HW2 Review

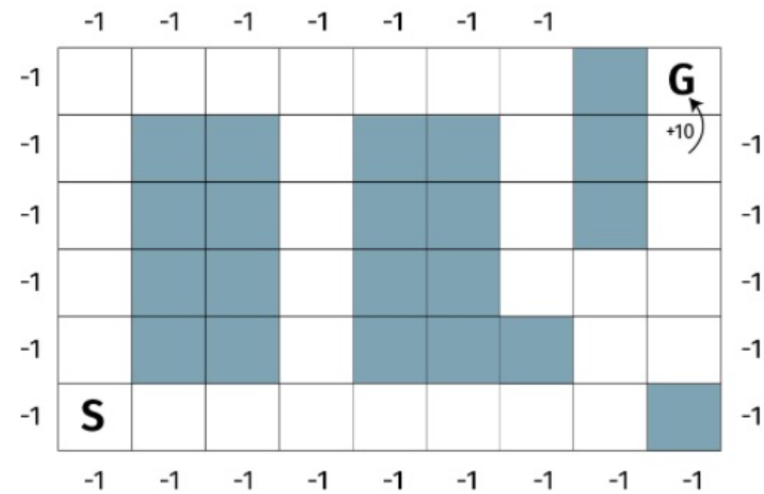
CCM Spring 2022

Yanli Zhou

Problem 1 - Dynamic Programming via Policy Iteration

Two key representations:

1. Policy $\pi(s, a)$ – probability mapping from states to actions
   - Maintained in the policy table
2. Value of each state under the current policy V^{π}
   - Maintained in the value table



Two key stages:

1. Policy evaluation: use the current policy to get values of states
   - Updating the value table given current policy
2. Policy improvement: use values of states to choose actions
   - Updating the policy table given computed values

Outer loop:

```
for i in range(large number):
    policy_evaluate
    policy_update
```

Problem 1 - Dynamic Programming via Policy Iteration

Policy evaluation: Bellman equation

$$V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'} [\mathcal{R}^a_{ss'} + \gamma V^{\pi}(s')]$$

Inner loop:

for state in randomized(all valid states):
    for action in all actions:
        - get $s', P^a_{ss'}, R^a_{ss'}, V^{\pi}(s')$
        - calculate value using Bellman equation
    update $V^{\pi}(s)$

Problem 1 - Dynamic Programming via Policy Iteration

Policy improvement:
Q-values: state-action pairs

$$\begin{aligned}\pi'(s) &= \arg\max_a Q^\pi(s,a) \\ &= \arg\max_a E\left\{r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_t = s, a_t = a\right\} \\ &= \arg\max_a \sum_{s'} \mathcal{P}^a_{ss'}\Big[\mathcal{R}^a_{ss'} + \gamma V^\pi(s')\Big],\end{aligned}$$

Inner loop:

```
for state in all valid states:
    for action in all actions:
        - get s′, R^a_{ss′}, V^π(s′)
        - calculate Q-value Q^π(s, a)
    update π(s) greedily using Q-value
```

Q-value $\neq$ Q-learning

Q-value:

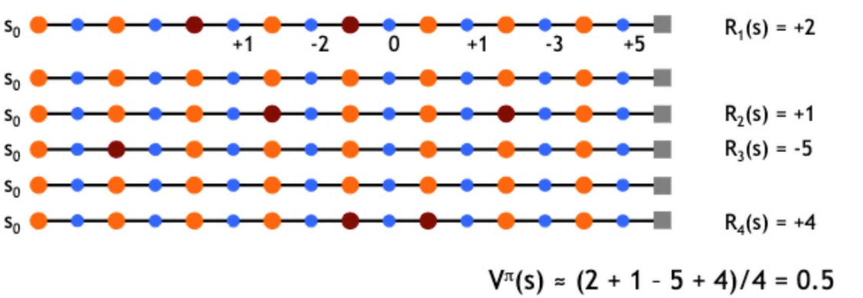
- Just the name for the value that is associated with an action
- Many RL methods involve estimating Q-values

Q-learning:

- A type of RL learning algorithm

Problem 2 – First-visit Monte Carlo

- We generally don't have information on $P_{ss'}^{a}$, or $R_{ss'}^{a}$ - in this problem we don't have access to these variables.
- Still, we would like to estimate the value of each state-action pair, $Q^{\pi}(s, a)$
- In first-visit Monte-Carlo, we estimate this by recording the rewards received after the first visit of a state-action pair until the end of an episode and averaging over many episodes.



Outer loop:

for i in range(n_episodes):
    select random start state
    generate episode
        episode inner loop
    update Q value table - discounted_average(Returns(s,a))
    improve policy - $argmax_a(Q(s,a))$

Problem 2 – First-visit Monte Carlo

Inner loop:

for t in range(episode steps T):
    get current state s and action a
    if visiting (s, a) for the first time:
        get the subsequent steps
        calculate the discounted return*
        update the returns table – both $rsum$, and n
    (s, a) has now been visited

*Discounted return = $R_{t+1} + \gamma R_{t+2} + +\gamma^2\ R_{t+3} + \cdots$

Problem 4 – Bandit problem: Incremental ϵ-greedy

- K-arm bandit problem: we want to pull the arm that we think would give the highest reward
- We would like to estimate the value of each action (no state here), $\mathrm{Q}(a)$

$$V(s) = V(s) + \frac{1}{n(s)}[R - V(s)]$$

- What are some of the things our agent should keep track of/ have as attribute?
  $Q(a), N(a), \epsilon$

- How does the agent choose actions?
  - With $\mathrm{p} = \epsilon$: choose randomly from k arms
  - With $p = 1 - \epsilon$: choose $argmax_a(Q)$

- How does the agent learn?
  1. Chooses arm a and pull
  2. Observe reward
  3. Update $N(a)$
  4. Update $\mathrm{Q}(a)$

Problem 5 – Bandit problem: Constant step size ϵ-greedy

- Constant step size MC update with α

$$V(s) = V(s) + \alpha[R - V(s)]$$

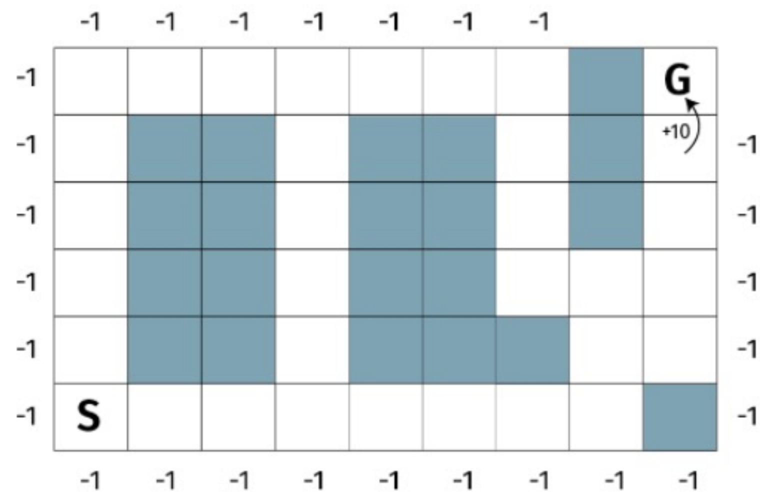
- What are some of the things our agent should keep track of/ have as attribute?
  $Q(a), \epsilon, \alpha$

- How does the agent choose actions?
  - With $\mathrm{p} = \epsilon$: choose randomly from k arms
  - With $p = 1 - \epsilon$: choose $argmax_a(Q)$

- How does the agent learn?
  1. Chooses arm a and pull
  2. Observe reward
  3. Update $\mathrm{Q}(a)$

Problem 7 – Grid world with Q-learning

- When does Q-value table update occur, between or within episodes?
- What are some key values to keep track of?
  - $Q(s,a), \epsilon, \gamma, \alpha$, policy table
- Hint: modify mc_episode() function such that it updates Q-value table within episode



- Initialise $Q(s,a)$
- Repeat `many times`
  - Pick s `start state`
  - Repeat `each step to goal`
    - ∗ Choose a based on $Q(s,a)$ `ε-greedy`
    - ∗ Do a, observe r, s'
    - ∗ $Q(s,a) = Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
    - ∗ $s = s'$
  - Until s terminal