پروژه یادگیری تقویتی

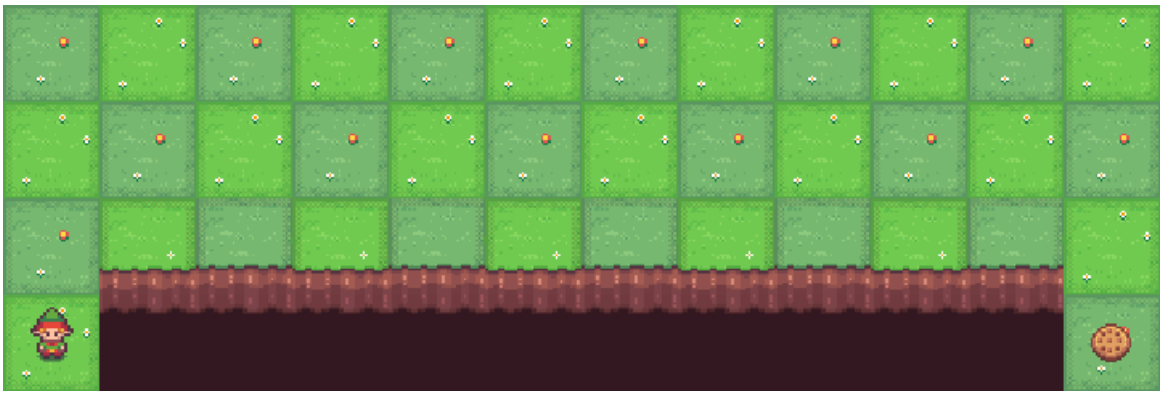
مبانی و کاربردهای هوش مصنوعی بهار ۱۴۰۳

در این پروژه قصد داریم دو الگوریتم یادگیری تقویتی Q-learning و Sarsa را پیاده سازی کنیم.

می‌خواهیم با استفاده از کتابخانه gymnasium OpenAI(0.29.1) در محیط CliffWalking-v0 با پیاده سازی الگوریتم Q-learning و Sarsa، به عامل نحوه حرکت در محیط را آموزش دهیم و به مقایسه عملکرد این دو الگوریتم بپردازیم. برای درک بهتر تعدادی متود جهت نمایش مقادیر سیاست و ارزش حالتها از قبل پیاده سازی شده و می‌توانید از آن ها استفاده کنید.

محیط Cliff Walking:

در این محیط عامل به دنبال مسیری برای رسیدن به خانه مقصد (خانه دارای شیرینی) می‌باشد به گونه ای که در دره نیوفتد (وارد خانه های دره نشود). بنابراین در این محیط عامل باید از S(Start) به G(Goal) بدون سقوط در دره C(Cliff) برسد. صخره‌ها در حاشیه محیط قرار دارند و اگر عامل در یکی از خانه‌های صخره قرار بگیرد، باید مجددا از خانه شروع (S) پیمایش را آغاز کند. به دلیل خطرناک بودن صخره‌ها، عامل باید با دقت حرکت کند. عامل می‌تواند در چهار جهت چپ، پایین، راست، بالا حرکت کند و در صورتی که حرکتی کند که سبب عبور از مرز محیط شود، موقعیتش تغییر نمی‌کند. با توجه به مسئله مورد بررسی، رسیدن به هر کدام از حالت‌های S و G و F و C دارای پاداش مشخصی می‌باشد. (برای توضیحات بیشتر می‌توانید صفحه اصلی آن را مطالعه کنید.)



فضای اکشن ها: اعداد ۰، ۱، ۲ و ۳ به ترتیب، اکشن های حرکت به بالا، راست، پایین و چپ را نشان می‌دهند.

فضای حالت ها: هر حالت (state) به صورت عدد بین ۰ تا ۴۷ نمایش داده می‌شود، به طوریکه خانه بالا سمت چپ خانه صفر، خانه بالا سمت راست خانه ۱۱، خانه شروع بازیکن خانه ۳۶ و خانه دارای شیرینی ۴۷ امین است. به همین نسبت برای هر خانه، عددی اختصاص داده شده.

پاداش ها: هر قدم بازیکن دارای پاداش 1- و خانه های صخره ای پاداش 100- دارند.

لینک صفحه اصلی:
https://gymnasium.farama.org/environments/toy_text/cliff_walking/

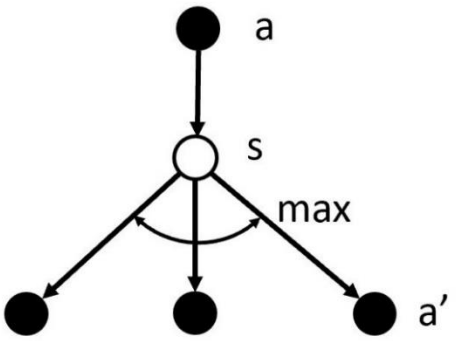
فاز اول: پیاده سازی الگوریتم Q-learning

کد قرار گرفته در ویو را دانلود و به پروژه خود اضافه کنید. همانطور که مشاهده میکنید، تمامی متودهای مورد نیاز جهت ارتباط با کتابخانه gymnasium ، متودهای نمایش نمودار و... از قبل پیاده سازی شده اند و می‌توانید با استفاده از آنها پروژه خود را انجام دهید. الگوریتم ابتدایی که باید پیاده سازی کنید Q-learning می‌باشد.

Q-learning یک الگوریتم تفاوت زمانی (Temporal Difference) برای یادگیری تقویتی است. برخلاف SARSA که یک روش کنترل on-policy است، Q-learning یک روش کنترل off-policy می‌باشد. تفاوت اصلی این دو الگوریتم در این است که Q-learning از سیاست به‌روزرسانی مستقل از سیاستی که برای جمع‌آوری داده‌ها استفاده می‌شود، بهره می‌برد. در Q-learning، عامل تلاش می‌کند تا بهترین اکشن ممکن را از طریق تجربه و تعامل با محیط بیاموزد.

در Q-learning، عامل با استفاده از جدول Q (ارزش اکشن‌ها) و با دریافت بازخورد از محیط، ارزش هر اکشن را به‌روزرسانی می‌کند. تفاوت اصلی با SARSA این است که در Q-learning، به‌روزرسانی Q بر اساس اکشن بهینه بعدی (بیشترین مقدار Q) انجام می‌شود، نه اکشن واقعی که عامل انتخاب می‌کند.

دیاگرام الگوریتم Q-learning را می‌توانید در شکل زیر مشاهده کنید:



$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Old Q Value — Reward — Discount Rate (0 ~ 1)

New Q Value — Learning Rate (0 ~ 1) — Maximum Q value of transition destination state

TD error

تفاوت این الگوریتم با الگوریتم های Policy Iteration و Value Iteration این است که ما در آنها به توزیع احتمال محیط دسترسی داشتیم و می‌توانستیم بدون آنکه با محیط تعاملی داشته باشیم، سیاست بهینه و ارزش حالت ها را برای آن محیط بدست آوریم. اما در زمان هایی که ما به طور مستقیم به این توزیع دسترسی نداریم، نیاز داریم تا عامل با محیط تعامل داشته و با توجه به بازخورد محیط نسبت به اکشن هایش و تجربه هایی که کسب می‌کند به یک پالیسی (نسبتا) بهینه برسد.

همانطور که در دیاگرام نشان داده شده، در این الگوریتم به هنگام آپدیت ارزش یک اکشن برای یک حالت، ابتدا میزان پاداش (reward) را با ماکزیمم پاداشی که از استیت بعدی می‌توانیم بگیریم جمع می‌کنیم و سپس با کم کردن آن از ارزش واقعی استیت فعلی، مقدار temporal difference error را محاسبه کرده تا این اختلاف را با ضریبی مشخص (نرخ یادگیری) به مقدار ارزش استیت فعلی اضافه کنیم.

بدین صورت Q-table (ارزش هر اکشن برای هر حالت) آپدیت شده و پس از انجام تعدادی episode (بازی تا زمان افتادن در دره یا بدست آوردن شیرینی)، مقادیر مناسبی برای Q-table خواهیم داشت که پالیسی نسبتا بهینه نهایی بر اساس آنها تعیین می‌شود.

همانطور که در سودوکد این الگوریتم مشاهده می‌کنید، عامل ما در یک حلقه شروع به کسب تجربه می‌کند و هر بار اجرای این حلقه به معنای شروع از یک استیت اولیه و انجام تعدادی اکشن (step) و دریافت پاداش است. تنها در صورتی که به یکی از استیت های نهایی (terminal state) برسد آن تجربه (episode) به پایان رسیده و این حلقه مجددا اجرا می‌شود.

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(terminal, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

از پالیسی های مطلوب، پالیسی ε-greedy می‌باشد. این سیاست به این صورت عمل کرده که با احتمال ε – 1 اکشن حریصانه (greedy) انتخاب می‌کند و با احتمال ε به صورت رندوم انتخاب می‌شود.

$$A \leftarrow \begin{cases} \arg\max_a Q(a) & \text{with probability } 1 - \varepsilon \quad \text{(breaking ties randomly)} \\ \text{a random action} & \text{with probability } \varepsilon \end{cases}$$

متغیر های مهم:

- cliffEnv: محیط پیاده سازی شده با استفاده از کتابخانه gymnasium

- q_table: آرایه ای که برای هر استیت، ارزش اکشن ها را به صورت یک آرایه چهارتایی ذخیره دارد. (هر خانه از این آرایه خود یک آرایه دارای چهار خانه است که هر خانه از آن، ارزش یک اکشن مشخص مانند حرکت به بالا برای آن استیت را نشان میدهد)

- NUM_EPISODES: تعداد اپیزود ها (راند های بازی)

- ALPHA (learning_rate): نرخ یادگیری برای تنظیم تاثیر temporal difference error

- EPSILON: احتمال explore در مقابل exploit

- GAMMA (discount factor): ضریب تنزیل که بر افق دید الگوریتم تاثیرگذار است (گاما - γ)

فاز دوم، پیاده سازی الگوریتم SARSA:

الگوریتم دومی که قصد بررسی آن را داریم، SARSA است که مخفف State-Action-Reward-State-Action می‌باشد. این الگوریتم یک روش کنترل policy-on از الگوریتم تفاوت زمانی (Temporal Difference) است.

توجه کنید که اکشن‌های A و A' مطابق با سیاست عامل و جدول Q (ارزش هر اکشن) انتخاب می‌شوند. همچنین، حالت S' و پاداش R توسط محیط مشخص می‌شوند و ممکن است همیشه به ازای انتخاب اکشن A در حالت S به پاداش R یا حالت S' نرسیم (در شرایطی که نتایج اکشن‌ها یا پاداش‌ها قطعی نباشند).

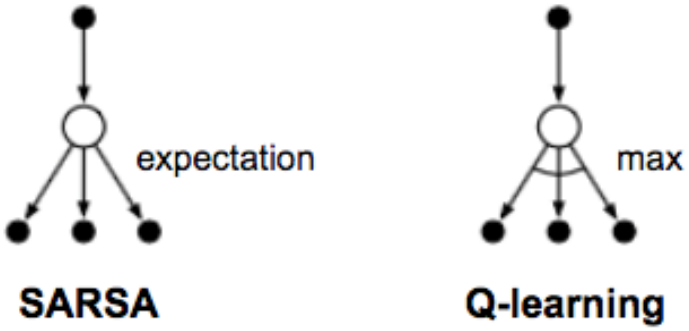
برای پیاده‌سازی الگوریتم SARSA نیاز است که داخل حلقه while در کد داده شده را تکمیل کنید. در نهایت، باید تابع بهینه Q را به دست آورید.

سودوکد آن به این صورت می‌باشد و متغیر های آن مشابه الگوریتم Q-learning هستند:

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

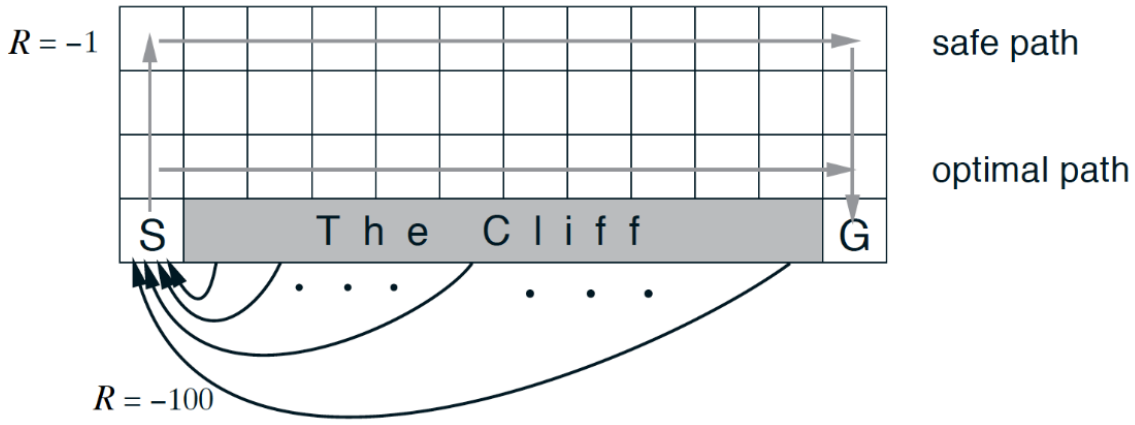
Initialize $Q(s,a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal\text{-}state,\cdot) = 0$
Repeat (for each episode):
  Initialize S
  Choose A from S using policy derived from Q (e.g., ϵ-greedy)
  Repeat (for each step of episode):
    Take action A, observe R, S'
    Choose A' from S' using policy derived from Q (e.g., ϵ-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$
    $S \leftarrow S'$; $A \leftarrow A'$;
  until S is terminal

همچنین در شکل زیر می‌توانید مقایسه ای بین دیاگرام Q-learning و SARSA مشاهده کنید:



تحلیل و بررسی

۱. با تکمیل قسمت های مشخص شده برای هردو الگوریتم، و بررسی تصاویری که خروجی پالیسی بدست آمده از آنها هستند، تفاوت عملکرد الگوریتم Q-learning و SARSA برای این مسئله را بررسی کنید. علت این تفاوت چیست؟



The cliff-walking task (Sutton and Barto, 2017)

۲. تعداد اپیزود ها را به ۲۰۰، ۲۵۰ و ۳۰۰ تغییر دهید و با توجه به عملکرد عامل پس از آموزش توسط الگوریتم ها، میزان مناسب بودن آنها را با هم مقایسه کنید.

نمره اضافه

۱. تعریف متغیری ثابت به نام epsilon_decay که کد به گونه ای باشد که ابتدا ε با بیشترین مقدار ممکن (مقدار ۱) تعریف شود و سپس در طی بازی، پس از هر اپیزود، مقدار ثابت epsilon_decay از آن کم بشود تا به حداقل مقداری که می‌تواند داشته باشد برسد. بدین شکل در ابتدا عامل تمایل به کشف (explore) بیشتری دارد و در گذر زمان این تمایل کاهش پیدا می‌کند و اکشن ها را حریصانه تر انتخاب می‌کند.

۲. آرایه ای به نام training_error تعریف کنید که هر بار هنگام آپدیت ارزش یک اکشن برای یک حالت، مقدار termporal difference error را به خود اضافه می‌کند و سپس با کشیدن نمودار برای این مقادیر (plot کردن آنها با استفاده از کتابخانه matplotlib) بررسی کنید که این ارور به چه مقداری نزدیک تر می‌شود و دلیل آن چیست.

۳. تابعی تعریف کنید که با دریافت q_table (برای الگوریتم Q-learning)، محیط را پلات میکند تا اکشن مناسب برای هر خانه را در داخل آن (با عدد یا رنگ مشخص) مشخص کند.

الزامات پروژه

- فایل ارسالی باید شامل کدهای تکمیل شده به همراه گیف نشان دهنده سیاست هردو الگوریتم و فایل های تولید شده باشد. پاسخ سئوالات بخش تحلیل و بررسی و همچنین هر توضیح مربوط به بخش نمره اضافه را در فرمت پی دی اف یا تکست، همراه با سایر فایل ها قرار دهید.
- حتما در گزارش ارسالی شماره دانشجویی هر دو نفر وجود داشته باشد.
- پاسخ‌های شما باید کامل، واضح و با ارائه استدلال باشد. از ارسال تنها نمودار خروجی هر بخش خودداری کنید.
- از کپی برداری و استفاده از تمرین‌های دانشجویان دیگر به شدت خودداری کنید. در صورت مشاهده شباهت نامتعارف، به هر دو گروه نمره ۱۰۰- داده خواهد شد.

موفق باشید