PROGRAMMING REINFORCEMENT LEARNING ALGORITHMS

SAN DIEGO MACHINE LEARNING

JUNE 26, 2021

HOW TO PARTICIPATE

- One discussion leader, and everyone welcome to participate
- Majority of material comes from Reinforcement Learning by Sutton and Barto
- Options to approach the content:
  - Treat this as a standalone webinar
  - Read the book first, and come with questions and discussion items
  - Use this meetup as a primer and read the chapters afterward
- Ask questions
- Give feedback. Too fast or too slow? Want to see more of something or less of something else?
- Have fun!

AGENDA

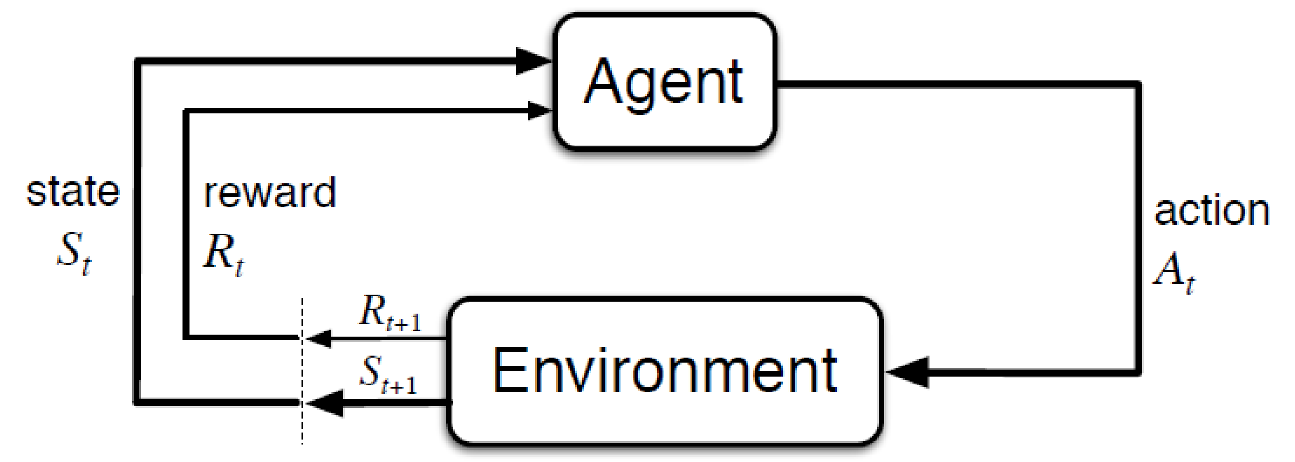
- Recap what reinforcement learning (RL) is
    - Elements and formulation as Markov decision processes (MDP)
    - Terminology and notation used in RL
    - The Bellman equations
    - Generalized policy iteration
- Programming reinforcement learning algorithms
    - Open AI Gym
    - Monte Carlo
    - SARSA
    - Q-learning

REINFORCEMENT LEARNING

- Reinforcement learning (RL) is about an *agent* learning from interacting with its uncertain *environment*
  - The agent interacts by choosing from a set of allowed *actions*
  - It gets feedback from a numeric *reward* signal
  - Goal is to maximize the *return*, which is the total rewards received
- Reinforcement learning is about exploring the environment and recording useful information for the future
- RL is sequential decision making; time is intrinsic

MARKOV DECISION PROCESSES

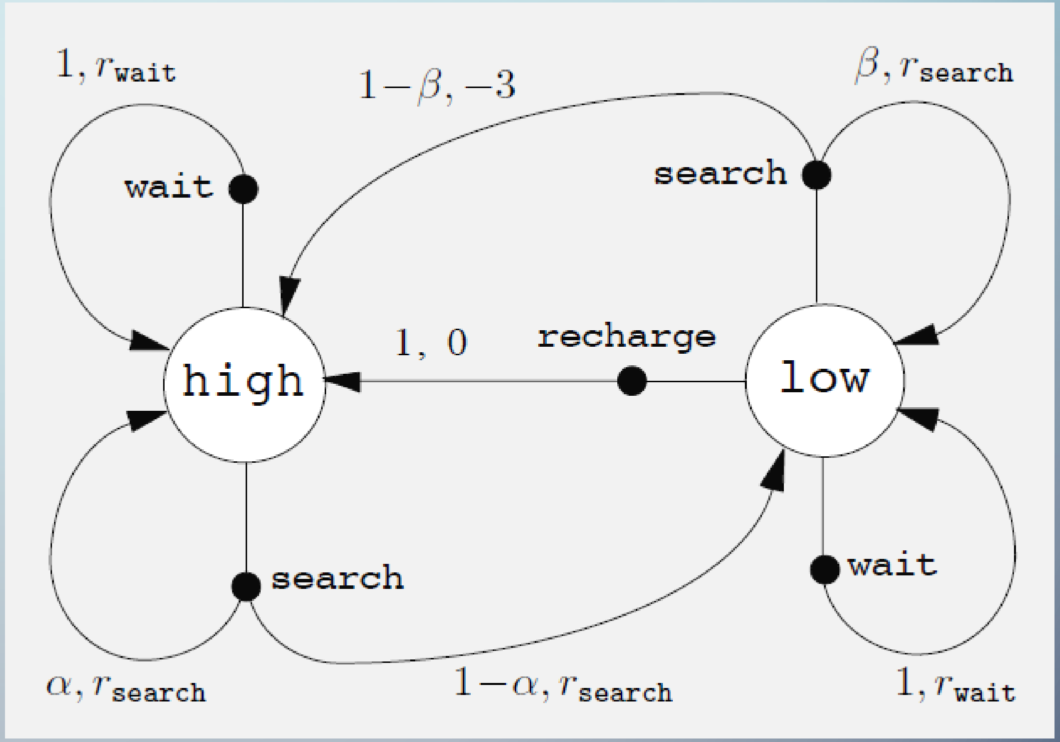
- Elements of the fully observable Markov Decision Process (MDP):
  - State - at each time step t, the environment is in some state S_t
  - Action - at each time step t, the agent chooses an action A_t
  - Reward - after taking the action, the agent is given a reward signal R_{t+1} and subsequently finds itself in a new state S_{t+1}



- In a *Markov* Decision Process, the transition at any given time t only depends on the state S_t and action chosen A_t

MDPS AS A GRAPH

- Sometimes it is easier to visualize a MDP as a directed graph
  - The states are nodes (big white circles)
  - The actions are edges leading from nodes (here with small black circles)
  - The rewards are values along directed edges that take you to a new state
- Here is the recycling robot from the book:



REINFORCEMENT LEARNING NOTATION

Letter	Used for
s	**S**tate
a	**A**ction
r	**R**eward
γ	Discount rate
G	Return – sum of all future rewards
p	Transition **p**robability
v	**V**alue function for states
q	Value function for state-action pairs
π	Policy (**π**ολιτική)
*	Optimal choices, e.g. π_*

BELLMAN EQUATION

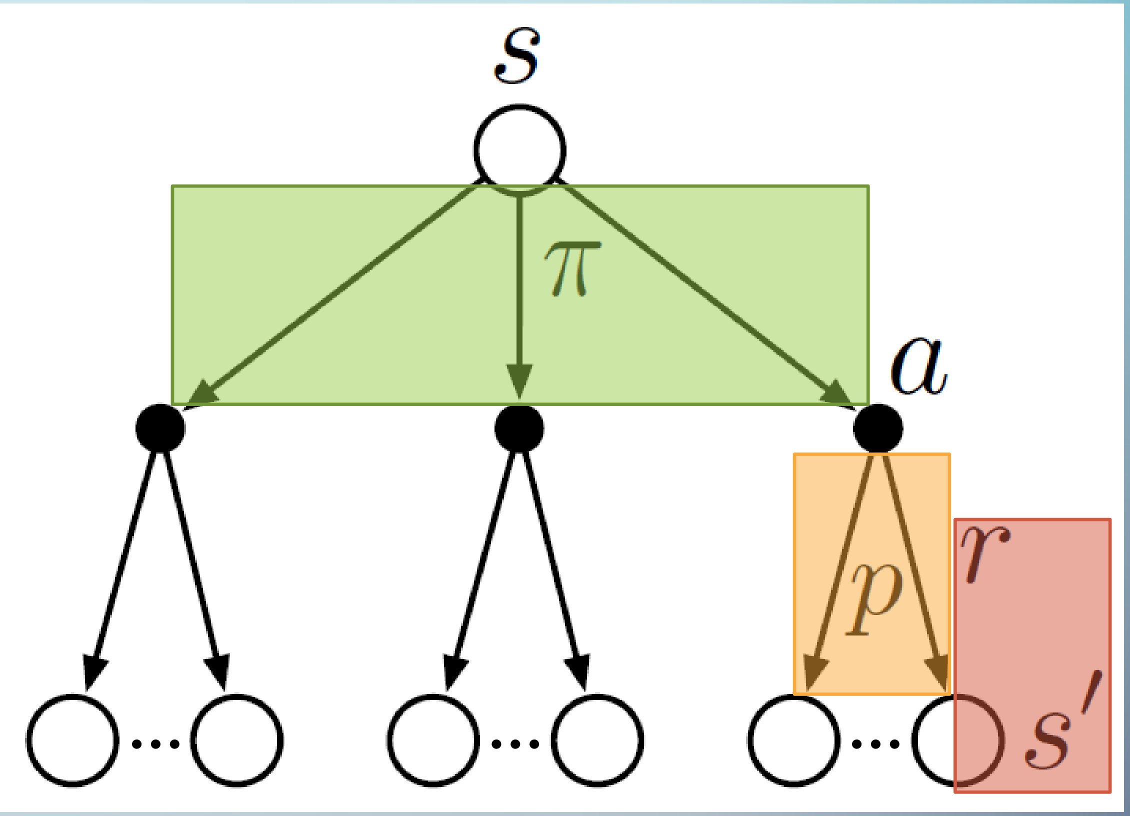
- The value function for state *s* under policy π is a sum of the rewards received and the value functions for each future state *s'* times the probability of winding up there
- Formally:

$$v_\pi(s) = \sum_a \underbrace{\pi(a,s)}_{\text{Probability you take action } a} \sum_{s',r} \underbrace{p(s',r|s,a)}_{\text{Probability you get reward } r \text{ and end in state } s'} \underbrace{[r + \gamma v_\pi(s')]}_{\text{Reward plus discounted value of new state } s'}$$

BELLMAN EQUATION VISUALIZED

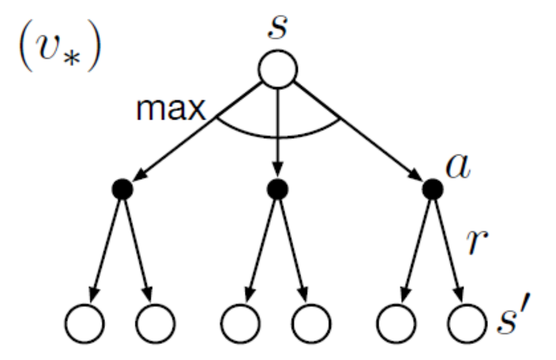
This is a *backup diagram* for $v_\pi(s)$. To compute it:

- We need to sum over each branch of π(), based on the probability of each action *a*
- And sum over of each branch of *p*(), based on probability we wind up in state *s’*
- The quantity we sum is the reward and the discounted value of possible state *s’*



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

BELLMAN OPTIMALITY EQUATIONS – V()

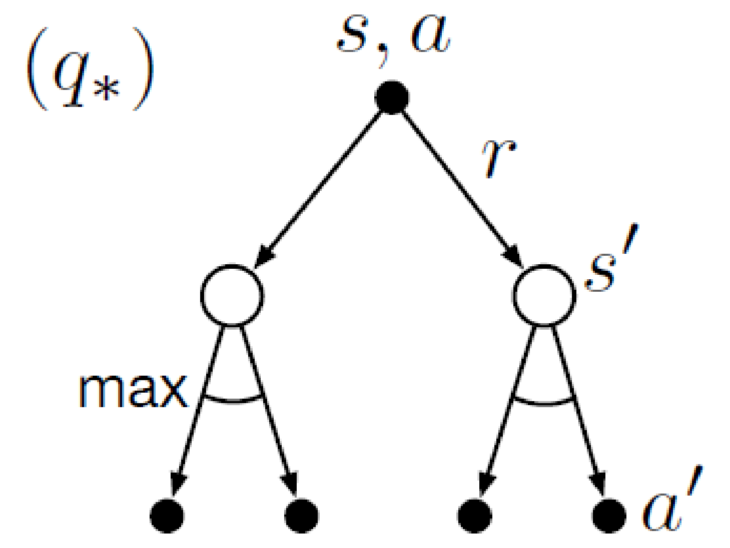


- The *Bellman optimality equation* says the optimal value for a state must be the same as the return from the best action
- We can rewrite it recursively

$$\begin{aligned}
v_*(s) &= \max_{a\in\mathcal{A}(s)} q_{\pi_*}(s,a) \\
&= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t=s, A_t=a] \\
&= \max_a \mathbb{E}_{\pi_*}[R_{t+1}+\gamma G_{t+1} \mid S_t=s, A_t=a] && \text{(by (3.9))} \\
&= \max_a \mathbb{E}[R_{t+1}+\gamma v_*(S_{t+1}) \mid S_t=s, A_t=a] && (3.18) \\
&= \max_a \sum_{s',r} p(s',r|s,a)\big[r+\gamma v_*(s')\big]. && (3.19)
\end{aligned}$$

BELLMAN OPTIMALITY EQUATIONS – Q()

- The *Bellman optimality equation* for state-action pairs is very similar.
- The optimal value for a state-action pair must be the same as the return from the reward and best next action
- It also can be written recursively



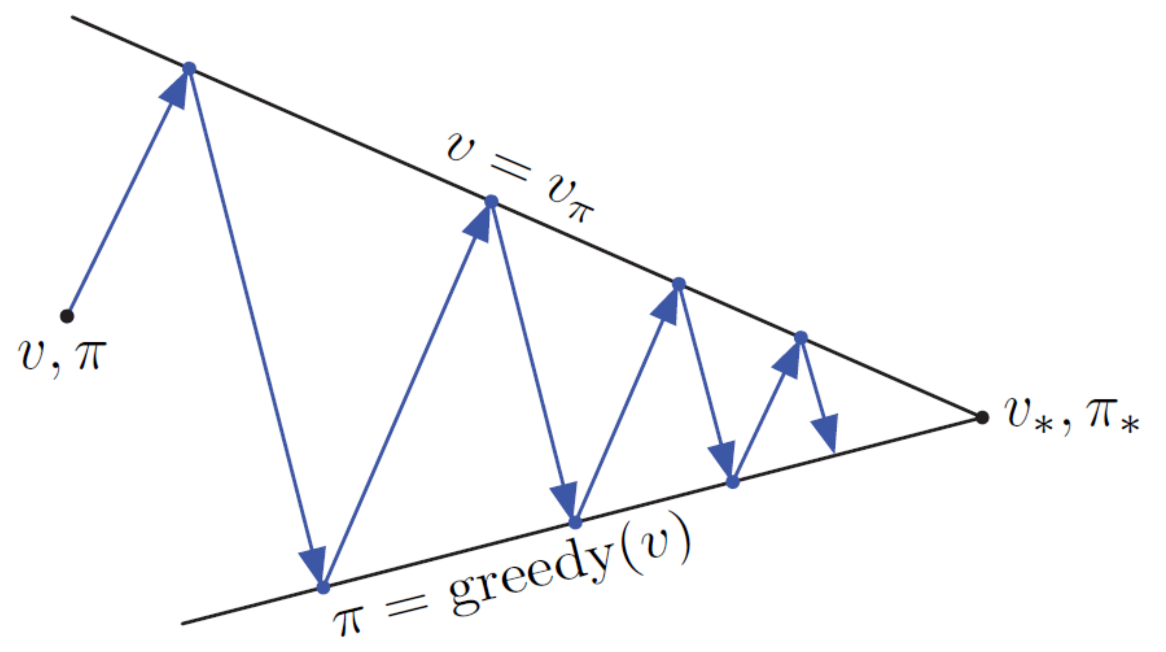
$$\begin{aligned} q_*(s,a) &= \mathbb{E}\Big[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1},a') \;\Big|\; S_t = s, A_t = a\Big] \\ &= \sum_{s',r} p(s',r|s,a)\Big[r + \gamma \max_{a'} q_*(s',a')\Big]. \qquad (3.20) \end{aligned}$$

POLICY ITERATION

- The book shows a sequence like this:

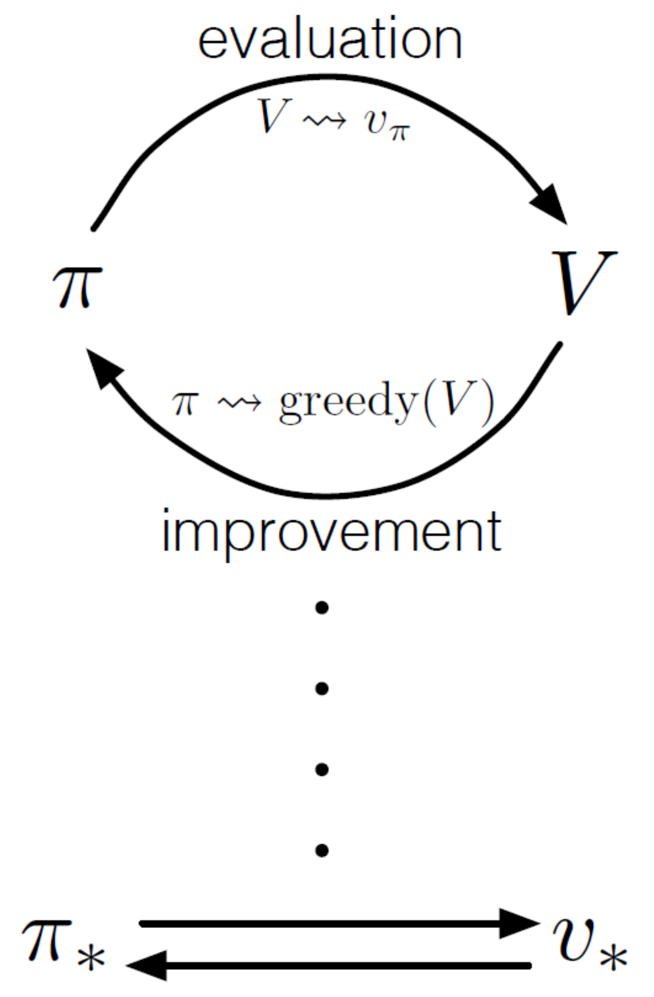
$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_1 \xrightarrow{E} \ \ldots \ \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$

- The arrows with E's are full cycles of iterative policy evaluation
- And the arrows with I's are policy improvement



GENERALIZED POLICY ITERATION

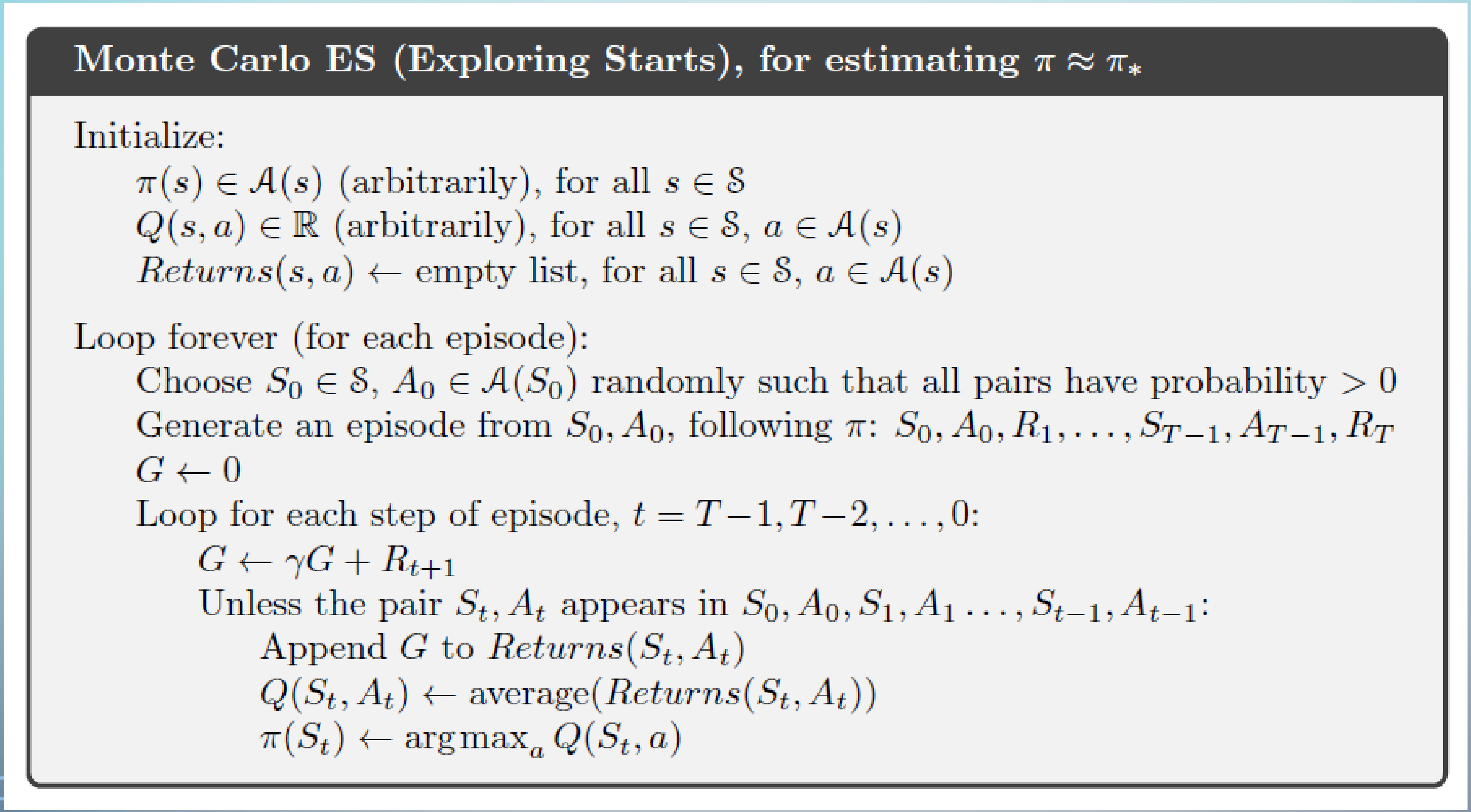
- The term *generalized policy iteration* (GPI) refers to the general idea of letting policy evaluation and policy improvement processes interact
  - Doesn't matter how fully each evaluation or improvement step runs, or if they exactly alternate



REINFORCEMENT LEARNING CONTROL

- With this foundation, there’s a lot we can tackle
  - Algorithms for learning
  - Dealing with memory and compute limitations
  - Getting models to converge quickly
- We also still have many challenges
  - Reward design – effectively communicating the real goal
  - Sparse rewards
  - Credit assignment – which actions in trajectory contributed
  - Exploration vs. exploitation

MONTE CARLO EXPLORING STARTS

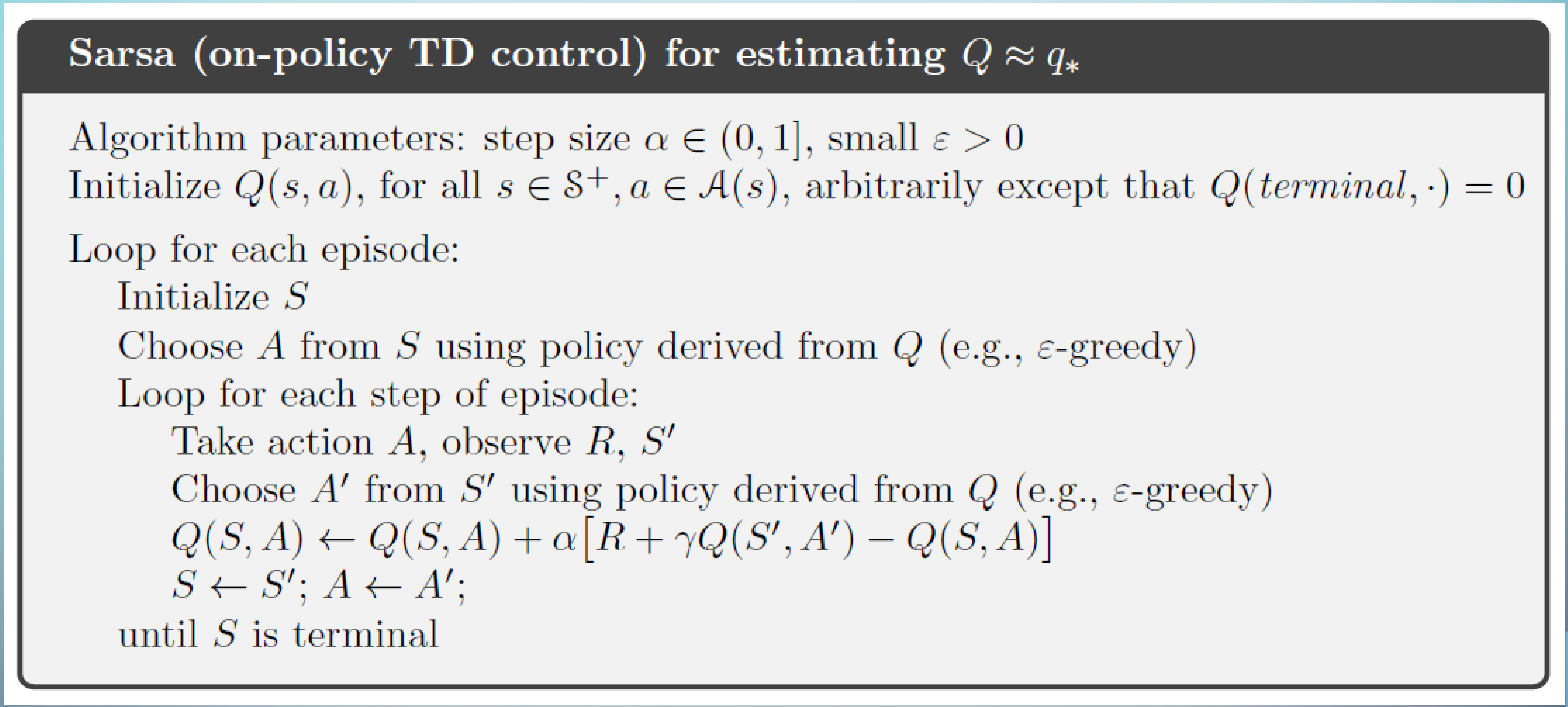


Monte Carlo ES (Exploring Starts), for estimating $\pi \approx \pi_*$

```
Initialize:
    $\pi(s) \in \mathcal{A}(s)$ (arbitrarily), for all $s \in \mathcal{S}$
    $Q(s, a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$
    $Returns(s, a) \leftarrow$ empty list, for all $s \in \mathcal{S}$, $a \in \mathcal{A}(s)$

Loop forever (for each episode):
    Choose $S_0 \in \mathcal{S}$, $A_0 \in \mathcal{A}(S_0)$ randomly such that all pairs have probability $> 0$
    Generate an episode from $S_0, A_0$, following $\pi$: $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
    Loop for each step of episode, $t = T-1, T-2, \ldots, 0$:
        $G \leftarrow \gamma G + R_{t+1}$
        Unless the pair $S_t, A_t$ appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$:
            Append $G$ to $Returns(S_t, A_t)$
            $Q(S_t, A_t) \leftarrow$ average$(Returns(S_t, A_t))$
            $\pi(S_t) \leftarrow \arg\max_a Q(S_t, a)$
```

SARSA

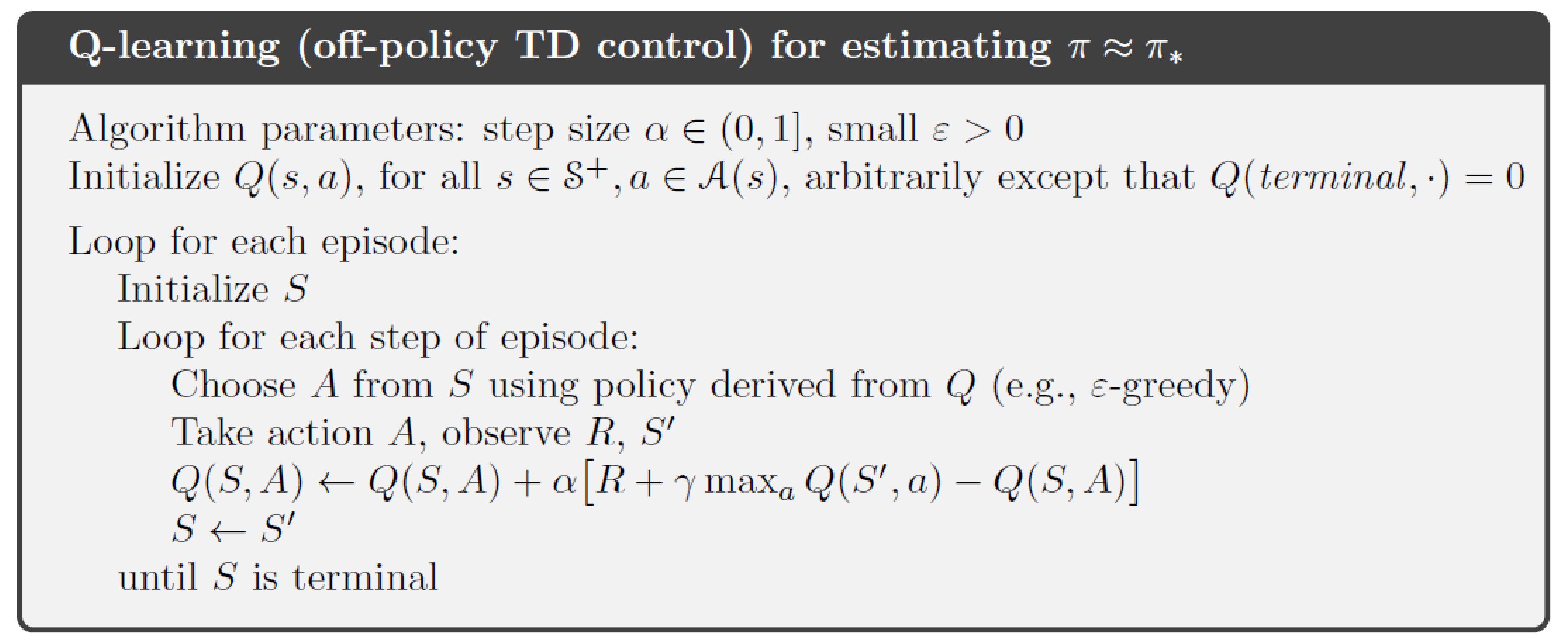


Sarsa (on-policy TD control) for estimating $Q \approx q_*$

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

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

Q-LEARNING



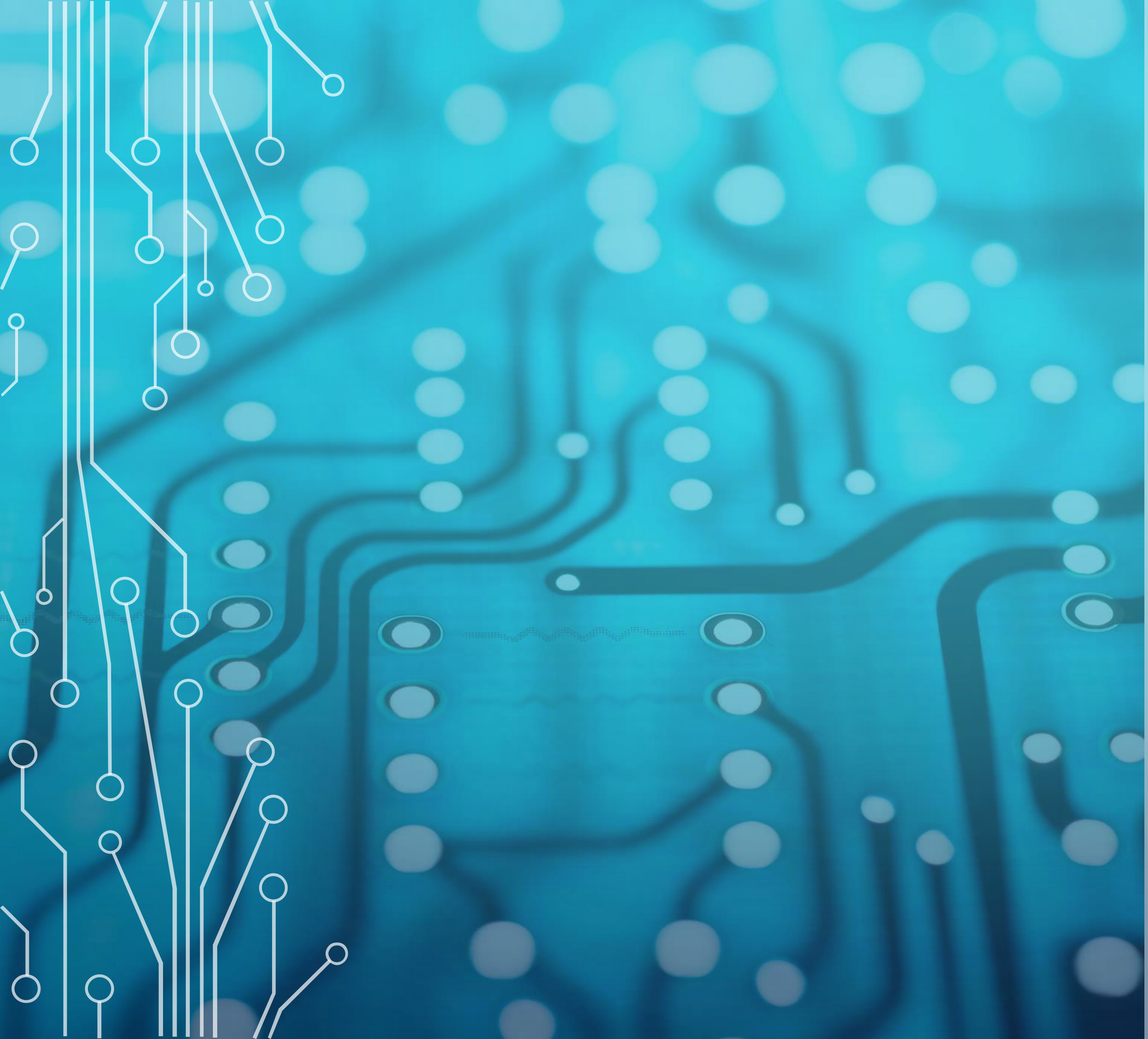
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

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

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

RECAP

- Review what reinforcement learning (RL) is
  - Elements and formulation as Markov decision processes (MDP)
  - Terminology and notation used in RL
  - The Bellman equations
  - Generalized policy iteration
- Programming reinforcement learning algorithms
  - Open AI Gym
  - Monte Carlo
  - SARSA
  - Q-learning



QUESTIONS

&

DISCUSSION

NEXT SESSION

- Next week, Sat. July 3, there will be no book club
- The following week, on Sat. July 10, we will discuss planning and learning. The core material will be from chapter 8 of Sutton & Barto