Reinforcement Learning Contd.

Passive RL

- How to learn from already given experiences
- Given:
  - Set of states $s \in S$
  - Set of actions (per state) A
- Looking for $\pi(s)$
- Don't know
  - A model $T(s, a, s')$
  - A reward function $R(s, a, s')$
- Big idea $\triangleq$ compute all averages over T using sample outcomes

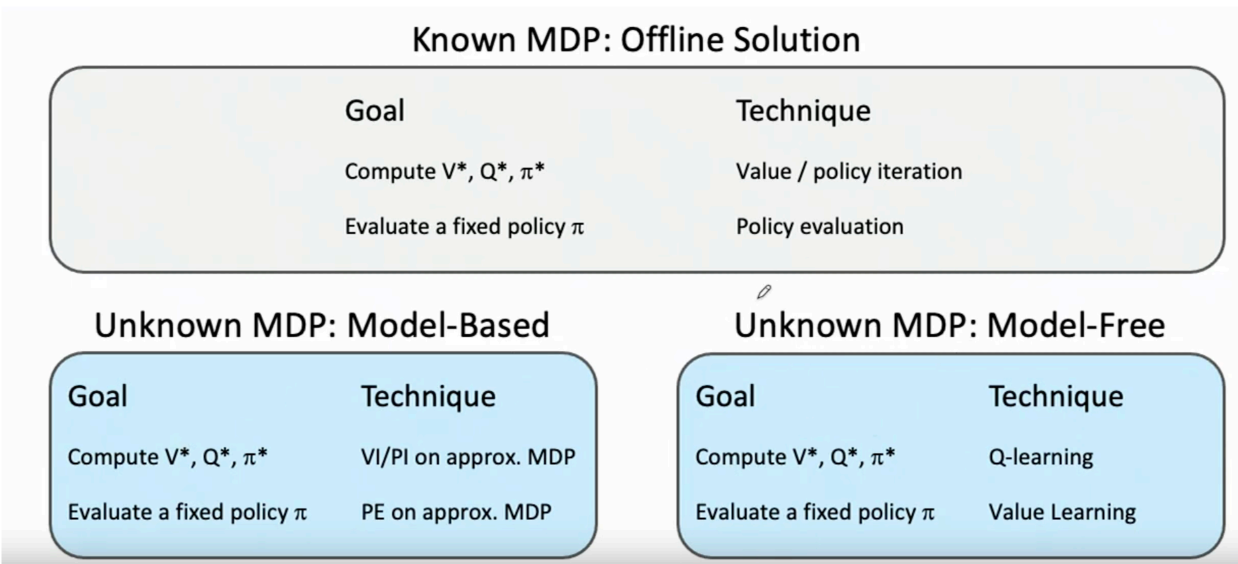

Figure 1: Screenshot_2023-10-02_at_5.18.04_PM.png

-

Model-Free Learning

- Temporal difference learning
- Receives stream of experiences from the world eg: $(s, a, r, s', a', r', s'', ...)$
- Update estimates each transition
- Over time, updates will mimic Bellman updates
- Q-iteration: do Q-value updates to each Q-state:

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

  - $Q_0(s, a) = 0$
  - Can't compute this update without T and R
- Q-learning: instead, compute average as we go
  - Receive a sample transition (s, a, r, s')

– Use initial approximation $Q(s,a) \approx r + \gamma \max_{a'} Q(s',a')$ to compute:

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

– $\pi^*(s) = \arg\max_a(Q(s,a))$
– AKA off-policy learning
– Have to explore enough to be good
– Have to eventually make the learning rate small enough (but not decrease it too quickly)
– Type of active RL

Active RL

- Need to consider exploration vs exploitation

Choosing How to Explore

- Simplest: random actions (ε-greedy)
  - Every time step, flip a coin
  - With (small) probability ε, act randomly
  - With (large) probability $1-\varepsilon$, act on current policy
  - ε high during early learning stage and low at end
- Better idea: explore areas whose badness is not (yet) established; eventually stop exploring
  - Use an exploration function that takes a value estimate u and a visit count n and returns an optimistic utility, eg: $f(u,n) = u + \frac{k}{n}|_{k \,\hat{=}\, \text{preset constant}}$
  - Then use

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

    - Note: $x \leftarrow_\alpha \implies x \leftarrow (1-\alpha)x + \alpha v$

Regret

- Even if you learn the optimal policy, you still make mistakes along the way
- Regret $\triangleq$ a measure of total mistake cost
  - Difference between your (expected) rewards, including youthful suboptimality, and optimal (expected) rewards
- eg: random exploration and exploration functions both end up optimal, but random exploration has higher regret

Dealing with Large State Spaces

- Use approximate Q-learning
- Basic Q-Learning keeps a table of all q-values
  - This is not possible in reality; too much state
- Idea: Learn about some small number of training states from experience
  - Generalize that experience to new, similar situations
- Solution: linear value functions: describe state using a vector of features (properties) $(f_1, f_2, f_3, ..., f_n)$
  - Features are functions from states to real numbers (often 0/1) that capture important properties of the state
    - eg: distance to closest ghost, is pacman in a tunnel, etc.
    - Can also describe a q-state (s,a) with features
- Using a feature representation, we can write a Q-function (or value function) for any state using a few weights $(w_1, w_2, ..., w_n)$:

$$V(s) = w_1 f_1(s) + w_2 f_2(s) + ... + w_n f_n(s)$$
$$Q(s,a) = w_1 f_1(s,a) + w_2 f_2(s,a) + ... + w_n f_n(s,a)$$

  - Advantage: our experience is summed up in a few powerful numbers $\{w_1, w_2, ..., w_n\}$
  - Disadvantage: states may share features but actually be very different in value
- Q-learning with linear Q-functions:

$$\begin{aligned}
&\text{transition} = (s, a, r, s') \\
&\text{difference} = [r + \gamma \max_{a'} Q(s', a')] - Q(s, a) \\
&Q(s, a) \leftarrow Q(s, a) + \alpha[\text{difference}] \\
&w_i \leftarrow w_i + \alpha[\text{difference}] f_i(s, a)
\end{aligned}$$