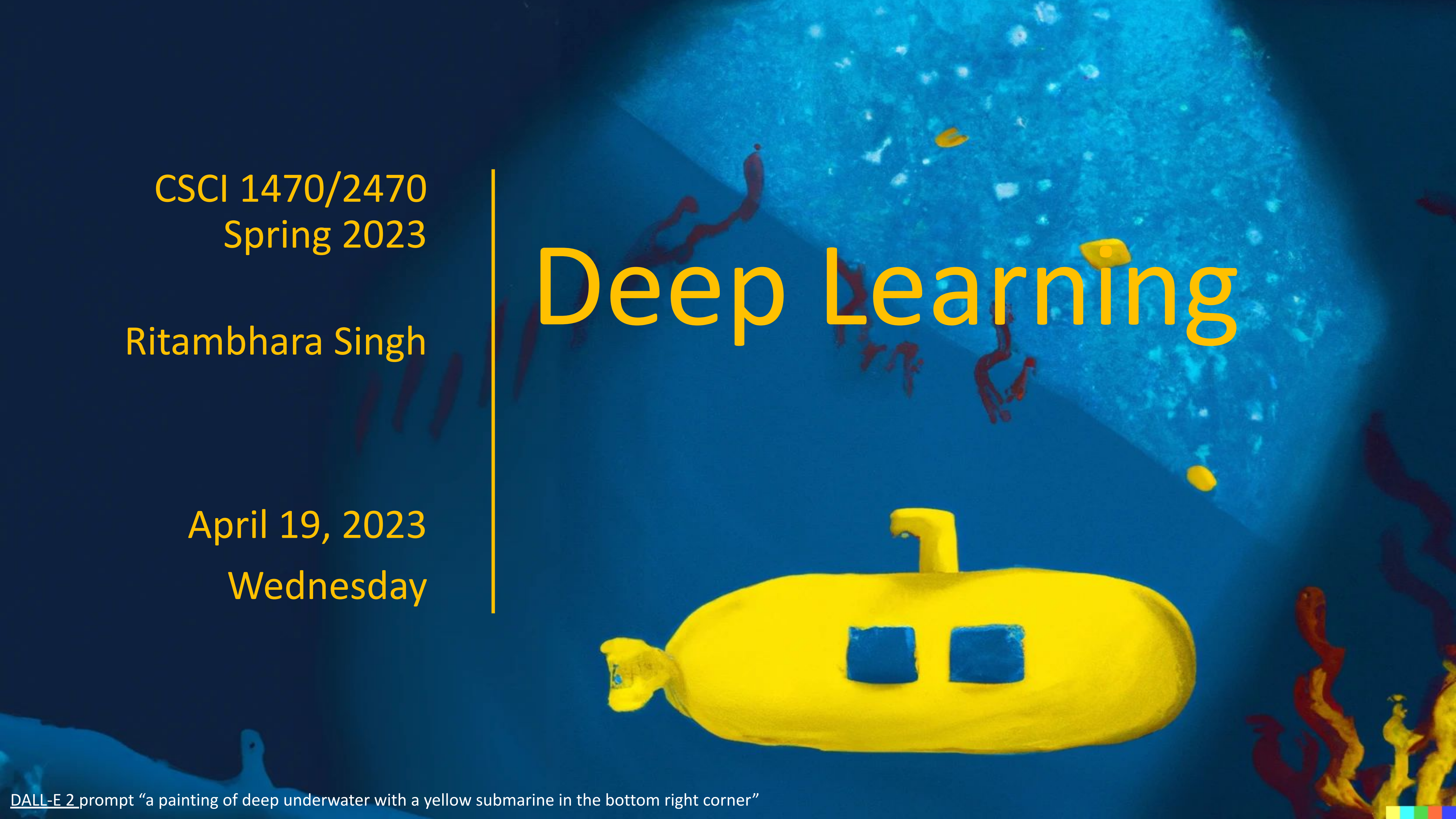
https://www.alignmentawards.com/

CSCI 1470/2470
Spring 2023

Ritambhara Singh

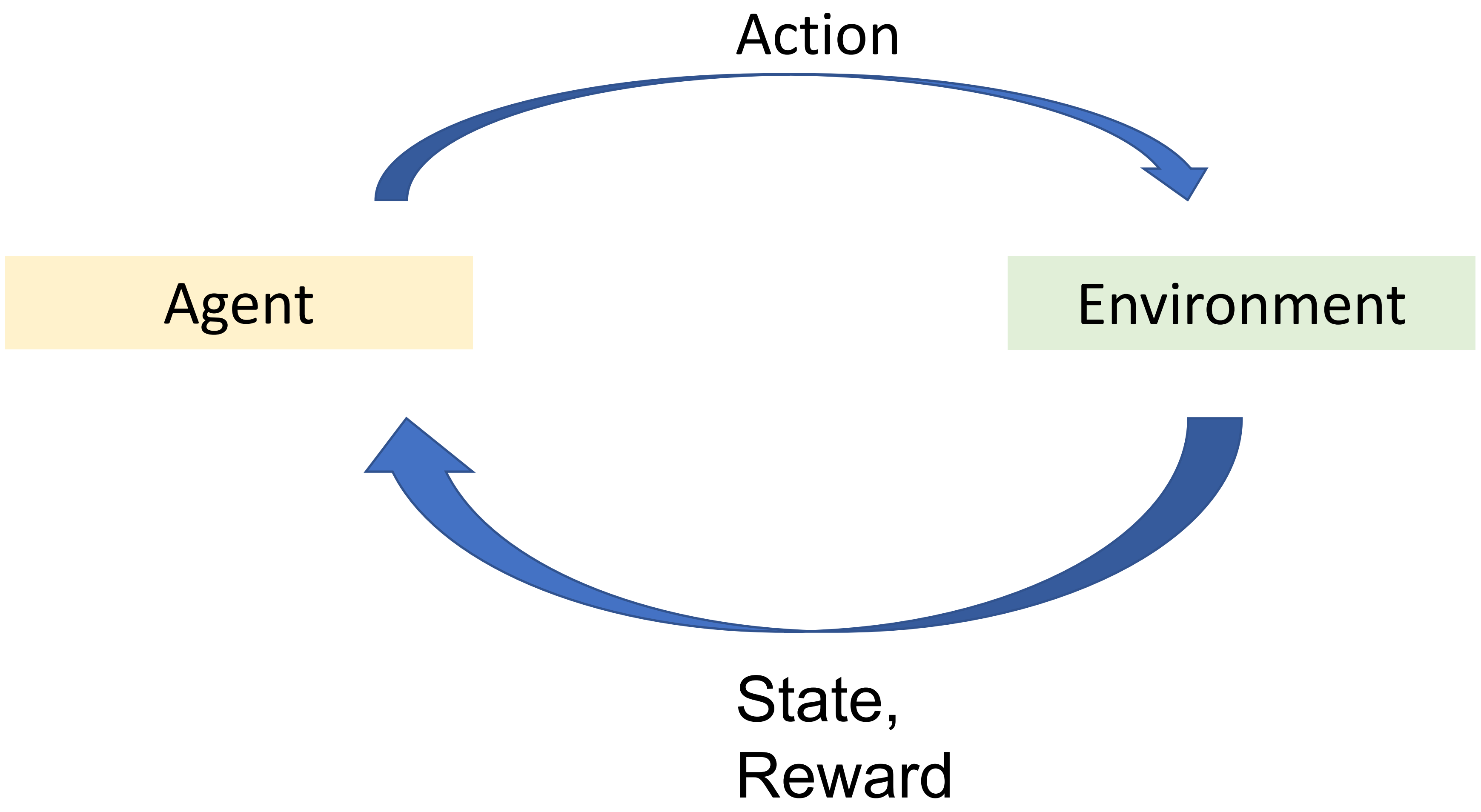
April 19, 2023
Wednesday

Deep Learning



DALL-E 2 prompt “a painting of deep underwater with a yellow submarine in the bottom right corner”

Recap: RL framework



Recap: Markov Decision Process (MDP)

- States – set of possible situations in a world, denoted S
- Actions – set of different actions an agent can take, denoted A
- Transition function – returns the probability of transitioning to state s' after taking action a in state s, denoted $T(s, a, s')$
- Reward function – returns the reward received by the agent for transitioning to state s' after taking action a in state s, denoted $R(s, a, s')$

Recap: Policy Function

- What action should the agent take in a given state?
- Concretely:
- $\pi: S \to A$
- Input: state $s \in S$
- Output: action to be chosen in that state
- $\pi(s) = a$ means in state s, take action a

Recap: Goal of RL

- Learn optimal policy π^* that maximizes the expected future cumulative reward
  - "Expected" because transitions can be non-deterministic
- Solving MDPs ←→ find this optimal policy!

Organizing RL problems/algorithms

	Know T and R	Don't know T and R
Simple/discrete	Value iteration	Q-Learning
Complex/continuous		Deep Q-Networks REINFORCE Actor-Critic

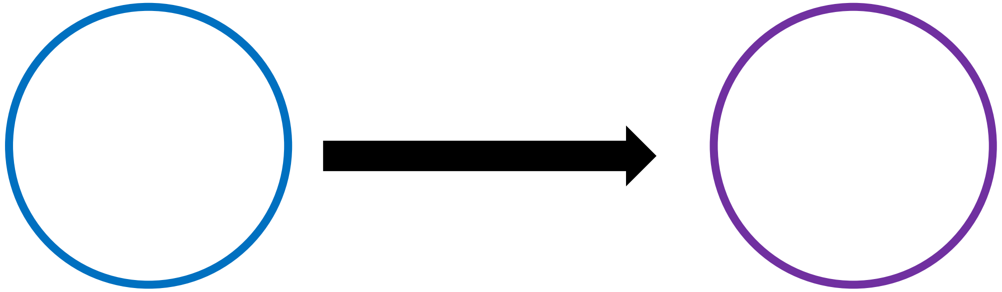
For a more complete taxonomy of RL algorithms, see https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html#citations-below

Value Iteration

Value Function

What would motivate us to move from a state s to s'?

We assign a "value" to each state



Value Function

- Function that returns the “value” of each state

Value Function

- Function that returns the “value” of each state
- Value of state s under policy π is the expected return when starting in s and following π

Value Function

- Function that returns the "value" of each state
- Value of state s under policy π is the expected return when starting in s and following π
- "Value of my current state is the total discounted future reward I expect from following a policy π from now on"

Value Function

- Function that returns the "value" of each state
- Value of state s under policy π is the expected return when starting in s and following π
- "Value of my current state is the total discounted future reward I expect from following a policy π from now on"
- $V_\pi : S \to \mathbb{R}$

Value Function

- Function that returns the "value" of each state
- Value of state s under policy π is the expected return when starting in s and following π
- "Value of my current state is the total discounted future reward I expect from following a policy π from now on"
- $V_{\pi}: S \rightarrow \mathbb{R}$
- $V_{\pi}(s) = E[G_t \mid S_t = s]$ for all $s \in S$, where G_t is the return

Value Function

- Function that returns the "value" of each state
- Value of state s under policy π is the expected return when starting in s and following π
- "Value of my current state is the total discounted future reward I expect from following a policy π from now on"
- $V_{\pi}: S \to \mathbb{R}$
- $V_{\pi}(s) = E[G_t \mid S_t = s]$ for all $s \in S$, where G_t is the return
- $\quad\quad = E[R(s,a,s') + \gamma V_{\pi}(s') \mid S_t = s]$

Any questions?

Value Function

- Function that returns the "value" of each state
- Value of state s under policy π is the expected return when starting in s and following π
- "Value of my current state is the total discounted future reward I expect from following a policy π from now on"
- $V_\pi : S \rightarrow \mathbb{R}$
- $V_\pi(s) = E[G_t \mid S_t = s]$ for all $s \in S$, where G_t is the return
- $\quad\quad = E[R(s,a,s') + \gamma V_\pi(s') \mid S_t = s]$
- $V_\pi(s) = \sum_{s' \in S} P(s'|s,a)\ [R(s,a,s') + \gamma V_\pi(s')]$

Expectation across transition probabilities- deals with the potential stochasticity of transitioning to s'

NOTE: recursively defined! Literally "reward agent receives now + value of the next state"

Remember, for now we are dealing with discrete/simple case

Example (made-up) Value Table

State	Value
State #1	0
State #2	1
State #3	-1
State #4	1.9
State #5	10
State #6	-10

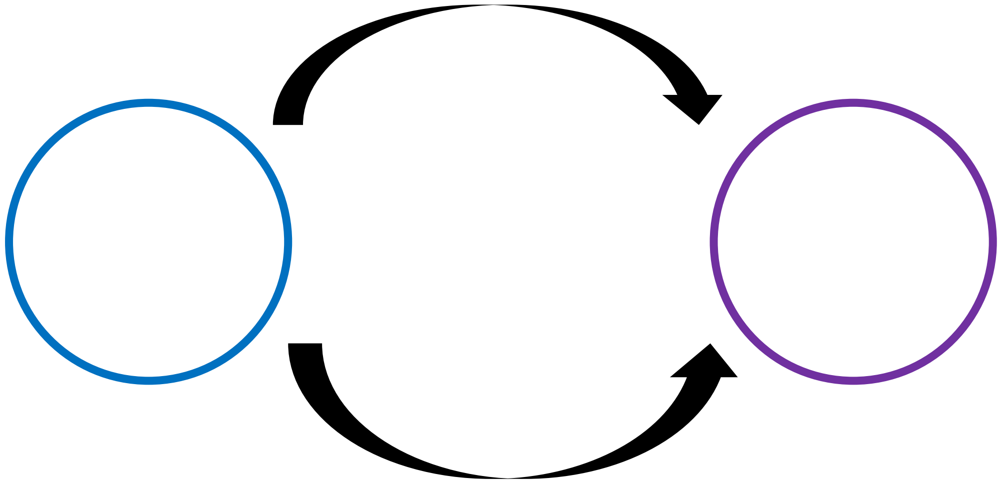
Which is the favorable state?

"If we transition from state #5 using the (our made-up) policy to other states s' the expected total discounted future reward is 10"

Q-function

What if we have multiple actions to take from s to s'?

We assign "value" to each action at a given state



Q-function

- $q_\pi : S \times A \to \mathbb{R}$

Q-function

- $q_\pi: S \times A \to \mathbb{R}$
- $q_\pi(s, a) = E[G_t \mid S_t = s, A_t = a]$ for all $s \in S, a \in A$

Q-function

- $q_\pi: S \times A \to \mathbb{R}$
- $q_\pi(s, a) = E[G_t \mid S_t = s, A_t = a]$ for all $s \in S, a \in A$
- AKA "action-value function"

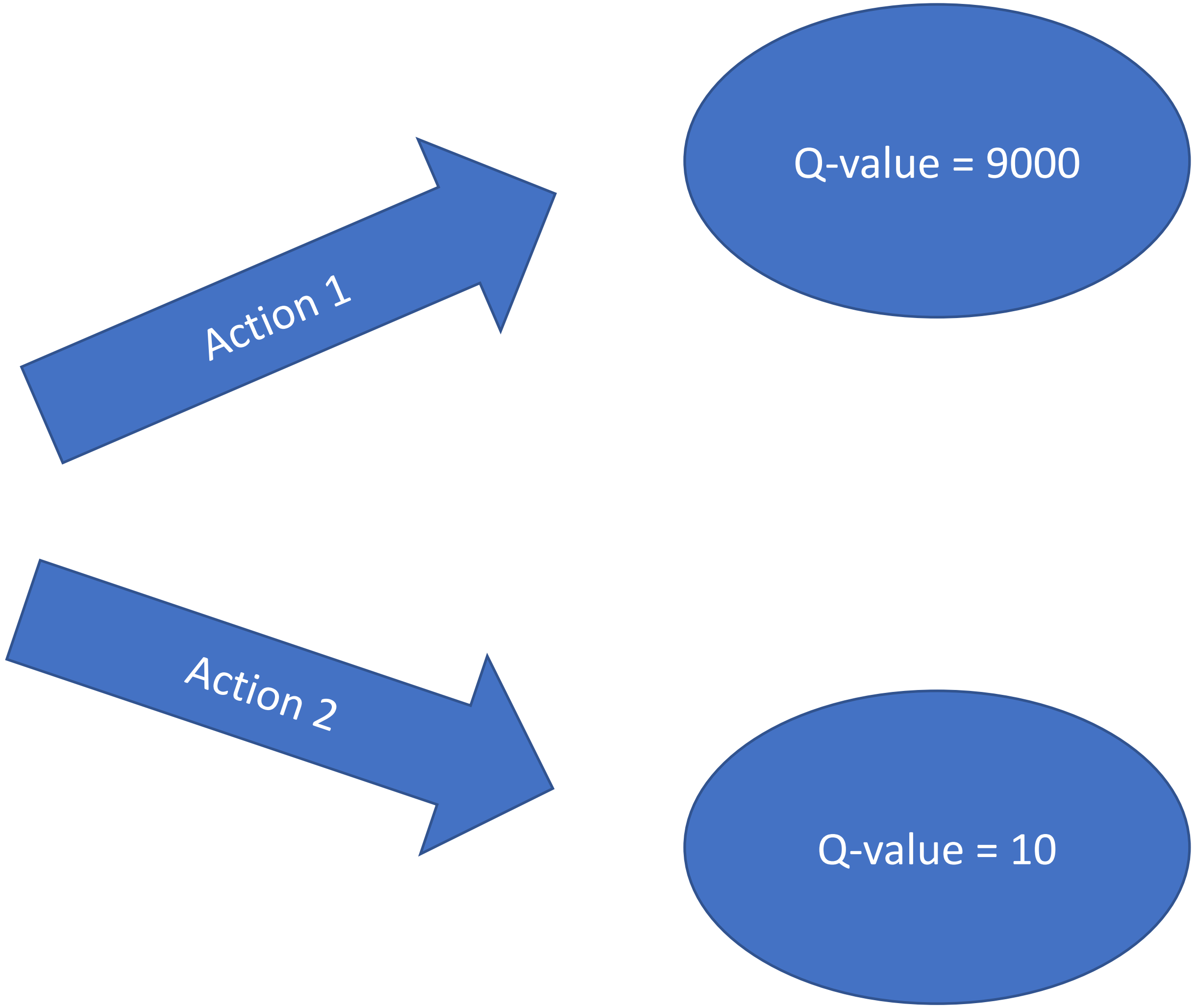
Q-function

- $q_\pi: S \times A \to \mathbb{R}$
- $q_\pi(s, a) = E[G_t \mid S_t = s, A_t = a]$ for all $s \in S, a \in A$
- AKA "action-value function"
- Outputs expected return from taking action a in state s and following policy π thereafter

Q-value Table (made up)

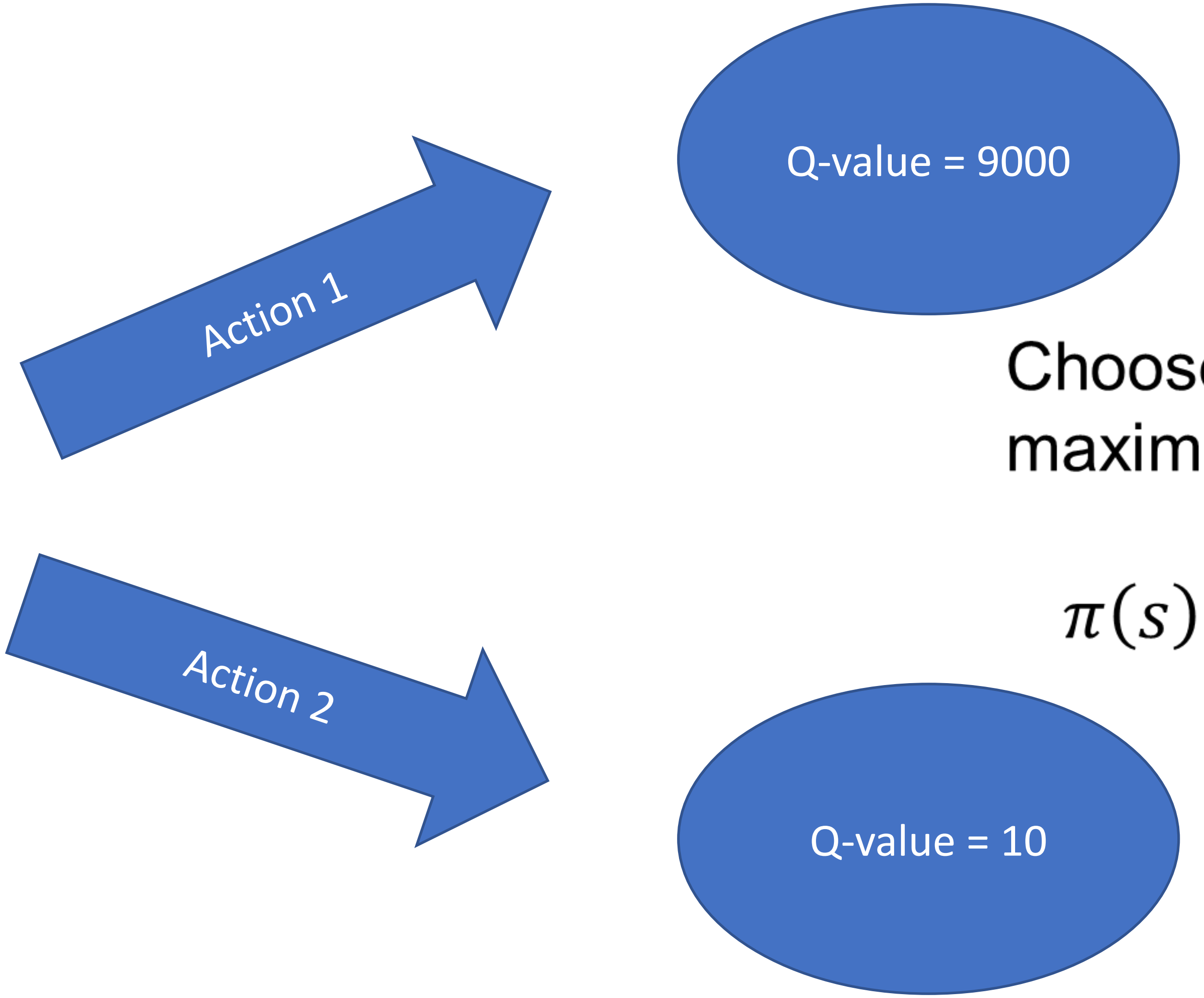
	Action #1	Action #2
State #1	0	-1
State #2	0.1	1
State #3	-1	-10
State #4	0	1.9
State #5	10	0
State #6	-10	-10

How to determine policy from Q-function?



Any ideas?

How to determine policy from Q-function?



Choose the action that maximizes your Q-value!

$$\pi(s) = argmax_a \, Q(s,a)$$

Q-value Table (made up)

	Action #1	Action #2
State #1	0	-1
State #2	0.1	1
State #3	-1	-10
State #4	0	1.9
State #5	10	0
State #6	-10	-10

What actions to pick for each state for the optimal policy?

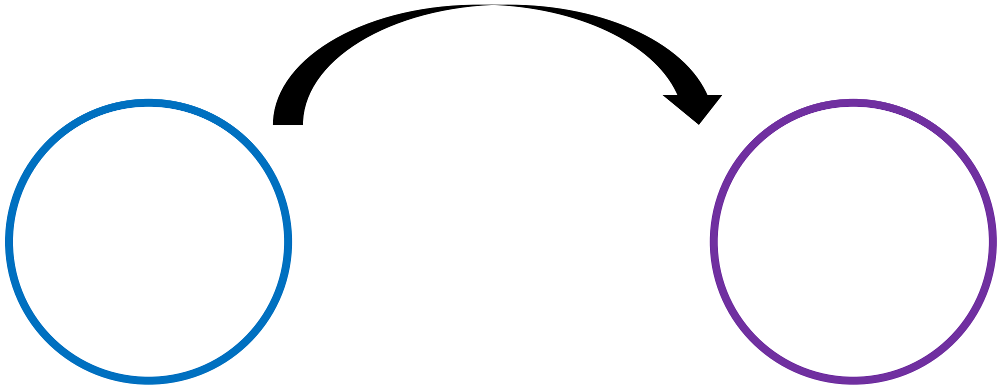
Q-function can be expressed in terms of the V-function

Q-function can be expressed in terms of the V-function

- $Q^{\pi}(s,a) = E[R(s,a,s') + \gamma V^{\pi}(s')]$

Q-function can be expressed in terms of the V-function

- $Q^\pi(s,a) = E[R(s,a,s') + \gamma V^\pi(s')]$
- $Q^\pi(s,a) = \sum_{s' \in S} P(s'|s,a)[R(s,a,s') + \gamma V^\pi(s')\,]$



https://en.wikipedia.org/wiki/Richard_E._Bellman

Q-value and V-value Tables (made up)

	Action #1	Action #2
State #1	0	-1
State #2	0.1	1
State #3	-1	-10
State #4	0	1.9
State #5	10	0
State #6	-10	-10

State	Value
State #1	0
State #2	1
State #3	-1
State #4	1.9
State #5	10
State #6	-10

Any questions?

Optimal policy and value functions

Optimal policy and value functions

- Goal of RL: find optimal policy, π^*

Optimal policy and value functions

- Goal of RL: find optimal policy, π^*
- Approach: learn optimal value functions, V^* and Q^*, then define optimal policy from value functions

How do we actually learn V^* and Q^*?

Value iteration pseudocode

Value iteration pseudocode

1. For all s, set $V(s) := 0$.

Value iteration pseudocode

1. For all s, set $V(s) := 0$.
2. Repeat until convergence:

Value iteration pseudocode

1. For all s, set $V(s) := 0$.
2. Repeat until convergence:
   1. For all s:

Value iteration pseudocode

1. For all s, set $V(s) := 0$.
2. Repeat until convergence:
   1. For all s:
      1. For all a, set $Q(s,a) \coloneqq \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(s')]$

Value iteration pseudocode

1. For all s, set $V(s) := 0$.
2. Repeat until convergence:
   1. For all s:
      1. For all a, set $Q(s,a) \coloneqq \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(s')]$
      2. $V(s) := max_a \, Q(s,a)$

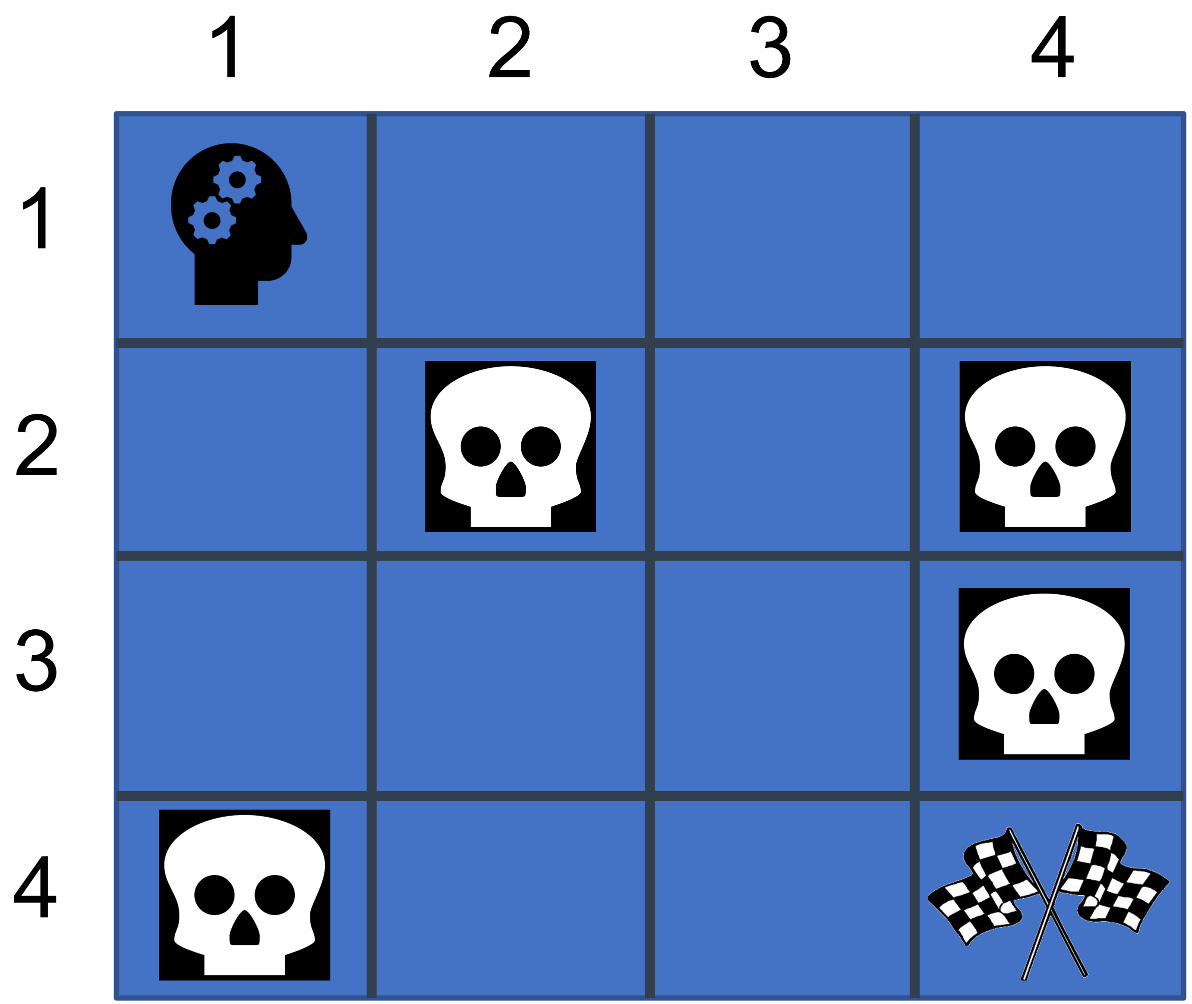
Value iteration pseudocode

1. For all s, set $V(s) := 0$.
2. Repeat until convergence:
   1. For all s:
      1. For all a, set $Q(s,a) := \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(s')]$
      2. $V(s) := max_a \, Q(s,a)$
3. Return Q

How do we get the optimal policy?

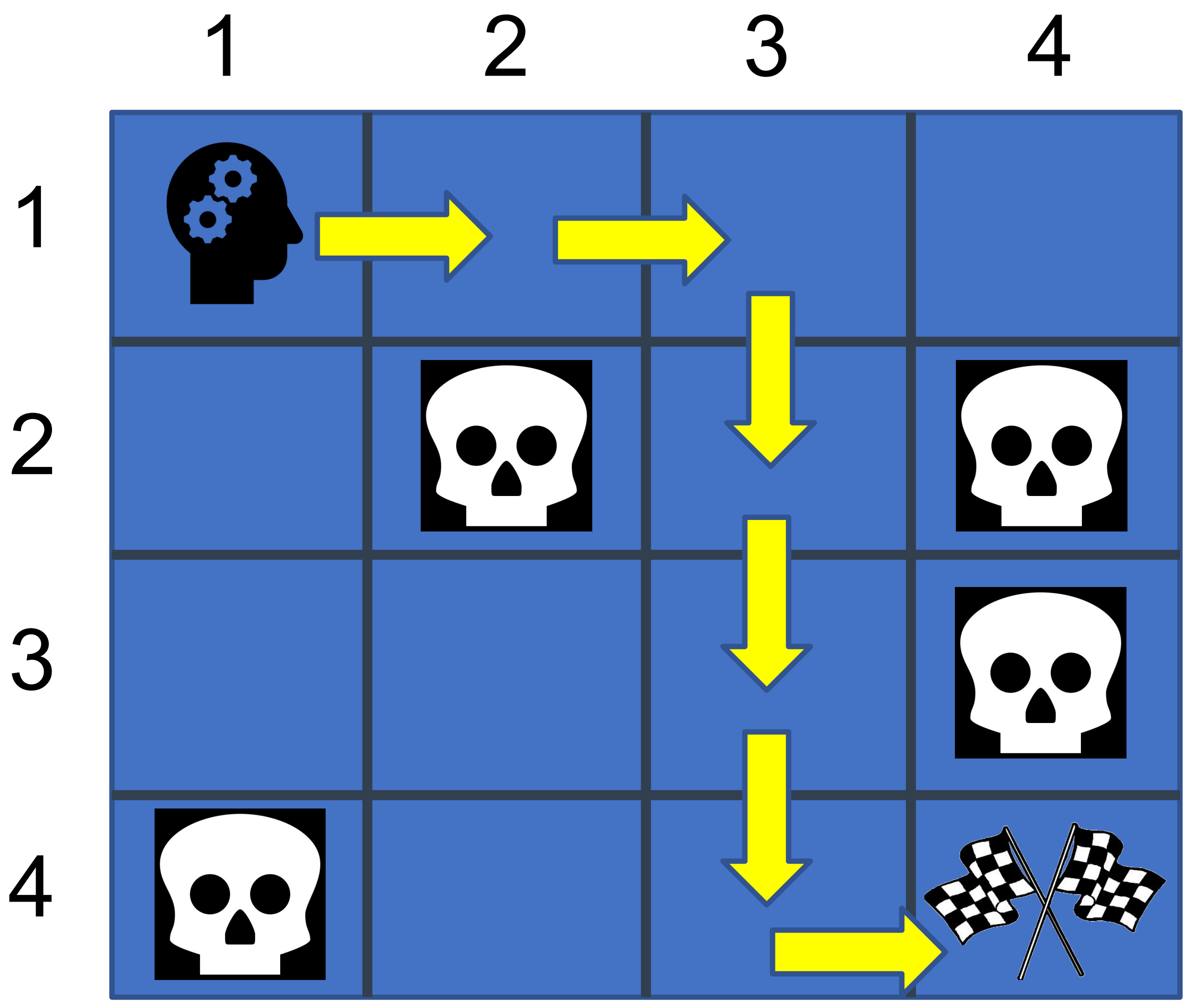
Concrete Example: Frozen Lake Problem

Frozen Lake Problem



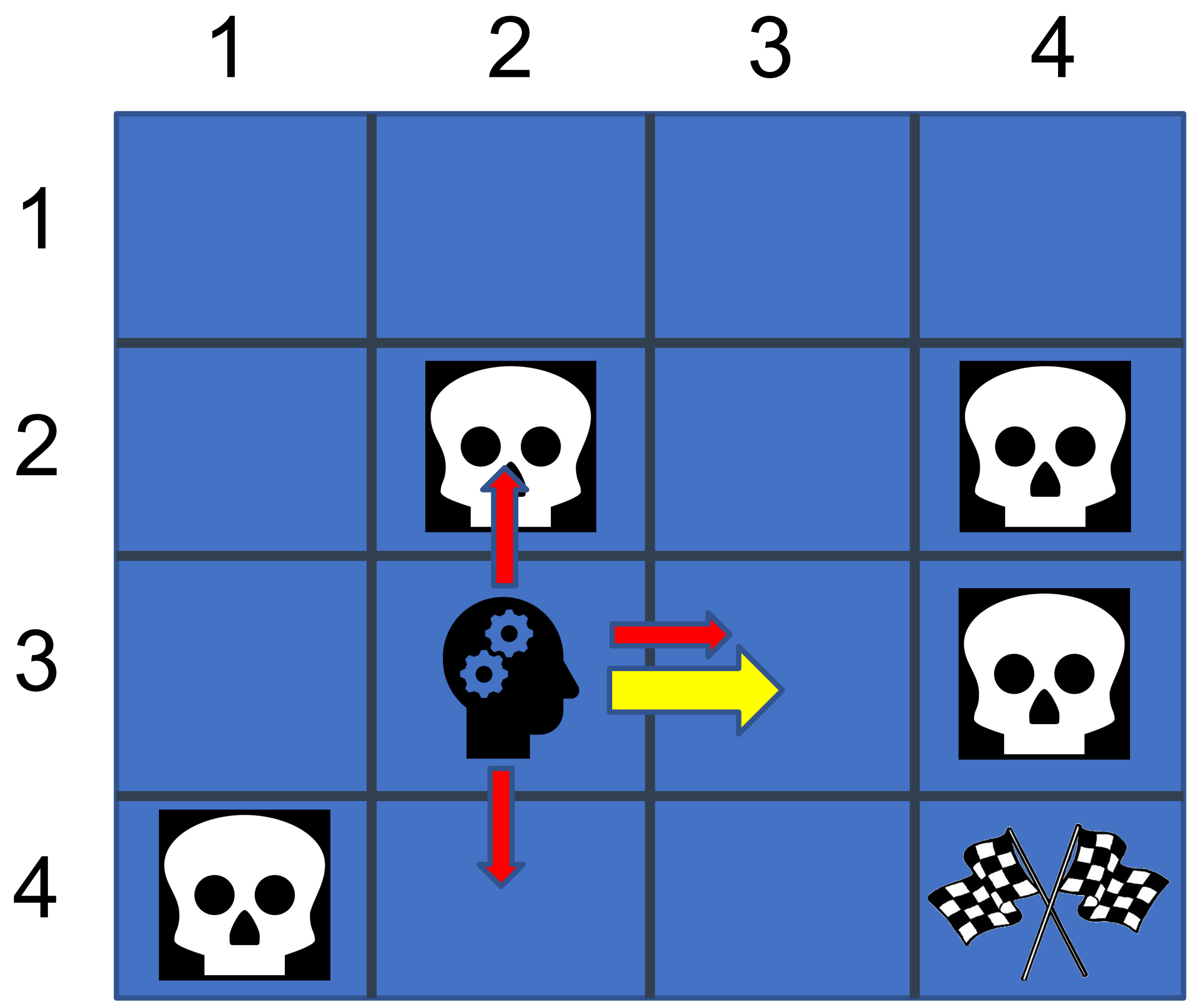
- Agent starts in top left corner
- Goal: Reach the bottom right without falling into any of the holes (skulls)
- Game terminates when agent falls into hole or reaches goal

Optimal policy is easy, right?



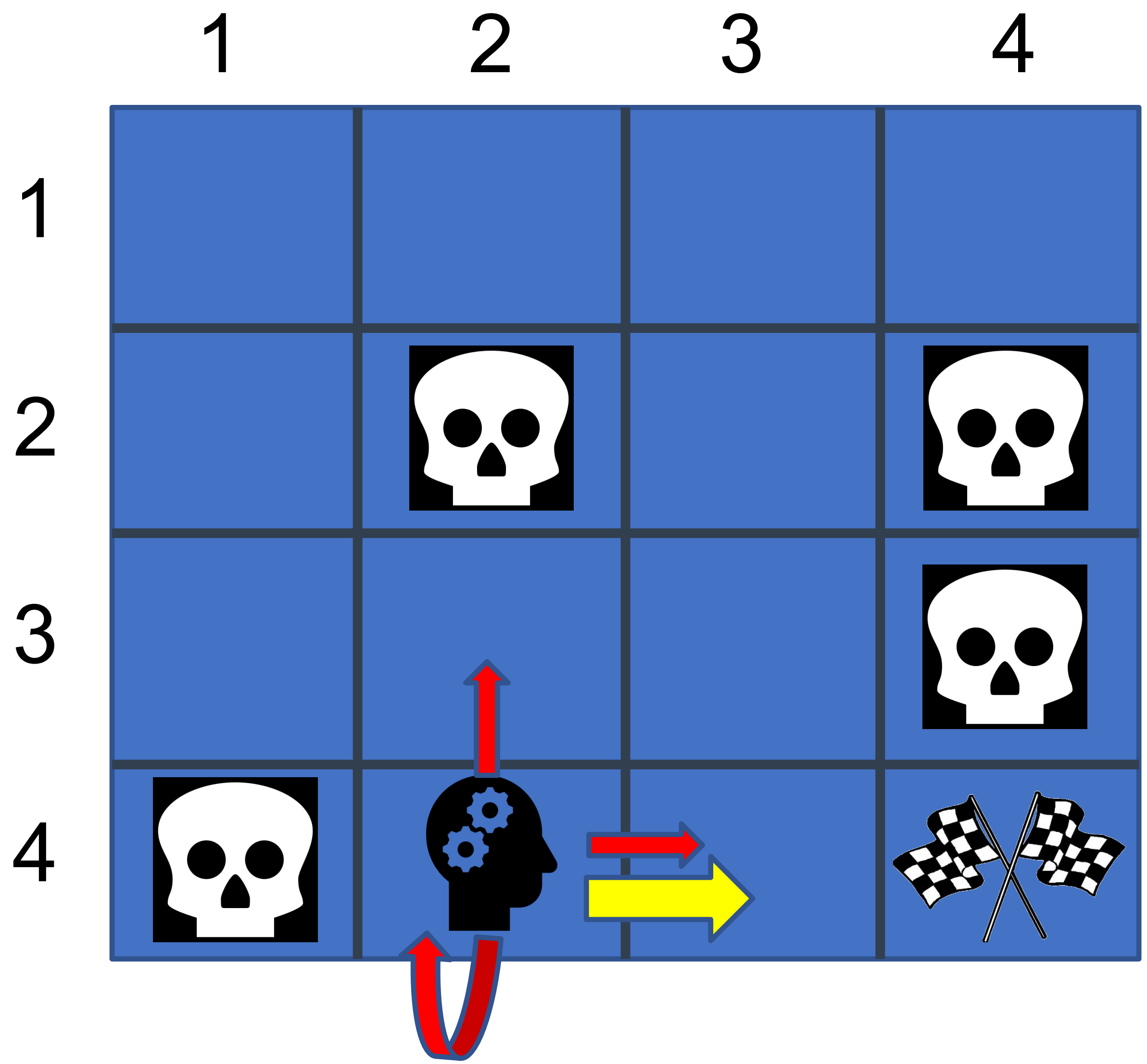
- Multiple optimal policies, actually
- Solve using shortest path algorithm

Not quite - frozen lakes are slippery!



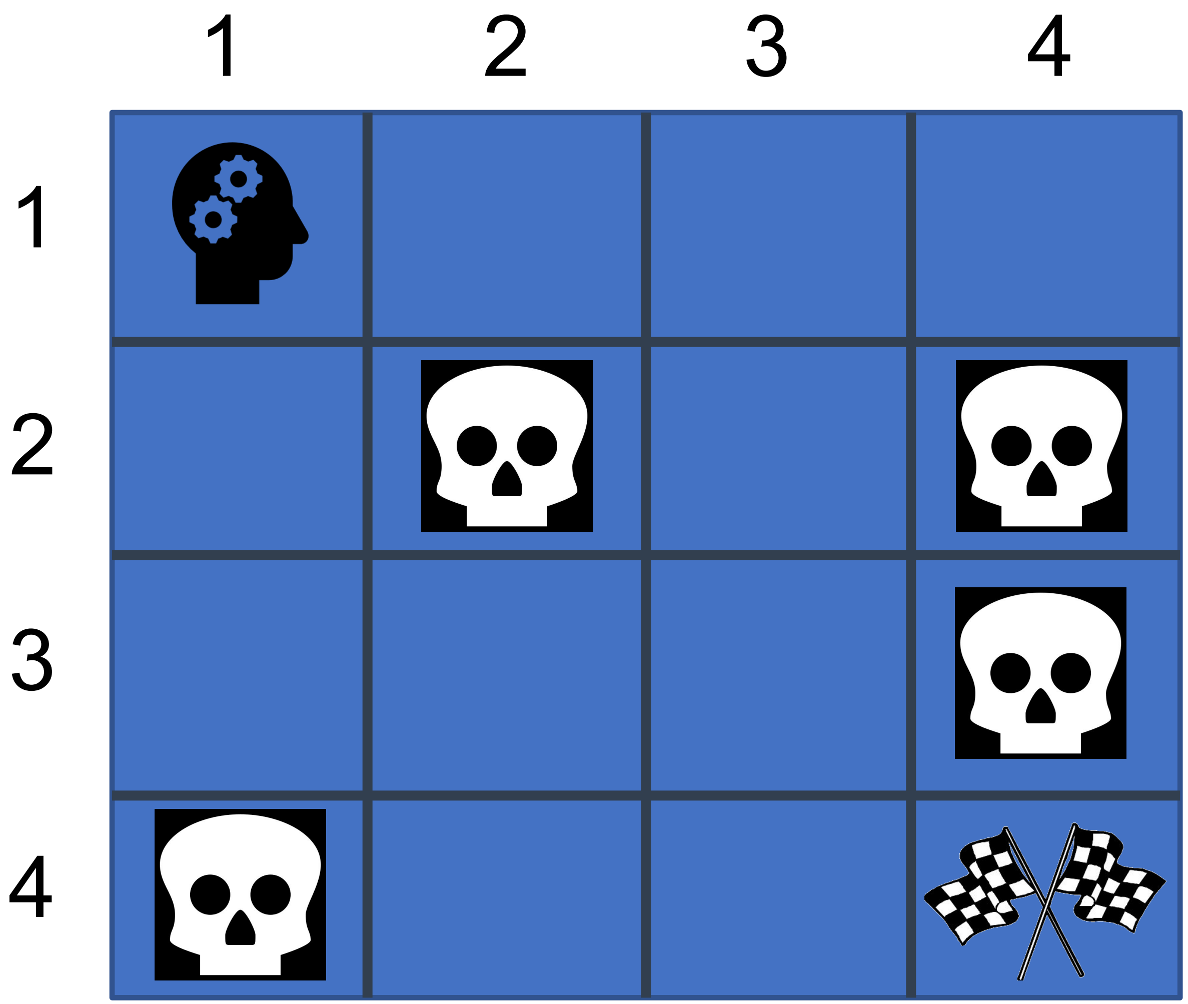
- Agent may not actually move in the direction of the action!
- Yellow arrow indicates the action
- Red arrows indicate where the agent may end up, each with probability 1/3

Can’t “fall off” frozen lake



- Transitioning beyond an edge will keep you in same state

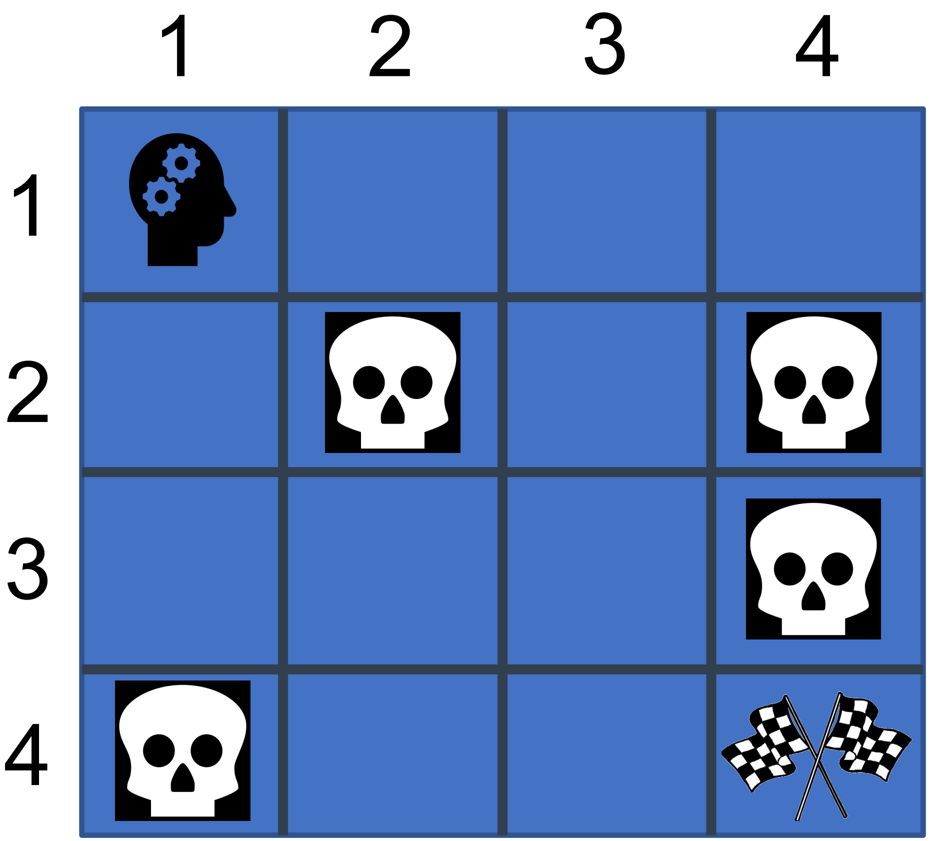
Frozen Lake Problem as an MDP



- States: each square - (1, 1), (1, 2), … , (4, 4)
- Actions: left, right, up, down
- Reward: +1 when you reach the goal, 0 elsewhere
- Transition function: stochastic (because ice is slippery!)
  Equal probability of moving in any direction except chosen action, e.g. if agent is in (1, 3) and action is down:
  - 1/3 chance of moving to (1, 2)
  - 1/3 chance of moving to (2, 3)
  - 1/3 chance of moving to (1, 4)

Frozen Lake - initialization

Environment

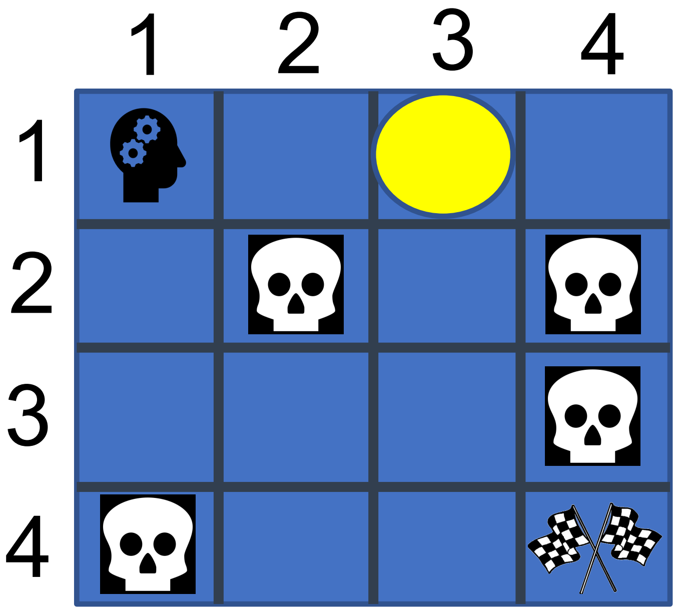


VALUE TABLE

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

Frozen Lake – iteration 1: update square (1, 3)

Environment



Old Value Table

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

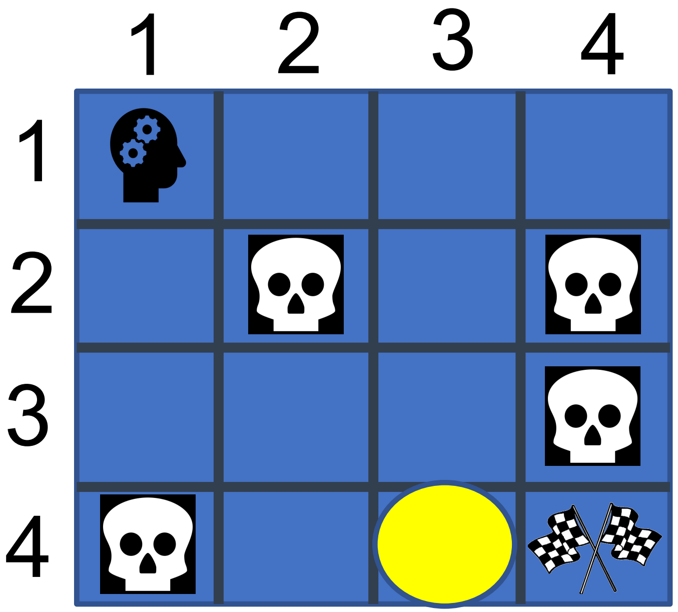
New Value Table

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

V((1, 3)) is still 0, because the adjacent values of (1, 3) are all 0 and no rewards are gained for any possible action taken in (1,3).

Frozen Lake – iteration 1: update square (4, 3)

Environment



Old Value Table

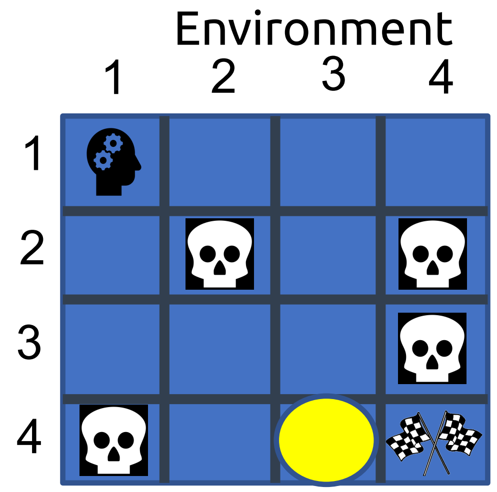
	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

New Value Table

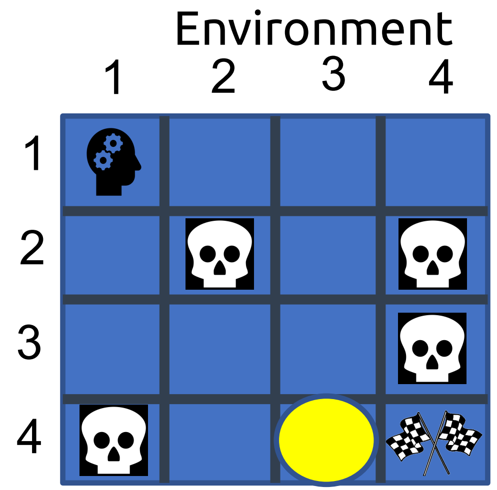
	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0.33	0

How did we get 0.33?

Update (4, 3) explanation



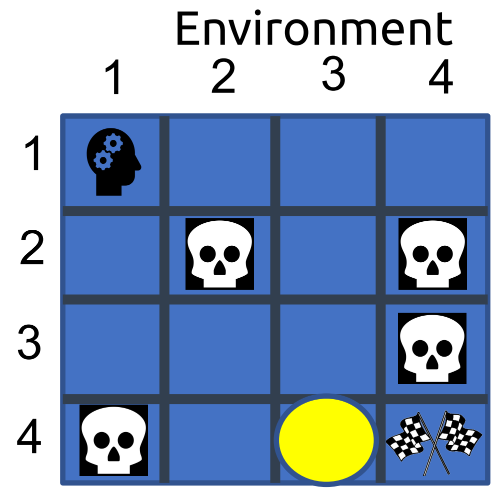
Update (4, 3) explanation



- Update equation:

$$V(s) = \max_a Q(s,a), \text{ where } Q(s,a) = \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(s')]$$

Update (4, 3) explanation

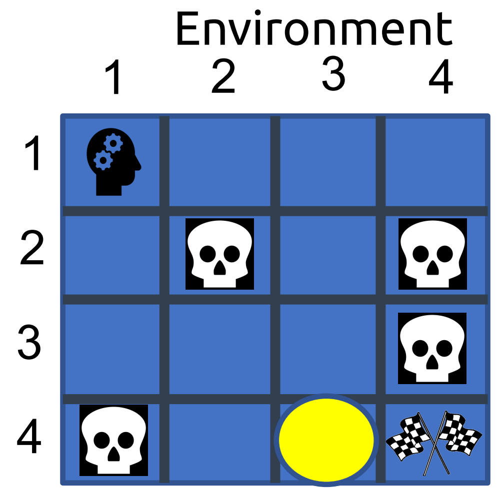


- Update equation:

$$V(s) = \max_a Q(s,a), \text{ where } Q(s,a) = \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(s')]$$

Finding Q values for all actions:

Update (4, 3) explanation



Update equation:

$$V(s) = \max_a Q(s,a), \text{ where } Q(s,a) = \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(s')]$$

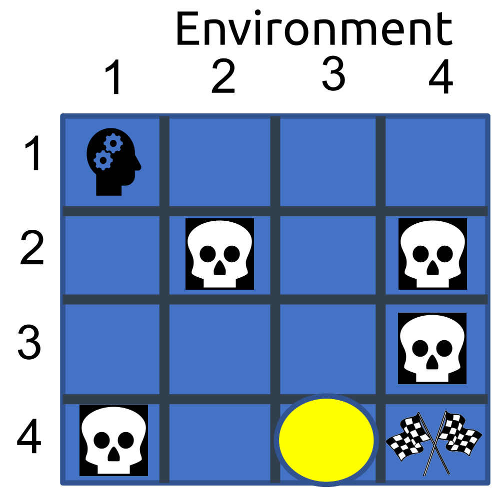
Finding Q values for all actions:

- $Q((4,3), right) = 1/3(1 + \gamma V((4,4))) + 1/3(0 + \gamma V((3,3))) + 1/3(0 + \gamma V((4,3)))$ = 0.33

Calculate Q((4,3),a) for all actions

Calculate (V(4,3))

Update (4, 3) explanation



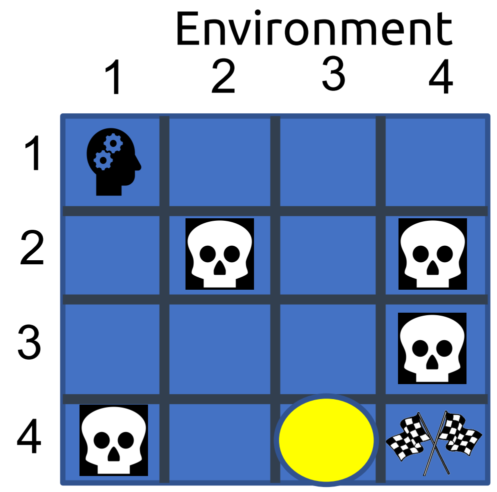
- Update equation:

$$V(s) = \max_a Q(s,a), \text{ where } Q(s,a) = \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(s')]$$

Finding Q values for all actions:

- $Q((4,3), right) = 1/3(1 + \gamma V((4,4))) + 1/3(0 + \gamma V((3,3))) + 1/3(0 + \gamma V((4,3))) = 0.33$
- $Q((4,3), up) = 1/3(0 + \gamma V((3,3))) + 1/3(1 + \gamma V((4,4))) + 1/3(0 + \gamma V((4,2))) = 0.33$

Update (4, 3) explanation



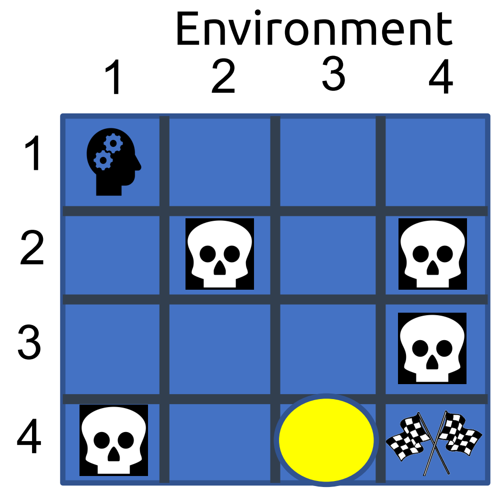
Update equation:

$$V(s) = \max_a Q(s,a), \text{ where } Q(s,a) = \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(s')]$$

Finding Q values for all actions:

- $Q((4,3), right) = 1/3(1 + \gamma V((4,4))) + 1/3(0 + \gamma V((3,3))) + 1/3(0 + \gamma V((4,3))) = 0.33$
- $Q((4,3), up) = 1/3(0 + \gamma V((3,3))) + 1/3(1 + \gamma V((4,4))) + 1/3(0 + \gamma V((4,2))) = 0.33$
- $Q((4,3), down) = 1/3(0 + \gamma V((4,3))) + 1/3(1 + \gamma V((4,4))) + 1/3(0 + \gamma V((4,2))) = 0.33$

Update (4, 3) explanation



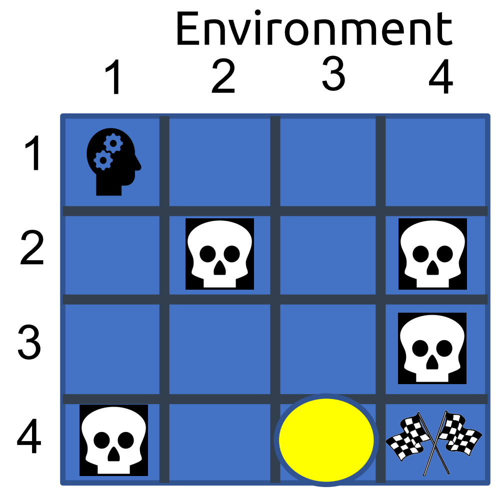
Update equation:

$$V(s) = \max_a Q(s,a), \text{ where } Q(s,a) = \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(s')]$$

Finding Q values for all actions:

- $Q((4,3), right) = 1/3(1 + \gamma V((4,4))) + 1/3(0 + \gamma V((3,3))) + 1/3(0 + \gamma V((4,3))) = 0.33$
- $Q((4,3), up) = 1/3(0 + \gamma V((3,3))) + 1/3(1 + \gamma V((4,4))) + 1/3(0 + \gamma V((4,2))) = 0.33$
- $Q((4,3), down) = 1/3(0 + \gamma V((4,3))) + 1/3(1 + \gamma V((4,4))) + 1/3(0 + \gamma V((4,2))) = 0.33$
- $Q((4,3), left) = 1/3(0 + \gamma V((4,2))) + 1/3(0 + \gamma V((4,3))) + 1/3(0 + \gamma V((3,3))) = 0$

Update (4, 3) explanation



Update equation:

$$V(s) = \max_a Q(s,a), \text{ where } Q(s,a) = \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(s')]$$

Finding Q values for all actions:

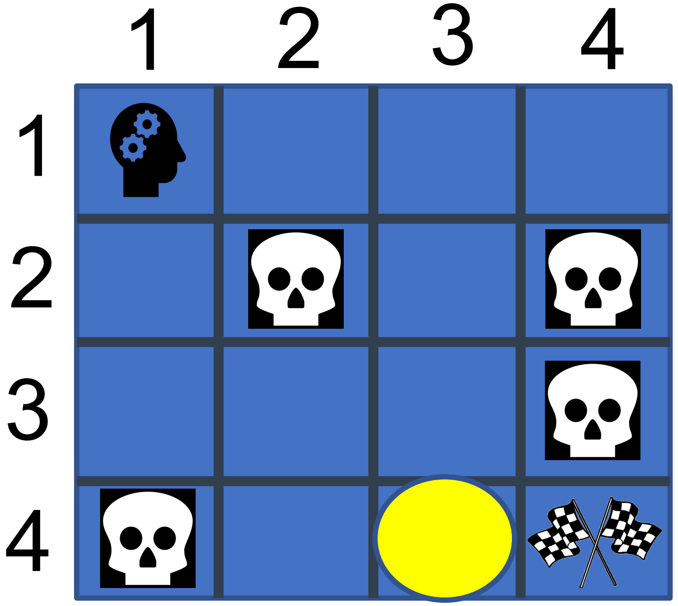
- $Q((4,3), right) = 1/3(1 + \gamma V((4,4))) + 1/3(0 + \gamma V((3,3))) + 1/3(0 + \gamma V((4,3))) = 0.33$
- $Q((4,3), up) = 1/3(0 + \gamma V((3,3))) + 1/3(1 + \gamma V((4,4))) + 1/3(0 + \gamma V((4,2))) = 0.33$
- $Q((4,3), down) = 1/3(0 + \gamma V((4,3))) + 1/3(1 + \gamma V((4,4))) + 1/3(0 + \gamma V((4,2))) = 0.33$
- $Q((4,3), left) = 1/3(0 + \gamma V((4,2))) + 1/3(0 + \gamma V((4,3))) + 1/3(0 + \gamma V((3,3))) = 0$

Then,

$$V((4,3)) = \max_a Q((4,3), a) = 0.33$$

Frozen Lake – iteration 2

Environment



Old Value Table

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0.33	0

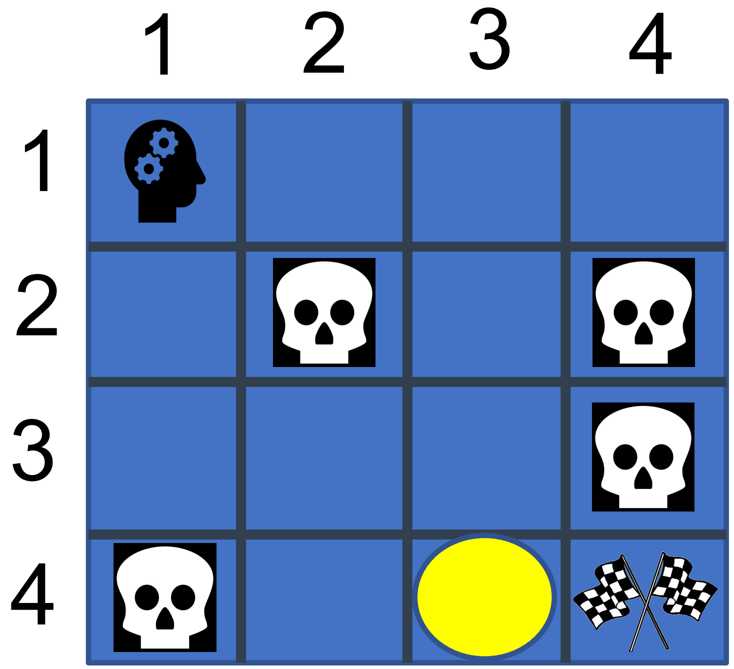
New Value Table

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0.1	0
4	0	0.1	0.47	0

- During this iteration, the value from (4, 3) “backs up” to its adjacent states, (3, 3) and (4,2).
- Value of (4, 3) increases because its adjacent states (3, 3) and (4, 2) have positive values.

Frozen Lake – final value table & optimal policy

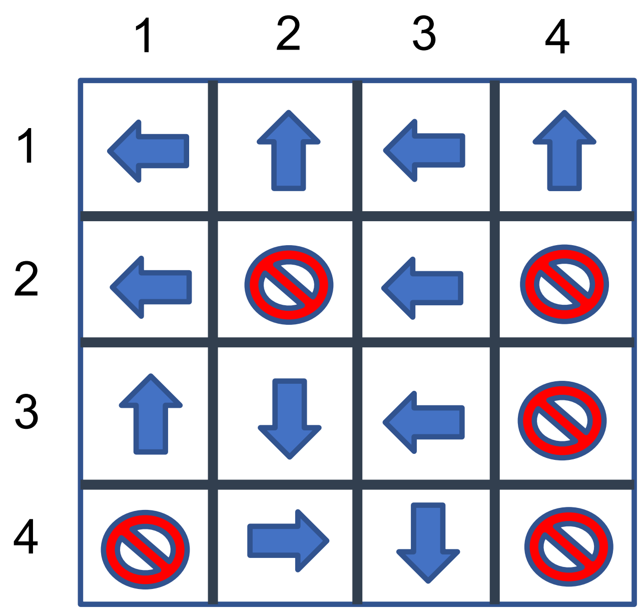
Environment



Example Final Value Table

	1	2	3	4
1	0.068	0.061	0.074	0.055
2	0.092	0	0.112	0
3	0.145	0.247	0.3	0
4	0	0.38	0.639	0

Final Policy



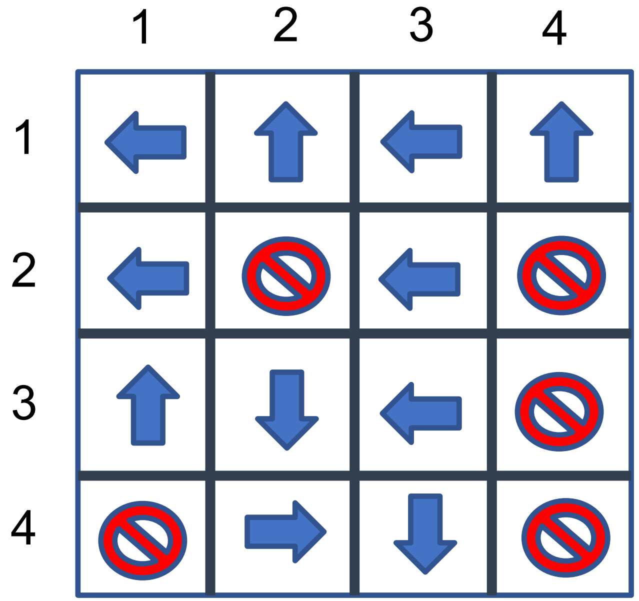
Now what?

Frozen Lake – demo

https://colab.research.google.com/drive/1RFFdzJ8VshmpvnbCbLNggwxfwMEBw222?usp=sharing

Frozen Lake – optimal policy in action

Final Policy



https://towardsdatascience.com/value-iteration-to-solve-openai-gyms-frozenlake-6c5e7bf0a64d

Play more with value iteration!

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

Recap

Value functions

- V function
- Q function (“action-value” function)
- Connection between V and Q

Value Iteration

- Pseudocode
- Frozen Lake Problem
- Demo: Learning Optimal Policy