10-301/601: Introduction to Machine Learning Lecture 20: Markov Decision Processes

Matt Gormley & Henry Chai

3/26/25

Front Matter

- Announcements
  - Exam 2 on 3/26 (today!) from 7 – 9 PM
    - Please review the seating chart on Piazza and make sure you have a seat / know where you're going
  - HW7 to be released 3/26, due 4/8 at 11:59 PM
    - Please be mindful of your grace day usage (see the course syllabus for the policy)
    - If you have not used PyTorch before, I ***strongly*** encourage you to go to recitation on Friday (3/28)

Learning Paradigms

- Supervised learning - $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(n)}, y^{(n)}\right)\right\}_{n=1}^{N}$
  - Regression - $y^{(n)} \in \mathbb{R}$
  - Classification - $y^{(n)} \in \{1, \dots, C\}$
- Reinforcement learning - $\mathcal{D} = \left\{\boldsymbol{s}^{(n)}, \boldsymbol{a}^{(n)}, r^{(n)}\right\}_{n=1}^{N}$

Source: https://techobserver.net/2019/06/argo-ai-self-driving-car-research-center/

Source: https://www.wired.com/2012/02/high-speed-trading/

Reinforcement Learning: Examples

Source: https://www.cnet.com/news/boston-dynamics-robot-dog-spot-finally-goes-on-sale-for-74500/

Source: https://twitter.com/alphagomovie

AlphaGo

Source: https://www.youtube.com/watch?v=WXuK6gekU1Y&ab_channel=DeepMind

Outline

- Problem formulation
  - Time discounted cumulative reward
  - Markov decision processes (MDPs)
- Algorithms:
  - Value & policy iteration (dynamic programming)
  - (Deep) Q-learning (temporal difference learning)

Reinforcement Learning: Problem Formulation

- State space, $\mathcal{S}$
- Action space, $\mathcal{A}$
- Reward function
  - Stochastic, $p(r \mid s, a)$
  - Deterministic, $R\colon \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- Transition function
  - Stochastic, $p(s' \mid s, a)$
  - Deterministic, $\delta\colon \mathcal{S} \times \mathcal{A} \to \mathcal{S}$

Reinforcement Learning: Problem Formulation

- Policy, $\pi : \mathcal{S} \rightarrow \mathcal{A}$
  - Specifies an action to take in *every* state
- Value function, $V^{\pi}: \mathcal{S} \rightarrow \mathbb{R}$
  - Measures the expected total payoff of starting in some state s and executing policy π, i.e., in every state, taking the action that π returns

Toy Example

- $\mathcal{S}$ = all empty squares in the grid
- $\mathcal{A}$ = {up, down, left, right}
- Deterministic transitions
- Rewards of +1 and -1 for entering the labelled squares
- Terminate after receiving either reward

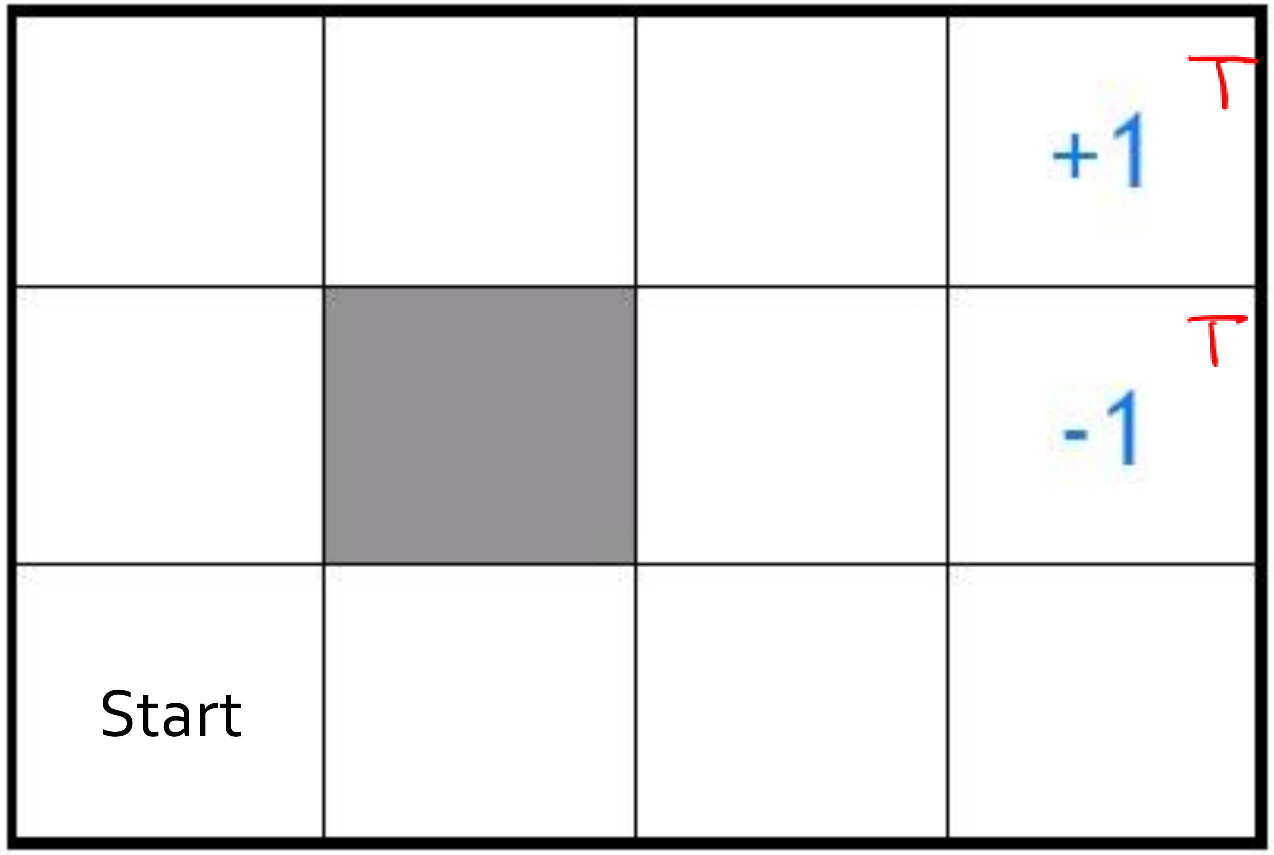


Figure courtesy of Eric Xing

Toy Example

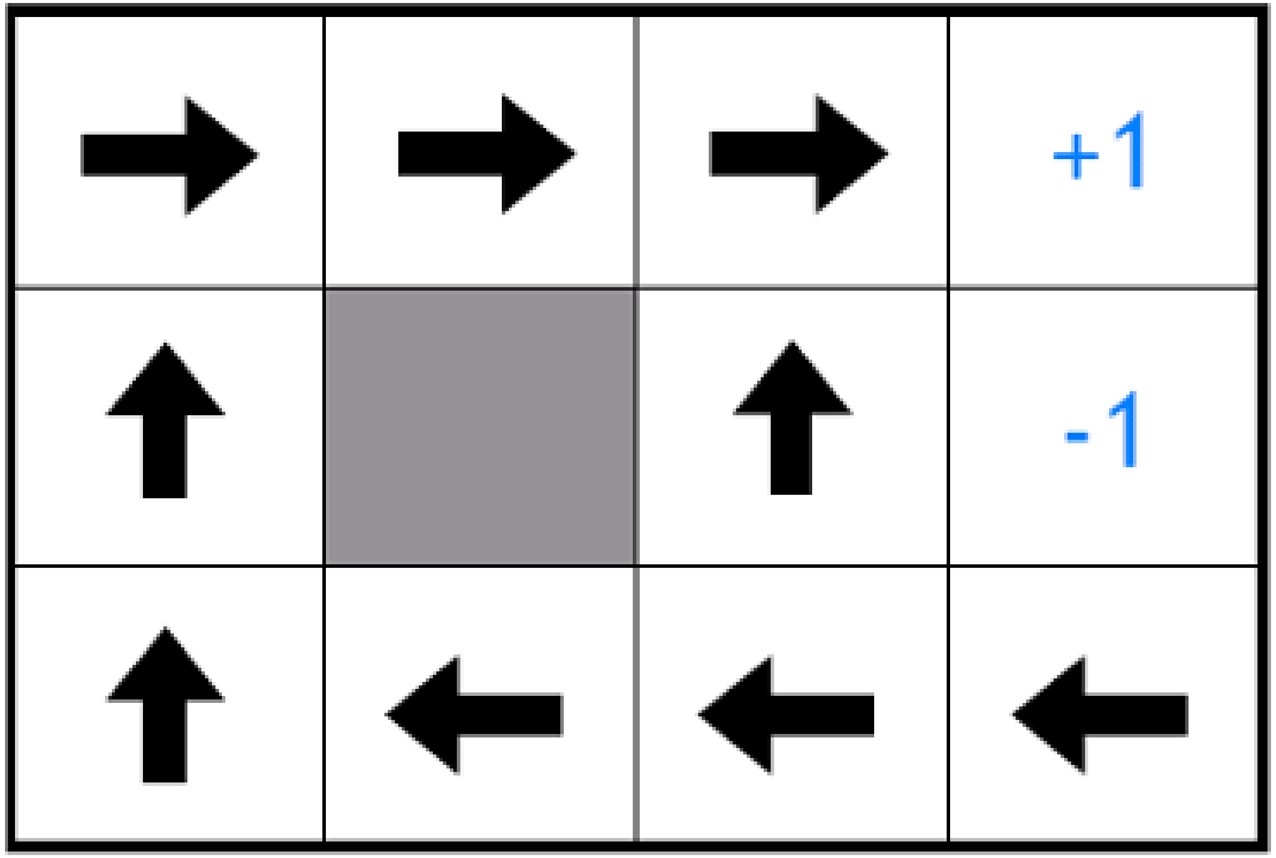


Figure courtesy of Eric Xing

Poll Question 1:

Is this policy optimal?

A. Yes

B. TOXIC

C. No

Poll Question 2:

Justify your answer to the previous question

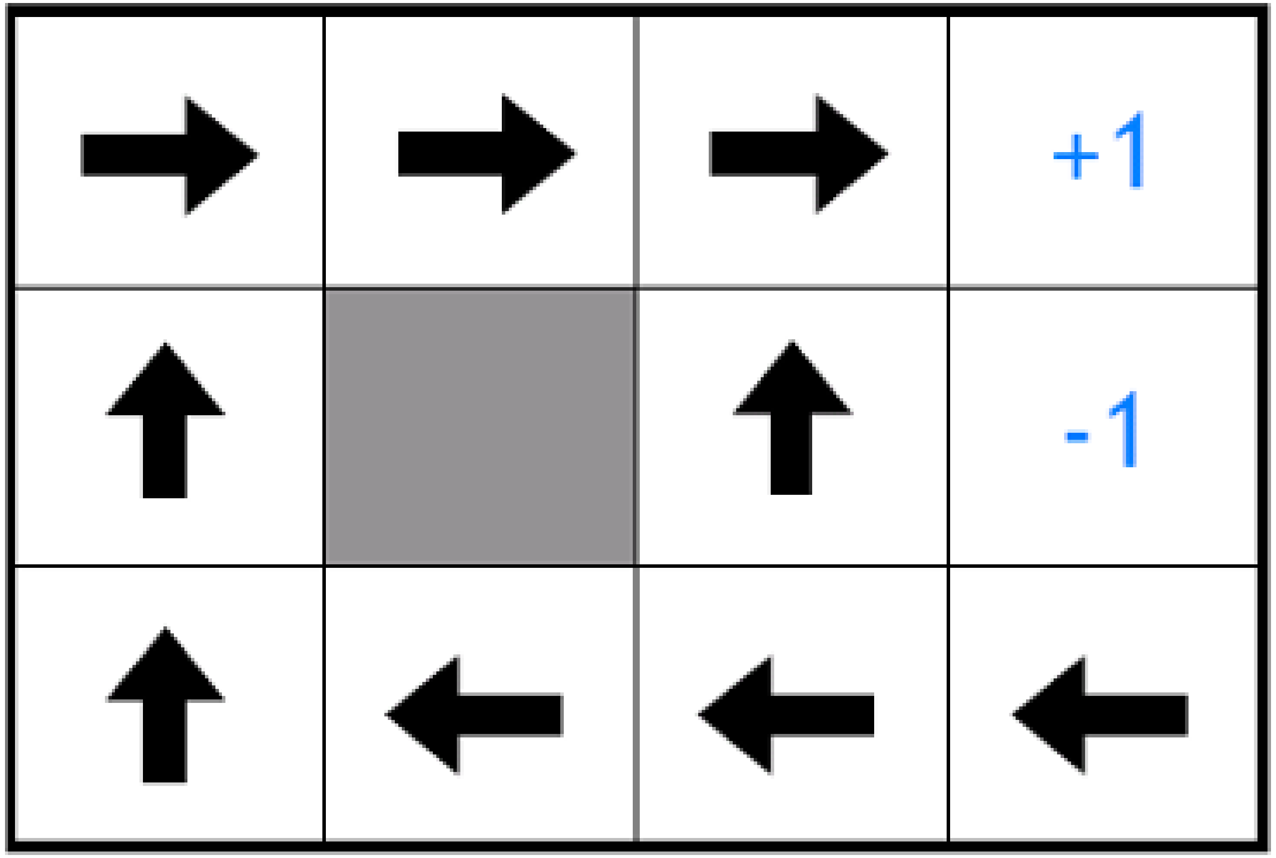


Figure courtesy of Eric Xing

Toy Example

Optimal policy given a reward of -2 per step

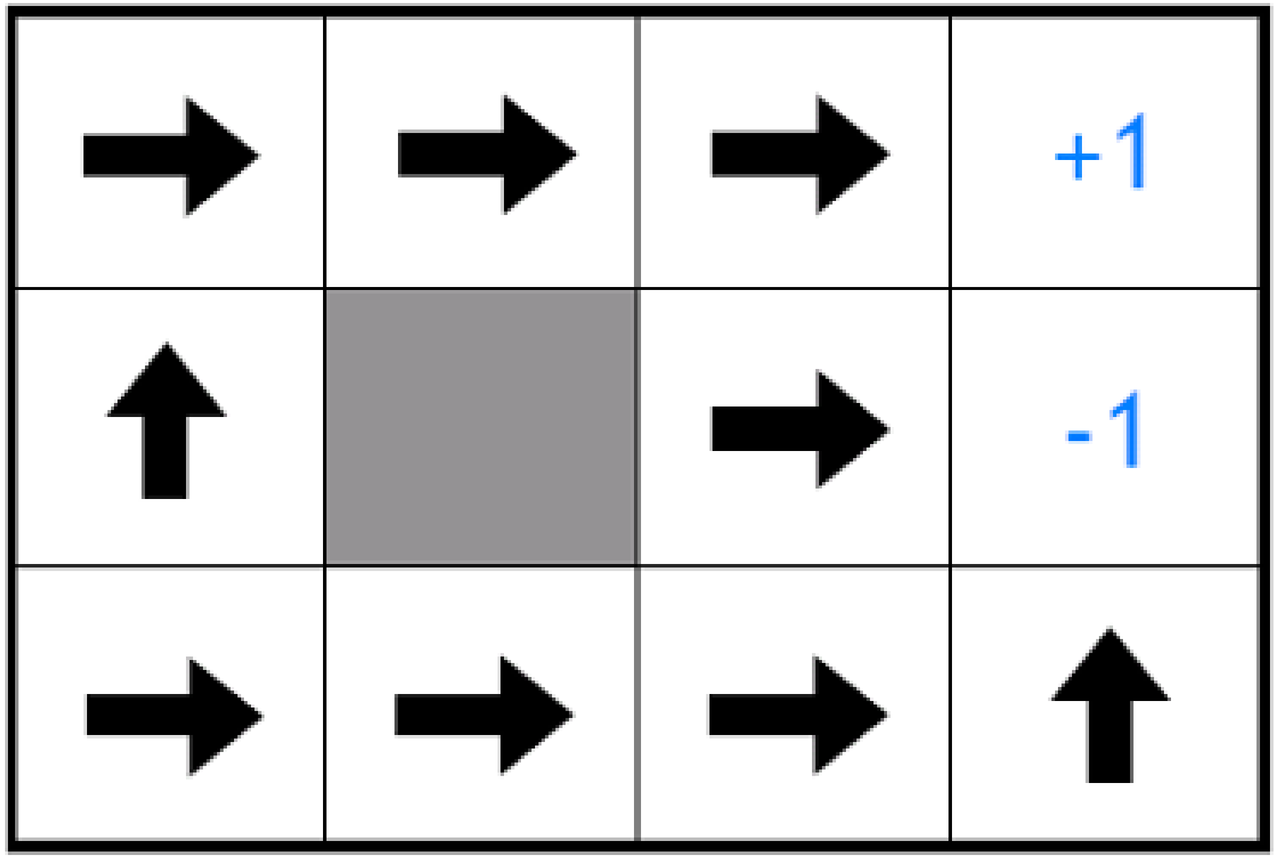


Figure courtesy of Eric Xing

Toy Example

Optimal policy given a reward of -0.1 per step

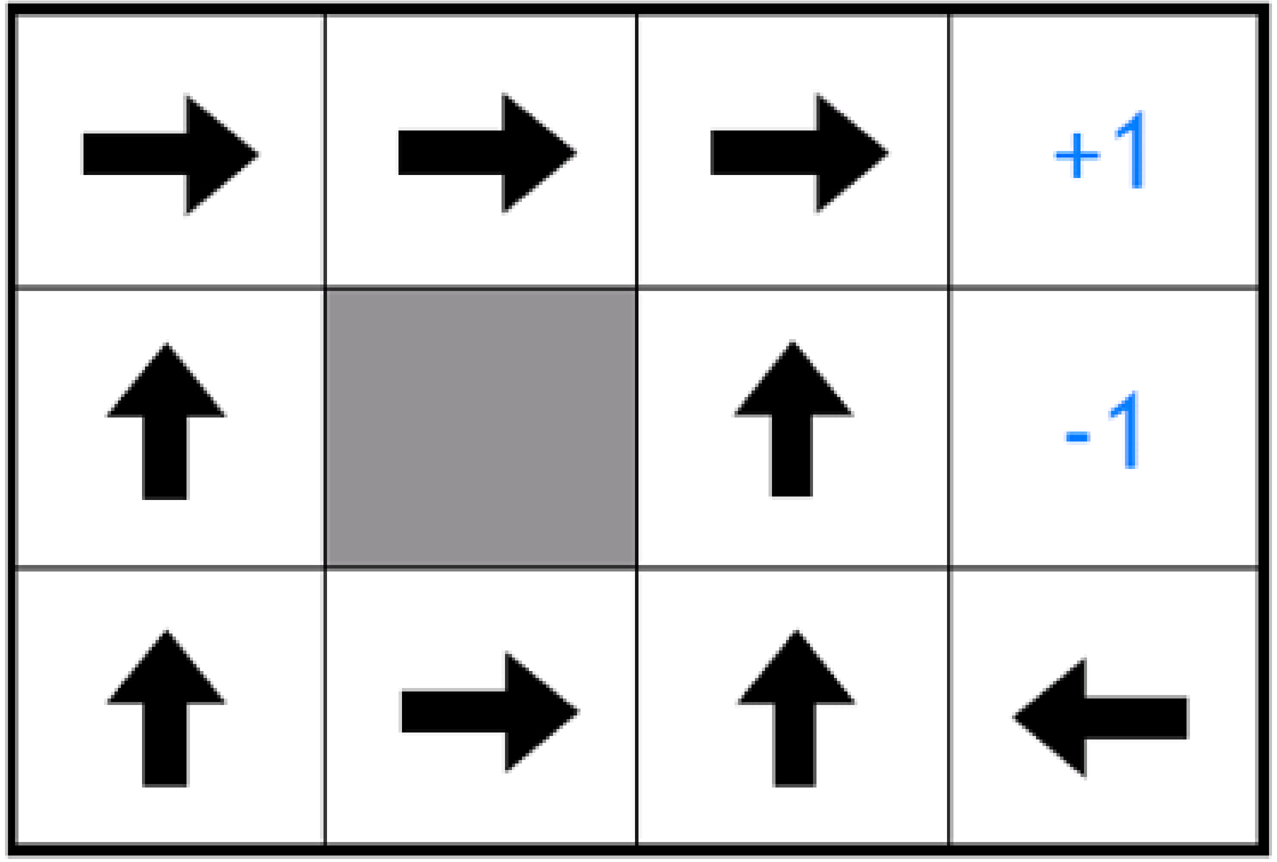


Figure courtesy of Eric Xing

Markov Decision Process (MDP)

- Assume the following model for our data:

1. Start in some initial state s_0
2. For time step t:
    1. Agent observes state s_t
    2. Agent takes action $a_t = \pi(s_t)$
    3. Agent receives reward $r_t \sim p(r \mid s_t, a_t)$
    4. Agent transitions to state $s_{t+1} \sim p(s' \mid s_t, a_t)$
3. Total reward is $\sum_{t=0}^{\infty} \gamma^t r_t$

discount factor
$0 \leq \gamma < 1$

- MDPs make the *Markov assumption*: the reward and next state only depend on the current state and action.

Reinforcement Learning: 3 Key Challenges

1. The algorithm has to gather its own training data
2. The outcome of taking some action is often stochastic or unknown until after the fact
3. Decisions can have a delayed effect on future outcomes (exploration-exploitation tradeoff)

MDP Example: Multi-armed bandit

- Single state: $|\mathcal{S}| = 1$
- Three actions: $\mathcal{A} = \{1, 2, 3\}$
- Deterministic transitions
- Rewards are stochastic

Reinforcement Learning: Objective Function

- Find a policy $\pi^* = \underset{\pi}{\operatorname{argmax}} \ \underbrace{V^\pi(s)} \ \forall \ s \in \mathcal{S}$
- Assume deterministic transitions and deterministic rewards
- $V^\pi(s) =$ *discounted* total reward of starting in state s and executing policy π forever

$E[a+b]$
$= E[a] + E[b]$

Reinforcement Learning: Objective Function

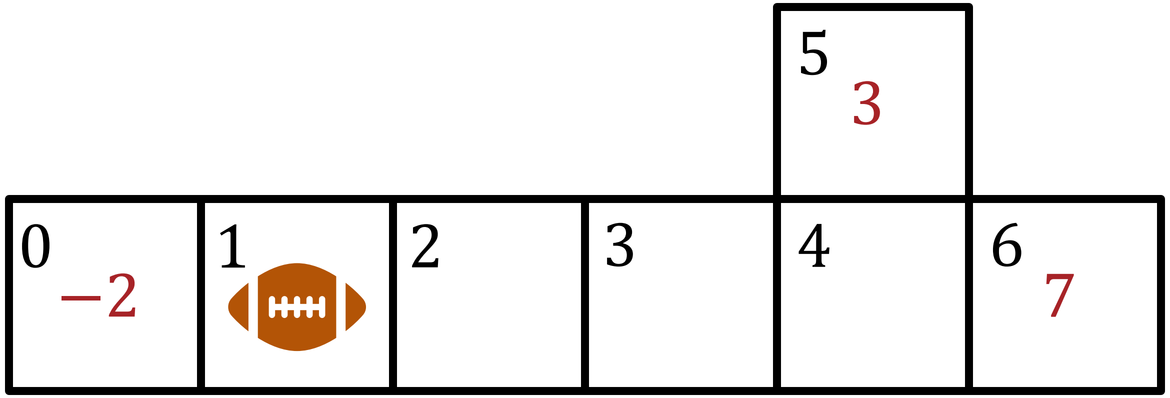
- Find a policy $\pi^* = \underset{\pi}{\text{argmax}}\ V^\pi(s)\ \forall\, s \in \mathcal{S}$
- Assume stochastic transitions and deterministic rewards
- $V^\pi(s) = \mathbb{E}[$*discounted* total reward of starting in state s and executing policy π forever$]$

$$= E_{p(s'|s,a)}\Big[R(s_0 = s, \pi(s_0)) + \gamma R(s_1, \pi(s_1)) + \gamma^2 R(s_2, \pi(s_2)) + \dots\Big]$$

$$= \sum_{t=0}^{\infty} \gamma^t E_{p(s'|s,a)}\big[R(s_t, \pi(s_t))\big]$$

where γ is my discount factor: $0 \leq \gamma < 1$

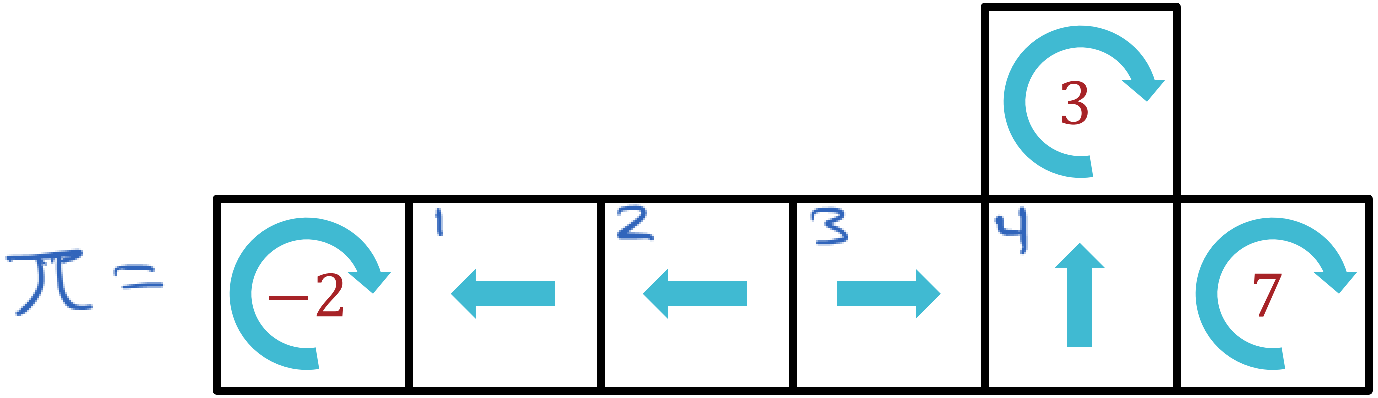
Value Function: Example



$$R(s,a) = \begin{cases} -2 \text{ if entering state 0 (safety)} \\ 3 \text{ if entering state 5 (field goal)} \\ 7 \text{ if entering state 6 (touch down)} \\ 0 \text{ otherwise} \end{cases}$$

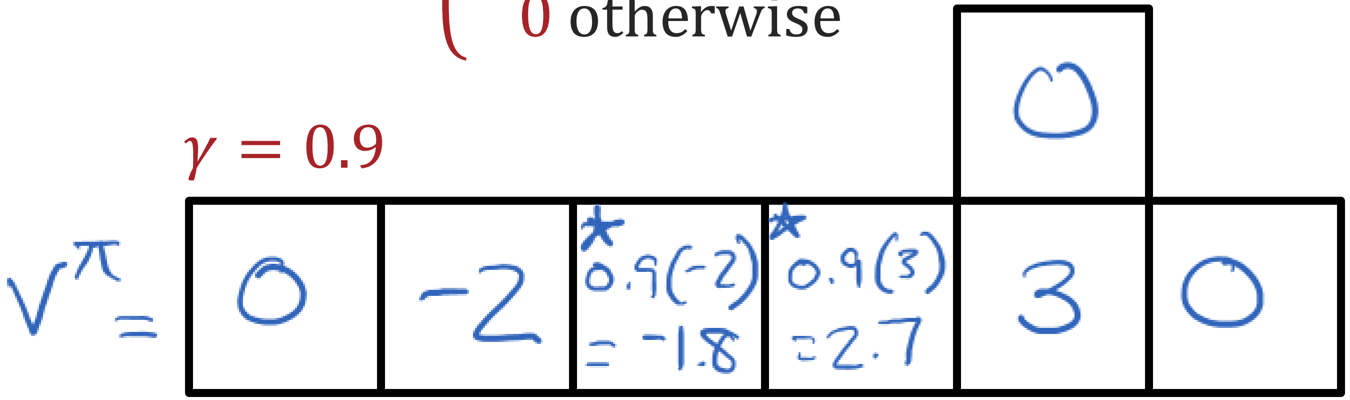
$$\gamma = 0.9$$

Value Function: Example



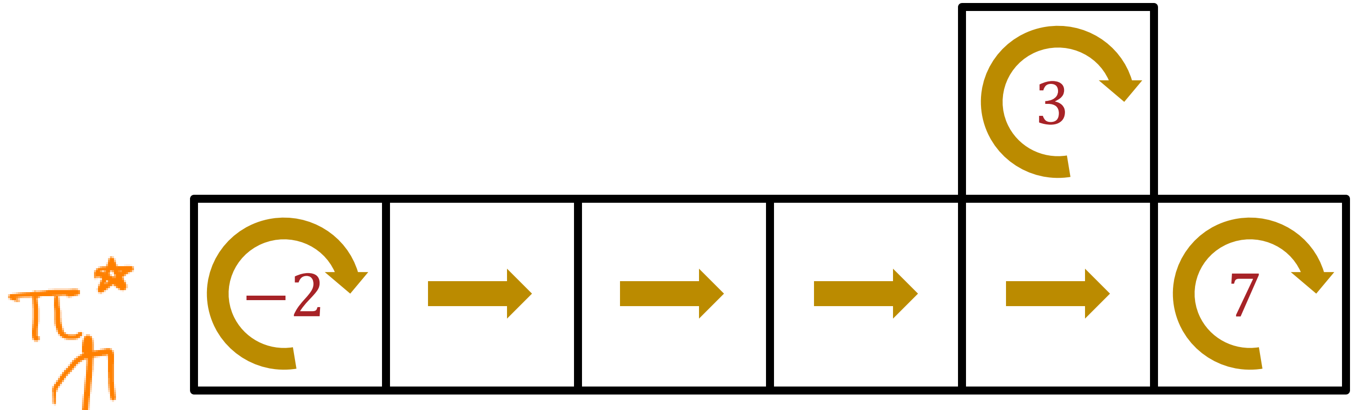
$$R(s,a) = \begin{cases} -2 \text{ if entering state 0 (safety)} \\ 3 \text{ if entering state 5 (field goal)} \\ 7 \text{ if entering state 6 (touch down)} \\ 0 \text{ otherwise} \end{cases}$$

$\gamma = 0.9$



Okay, now how do we go about learning this optimal policy?

Value Function: Example



$$R(s,a) = \begin{cases} -2 \text{ if entering state 0 (safety)} \\ 3 \text{ if entering state 5 (field goal)} \\ 7 \text{ if entering state 6 (touch down)} \\ 0 \text{ otherwise} \end{cases}$$

$\gamma = 0.9$