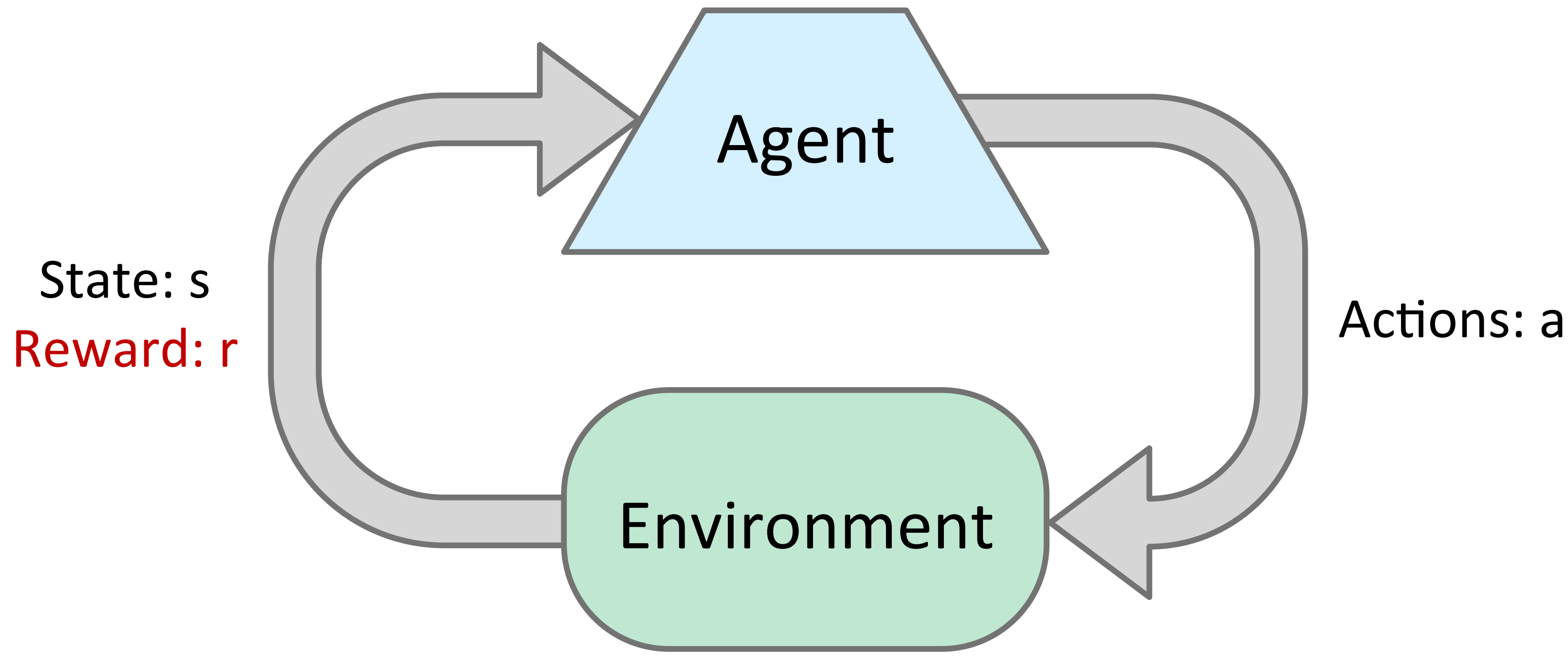
Reinforcement learning

Tomas Svoboda, BE5B33KUI
2017-04-24

Slide material partly from CS 188: Artificial Intelligence at UCB
by Dan Klein, and Pieter Abbeel, used with permision.
Some figures from the AIMA book 3rd edition

Reinforcement learning



Agent acts - executes an action

- Receives feedback in the form of **rewards**
- Must learn to act so as to **maximize expected rewards**
- All learning based on what it observes

Learning from failures

Autonomous Flipper Control with Safety Constraints

Martin Pecka, Vojtěch Šalanský,
Karel Zimmermann, Tomáš Svoboda

experiments utilizing
Constrained Relative Entropy Policy Search

MDPs and Reinforcement Learning

Markov decision process – MDPs

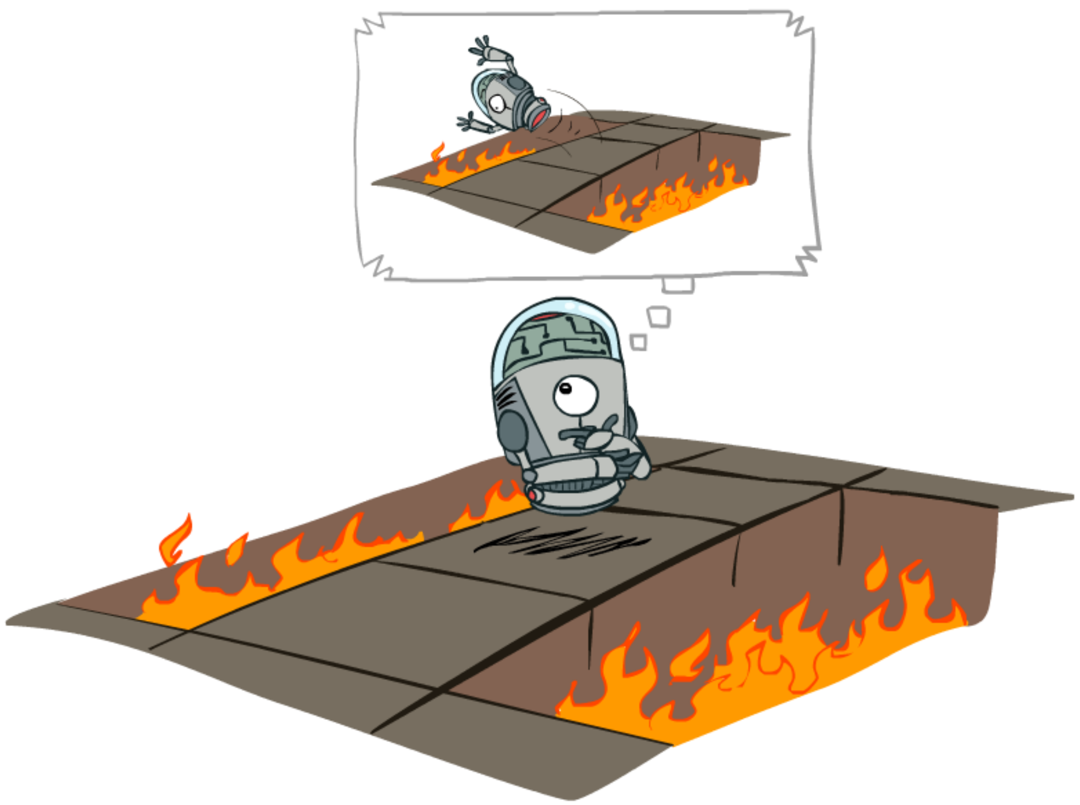
- A set of states $s \in S$
- A set of actions per state
- A transition model $T(s, a, s')$ or $P(s'|s, a)$
- A reward function $R(s, a, s')$

Looking for the optimal policy $\pi(s)$.

Now, we do not know *T* and *R*

- We do not know what states are good
- We must try and learn from the result(s)

off-line (MDPs) vs on-line (RL)



Offline Solution

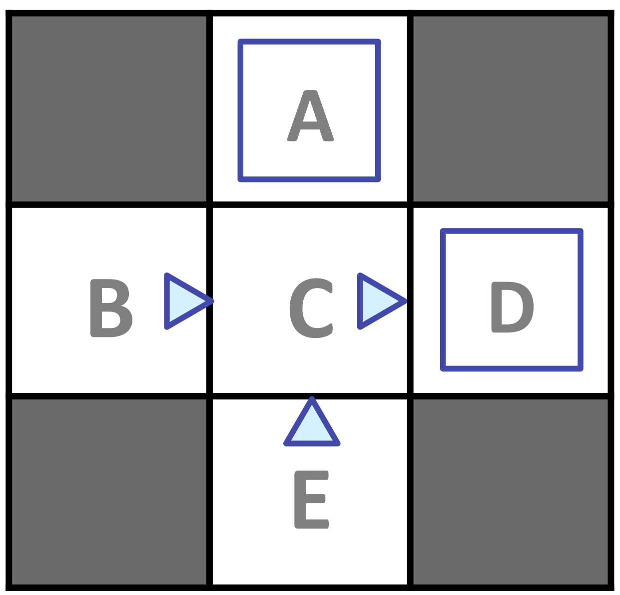
Online Learning

Model-based learning

- The main idea:
  - learn an approximate model from experiences
  - solve as if the learned model were corrent
- Learning MPD model
  - count s' for each s,a
  - normalize to get an estimate of T(s,a,s')
- Solve the learned MDP (e.g Value iteration)

Example of learning model

Input Policy π



Assume: $\gamma = 1$

Observed Episodes (Training)

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

Learned Model

$\hat{T}(s, a, s')$

T(B, east, C) = 1.00
T(C, east, D) = 0.75
T(C, east, A) = 0.25
…

$\hat{R}(s, a, s')$

R(B, east, C) = -1
R(C, east, D) = -1
R(D, exit, x) = +10
…

Adaptive dynamic programming

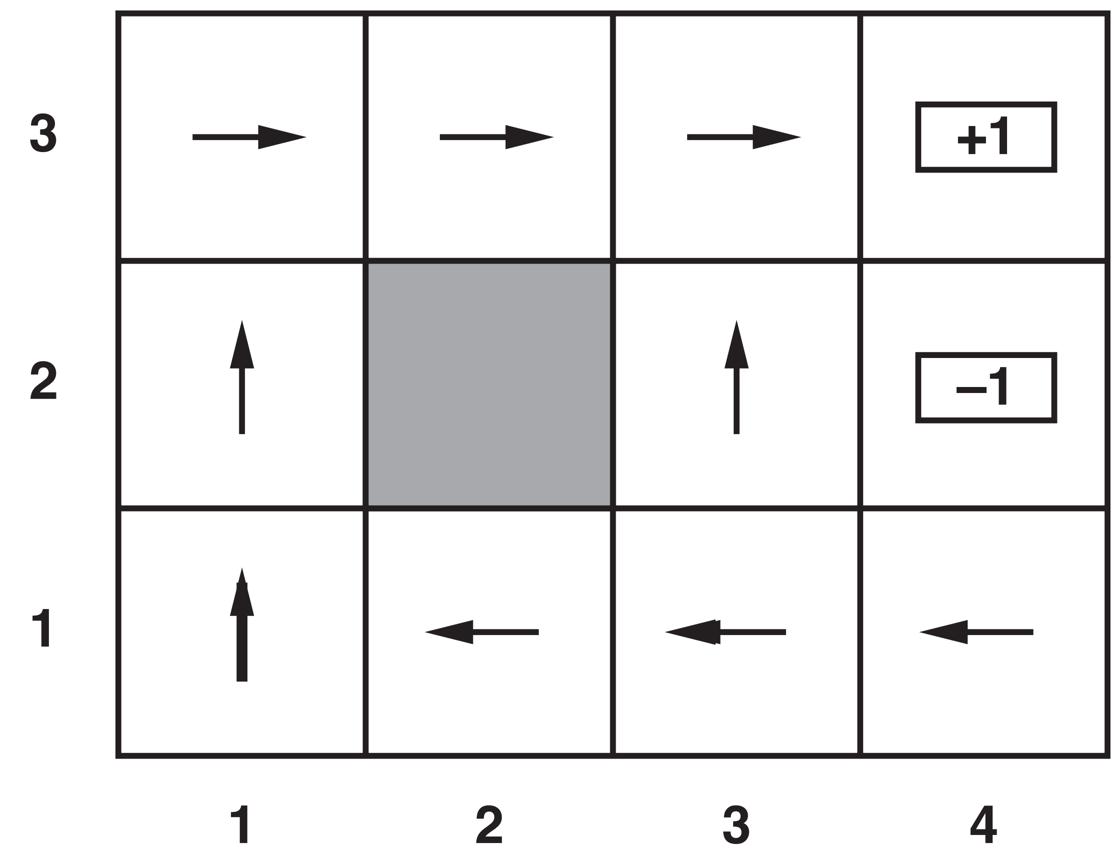
function PASSIVE-ADP-AGENT(*percept*) **returns** an action
  inputs: *percept*, a percept indicating the current state s' and reward signal r'
  persistent: π, a fixed policy
    mdp, an MDP with model P, rewards R, discount γ
    U, a table of utilities, initially empty
    N_{sa}, a table of frequencies for state–action pairs, initially zero
    $N_{s'|sa}$, a table of outcome frequencies given state–action pairs, initially zero
    s, a, the previous state and action, initially null

  if s' is new **then** $U[s'] \leftarrow r'$; $R[s'] \leftarrow r'$
  if s is not null **then**
    increment $N_{sa}[s, a]$ and $N_{s'|sa}[s', s, a]$
    for each t such that $N_{s'|sa}[t, s, a]$ is nonzero **do**
      $P(t \mid s, a) \leftarrow N_{s'|sa}[t, s, a] \,/\, N_{sa}[s, a]$
  $U \leftarrow$ POLICY-EVALUATION(π, U, *mdp*)
  if s'.TERMINAL? **then** $s, a \leftarrow$ null **else** $s, a \leftarrow s', \pi[s']$
  return a

Model-free learning

Passive learning

- **Input:** a fixed policy $\pi(s)$
- Execute policy . . .
- and learn on the way
- **Goal:** learn the state values $U^\pi(s)$



$$(1,1)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (2,3)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (4,3)_{\textbf{+1}}$$
$$(1,1)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (2,3)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (3,2)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (4,3)_{\textbf{+1}}$$
$$(1,1)_{\textbf{-.04}} \rightsquigarrow (2,1)_{\textbf{-.04}} \rightsquigarrow (3,1)_{\textbf{-.04}} \rightsquigarrow (3,2)_{\textbf{-.04}} \rightsquigarrow (4,2)_{\textbf{-1}} \ .$$

$$U^\pi(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(S_t)\right]$$

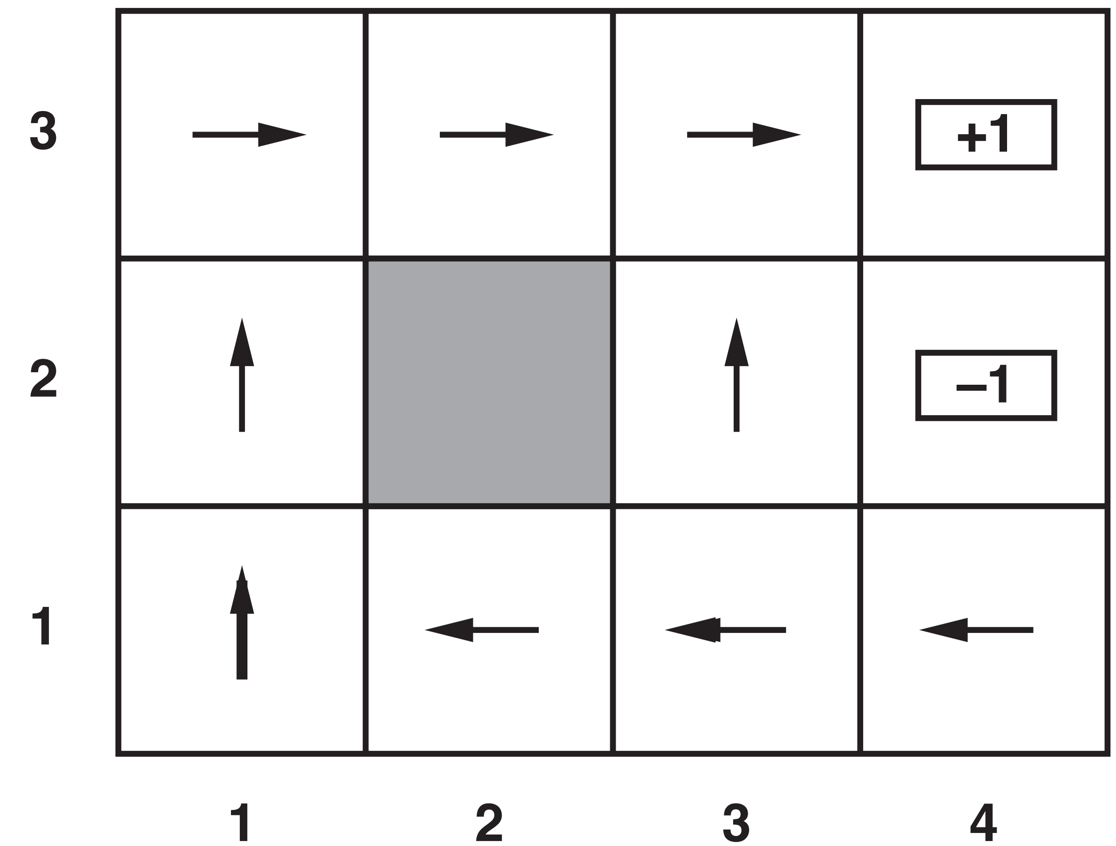
Direct utility estimation

$$(1,1)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (2,3)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (4,3)_{\textbf{+1}}$$
$$(1,1)_{\textbf{-.04}} \rightsquigarrow (1,2)_{\textbf{-.04}} \rightsquigarrow (1,3)_{\textbf{-.04}} \rightsquigarrow (2,3)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (3,2)_{\textbf{-.04}} \rightsquigarrow (3,3)_{\textbf{-.04}} \rightsquigarrow (4,3)_{\textbf{+1}}$$
$$(1,1)_{\textbf{-.04}} \rightsquigarrow (2,1)_{\textbf{-.04}} \rightsquigarrow (3,1)_{\textbf{-.04}} \rightsquigarrow (3,2)_{\textbf{-.04}} \rightsquigarrow (4,2)_{\textbf{-1}} \ .$$

- Act according to the policy
- When visiting a state, remeber what the sum of discounted rewards turned out to be
- Compute average
- Utility of a state - expected total reward from that state onward
- Each trial provides a *sample* of this quantity

Direct utility estimation

What is U(3,2) after 3 trials?



$$(1,1)_{\mathbf{-.04}} \rightsquigarrow (1,2)_{\mathbf{-.04}} \rightsquigarrow (1,3)_{\mathbf{-.04}} \rightsquigarrow (1,2)_{\mathbf{-.04}} \rightsquigarrow (1,3)_{\mathbf{-.04}} \rightsquigarrow (2,3)_{\mathbf{-.04}} \rightsquigarrow (3,3)_{\mathbf{-.04}} \rightsquigarrow (4,3)_{\mathbf{+1}}$$

$$(1,1)_{\mathbf{-.04}} \rightsquigarrow (1,2)_{\mathbf{-.04}} \rightsquigarrow (1,3)_{\mathbf{-.04}} \rightsquigarrow (2,3)_{\mathbf{-.04}} \rightsquigarrow (3,3)_{\mathbf{-.04}} \rightsquigarrow (3,2)_{\mathbf{-.04}} \rightsquigarrow (3,3)_{\mathbf{-.04}} \rightsquigarrow (4,3)_{\mathbf{+1}}$$

$$(1,1)_{\mathbf{-.04}} \rightsquigarrow (2,1)_{\mathbf{-.04}} \rightsquigarrow (3,1)_{\mathbf{-.04}} \rightsquigarrow (3,2)_{\mathbf{-.04}} \rightsquigarrow (4,2)_{\mathbf{-1}} \; .$$

Direct utility estimation - what is good and what is bad

- The good:
  - Simple, easy to implement and understand
  - Does not need T, R and it computes true U
- The bad:
  - Each state utility learned separately
  - It does not use information about the state connection!
  - State utilities are *not independent*

$$U^{\pi}(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi(s)) U^{\pi}(s')$$

What about policy evaluation

In each round, replace U with a one-step-look-ahead

$U_0^\pi(s) = 0$
$U_{i+1}^\pi(s) \leftarrow \sum_{s'} T(s, \pi(s), s')\left[R(s, \pi(s), s') + \gamma U_i^\pi(s')\right]$

Problem: both $T(s, \pi(s), s')$ and $R(s, \pi(s), s')$ unknown!

Use samples for evaluating policy?

MDP (T, R known) : Update U estimate by a weighted average:

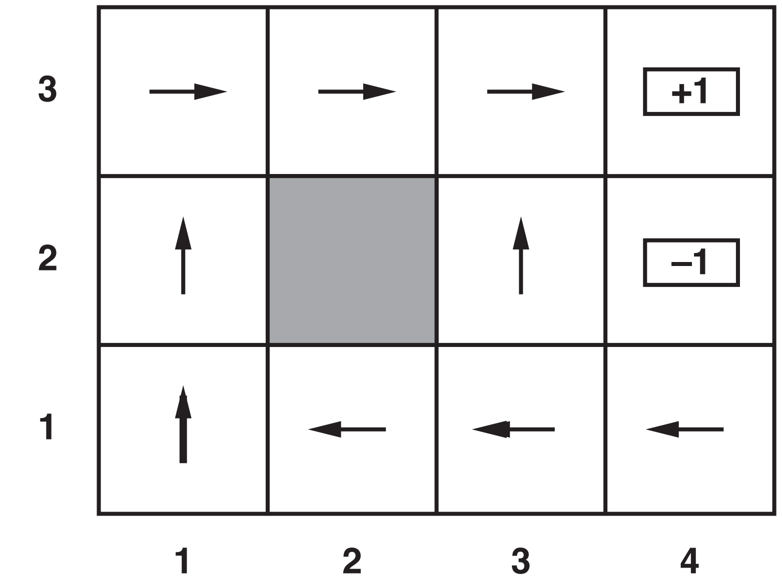
$U^{\pi}_{i+1}(s) \leftarrow \sum_{s'} T(s, \pi(s), s') \big[R(s, \pi(s), s') + \gamma U^{\pi}_{i}(s')\big]$

What about: try (sample) and average:

$$
\begin{aligned}
\text{trial}_1 &= R(s, \pi(s), s'_1) + \gamma U^{\pi}_{i}(s'_1) \\
\text{trial}_2 &= R(s, \pi(s), s'_2) + \gamma U^{\pi}_{i}(s'_2) \\
\vdots &= \vdots \\
\text{trial}_n &= R(s, \pi(s), s'_n) + \gamma U^{\pi}_{i}(s'_n)
\end{aligned}
$$

$$U^{\pi}_{i+1}(s) \leftarrow \frac{1}{n} \sum_{i} \text{trial}_i$$

Temporal-difference learning



$(1,1)_{\mathbf{-.04}} \rightsquigarrow (1,2)_{\mathbf{-.04}} \rightsquigarrow (1,3)_{\mathbf{-.04}} \rightsquigarrow (1,2)_{\mathbf{-.04}} \rightsquigarrow (1,3)_{\mathbf{-.04}} \rightsquigarrow (2,3)_{\mathbf{-.04}} \rightsquigarrow (3,3)_{\mathbf{-.04}} \rightsquigarrow (4,3)_{\mathbf{+1}}$

$(1,1)_{\mathbf{-.04}} \rightsquigarrow (1,2)_{\mathbf{-.04}} \rightsquigarrow (1,3)_{\mathbf{-.04}} \rightsquigarrow (2,3)_{\mathbf{-.04}} \rightsquigarrow (3,3)_{\mathbf{-.04}} \rightsquigarrow (3,2)_{\mathbf{-.04}} \rightsquigarrow (3,3)_{\mathbf{-.04}} \rightsquigarrow (4,3)_{\mathbf{+1}}$

$(1,1)_{\mathbf{-.04}} \rightsquigarrow (2,1)_{\mathbf{-.04}} \rightsquigarrow (3,1)_{\mathbf{-.04}} \rightsquigarrow (3,2)_{\mathbf{-.04}} \rightsquigarrow (4,2)_{\mathbf{-1}}$.

$$U(2,3) = 0.92,\ U(1,3) = 0.84$$

$$U(1,3) = R(1,3) + U(2,3) = -0.04 + 0.92 = 0.88$$

The current $U(1,3)$ estimate, 0.84, should be increased.

$$U(s) \leftarrow U(s) + \alpha([R(s) + \gamma U(s')] - U(s))$$

where α is the **learning rate**.

Problems with temporal difference learning

Utilities learned through policy evaluation by mimicking Bellman updates

How to construct a new policy?

$$\pi(s) = \arg\max_a Q(s,a)$$

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

Active reinforcement learning

Q-learning

Value/Utility and Q-value iteration

Value/Utility iteration (depth limited evaluation):

- Start: $U_0(s) = 0$
- In each step update U by looking one step ahead:

$$U_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma U_i(s')\right]$$

Q values more useful (think about updating π)

- Start: $Q_0(s,a) = 0$
- In each step update U by looking one step ahead:

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

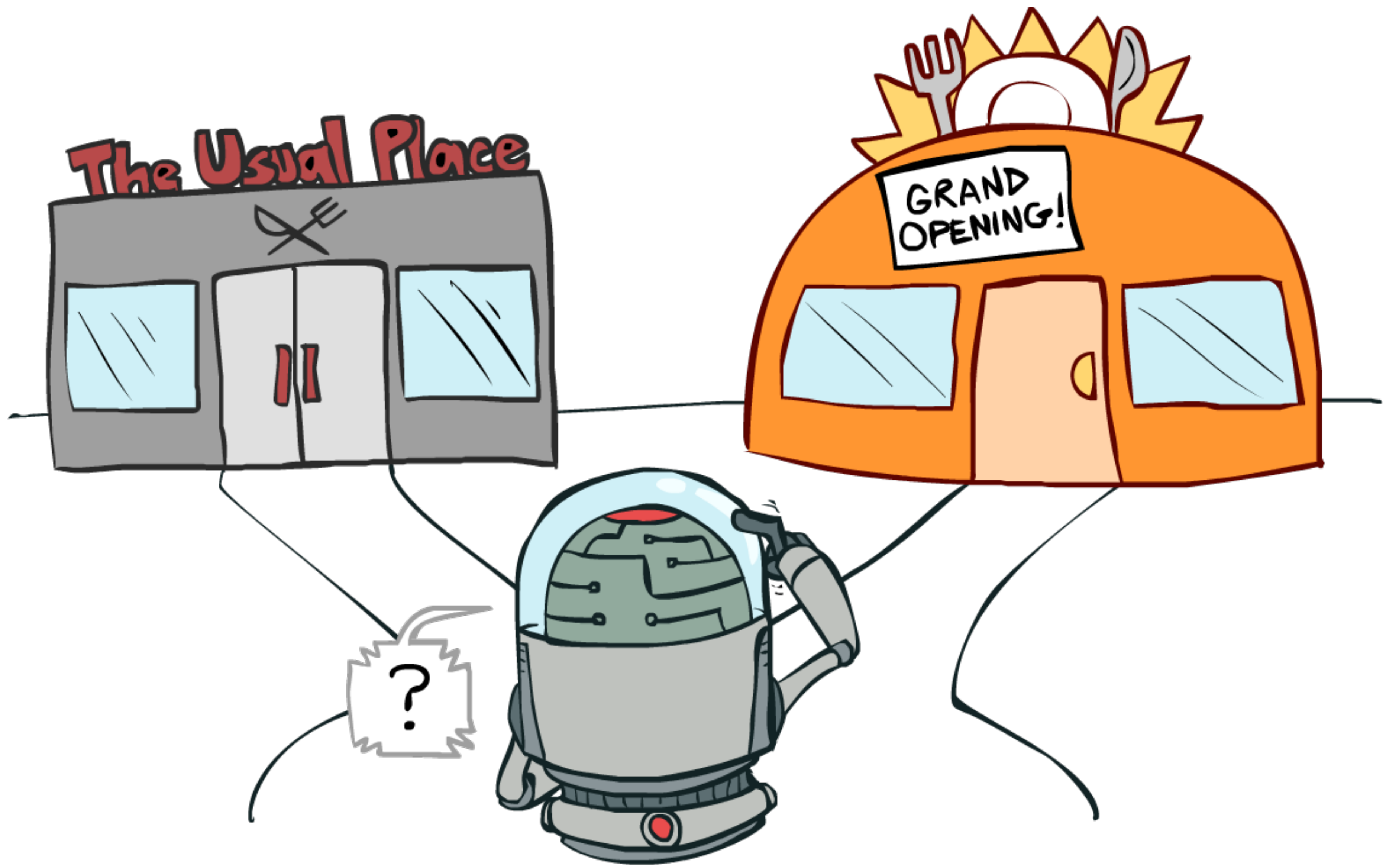
Q-learning

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

Learn Q values as the robot/agent goes (temporal difference)

- Drive the robot and fetch: s, a, s', r
- We know old estimates $Q(s,a)$
- A new trial/sample estimate
  $\text{trial} = r + \gamma \max_{a'} Q_i(s',a')$
- α update
  $Q(s,a) \leftarrow Q(s,a) + \alpha(\text{trial} - Q(s,a))$
  $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\,\text{trial}$

Exploration vs Exploitation



How to explore

Standard Q-learning:

from a learned value: trial $= r + \gamma \max_{a'} Q_i(s', a')$

we update with α rate: $Q(s,a) \leftarrow Q(s,a) + \alpha(\text{trial} - Q(s,a))$

Modify the learned value by **boosting yet overlooked areas**:

trial $= r + \gamma \max_{a'} f(Q_i(s', a'), N(s', a'))$

where f is an *exploration function*. It returns a more optimistic utility from a value estimate u and visit count n, e.g.:

$f(u, v) = u + i/n$ (i is iteration)

Q-learning agent

function Q-LEARNING-AGENT($percept$) **returns** an action
  inputs: $percept$, a percept indicating the current state s' and reward signal r'
  persistent: Q, a table of action values indexed by state and action, initially zero
    N_{sa}, a table of frequencies for state–action pairs, initially zero
    s, a, r, the previous state, action, and reward, initially null

  if TERMINAL?(s) **then** $Q[s, None] \leftarrow r'$
  if s is not null **then**
    increment $N_{sa}[s, a]$
    $Q[s, a] \leftarrow Q[s, a] + \alpha(N_{sa}[s, a])(r + \gamma \max_{a'} Q[s', a'] - Q[s, a])$
  $s, a, r \leftarrow s', \text{argmax}_{a'} f(Q[s', a'], N_{sa}[s', a']), r'$
  return a

Reinforcement learning

- We can estimate Q(s,a) and thus the policy while executing an exploration policy
- More at the last lecture?