Reinforcement Learning I

Aarti Singh

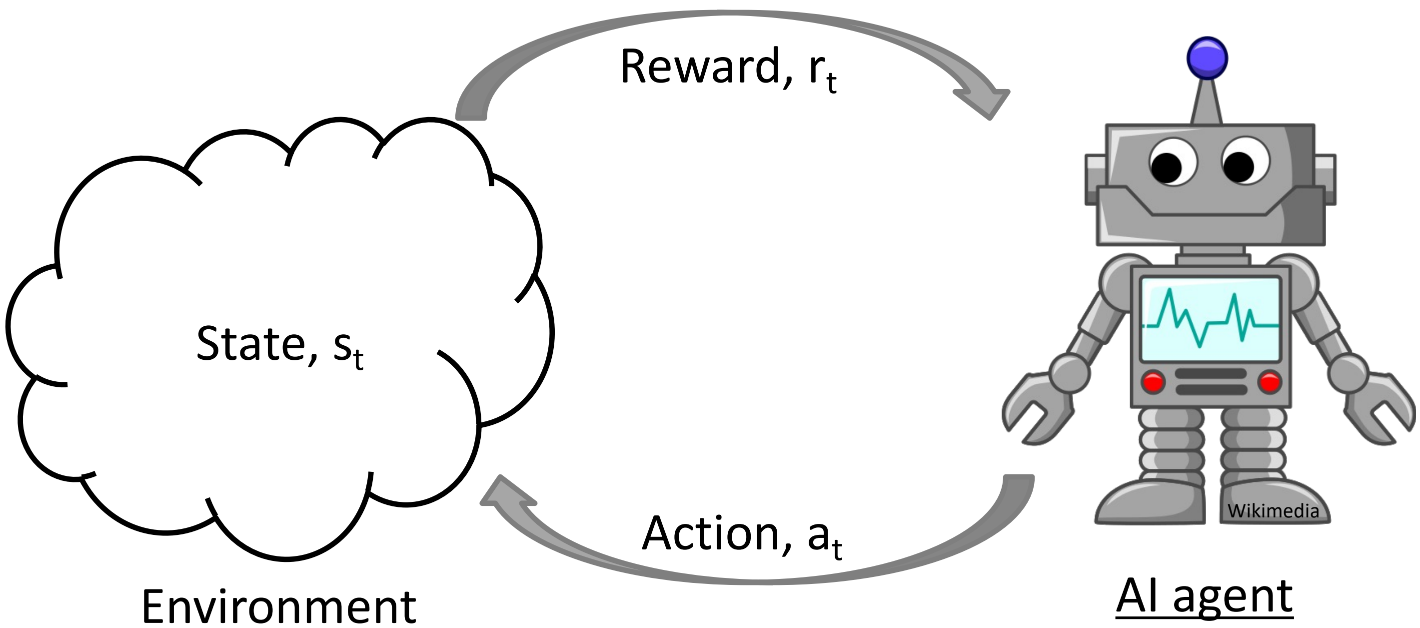
Machine Learning 10-701
Apr 3, 2023

Slides courtesy: Henry Chai, Eric Xing

Learning Tasks

- Supervised learning - $\mathcal{D} = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$
  - Regression - $y^{(i)} \in \mathbb{R}$
  - Classification - $y^{(i)} \in \{1, \ldots, C\}$
- Unsupervised learning - $\mathcal{D} = \left\{\boldsymbol{x}^{(i)}\right\}_{i=1}^{N}$
  - Clustering
  - Dimensionality reduction
- Reinforcement learning - $\mathcal{D} = \left\{\boldsymbol{s}^{(t)}, \boldsymbol{a}^{(t)}, r^{(t)}\right\}_{t=1}^{T}$

RL setup



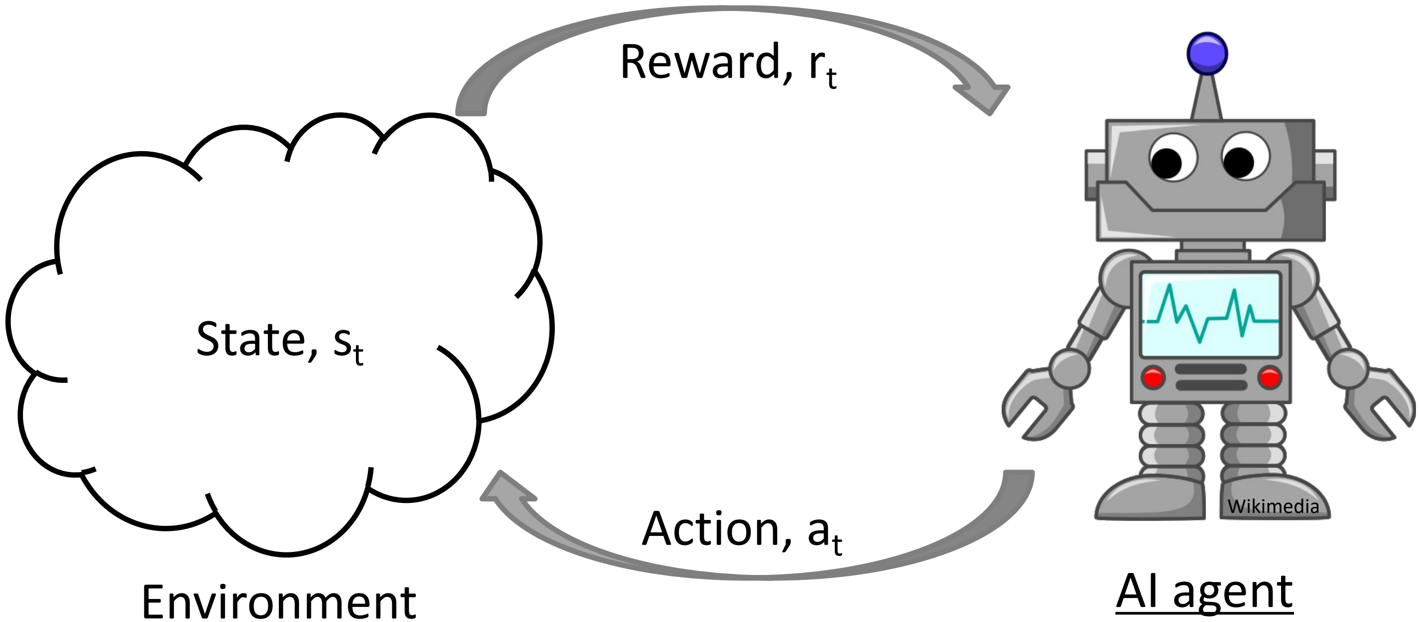
Agent chooses **actions** which can depend on past

Environment can change **state** with each action

Reward (Output) depends on (Inputs) action and state of environment

Goal: Maximize total reward

Differences from supervised learning



- Maximize reward (rather than learn reward)
- Inputs are not iid – state & action depends on past
- Can control some inputs - actions

RL examples

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

https://www.cmu.edu/news/stories/archives/2017/september/snakebot-mexico.html

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

https://twitter.com/alphagomovie

RL setup

- State space, $\mathcal{S}$
- Action space, $\mathcal{A}$
- Reward function
    - Stochastic, $p(r \mid s, a)$
    - Deterministic, $R: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$
- Transition function
    - Stochastic, $p(s' \mid s, a)$
    - Deterministic, $\delta: \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$
- Reward and transition functions can be known or unknown

RL setup

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

RL example - gridworld

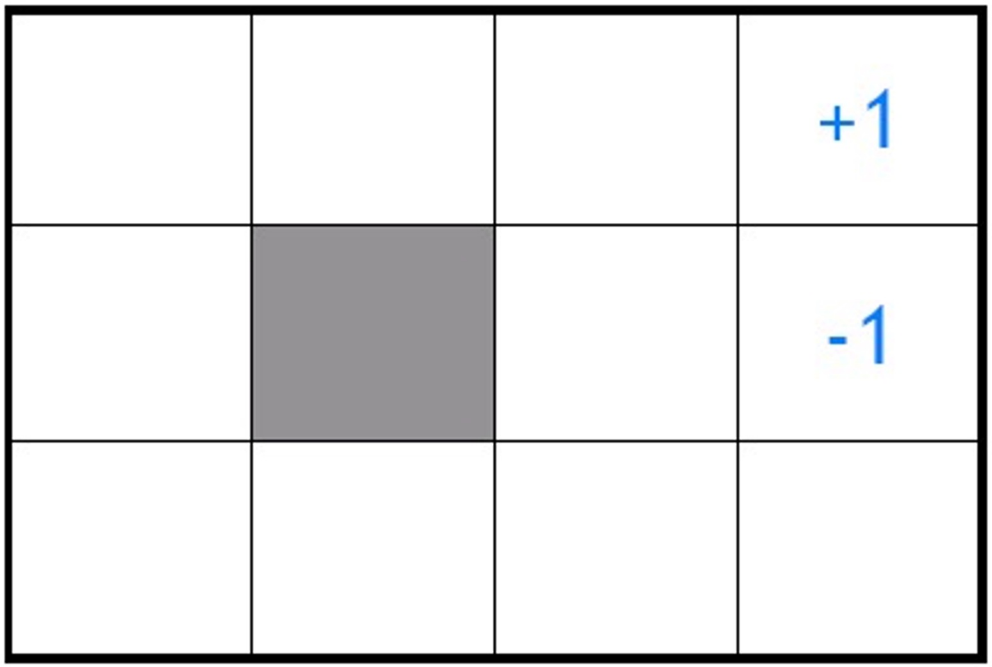
$\mathcal{S}$ = all empty squares in the grid

$\mathcal{A}$ = {up, down, left, right}

Deterministic transitions

Rewards of +1 and -1 for entering the labelled squares

Terminate after receiving either reward



RL example - gridworld

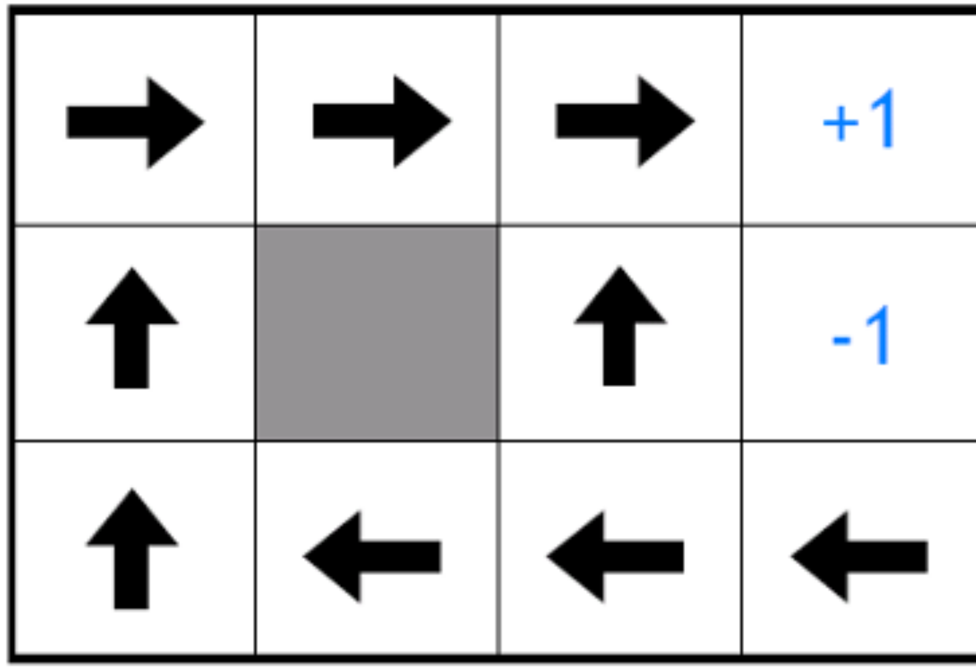
$\mathcal{S}$ = all empty squares in the grid

$\mathcal{A}$ = {up, down, left, right}

Deterministic transitions

Rewards of +1 and -1 for entering the labelled squares

Terminate after receiving either reward



Poll: Is this policy optimal?

RL example - gridworld

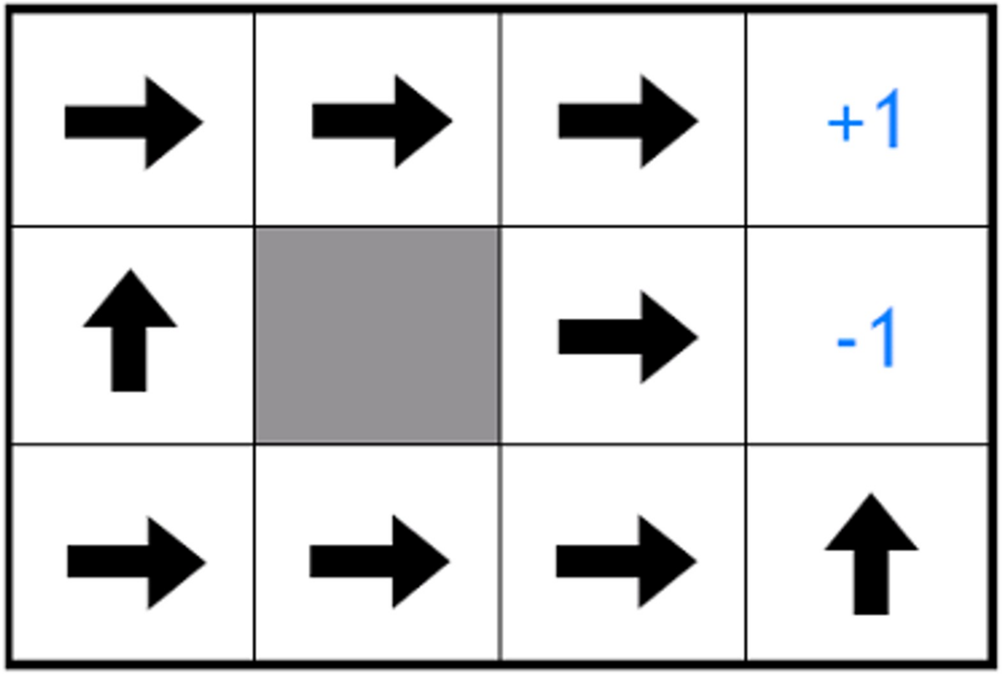
$\mathcal{S}$ = all empty squares in the grid

$\mathcal{A}$ = {up, down, left, right}

Deterministic transitions

Rewards of +1 and -1 for entering the labelled squares

Terminate after receiving either reward



Optimal policy given a reward of -2 per step

RL example - gridworld

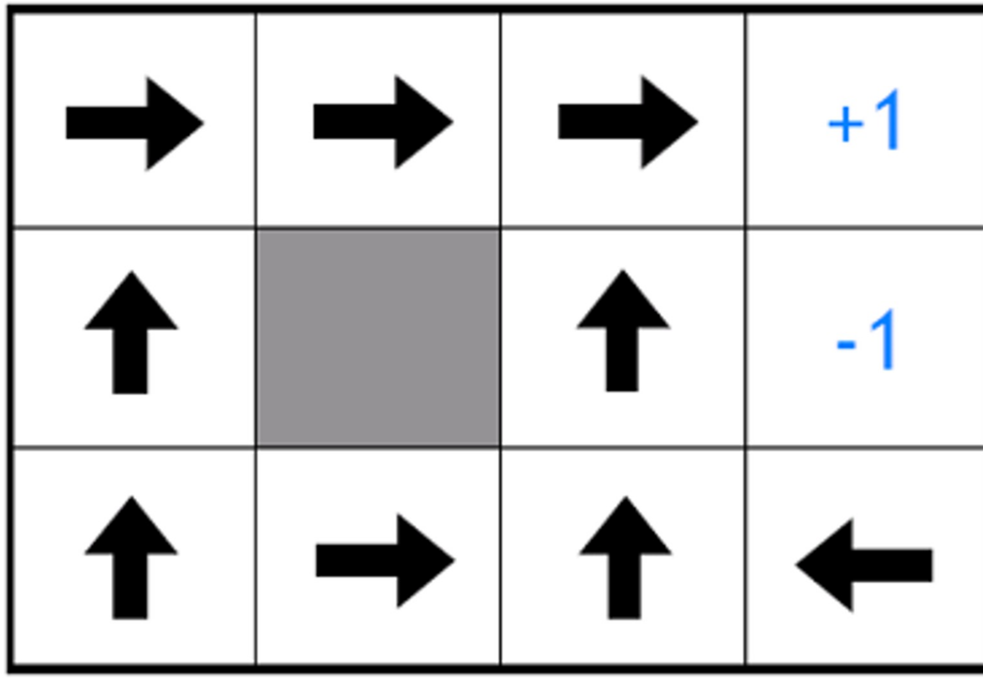
$\mathcal{S}$ = all empty squares in the grid

$\mathcal{A}$ = {up, down, left, right}

Deterministic transitions

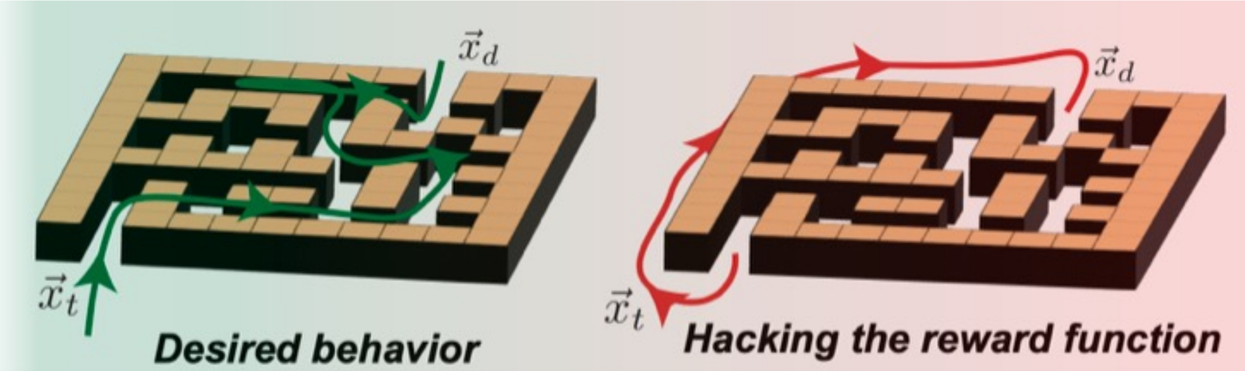
Rewards of +1 and -1 for entering the labelled squares

Terminate after receiving either reward



Optimal policy given a reward of -0.1 per step

Reward hacking



AIhub.org

[Amodei-Clark'16]

Markov Decision Process

1. Start in some initial state s_0
2. For time step t:
   a. Agent observes state s_t
   b. Agent takes action $a_t = \pi(s_t)$ Deterministic policy
   c. Agent receives reward $r_t \sim p(r \mid s_t, a_t)$
   d. Agent transitions to state $s_{t+1} \sim p(s' \mid s_t, a_t)$

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

Discounted Reward

Total reward is $\sum_{t=0}^{\infty} \gamma^t r_t = r_0 + \gamma\, r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \ \ldots$

where $0 < \gamma < 1$ is some discount factor for future rewards

Why discount?

- Mathematically tractable – total reward doesn't explode

  $1 + 1 + 1 + \ldots = \infty$ but $1 + 0.8*1 + (0.8)^2*1 + \ldots = 5$

- Risk aversion under uncertainty
- Actions don't have lasting impact

Key challenges

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

  explore decisions whose reward is uncertain

  exploit decisions which give high reward

MDP example: Multi-armed bandits

Single state: $|\mathcal{S}| = 1$

Three actions: $\mathcal{A} = \{1, 2, 3\}$

Deterministic transitions

Rewards are stochastic

MDP example: Multi-armed bandits

Bandit arm 1	Bandit arm 2	Bandit arm 3
1	???	???
1	???	???
1	???	???
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???

RL: objective function

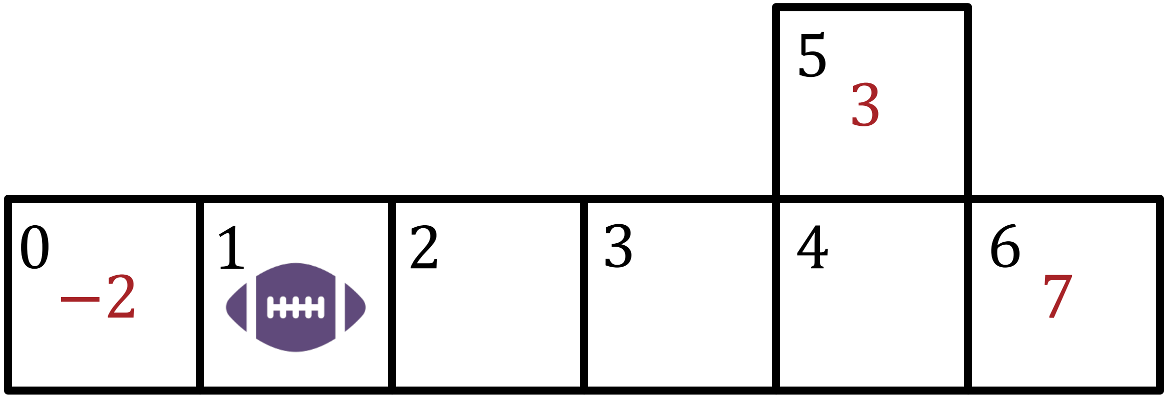
- Find a policy $\pi^* = \underset{\pi}{\operatorname{argmax}}\ V^\pi(s)\ \forall\, s \in \mathcal{S}$
- $V^\pi(s) = \mathbb{E}[$*discounted* total reward of starting in state s and executing policy π forever$]$

$$= \mathbb{E}\left[R\big(s_0 = s, \pi(s_0)\big) + \gamma R\big(s_1, \pi(s_1)\big) + \gamma^2 R\big(s_2, \pi(s_2)\big) + \cdots\right]$$

$$= \sum_{t=0}^{\infty} \gamma^t \mathbb{E}\big[R\big(s_t, \pi(s_t)\big)\big]$$

where $0 < \gamma < 1$ is some discount factor for future rewards

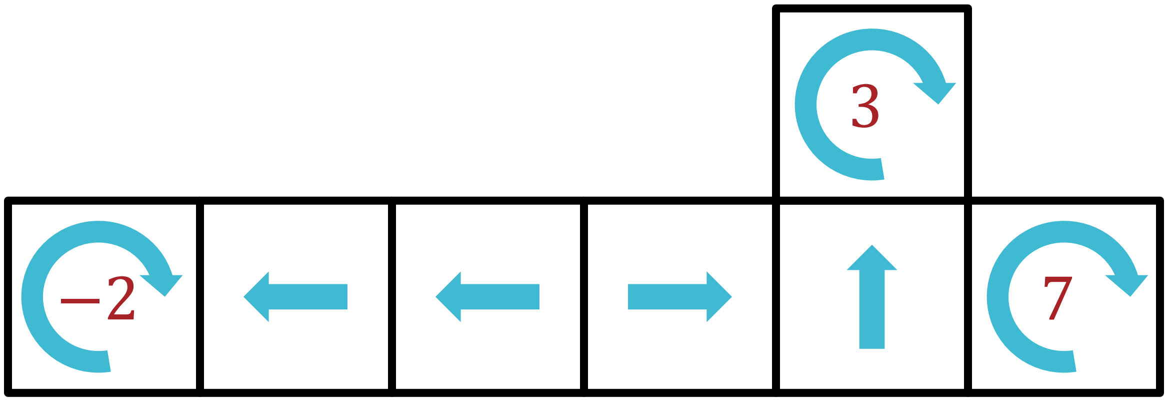
Value function: example



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

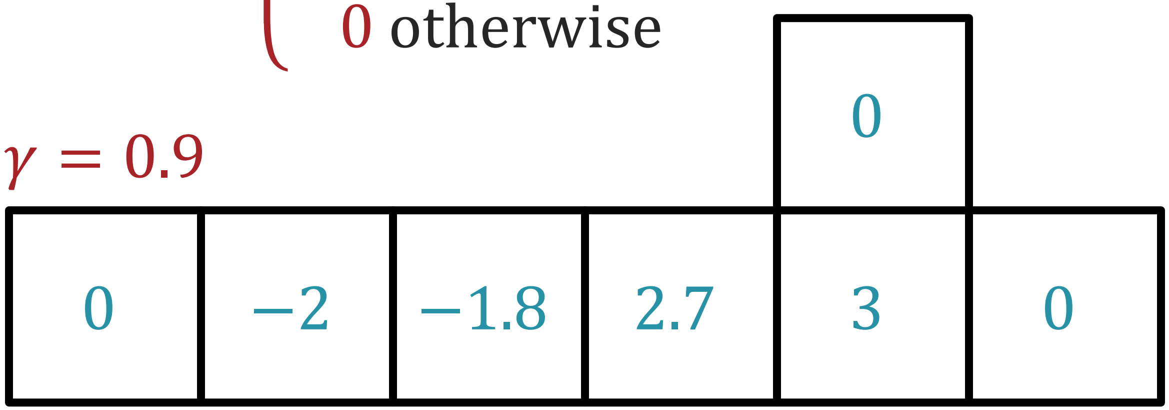
$\gamma = 0.9$

Value function: example

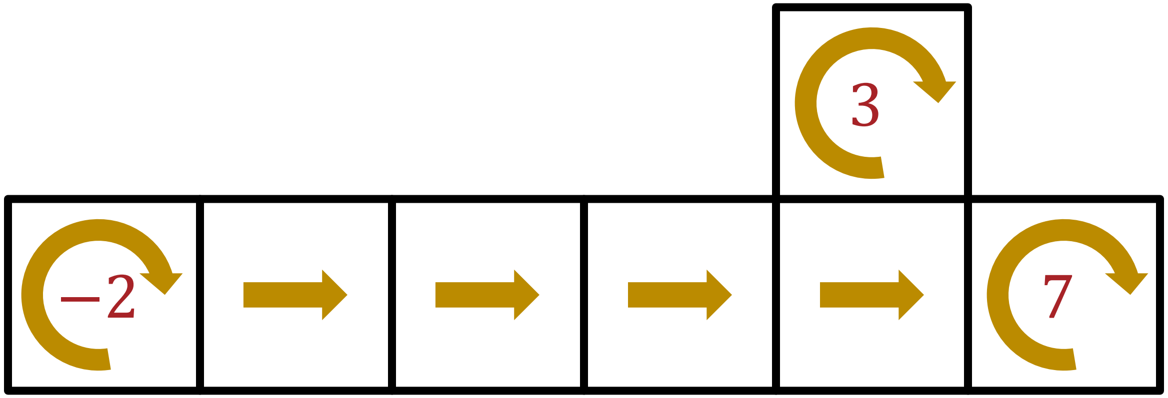


$$R(s,a) = \begin{cases} -2 & \text{if entering state 0 (safety)} \\ 3 & \text{if entering state 5 (field goal)} \\ 7 & \text{if entering state 6 (touch down)} \\ 0 & \text{otherwise} \end{cases}$$

$\gamma = 0.9$



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 – deterministic reward

- $V^{\pi}(s) = \mathbb{E}[$discounted total reward of starting in state s and executing policy π forever$]$

$$= \mathbb{E}[R(s_0, \pi(s_0)) + \gamma R(s_1, \pi(s_1)) + \gamma^2 R(s_2, \pi(s_2)) + \cdots \mid s_0 = s]$$

$$= R(s, \pi(s)) + \gamma \mathbb{E}[R(s_1, \pi(s_1)) + \gamma R(s_2, \pi(s_2)) + \ldots \mid s_0 = s]$$

$$= R(s, \pi(s)) + \gamma \sum_{s_1 \in \mathcal{S}} p(s_1 \mid s, \pi(s)) \big(R(s_1, \pi(s_1)) + \gamma \mathbb{E}[R(s_2, \pi(s_2)) + \cdots \mid s_1]\big)$$

$$\boxed{V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s_1 \in \mathcal{S}} p(s_1 \mid s, \pi(s)) V^{\pi}(s_1)}$$

Bellman equations

Optimal value function and policy

- Optimal value function:

$$V^*(s) = \max_{a \in \mathcal{A}} \left[R(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s,a) V^*(s')\right]$$

  - System of $|\mathcal{S}|$ equations and $|\mathcal{S}|$ variables – nonlinear!

- Optimal policy:

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} \underbrace{R(s,a)}_{\text{Immediate reward}} + \underbrace{\gamma \sum_{s' \in \mathcal{S}} p(s' \mid s,a) V^*(s')}_{\text{Expected (Discounted) Future reward}}$$

- Insight: if you know the optimal value function, you can solve for the optimal policy!

Value iteration

- Inputs: $R(s, a), p(s' \mid s, a), 0 < \gamma < 1$
- Initialize $V^{(0)}(s) = 0 \; \forall \; s \in \mathcal{S}$ (or randomly) and set $t = 0$
- While not converged, do:
  - For $s \in \mathcal{S}$

$$V^{(t+1)}(s) \leftarrow \max_{a \in \mathcal{A}} \underbrace{\left[R(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) V^{(t)}(s')\right]}_{Q(s, a)}$$

  - $t = t + 1$
- For $s \in \mathcal{S}$

$$\pi^*(s) \leftarrow \underset{a \in \mathcal{A}}{\operatorname{argmax}} \left[R(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) V^{(t)}(s')\right]$$

- Return π^*

Value iteration

- Inputs: $R(s, a)$, $p(s' \mid s, a)$, $0 < \gamma < 1$
- Initialize $V^{(0)}(s) = 0 \ \forall\, s \in \mathcal{S}$ (or randomly) and set $t = 0$
- While not converged, do:
  - For $s \in \mathcal{S}$
    - For $a \in \mathcal{A}$

$$Q(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) V^{(t)}(s')$$

    - $V^{(t+1)}(s) \leftarrow \max_{a \in \mathcal{A}} Q(s, a)$
  - $t = t + 1$
- For $s \in \mathcal{S}$

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

- Return π^*

Poll

- What is the runtime per iteration?
    A. O(1)
    B. $|S||A|$
    C. $|S||A|^2$
    D. $|A||S|^2$
    E. $|A|^2|S|^2$

Value iteration: convergence

- Runtime per iteration: O(|S|²|A|)

Theorem 1: Value function convergence

V will converge to V^* if each state is "visited" infinitely often (Bertsekas, 1989)

Theorem 2: Convergence criterion

if $\max_{s \in \mathcal{S}} \left|V^{(t+1)}(s) - V^{(t)}(s)\right| < \epsilon$,

then $\max_{s \in \mathcal{S}} \left|V^{(t+1)}(s) - V^{*}(s)\right| < \frac{2\epsilon\gamma}{1-\gamma}$ (Williams & Baird, 1993)

Theorem 3: Policy convergence

The "greedy" policy, $\pi(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q(s,a)$, converges to the optimal π^* in a finite number of iterations, often before the value function has converged! (Bertsekas, 1987)

Policy iteration

- Can we learn the policy directly, instead of first learning the value function?

- Inputs: $R(s, a)$, $p(s' \mid s, a)$, $0 < \gamma < 1$
- Initialize π randomly
- While not converged, do:
  - Solve the Bellman equations defined by policy π

Now linear!

$$V^{\pi}(\mathrm{s}) = R\big(s, \pi(s)\big) + \gamma \sum_{s' \in \mathcal{S}} p\big(s' \mid s, \pi(s)\big) V^{\pi}(s')$$

  - Update π

$$\pi(s) \leftarrow \underset{a \in \mathcal{A}}{\operatorname{argmax}}\; R(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s' \mid s, a) V^{\pi}(s')$$

- Return π

Policy iteration: convergence

- Runtime per iteration: $O(|S|^2|A| + |S|^3)$

Poll

- How many policies are there?
  A. $|S|+|A|$
  B. $|S||A|$
  C. $|S|^{|A|}$
  D. $|A|^{|S|}$

Policy iteration: convergence

- Runtime per iteration: $O(|S|^2|A| + |S|^3)$

- Number of policies: $|S|^{|A|}$
- Policy improves each iteration
- Thus, the number of iterations needed to converge is bounded!

- Empirically, policy iteration requires fewer iterations than value iteration.

Next Questions

- How to handle unknown state transition and reward functions?

- How to handle continuous states and actions?