CPSC 4420/6420

Artificial Intelligence

08 – Reinforcement Learning I

September 15, 2020

Announcements

- Project 2 is due on 9/24
- Quiz 3 is due this Thursday

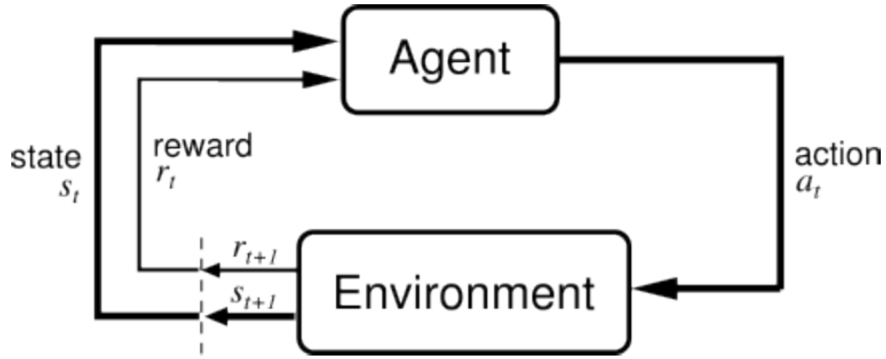
Lecture 8

Slide Credits:
Stuart Russell
Pieter Abbeel
Dan Klein
Ioannis Karamouzas

Defining RL problems

- Still assume a MDP:
  - A set of states S
  - A set of actions A per state
  - A transition model $P(s'|\, s, a)$
  - A reward function $R(s, a, s')$
- Still looking for a policy π(s)
- New twist: don't know *P* and *R*
  - We don't know which states are good or what the actions do
  - Must actually try out actions and states to learn

Reinforcement learning framework

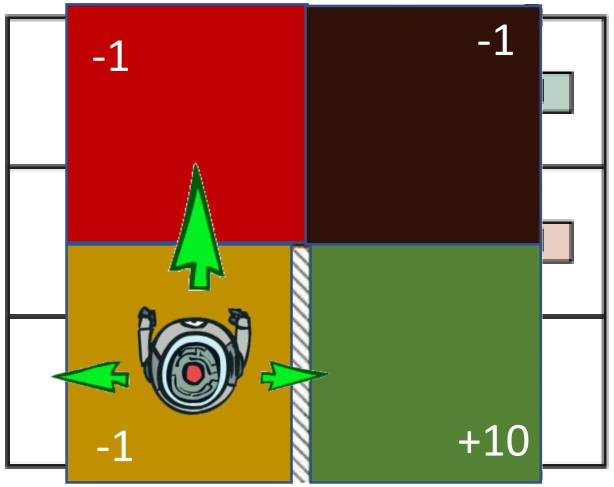


Basic idea:

- Receive feedback in the form of rewards
- Must learn to act so as to maximize expected rewards
- All learning is based on observed samples of outcomes!

Figure source: [Sutton & Barto, Reinforcement Learning: An Introduction]

Simple gridworld



Action Space

$$\gamma = 1$$

Episode

$$s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_T$$

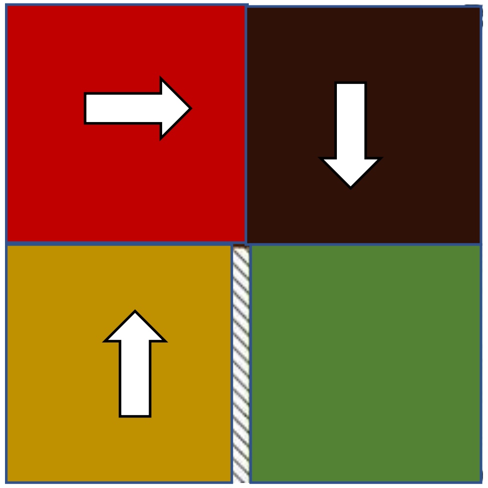
Model-Based Learning

Model-Based Monte Carlo

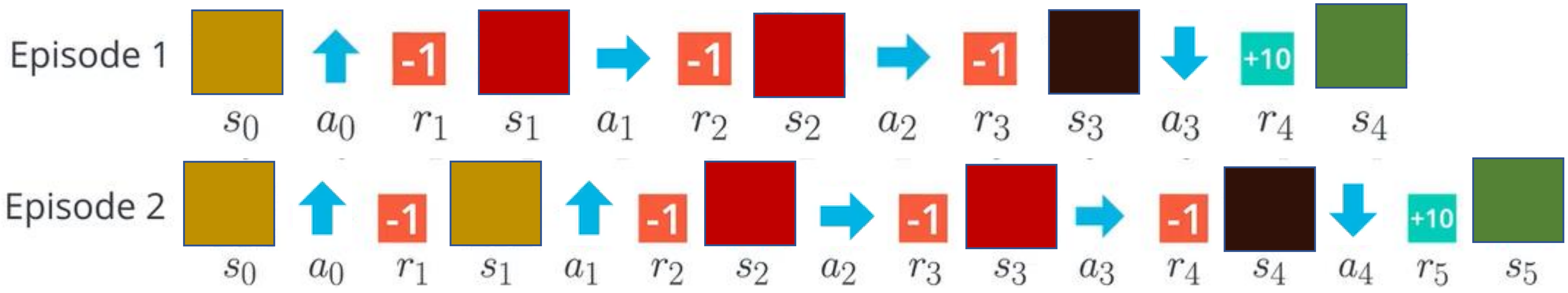
- Step 1: Learn empirical MDP using Monte Carlo simulation
  - Episodic data: $s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_T$
  - Estimate transitions and rewards
    - $\hat{P}(s' \mid s, a) = \frac{\text{\# times } (s,a,s') \text{ occurs}}{\text{\# times } (s,a) \text{ occurs}}$
    - Discover each $\hat{R}(s, a, s')$ when we experience (s, a, s')
  - Estimates converge to true values under certain conditions
- Step 2: Solve the learned MDP
  - For example, compute policy using value iteration

Example

Input Policy π

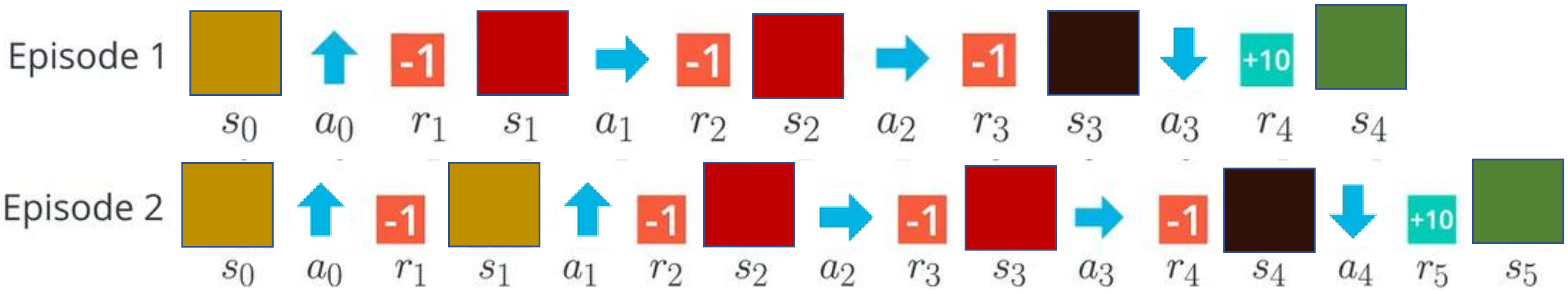


- Data (following above policy)



Example

- Data (following above policy)



- Learned Model

Model-Free Learning

Example

Compute expected grade of quiz 1

Known P(G)

$$E[\mathrm{G}] = \sum_{\mathrm{g}} P(\mathrm{g}) \cdot \mathrm{g}$$ = 0.2 x 8 + 0.3 x 4 +…

Without P(G), instead collect samples [g_1, g_2, … g_N]

Unknown P(G): "Model Based"

$$\hat{P}(\mathrm{g}) = \frac{\mathrm{num}(\mathrm{g})}{N}$$

$$E[\mathrm{G}] \approx \sum_{\mathrm{g}} \hat{P}(\mathrm{g}) \cdot \mathrm{g}$$

Unknown P(G): "Model Free"

$$E[\mathrm{G}] \approx \frac{1}{N} \sum_{i} \mathrm{g}_i$$

Passive Reinforcement Learning

Passive RL

- Simplified task
  - Input: a fixed policy $\pi(s)$
  - You don't know the transitions $P(s'|s,a)$
  - You don't know the rewards $R(s,a,s')$
  - Goal: learn the values for each state under π

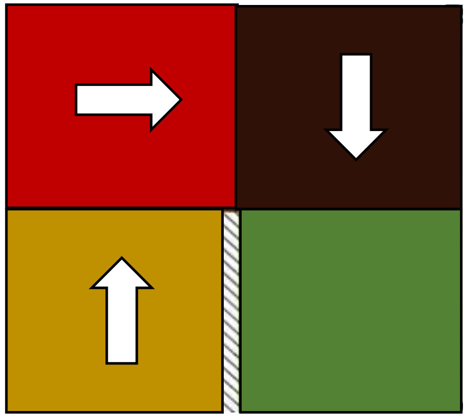
- Learner is "along for the ride"
- No choice about what actions to take
- Just execute the policy and learn from experience

Direct evaluation (model-free Monte Carlo)

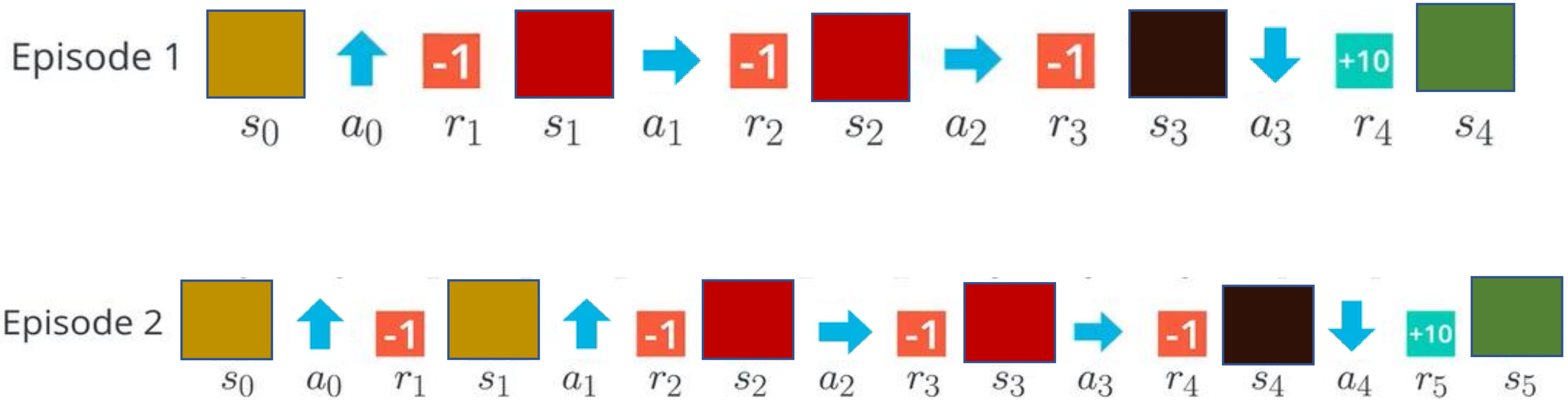
- Goal: Estimate $V^{\pi}(s)$ under policy π
  - Recall: $V^{\pi}(s)$ is the expected utility starting at s and following policy π
- Average together observed sample values
  - Act according to π and collect data: $s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_T$
  - First time you visit a state, write down the sum of discounted rewards: $G_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \ldots$
  - Average those samples to compute $\hat{V}^{\pi}(s)$
- Same idea for estimating $Q^{\pi}(s, a)$, focusing though on q-states

Example for estimating state value

Input Policy π

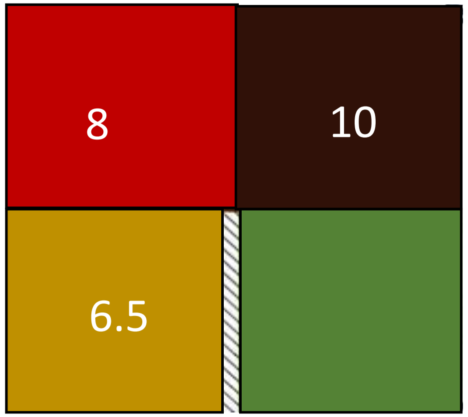


Observed Episodes

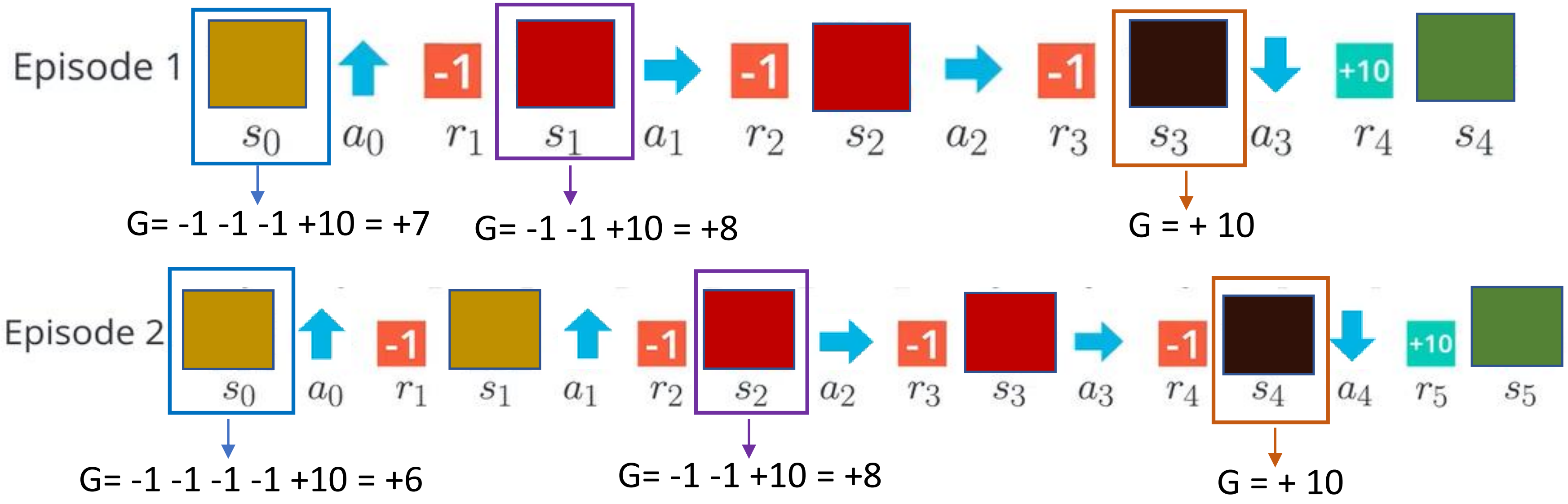


Example for estimating state value

Output values

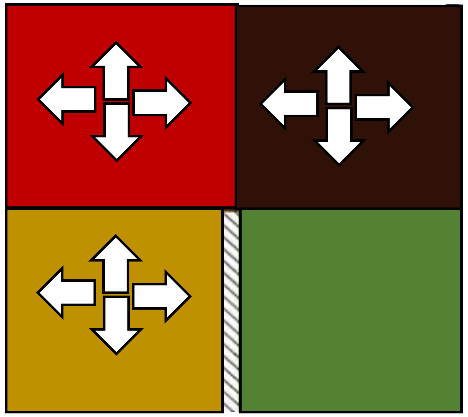


Observed Episodes

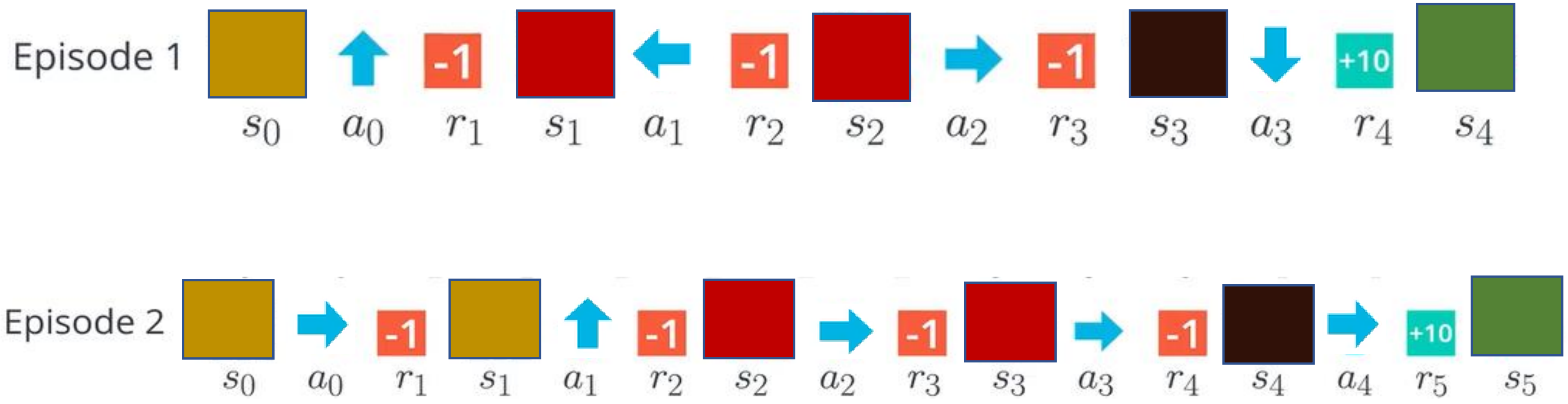


Example for estimating action-state values

Random Policy



Observed Episodes

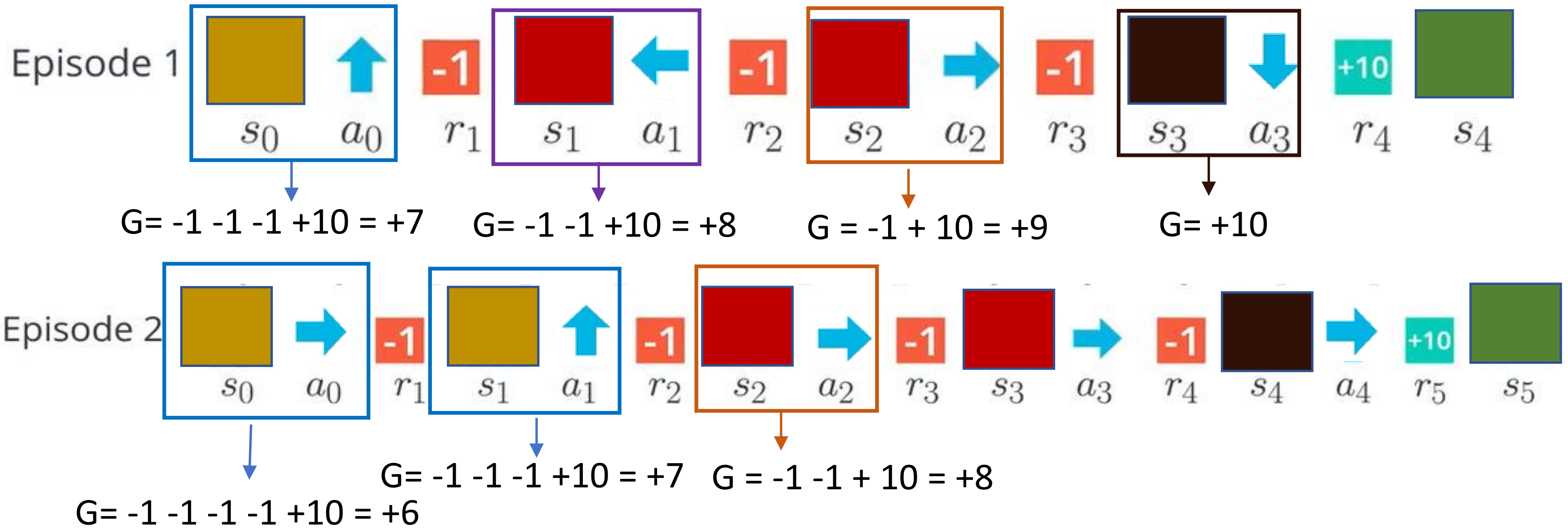


Example for estimating action-state values

Output values

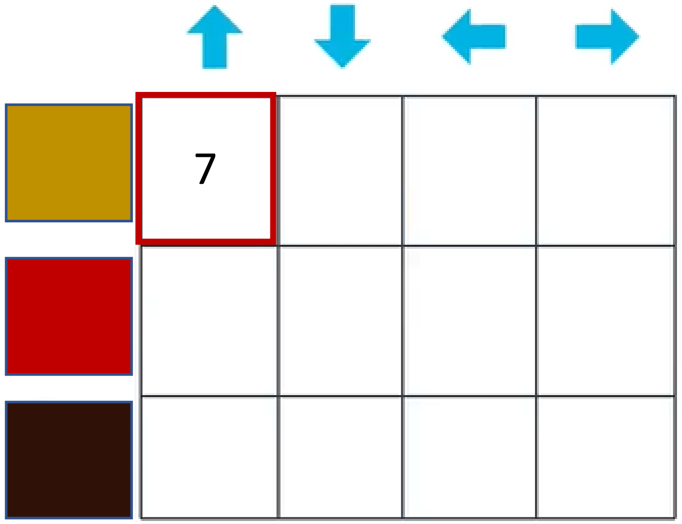
	↑	↓	←	→
	7			6
			8	8.5
		10		10

Observed Episodes

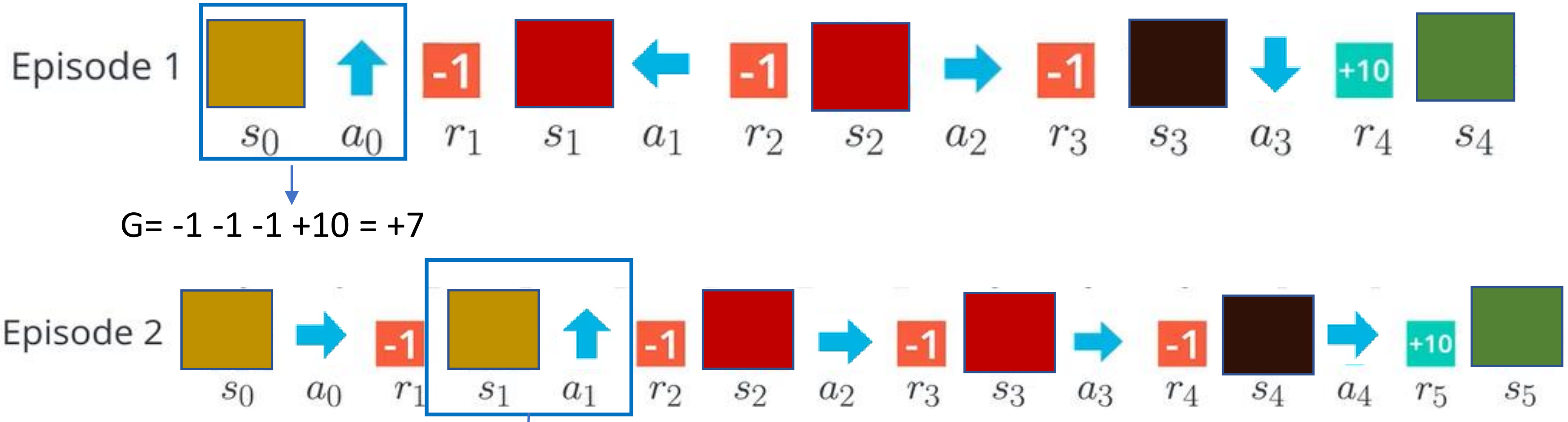


Incremental mean

Output values



Observed Episodes



G= -1 -1 -1 +10 = +7

G= -1 -1 -1 +10 = +7

Idea: Incorporate returns, G, as they come

Incremental mean

- Data (following policy π): $s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_T$
- $\hat{Q}^\pi(s, a)$ = average of $G_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \ldots$, where $s_t = s, a_t = a$
- Alternative formulation
  - Update $\hat{Q}^\pi(s, a)$ once a G_t has been computed (episode ends)
  - Keep a running average between current estimate and new returns

Sample: $G_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \ldots$

Update: $\hat{Q}^\pi(s, a) \leftarrow (1 - \alpha)\,\hat{Q}^\pi(s, a) + \alpha\, G_t$

Incremental mean

- Data (following policy π): $s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_T$
- $\hat{Q}^\pi(s, a)$ = average of $G_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \ldots$, where $s_t = s, a_t = a$
- Alternative formulation
  - Update $\hat{Q}^\pi(s, a)$ once a G_t has been computed (episode ends)
  - Keep a running average between current estimate and new returns

Sample: $G_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \ldots$

Update: $\hat{Q}^\pi(s, a) \leftarrow (1 - \alpha)\, \hat{Q}^\pi(s, a) + \alpha\, G_t$

Same update: $\hat{Q}^\pi(s, a) \leftarrow \hat{Q}^\pi(s, a) + \alpha(\underbrace{G_t}_{\text{target}} - \underbrace{\hat{Q}^\pi(s, a)}_{\text{prediction}})$

Interpretation: Gradient descent on $(G_t - \hat{Q}^\pi(s, a))^2$

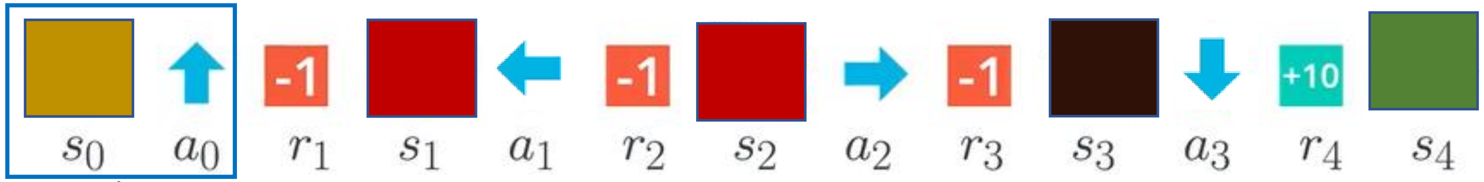
Example

Current estimates

	↑	↓	←	→
	7.5 ~~8~~	6	6	5
	9	8	7	9
	10	10	9	10

Assume: $\gamma = 1$, $\alpha = 1/2$

Observed Episodes

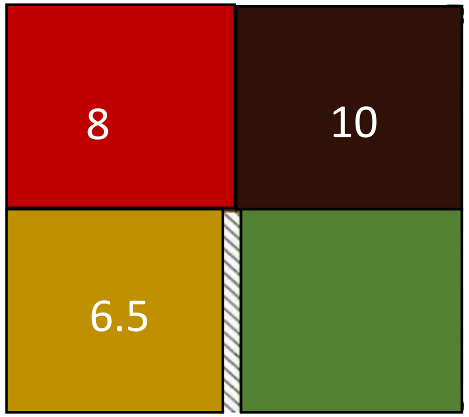


G= -1 -1 -1 +10 = +7

Direct evaluation (model-free Monte Carlo)

- The good
  - It’s easy to understand
  - It doesn’t require any knowledge of P, R
  - It eventually computes the correct average values, using just sample transitions (no bias)

- The not so good
  - Doesn’t recognize that the underlying model is an MDP
  - Each state must be learned separately. So it takes a long time to learn
  - Need to wait till the end of an episode to update values
  - High variance



Temporal Difference Learning

Temporal difference learning

- $Q^{\pi}(s, a)$ is expected utility starting in s, taking action a, and then following policy π
  - $Q^{\pi}(s,a) = E_{\pi}[G_t | s_t = s, a_t = a]$, where $G_t = r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+2} + \ldots$
  - $Q^{\pi}(s,a) = E_{\pi}[R_{t+1} + \gamma Q^{\pi}(s_{t+1}, \pi(s_{t+1})) | s_t = s, a_t = a]$
- Monte Carlo Evaluation
  - Act according to π and collect data: $s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_T$
  - $\hat{Q}^{\pi}(s,a)$ = average of G_t where $s_t = s, a_t = a$
- Recall: policy evaluation?

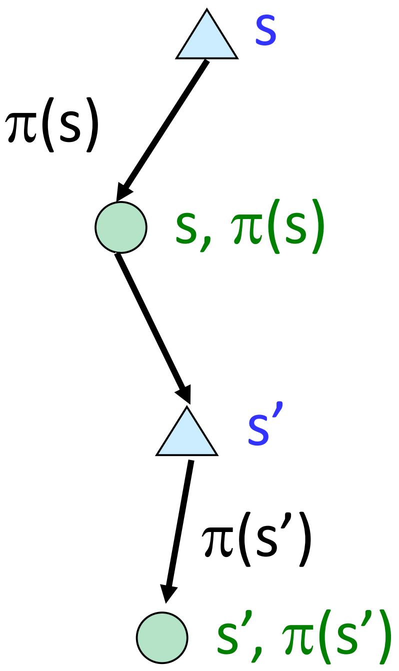
$$V_k^{\pi}(s) \leftarrow \sum P(s'|s,\pi(s))(R(s,\pi(s),s') + \gamma V_{k-1}^{\pi}(s'))$$

$$Q_k^{\pi}(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma Q_{k-1}^{\pi}(s', \pi(s')))$$

Idea: What if we make policy evaluation sample-based?

Temporal difference learning

- Big idea: learn from every experience!
  - Update $\hat{Q}^{\pi}(s,a)$ each time we experience a transition (s, a, r, s', a')
  - Keep a running average between current estimate and new experiences



Sample of $\hat{Q}^{\pi}(s,a)$: $\quad target = R(s,\pi(s),s') + \gamma \hat{Q}^{\pi}(s',a')$

Update $\hat{Q}^{\pi}(s,a)$: $\quad \hat{Q}^{\pi}(s,a) \leftarrow (1-\alpha)\,\hat{Q}^{\pi}(s,a) + \alpha\, target$

- The above is known as SARSA
- We can use TD-learning to estimate $\hat{V}^{\pi}(s)$ based on (s, a, r, s') transitions

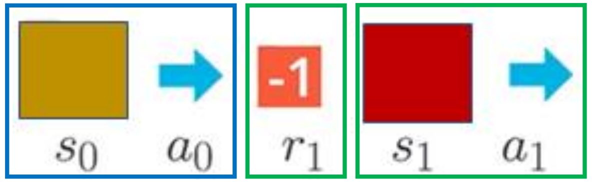
SARSA example

Current estimates

	↑	↓	←	→
	8	6	6	6.5 ~~5~~
	8	8	7	9
	10	10	9	10

Assume: $\gamma = 1$, $\alpha = 1/2$

Observed Transitions



Current estimate $\hat{Q}^{\pi}(\; , \;)$ = 5

Sample (target) = $-1 + \gamma \hat{Q}^{\pi}(\; , \;)$ = 8

Active Reinforcement Learning

Active RL

- Full reinforcement learning
  - You don’t know the transitions P(s’|s,a)
  - You don’t know the rewards R(s,a,s’)
  - You choose the actions now
  - Goal: learn the optimal policy / values

- Learner makes choices!
- Fundamental tradeoff: exploration vs. exploitation
- This is not offline planning! You actually take actions in the world

Recap: Q-Values

- Q*(s, a) = expected utility starting in s, taking action a and (thereafter) acting optimally
- Bellman equation

$$Q^*(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q^*(s',a'))$$

- Q-Value Iteration

$$Q_0^*(s,a) \leftarrow 0$$

$$Q_{k+1}^*(s,a) \leftarrow \sum_{s'} P(s'|s,a)(R(s,a,s') + \gamma \max_{a'} Q_k^*(s',a'))$$

(Tabular) Q-Learning

- Q-Learning: sample-based Q-value iteration

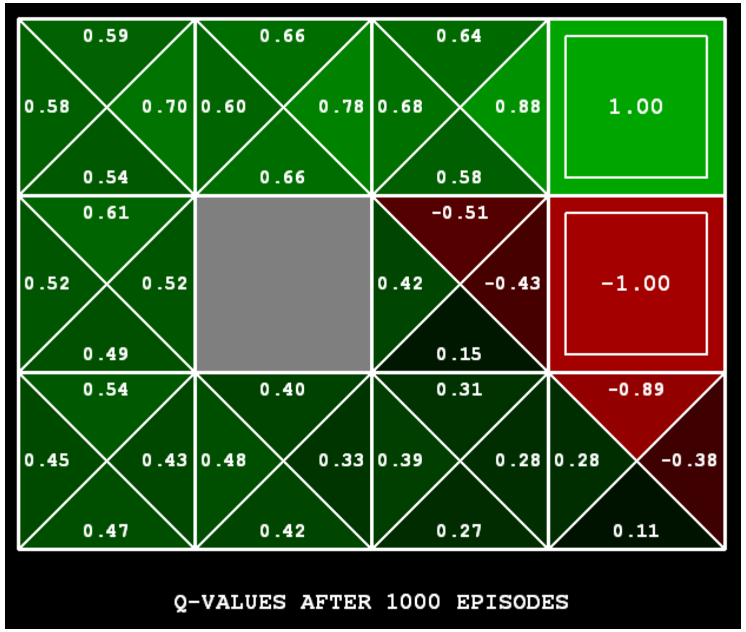
$$Q_{k+1}(s,a) \leftarrow \sum_{s'} P(s'|s,a)\left(R(s,a,s') + \gamma \max_{a'} Q_k(s',a')\right)$$

- Learn Q-values as you go
  - Receive a sample (s, a, s', r)
  - Consider your previous estimate: $\hat{Q}(s,a)$
  - Consider your new sample estimate

$$\textit{target} = R(s,a,s') + \gamma \max_{a'} \hat{Q}(s',a')$$

  - Incorporate the new estimate into a running average

$$\hat{Q}(s,a) \leftarrow (1-\alpha)\hat{Q}(s,a) + (\alpha)\,[\textit{target}]$$



(Tabular) Q-Learning

Initialize $Q(s, a)$ for all s, a

Repeat (for each episode):

Get initial state s

Repeat (for each step of episode):

Sample action a from s, observe reward r and next state s'

If s' is terminal:

$$target = r$$

else:

$$target = r + \gamma \max_{a'} Q(s', a')$$

$$Q(s, a) \leftarrow (1 - \alpha)Q(s, a) + (\alpha) \; [\, target \,]$$

$$s \leftarrow s'$$

until s is terminal

until convergence

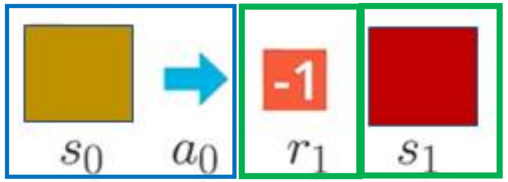
Example

Current q-table

	↑	↓	←	→
	$\boxed{8}$	6	6	7
	8	8	7	$\boxed{9}$
	10	10	9	10

Assume: $\gamma = 1$, $\alpha = 1/2$

Observed Transitions



Current estimate of $\hat{Q}^{\pi}(\blacksquare, \rightarrow)$ = 8

Sample (target) = -1 + 9 = 8

New estimate = 0.5×8 + 0.5×8 =8

$$\hat{Q}(s,a) \leftarrow (1-\alpha)\hat{Q}(s,a) + (\alpha)\left[R(s,a,s') + \gamma \max_{a'} \hat{Q}(s',a')\right]$$

Example: cliffworld

Q-learning properties

- Q-learning converges to optimal policy, even if you're acting suboptimally!
- This is called off-policy learning
- Caveats
  - You have to explore enough
  - You have to eventually make the learning rate small enough
    - … but not decrease it too quickly
  - Basically, in the limit, it doesn't matter how you select actions!

Q-learning properties

- Q-learning converges to optimal policy, even if you're acting suboptimally!
- This is called off-policy learning
- Caveats
  - You have to explore enough
  - You have to eventually make the learning rate small enough
    - … but not decrease it too quickly
  - Basically, in the limit, it doesn't matter how you select actions!

See [Jaakkola et. al 1994, "On the convergence of stochastic iterative dynamic programming algorithms"] for more