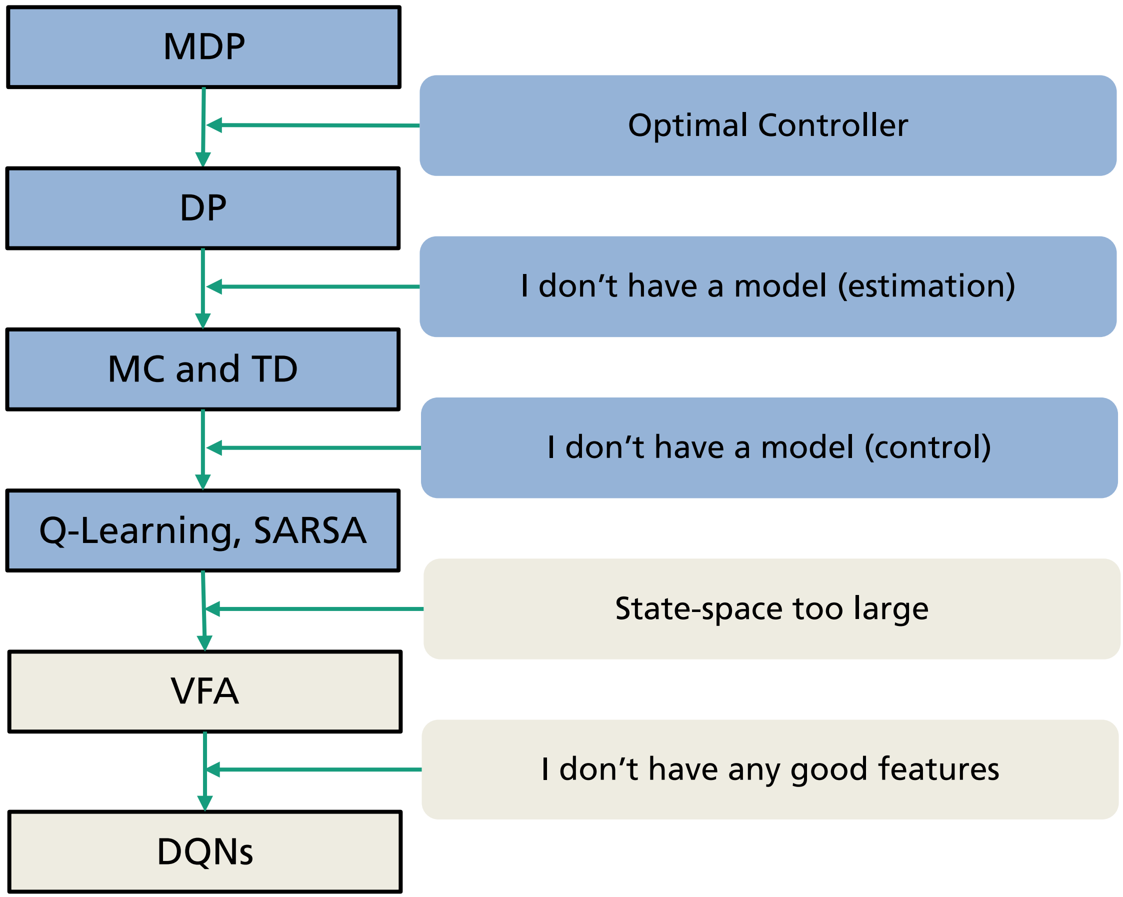
Model-free Control

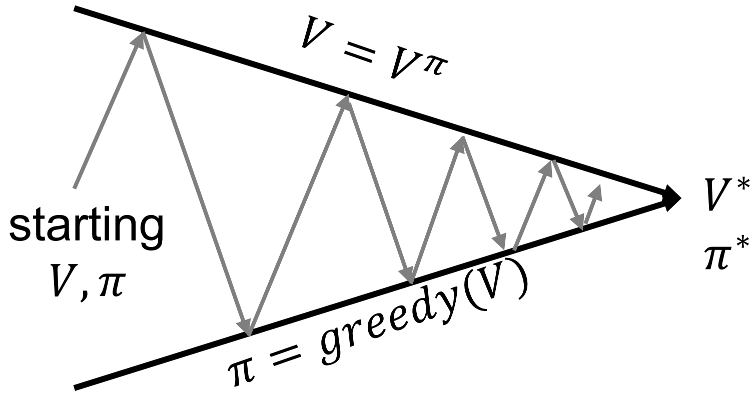
Christopher Mutschler

Overview


MDP
Optimal Controller
DP
I don’t have a model (estimation)
MC and TD
I don’t have a model (control)
Q-Learning, SARSA
State-space too large
VFA
I don’t have any good features
DQNs

Q-Learning and SARSA Algorithms

- The (model-free) control problem:
  - **Given** experience samples $s(s, a, r, s')$
  - **Learn** a close-to optimal policy π
- Simple idea:
  - If we have calculated the value function for a given policy π (e.g., from MC/TD policy evaluation from last week), we can use it for deriving a better policy π' through greedy policy improvement over $V(s)$

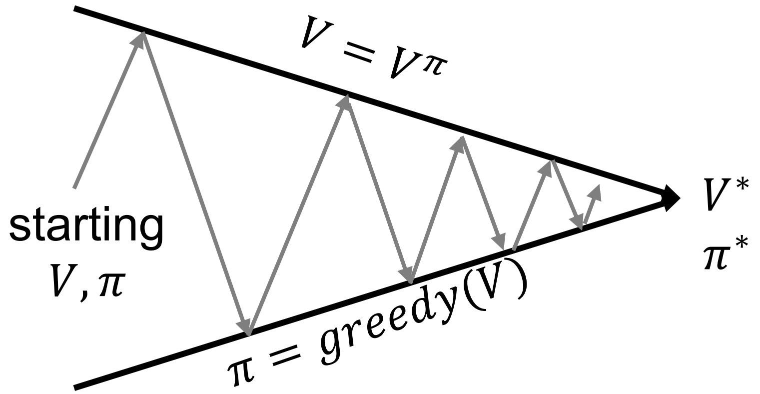


Policy Evaluation: Estimate $V = v_{\pi}$
e.g., Iterative Policy Estimation

Policy Improvement: Generate $\pi' \geq \pi$
e.g., Greedy Policy Improvement

Q-Learning and SARSA Algorithms

- The (model-free) control problem:
  - **Given** experience samples $s(s, a, r, s')$
  - **Learn** a close-to optimal policy π
- Simple idea:
  - If we have calculated the value function for a given policy π (e.g., from MC/TD policy evaluation from last week), we can use it for deriving a better policy π' through greedy policy improvement over $V(s)$



Policy Evaluation: Estimate $V = v_{\pi}$
e.g., Monte Carlo Policy Estimation

Policy Improvement: Generate $\pi' \geq \pi$
e.g., Greedy Policy Improvement

Q-Learning and SARSA Algorithms

- The (model-free) control problem:
  - **Given** experience samples $s(s, a, r, s')$
  - **Learn** a close-to optimal policy π
- Simple idea:
  - If we have calculated the value function for a given policy π (e.g., from MC/TD policy evaluation from last week), we can use it for deriving a better policy π' through greedy policy improvement over $V(s)$

$$\pi'(\mathrm{s}) = \arg\max_{a \in A} \left\{ R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\pi}(s') \right\}, s \in S, V^{\pi'}(s) \geq V^{\pi}(s)$$

Requires a model!

  - Greedy policy improvement over $Q(s, a)$ is model-free

$$\pi'(\mathrm{s}) = \arg\max_{a \in A} Q(s,a)$$

Recap: Dynamic Programming

- How do we find optimal controllers for given (known) MDPs?
- Unfortunately, we need some definitions:
  - state-value function V for policy π

$$s \xrightarrow{\pi(s),\,R_0} S_1 \xrightarrow{\pi(S_1),\,R_1} S_2 \xrightarrow{\pi(S_2),\,R_2} S_3 \ldots\ S_{h-1} \xrightarrow{\pi(S_{h-1}),\,R_{h-1}} S_h$$

$$V^\pi(s) \triangleq Q^\pi(s,\pi(s)) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s\right].$$

  - state-action-value function Q for policy π

$$s \xrightarrow{a,\,R_0} S_1 \xrightarrow{\pi(S_1),\,R_1} S_2 \xrightarrow{\pi(S_2),\,R_2} S_3 \ldots\ S_{h-1} \xrightarrow{\pi(S_{h-1}),\,R_{h-1}} S_h$$

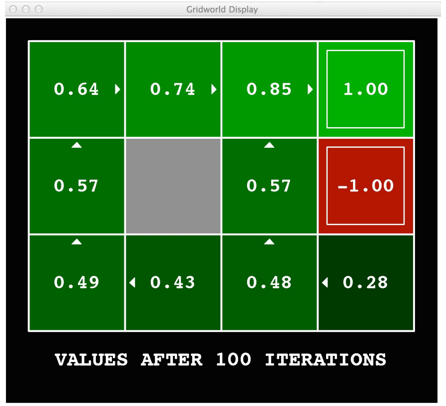
$$Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s, A_0 = a\right].$$

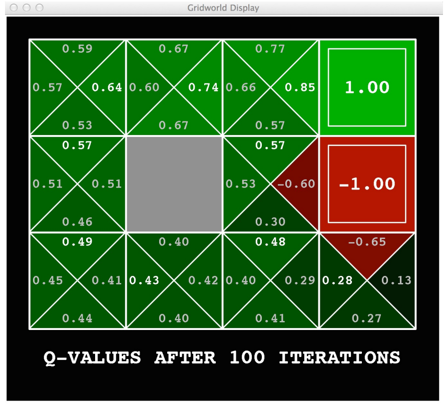
Recap: Dynamic Programming

- State-action-value function Q for policy π

$$s \xrightarrow{a, r_0} s_1 \xrightarrow{\pi(S_1), r_1} s_2 \xrightarrow{\pi(S_2), r_2} s_3 \ldots \; s_{h-1} \xrightarrow{\pi(s_{h-1}), r_{h-1}} s_h$$

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a\right]$$





Recap: Dynamic Programming

- How do we find optimal controllers for given (known) MDPs?
- Unfortunately, we need some definitions:
  - Bellman Equation for Q, given policy π

$$s \xrightarrow{a, \mathcal{R}(s,a)} s' \xrightarrow{\pi(s'), R_1} S_2 \xrightarrow{\pi(S_2), R_2} S_3 \dots\ S_{h-1} \xrightarrow{\pi(S_{h-1}), R_{h-1}} S_h$$

$$Q^\pi(s,a) = \underbrace{\mathcal{R}(s,a)}_{\text{first step}} + \underbrace{\gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, Q^\pi(s', \pi(s'))}_{\text{subsequent steps}}$$

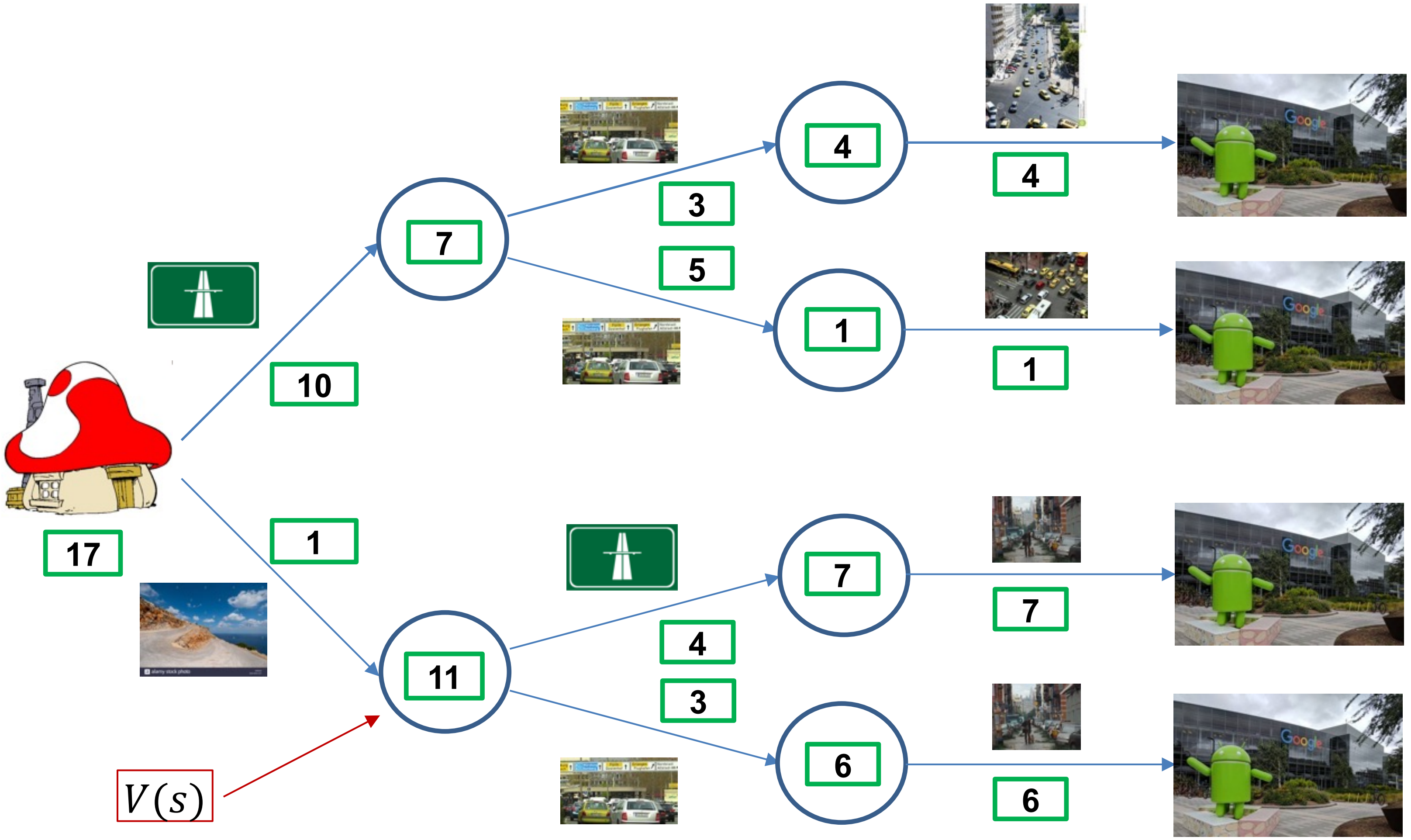
Recap: Dynamic Programming

- How do we find optimal controllers for given (known) MDPs?
- Unfortunately, we need some definitions:
  - Bellman Optimality Equation for Q

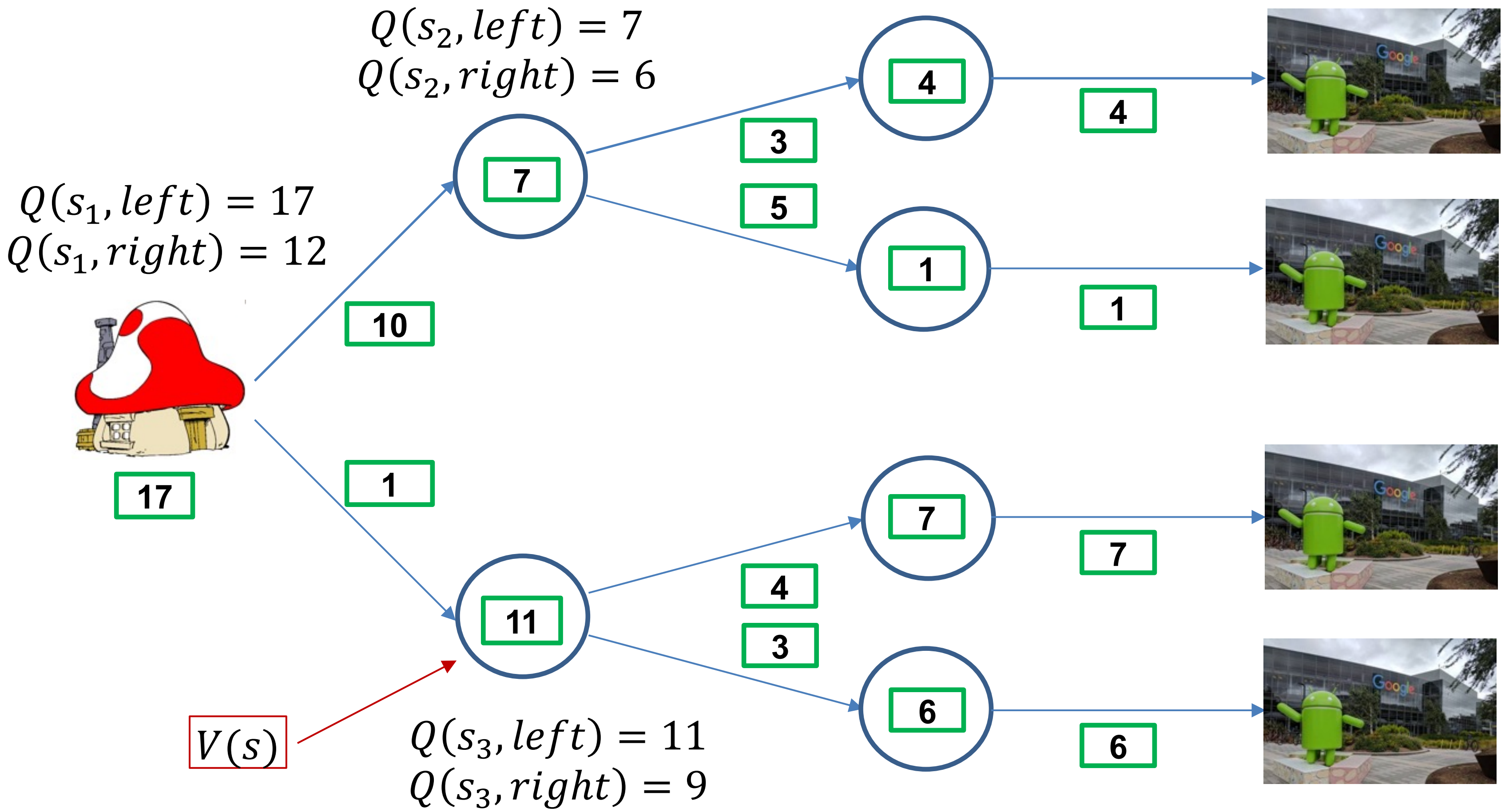
$$s \xrightarrow{a,\,\mathcal{R}(s,a)} s' \xrightarrow{\pi^*(s'),\,R_1} S_2 \xrightarrow{\pi^*(S_2),\,R_2} S_3 \dots\ S_{h-1} \xrightarrow{\pi^*(S_{h-1}),\,R_{h-1}} S_h$$

$$Q^{\pi^*}(s,a) = \underbrace{\mathcal{R}(s,a)}_{\text{first step}} + \underbrace{\gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a) \max_{a' \in \mathcal{A}} Q^{\pi^*}(s',a')}_{\text{subsequent steps}}$$

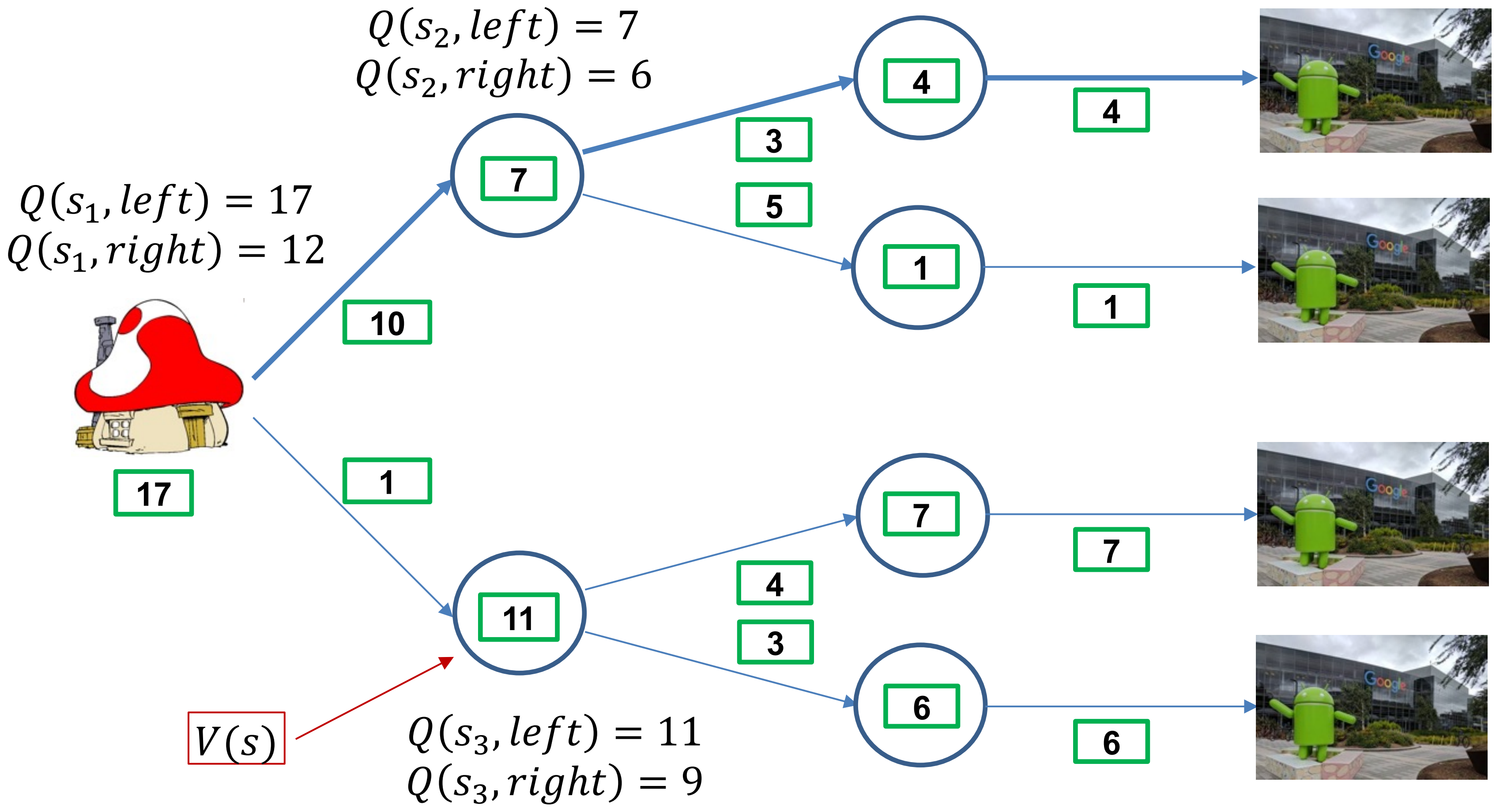
Recap: Dynamic Programming



Recap: Dynamic Programming



Recap: Dynamic Programming



Recap: Dynamic Programming

- How do we find optimal controllers for given (known) MDPs?
- Unfortunately, we need some definitions:
  - Greedy Policy Improvement over Q

$$\pi'(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q^{\pi}(s,a)$$

$$\forall s \in \mathcal{S}, \qquad Q^{\pi'}\big(s, \pi'(s)\big) \geq Q^{\pi}\big(s, \pi(s)\big)$$

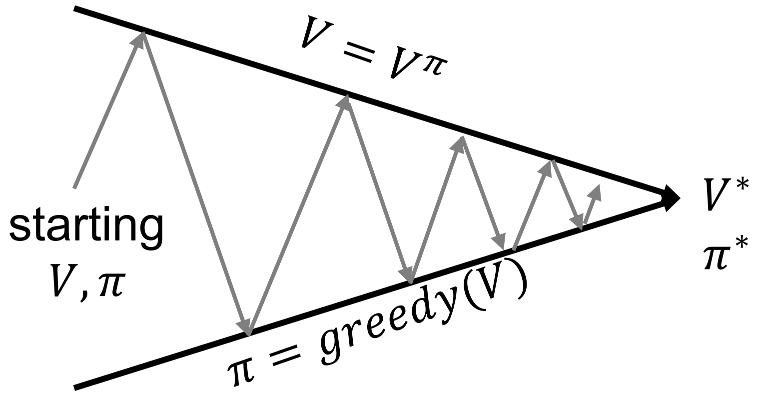
  - Greedy Policy Improvement over V

$$\pi'(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left\{ \mathcal{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}(s'|s,a)\, V^{\pi}(s') \right\}$$

$$\forall s \in \mathcal{S}, \qquad V^{\pi'}(s') \geq V^{\pi}(s')$$

Q-Learning and SARSA Algorithms

- The (model-free) control problem:
  - **Given** experience samples $s(s, a, r, s')$
  - **Learn** a close-to optimal policy π
- Simple idea:
  - If we have calculated the value function for a given policy π (e.g., from MC/TD policy evaluation from last week), we can use it for deriving a better policy π' through greedy policy improvement over $V(s)$

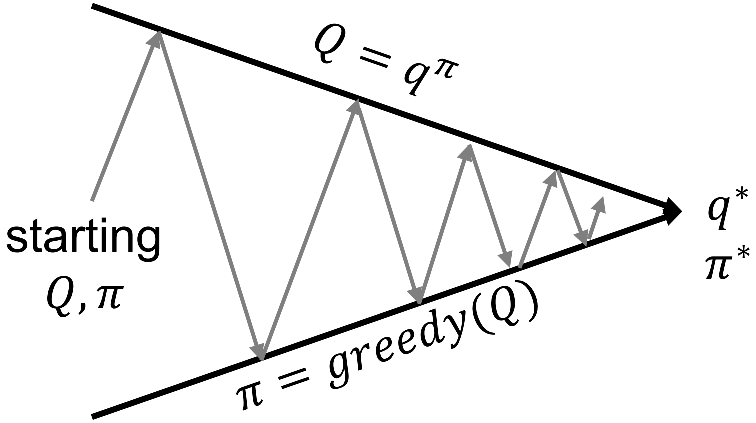


Policy Evaluation: Estimate $V = v_{\pi}$
e.g., Monte Carlo Policy Estimation

Policy Improvement: Generate $\pi' \geq \pi$
e.g., Greedy Policy Improvement

Q-Learning and SARSA Algorithms

- The (model-free) control problem:
  - **Given** experience samples $s(s, a, r, s')$
  - **Learn** a close-to optimal policy π
- Simple idea:
  - If we have calculated the value function for a given policy π (e.g., from MC/TD policy evaluation from last week), we can use it for deriving a better policy π' through greedy policy improvement over $Q(s)$



Policy Evaluation: Estimate $Q = q_\pi$
e.g., Monte Carlo Policy Evaluation

Policy Improvement: Generate $\pi' \geq \pi$
e.g., Greedy Policy Improvement over Q

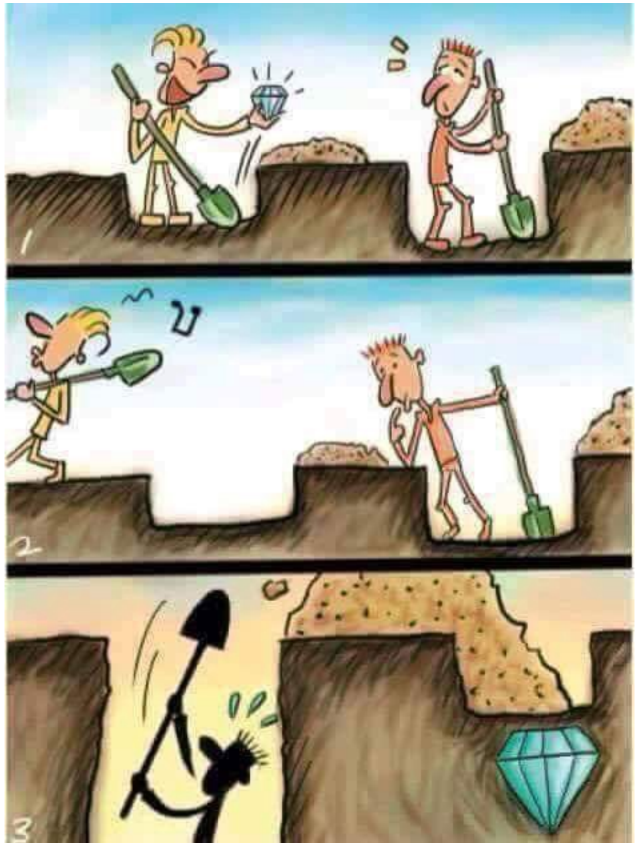
Q-Learning and SARSA Algorithms

- Idea: your agent should not be afraid of trying something new ☺

thanks to @bomboclaater

Q-Learning and SARSA Algorithms

- Exploration vs. Exploitation



https://medium.com/deep-math-machine-learning-ai/ch-12-1-model-free-reinforcement-learning-algorithms-monte-carlo-sarsa-q-learning-65267cb8d1b4

Q-Learning and SARSA Algorithms

- Exploration vs. Exploitation

"Behind one door is tenure - behind the other is flipping burgers at McDonald's."

- There are two doors in front of you.
- You open the left door and get reward 0
  $V(left) = 0$
- You open the right door and get reward $+1$
  $V(right) = +1$
- You open the right door and get reward $+3$
  $V(right) = +2$
- You open the right door and get reward $+2$
  $V(right) = +2$

$\vdots$

- Are you sure you've chosen the best door?

David Silver: Lectures on Reinforcement Learning. UCL Course on RL. 2015.

Q-Learning and SARSA Algorithms

- Greedy policy improvement problems:
  - Executes the best action according to the estimated value function
  - Non-optimal initial choices may disorient exploration
  - Areas of the state space remain unexplored!
- Solution: ε-greedy exploration
  - Seems simplistic but very difficult to do better in practice!
  - Either take the best action or explore the action space
  - ε = "probability of exploration"
  - It can be proven that any ε-greedy policy π' is an improvement over the ε-greedy policy π, $v_{\pi'}(s) \geq v_{\pi}(s)$

See "David Silver: Lectures on Reinforcement Learning. UCL Course on RL. 2015." Lecture 5 on Control, slide 12 for a proof on this

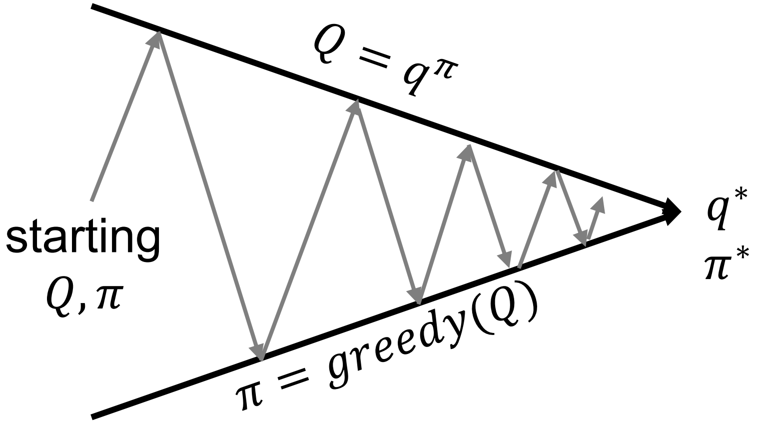
- GLIE MC Control:

$$N(S_t, A_t) \leftarrow N(S_t, A_t) + 1 \quad Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)} \left(G_t - Q(S_t, A_t)\right)$$

  - Improve policy based on new action-value function ($\varepsilon \leftarrow \frac{1}{k}, \pi \leftarrow \varepsilon$-greedy$(Q)$)
  - Converges to the optimal action-value function $Q(S_t, A_t) \rightarrow q_*(s, a)$

Q-Learning and SARSA Algorithms

- The (model-free) control problem:
  - **Given** experience samples $s(s, a, r, s')$
  - **Learn** a close-to optimal policy π
- Simple idea:
  - If we have calculated the value function for a given policy π (e.g., from MC/TD policy evaluation from last week), we can use it for deriving a better policy π' through greedy policy improvement over $Q(s)$

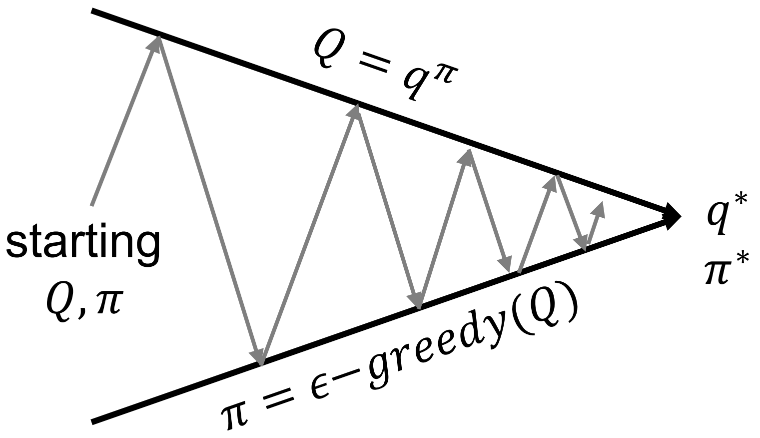


Policy Evaluation: Estimate $Q = q_\pi$
e.g., Monte Carlo Policy Evaluation

Policy Improvement: Generate $\pi' \geq \pi$
e.g., Greedy Policy Improvement over Q

Q-Learning and SARSA Algorithms

- Greedy policy improvement problems:
  - Executes the best action according to the estimated value function
  - Non-optimal initial choices may disorient exploration
  - Areas of the state space remain unexplored!
- Solution: ε-greedy exploration

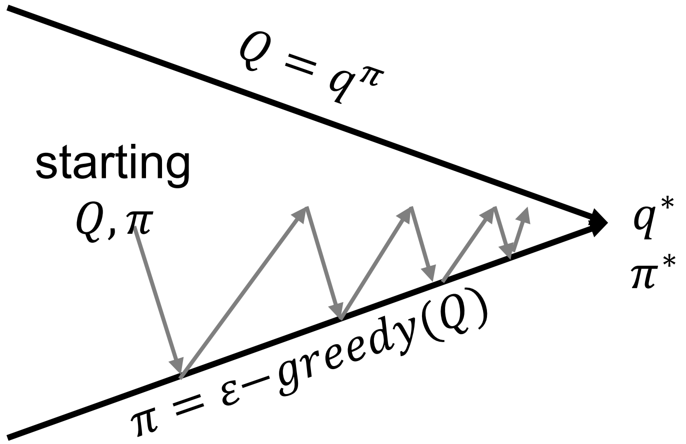


Policy Evaluation: Estimate $Q = q_\pi$
e.g., Monte Carlo Policy Evaluation

Policy Improvement: Generate $\pi' \geq \pi$
e.g., ϵ-greedy Policy Improvement over Q

Q-Learning and SARSA Algorithms

- Greedy policy improvement problems:
  - Executes the best action according to the estimated value function
  - Non-optimal initial choices may disorient exploration
  - Areas of the state space remain unexplored!
- Solution: ε-greedy exploration



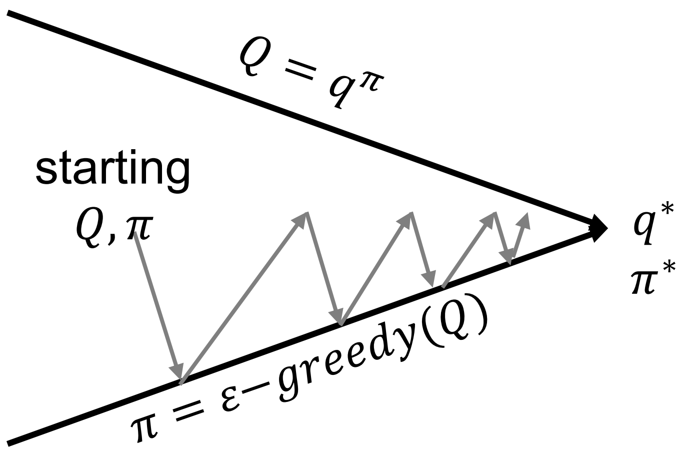
Every episode:

Policy Evaluation: Estimate $Q \approx q_{\pi}$
e.g., Monte Carlo Policy Evaluation

Policy Improvement: Generate $\pi' \geq \pi$
e.g., ϵ-greedy Policy Improvement over Q

Q-Learning and SARSA Algorithms

- Greedy policy improvement problems:
  - Executes the best action according to the estimated value function
  - Non-optimal initial choices may disorient exploration
  - Areas of the state space remain unexplored!
- Solution: ε-greedy exploration



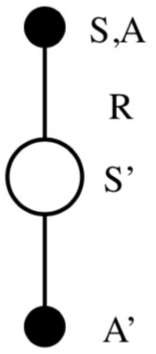
Every time step:

Policy Evaluation: Estimate $Q \approx q_\pi$
e.g., SARSA

Policy Improvement: Generate $\pi' \geq \pi$
e.g., ϵ-greedy Policy Improvement over Q

Q-Learning and SARSA Algorithms

- SARSA algorithm (on-policy control)
  - Apply TD to $Q(s, a)$
  - Use ε-greedy policy improvement
  - Update at every time-step



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

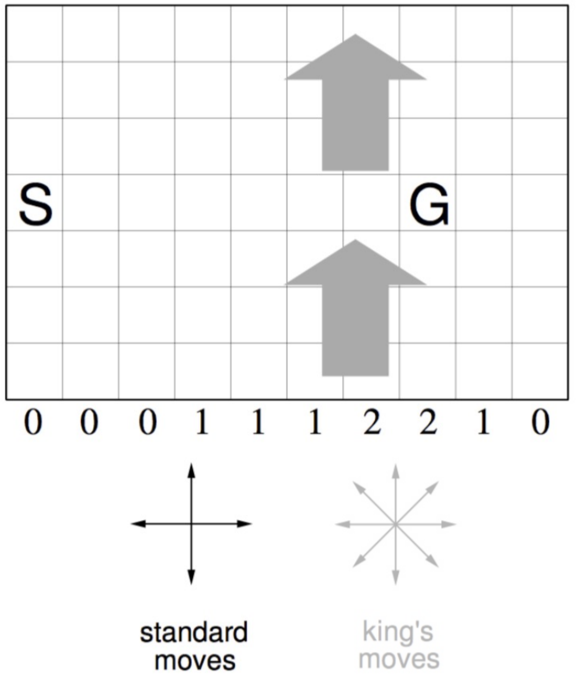
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

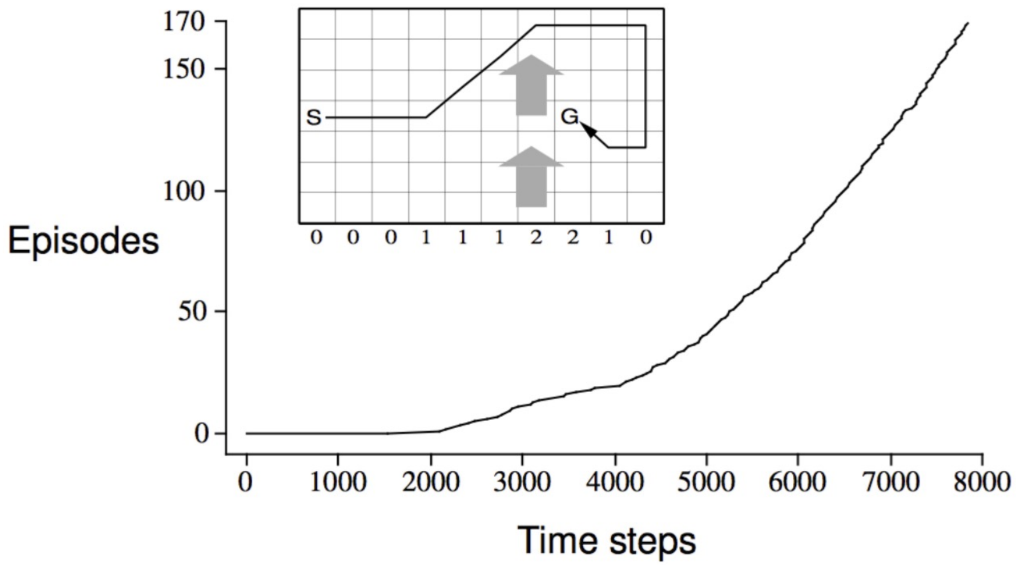
Loop for each episode:
  Initialize S
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ε-greedy)
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma Q(S', A') -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

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Q-Learning and SARSA Algorithms

- SARSA algorithm (on-policy control)
- Example: Windy Gridworld
  - Reward: -1 per time step; no discount





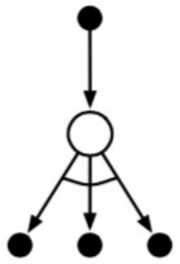
Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Q-Learning and SARSA Algorithms

- SARSA algorithm (on-policy control)
  + Processes each sample immediately
  + Minimal update cost per sample
  - Poses constraints on sample collection (on-policy)
  - Requires a huge number of samples
  - Requires careful schedule for the learning rate
  - Makes minimal use of each sample
  - The ordering of samples influences the outcome
  - Exhibits instabilities under approximate representations
  - Requires careful handling on the policy greediness

Q-Learning and SARSA Algorithms

- Q-Learning algorithm (off-policy control)
  - Evaluate one policy while following another
  - Can re-use experience gathered from old policies



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

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

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

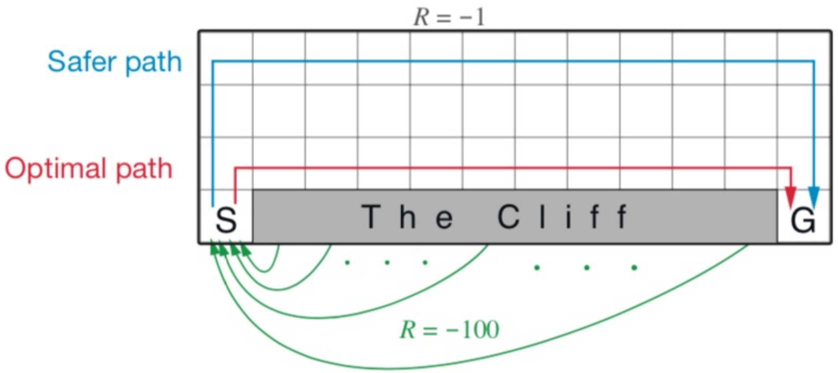
Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

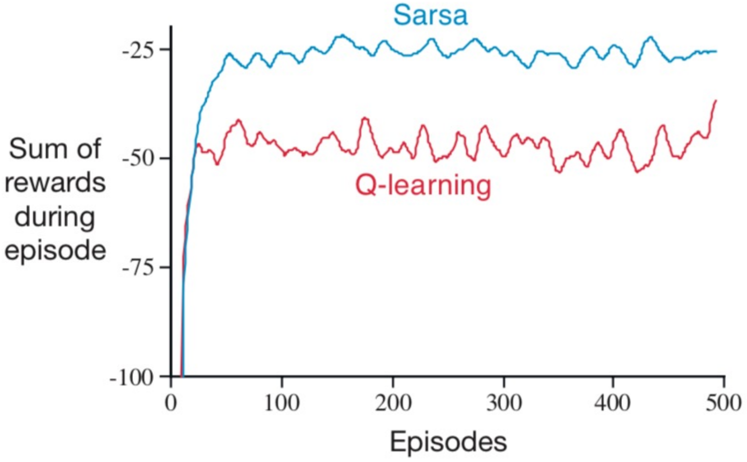
Q-Learning and SARSA Algorithms

- Q-Learning algorithm (off-policy control)
  + Processes each sample immediately
  + Minimal update cost per sample
  + Poses no constraints on sample collection (off-policy)
  - Requires a huge number of samples
  - Requires careful schedule for the learning rate
  - Makes minimal use of each sample
  - The ordering of samples influences the outcome
  - Exhibits instabilities under approximate representations

Q-Learning and SARSA Algorithms

- Example: Cliff Walking
  - Every transition has reward of -1, falling off the cliff gives a reward of -100 and ends the episode
  - No discounting
  - Assume we use ε-greedy (0.1) for SARSA and Q-Learning, no decay.





- SARSA chooses the safe route, because SARSA incorporates the current policy (ε-greedy)
- Q-Learning chooses the optimal path (and falls of the cliff using the ε-greedy)

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

Model-free Control: Remarks

- We only studied TD-based control in this video.
- Note: there is also an MC-based way to do control: (see Sutton and Barto's RL book pp. 97 – 103)

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

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

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

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

References

Books:

- Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.
- Bellman, R.E. 1957. Dynamic Programming. Princeton University Press.

Lectures:

- UC Berkeley CS188 Intro to AI. http://ai.berkeley.edu/lecture_slides.html
- UCL Course on RL. http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
- Advanced Deep Learning and Reinforcement Learning (UCL + DeepMind). http://www.cs.ucl.ac.uk/current_students/syllabus/compgi/compgi22_advanced_deep_learning_and_reinforcement_learning
- Pieter Abbeel: CS 188 Introduction to Artificial Intelligence. Fall 2018

Blogs etc.:

- https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_td.html