دانشکده مهندسی کامپیوتر

دکتر محمدرضا محمدی

بهار ۱۴۰۰

تمرین سری دوازدهم

یادگیری عمیق

مجتبی نافذ 96431335

مهلت تحویل : ۲۴ خرداد ۱۴۰۰ ساعت ۲۳:۵۹:۵۹

---

۱. به سوالات زیر در رابطه با یادگیری تقویتی پاسخ بدهید.

الف) تفاوت میان Exploration و Exploitation چیست؟ آیا عاملی که فقط Exploit و یا فقط Explore می‌کند، می‌تواند موفق عمل کند؟ توضیح دهید.

ب) تفاوت میان ارزش وضعیت و پاداش را توضیح دهید.

پ) فرض کنید در یک مسئله *episodic* پاداش‌های زیر دریافت شود و این قسمت در $T = 5$ به پایان برسد.

$$R_1 = 2, R_2 = 0, R_3 = -1, R_4 = 2, R_5 = 8$$

مقادیر بازده (return) برای تمام گام‌ها را به ازای $\gamma = 0.5$ و $\gamma = 1.0$ محاسبه کنید.

در exploitation در واقع agent ما در هر state ترجیح می دهد action هایی را انتخاب کند که در گذشته انتخاب کرده و در گذشته بیشترین پاداش را داشته اند

اما در exploration در واقع agent ما برای اکتشاف اکشن هایی را انتخاب می کند که تا به حال دیده نشده اند.

و عاملی که یکی را انتخاب کند نمی تواند موفق عمل کند. چون اگر فقط explore کند که تا فقط رندوم خواهد بود و هیچگاه اکشن های بهینه را تضمین نخواهد کرد.

و اگر فقط exploit کند با ممکن است به یک پاداش کم قانع شود و دیگر هیچ گاه دنبال بهتر از آن اکشن نباشد و هیچگاه هم بهترین اکشن را پیدا نخواهد کرد.

ارزش وضعیت: مقدار expected discounted return ای که agent با شروع از start state و عمل کردن براساس policy مورد نظرش (انتخاب state با بیشترین ارزش)به دست می آورد.

$$\underbrace{v_\pi(s)}_{\text{Value function}} = \underbrace{\mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots\right]}_{\text{Expected discounted return}} \Big| \underbrace{S_t = s}_{\text{Starting at state s}}$$

پاداش: مقدار بازخورد agent در هر state جدید.

وارد هر state که می شویم یک مقدار پاداش یا همان بازخورد دریافت می کنیم.

تفاوت شان این بود که پاداش مقدار پاداش تنها یک state است اما ارزش یک state مقدار مجموع تخفیف دار پاداش ها از آن state تا رسیدن به هدف است.

***γ=1* :**

گام صفرم : $V\pi(s) = 2 + 0 - 1 + 2 + 8 = 11$

گام اول : $V\pi(s) = 0 - 1 + 2 + 8 = 9$

گام دوم : $V\pi(s) = -1 + 2 + 8 = 9$

گام سوم: $V\pi(s) = 2 + 8 = 10$

گام چهارم : $V\pi(s) = 8 = 8$

$\gamma=0.5$:

گام صفرم : $V\pi(s) = (1)*2 + (0.5)*0 - (0.25)*1 + (0.125)*2 + (0.0625)*8 = 2.5$

گام اول : $V\pi(s) = (1)*0 - (0.5)*1 + (0.25)*2 + (0.125)*8 = 1$

گام دوم : $V\pi(s) = (1)*-1 + (0.5)*2 + (0.25)*8 = 2$

گام سوم: $V\pi(s) = (1)*2 + (0.5)*8 = 6$

گام چهارم : $V\pi(s) = (1)*8 = 8$

۲. همانطور که می‌دانیم در روش‌های یادگیری Temporal Difference و Monte Carlo نیازی به داشتن مدلی از محیط نداریم، یعنی از قبل مشخص نیست با انجام یک عمل، با چه احتمالی به کدام حالت می‌رویم و چه پاداشی می‌گیریم، به همین دلیل به این روش‌ها model-free می‌گویند. در مورد این دو روش به سوالات زیر پاسخ دهید.

الف) به نظر شما کدام یک از این دو روش برای مسائل اپیزودیک و کدام یک برای مسائل ادامه‌دار (continuous) مناسب است؟

ب) تعداد دفعات به روز رسانی ارزش حالت‌ها در کدام روش بیشتر است؟

پ) فرض کنید نتایج حاصل از ۳ اپیزود در یک مساله اپیزودیک به صورت زیر بوده است و دنباله حالات و پاداش‌های زیر تاکنون بدست آمده: (حروف نشان دهنده حالت‌ها هستند و پس از آنها پاداش بدست آمده به صورت یک عدد نوشته شده است).

Episode1: A, 0, B, 0, C, 0, D, 1, T
Episode2: B, 1, C, 1, T
Episode3: D, 0, T

ت) فرض کنید ارزش اولیه همه‌ی حالات 0 و $\alpha = 0.2, \gamma = 0.9$ باشند. ارزش حالات A, B, C, D را پس از این اپیزودها با دو روش TD(0) و Monte Carlo بدست آورید.

الف) روش Monte Carlo مناسب است برای مسائل اپیزودیک زیرا در Monte carlo ما ابتدا یک episode را

کامل پیش می رویم و سپس Gt یا همان return را محاسبه و برای آپدیت Vst استفاده می کنیم پس به عبارت دیگر در Monte carlo قبل از آپدیت ما به یک episode کامل interaction با محیط نیازمندیم.
در مسایل continuous ما چون episode های مشخصی نداریم پس راه حل مناسب استفاده از Temporal Difference است. که هر step آپدیت ها را انجام دهد.

ب) به طبع در روش Temporal Difference به نظر میرسد تعداد آپدیت ها بیشتر است چون هر step از هر episode باید به روز رسانی انجام دهیم اما در Monte Carlo فقط هر episode یکبار آپدیت می کنیم.
اما در اصل تعداد آپدیت ها برابر است چون state های مسیر طی شده همگی این state ها یکبار آپدیت می شوند یکی یکجا همه را آپدیت می کند و یکی در طول زمان و روند.

ت)

Monte Carlo : initial values=0 , $\gamma=0.9$, $\alpha=0.2$

$$Gt = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots =$$

Episode 1:

$Ga = 1*0 + 0.9*0 + 0.81*0 + 0.729*1 = 0.729$

$V(a) = 0 + 0.2*(0.729 - 0) = 0.1458$

$Gb = 1*0 + 0.9*0 + 0.81*1 = 0.81$

$V(b) = 0 + 0.2*(0.81 - 0) = 0.162$

$Gc = 1*0 + 0.9*1 = 0.9$

$V(c) = 0 + 0.2*(0.9 - 0) = 0.18$

$Gd = 1*1 = 1$

$V(d) = 0 + 0.2*(1 - 0) = 0.2$

Episode 2:

$Gb = 1*1 + 0.9*1 = 1.9$

$V(b) = 0.162 + 0.2*(1.9 - 0.162) = 0.5096$

$Gc = 1*1 = 1$

$V(c) = 0.18 + 0.2*(1 - 0.18) = 0.344$

Episode 3:

$Gd = 1*0 = 0$

$V(d) = 0.2 + 0.2*(0 - 0.2) = 0.16$

Temporal Difference: initial values=0 , $\gamma=0.9$, $\alpha=0.2$

Episode 1:

$V(start) = 0 + 0.2*(0 + 0.9*0 - 0) = 0$

$V(a) = 0 + 0.2*(0 + 0.9*0 - 0) = 0$

$V(b) = 0 + 0.2*(0 + 0.9*0 - 0) = 0$

$V(c) = 0 + 0.2*(1 + 0.9*0 - 0) = 0.2$

Episode 2:

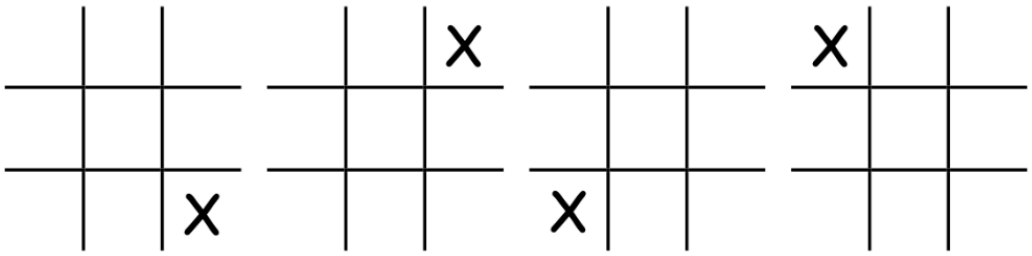
$V(start) = 0 + 0.2*(1 + 0.9*0 - 0) = 0.2$

$V(b) = 0 + 0.2*(1 + 0.9*0.2 - 0) = 0.236$

Episode 3:

$V(start) = 0.2 + 0.2*(0 + 0.9*0 - 0.2) = 0.16$

۳. در بازی tic tac toe حدود هشت هزار حالت مختلف ممکن است اتفاق بیافتد. از آنجایی که در این بازی، قرار گرفتن افقی، عمودی یا قطری مهره‌ها اهمیت دارد، برخی از حالت‌ها معادل با یکدیگر هستند و ارزش یکسانی دارند. به طور نمونه، هر ۴ حالت زیر دارای یک ارزش هستند و با یک احتمال مساوی منجر به پیروزی می‌شوند.



بنابراین، تعداد حالت‌های متمایز بسیار کمتر است. حال می‌توان روش را طوری تغییر داد که با کم شدن تعداد وضعیت‌ها، عامل سریع‌تر یاد بگیرد (نیاز نباشد هر کدام از ۴ حالت بالا بارها تجربه شوند و مستقل در نظر گرفته شوند). در پوشه تمرین دو فایل پایتون وجود دارد:

- tictactoe_self_play: دو عامل x و o همزمان در حال یادگیری هستند. این کد برای آموزش دو عامل است.
- tictactoe_human_ai: در این کد عامل آموزش دیده o با عامل انسانی x بازی می‌کند و برای آزمون عملکرد عامل آموزش دیده قابل استفاده است.

این دو فایل را مطالعه کنید و سپس فایل tictactoe_self_play را به صورت زیر تغییر دهید:

الف) در هر iteration یک پاداش دریافت می‌شود. با جمع‌آوری این پاداش‌ها نمودار پاداش را در انتهای آموزش رسم کنید.

ب) از ویژگی تقارن در tic tac toe استفاده کنید و تعداد حالت‌های موجود را کاهش دهید.

پ) نمودار پاداش را برای قسمت (**ب**) رسم کنید و با قسمت (**الف**) مقایسه کنید.

الف) کد ضمیمه شده است.

ب) کد ضمیمه شده است.

پ) پاداش ها در در دو حالت در نهایت باید به بازی های همیشه مساوی میل کند. و پاداش صفر بگیرند

در قسمت ب با کاهش تعداد state ها در نتیجه تعداد بیشتری اکشن و state را می توان تست کرد.

در نتیجه زودتر agent ها باهوش می شوند پس زودتر به حالت همیشه مساوی میل می کند.

ولی متاسفانه هر چه تلاش کردم کد این نتایج را نداد.