بسم تعالی

هوش مصنوعی

تمرین سوم

استاد:

مهدی سمیعی

نویسنده :

محمدهومان کشوری

شماره دانشجویی :

99105667

تمرینات تئوری

با تشکر از آقای هیربد بهنام برای کمک در حل سوالات

سوال 1.

الف)

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s')\right]$$

ابتدا Transition , Reward ها را مشخص می‌کنیم.

$R(s_i$, shoot, goal$) = 3$, $i = \{1,5,6,10\}$

$R(s_i$, shoot, goal$) = 2$, $i = \{2,3,4,7,8,9\}$

$R(s_i$, shoot, miss$) = -10$

$R(s_i$, change position, $s_{i`}) = -1$

$T(s_i$, shoot, Done$) = 1$

T(1, change position, s\`) = 1 , s\` = { 2, 6 }

T(2, change position, s\`) = 1 , s\` = { 1, 3, 7 }

T(3, change position, s\`) = 1 , s\` = { 2, 4, 8 }

…

T(s, change position, s\`) = 1 ,  s = { 2, 3, 4, 6, 7, 8, 9 }, s\` = {(s-1, s+1, s+-5)}

T(s, change position, s\`) = 1  ,  s = { 1, 5, 6, 10 }, s\` = { (s+-1, s+-5) }

T(s, a’, s’) = 0 , o.w

$V_i(1)$ = max(shoot, change position)

V_i(1, shoot) = T(1, shoot, Done) * (R(1, shoot, goal) * P(goal | 1) + R(1,shoot, miss) * P(miss | 1) + γ * V_{i-1}(Done))

V_i(1, change position) = T(1, change position, 2) * [R(1, change position, 2) + γ * V_{i-1}(2)] + T(1, change position, 6) * [R(1, change position, 6) + γ * V_{i-1}(6)]

****

$V_i(2)$ = max(shoot, change position)

V_i(2, shoot) = T(2, shoot, Done) * (R(2, shoot, goal) * P(goal | 2) + R(2,shoot, miss) * P(miss | 2) + γ * V_{i-1}(Done))

V_i(2, change position) = T(2, change position, 1) * [R(2,change position, 1) + γ * V_{i-1}(1)] + T(2, change position, 3) * [R(2,change position, 3) + γ * V_{i-1}(3)] + T(2, change position, 7) * [R(2,change position, 7) + γ * V_{i-1}(7)]

****

دو حالت بالا برای استیت‌های ۱ و ۲ گرفته شده‌اند. می‌دانیم حالت‌های ۱،۵،۶،۱۰ **کاملا مشابه** هستند و نیز حالت‌های ۲،۳،۴،۷،۸،۹ نیز **تقریبا مشابه‌اند** با این تفاوت که در احتمالات گل زدن آنها کمی تفاوت وجود دارد. (عملا در دومین مورد حالات بجز ۳ نیز کاملا مشابه‌اند).

برای بررسی دقیق‌تر می‌توان **طبق تقارن** نتیجه زیر را گرفت :

$V_i(1) = V_i(5) = V_i(6) = V_i(10)$

$V_i(2) = V_i(4)$

$V_i(7) = V_i(9)$

$V_1(1, \text{shoot}) = 1 * (3 * 0.6 + -10 * 0.4 + 1 * 0) = -2.2$

$V_1(1, \text{change position}) = 1 * (-1 + 0), 1 * (-1 + 0)$

max $V_1(1) \rightarrow$ (shoot,change position) = **–1**

$V_1(2, \text{shoot}) = 1 * (2 * 0.75 + -10 * 0.25 + 1 * 0) = -1$

$V_1(2, \text{change position}) = 1 * (-1 + 0) ,1 * (-1 + 0) , 1 * (-1 + 0)$

max $V_1(2) \rightarrow$ (shoot, change position) = **–1**

$V_1(3, \text{shoot}) = 1 * (2 * 0.9 + -10 * 0.1 + 1 * 0) = 0.8$

$V_1(3, \text{change position}) = 1 * (-1 + 0), 1 * (-1 + 0), 1 * (-1 + 0)$

max $V_1(3) \rightarrow$ (shoot, change position) = **0.8**

——————————————————————

$V_2(1, \text{shoot}) = 1 * (3 * 0.6 + -10 * 0.4 + 1 * 0) = -2.2$

$V_2(1, \text{change position}) = 1 * (-1 + 1 * -1), 1 * (-1 + 1 * -1)$

max $V_2(1) \rightarrow$ (shoot, change position) = **–2**

$V_2(2, \text{shoot}) = 1 * (2 * 0.75 + -10 * 0.25 + 1 * 0) = -1$

$V_2(2, \text{change position}) = 1 * (-1 + 1 * -1), 1 * (-1 + 1 * 0.8), 1 * (-1 + 1 * -1)$

max $V_2(2) \rightarrow$ (shoot, change position) = **–0.2**

$V_2(3, \text{shoot}) = 1 * (2 * 0.9 + -10 * 0.1 + 1 * 0) = 0.8$

$V_2(3, \text{change position}) = 1 * (-1 + 1 * -1) ,1 * (-1 + 1 * -1), 1 * (-1 + 1* -1)$

max $V_2(3) \rightarrow$ (shoot, change position) = **0.8**

$V_2(7, \text{shoot}) = 1 * (2 * 0.75 + -10 * 0.25 + 1 * 0) = -1$

$V_2(7, \text{change position}) = 1 * (-1 + 1 * -1), 1 * (-1 + 1 * -1), 1 * (-1 + 1 * -1)$

max $V_2(7) \rightarrow$ (shoot, change position) = **-1**

$V_2(8, \text{shoot}) = 1 * (2 * 0.75 + -10 * 0.25 + 1 * 0) = -1$

$V_2(8, \text{change position}) = 1 * (-1 + 1 * -1), 1 * (-1 + 1 * 0.8), 1 * (-1 + 1 * -1) =$ **-0.2**

max $V_2(8) \rightarrow$ (shoot, change position) = **-0.2**

State	1	2	3	4	5	6	7	8	9	10
V_0	∘	∘	∘	∘	∘	∘	∘	∘	∘	∘
V_1	-1	-1	0.8	-1	-1	-1	-1	-1	-1	-1
V_2	-2	-0.2	0.8	-0.2	-2	-2	-1	-0.2	-1	-2

ب)

الگوریتم کیولرنینگ :

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

$$sample = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + (\alpha)\,[sample]$$

Episode 1 :

Q(s_8, move R) = (1 - 0.5) * Q(s_8, move R) + 0.5 * (R(s_8, move R, s_9) + γ * max Q(s_9,a')) = 0.5 * 0 + 0.5 * (-1 + 1 * max[Q(s_9,move R),Q(s_9,move L)]) = 0.5 * (-1 + 1 * max [0,0]) = -0.5

Q(s_9 move U) = 0.5 * Q(s_9, move U) + 0.5 * (R(s_9, move U, s_4) + γ* max(s_4, a')) = 0 + 0.5 * -1 = -0.5

Q(s_4, shoot) = 0.5 * Q(s_4, shoot) + 0.5 * (R(s_4, shoot, goal) + γ * max(goal, a')) = 0 + 0.5 * 1 = 0.5

——————————————————

Episode 2 :

Q(s_8, move R) = 0.5 * Q(s_8, move R) + 0.5 * (R(s_8, move R, s_7) + γ * max(s_7, a')) = 0.5 * -0.5 + 0.5 * (-1 + 0) = -0.75

Q(s_7, move R) = 0.5 * Q(s_7, move R) + 0.5 * (R(s_7, move R,s_8) + γ * max(s_8, a')) = 0.5 * 0 + 0.5 * (-1 + 1 * max(-0.75, 0)) = -0.5

Q(s_8, shoot) = 0.5 * Q(s_8, shoot) + 0.5 * (1 + 1 * 0) = 0.5

Episode 3 :

$Q(s_8$, move R) = 0.5 * $Q(s_8$, move R) + 0.5 * (-1 + 1 * max(-0.5, 0)) =

-0.5 * 0.75 + 0.5 * -1 = -0.875

$Q(s_9$, move R) = 0.5 * $Q(s_9$, move R) + 0.5 * (-1 + 1 * 0) = -0.5

$Q(s_{10}$, shoot) = 0.5 * $Q(s_{10}$, move R) + 0.5 * (2 + 0) = 1

—----------------------------------

Episode 4 :

$Q(s_8$, move R) = 0.5 * -0.875 + 0.5 * (-1 + 0) = -0.9375

$Q(s_7$, move L) = 0.5 * 0 + 0.5 * (-1 + 1 * max (0.5, -0.9375)) = -0.25

$Q(s_8$, shoot) = 0.5 * 0.5 + 0.5 * (-5 + 0) = -2.25

—----------------------------------

Episode 5 :

$Q(s_8$, move R) = 0.5 * -0.9375 + 0.5 * (-1 + 0) = -0.96875

$Q(s_9$, move U) = 0.5 * -0.5 + 0.5 * (-1 + 1 * max(1,0)) = -0.25

$Q(s_{10}$, shoot) = 0.5 * 1 + 0.5 * (-5 + 0) = -2

سوال 2.

الف) درست

ب) نادرست، این سیاست‌ها در این برنامه در حال قوی شدن هستند و در فضای حالت بزرگ، policy iteration سریع‌تر به سیاست مورد نظر می‌رسد.

پ) درست

ت) نادرست، با ضریب تخفیف کوچکتر از ۱ نمی‌توان پاداش منفی تولید کرد.

ث) درست

سوال 3.

الف)

5	A	S	B	C	10
			0	0	

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s')\right]$$

Start from C ⇒ $10 * p + (1-p) * 0 = 10p$

$B = p * (0 + \gamma\, 10 * p + (1 - p) * 0) + (1-p) * 0 = 10\gamma p^2$

$S = \gamma * (p * (0 + \gamma\, 10 * p + (1 - p) * 0) + (1-p) * 0) = 10\gamma^2 p^2$

$A = \gamma * (\gamma * (p * (0 + \gamma\, 10 * p + (1 - p) * 0) + (1-p) * 0)) = 10\gamma^3 p^2$

ب)

Start from A :

$A = 5$

$S = 5\gamma$

$B = p * (\gamma * 5\gamma) + (1-p) * 0 = 5\gamma^2 p$

$C = p * (\gamma * (p * (\gamma * 5\gamma) + (1-p) * 0)) + (1-p) * 0 = 5\gamma^3 p^2$

ج)

می‌دانیم درنهایت باید به جواب قابل قبول همگرا شویم پس :

$10p > 10\gamma p^2$ ⇒ بدیهتا درست چرا که ضریب تخفیف بین ۰ و ۱ و همچنین احتمال نیز بین ۰ و ۱

همچنین سیاست رفتن به شرق باید بهتر از سیاست رفتن به غرب عمل کند.

$10p > 5 \Rightarrow p > ½$,

$10p > 5\gamma^3 \Rightarrow p > \gamma^3/2$

$5 > 5\gamma^3 p^2 \Rightarrow p < \frac{1}{\sqrt{\gamma^3}}$

د)

هر دوی این توابع، الگوریتم‌هایی برای پیدا کردن سیاست بهینه در مسئله مارکوف هستند.
value iteration با توجه به مرحله فعلی سعی می‌کند value هر حالت را بهینه کند تا جایی که مقادیر، به مقادیر واقعی همگرا شوند.
در policy iteration ، یک سیاست تقریبی برای حل مسئله در نظر گرفته می‌شود و هر چند قدم سعی در بهتر کردن سیاست می‌شود.
تفاوت این دو در این است که در policy iteration در ابتدا نیاز به یک سیاست تقریبی داریم در صورتی که در value iteration نیاز به همچین سیاستی نیست.
در policy iteration هدف بهینه کردن سیاست است و هر چندمرحله سیاست ما بهتر می‌شود در نتیجه می‌توان گفت **policy iteration سریع‌تر از value iteration** به **سیاست بهینه** می‌رسد.
همچنین به علت چندمرحله‌ای بودن policy iteration ، برای محاسبه پیچیدگی محاسباتی بیشتری نسبت به value iteration دارد.

سوال 4.

الف)

اولین مشکل این تابع این است که معمولا در پیدا کردن utility اطلاعات جدیدتر اهمیت بیشتری از اطلاعات قدیمی دارند اما در این تابع ذکر نشده، در U_2 با دادن یک ضریب تخفیف این مشکل برطرف شده.

دومین مشکل این تابع این است که در صورتی که پاداش‌ها کوچک باشند، نمی‌توانیم تمایزی بین حالت‌ها بگذاریم و عملا بین دو گزینه یکسان انتخابی نداریم.

ب)

در صورتی که تعداد وضعیت‌های ما کم باشند بله اما مثلا فرض کنید بازی‌ای مانند شطرنج داشته باشیم که تعداد وضعیت‌های آن بسیار زیاد باشند، در این صورت روش‌های stationary preference نمی‌توانند خوب عمل کنند.

یا اینکه مثلا فرض کنید در طول بازی، نیازی به تعویض نوع بازی agent باشیم، در این صورت گزینه‌های بهتری نسبت به stationary preference وجود خواهد داشت.

ج)

ابتدا به این نکته توجه می‌کنیم که ضریب تخفیف(γ) عددی بین ۰ و ۱ است چرا که در غیر این صورت تابع ما اصلا همگرا نخواهد بود!!!

و نکته دوم این که

$$\sum_{i=0}^{\infty} x^i = \frac{1}{1-x} \quad \text{when} \quad |x| < 1$$

حال بررسی را انجام می‌دهیم.

$$\text{I)}\ R(s_i) \geq R_{min} \Rightarrow \sum_{i=0}^{inf} \gamma^i R_{min} \leq \sum_{i=0}^{inf} \gamma^i R(s_i) \rightarrow R_{min} * \sum_{i=0}^{inf} \gamma^i$$

$$|\gamma| < 1 \Rightarrow R_{min} * \sum_{i=0}^{inf} \gamma^i = R_{min} * \frac{1}{1-\gamma} = \frac{R_{min}}{1-\gamma}$$ ✅

II) $R(s_i) \ \leq \ R_{max} \ \Rightarrow \ \sum_{i=0}^{inf} \gamma^i R(s_i) \ \leq \sum_{i=0}^{inf} \gamma^i R_{max} \ = \ R_{max} \sum_{i=0}^{inf} \gamma^i$

$|\gamma| \ < \ 1 \ \Rightarrow R_{max} \ * \sum_{i=0}^{inf} \gamma^i \ = R_{max} \ * \ \frac{1}{1-\gamma} \ = \ \frac{R_{max}}{1-\gamma}$ ✅

I + II $\Rightarrow \frac{R_{min}}{1-\gamma} < \sum_{i=0}^{inf} \gamma^i R(s_i) < \frac{R_{max}}{1-\gamma}$

حال نشان دادیم که U_2 کراندار است.