JUSTINE JOHN A. SERODNCILLO
IE 5321 DYNAMIC PROGRAMMING & REINFORCEMENT LEARNING

OCTOBER 9, 2023
HOMEWORK #2

EXERCISE 3.7

{Robot running a maze} → +1 escape episodic task; 0 otherwise

Maximize $G_t \doteq R_{t+1} + R_{t+2} + \dots R_t$ (3.7)

run ⇒ !! no improvement — why?

The reward is the same regardless of amount of time in the maze. Need to communicate that the faster to escape the better.

EXERCISE 3.8 $\gamma = 0.5$, $R_{n\in[1,5]} = [-1, 2, 6, 3, 2]$ $T = 5$ $G_{n\in[0,5]} = ?$

Using (3.8) discounted return $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

$G_5 = \sum_{k=0}^{\infty} (0.5)^k R_{6+k}$ but $R_{6+k} = 0$ for $k = [0, \infty)$ then $G_5 = 0$

$G_4 = R_5 + \gamma G_5 = 2 + 0.5(0) = 2$
$G_3 = R_4 + \gamma(G_4) = 3 + 0.5(2) = 4$
$G_2 = R_3 + \gamma(G_3) = 6 + 0.5(4) = 8$
$G_1 = R_2 + \gamma(G_2) = 2 + 0.5(8) = 6$
$G_0 = R_1 + \gamma(G_1) = -1 + 0.5(6) = 2$

$$G_{n\in[0,5]} = [2, 6, 8, 4, 2, 0]$$

EXERCISE 3.9 $\gamma = 0.9$ $R_1 = 2$ $R_{[2,\infty)} = 7$ G_0 & G_1 ?

Using (3.8) $G_1 = \sum_{k=0}^{\infty} (0.9)^k R_{2+k}$ = infinite series of $7[1 + 0.9 + 0.9^2 + \dots] = 7 \cdot \frac{1}{1-0.9} = 70$

$G_0 = R_1 + \gamma G_1 = 2 + 0.9(70) = 65$

$$G_0, G_1 = 65, 70$$

Exercise 3.12 Give $v_\pi = f(q_\pi \& \pi)$

$$v_\pi = \sum_{a \in A(s)} \pi(a|s) q_\pi(s,a)$$

$a \in A(s)$: sum through all actions

$\pi(a|s)$: probability of action a given s

$q_\pi(s,a)$: ~ state-action value function

Exercise 3.13 Give $q_\pi = f(v_\pi, p)$

$$q_\pi = \sum_{s' \in S} \sum_{r \in R} p(s', r | s, a)(r + \gamma v_\pi(s'))$$

$s' \in S$: sum all new states

$r \in R$: sum all rewards

$p(s', r | s, a)$: probability of new state & reward given state & action

r: reward

γ: discount

$v_\pi(s')$: value of new state

EXERCISE 3.15

Gridworld → $r = "+"$ goals
→ $r = "-"$ edge
→ $r = 0$ otherwise

sgn(r) important? / intervals?

Prove w/ (3.8) $\forall r + c \neq$ new values of states

find $V_c = V_c(c, \gamma)$

---

using (3.8)

$$G_t = \sum_{k=0}^{\infty} \gamma^k (R_{t+k+1})$$

↳ R_n can be "−", 0, "+"

now add c to all Rewards $\Rightarrow$ $G'_t = \sum_{k=0}^{\infty} \gamma^k (R_{t+k+1} + c)$

↳ can add c such that R_n is all "+", all same sign or "−"

$$G'_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} + \sum_{k=0}^{\infty} c\gamma^k = G_t + \frac{c}{1-\gamma}$$

$$\boxed{V_c = \frac{c}{1-\gamma}}$$

EXERCISE 3.17 Bellman equations for a, q_π

$$q_\pi(s,a) = f(q_\pi(s',a'))$$

Using result from EXERCISE 3.13 $q_\pi(s,a) = \sum_{s' \in S} \sum_{r \in R} p(s', r | s, a) [r + \gamma V_\pi(s')]$ (A)

& EXERCISE 3.12 $V_\pi(s) = \sum_{a \in A} \pi(a|s) \cdot q_\pi(s,a)$ (B)

plug in (B) to (A)

$$\boxed{q_\pi(s,a) = \sum_{s', r} p(s', r | s, a) \left[r + \sum_{a' \in A} \pi(a'|s') \cdot \gamma q_\pi(s', a') \right]}$$

EXERCISE 3.25. Give $v_* = V_*(q_*)$

$$V_*(s) = \max_{a \in A(s)} q_*(s,a)$$

$q_*(s,a)$: optimal q-factor

EXERCISE 3.26 Give $q_* = q_*(V_*, p)$.

$$q_*(s,a) = \max \sum_{s' \in S} \sum_{r \in R} p(s', r \mid s, a) \left[r + \gamma V_*(s') \right]$$

$V_*(s')$: best value

EXERCISE 3.27 Give $\pi_* = \pi_*(q_*)$

$$\pi_*(a \mid s) = \max_{a \in A} q_*(s', a)$$

$q_*(s', a)$: best state-action value

EXERCISE 4.7 Write a program for policy iteration & resolve Jack's car rental problem w/ the following changes

① Jack's employer at 1st rides bus & lives in 2nd. Shuttle 1 car to 2nd for free

② 10 cars limit at each location, 4$ cost for extra

From Example 4.2 Jack's Car Rental

10$ per car rented

2$ to move/car

	①	②
requests	$P(n) = \frac{3^n}{n!} e^{-3}$ Poisson	$P(n) = \frac{4^n}{4!} e^{-4}$
return	$P(n) = \frac{3^n}{n!} e^{-3}$	$P(n) = \frac{2^n}{2!} e^{-2}$
	max 20	max 20

max 5 cars move/night

$\gamma = 0.9$

Also use Policy Iteration Method.

EXERCISE 9.5 How would policy iteration be defined for action values? Give a complete algorithm for computing q_π, analogous to that on page 80 for computing v_π. Please pay special attention to this exercise, because the ideas involved will be used throughout the rest of the book.

Policy iteration for action values is essentially trying to find the best policy

1. Initialization

$V(s) \in \mathbb{R}$ & $\pi(s) = A(s)$ arbitrarily $\forall s \in S$; $V(\text{terminal}) = 0$

$Q(s,a)$ arbitrarily $\forall s \in S, a \in A(s)$

Set learning rate & discount

while Q not converged

while S not terminal

$\pi(s) = \underset{a \in A(s)}{\text{argmax}}\, Q(s,a)$

Store action, reward & new state

Calculate new $Q(s,a)$ based on $Q(s,a)$ & learning rate

Set s as the s'

return Q