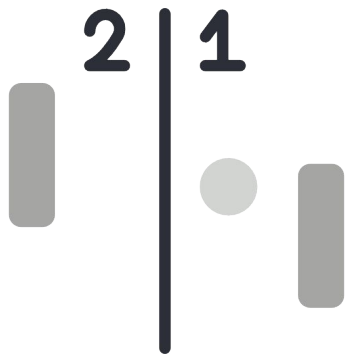


# Chapter 3: Finite Markov Decision Processes

Ryan Lee

# Markov Decision Process (MDP)

- Simplified, flexible reinforcement learning problem
- Consists of States  $\mathcal{S}$  , Actions  $\mathcal{A}$  , Rewards  $\mathcal{R}$



**States**

Info available to agent



**Actions**

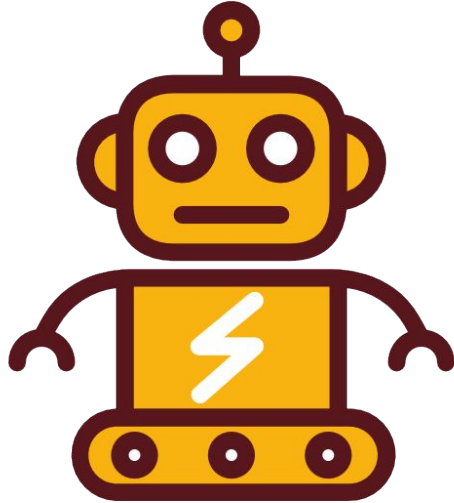
Choice made by agent



**Rewards**

Basis for evaluating choices

# Agent



**The learner**  
Takes action

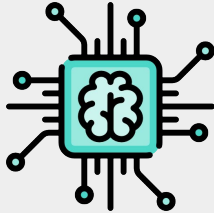
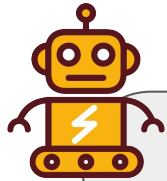
# Environment



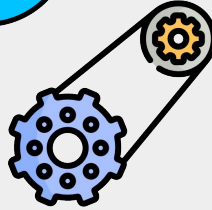
**Everything outside the agent**  
Returns state and reward

# Agent-Environment Boundary

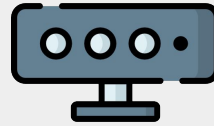
- Anything the agent cannot **arbitrarily change** is part of the environment
  - Agent might still **know** everything about the environment
- Different boundaries for different purposes



“Brain”



Machinery



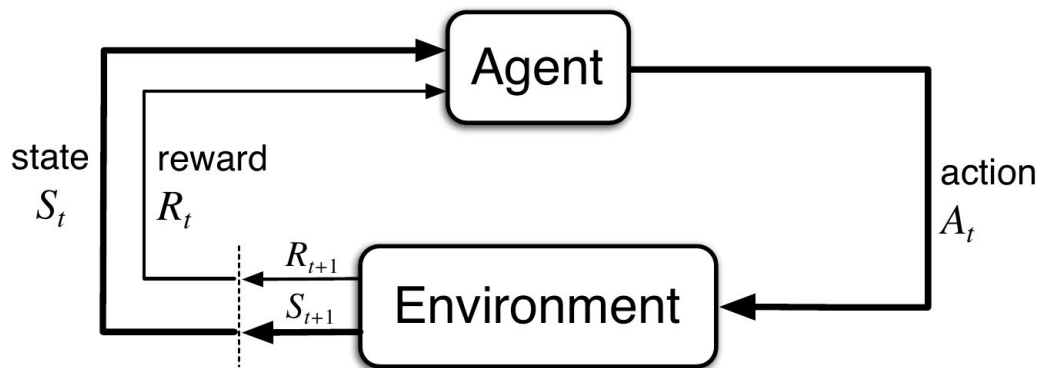
Sensors



Battery

# Agent-Environment Interactions

1. Agent observes a state  $S_0$
2. Agent takes action  $A_0$
3. Agent receives reward  $R_1$  and new state  $S_1$
4. Agent takes another action  $A_1$
5. Repeat



# Transition Probability

- Probability of reaching state  $s'$  and reward  $r$  by taking action  $a$  on state  $s$
- **Fully describes the dynamics of a finite MDP**

$$p(s', r \mid s, a) := \Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_t = a\}$$

- Can deduce other properties of the environment

$$p(s' \mid s, a) := \Pr\{S_t = s' \mid S_{t-1} = s, A_t = a\} = \sum_{r \in \mathcal{R}} p(s', r \mid s, a)$$

# Expected Rewards

- Expected reward of taking action  $a$  on state  $s$

$$r(s, a) := \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r \mid s, a)$$

- Expected reward of arriving in state  $s'$  by taking action  $a$  on state  $s$

$$r(s, a, s') := \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r \in \mathcal{R}} r \frac{p(s', r \mid s, a)}{p(s' \mid s, a)}$$

# Recycling Robot Example

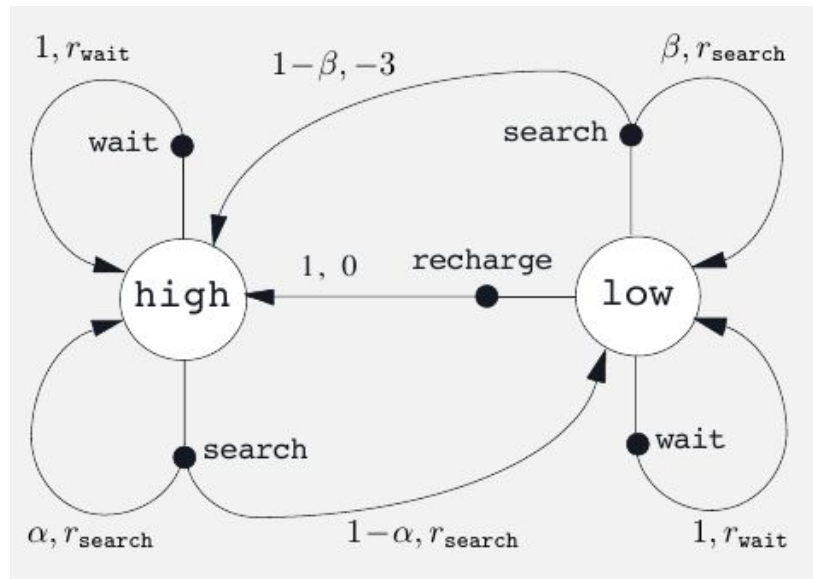
- States: Battery status (high or low)
- Actions
  - Search: High reward. Battery status can be lowered or depleted.
  - Wait: Low reward. Battery status does not change.
  - Recharge: No reward. Battery status changed to high.
- If battery is depleted, -3 reward and battery status changed to high.

$s$	$a$	$s'$	$p(s'   s, a)$	$r(s, a, s')$
high	search	high	$\alpha$	$r_{\text{search}}$
high	search	low	$1 - \alpha$	$r_{\text{search}}$
low	search	high	$1 - \beta$	-3
low	search	low	$\beta$	$r_{\text{search}}$
high	wait	high	1	$r_{\text{wait}}$
high	wait	low	0	$r_{\text{wait}}$
low	wait	high	0	$r_{\text{wait}}$
low	wait	low	1	$r_{\text{wait}}$
low	recharge	high	1	0
low	recharge	low	0	0



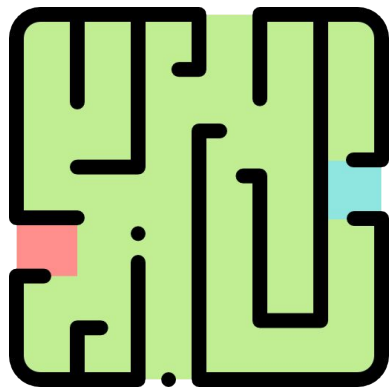
# Transition Graph

- Graphical summary of MDP dynamics

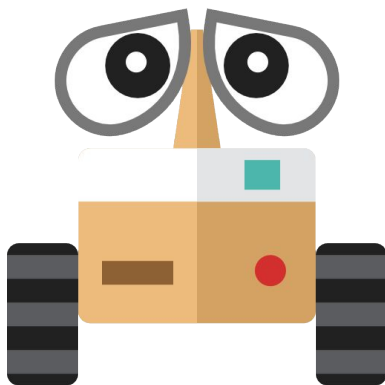


# Designing Rewards

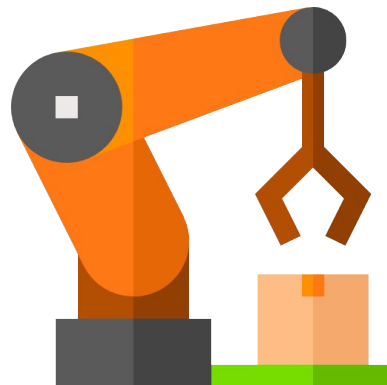
- Reward hypothesis
  - Goals and purposes can be represented by maximization of cumulative reward
- Tell *what* you want to achieve, not *how*



Always -1



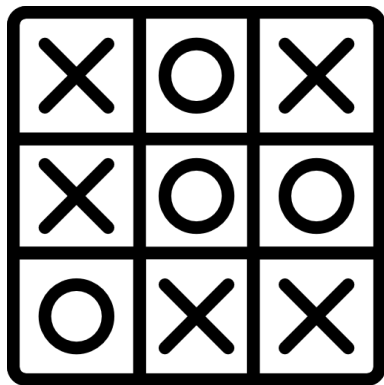
Proportional to  
forward action



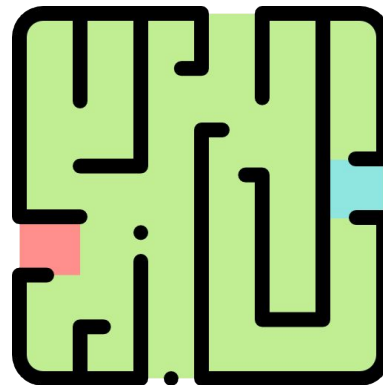
+1 for each box

# Episodic Tasks

- Interactions can be broken into *episodes*
- Episodes end in a special *terminal state*
- Each episode is independent



Finished when the game ends



Finished when the agent is out of the maze

# Return for Episodic Tasks

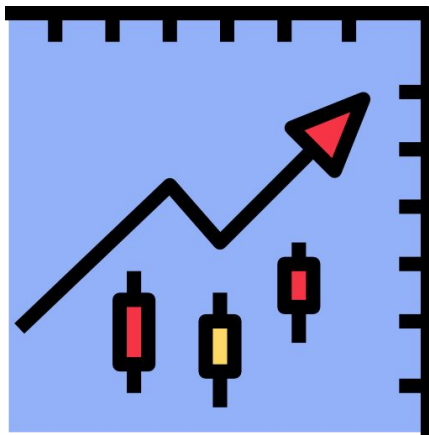
- Sum of rewards from time step  $t$
- Time of termination:  $T$

$$G_t = R_{t+1} + R_{t+2} + \dots R_T$$

$$G_t = \sum_{k=t+1}^T R_k$$

# Continuing Tasks

- Cannot be naturally broken into episodes
- Goes on without limit



Stock Trading

# Return for Continuing Tasks

- Sum of rewards is almost always infinite
- Need to *discount* future rewards by factor  $0 \leq \gamma < 1$ 
  - If  $\gamma = 0$ , the return only considers immediate reward (*myopic*)

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \dots$$

$$G_t = \sum_{k=t+1}^{\infty} \gamma^{k-t-1} R_k$$

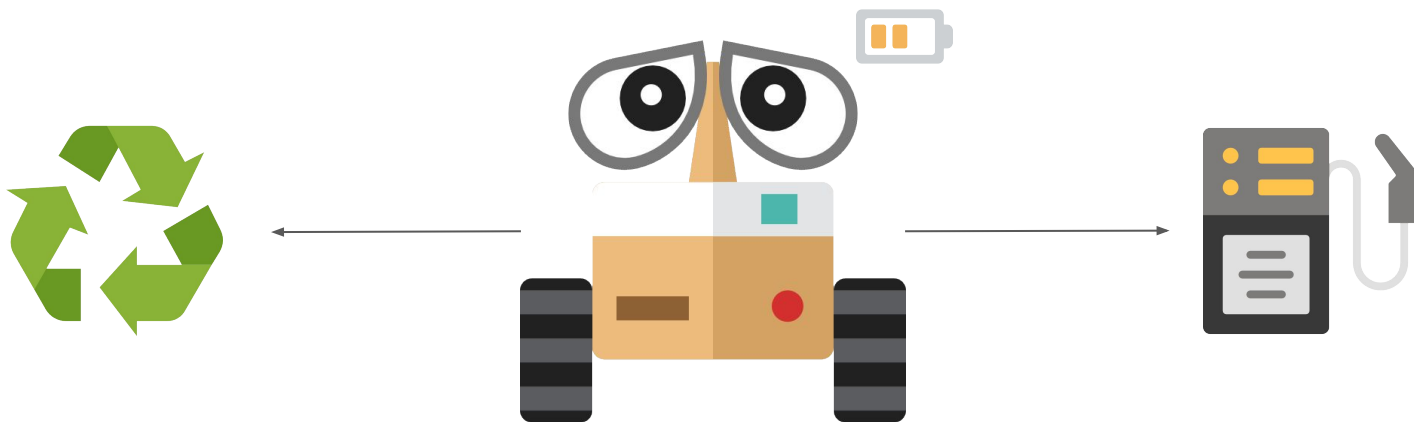
# Unified Notation for Return

- Cumulative reward
- $T$  can be a finite number or infinity
- Future rewards can be *discounted* with factor  $\gamma$ 
  - If  $T = \infty$ , then  $\gamma$  must be less than 1.

$$G_t := \sum_{k=t+1}^T \gamma^{k-t-1} R_k$$

# Policy

- Mapping from states to probabilities of selecting each possible action
- $\pi(a \mid s)$ : Probability of selecting action  $a$  in state  $s$





# State-value function

- Expected return from state  $s$  and following policy  $\pi$

$$v_{\pi} := \mathbb{E}_{\pi}[G_t \mid S_t = s]$$
$$:= \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right]$$

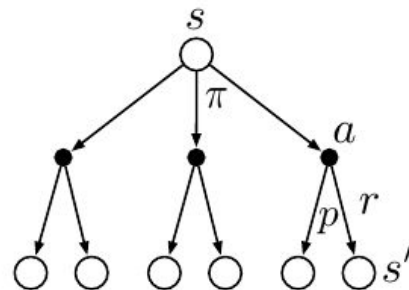
# Action-value function

- Expected return from taking action  $a$  in state  $s$  and following policy  $\pi$

$$q_{\pi} := \mathbb{E}_{\pi}[G_t \mid S_t = s, A_t = a]$$
$$:= \mathbb{E}_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right]$$

# Bellman Equation

- Recursive relationship between  $v_\pi(s)$  and  $v_\pi(s')$



$$v_\pi(s) = \sum_a \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a) [r + \gamma v_\pi(s')]$$

# Optimal Policies $\pi_*$ and Value Functions $v_*, q_*$

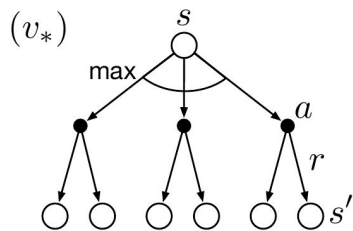
- For any policy  $\pi$ ,  $v_{\pi_*}(s) \geq v_{\pi}(s)$  for all states  $s$
- There can be multiple optimal policies
- All optimal policies share same optimal value functions:

$$v_*(s) \doteq \max_{\pi} v_{\pi}(s)$$

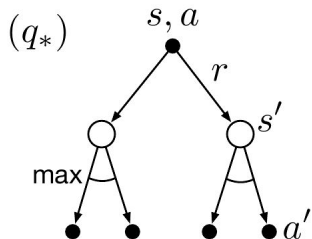
$$q_*(s, a) \doteq \max_{\pi} q_{\pi}(s, a)$$

# Bellman Optimality Equation

- Bellman Equation for optimal policies



$$v_*(s) = \max_a \sum_{s', r} p(s', r \mid s, a) [r + \gamma v_*(s')]$$



$$q_*(s, a) = \sum_{s', r} p(s', r \mid s, a) \left[ r + \gamma \max_{a'} q_*(s', a') \right]$$

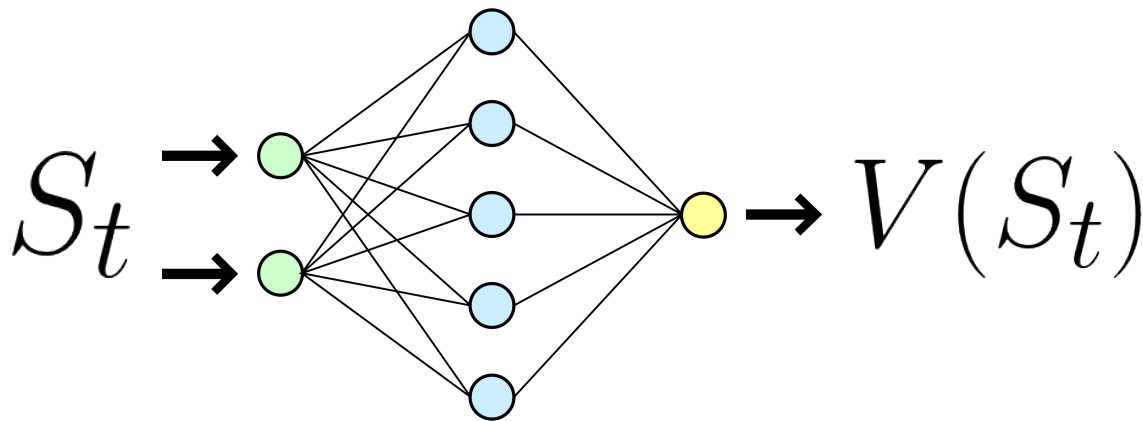
# Solving Bellman Optimality Equation

- Linear system:  $|\mathcal{S}|$  equations,  $|\mathcal{S}|$  unknowns
- Possible to find the exact optimal policy
- Impractical in most environments
  - Need to know the dynamics of the environment
  - Need extreme computational power
  - Need Markov property

→ In most cases, approximation is the best possible solution.

# Approximation

- Does not require complete knowledge of environment
- Less memory and computational power needed
- Can focus learning on frequently encountered states



# Thank you!

Original content from

- [Reinforcement Learning: An Introduction by Sutton and Barto](#)

You can find more content in

- [github.com/seungjaeryanlee](#)
- [www.endtoend.ai](#)