گزارش کد حل ماز با استفاده از یادگیری تقویتی(SARSA Max)

معرفی کلی کد

این کد با استفاده از الگوریتم یادگیری تقویتیSARSA Max ، یک ماز را حل می‌کند. هدف این است که یک عامل (Agent) از موقعیت شروع (Start)به موقعیت هدف (Goal) برسد، در حالی که از دیوارها (Walls) اجتناب می‌کند و پاداش (Reward) را به حداکثر می‌رساند. کد شامل چندین بخش است که هر کدام وظیفه خاصی را انجام می‌دهند. در ادامه، اجزای مختلف این کد معرفی و توضیح داده می‌شوند.

اجزای اصلی کد

۱ .کلاسMaze

کلاس Maze ساختار اصلی برنامه است که شامل تمام ویژگی‌ها و توابع لازم برای ساخت، نمایش و حل ماز است. این کلاس شامل متغیرها و توابع زیر است:

متغیرهای اصلی:

- **n**و **m:** اندازه ماز (n × n) و تعداد دیوارها.(m)
- **maze:**ماتریسی که ماز را نمایش می‌دهد.
- **walls:**مجموعه‌ای از موقعیت‌های دیوارها.
- **start**و **goal:** موقعیت شروع و هدف در ماز.
- **qmatrix:**ماتریسی که مقادیر Q برای هر حالت و عمل را ذخیره می‌کند.
- **cumulative_rewards:**آرایه‌ای که پاداش تجمعی برای هر اپیزود را ذخیره می‌کند.

تنظیمات الگوریتم:

- **epsilon:**نرخ اکتشاف (Exploration Rate) که به تدریج کاهش می‌یابد.
- **alpha:**نرخ یادگیری.
- **gamma:**ضریب تخفیف.(Discount Factor)
- **total_episodes:**تعداد کل اپیزودها.
- **action:**لیستی از چهار حرکت ممکن (بالا، پایین، چپ و راست).

۲ .تابعcreate_maze

این تابع ماز را ایجاد می‌کند. ابتدا موقعیت‌های دیوارها به صورت تصادفی تعیین می‌شوند. سپس موقعیت‌های شروع و هدف مشخص می‌شوند، به طوری که موقعیت شروع و هدف روی دیوارها قرار نگیرند.

۳ .تابعreward_func

این تابع مقدار پاداش برای یک حالت خاص را تعیین می‌کند:

- **هدف :(Goal)** پاداش مثبت.(10+)
- **دیوارها :**پاداش منفی.(10-)
- **سایر حالات :**پاداش منفی کوچک.(1-)

۴ .تابعsarsa_max

این تابع الگوریتم اصلی یادگیری تقویتی SARSA Max را پیاده‌سازی می‌کند. در هر اپیزود:

1. عامل از موقعیت شروع حرکت می‌کند.
2. برای هر حالت، با احتمال ε یک عمل تصادفی (Exploration) یا بهترین عمل (Exploitation) را انتخاب می‌کند.
3. -Qماتریس به‌روزرسانی می‌شود.
4. پاداش تجمعی برای اپیزود محاسبه و ذخیره می‌شود.

۵ .تابعfetch_policy

این تابع سیاست استخراج‌شده (Optimal Policy) را نمایش می‌دهد. برای هر حالت، بهترین عمل با استفاده از Q- ماتریس مشخص می‌شود و در ماتریس ماز نمایش داده می‌شود.

۶ .تابعtrace_solution

این تابع مسیر حل‌شده را از موقعیت شروع تا هدف با استفاده از سیاست استخراج‌شده ردیابی می‌کند. در نهایت، مسیر به صورت گرافیکی در ماز نمایش داده می‌شود.

۷ .تابعplot_cumulative_rewards

این تابع نمودار پاداش تجمعی را بر اساس تعداد اپیزودها رسم می‌کند. این نمودار نشان‌دهنده پیشرفت عامل در یادگیری است.

8. تابع **animate_solution**

این تابع وظیفه انیمیشن کردن مسیر عامل را دارد.

9. تابع **plot_q_heatmaps**

این تابع نمودار حرارتی برای مقادیر کیو را رسم می کند.

روش کار و توضیح رویکرد

۱ .طراحی ساختار ماز

ابتدا ساختار ماز با استفاده از ماتریس و مجموعه‌ای از دیوارها طراحی شد. موقعیت‌های شروع و هدف به گونه‌ای انتخاب شدند که مسیر حل‌شدنی (Solvable)تضمین شود.

۲ .پیاده‌سازی الگوریتمSARSA Max

الگوریتم SARSA Max به عنوان یک روش یادگیری تقویتی انتخاب شد. این الگوریتم به دلیل توانایی در مدیریت محیط‌های قطعی و غیرقطعی مناسب است. نرخ اکتشاف (ε) به تدریج کاهش می‌یابد تا عامل از حالت اکتشاف به بهره‌برداری تغییر کند.

۳ .تحلیل نتایج

برای ارزیابی عملکرد عامل، از پاداش تجمعی استفاده شد. این معیار نشان می‌دهد که عامل چگونه در طول زمان بهینه‌تر عمل می‌کند. همچنین، مسیر حل‌شده و سیاست بهینه به صورت گرافیکی نمایش داده می‌شوند.

۴ .هایپرپارامتر های این مسیله

1. **نرخ یادگیری :(Alpha)** مقدار 0.1برای نرخ یادگیری انتخاب شده است. این مقدار کوچک باعث می‌شود که به‌روزرسانی‌های ماتریس Q به‌صورت تدریجی انجام شود و از نوسانات بزرگ در یادگیری جلوگیری شود. کاهش نرخ یادگیری به پایداری فرآیند یادگیری کمک می‌کند.
2. **نرخ کاهش تخفیف :(Gamma)** مقدار 0.99برای گاما انتخاب شده است که نشان‌دهنده تأکید زیاد بر پاداش‌های بلندمدت است. این مقدار باعث می‌شود که عامل (Agent) مسیرهایی را که در نهایت به هدف می‌رسند، ترجیح دهد.

3. **نرخ اکتشاف (Epsilon)**: مقدار اولیه 1.0به معنای اکتشاف کامل در ابتدای فرآیند است، و این مقدار به تدریج با نرخ کاهش 0.995کم می‌شود تا به حداقل مقدار 0.01برسد. این تنظیم باعث می‌شود که عامل در ابتدا محیط را بیشتر کشف کند و با گذشت زمان به بهره‌برداری از دانش به‌دست‌آمده بپردازد.

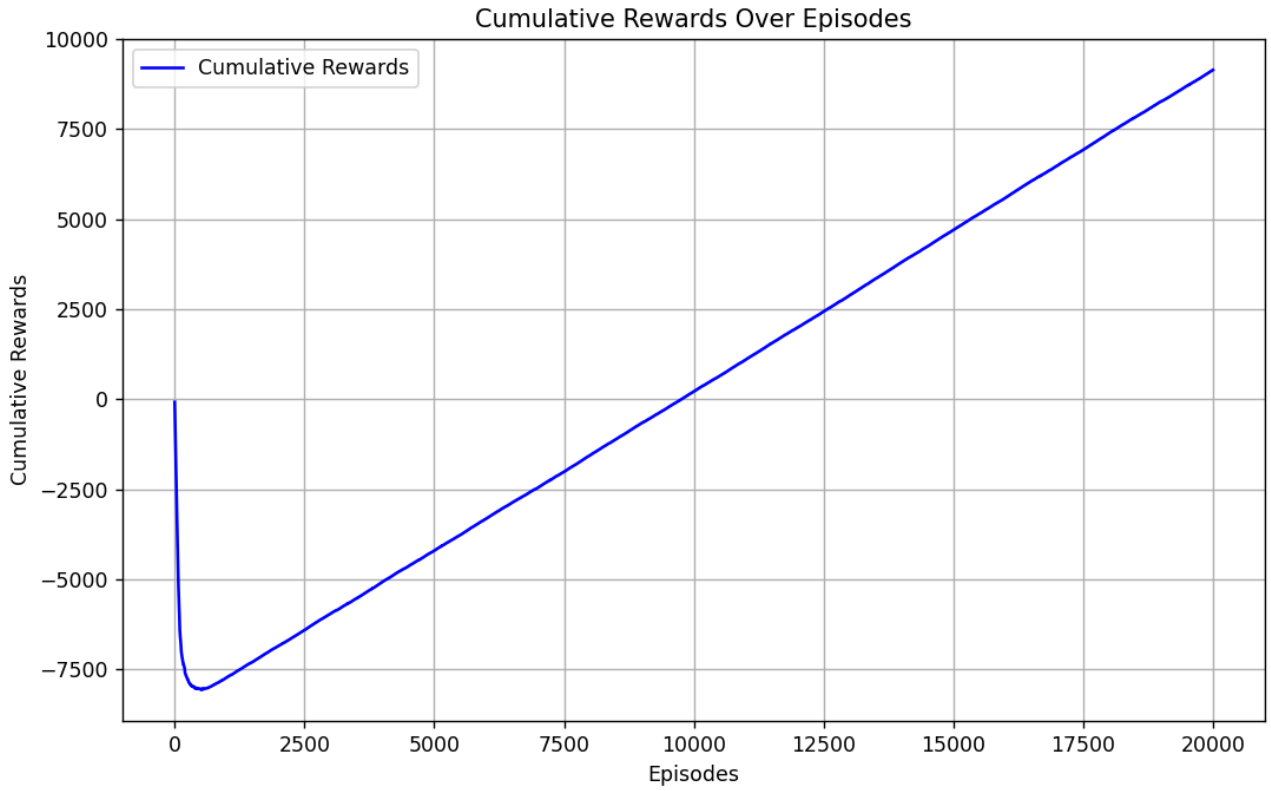
4. **تعداد اپیزودها (Total Episodes)**: تعداد 20000اپیزود برای یادگیری تعیین شده است. این تعداد زیاد تضمین می‌کند که عامل فرصت کافی برای کشف تمام مسیرهای ممکن و یادگیری سیاست بهینه را داشته باشد.

تأثیر هایپرپارامترها :

- **نرخ یادگیری** کوچک باعث یادگیری تدریجی و جلوگیری از نوسانات می‌شود، اما اگر خیلی کوچک باشد، فرآیند یادگیری کند خواهد شد.
- **گاما** بالا باعث می‌شود عامل به پاداش‌های بلندمدت توجه بیشتری داشته باشد، اما اگر خیلی بزرگ باشد، ممکن است از پاداش‌های کوتاه‌مدت چشم‌پوشی کند.
- **اپسیلون** به تدریج عامل را از حالت اکتشاف به بهره‌برداری هدایت می‌کند. اگر نرخ کاهش اپسیلون خیلی سریع باشد، عامل ممکن است قبل از کشف تمام محیط، وارد حالت بهره‌برداری شود.
- **تعداد اپیزودها** کافی باعث می‌شود که عامل فرصت کافی برای یادگیری داشته باشد. اگر تعداد اپیزودها کم باشد، یادگیری ناقص خواهد بود.

این تنظیمات به‌صورت متعادل انتخاب شده‌اند تا یادگیری پایدار و کارآمدی حاصل شود.

- **نمودار پاداش تجمعی به ازای هر اپیزود در اجرای یک نمونه رندوم :**



توضیحات نمودار :

این نمودار نشان می‌دهد که عامل در ابتدا عملکرد ضعیفی دارد اما با گذشت زمان و تکرار اپیزودها، یادگیری انجام می‌شود و عملکرد به طور قابل توجهی بهبود می‌یابد.

شیب مثبت در بخش نهایی نمودار بیانگر این است که هایپرپارامترها به درستی تنظیم شده‌اند و عامل توانسته است سیاستی بهینه برای حل مسئله پیدا کند.

پیاده سازی بخش های امتیازی :

- آزمایش با mazeهای بزرگتر یا نامنظم برای افزایش پیچیدگی.

با تغییر مقادیر n و m میتوان ابعاد ماز و تعداد دیوار ها را تغییر داد.

برای راحتی کار در تابع main دو متغیر maze_dim و wall_count در نظر گرفته شده است.

- کد خود را به وضوح مستند کنید و توضیحاتی برای شرح مراحل کلیدی اضافه کنید.

تا جایی که توانستم مستند کردم. 😊

- تجسم‌سازی‌ها را شامل کنید، مانند نقشه‌های حرارتی مقادیر Q یا انیمیشن‌های گام به گام مسیر عامل.

هم نقشه حرارتی و هم انیمیشن سازی برای مسیر عامل را با استفاده از این دو تابع انجام دادم : هر دو جز کلاس Maze می باشند.

1. **animate_solution**
2. **plot_q_heatmaps**