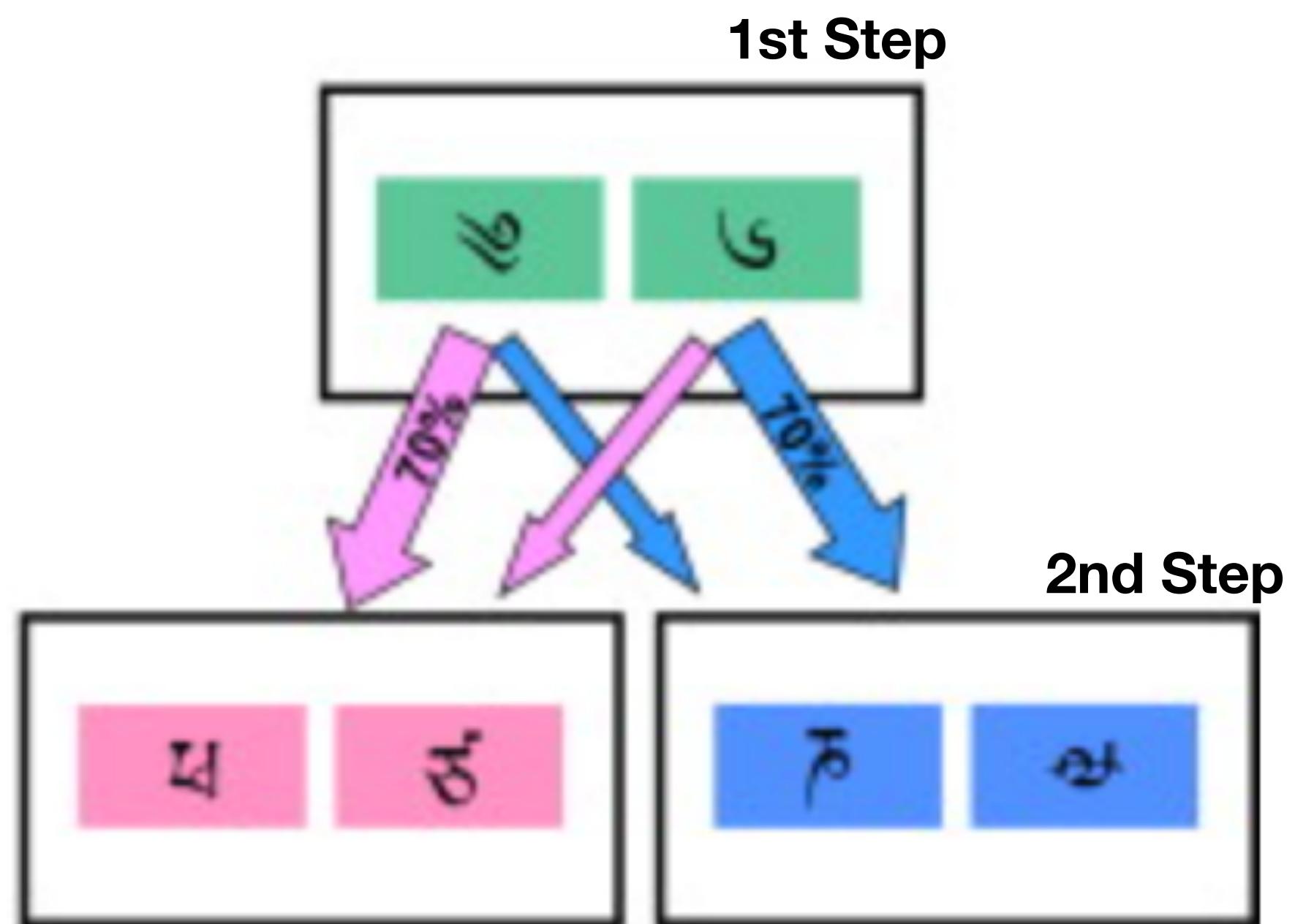
Two-step task

- Two-stage decision-making task used to distinguish model-free vs. model-based learning
- 1st step choices have common (70%) and rare (30%) transitions to different sets of 2nd step options
- 2nd step options have different P(reward)
- Model-free predictions depend solely on reward
- Model-based RL predicts different responses depending on common vs. rare



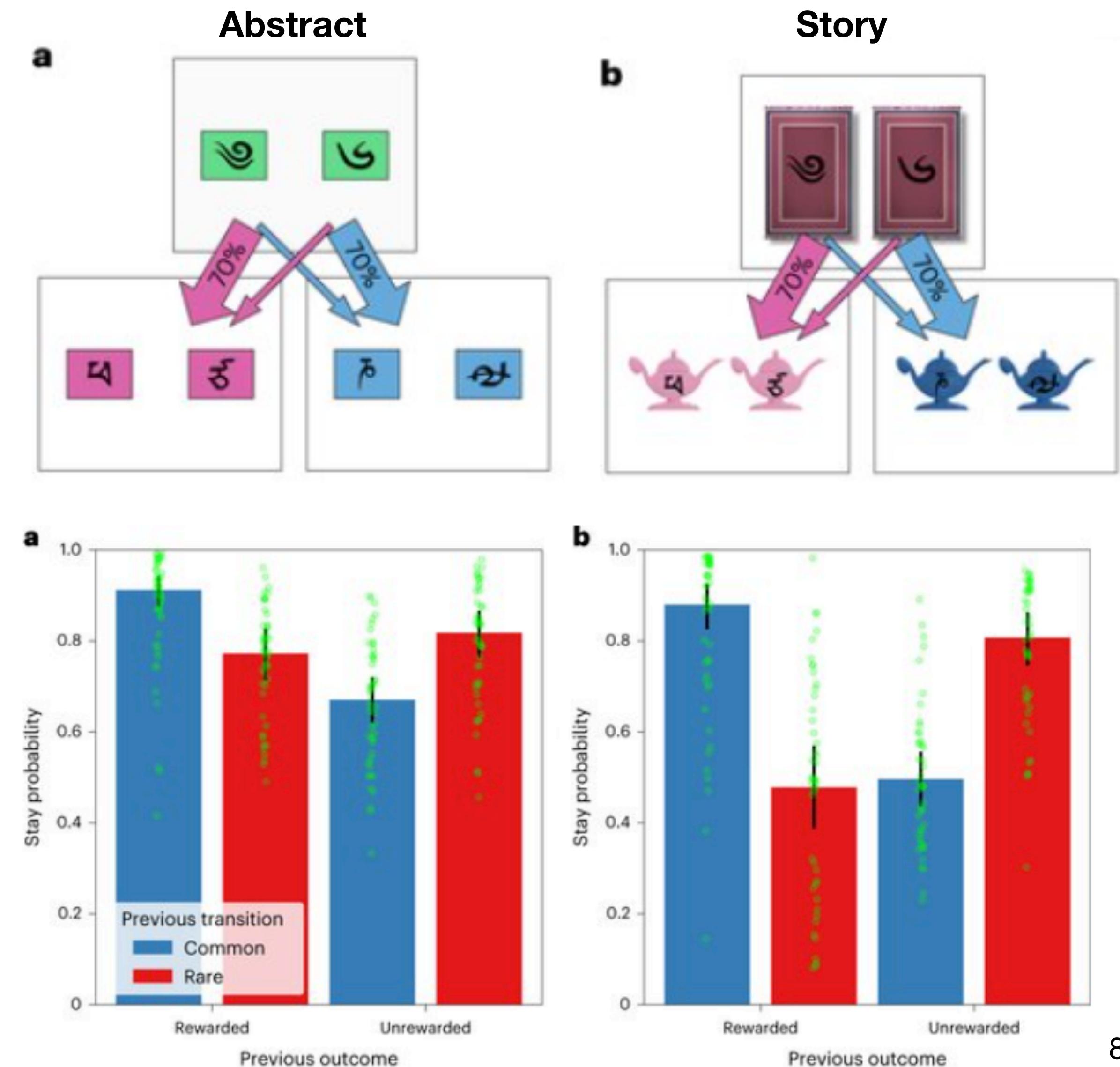
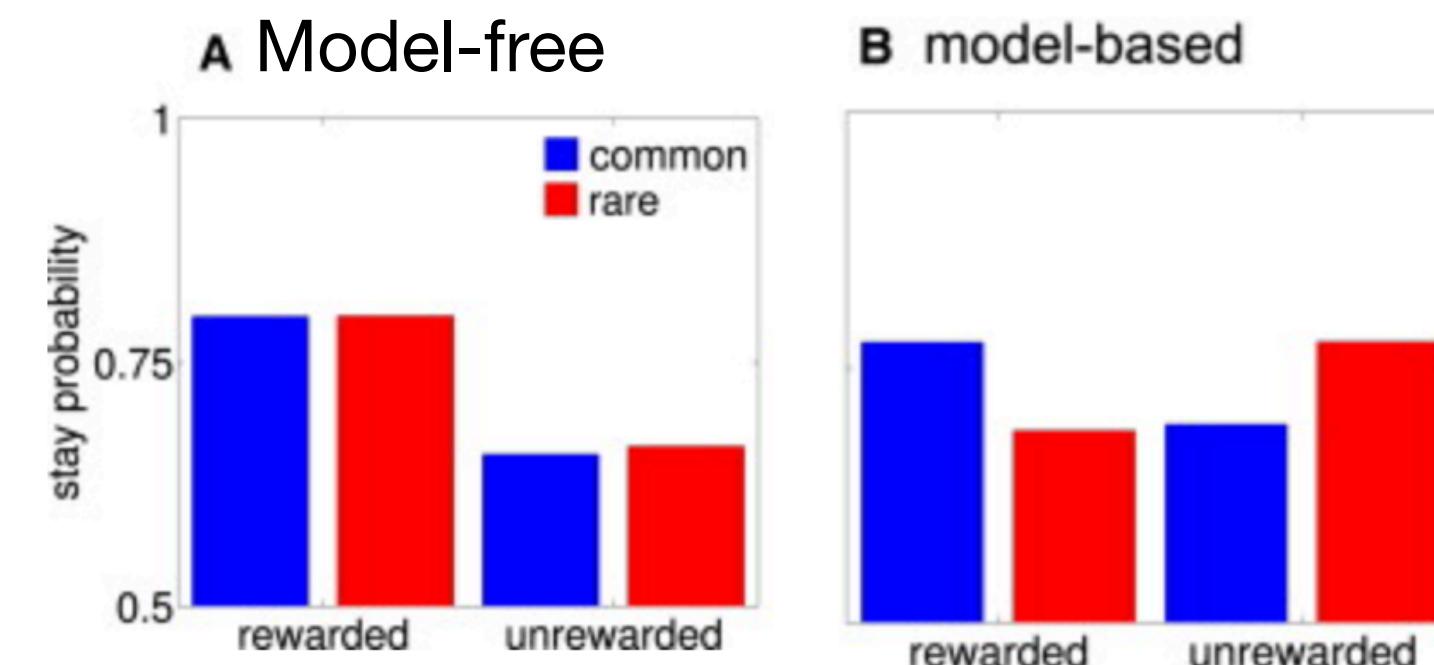
Two-step task

- Two-stage decision-making task used to distinguish model-free vs. model-based learning
- 1st step choices have common (70%) and rare (30%) transitions to different sets of 2nd step options
- 2nd step options have different P(reward)
- Model-free predictions depend solely on reward
- Model-based RL predicts different responses depending on common vs. rare
- Data suggests a mixture of both



Two-step task (revisited)

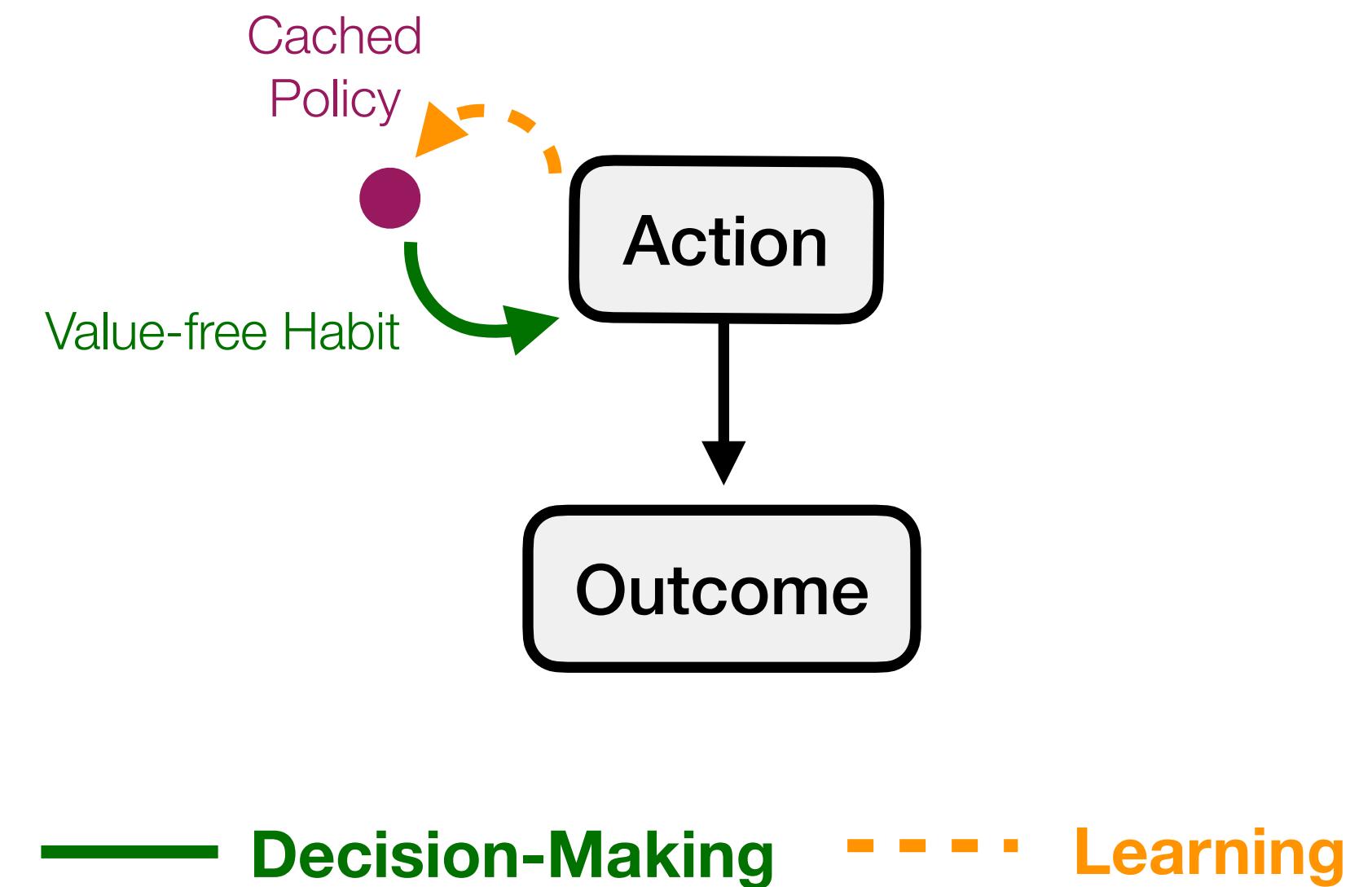
- More recent work suggests a different interpretation of that classic result from Daw et al., (2011)
- Abstract vs. story condition to manipulate how easily it is to understand the nature of transitions
- Story condition was almost perfectly aligned with model-based predictions



Two Pathways for Learning

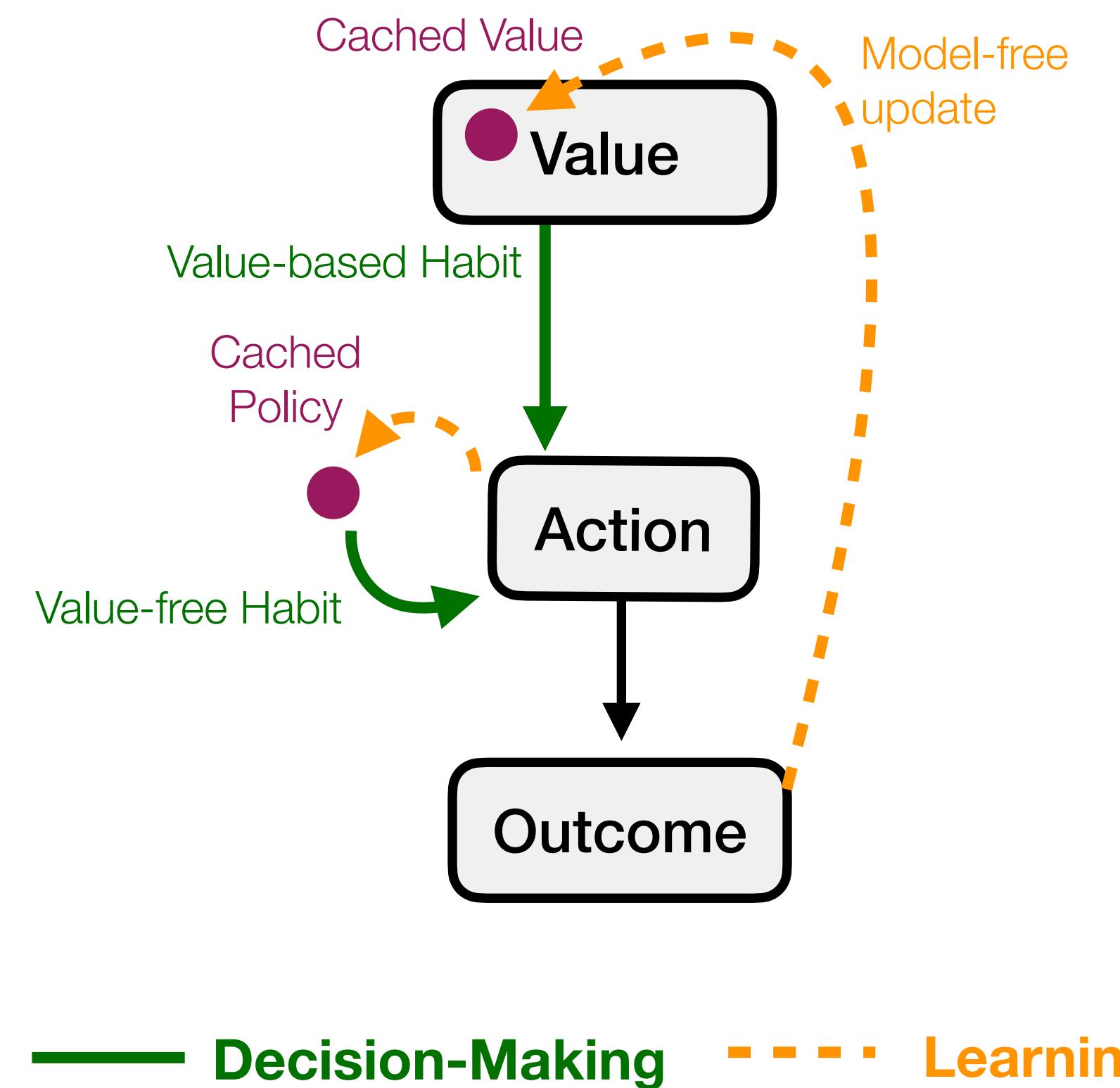
Two Pathways for Learning

- **Law of Exercise:** Repeat actions performed in the past by learning a **cached policy** (independent of reward)



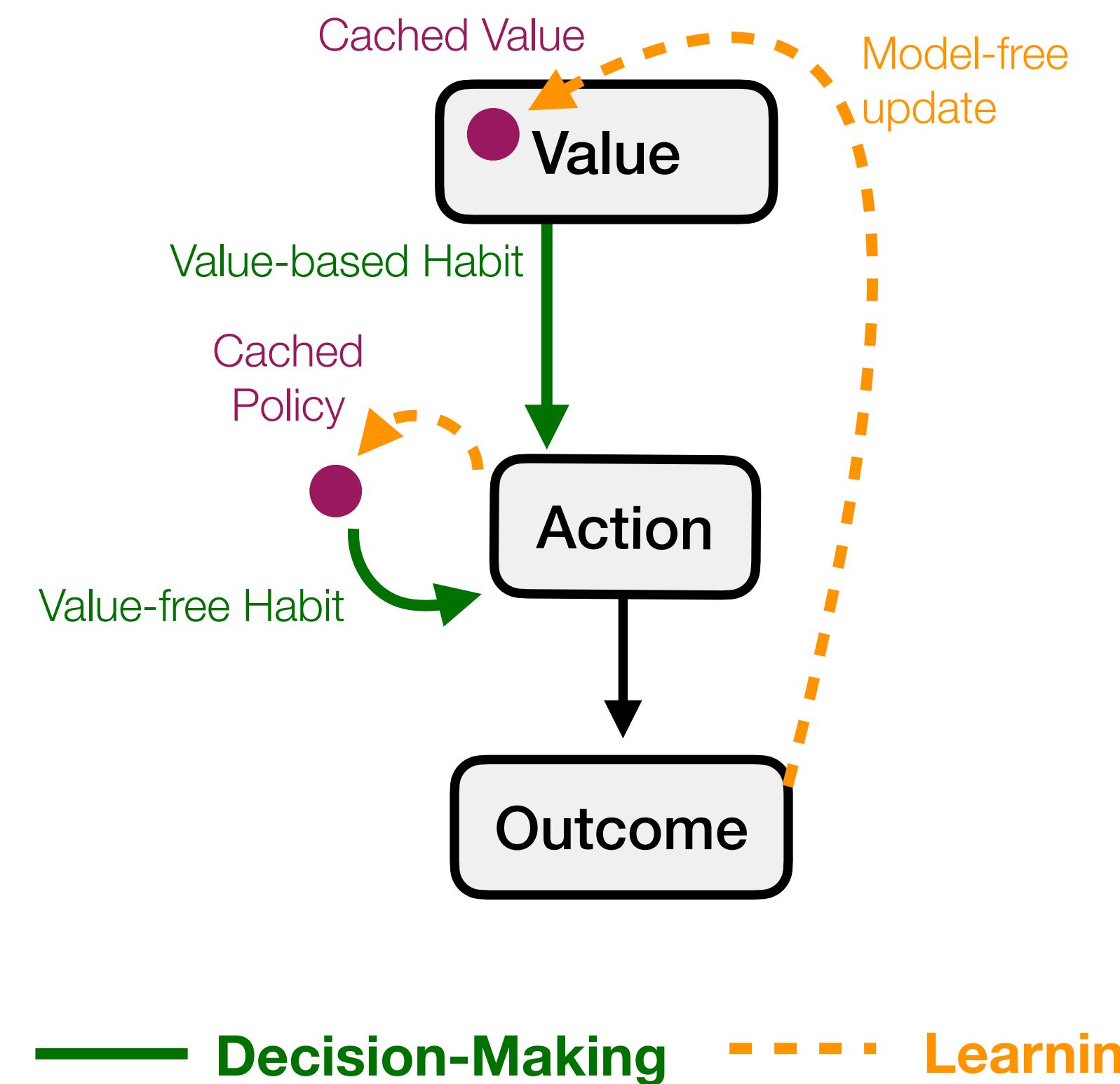
Two Pathways for Learning

- **Law of Exercise:** Repeat actions performed in the past by learning a **cached policy** (independent of reward)
- **Law of Effect:** Choose actions on the basis of what worked in the past by forming **cached value**



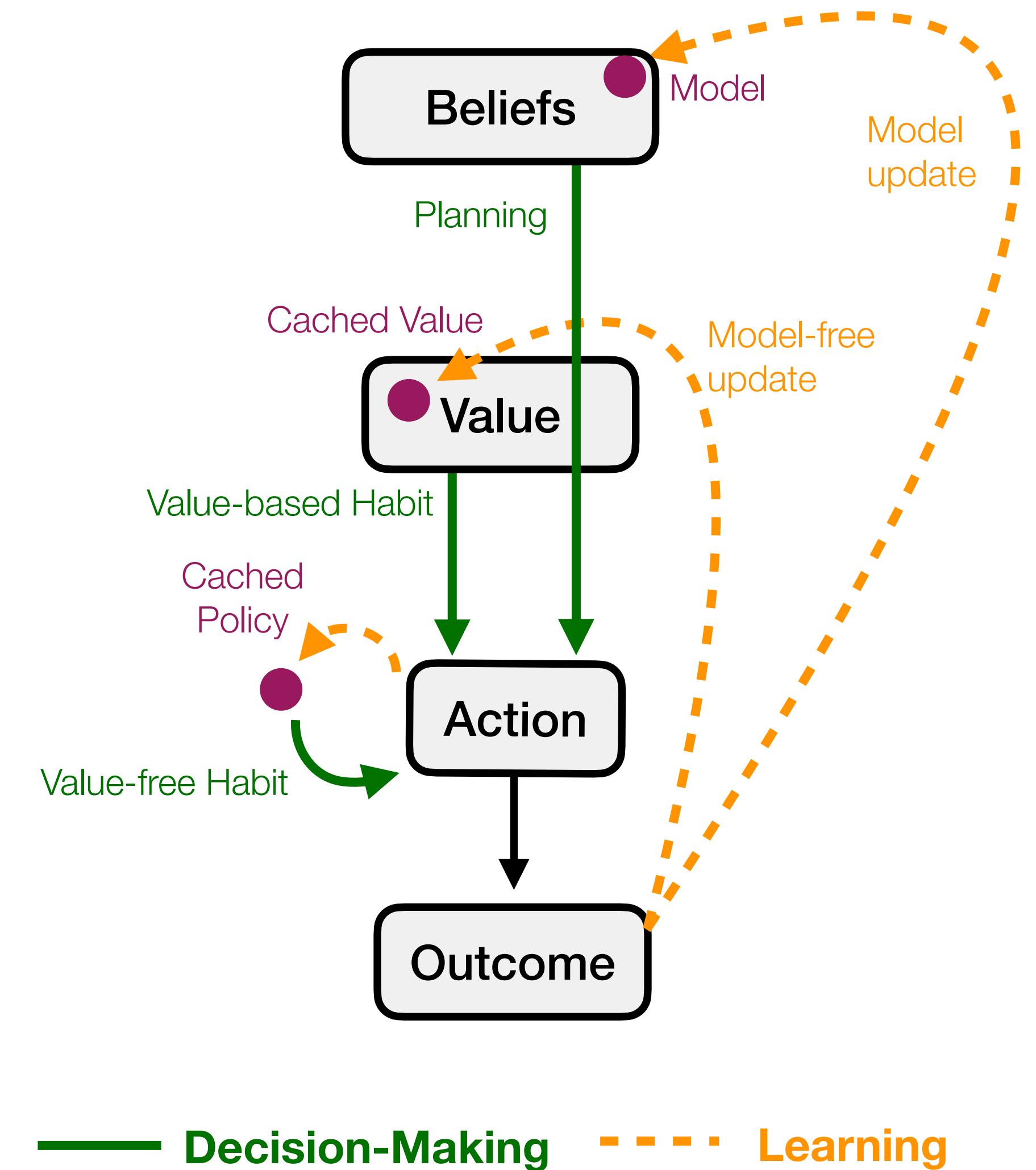
Two Pathways for Learning

- **Law of Exercise:** Repeat actions performed in the past by learning a **cached policy** (independent of reward)
- **Law of Effect:** Choose actions on the basis of what worked in the past by forming **cached value**
 - **Model-free RL**



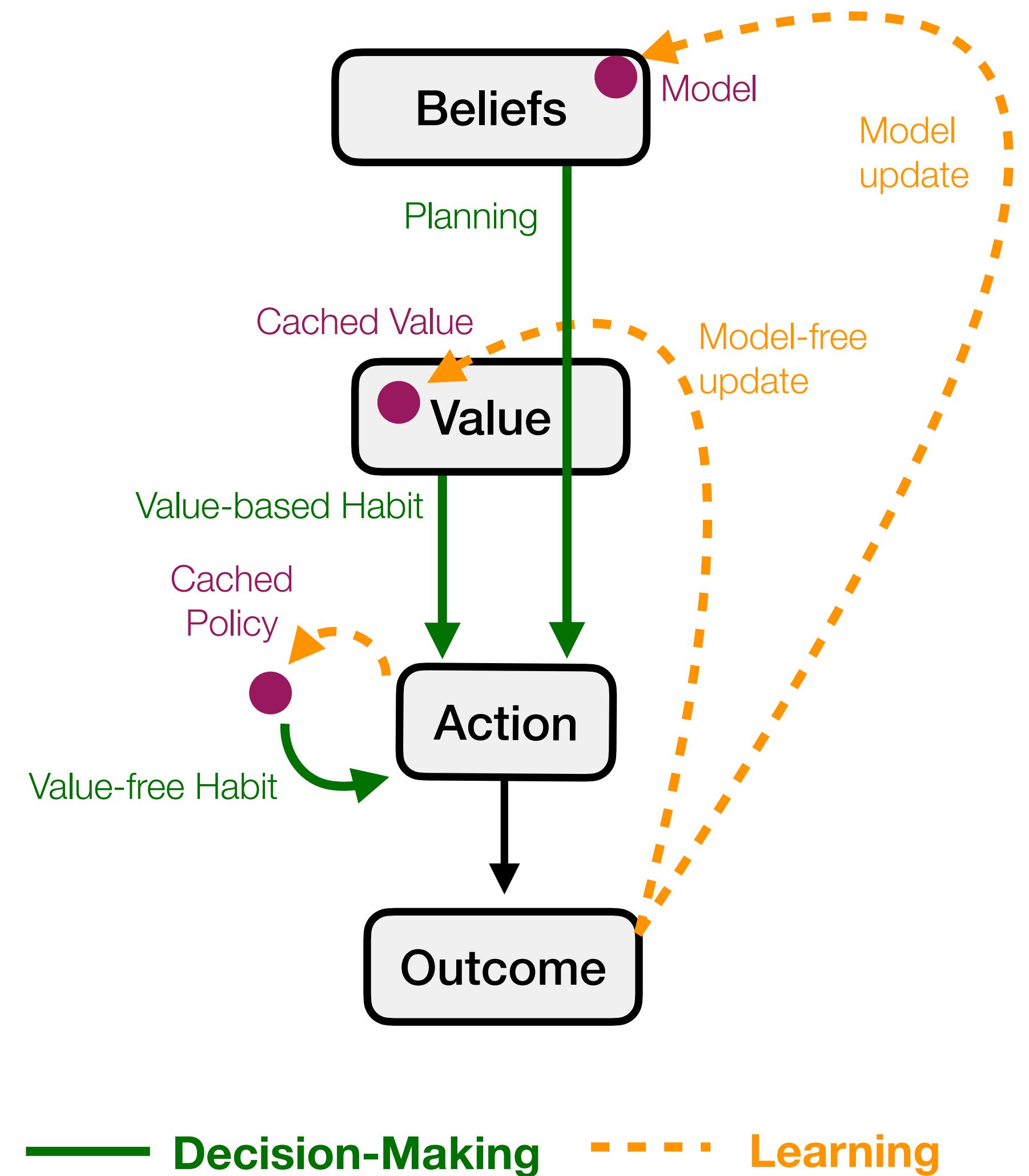
Three ~~Two~~ Pathways for Learning

- **Law of Exercise:** Repeat actions performed in the past by learning a **cached policy** (independent of reward)
- **Law of Effect:** Choose actions on the basis of what worked in the past by forming **cached value**
 - **Model-free RL**
- **Model-based planning:** Select actions expected to produce the best outcomes based on our **model of the world**



Three ~~Two~~ Pathways for Learning

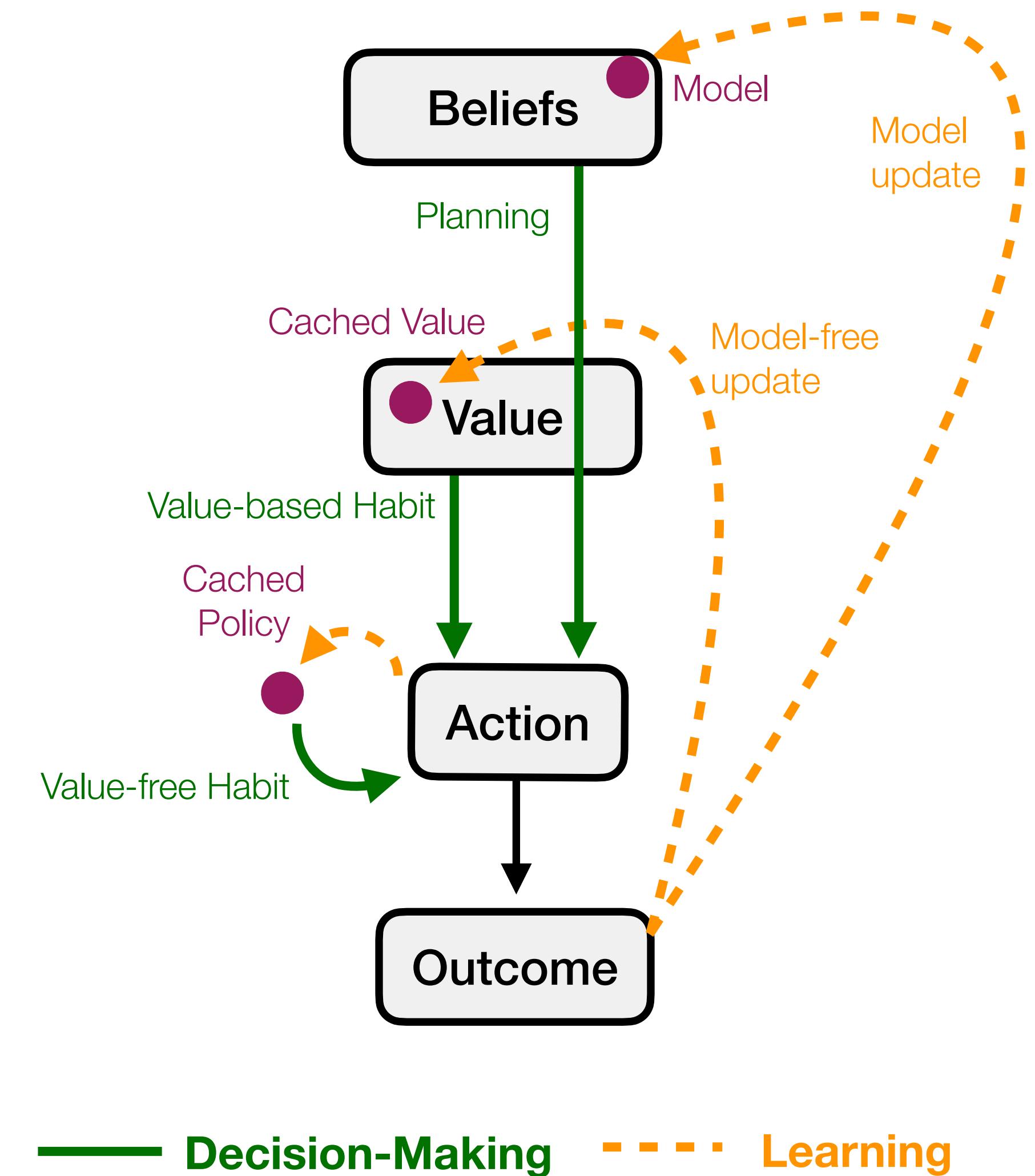
- **Law of Exercise:** Repeat actions performed in the past by learning a **cached policy** (independent of reward)
- **Law of Effect:** Choose actions on the basis of what worked in the past by forming **cached value**
 - **Model-free RL**
- **Model-based planning:** Select actions expected to produce the best outcomes based on our **model of the world**
 - **Model-based RL**



Three ~~-Two~~ Pathways for Learning

- **Law of Exercise:** Repeat actions performed in the past by learning a **cached policy** (independent of reward)
- **Law of Effect:** Choose actions on the basis of what worked in the past by forming **cached value**
 - **Model-free RL**
- **Model-based planning:** Select actions expected to produce the best outcomes based on our **model of the world**
 - **Model-based RL**

Different pathways not always in competition, but can inform one another! Model-based planning builds better habits!

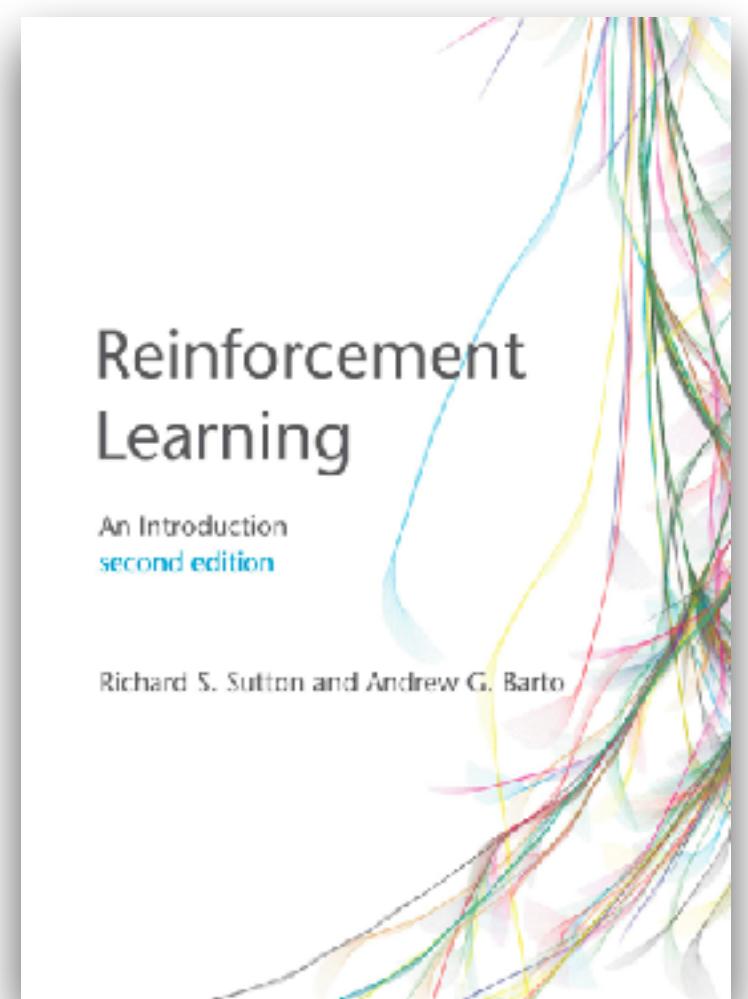


Model-free vs. Model-based summary

- Computationally cheap to use **model-free learning**
 - Maps onto habits and S-R learning
- Costly but potentially more impactful to use **model-based learning**
 - Maps onto goal-directed and S-S learning
- Model-based learning can help train model-free value functions and policies
- Still open questions about how model-based representations are learned and used in humans (find out next week!)

Further study

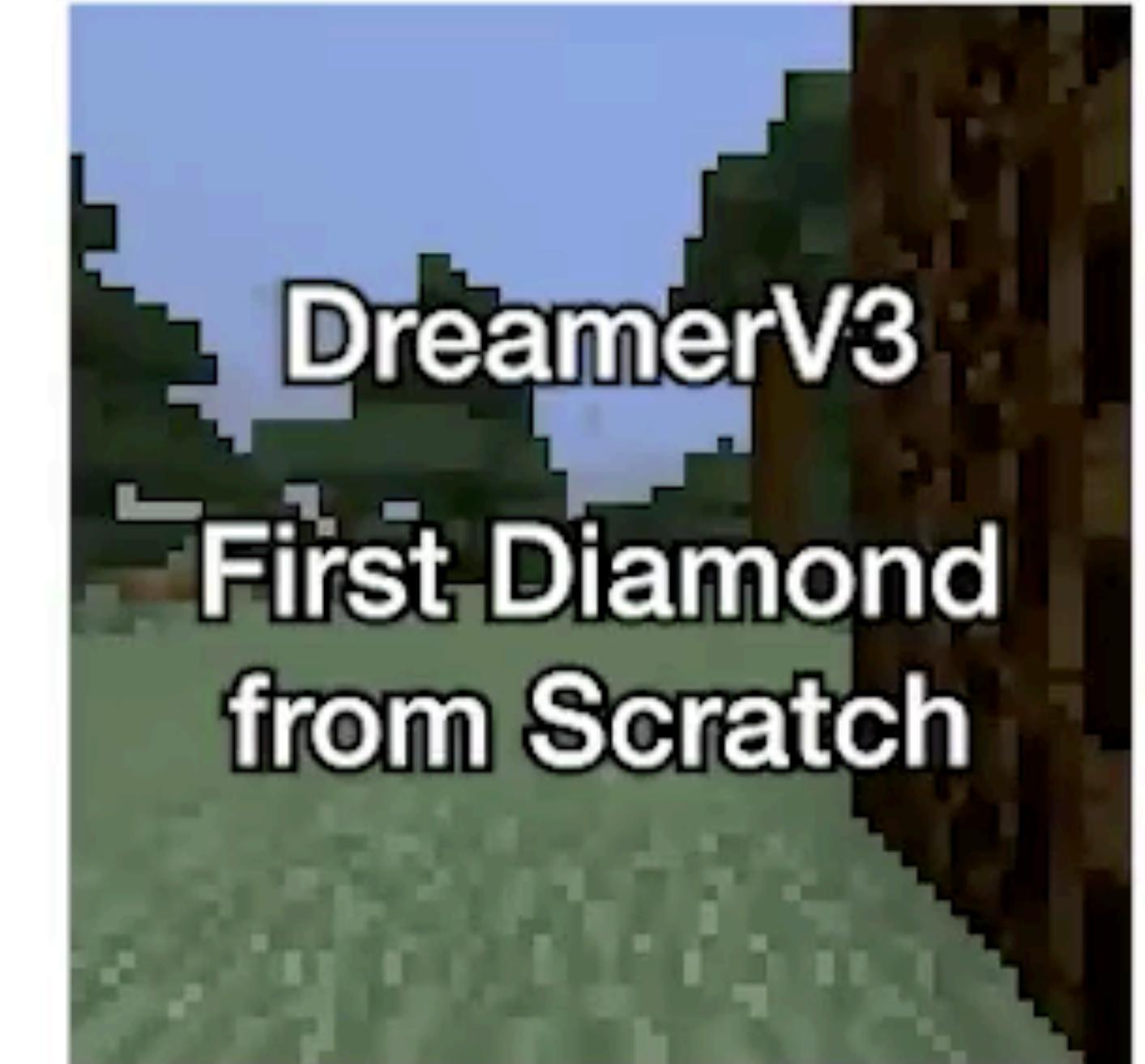
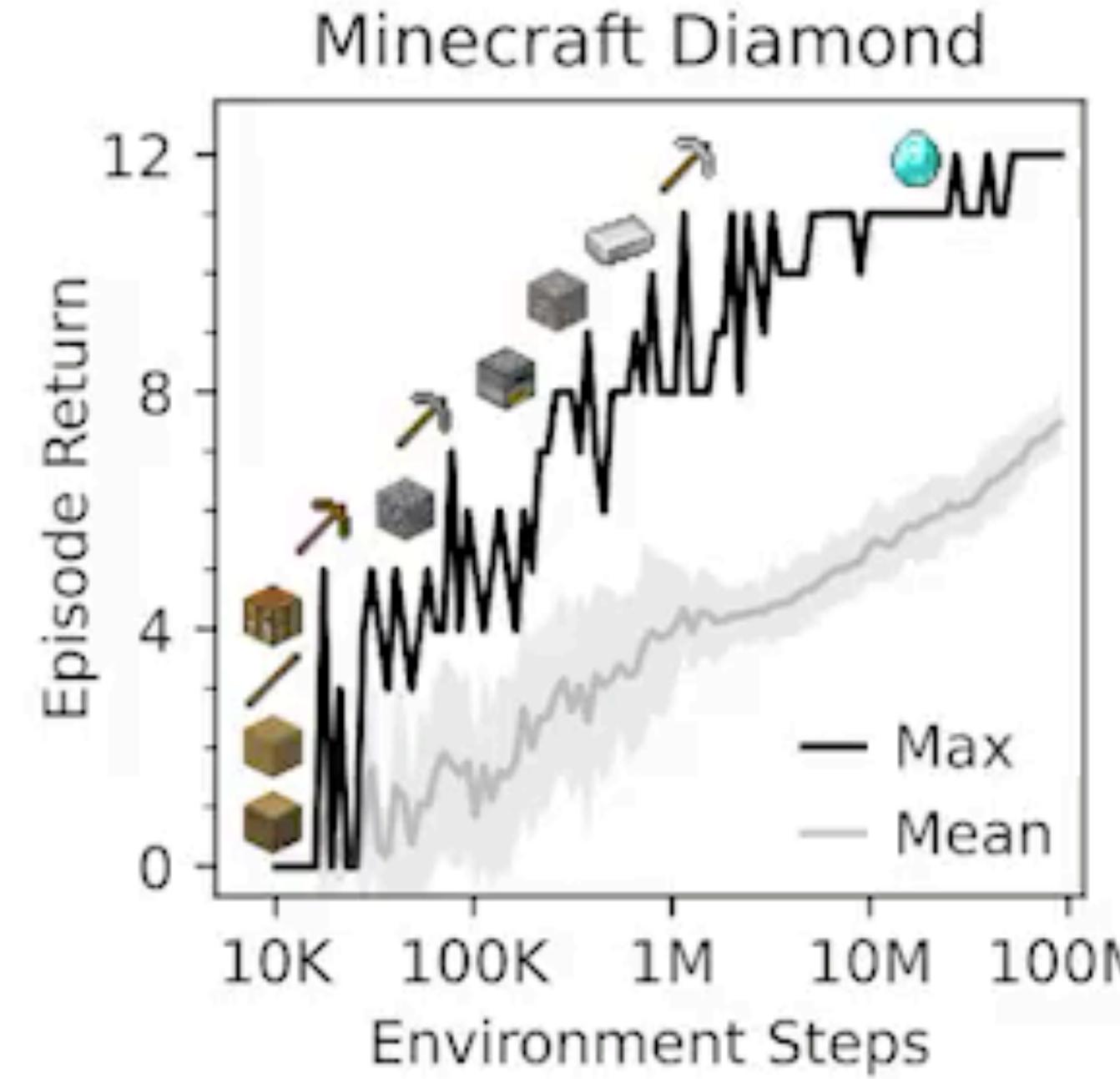
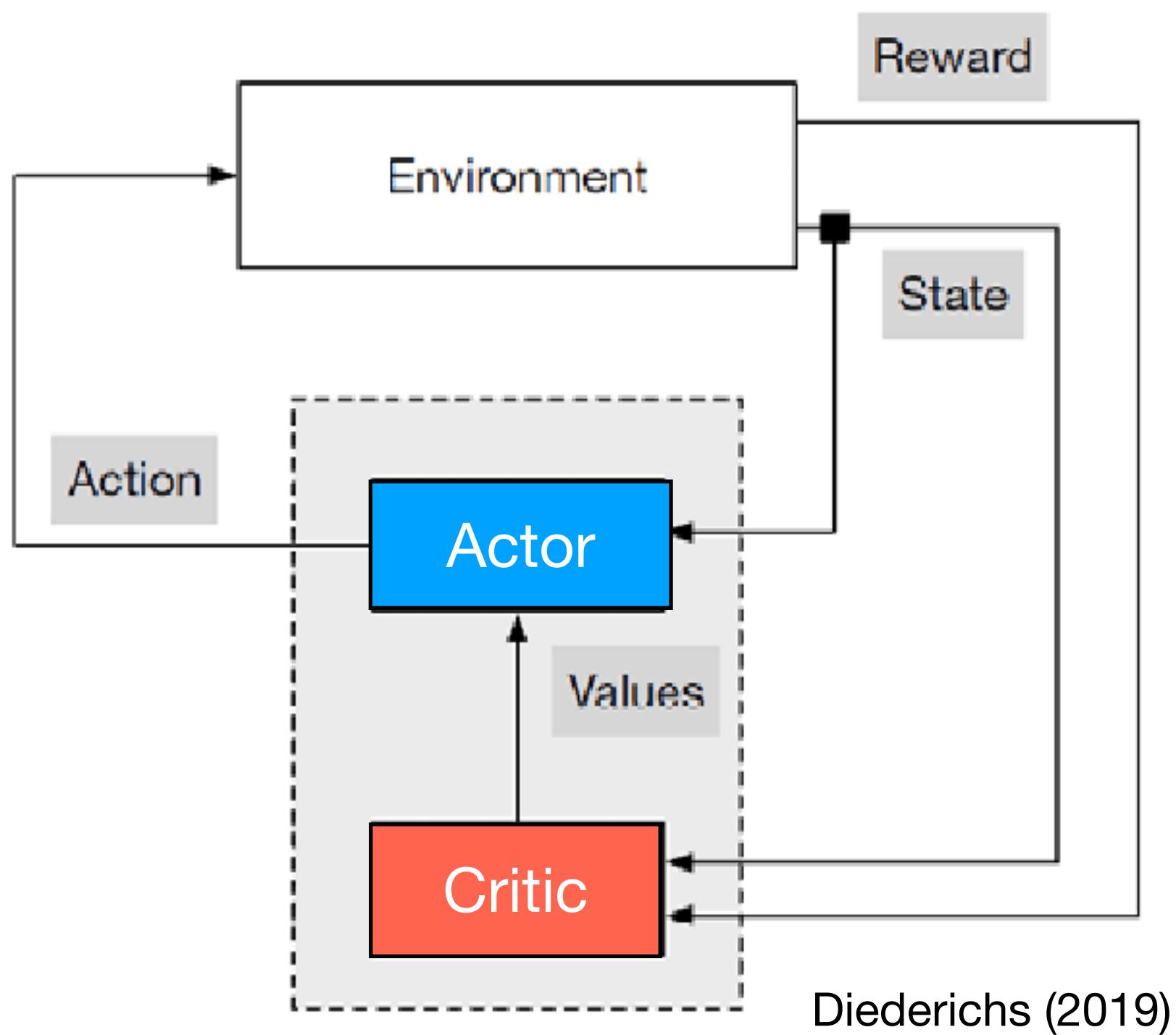
Sutton & Barto book ([free PDF link](#))



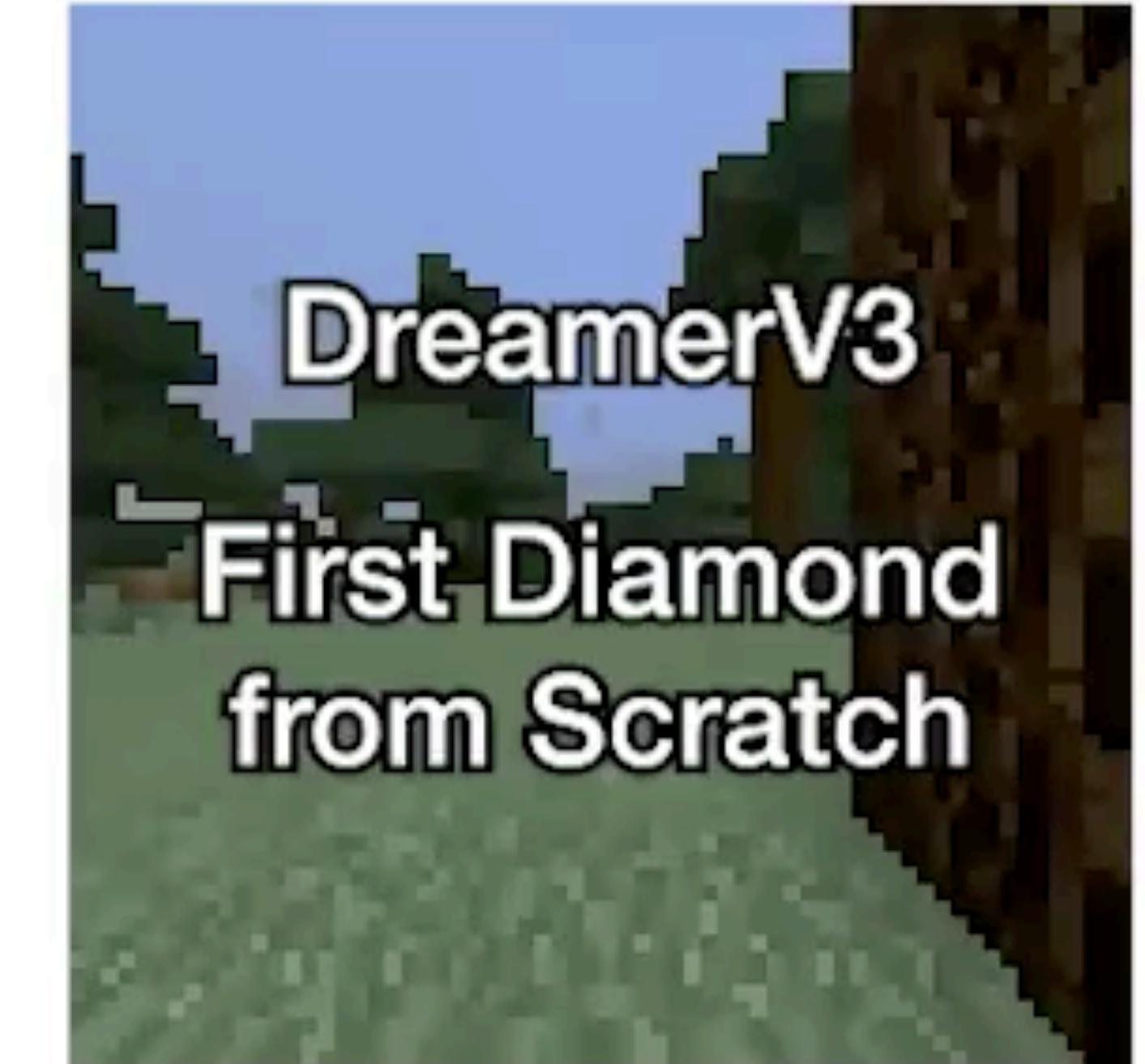
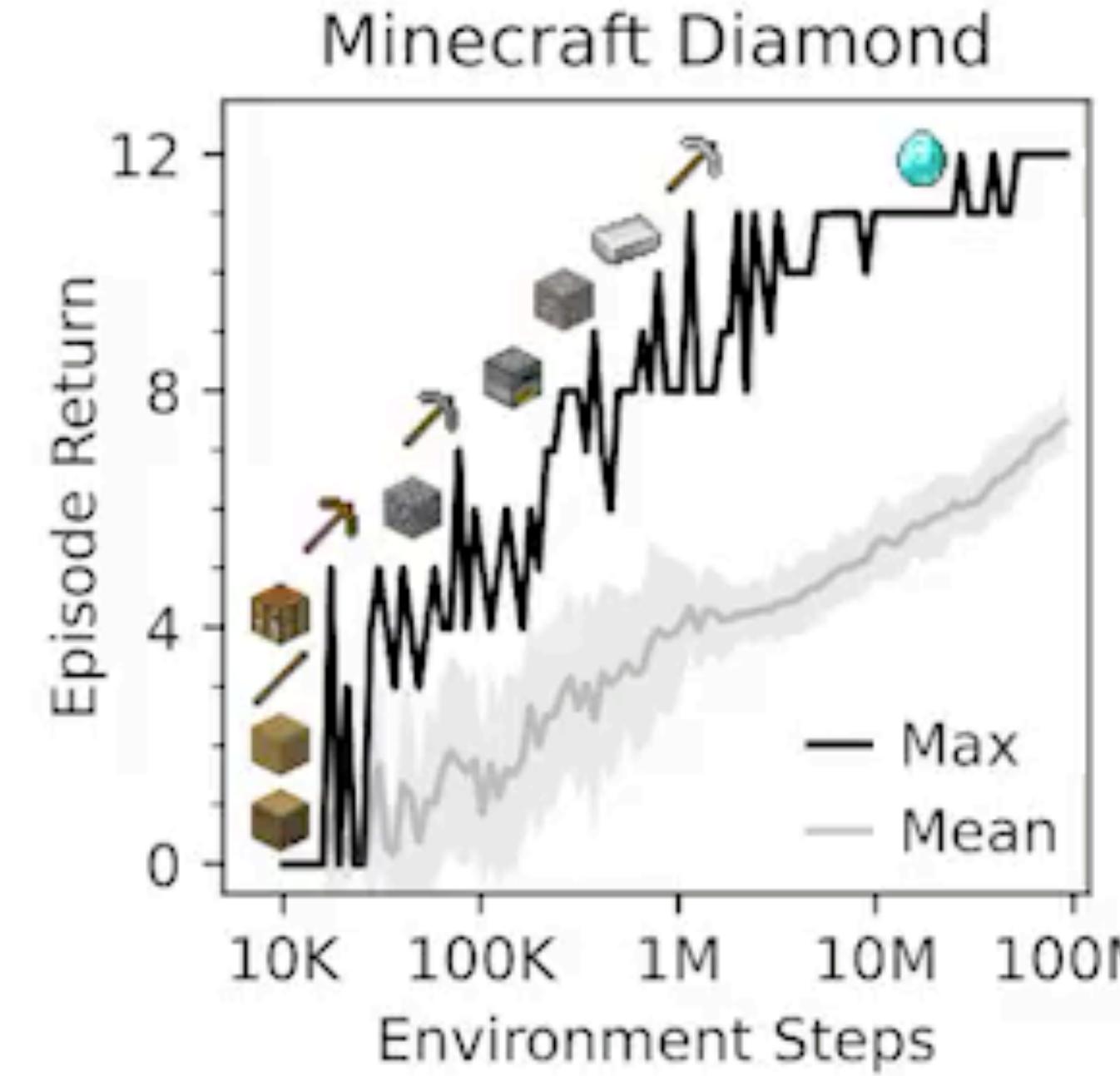
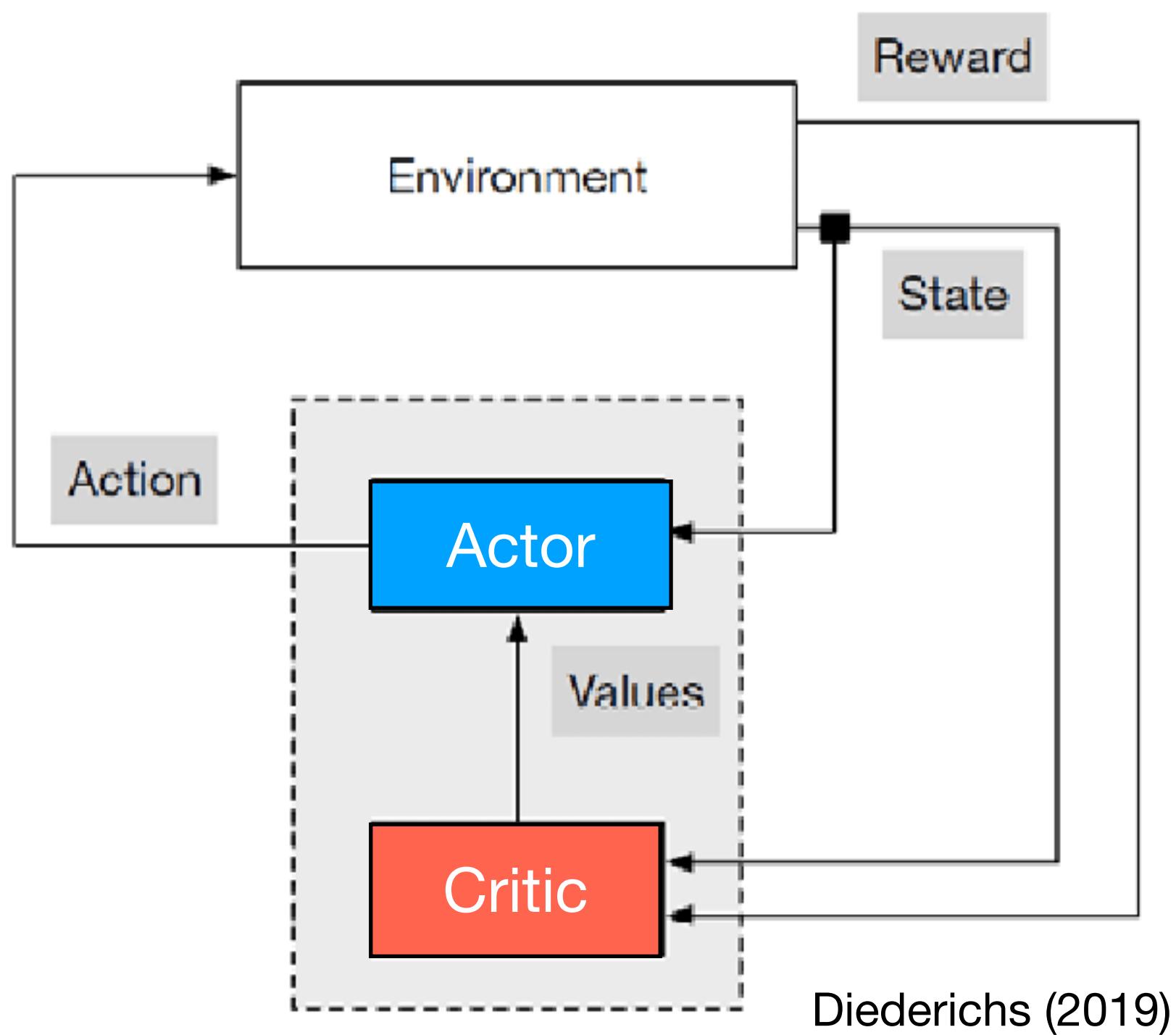
R code notebooks for using RL models (with a focus on social learning)
<https://cosmos-konstanz.github.io/materials/>

Python tutorial from Neuromatch academy
https://compneuro.neuromatch.io/tutorials/W3D4_ReinforcementLearning/student/W3D4_Intro.html

Next week Advances in RL



Next week Advances in RL



Haffner et al., (2023)

