Reinforcement Learning I

Aarti Singh

Machine Learning 10-701
Apr 3, 2023

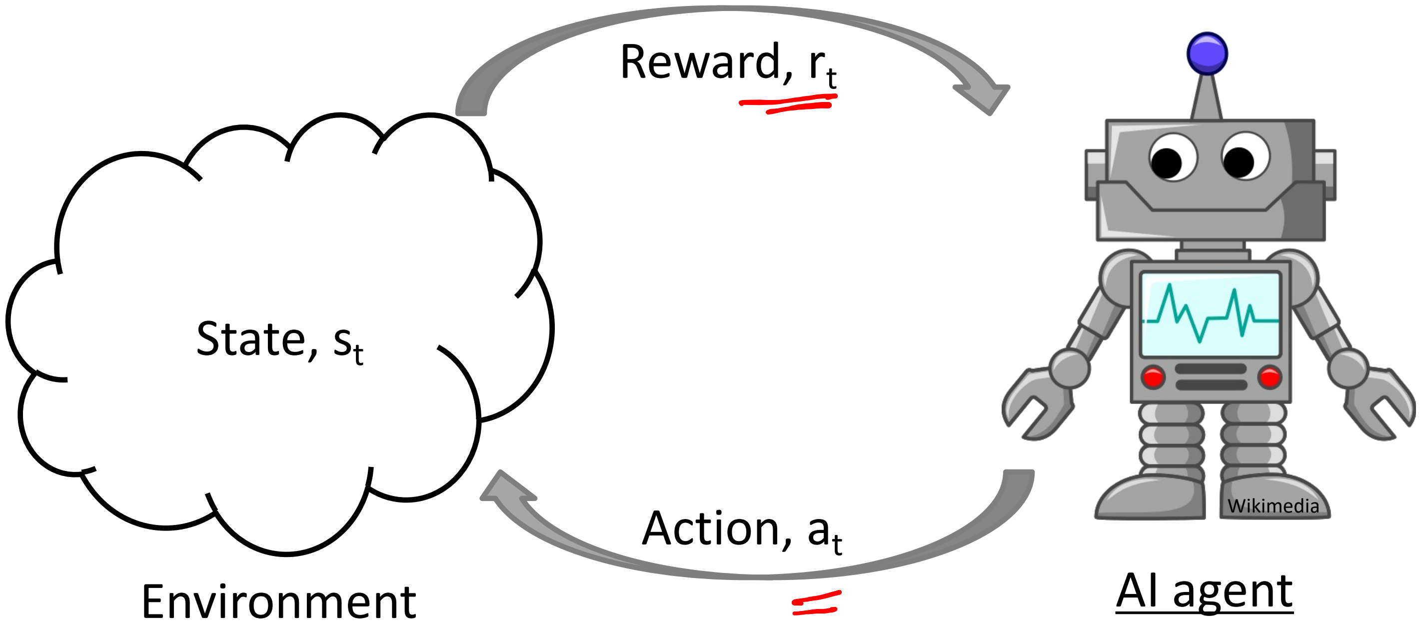
Slides courtesy: Henry Chai, Eric Xing

Learning Tasks

- Supervised learning - $\mathcal{D} = \{(\underline{x}^{(i)}, \underline{y}^{(i)})\}_{i=1}^{N}$ / iid $\sim$ $p(x,y)$
  - Regression - $y^{(i)} \in \mathbb{R}$
  - Classification - $y^{(i)} \in \{1, \ldots, \underline{C}\}$
- Unsupervised learning - $\mathcal{D} = \{\underline{\underline{x}}^{(i)}\}_{i=1}^{N}$ / iid $\sim$ $p(x)$ density estimation /distribution
  - Clustering
  - Dimensionality reduction
- Reinforcement learning - $\mathcal{D} = \{s^{(t)}, a^{(t)}, r^{(t)}\}_{\underline{\underline{t}}=1}^{T}$ sequential "t"

  states actions rewards

RL setup



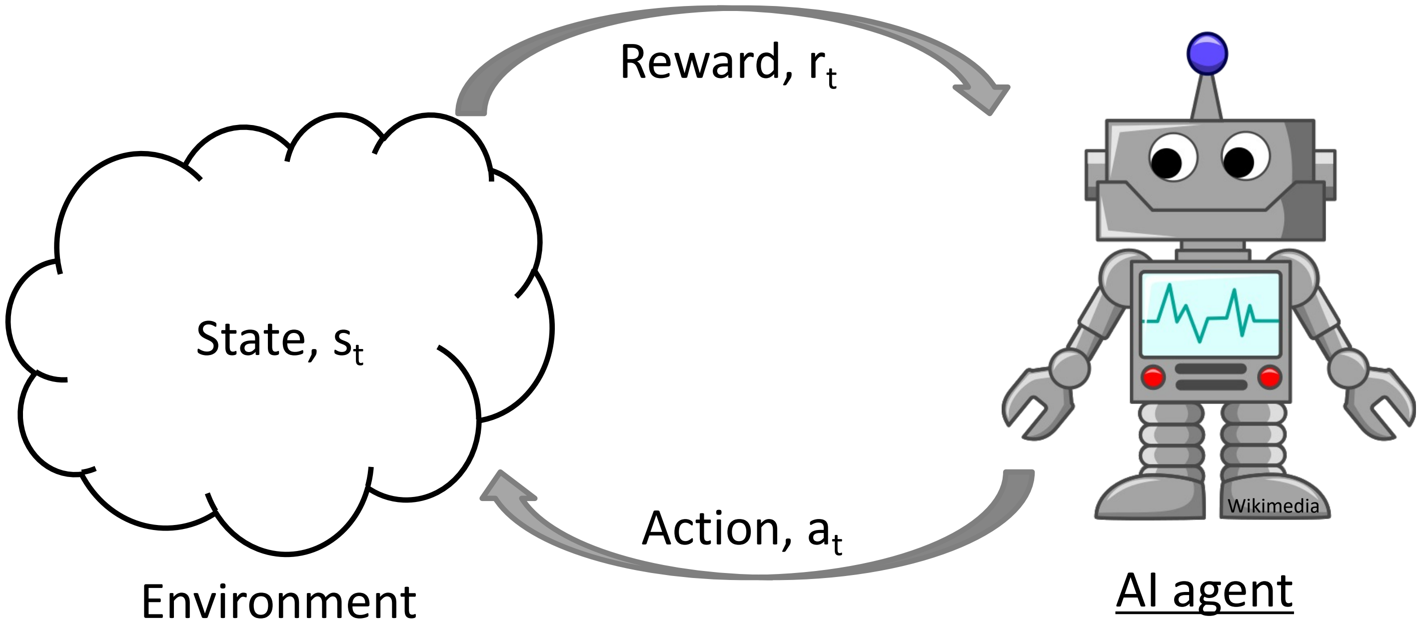
Agent chooses **actions** which can depend on past

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 (states are not controlled)

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

RL setup

deterministic (can be stochastic)

- Policy, $\pi : \mathcal{S} \to \mathcal{A}$
  - Specifies an action to take in *every* state
- Value function, V^{π}: $\mathcal{S} \to \mathbb{R}$ $V^{\pi}(s)$
  - 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

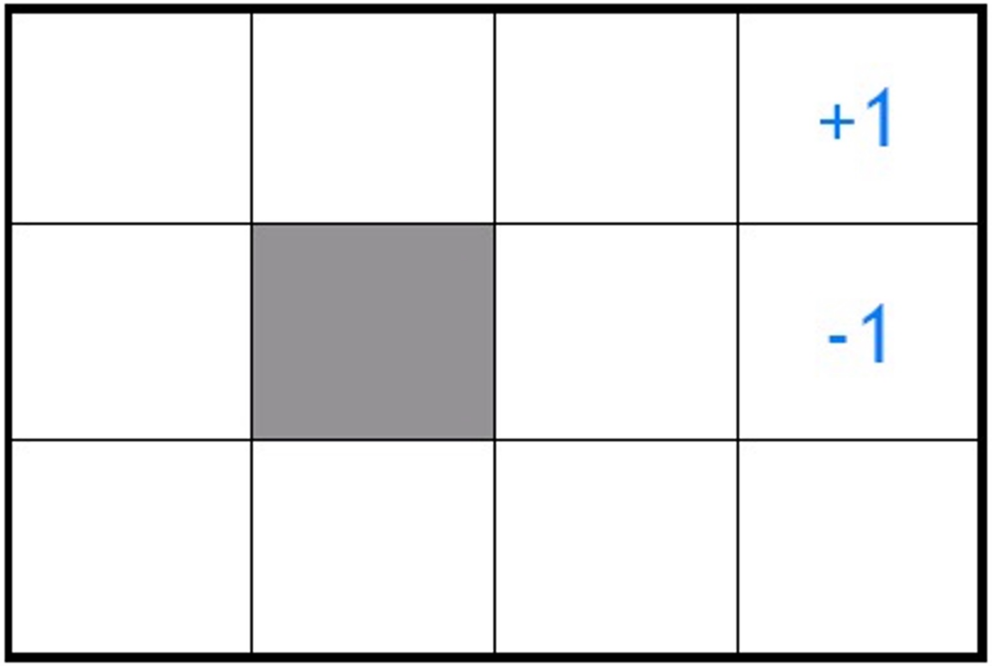
$\mathcal{S}$ = all empty squares in the grid

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

Deterministic transitions

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

Terminate after receiving either reward



RL example - gridworld

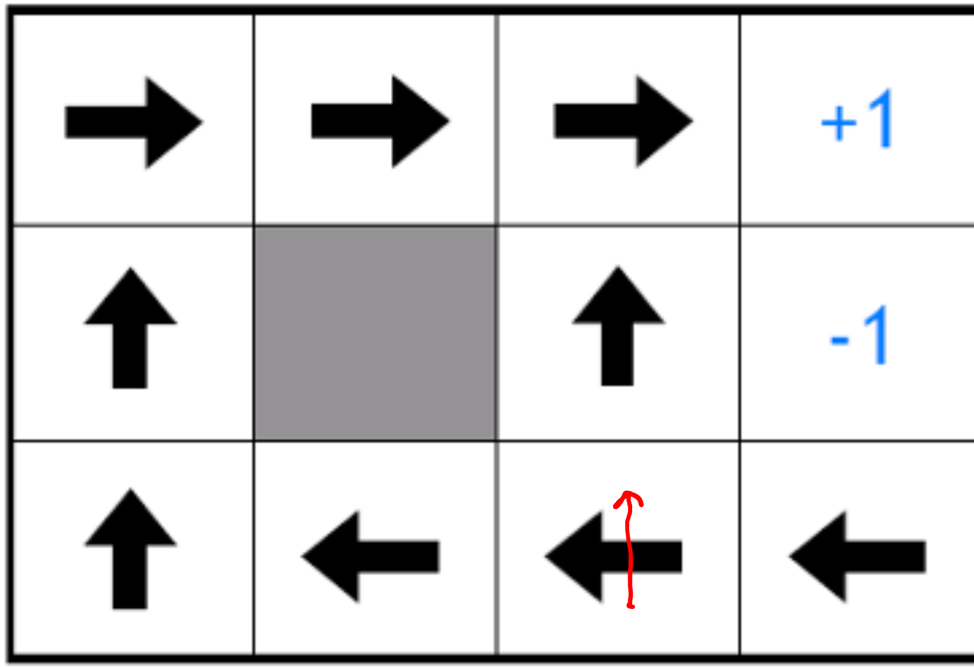
$\mathcal{S}$ = all empty squares in the grid

$\mathcal{A}$ = {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

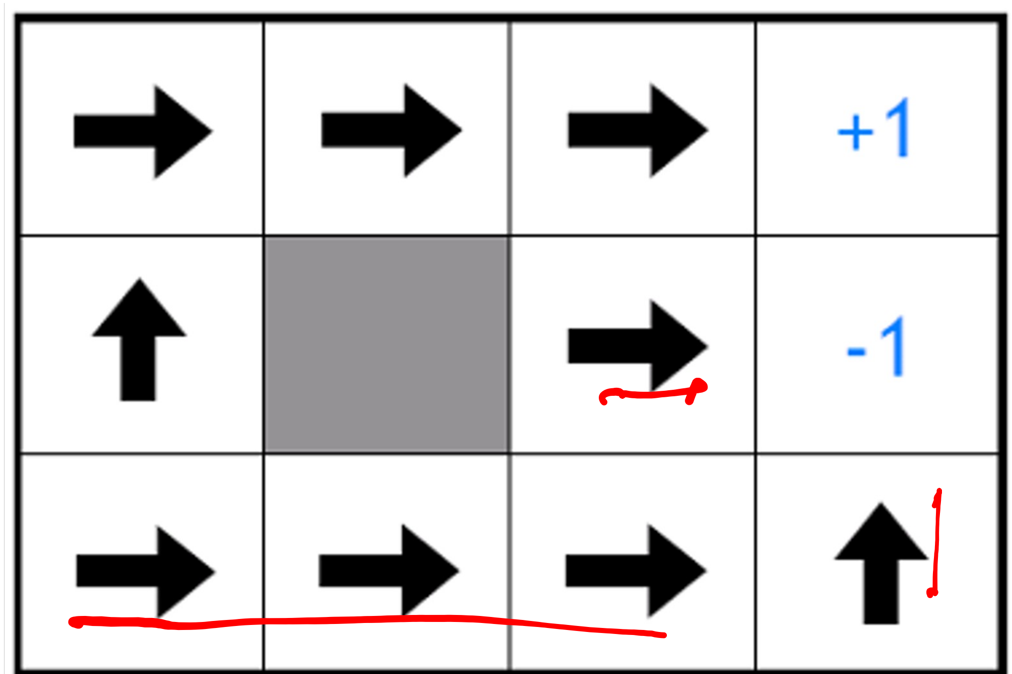
$\mathcal{S}$ = all empty squares in the grid

$\mathcal{A}$ = {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

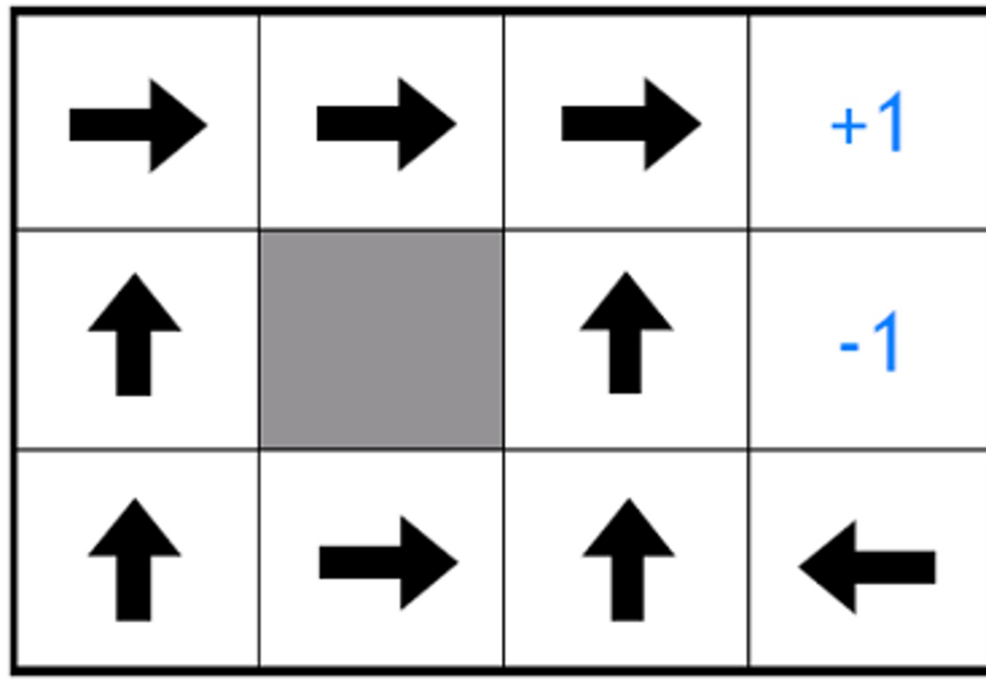
$\mathcal{S}$ = all empty squares in the grid

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

Deterministic transitions

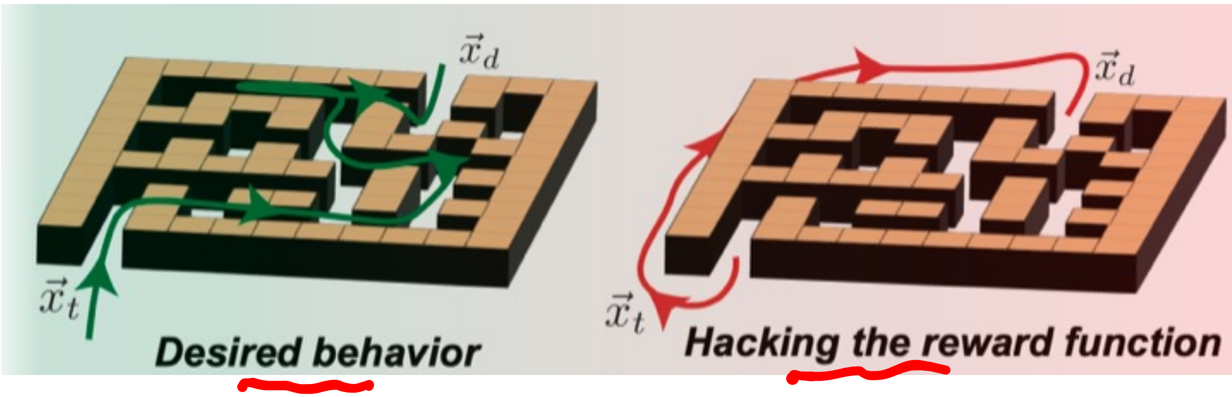
Rewards of +1 and -1 for entering the labelled squares

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 $s_{t+1} \sim p(s' \mid s_t, a_t)$

stochastic policy
$a_t \sim \pi(s_t)$

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

Discounted Reward

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

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

Why discount?

- Mathematically tractable – total reward doesn't explode

  1 + 1 + 1 + … = ∞ but 1 + 0.8*1 + $(0.8)^2$*1 + … = 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	2	1
1	0	0
1	???	3
???	???	2
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???
???	???	???

RL: objective function

$r(s,a)$

- 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$]$

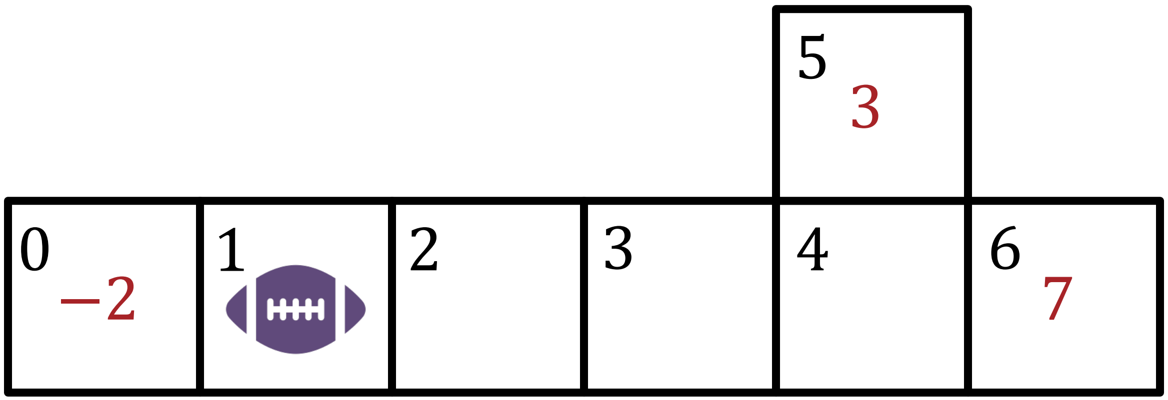
$$= \mathbb{E}\left[R\big(s_0 = s, \pi(s_0)\big) + \gamma R\big(s_1, \pi(s_1)\big) + \gamma^2 R\big(s_2, \pi(s_2)\big) + \cdots\right]$$

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

a_t

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

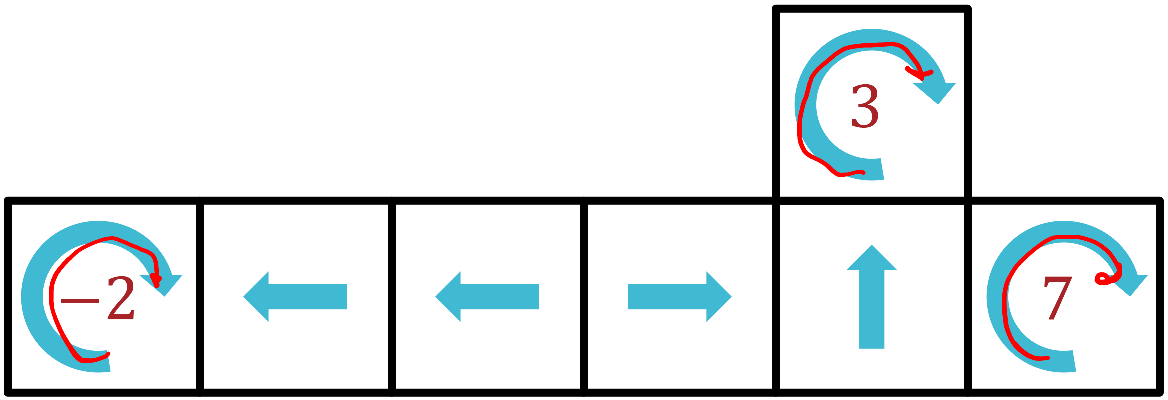
Value function: example



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

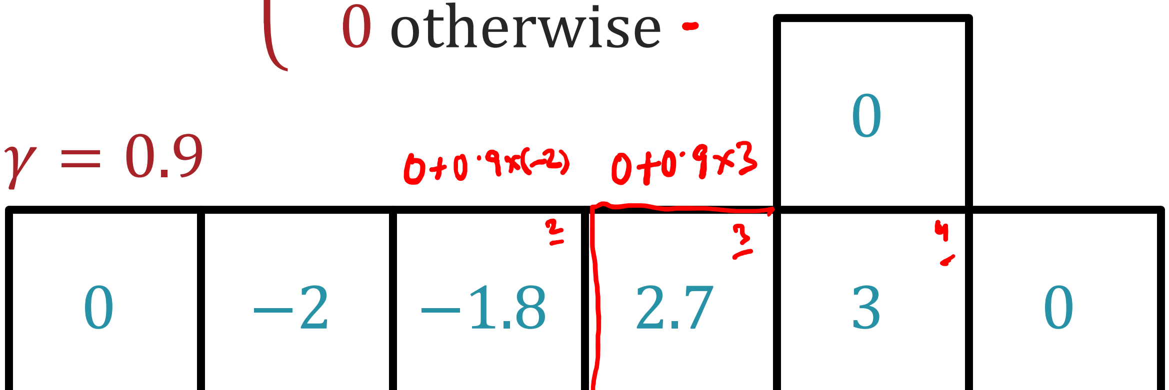
$\gamma = 0.9$

Value function: example

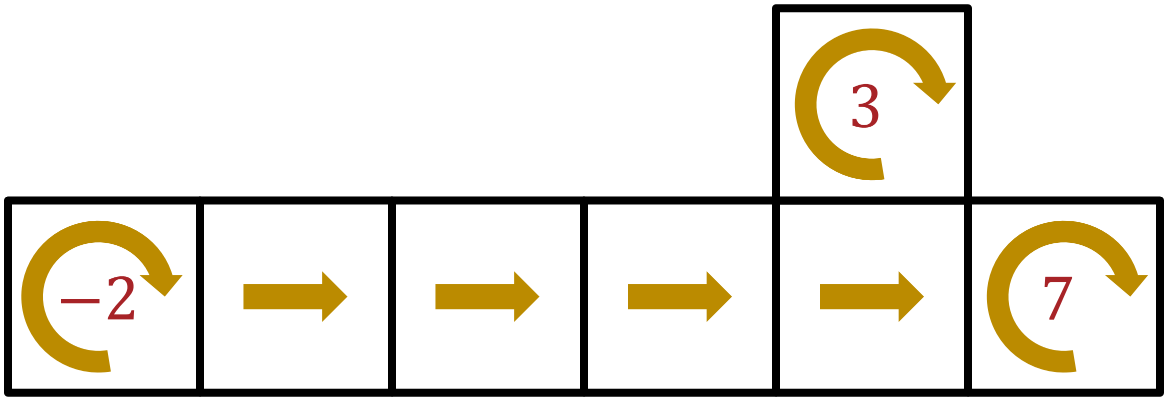


$$R(s,a) = \begin{cases} -2 \text{ if entering state 0 (safety)} \\ 3 \text{ if entering state 5 (field goal)} \\ 7 \text{ if entering state 6 (touch down)} \\ 0 \text{ otherwise} \end{cases}$$

$\gamma = 0.9$

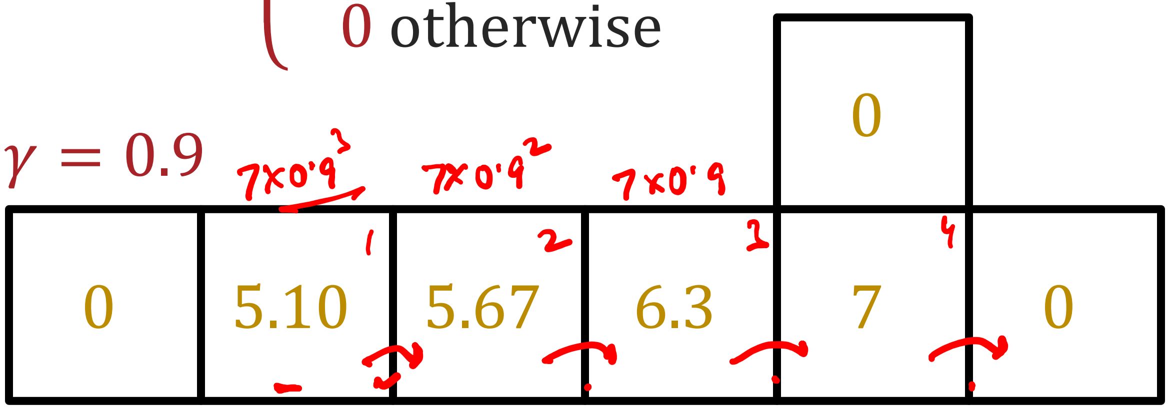


Value function: example



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

$\gamma = 0.9$



Value function – deterministic reward

$p(s' \mid s,a)$

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

$\mathbb{E}[f(z)] = \sum_z p(z) f(z)$

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

$\gamma^2 R(s_2, \pi(s_2)) + \cdots$

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

$$= R(s, \pi(s)) + \gamma \sum_{s_1 \in \mathcal{S}} p(s_1 \mid s, \pi(s)) \underbrace{\left(R(s_1, \pi(s_1)) + \gamma \mathbb{E}[R(s_2, \pi(s_2)) + \cdots \mid s_1]\right)}_{V^\pi(s_1)}$$

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

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 $s \in \mathcal{S}$

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

- Return π^*