یادگیری در محیط Taxi

حامد محمدزاده

۱) چکیده

در این پروژه قصد داریم بازی Taxi را تغییر داده و در محیط Gymnasium شبیه سازی کرده و نهایتا با الگوریتم‌های MC first visit ، Q-Learning و SARSA برای این بازی سیاستی [۱] بهینه پیدا کنیم.

در بخش اول به تغییرات بازی Taxi و نحوه پیاده سازی این تغییرات می‌پردازیم. سپس در بخش‌های بعدی به ترتیب آزمایشات الگوریتم‌های MC first visit ، Q-Learning و SARSA را بررسی می‌کنیم. در بخش پنجم هم Q-Learning و SARSA را مقایسه می‌کنیم.

۲) تغییرات بازی Taxi

این بازی در حالت عادی شش حرکت پایین، بالا، راست، چپ، برداشتن مسافر و پیاده‌کردن مسافر را دارد. می‌خواهیم ۴ حرکت جابجایی دیگر برای عامل بازی معرفی کنیم: حرکت مورب به پایین-چپ، حرکت مورب به پایین-راست، حرکت مورب به بالا-چپ و حرکت مورب به بالا-راست. قرار داد میکنیم که زمانی می‌شود حرکت مورب x-y را انجام داد که عامل در موقعیت کنونی خود، هم بتواند حرکت x را انجام دهد هم بتواند حرکت y را انجام دهد، در غیر اینصورت عامل با انجام حرکت مورب x-y در جای خود باقی می‌ماند (انگار که عامل به دیوار برخورد کرده است). برای پیاده سازی این تغییرات یک Wrapper برای محیط [۲] می‌نویسیم و تابع ()step که فعل و انفعال محیط و عامل با این تابع صورت می‌گیرد را در این Wrapper بازنویسی می‌کنیم. حرکات شماره شش تا نه را به ترتیب برای حرکات حرکت مورب به پایین-چپ، حرکت مورب به پایین-راست، حرکت مورب به بالا-چپ و حرکت مورب به بالا-راست تغریف

[۱] Policy
[۲] Environment

می‌کنیم. همانطور که گفته شد، به عنوان مثال اگر حرکت هشت (بالا-چپ) انتخاب شود، در Wrapper نوشته شده، با استفاده از action_mask محیط، بررسی می‌شود ایا امکان رفتن به بالا و رفتن به بالا از موقعیت فعلی برقرار است، اگر برقرار بود این دو حرکت با هم انجام می‌شود (دو بار تابع step() صدا زده می‌شود)، در غیر این صورت، حرکتی انجام می‌شود که حالت بازی را تغییر نمی‌دهد. پیاده سازی این Wrapper در فایل utils.py به عنوان کلاس BasicWrapper موجود است.

۳) آزمایشات مربوط به MC first visit

در این آزمایشات، عامل طبق یک سیاست epsilon greedy رفتار می‌کند، همچنین برای میانگین گیری از مقادیر ارزش حالت-حرکت از Moving Average با رابطه زیر استفاده می‌کنیم:

$$Q(S,A) = Q(S,A) + \frac{1}{N(S,A)} \times (G - Q(S,A))$$

که در اینجا $N(S,A)$ تعداد بازدیدهای اول جفت حالت و حرکت مربوطه در آن مرحله از الگوریتم است. [۳]

انتخاب پارامتر مناسب برای آزمایشات MC first visit چالش بر انگیز بود. در ابتدا آزمایشاتی با $\gamma = 0.9$ ، $\epsilon = 0.1$ با هزار اپیزود انجام شد. (در این آزمایشات مقدار اولیه $Q(S,A)$ برای همه حالات و حرکات صفر مقدار دهی شد.

اما نتایج این آزمایشات [۴] همگرا نمی‌شد و بعضی از اپیزودها بسیار طولانی می‌شدند و تا حتی آزمایشات پایان آزمایشات عملی نبود. در حالی که اگر عامل کاملا تصادفی رفتار می‌کرد، به طور معمول طول هر اپیزود حداکثر ده هزار مرحله بود، در آزمایشات ذکر شده طول بعضی از اپیزود ها در مرتبه میلیون بود. طبق بررسی انجام شده، این مشکل بخاطر کمبود کاوش [۵] عامل می‌باشد، با $\epsilon = 0.1$ عامل در اکثر اوقات حریصانه رفتار می‌کند و از طرفی بخاطر واریانس زیاد الگوریتم MC ، طول برخی از اپیزود ها بسیار طولانی می‌شود.

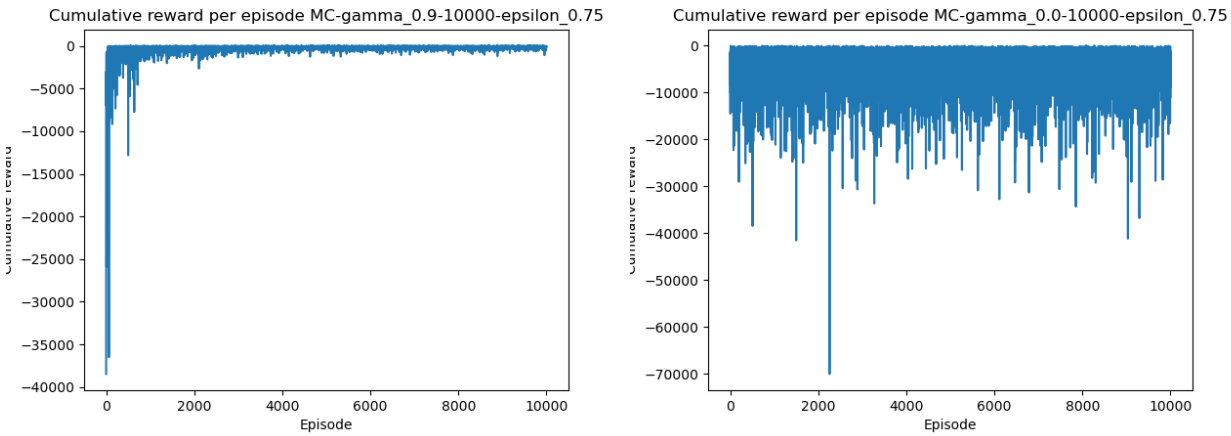
---

[۳] State-Action value
[۴] Agent
[۵] Exploration

فلذا برای رفع این مشکل، دو راهکار افزایش کاوش در آزمایشات انجام شد:

- مقدایر اولیه $Q(S, A)$ برای هر جفت حالت-حرکت، بجای صفر مقدار 0.1 قرار داده شد تا با مقادیر اولیه خوشبینانه [۶] مقدار کاوش عامل بیشتر شود.
- مقدار ϵ در آزمایشات 0.75 قرار داده شد.

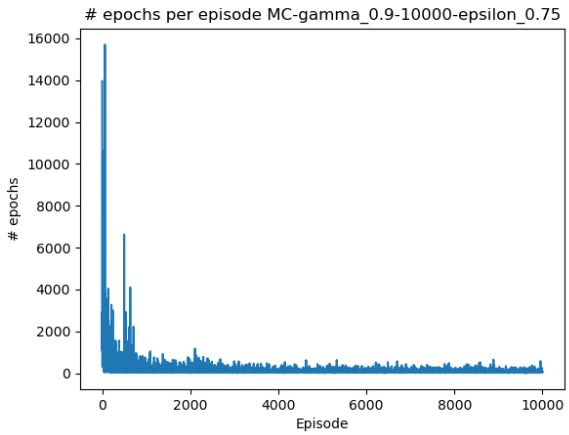
شکل ۱ نمودارهای پاداش تجمعی [۷] الگوریتم MC firt visit را برای مقادیر $\gamma = 0.9$ و $\gamma = 0.0$ نشان می‌دهد. همچنین شکل ۲ نمودارهای تعداد مرحله در هر اپیزود را برای این دو آزمایش نشان می‌دهد.



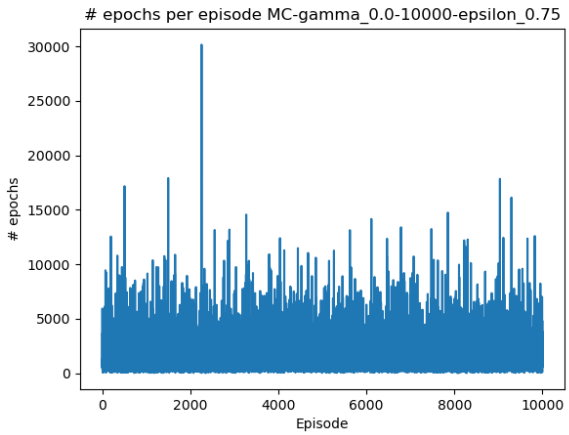
(ب) $\gamma = 0.9$ **(آ)** $\gamma = 0.0$

شکل ۱: محور افقی شماره اپیزود و محور عمودی مقدار پاداش تجمعی است. در هر دو آزمایش تعداد اپیزودها ده هزار، $\epsilon = 0.75$ است.

[۶] Optimisitc initial value
[۷] Cumulative reward



(ب) $\gamma = 0.9$



(آ) $\gamma = 0.0$

شکل ۲: محور افقی شماره اپیزود و محور عمودی تعداد مراحل آن اپیزود است در هر دو آزمایش تعداد اپیزودها ده هزار، $\epsilon = 0.75$ است.

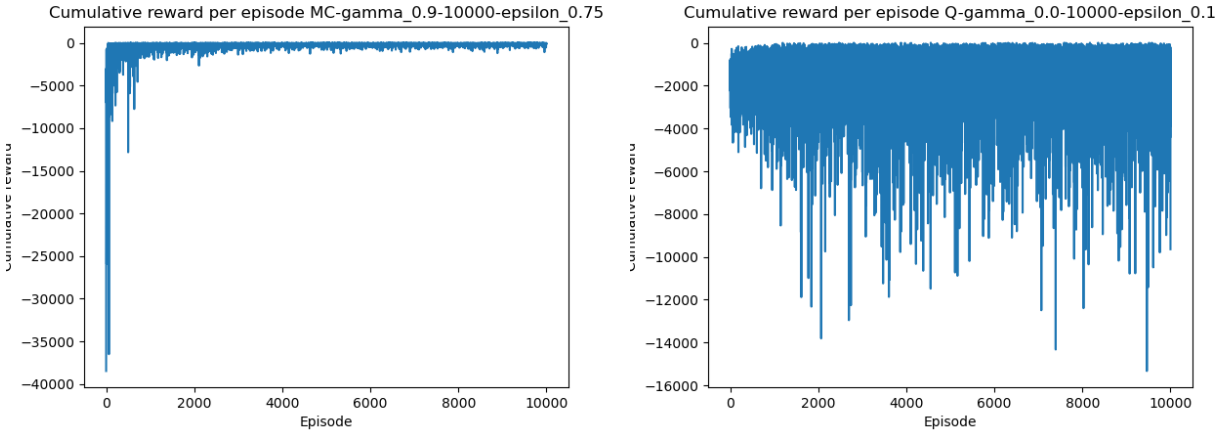
همانطور که می‌دانیم در الگوریم MC مقدار واریانس بالا است، این موضوع در شکل ۱ و ۲ هم دیده می‌شود.

پیاده سازی این الگوریتم یادگیری سیاست در تابع train_mc در کلاس RL_agent می‌باشد.

۴) آزمایشات مربوط به Q-learning

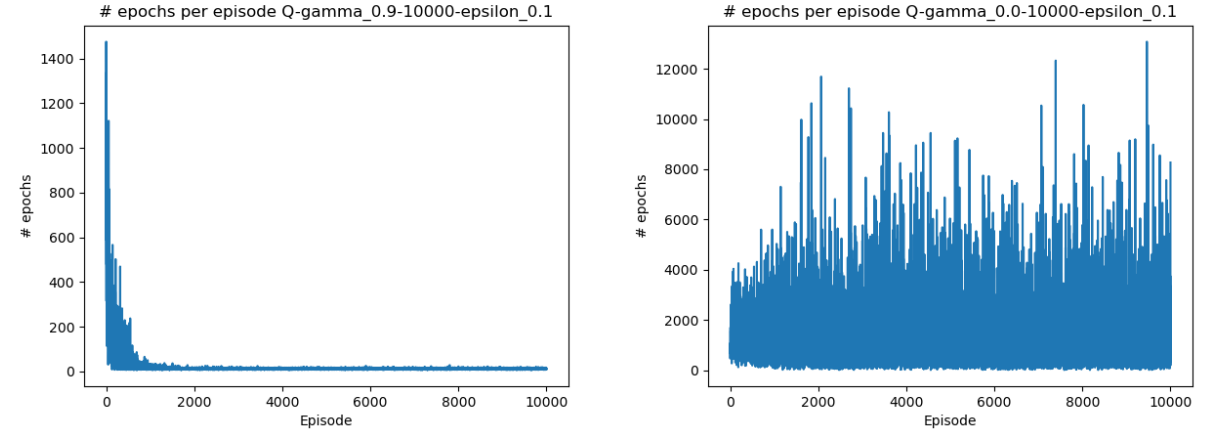
در طی انجام آزمایشات مشاهده شد که سرعت پیدا کردن سیاستی که به عامل کمترین تعداد مرحله در هر اپیزود را می‌دهد، در این الگوریتم از بقیه بیشتر است.

شکل ۳ نمودارهای پاداش تجمعی الگوریتم Q-learning را برای مقادیر $\gamma = 0.9$ و $\gamma = 0.0$ نشان می‌دهد. همچنین شکل ۴ نمودارهای تعداد مرحله در هر اپیزود را برای این دو آزمایش نشان می‌دهد. دقت شود که برای این آزمایشات مقدار $\epsilon = 0.1$ انتخاب شده است و از مقداردهی اولیه خوشبینانه استفاده نشده.



(آ) $\gamma = 0.0$ **(ب)** $\gamma = 0.9$

شکل ۳: محور افقی شماره اپیزود و محور عمودی مقدار پاداش تجمعی است برای آزمایشات Q-learning است.



(آ) $\gamma = 0.0$ **(ب)** $\gamma = 0.9$

شکل ۴: محور افقی شماره اپیزود و محور عمودی تعداد مراحل آن اپیزود است برای آزمایشات Q-learning است.

۵) آزمایشات مربوط به SARSA

این متود پیاد کردن سیاست بهینه طبق الگوریتم شکل ۵ پیاده سازی شده.

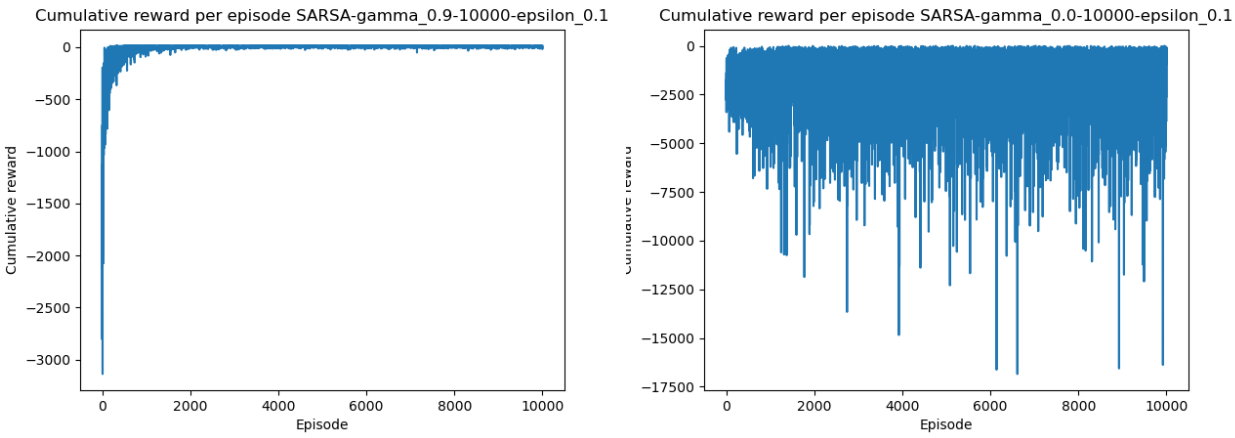
Sarsa (on-policy TD control) for estimating $Q \approx q_*$

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
  Choose A from S using policy derived from Q (e.g., ε-greedy)
  Loop 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

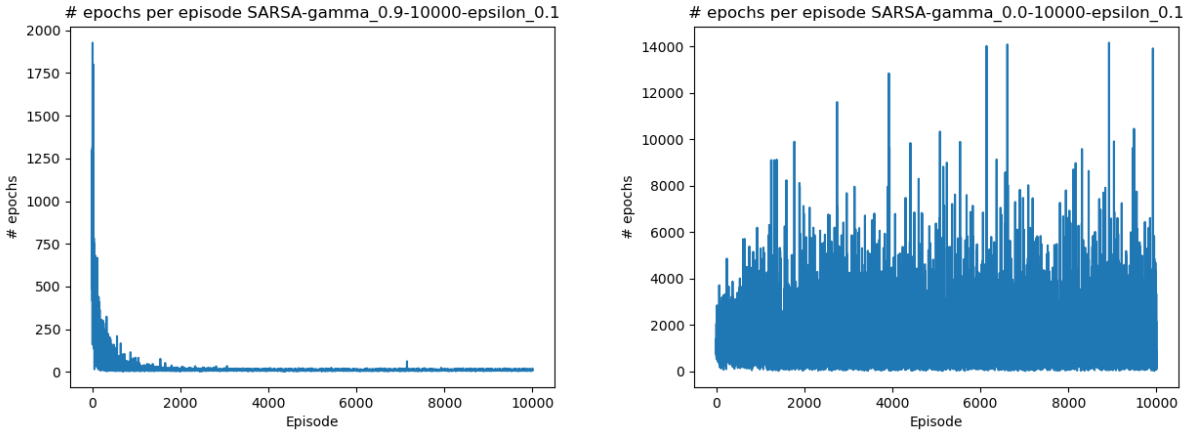
شکل ۵: الگوریتم SARSA

شکل ۶ نمودارهای پاداش تجمعی الگوریتم SARSA را برای مقادیر $\gamma = 0.9$ و $\gamma = 0.0$ نشان می‌دهد. همچنین شکل ۷ نمودارهای تعداد مرحله در هر اپیزود را برای این دو آزمایش نشان می‌دهد. دقت شود که برای این آزمایشات مقدار $\epsilon = 0.1$ انتخاب شده است و از مقداردهی اولیه خوشبینانه استفاده نشده.



$\gamma = 0.9$ **(ب)** $\gamma = 0.0$ **(آ)**

شکل ۶: محور افقی شماره اپیزود و محور عمودی مقدار پاداش تجمعی است برای آزمایشات SARSA است.



$\gamma = 0.9$ **(ب)** $\gamma = 0.0$ **(ا)**

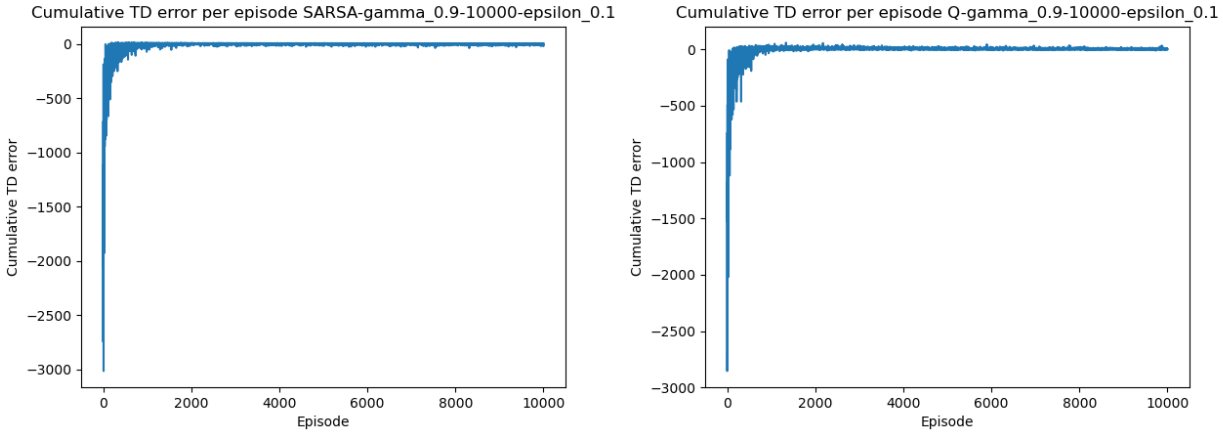
شکل ۷: محور افقی شماره اپیزود و محور عمودی تعداد مراحل آن اپیزود است برای آزمایشات SARSA است.

۶) مقایسه نتایج SARSA و Q-learning

از آنجایی که با $\gamma = 0.0$ هیچکدام از SARSA و Q-learning همگرا نشده اند (شکل‌های ۳.الف و ۶.الف) و عامل چیز مفیدی یادنگرفته، نتایج هرکدام از الگوریتم‌ها را با $\gamma = 0.9$ مقایسه می‌کنیم.

با مقایسه نمودارهای ۳.ب و ۴.ب با نمودارهای ۶.ب و ۷.ب مشاهده می‌شود که Q-learning سریعتر از SARSA مسیرهای با ریوارد بیشتر (که در این مسئله خاص متناظر با مسیرهای کوتاهتر هم می‌باشند) را پیدا کرده.

از طرفی با بررسی مقدار تجمعی TD error برای هرکدام از الگوریتم ها در شکل ۸ ، مشاهده می‌شود که SARSA با سرعت بیشتری مقدار TD error را کاهش می‌دهد.



SARSA (ب) Q-learning (آ)

شکل ۸: مقدار TD error تجمعی در هر اپیزود برای هر دو الگوریتم. دقت شود مقدار گاما برای هردو ۹.۰ است و شرایط دیر هردو یکسان است