دانشگاه تهران

پردیس دانشکده‌های فنی

دانشکده برق و کامپیوتر

گزارش تمرین شماره ۴

درس یادگیری تعاملی

پاییز ۱۴۰۱

نام و نام خانوادگی

مهیار ملکی

شماره دانشجویی

۸۱۰۱۰۰۴۷۶

فهرست

چکیده ۴

سوال ۱ – سوالات تحلیلی ۵

سوال ۱ – الگوریتم‌های Sarsa و Expected-Sarsa ۵

سوال ۲ – الگوریتم n-Step Return ۷

سوال ۲ – سوال پیاده‌سازی ۸

سوال ۱ – الگویتم Q-Learning ۸

نتایج ۸

سوال ۲ – حالات غیرقابل دستیابی ۱۰

سوال ۳ – الگوریتم‌های Sarsa و n-Step tree backup ۱۱

سوال ۴ – الگوریتم on-Policy MC ۱۳

سوال ۵ ..... ۱۵

امتیازی ۱۵

نکات پیاده‌سازی ۱۷

روند اجرای کد پیاده‌سازی ۱۷

منابع ۱۸

فهرست شکل‌ها

شکل ۱- مقایسه الگوریتم‌های sarsa و expected-sarsa در تسک cliff-walking به ازای نرخ‌های یادگیری متفاوت.... ۶
شکل ۲- نمودار پاداش الگوریتم Q-learning به ازای نرخ یادگیری ثابت و کاهشی در ۳۰۰ اپیزود ابتدایی ۹
شکل ۳- نمودار حسرت الگوریتم Q-learning به ازای نرخ یادگیری ثابت و کاهشی..... ۹
شکل ۴- نمودار پاداش الگوریتم‌های sarsa و n-step tree backup در ۳۰۰ اپیزود ابتدایی ۱۲
شکل ۵- نمودار حسرت الگوریتم‌های sarsa و n-step tree backup..... ۱۲
شکل ۶-نمودار پاداش الگوریتم on-policy MC به ازای اپسیلون ثابت و کاهشی..... ۱۴
شکل ۷- نمودار حسرت الگوریتم on-policy MC به ازای اپسیلون ثابت و کاهشی..... ۱۴
شکل ۸- نمودار پاداش الگوریتم on-policy MC اصلی و تغییریافته..... ۱۶
شکل ۹-نمودار حسرت الگوریتم on-policy MC اصلی و تغییریافته ۱۶

چکیده

هدف این تمرین آشنایی با الگوریتم‌هایی برای حل مسائل MDP با فرض ناشناخته بودن محیط می‌باشد. از این روش‌ها در ادبیات به عنوان روش‌های بدون مدل یا Model-Free یاد می‌شود. بدین منظور در قسمت اول به بررسی چند مسئله تحلیلی و شناخت بیشتر الگوریتم‌های Model-Free و در قسمت دوم به پیاده‌سازی این الگوریتم‌ها و حل یک مسئله دنیای واقعی با استفاده از رابط gym خواهیم پرداخت.

سوال ۱ – سوالات تحلیلی

سوال ۱ – الگوریتم‌های Sarsa و Expected-Sarsa

۱.

با دقت در هر دوی این الگوریتم‌ها، می‌بینیم که در الگوریتم Expected-Sarsa به دلیل وجود ترم $\sum$، پیچیدگی محاسباتی بیشتری خواهیم داشت. از طرف دیگر در الگوریتم Sarsa مقادیر S’ و A’ به صورت تصادفی انتخاب می‌شود در حالی که در الگوریتم Expected-Sarsa مقدار expected ارزش حالت‌ها به ازای اعمال مختلف محاسبه می‌شود، لذا واریانس تخمین در روش Sarsaی معمولی بیشتر است.

الگوریتم Sarsa :

$$Choose\ A'\ from\ S'\ using\ policy\ derived\ from\ Q$$
$$Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma Q(S',A') - Q(S,A)]$$

الگوریتم Expected-Sarsa :

$$Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \sum_a \pi(a|S_{t+1})Q(S_{t+1},a) - Q(S,A)\right]$$

در نتیجه برای مقایسه این دو الگوریتم یک trade-off بین پیچیدگی محاسبات و sample efficiency وجود دارد. الگوریتم Sarsa کارایی محاسباتی بیشتر و زمان محاسباتی کمتری دارد، در حالی که الگوریتم Expected-Sarsa با کسب تجربه کمتر، عملکرد بهتری را ارائه می‌دهد. بنابراین با کسب تجربه ثابت، الگوریتم Expected-Sarsa با احتمال بیشتری به سیاست $\varepsilon_{optimal}$ همگرا خواهدشد و انتظار می‌رود میزان حسرت کمتری را نیز نتیجه دهد.

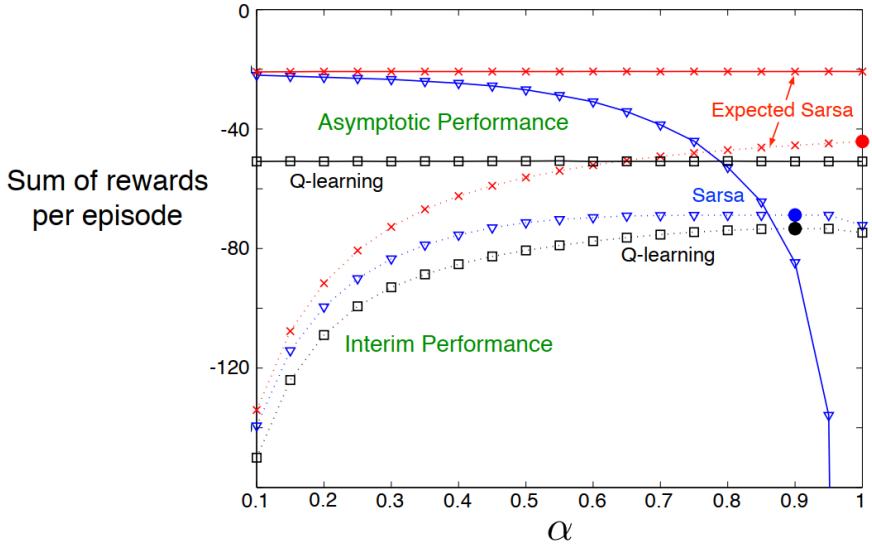
۲.

تفاوت دو سیاست بهینه و $\varepsilon_{optimal}$ در این است که برای رسیدن به سیاست بهینه نیاز به کاهش مقدار اپسیلون داریم در حالی که با ثابت بودن اپسیلون به یک سیاست بهینه $\varepsilon_{optimal}$ برای همان اپسیلون خواهیم‌رسید. لذا انتظار می‌رود در این مورد نیز تفاوت خاصی مشاهده نشود و مجددا با کسب تجربه ثابت، الگوریتم Expected-Sarsa با احتمال بیشتری به سیاست بهینه همگرا شده و میزان حسرت کمتری را نیز نتیجه دهد.

۳.

❖ نرخ یادگیری:

از نمودار موجود در کتاب که برای تسک cliff-walking می‌باشد، استفاده خواهیم کرد.



شکل ۱- مقایسه الگوریتم‌های **sarsa** و **expected-sarsa** در تسک **cliff-walking** به ازای نرخ‌های یادگیری متفاوت

چنان چه مشاهده می‌شود، با اجرای الگوریتم Sarsa در اپیزودهای اولیه، با افزایش نرخ یادگیری، پاداش بیشتری حاصل شده‌است ولی در ادامه پس از ۱۰۰۰۰۰ اپیزود، افزایش نرخ یادگیری منجر به کاهش پاداش شده‌است. این اتفاق بدین دلیل می‌باشد که در الگوریتم Sarsa نرخ‌های یادگیری بزرگتر منجر به Exploration بیشتر می‌شود، در حالی که با پیشروی الگوریتم نیاز داریم تا با کاهش Exploration ارزش اکشن‌ها کمتر دچار تغییر شود. این در حالی است که در الگوریتم Expected-Sarsa برای بروزرسانی ارزش اکشن‌ها به دلیل expected گرفتن، randomnessی وجود ندارد و حتی نرخ یادگیری ۱ نیز در اپیزودهای بالا توانسته‌است به نتیجه برسد.

❖ اپسیلون:

همان طور که می‌دانیم در سیاست ε-greedy کاهش اپسیلون منجر به حریصانه‌تر شدن انتخاب عمل می‌شود. در الگوریتم Expected-Sarsa نیز به دلیل این که sample efficiency بیشتری داریم، می‌توان با تعداد نمونه‌های کمتری به سیاست بهینه همگرا شد. بنابراین در الگوریتم Expected-Sarsa می‌توان مقدار اپسیلون را با نرخ بیشتری نسبت به الگوریتم Sarsa کاهش داد.

سوال ۲ – الگوریتم n-Step Return

در کتاب ساتن داریم:

یک ویژگی مهم الگوریتم‌های n-step return این است که تضمین می‌شود که بدترین خطای تخمین، کمتر یا برابر باشد با γ^n برابر بدترین خطا تحت سیسات V_{t+n-1} :

$$\max_s |\mathbb{E}_\pi[G_{t:t+n}|S_t = s] - v_\pi(s)| \leq \gamma^n \max_s |V_{t+n-1}(s) - v_\pi(s)| \qquad (for\ all\ n \geq 1)$$

(اثبات در https://ai.stackexchange.com/questions/9396/how-do-we-prove-the-n-step-return-error-reduction-property)

در الگوریتم‌های n-step return به این خاصیت error reduction property می‌گویند. به دلیل این خاصیت، می‌توان نشان داد که تمام روش‌های n-step تحت شرایط تکنیکی مناسب، به پیش‌بینی درست همگرا خواهندشد.

سوال ۲ – سوال پیاده‌سازی

سوال ۱ – الگویتم Q-Learning

- فرمول مورد استفاده برای کاهش مقدار اپسیلون:

$$\epsilon = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min}) \times e^{\left(\frac{-episode}{0.1 \times (number\ of\ episodes)}\right)} = 0.9 \times e^{(-0.005 \times episode)}$$

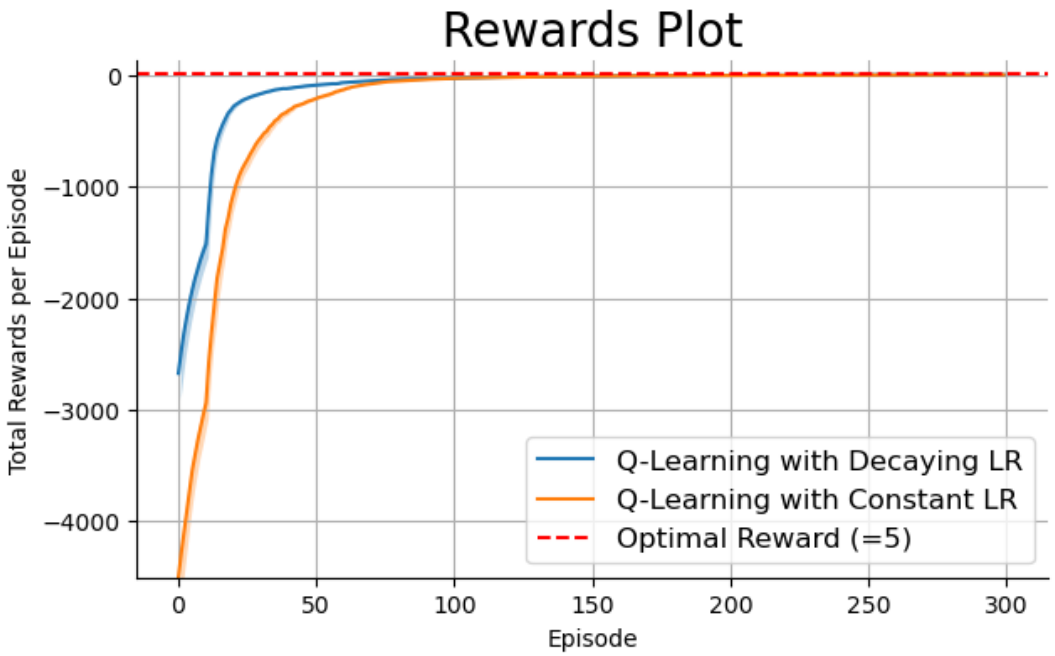
طبق این فرمول مقدار اپسیلون از 0.9 شروع شده و به سمت صف میل می‌کند. همچنین سرعت کاهش آن با توجه به تعداد اپیزودها تایین شده‌است. به عنوان مثال در اینجا که تعداد اپیزودها 2000 می‌باشد، نرخ کاهش برابر با 0.005 است.

لازم به ذکر است که برای کاهش نرخ یادگیری نیز از همین رابطه استفاده شده‌است.

نتایج

مشخص است که در ابتدای آموزش نیاز به exploration بیشتر بوده و به همین علت مقدار آلفا باید بالا باشد تا ارزش حالات مختلف تغییرات بیشتری داشته‌باشند، همچنین با پیشروی الگوریتم و نزدیک شدن به سیاست بهینه، نیاز است که مقدار exploration به تدریج کاهش یابد تا ارزش استیت‌ها و در نتیجه سیاست حاصل از آن‌ها کمتر دچار تغییر شود. لذا انتظار می‌رود تا نرخ یادگیری متغیر و کاهشی عملکرد بهتری را نتیجه دهد.

مطابق انتظار، همان طور که در شکل ۲ نیز قابل مشاهده است، با استفاده از نرخ یادگیری کاهشی، با سرعت بیشتر و در نتیجه در زمان کوتاه‌تری به سیاست بهینه همگرا شده‌ایم. مقدار پاداشی که هر دو الگوریتم به آن همگرا شده‌اند نیز یکسان است. همچنین در شکل ۳ نیز مشخص است که با کاهش نرخ یادگیری، میزان حسرت بسیار کمتری نتیجه شده‌است.



شکل ۲- نمودار پاداش الگوریتم **Q-learning** به ازای نرخ یادگیری ثابت و کاهشی در ۳۰۰ اپیزود ابتدایی

شکل ۳- نمودار حسرت الگوریتم **Q-learning** به ازای نرخ یادگیری ثابت و کاهشی

سوال ۲ – حالات غیرقابل دستیابی

- شناسایی محیط:

محیط این مسئله یک نقشه ۵ در ۵ می‌باشد که ۴ خانه سبز و قرمز و آبی و زرد در آن وجود دارد. تاکسی می‌تواند در هر یک از ۲۵ نقطه محیط قرار بگیرد. مسافر نیز می‌تواند در یکی از خانه‌های رنگی و یا در تاکسی قرار داشته باشد، لذا ۵ حالت را برای آن متصور می‌شویم. همچنین هر یک از این ۴ خانه‌ی رنگی می‌تواند مقصد مسافر باشد. بنابراین مسئله دارای (25*4*5=500) حالت متفاوت می‌باشد.

- حالات غیر قابل دستیابی:

حالات نهایی یا ترمینال ما زمانی است که هم مسافر و هم تاکسی در مقصد قرار داشته باشند، در واقع تاکسی مسافر را در مقصد پیاده کرده‌است. لذا حالاتی که مسافر در مقصد باشد ولی تاکسی آنجا نباشد، مطلوب ما نبوده و غیر قابل دستیابی‌اند.

تعداد حالاتی که مسافر در مقصد است: 1(passenger)*4(destination)*25(taxi) = 100

تعداد حالاتی که مسافر و تاکسی در مقصد اند: 1(passenger)*4(destination)*1(taxi) = 4

تعداد حالات غیر قابل دستیابی: 100 – 4 = 96

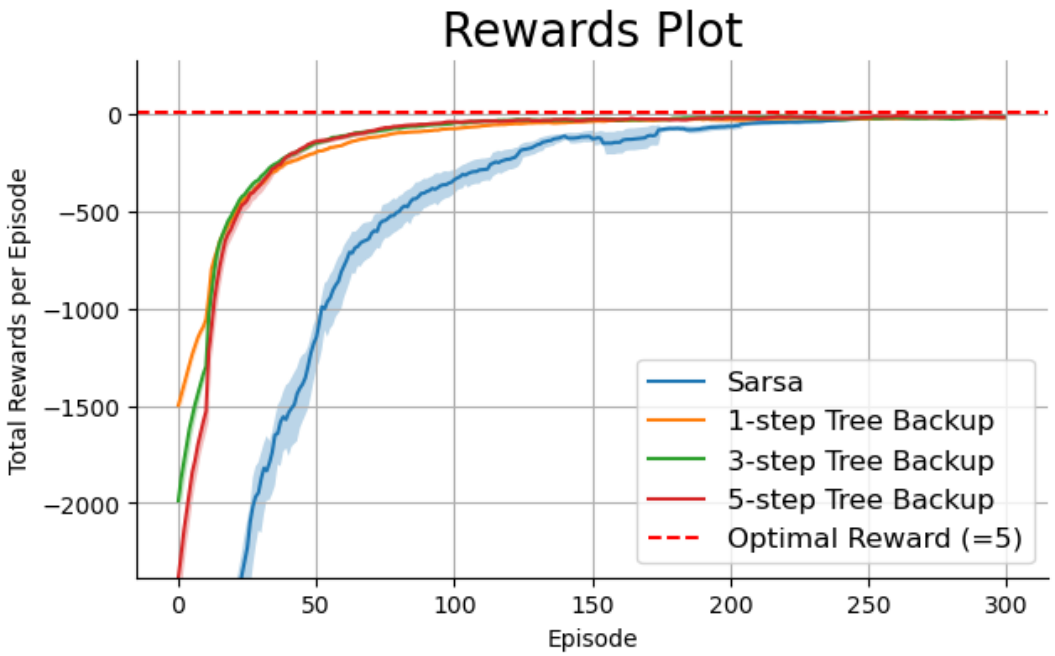
سوال ۳ – الگوریتم‌های Sarsa و n-Step tree backup

با دقت در شکل ۴ مشاهده می‌شود که الگوریتم Sarsa نسبت به الگوریتم‌های n-step tree backup به مراتب کندتر بوده و در زمان بیشتری به پاداش بهینه همگرا شده‌است. در نمودار حسرت شکل ۵ نیز اختلاف زیادی بین الگوریتم Sarsa و n-step قابل مشاهده است.

همچنین در بین الگوریتم‌های n-step با nهای مختلف نیز مشاهده می‌شود که نمودار پاداش هر سه بسیار به هم نزدیک بوده و لذا مقایسه آن‌ها سخت است. البته نکته‌ای که در نمودار پاداش قابل مشاهده است این است که با افزایش n مقدار پاداش اپیزودهای اولیه کاهش یافته است.

با دقت در نمودار حسرت شکل ۵ نیز می‌بینیم که با n برابر با ۵ به بیشترین میزان حسرت بین الگوریتم‌های n-step و با n برابر با ۳ به کمترین میزان حسرت رسیده‌ایم. در واقع با افزایش n به ۵ بهبودی در نتایج دیده نمی‌شود.

در الگوریتم‌های n-step مانند روش Monte Carlo عمل کرده و برای بروزرسانی ارزش هر خانه، با زندگی در محیط از پاداش‌های واقعی که به ازای n مرحله پیش رو از محیط می‌گیریم، استفاده می‌کنیم. به همین علت شاهد عملکرد بهتر این الگوریتم‌ها نسبت به الگوریتم Sarsa بودیم. همچنین اگر مقدار ارزش‌ها فاصله زیادی با مقدار بهینه داشته باشد، افزایش n و زندگی بیشتر در محیط، منجر به افزایش خطا و عملکرد بدتر الگوریتم خواهدشد. به همین علت است که در شکل ۵ در اپیزودهای اولیه با افزایش n شاهد کاهش پاداش دریافتی بودیم.



شکل ۴- نمودار پاداش الگوریتم‌های **sarsa** و **n-step tree backup** در ۳۰۰ اپیزود ابتدایی

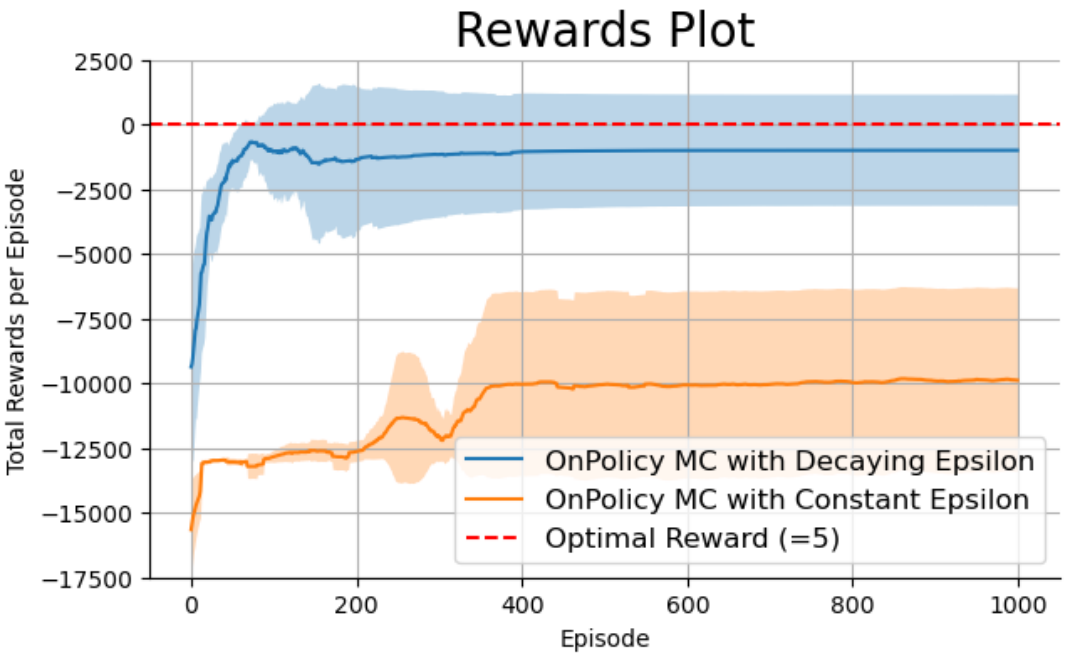
شکل ۵- نمودار حسرت الگوریتم‌های **sarsa** و **n-step tree backup**

سوال ۴ – الگوریتم on-Policy MC

- به دلیل کمبود زمان و منابع، برای کاهش حجم محاسبات تعداد تکرار را به ۱۰ و تعداد اپیزودها را به ۱۰۰۰ کاهش دادیم. همچنین هنگام تولید اپیزود، بیشینه تعداد گام‌های طی شده را برابر با ۱۰۰۰۰ عدد در نظر گرفتیم، زیرا ممکن است با یک سیاست غیر بهینه، حتی با تعداد گام‌های بسیار بالا، به هدف نرسیم و از حلقه while خارج نشویم.

همانطور که در نمودار پاداش شکل ۶ قابل مشاهده است، هر دو الگوریتم بازه اطمینان بزرگی دارند. همچنین مشاهده می‌شود که با اپسیلون ثابت به پاداشی بسیار پایین‌تر از پاداش بهینه همگرا شده‌ایم. (البته لازم به ذکر است که شرط گام‌های محدود در هر بار تولید اپیزود، باعث شده‌است تا عامل به هدف نرسیده و پاداش نهایی را نبیند، لذا اپیزود تولید شده عملکرد خوبی در بروزرسانی ارزش‌ها نخواهد داشت و نرسیدن به پاداش بهینه، امری قابل پیش‌بینی بود) در شکل ۷ نیز مشخص است که میزان حسرت اپسیلون ثابت بسیار بیشتر بوده و بدلیل همگرا شدن به بهینه محلی، پیوسته در حال افزایش است. با آزمودن بهترین عامل آموزش دیده مشاهده شد که در این بهینه محلی با ۲۰ گام به مقصد می‌رسیم و پاداش دریافتی نیز ۱ می‌باشد. زمان اجرای الگوریتم با اپسیلون کاهشی به ازای هر تکرار ۱۳۷ ثانیه و با اپسیلون ثابت (با وجود کاهش اپیزودها و تعداد تکرار) به ازای هر تکرار ۱۲۵۰ ثانیه بوده‌است.

اپسیلون ثابت با مقدار ۰.۱ باعث شده است که جستجو یا exploration در محیط بسیار کم باشد که این امر مطابق انتظار باعث خواهد شد تا زیادی به سیاست فعلی اطمینان داشته و به سیاست بهینه همگرا نشویم. در حالی که اپسیلون کاهشی با گزار تدریجی از exploration به exploitation عملکرد بهتری داشته و به سیاست بهینه همگرا می‌شود.



شکل ۶-نمودار پاداش الگوریتم **on-policy MC** به ازای اپسیلون ثابت و کاهشی

شکل ۷- نمودار حسرت الگوریتم **on-policy MC** به ازای اپسیلون ثابت و کاهشی

سوال ۵

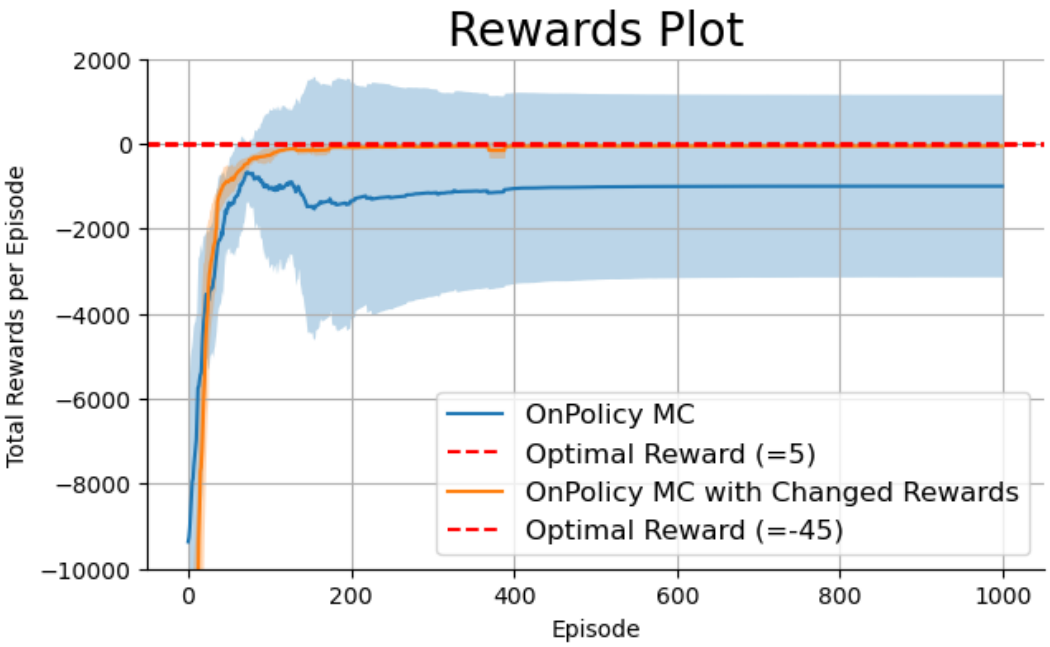
بله، به خصوص با استفاده از مقدار اپسیلون ثابت، مشاهده شد که سرعت آموزش کاهش چشمگیری داشت. علت این امر این است که در این الگوریتم‌ها نیاز است که در هر مرحله، ابتدا با زندگی در محیط، یک اپیزود کامل را تولید کنیم و سپس به بروزرسانی ارزش‌ها و سیاست بپردازیم. لذا در این الگوریتم sample efficiency کمتر و سرعت یادگیری کمتری داریم.

امتیازی

به طور خاص برای الگوریتم MC مشاهده کردیم که استفاده از اپسیلون کاهشی و گزار تدریجی از exploration به exploitation در نتیجه‌ی آن، منجر به افزایش سرعت آموزش می‌شود.

روش دیگری که به نظر می‌رسد موثر باشد، تغییر پاداش‌های محیط است. به عنوان مثال افزایش پاداش هدف یا کاهش پاداش حرکت در محیط می‌تواند موثر واقع شود. می‌دانیم که پاداش هدف ۲۰ و پاداش هر حرکت، منفی ۱ می‌باشد. حال این مقادیر را به ۳۰ و منفی ۵ تغییر می‌دهیم و نتایج را بررسی می‌کنیم.

مشاهده می‌شود که زمان اجرای الگوریتم به طور چشمگیری کاهش پیدا کرده و از ۱۳۷ ثانیه به ۹ ثانیه رسیده‌است. همچنین چنان چه در شکل ۸ نیز مشخص است، گستره بازه اطمینان نیز برای این الگوریتم به شدت کاهش یافته‌است و عدم قطعیت کمتری داریم و در نهایت به پاداش بهینه نیز همگرا شده‌ایم. با تست گرفتن از بهترین پاسخ بدست آمده، مشاهده شد که الگوریتم سیاست بهینه را یافته و همانند قبل در ۱۶ گام به هدف می‌رسد، البته این بار مقدار پاداش سیاست بهینه 45- بدست آمده‌است.



شکل ۸- نمودار پاداش الگوریتم **on-policy MC** اصلی و تغییریافته

شکل ۹-نمودار حسرت الگوریتم **on-policy MC** اصلی و تغییریافته

نکات پیاده‌سازی

- در این تمرین از نسخه جدید کتابخانه gym با نام gymnasium استفاده کرده‌ایم.
- مقدار پاداش بهینه پس از تکرار چندباره الگوریتم‌ها 5+ بدست آمد، که از این مقدار برای محاسبه میزان حسرت هر الگوریتم استفاده شده‌است.
- برای تست الگوریتم و ترسیم سیاست بدست آمده، از بهترین عامل استفاده شده‌است. بهترین عامل را، عاملی در نظر گرفته‌ایم که در آخرین اپیزود به پاداش بیشتری رسیده است.
- فرمول کاهشی هم برای مقادیر اپسیلون و هم برای سرعت آموزش مطابق زیر می‌باشد:

$$0.9 \times e^{(-0.005 \times episode)}$$

روند اجرای کد پیاده‌سازی

تمام کدها و پیاده سازی‌ها در فایل < HW4.ipynb > قرار دارد. تنها لازم است که تمام سلول‌ها به ترتیب از ابتدا اجرا شوند تا نتایج و نمودارها به دست آیند.

منابع

[1] https://jochemsoons.medium.com/a-comparison-between-sarsa-and-expected-sarsa-66b931202c75

[2] https://ai.stackexchange.com/questions/9396/how-do-we-prove-the-n-step-return-error-reduction-property

[3] https://www.analyticsvidhya.com/blog/2018/11/reinforcement-learning-introduction-monte-carlo-learning-openai-gym/