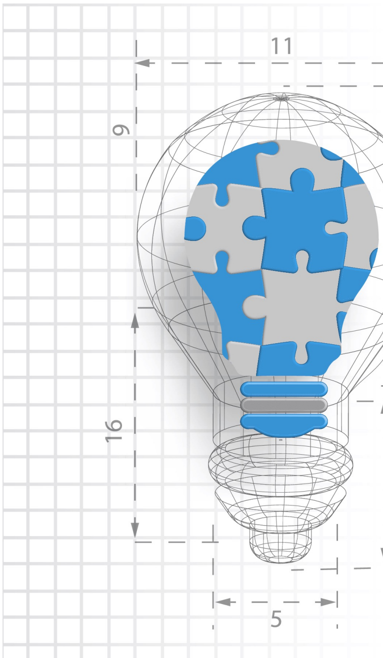
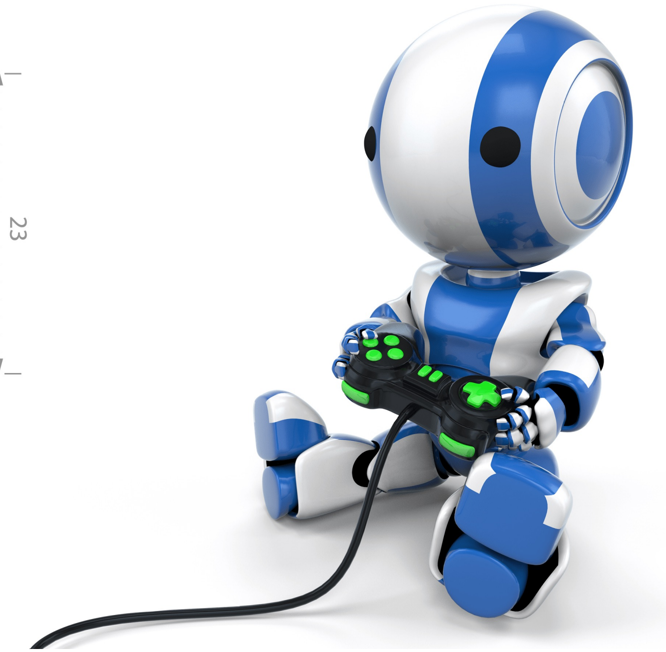
AIML231

Reinforcement Learning





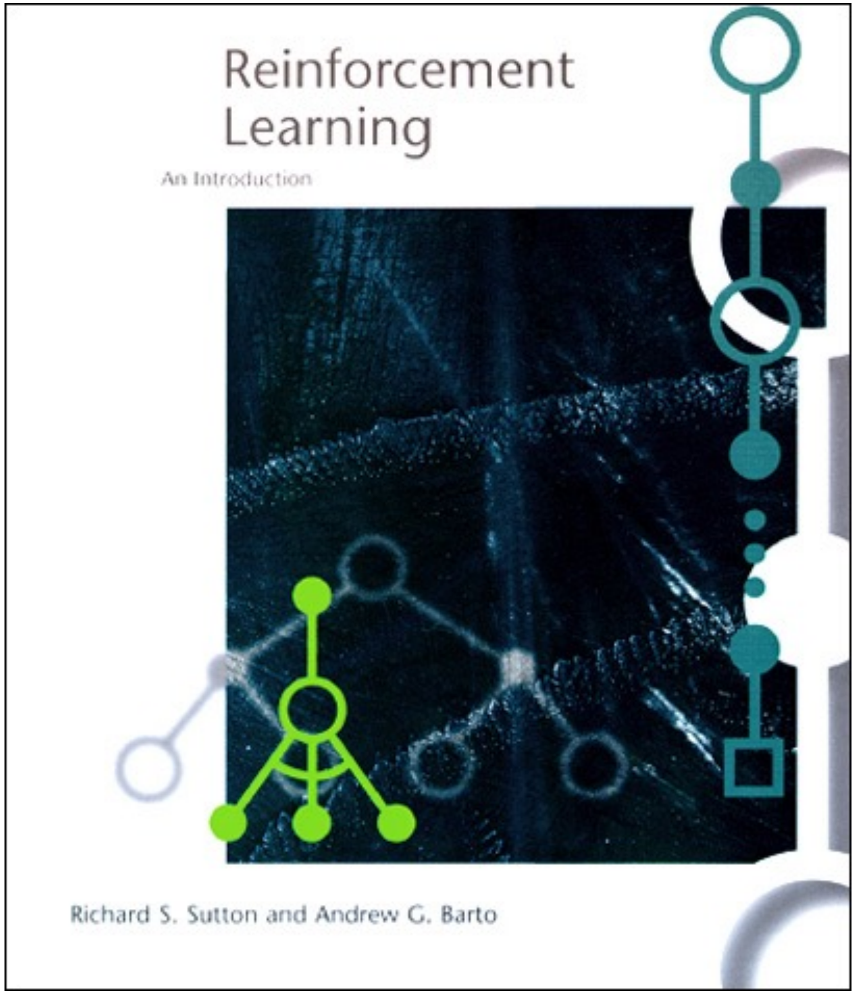
Outline

- RL examples

- Defining an RL problem
  - Markov Decision Processes

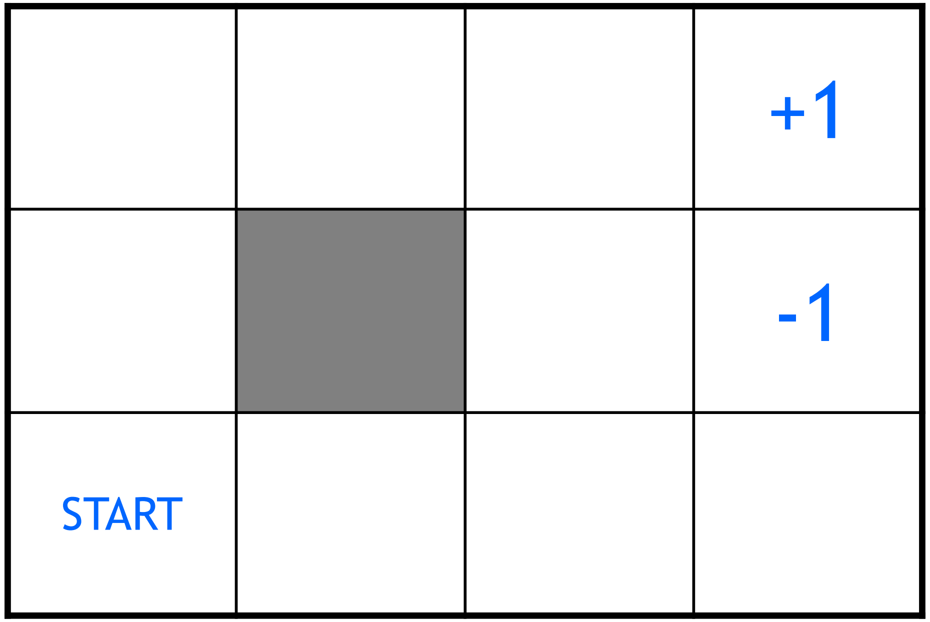
- Fundamental techniques for RL
  - Iterative policy improvement
  - Q-learning
  - Direct policy search algorithms

The RL introduction book



Richard Sutton and Andrew Barto, Reinforcement Learning – an Introduction
http://incompleteideas.net/book/bookdraft2017nov5.pdf

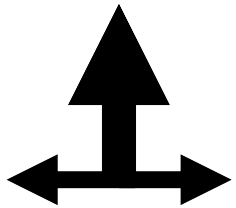
Robot in a room



actions: UP, DOWN, LEFT, RIGHT

UP

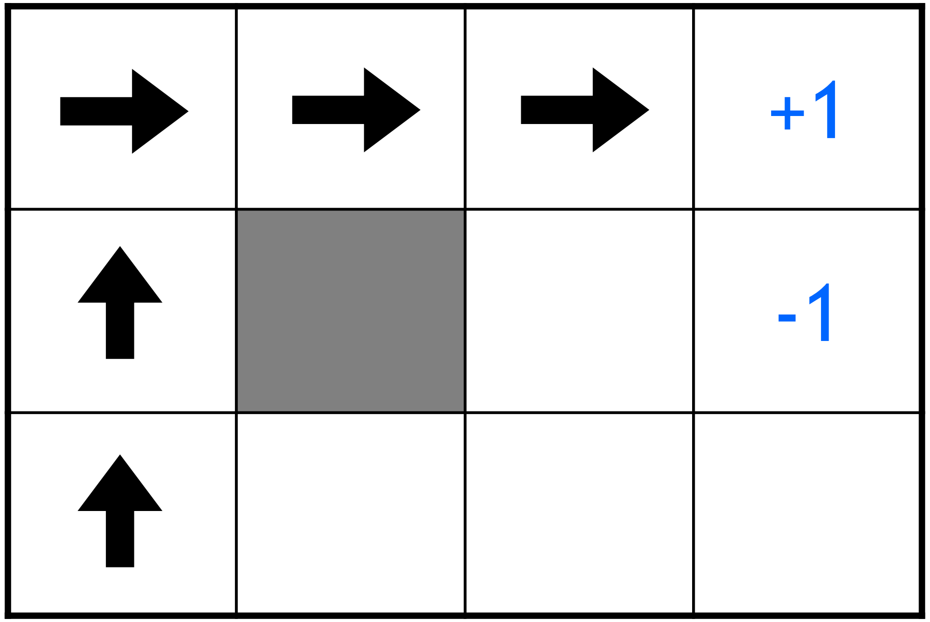
80% move UP
10% move LEFT
10% move RIGHT



- reward +1 at terminate state 1, -1 at terminal state
- reward -0.04 for each step

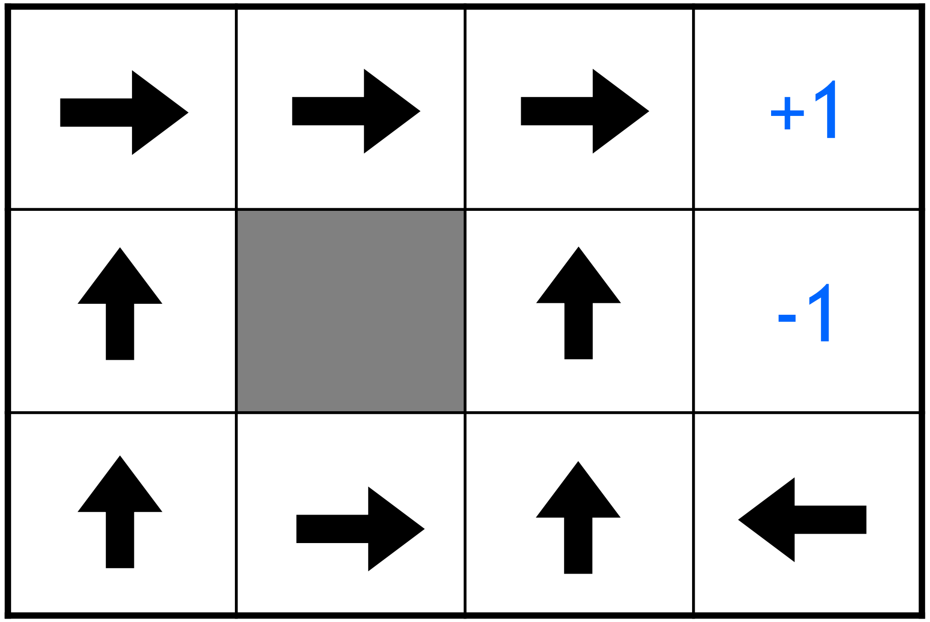
- what’s the strategy to achieve max reward?
- what if the actions were deterministic?

Is this a solution?

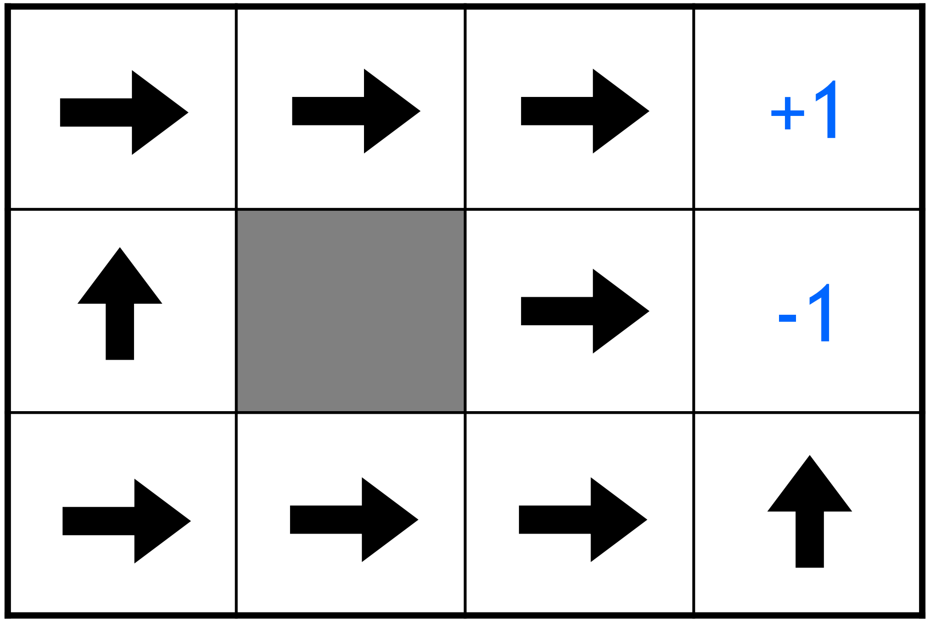


- Only if actions are deterministic
  - not in this case (actions are stochastic)
- Solution/policy
  - mapping from each state to an action

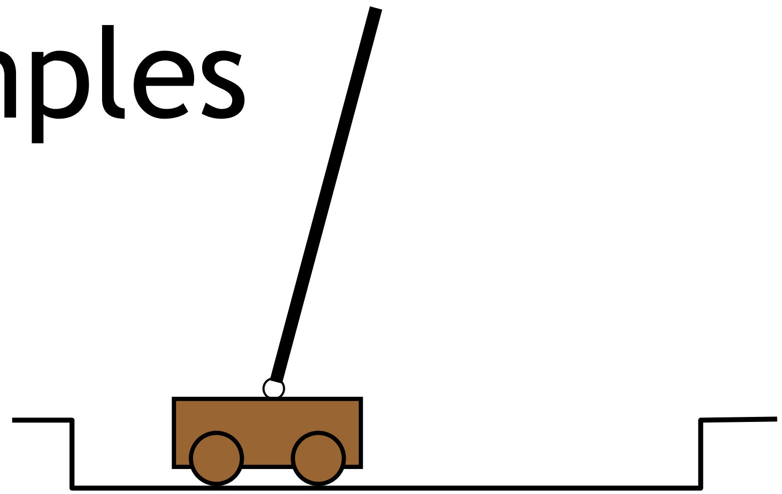
Optimal policy



Reward for each step: -2



Other examples

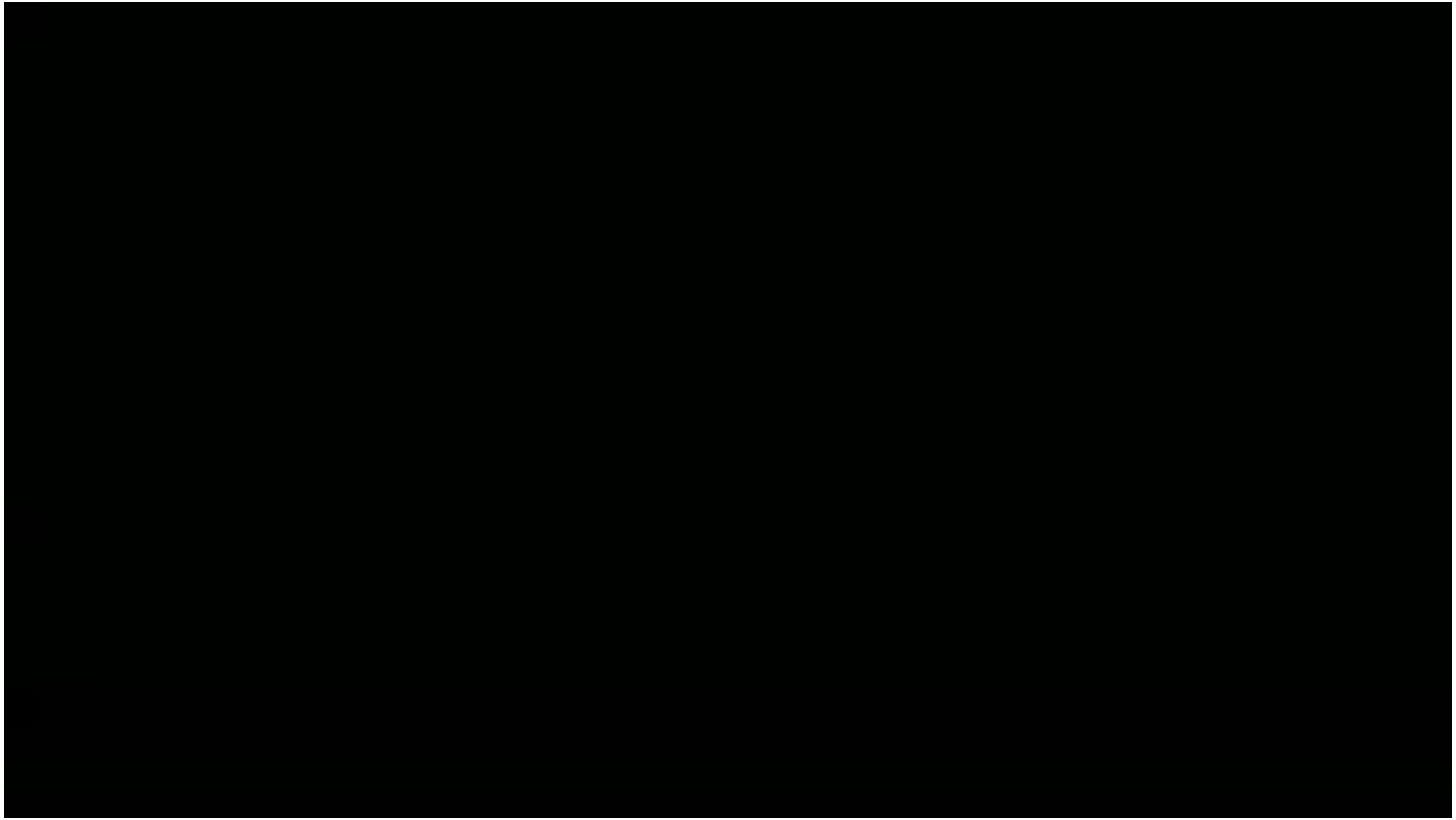


- Pole-balancing
- TD-Gammon [Gerry Tesauro]
- Helicopter [Andrew Ng]
  - *https://youtu.be/0JL04JJjocc*

- No teacher who would say “good” or “bad”
  - is reward “10” good or bad?
  - rewards could be delayed

- Explore the environment and learn from experience
  - not just blind search, try to be smart about it

Helicopter fly through RL



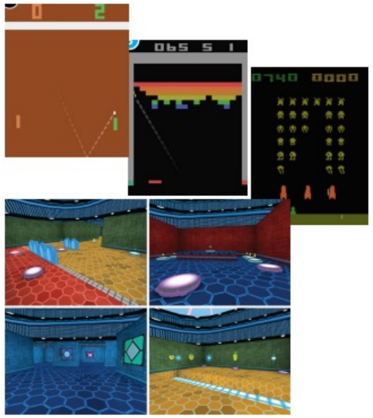
Successful stories

Kohl and Stone, 2004

Ng et al, 2004

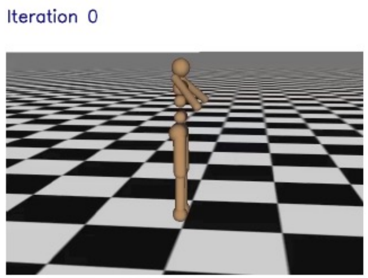
Tedrake et al, 2005

Kober and Peters, 2009

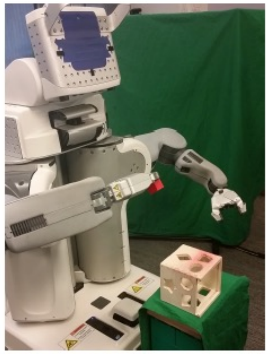


Mnih et al, 2015 (A3C)

Silver et al, 2014 (DPG)
Lillicrap et al, 2015 (DDPG)



Schulman et al, 2016 (TRPO + GAE)



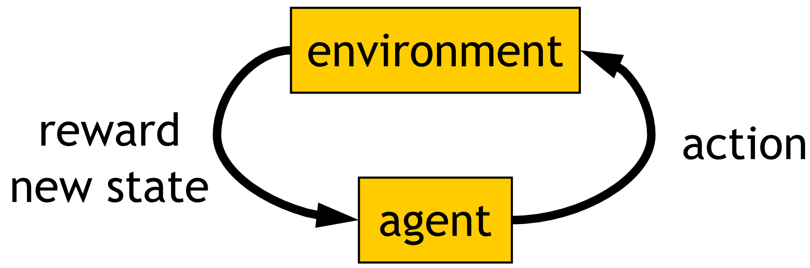
Levine*, Finn*, et al, 2016 (GPS)

Silver*, Huang*, et al, 2016 (AlphaGo)

- Practical tool in robotics, operations, animation, games
- Used by other ML methods (hard attention, neural architecture search)

Markov Decision Process (MDP)

- Set of states S, set of actions A, initial state S_0
- Transition model P(s,a,s')
  - P([1,1], up, [1,2]) = 0.8
- Reward function r(s,a)
  - r([4,3],up) = +1

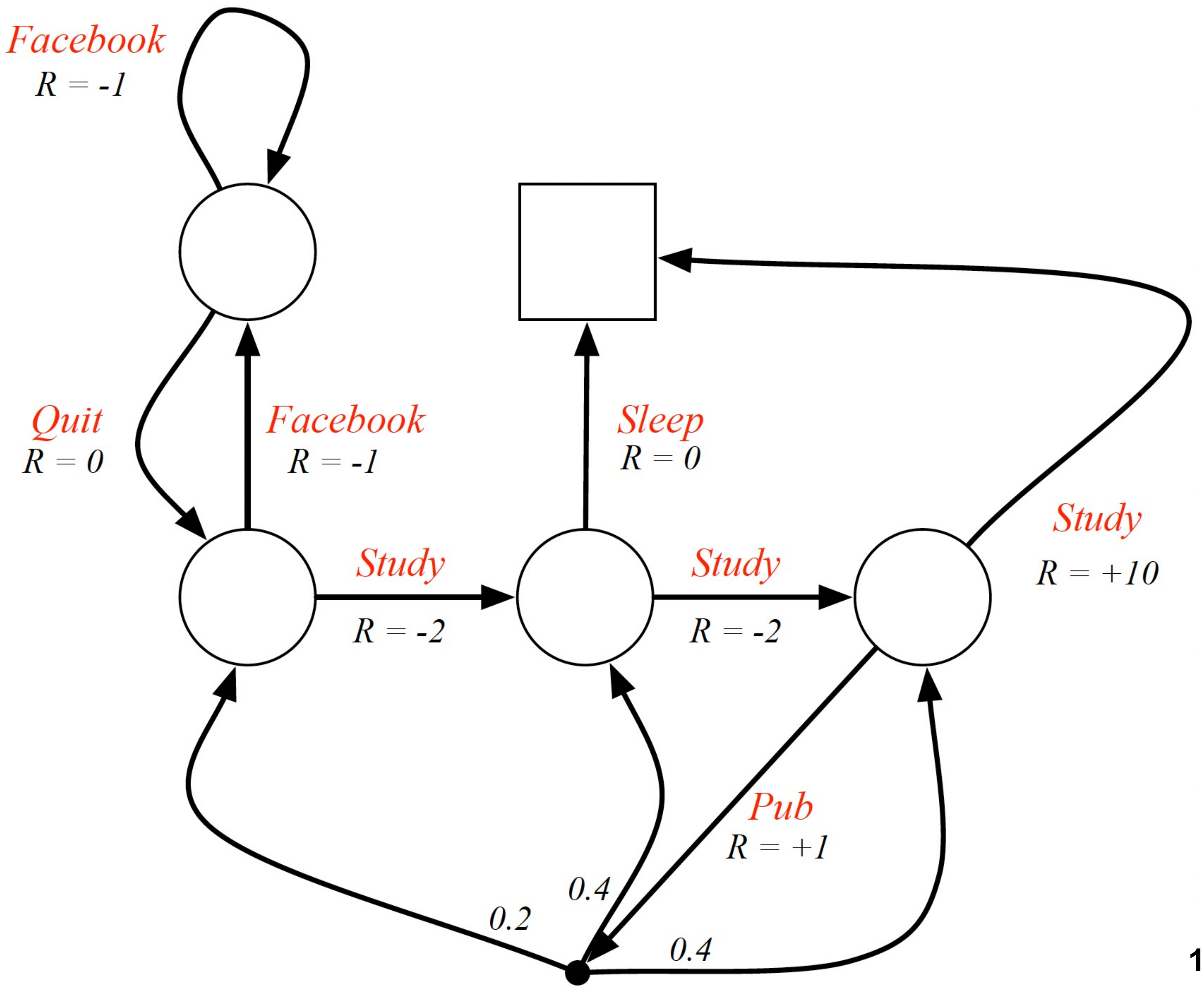


- Goal: maximize cumulative reward in the long run

- Policy: mapping from S to A
  - $\pi(s)$ or $\pi(s,a)$ (deterministic vs. stochastic)

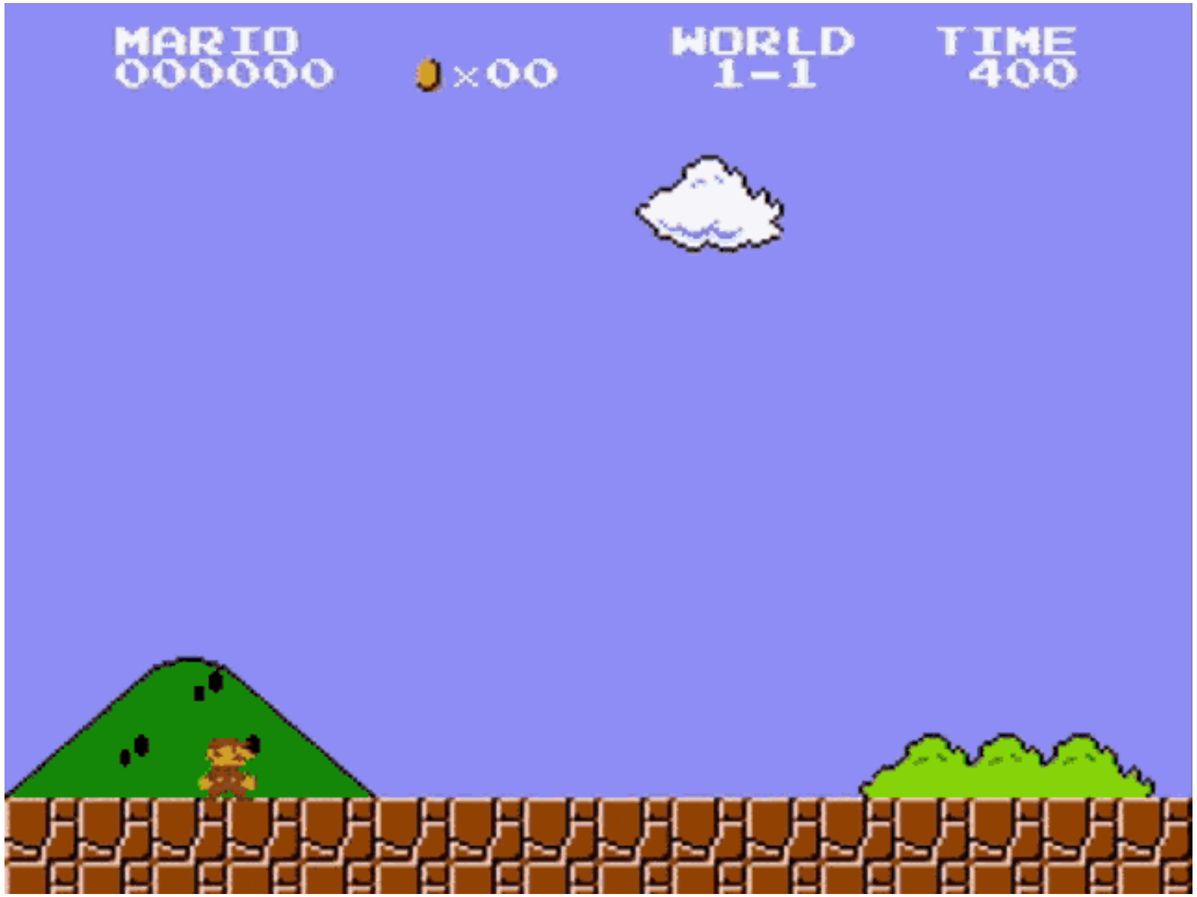
- Reinforcement learning
  - transitions and rewards usually not available
  - how to change the policy based on experience
  - how to explore the environment

Example: student MDP



Exercise question

- How to define the MDP of the Super Mario game?

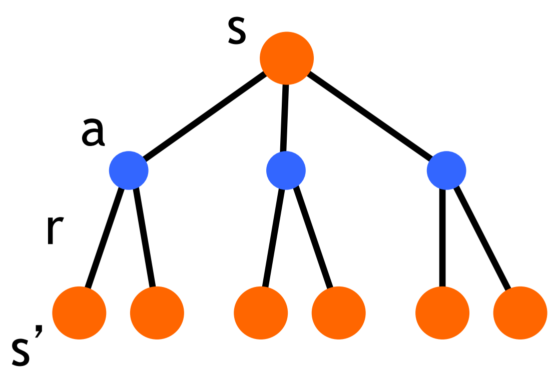


Value functions

- State value function: $V^{\pi}(s)$
  - expected return when starting in *s* and following π

- State-action value function: $Q^{\pi}(s,a)$
  - expected return when starting in *s*, performing *a*, and following π

- Useful for finding the optimal policy
  - can estimate from experience
  - pick the best action using $Q^{\pi}(s,a)$



- Bellman equation

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P^{a}_{ss'} \left[r^{a}_{ss'} + \gamma V^{\pi}(s') \right] = \sum_{a} \pi(s,a) Q^{\pi}(s,a)$$

Iterative policy improvement

- Two main components
  - Policy evaluation: compute V^{π} from π
  - Policy improvement: improve π based on V^{π}

- Start with an arbitrary policy
- Repeat evaluation/improvement until convergence

$$\pi_0 \rightarrow^{E} V^{\pi_0} \rightarrow^{I} \pi_1 \rightarrow^{E} V^{\pi_1} \rightarrow^{I} \ldots \rightarrow^{I} \pi^* \rightarrow^{E} V^*$$

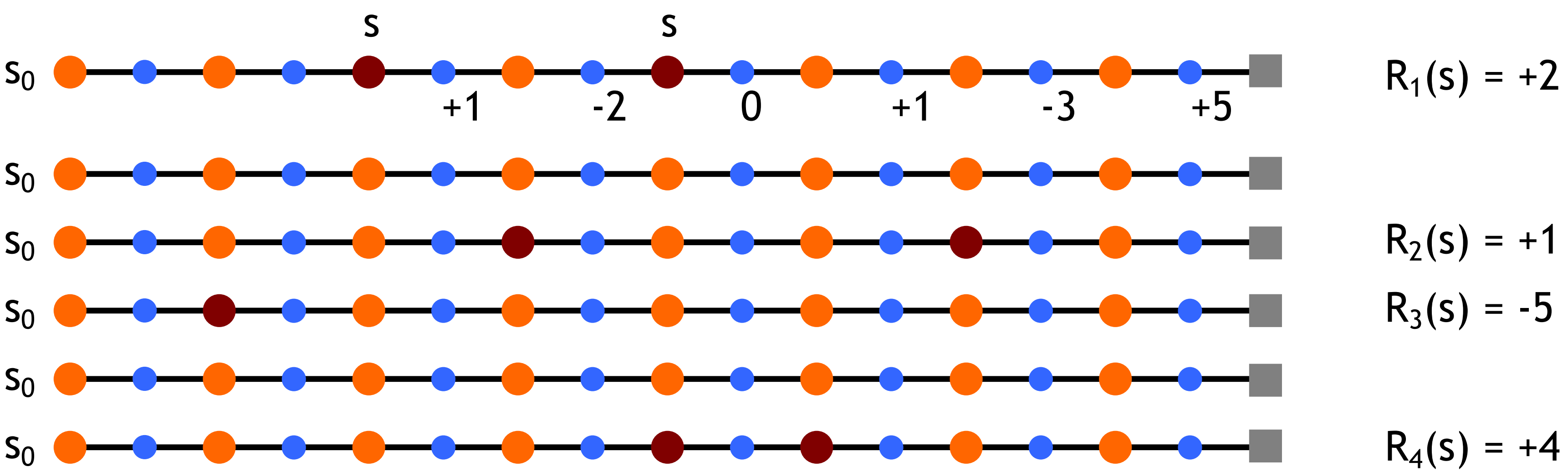
Monte Carlo methods

- Don't need full knowledge of environment
  - just experience, or
  - simulated experience
- Approximated policy evaluation and improvement

Monte Carlo policy evaluation

- Want to estimate $V^\pi(s)$
  - expected return starting from s and following π
  - estimate as average of observed returns in state s

- First-visit MC
  - average returns following the first visit to state s



$$V^\pi(s) \approx (2 + 1 - 5 + 4)/4 = 0.5$$

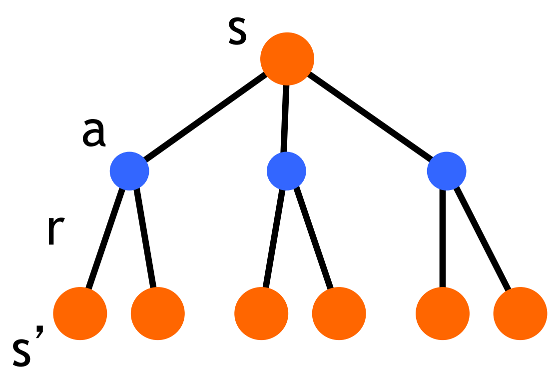
Optimal value functions

- There's a set of *optimal* policies
  - they share the same optimal value function

$$V^*(s) = \max_\pi V^\pi(s)$$

- Bellman optimality equation

$$V^*(s) = \max_a \sum_{s'} P^a_{ss'} \left[r^a_{ss'} + \gamma V^*(s') \right]$$



  - system of *n* non-linear equations
  - solve for V*(s)
  - easy to extract the optimal policy

- Having Q*(s,a) makes it even simpler

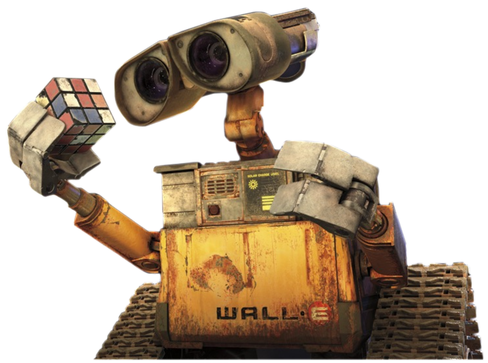
$$\pi^*(s) = \arg\max_a Q^*(s,a)$$

Q-learning

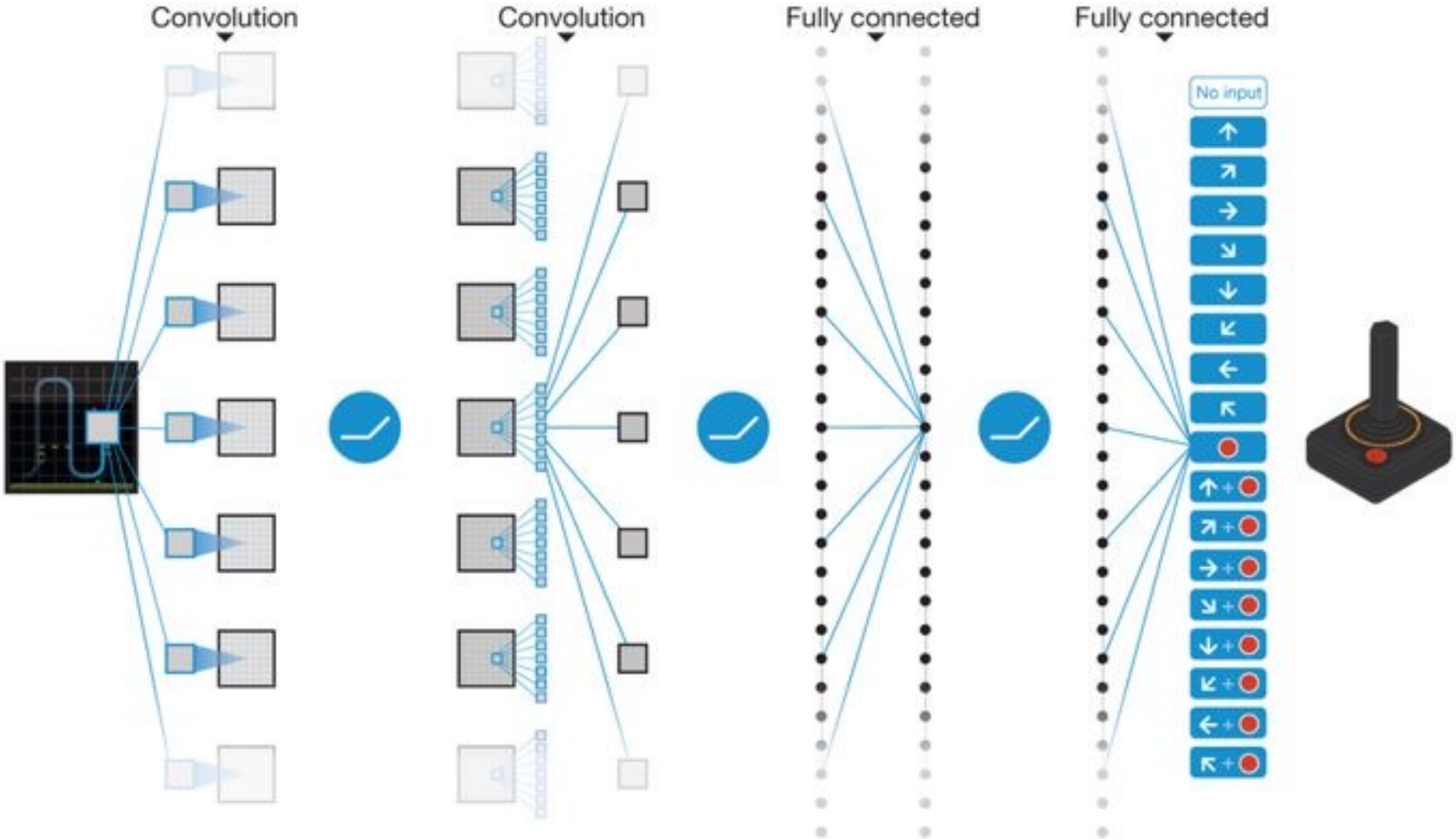
- Q-learning:
  - use any policy to estimate Q

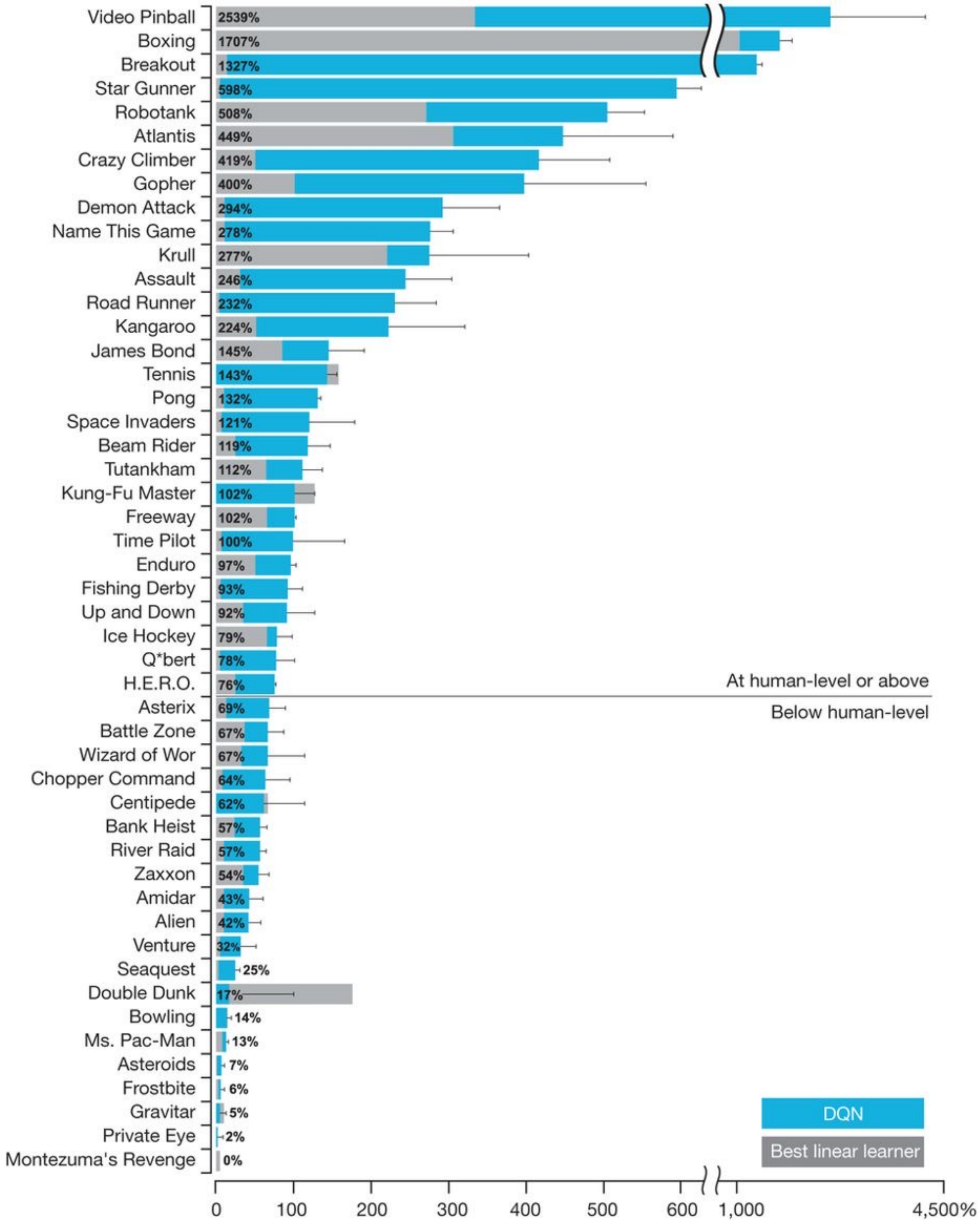
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

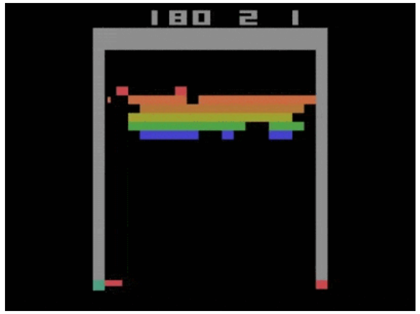
  - Q directly approximates Q* (Bellman optimality eqn)
  - Independent of the policy being followed
  - Only requirement: keep updating each (s,a) pair

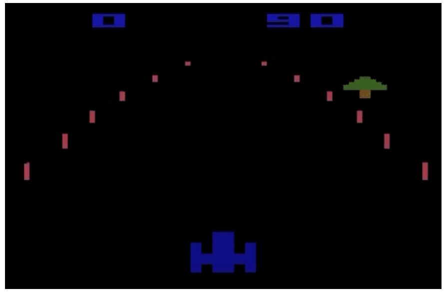


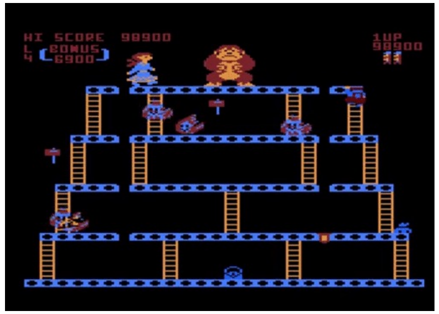
Deep Q-Network




Video Pinball 2539%
Boxing 1707%
Breakout 1327%
Star Gunner 598%
Robotank 508%
Atlantis 449%
Crazy Climber 419%
Gopher 400%
Demon Attack 294%
Name This Game 278%
Krull 277%
Assault 246%
Road Runner 232%
Kangaroo 224%
James Bond 145%
Tennis 143%
Pong 132%
Space Invaders 121%
Beam Rider 119%
Tutankham 112%
Kung-Fu Master 102%
Freeway 102%
Time Pilot 100%
Enduro 97%
Fishing Derby 93%
Up and Down 92%
Ice Hockey 79%
Q*bert 78%
H.E.R.O. 76%
At human-level or above
Below human-level
Asterix 69%
Battle Zone 67%
Wizard of Wor 67%
Chopper Command 64%
Centipede 62%
Bank Heist 57%
River Raid 57%
Zaxxon 54%
Amidar 43%
Alien 42%
Venture 32%
Seaquest 25%
Double Dunk 17%
Bowling 14%
Ms. Pac-Man 13%
Asteroids 7%
Frostbite 6%
Gravitar 5%
Private Eye 2%
Montezuma's Revenge 0%
DQN
Best linear learner
0
100
200
300
400
500
600
1,000
4,500%



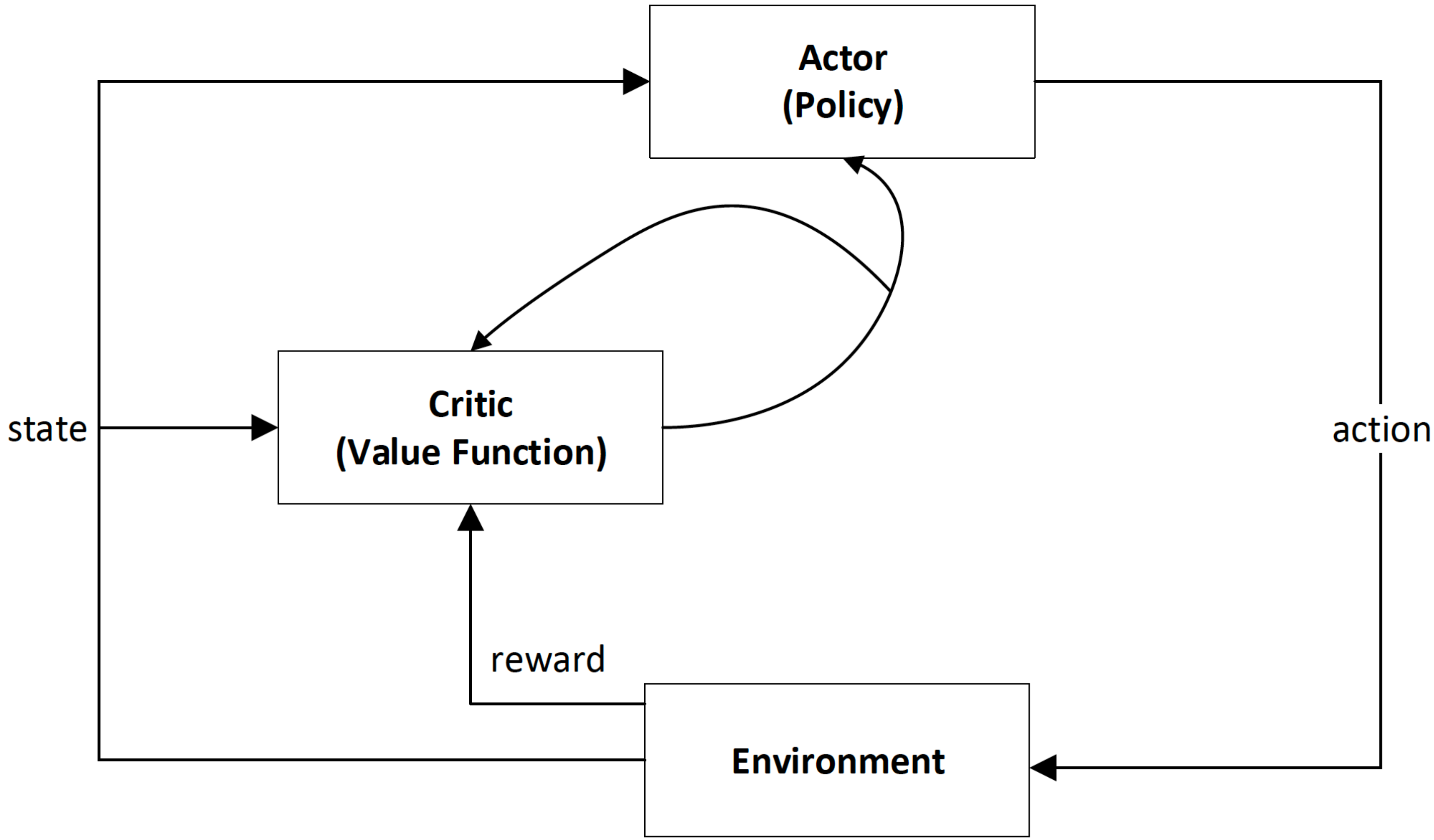




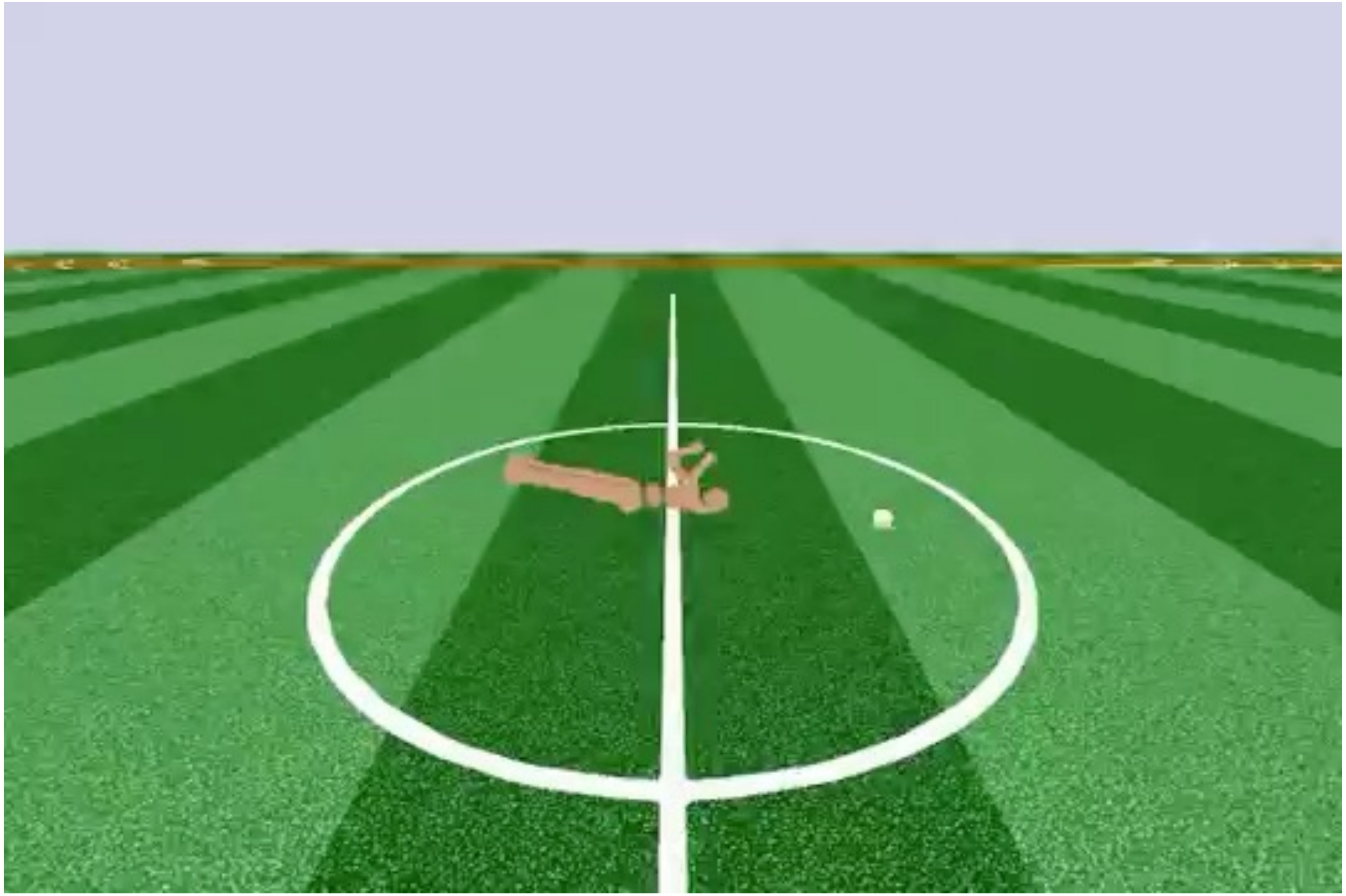
Parameterized policies

- A family of policies indexed by parameter vector $\theta \in \mathbb{R}^d$
  - Deterministic policy $a = \pi(s, \theta)$
  - Stochastic policy $\pi(a \mid s, \theta)$
- Analogous to classification or regression with input s and output a
  - Discrete action space: network outputs vector of probabilities
  - Continuous action space: network outputs mean and diagonal covariance of Gaussian

The actor-critic framework



Actor-critic algorithm in action



Q-Learning

- **Definition**: A model-free reinforcement learning algorithm that learns the value of an action in a particular state without requiring a model of the environment.
- **Objective**: To learn the optimal policy by learning the Q-values (quality of actions) that indicate the expected utility of taking an action in a given state.
- **Key Components**:
  - **Q-Value (Q(s, a))**: Represents the expected future rewards obtainable by taking action a in state s, following the optimal policy thereafter.
  - **Learning Rate (α)**: Determines to what extent newly acquired information overrides old information.
  - **Discount Factor (γ)**: Measures the importance of future rewards compared to immediate rewards.
- **Goal**: Converge to the optimal Q-values, allowing the agent to make the best action selections in any given state.

Q-Learning algorithm

1. **Initialize** the Q-values (Q(s, a)) arbitrarily for all state-action pairs.
2. For each episode:
   - 2.1. Initialize the starting state S.
   - 2.2. For each step of the episode:
     - 2.2.1. Choose an action A from state S using a policy derived from Q (e.g., ε-greedy).
     - 2.2.2. Take the action A, observe the reward R, and the next state S'.
     - 2.2.3. Update the Q-value for the state-action pair (S, A) using the equation:

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

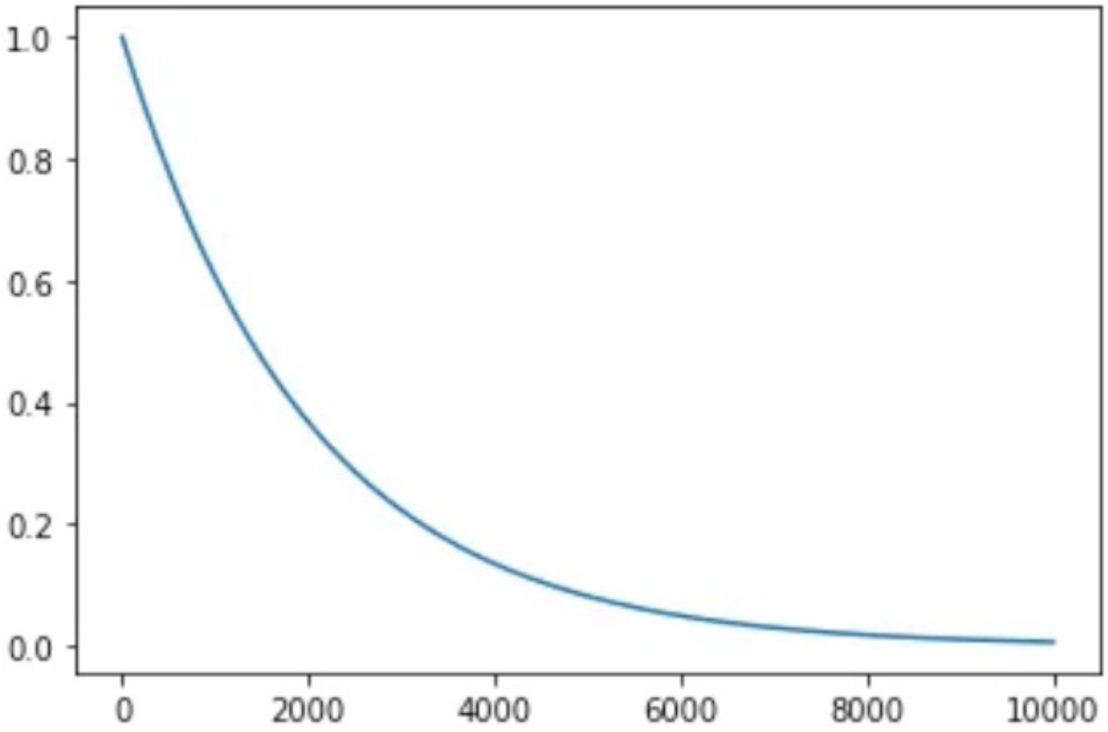
       Where:
       - $Q(S, A)$ is the current Q-value.
       - α is the learning rate.
       - R is the reward observed for moving from state S to the next state S' by taking action A.
       - γ is the discount factor.
       - $\max_a Q(S', a)$ is the estimated maximum future reward from the next state S'.
     - 2.2.4. S ← S' (update the current state).
   - 2.3. Repeat steps 2.1 to 2.4 until S is a terminal state.
3. Repeat step 2 for as many episodes as needed.

Probability for random action selection

- The exploration-exploitation trade-off

- Exponential decay formula

$$N(t) = N_0 \, e^{-\lambda t}$$



Exponetial decay graph whith N_0 = 1 and λ = 0.0005

A frozen lake example

- State space: 16
- Action space: 4
- Reward:
  - Reach goal: +1
  - Reach hole: 0
  - Reach frozen: 0
- Termination criteria:
  - The agent moves into a hole
  - The agent reaches the goal
  - The agent moved for 100 steps without reaching the goal

Example question

- Consider an MDP with three states (A,B,C) and two actions (L,R), $\gamma = 0.9$, $\alpha = 0.1$
- A new state transition sample

$$(s_t = A, a_t = L, s_{t+1} = B, r_{t+1} = 10)$$

- Current Q-table

State	Action	Q
A	L	0.5
A	R	1.0
B	L	1.5
B	R	2.0
C	L	2.5
C	R	3.0

- Question: what does the updated Q-table look like?