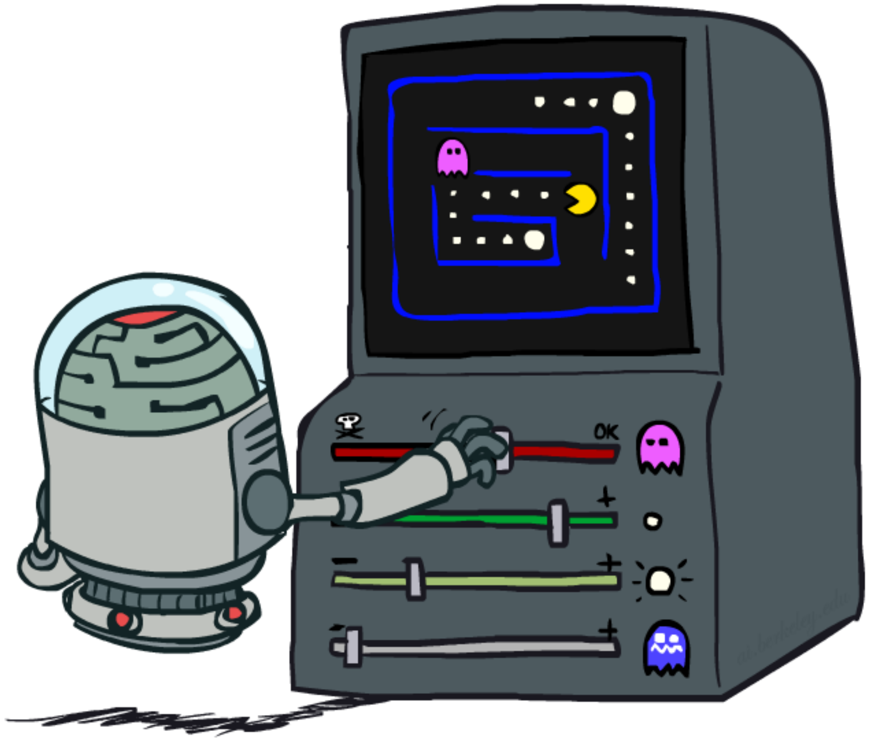
# CS 188: Artificial Intelligence

## Reinforcement Learning II




Instructors: Dan Klein and Pieter Abbeel --- University of California, Berkeley

# Reinforcement Learning

- We still assume an MDP:
  - A set of states $s \in S$
  - A set of actions (per state) A
  - A model T(s,a,s')
  - A reward function R(s,a,s')
- Still looking for a policy $\pi(s)$

- New twist: don't know T or R, so must try out actions
- Big idea: Compute all averages over T using sample outcomes

# The Story So Far: MDPs and RL

## Known MDP: Offline Solution

| Goal | Technique |
| --- | --- |
| Compute V*, Q*, $\pi$* | Value / policy iteration |
| Evaluate a fixed policy $\pi$ | Policy evaluation |

## Unknown MDP: Model-Based

| Goal | Technique |
| --- | --- |
| Compute V*, Q*, $\pi$* | VI/PI on approx. MDP |
| Evaluate a fixed policy $\pi$ | PE on approx. MDP |

## Unknown MDP: Model-Free

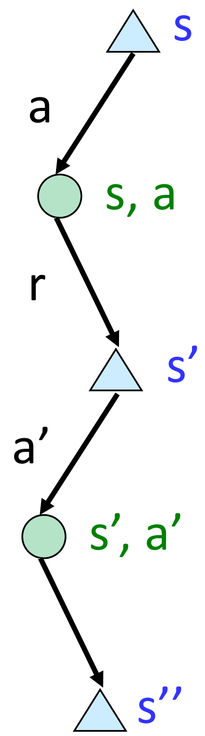
| Goal | Technique |
| --- | --- |
| Compute V*, Q*, $\pi$* | Q-learning |
| Evaluate a fixed policy $\pi$ | Value Learning |

# Model-Free Learning

- Model-free (temporal difference) learning
  - Experience world through episodes

$$(s, a, r, s', a', r', s'', a'', r'', s'''' \ldots)$$

  - Update estimates each transition $(s, a, r, s')$
  - Over time, updates will mimic Bellman updates



# Q-Learning

- We'd like to do Q-value updates to each Q-state:

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

  - But can't compute this update without knowing T, R

- Instead, compute average as we go
  - Receive a sample transition (s,a,r,s')
  - This sample suggests

$$Q(s,a) \approx r + \gamma \max_{a'} Q(s',a')$$

  - But we want to average over results from (s,a) (Why?)
  - So keep a running average

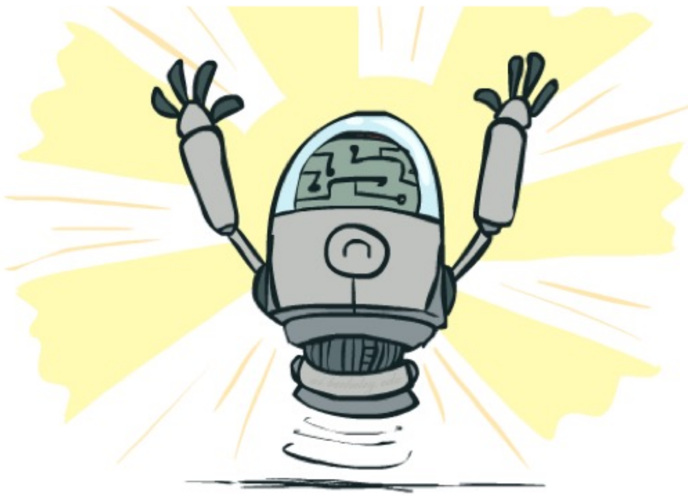
$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)\left[r + \gamma \max_{a'} Q(s',a')\right]$$

# Q-Learning Properties

- Amazing result: Q-learning converges to optimal policy -- even if you're acting suboptimally!

- This is called off-policy learning

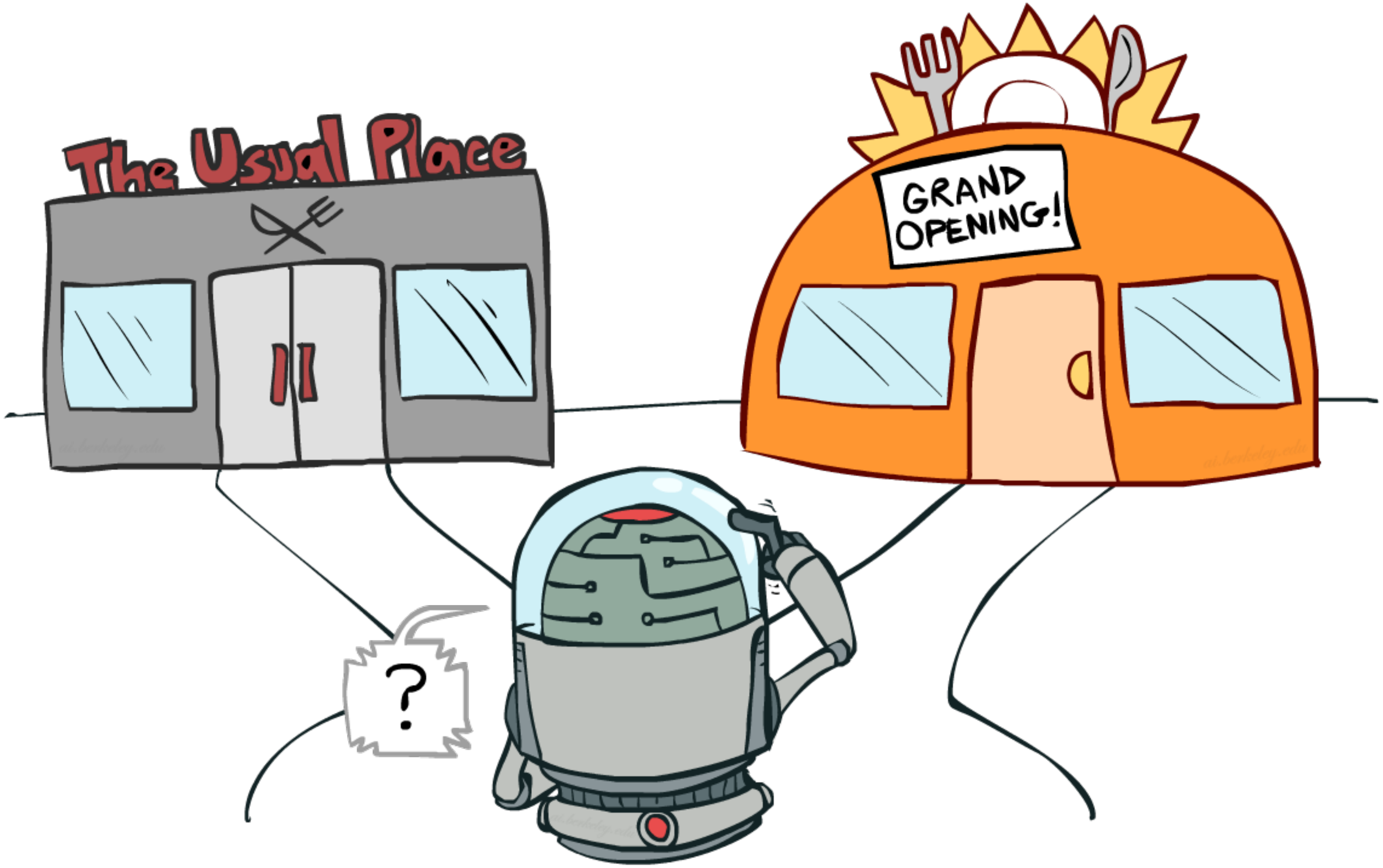
- Caveats:
  - You have to explore enough
  - You have to eventually make the learning rate small enough
  - … but not decrease it too quickly
  - Basically, in the limit, it doesn't matter how you select actions (!)



[Demo: Q-learning – auto – cliff grid (L11D1)]

# Video of Demo Q-Learning Auto Cliff Grid

# Exploration vs. Exploitation



# How to Explore?

- Several schemes for forcing exploration
  - Simplest: random actions ($\varepsilon$-greedy)
    - Every time step, flip a coin
    - With (small) probability $\varepsilon$, act randomly
    - With (large) probability $1-\varepsilon$, act on current policy
  - Problems with random actions?
    - You do eventually explore the space, but keep thrashing around once learning is done
    - One solution: lower $\varepsilon$ over time
    - Another solution: exploration functions

[Demo: Q-learning – manual exploration – bridge grid (L11D2)]
[Demo: Q-learning – epsilon-greedy -- crawler (L11D3)]

# Video of Demo Q-learning – Manual Exploration – Bridge Grid

# Video of Demo Q-learning – Epsilon-Greedy – Crawler

# Exploration Functions

- When to explore?
  - Random actions: explore a fixed amount
  - Better idea: explore areas whose badness is not (yet) established, eventually stop exploring


- Exploration function
  - Takes a value estimate u and a visit count n, and returns an optimistic utility, e.g. $f(u,n) = u + k/n$

Regular Q-Update: $Q(s,a) \leftarrow_{\alpha} R(s,a,s') + \gamma \max_{a'} Q(s',a')$

Modified Q-Update: $Q(s,a) \leftarrow_{\alpha} R(s,a,s') + \gamma \max_{a'} f(Q(s',a'), N(s',a'))$

  - Note: this propagates the "bonus" back to states that lead to unknown states as well!

[Demo: exploration – Q-learning – crawler – exploration function (L11D4)]

# Video of Demo Q-learning – Exploration Function – Crawler