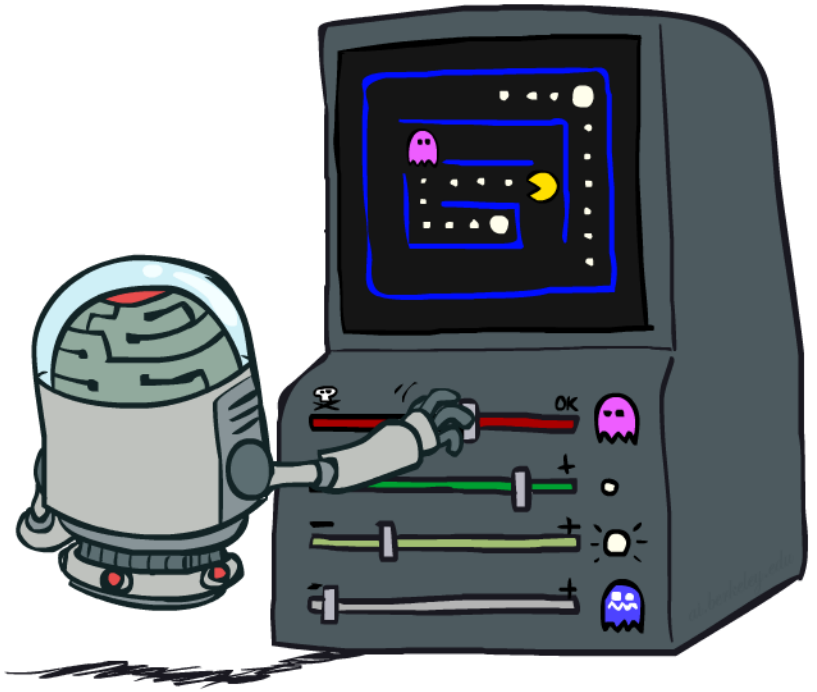
Reinforcement Learning II

CSE 473: Introduction to Artificial Intelligence



James Weichert (jpw@cs)
cs.uw.edu/473
acknowledgements

Welcome!
What was your favorite thing to learn?
sli.do **#cse473**

Administrivia

ANNOUNCEMENTS

- **Test 1** grades will be released soon

ASSIGNMENTS

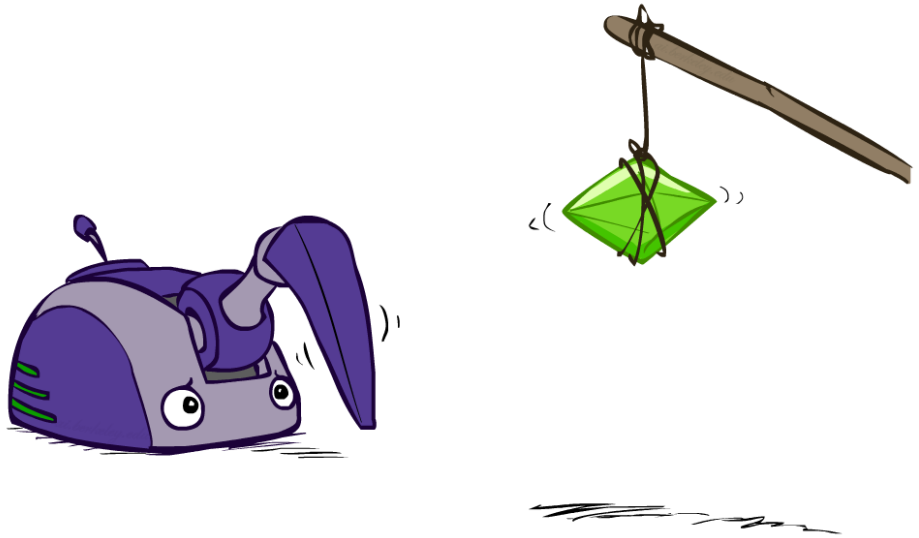
- **Project 2 due** tomorrow (7.16)
- **Homework 2 due** next week (7.23)

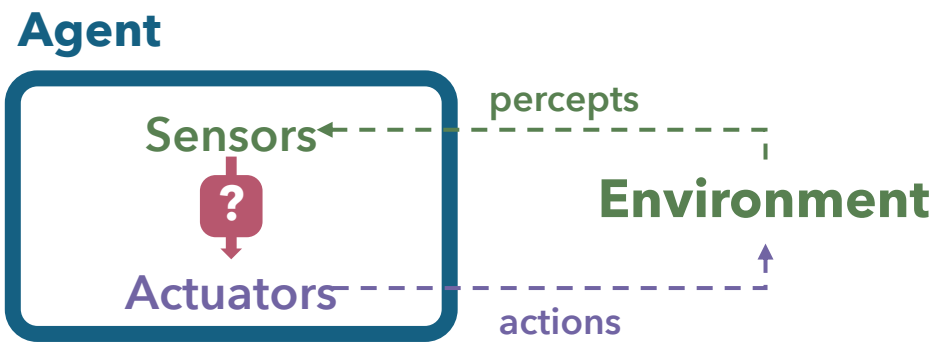
Recap: Reinforcement Learning

FOUNDATIONS

- Still a Markov Decision Process, with:
  - States $s \in S$
  - Actions $a \in A$
  - Transition model $T(s, a, s')$

    $$T(s, a, s') = P(s'|s, a)$$
  - Reward function $R(s, a, s')$
  - Start state s_0
  - Utility function $U(s)$
- But:
  - T and R are unknown.
  - Now must explore to learn best policy

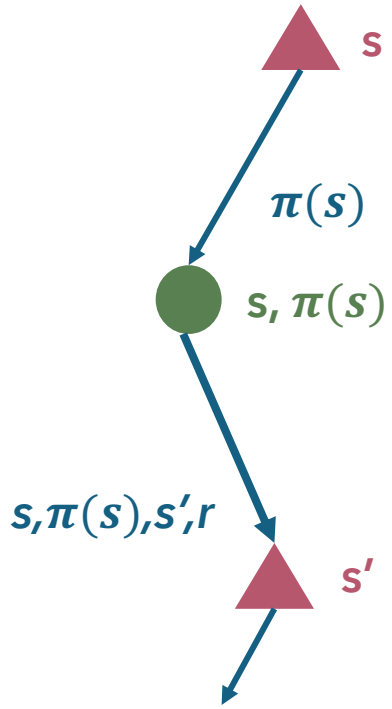




Recap: Temporal Difference Learning

FOUNDATIONS

- Idea: Learn from every experience!
  - Update $U(s)$ each time we experience a transition (s, a, s', r)
  - Likely outcomes s' will contribute updates more often
- Temporal Difference Learning
  - Policy is still fixed, still doing evaluation
  - Move $U(s)$ toward value of whichever successor s' occurs (running average)
  - **Sample of $U(s)$:** $\text{sample} = R(s, \pi(s), s') + \gamma U^{\pi}(s')$
  - **Update to $U(s)$:** $U^{\pi}(s) \leftarrow (1-\alpha)U^{\pi}(s) + \alpha \cdot \text{sample}$



Problems with TD Value Learning

CONTEXT

TD Value Learning performs model-free policy evaluation

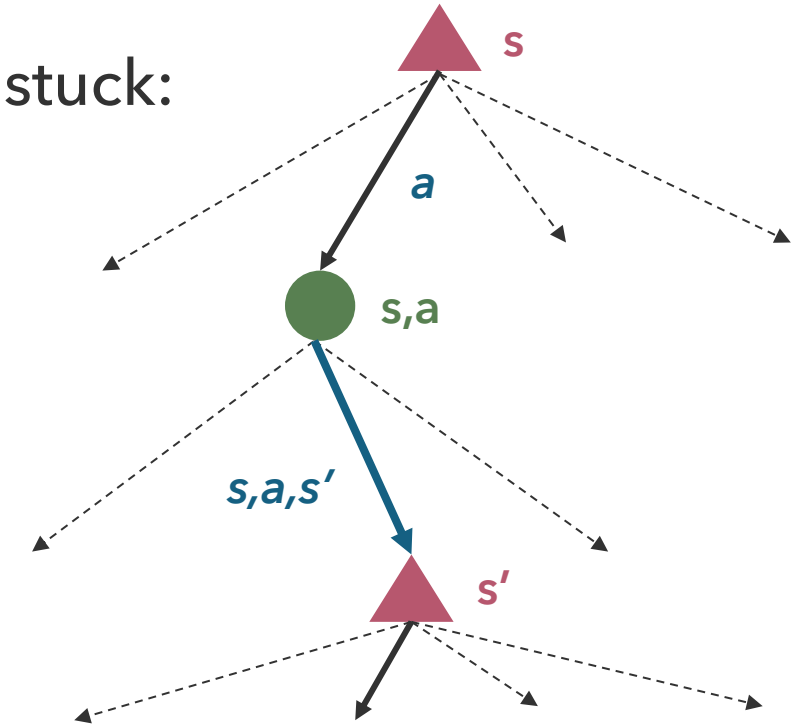
- Mimics Bellman updates with running average samples

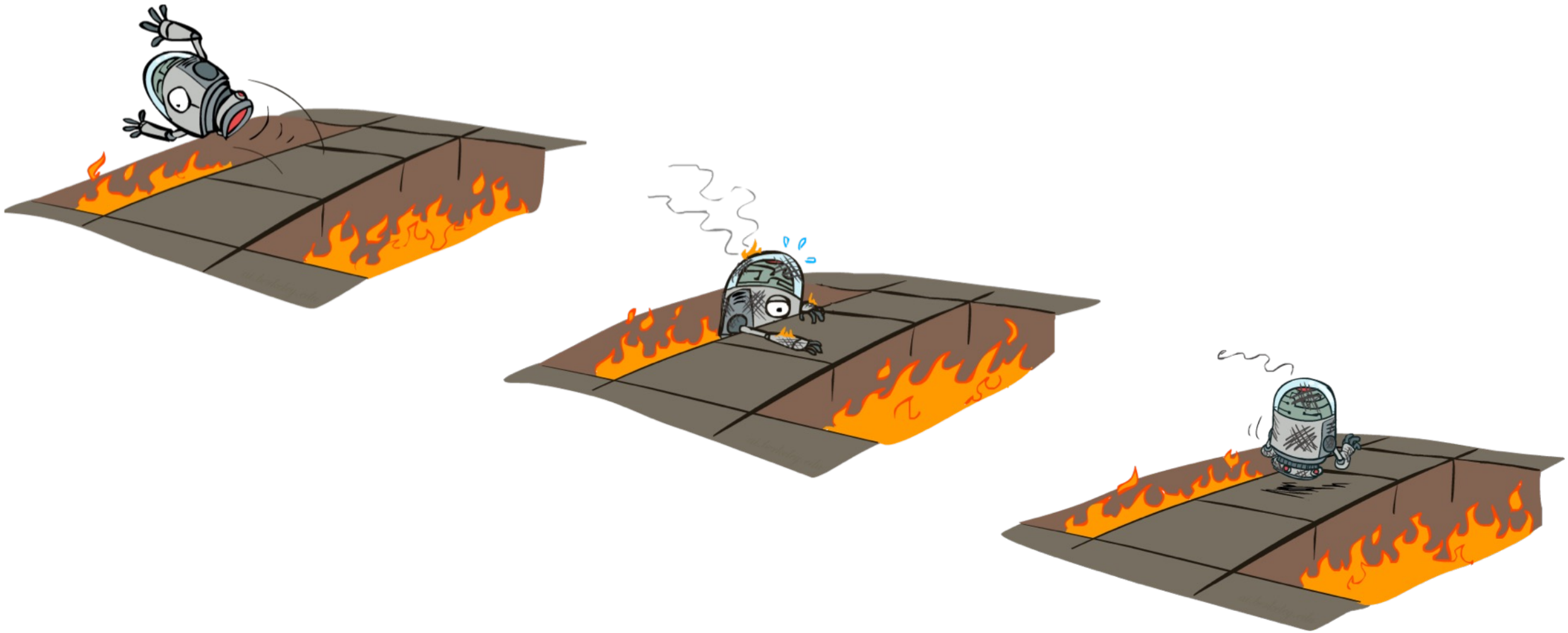
- However, if we want to turn values into policy, we're stuck:

$$\pi(s) = \text{argmax}_a Q(s,a)$$

$$Q(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma U(s')]$$

- Idea: Learn **Q-values**, not utilities
  - Makes action selection model-free, too!



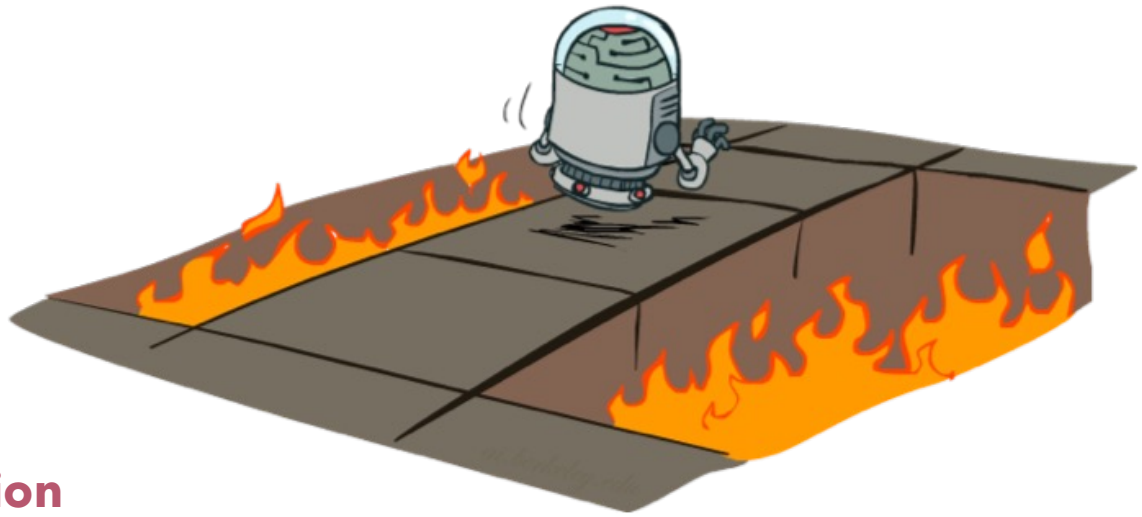


A More Hands-On Approach…

Active Reinforcement Learning

FOUNDATIONS

- *Full* Reinforcement Learning: Derive **optimal policies**
  - Don't know $T(s, a, s') = P(s'|s, a)$
  - Don't know $R(s, a, s)$
  - Choose actions
  - Goal: *learn the state* ***values and policy***



- In this case:
  - Learner makes choices!
  - Fundamental Tradeoff: **exploration** vs. **exploitation**
  - *Not* offline planning
    - Learner actually takes actions and learns from experience

QUESTIONS

Questions?

live and on sli.do #cse473

But First: Q-Value Iteration

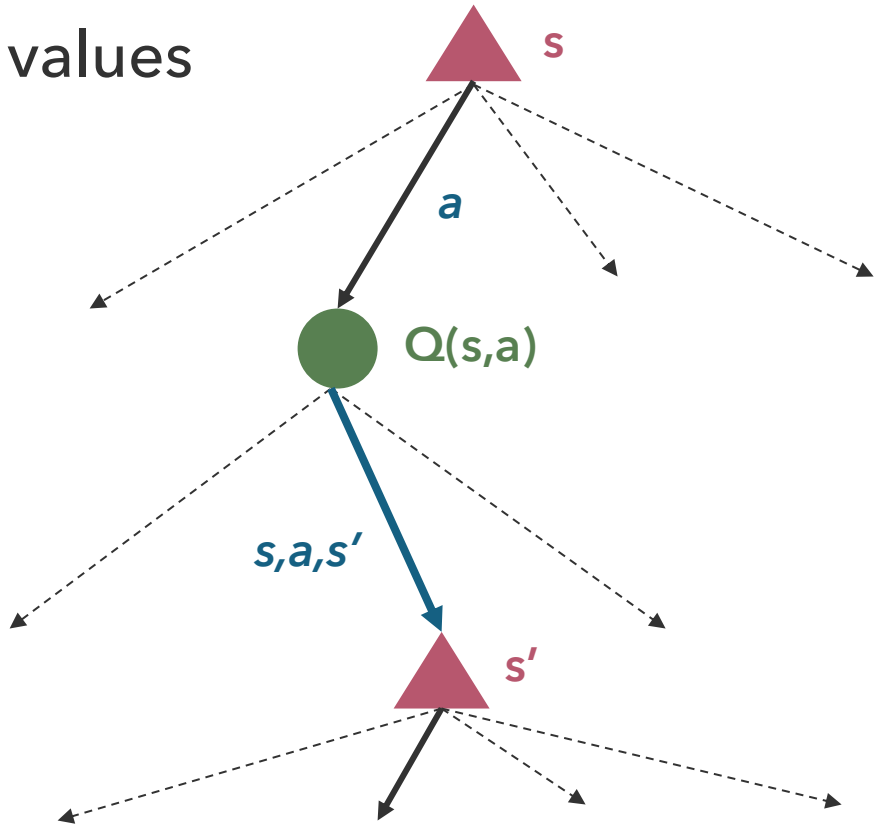
FOUNDATIONS

- **Value Iteration**: Find successive (depth-limited) values
  - Start with $U^{\pi}(s) = 0$
  - Given U_k, calculate depth $k+1$ values for all states:

$$U_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma U_k(s')]$$

- But **Q-values** are more useful…
  - Start with $Q_0(s,a) = 0$
  - Given Q_k, calculate depth $k+1$ Q-values for all states:

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q(s',a')]$$



Q-Learning

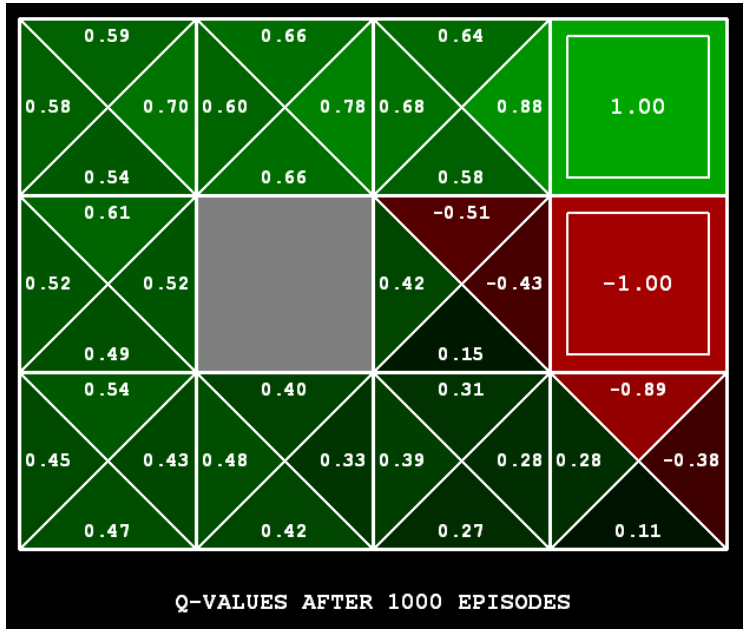
DEFINITION

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

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')]$$

- Learn Q-values as you go:
  - Receive sample (s,a,s',r)
  - Consider old estimate $Q_k(s,a)$
  - Consider new sample estimate: $\text{sample} = R(s,a,s') + \gamma \max_{a'} Q_k(s,a)$
  - Incorporate new estimate into a running average:

$$Q_{k+1}(s,a) \leftarrow (1-\alpha)Q_k(s,a) + \alpha \cdot \text{sample}$$



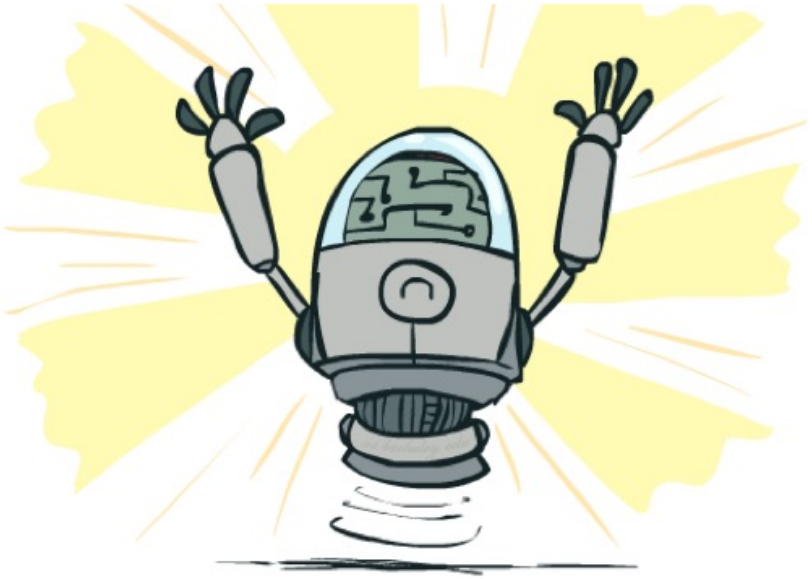
Properties of Q-Learning

DEFINITION

Off-Policy Learning

- Q-Learning converges to optimal policy, even if you're acting suboptimally!

- Caveats:
  - Must explore enough
  - Must eventually make the learning rate small enough
  - In the limit, it doesn't matter how actions are selected



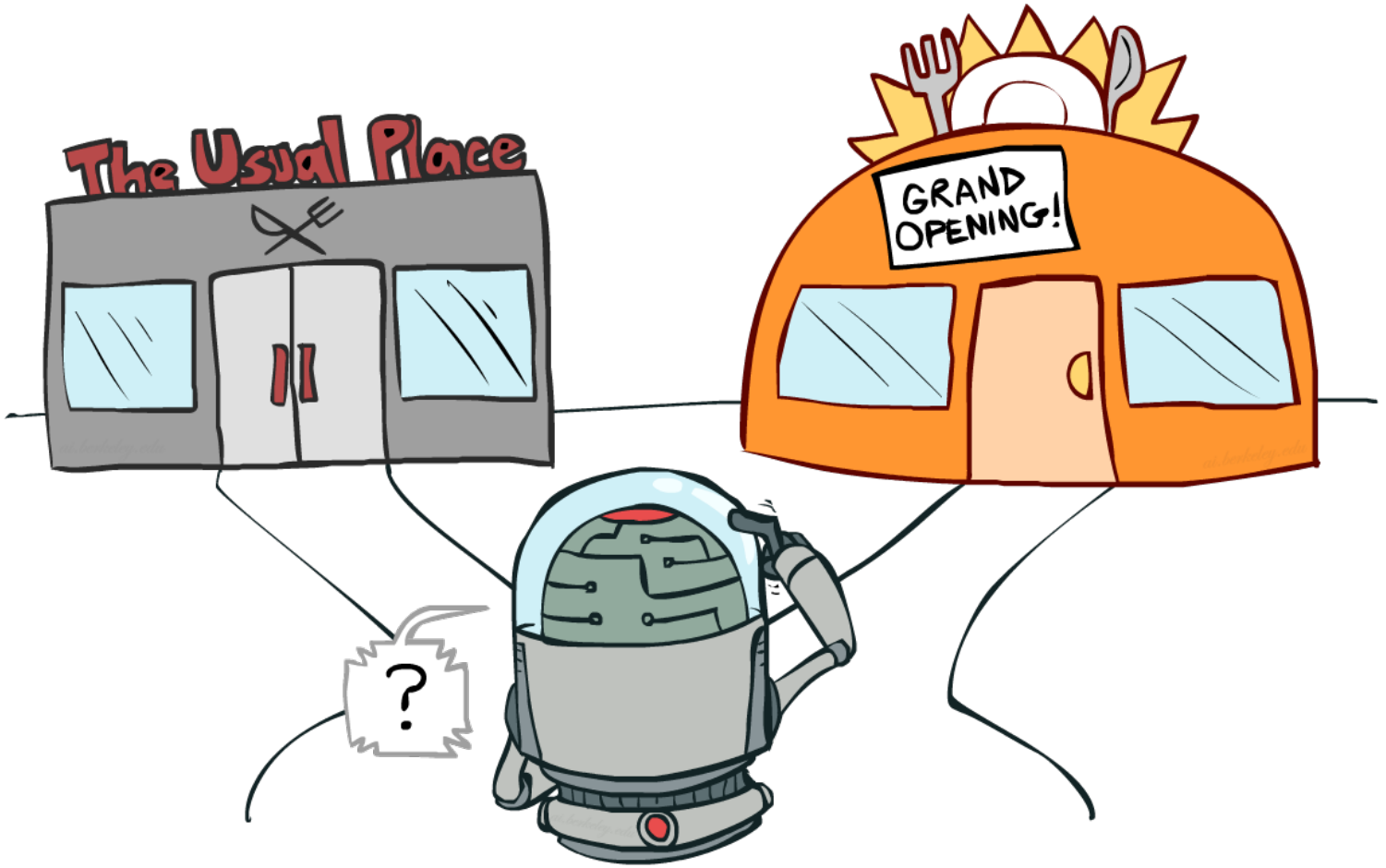
QUESTIONS

Questions?

live and on sli.do #cse473

Exploration vs. Exploitation

CONTEXT



How to Explore?

CONTEXT

How should the agent balance exploration and exploitation?

- Simple Approach: Random Chance (ε-greedy)
  - With (small) probability ε, take a random action
  - With (large) probability $1 - \varepsilon$, act on current policy
- *Problems with Random Actions?*
  - Eventually explore the space, but do so with a lot of thrashing around
  - One solution: lower ε over time
  - Another solution: **exploration functions**

Exploration Functions

FOUNDATIONS

- An **exploration function** takes value estimate u and visit count n and returns an optimistic utility $f(u, n)$
  - for example, $f(u, n) = \frac{u+c}{n}$ with constant c
  - Regular Q-Update:

$$Q_{k+1}(s, a) \leftarrow \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma \cdot \max_{a'} Q_k(s', a')]$$

  - **Modified Q-Update**:

$$Q_{k+1}(s, a) \leftarrow \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma \cdot \max_{a'} f(Q_k(s', a'), N(s', a')]$$

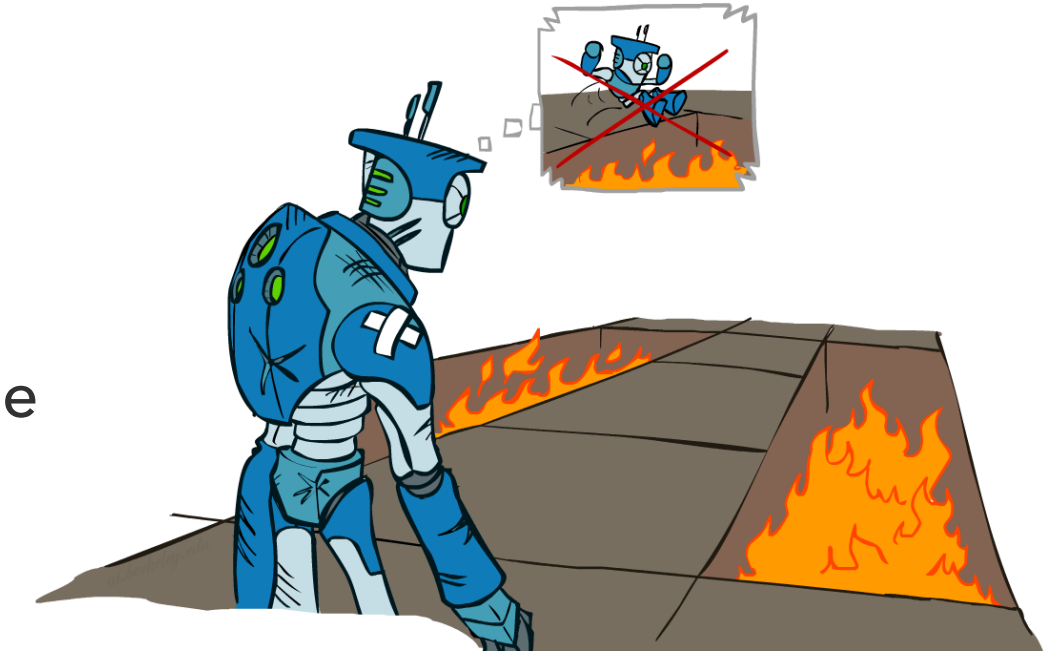
$N(s, a)$ is the number of times Q-state (s, a) has been visited

Regret

FOUNDATIONS

Even if you learn the optimal policy, you will make mistakes along the way!

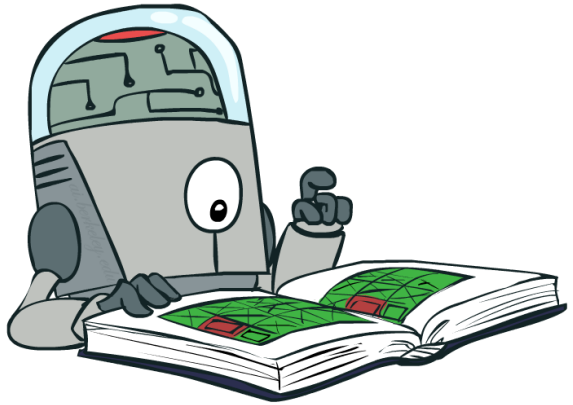
- **Regret** is a measure of the total mistake cost
  - Difference between (expected) rewards (including early suboptimality), and optimal (expected) rewards
  - For example, random exploration incurs more regret than exploring with exploration functions
- Minimizing regret applies optimization to the process of learning to be optimal



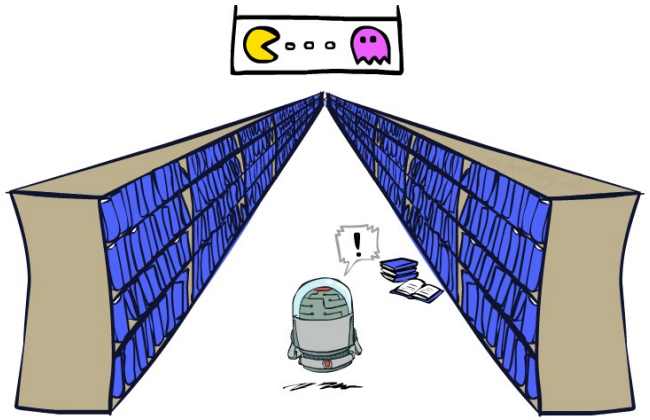
Generalizing Across States

CONTEXT

- Naïve Q-Learning keeps track of all Q-states
  - But in realistic situations, we cannot possibly learn about every single state!
    - Too many states to visit them all in training
    - Too many states to track in memory



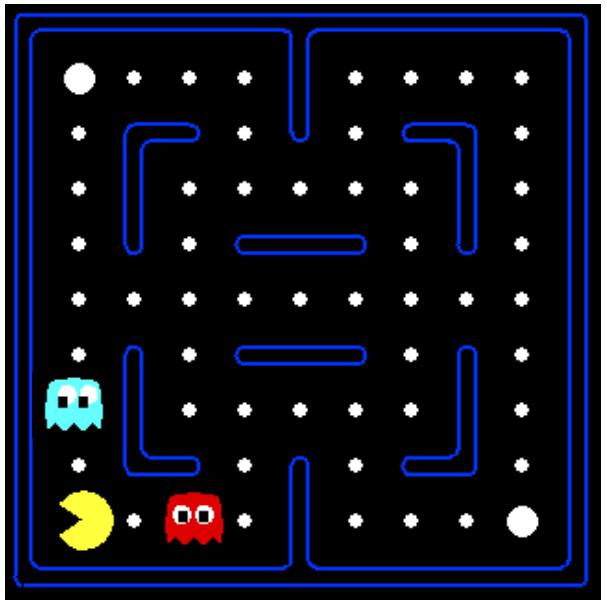
- Instead, we want to *generalize*:
  - Learn about some small number of states from experience
  - Generalize experiences to new, but similar situations



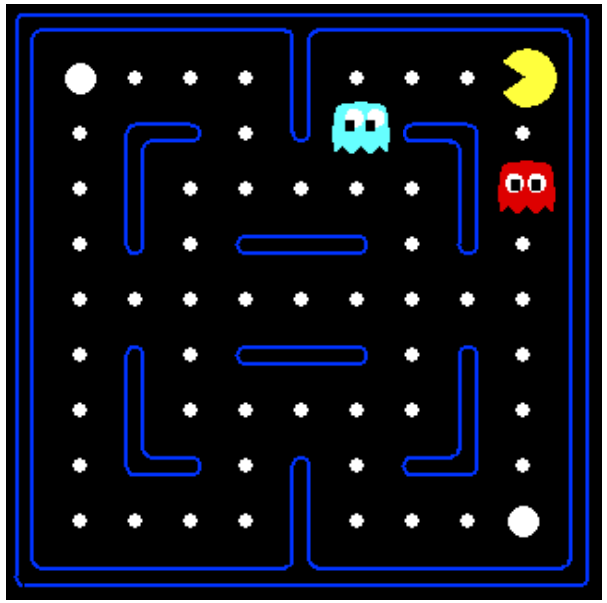
Generalizing learning is a fundamental challenge in AI!

Poor Generalization

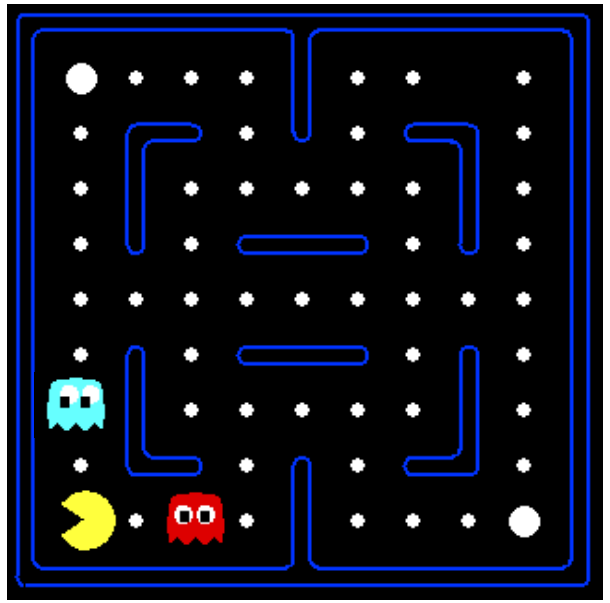
EXAMPLE



With naïve Q-learning, learning from experience in this state…



…doesn't provide information about this state…

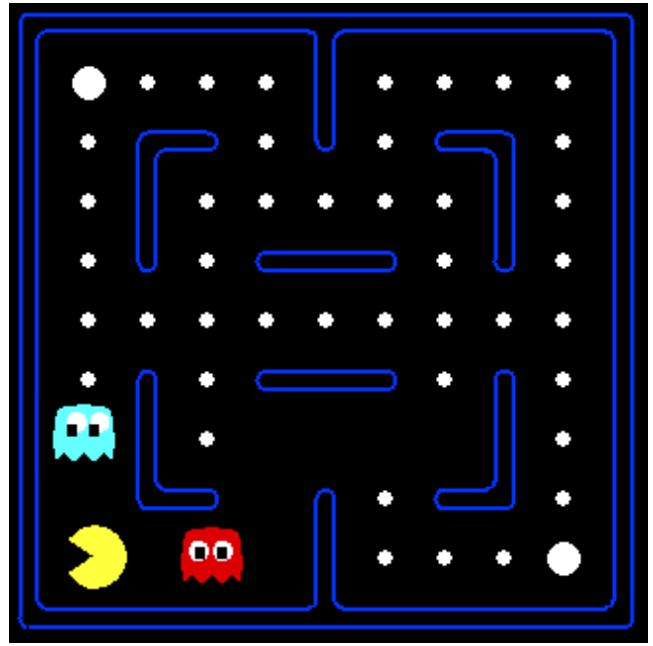


…or even this state!…

Feature-Based Learning

FOUNDATIONS

Solution: describe a state using a vector of features



- **Features** are functions mapping states to real numbers
  - For example:
    - Distance to closest ghost
    - Distance to closest dot
    - Number of ghosts
    - $\frac{1}{(\text{dist to closest dot})^2}$
    - Is Pacman in a tunnel?
    - Is it the exact state on this slide?
  - Can also describe a Q-state (s, a) with features (e.g. action moves closer to food)

Linear Value Functions

FOUNDATIONS

- Using a feature representation, define a **Q-function** (Q-value utility function)

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a)$$

(for n features)

- Advantage: Experience is summed up in a few powerful numbers
- Disadvantage: States may share features but differ in actual utility!

Approximate Q-Learning

DEFINITION

Approximate Q-Learning with linear combination of features as Q-function

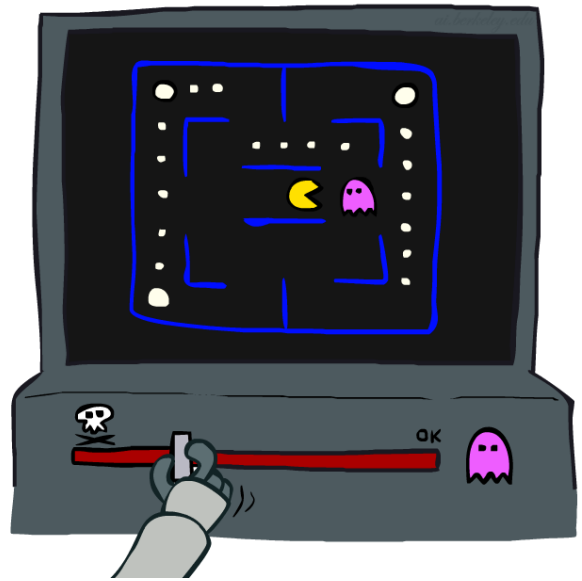
$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a)$$

- Process:
  - Collect transition samples (s, a, s', r)
  - Let difference $\leftarrow [r + \gamma \cdot \max_{a'} Q(s', a')] - Q(s,a)$
  - **Exact Q-Values:**

$$Q_{k+1}(s,a) \leftarrow Q_k(s,a) + \alpha \cdot \text{difference}$$

  - ***Approximate* Q-Values:**

$$w_i \leftarrow w_i + \alpha \cdot \text{difference}$$



Intuition Behind Q-Learning

DEFINITION

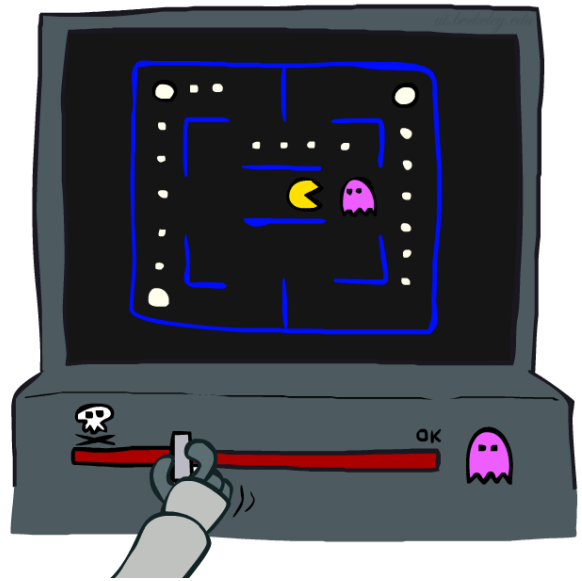
Approximate Q-Learning with linear combination of features as Q-function

$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + \cdots + w_n f_n(s,a)$$

- **Approximate Q-Values:**

$$w_i \leftarrow w_i + \alpha \cdot \text{difference}$$

- Intuition
  - Adjust weights of active features
  - If something unexpectedly bad happens, blame the features that were "on" – avoid all states with similar features



QUESTIONS

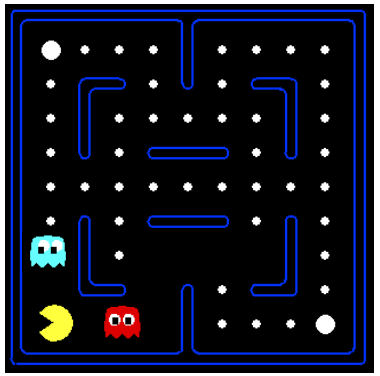
Questions?

live and on sli.do #cse473

Q-Pacman (1)

EXAMPLE

- Let $Q(s,a) = 4.0 \cdot f_{\text{DOT}}(s,a) - 1.0 \cdot f_{\text{GST}}(s,a)$

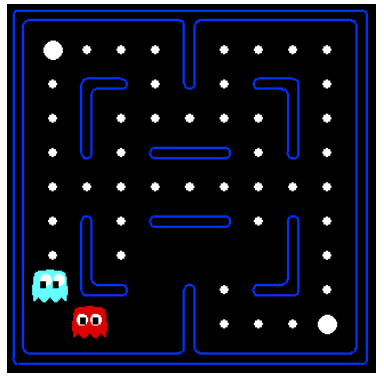
s 

$f_{\text{DOT}}(s, \text{NORTH}) = 0.5$

$f_{\text{GST}}(s, \text{NORTH}) = 1$

$a = \text{NORTH}$

$r = -500$



s'

$Q(s,a) = Q(s, NORTH) = +1$

$Q(s,\cdot) = 0$

$r + \gamma \cdot \max_{a'} Q(s,a) = -500 + 0$

$\text{difference} = -501$ → $w_{\text{DOT}} \leftarrow (4.0) + \alpha \cdot [-501] \cdot (0.5)$

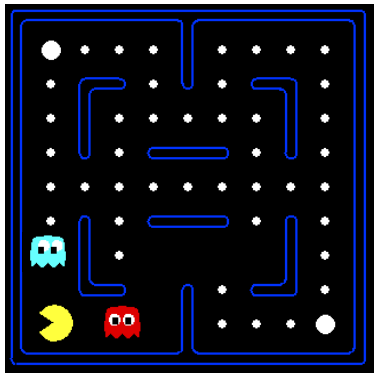
$w_{\text{GST}} \leftarrow (-1.0) + \alpha \cdot [-501] \cdot (1)$

Q-Pacman (2)

EXAMPLE

- Let $Q(s,a) = 4.0 \cdot f_{\text{DOT}}(s,a) - 1.0 \cdot f_{\text{GST}}(s,a)$

s

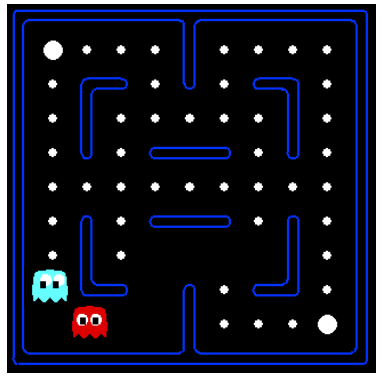


$f_{\text{DOT}}(s, \text{NORTH}) = 0.5$

$f_{\text{GST}}(s, \text{NORTH}) = 1$

$a = \text{NORTH}$

$r = -500$



s'

$Q(s,a) = Q(s, NORTH) = +1$

$Q(s,\cdot) = 0$

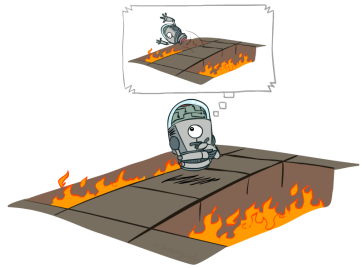
$r + \gamma \cdot \max_{a'} Q(s,a) = -500 + 0$

Update: $Q(s,a) = 3.0 \cdot f_{\text{DOT}}(s,a) - 3.0 \cdot f_{\text{GST}}(s,a)$

with $\alpha \approx 0.004$

Summary: Reinforcement Learning

CONTEXT



Offline Learning

Known MDP

Online Learning

Unknown MDP

Goal	Approach
Compute U^*, Q^*, π^*	Value and Policy Iteration
Evaluate fixed policy π^*	Policy Evaluation

Model-Based

Goal	Approach
Compute U^*, Q^*, π^*	Value, Policy Iter. on approx. MDP
Evaluate fixed policy π^*	Policy Eval. on approx. MDP

Model-Free

Goal	Approach
Compute U^*, Q^*, π^*	Q-Learning
Evaluate fixed policy π^*	Value Learning

QUESTIONS

Questions?

live and on sli.do #cse473

that's it for today!

SUMMARY

- Q-Learning
- Exploration vs. Exploitation Tradeoff
- Feature-Based Approximation

UPCOMING

- Reinforcement Learning in Practice

REMINDERS

- Complete **Practice Problem 9**
- Do **Project 2** (due 7.16)
- Do **Homework 2** (due 7.23)