فرهاد امان ۹۹۳۱۰۰۶

۱-

• در روش ارزش‌گذاری (Value Iteration)، ما به صورت تکراری مقادیر را به‌روزرسانی می‌کنیم. این موضوع با افزایش تعداد حالت‌ها و عمل‌ها، منجر به استفاده بیشتر از حافظه و رم می‌شود و پیچیدگی محاسبات را افزایش می‌دهد.
• در این روش، ما افزایش نمایی در مقدار تخفیف داریم. می‌دانیم که با افزایش ابعاد مسئله، مجبور به استفاده از منابع بیشتری هم می‌شویم. بنابراین، افزایش تعداد حالت‌های مسئله، کارایی الگوریتم را کاهش می‌دهد.
• در این روش، در هر تکرار، به ازای هر جفت حالت، کل فضای حالت را بررسی می‌کنیم. این به این معنی است که نمی‌توانیم این روش را به مسائلی با فضای حالت ناشناخته تعمیم دهیم.

۲-

در این روش، با استفاده از روش سعی و خطا، تاثیر تغییرات در مقادیر تخفیف و نویز را در رفتار برنامه بررسی می‌کنیم. مشاهده می‌شود که با کاهش هر دو مقدار، اعداد درون مربع‌ها مثبت‌تر می‌شوند و رنگ آنها به سمت سبز روشن تغییر می‌کند. همچنین، مشاهده می‌شود که با کاهش مقدار نویز، بهبود بیشتری را مشاهده می‌کنیم. مقدار نویز را به سمت صفر نزدیک می‌کنیم و مقدار تخفیف را همان ۰.۹ نگه می‌داریم. مقدار نویز را برابر با ۰.۰۰۱ قرار می‌دهیم.
تخفیف، یک مقدار عمری را برای عامل تعیین می‌کند، که به عامل‌ها اجازه می‌دهد تا تا زمانی که حذف می‌شوند، تعدادی قدم بردارند و جوایز را جمع کنند. پیاده‌سازی این تخفیف به صورت امتیازهایی است که مقدار آن‌ها به صورت نمایی افزایش می‌یابد. نکته مهم درباره این تخفیف این است که مقدار اولیه آن بین ۰ و ۱ است و پس از گذشت یک تعداد مشخص قدم، به صفر میل می‌کند.
یکی از روش‌های جایگزین برای value iteration، روش Q-learning است که به طور مستقیم روی تابع سیاست عمل می‌کند. در این الگوریتم، یک جدول Q برای ذخیره ارزش اقدامات در حالت‌های مختلف مورد استفاده قرار می‌گیرد. پس از هر قدم اقدام، عامل یک مجموعه از اقدامات را برای انتخاب دارد و پس از انتخاب یکی از آن‌ها، به حالت

جدید می‌رود و جایزه دریافت می‌کند. سپس، مقدار Q برای یک جفت حالت و عمل بروزرسانی می‌شود.

-۳

چون خروجی نزدیک به صفر است، مقدار جایزه را منفی در نظر می‌گیریم. در اینجا همچنین مقدار تخفیف را کمتر می‌کنیم.
در این حالت، نویز را به صفر می‌رسانیم و مقدار تخفیف را به یک نزدیک می‌کنیم تا به سمت حالت پایدار نزدیک شویم.
ما پاداش را برابر با صفر قرار می‌دهیم و مقدار تخفیف را به یک نزدیک می‌کنیم.
جایزه را منفی و کوچک می‌گیریم و مقدار تخفیف را نیز به صفر نزدیک می‌کنیم.
برای حل این مشکل، می‌توان از راهکارهای زیر استفاده کرد:
- افزودن محدودیت‌ها: با اضافه کردن برخی محدودیت‌ها، می‌توان از گیر کردن در حلقه بی‌نهایت جلوگیری کرد.
- شرط پایان: می‌توان شرایط پایانی را اضافه کرد تا وقتی که به حالت‌های خاصی برسیم، سیاست به پایان برسد.
- بهبود مقادیر: با تجربه و تنظیم مقادیر مختلف، می‌توان مقدار تخفیف و جایزه را بهبود بخشید تا به نتایج مطلوب برسیم.
زمانی که مقدار تخفیف نسبت به نویز و جایزه مناسب نباشد و نتیجه خوبی را تضمین نکند، به حلقه بی‌نهایت برخورد خواهیم کرد و همگرایی را نتیجه نخواهد داد.

-۴

روش دسته‌ای (batch) در عملیات بروزرسانی Q، با انجام محاسبات یکباره بر روی تمام عناصر، بهبود عملکرد و کارایی را تسریع می‌دهد. در این روش، تمام داده‌ها در حافظه ذخیره شده و همزمان پردازش می‌شوند. این به معنای این است که ما نیازی نداریم به صورت تک به تک داده‌ها را بررسی کنیم، بلکه می‌توانیم از قدرت موازی سیستم خود بهره ببریم و عملیات بروزرسانی را به صورت همزمان بر روی تمام داده‌ها انجام دهیم.
استفاده از روش دسته‌ای باعث می‌شود که محاسبات بسیار سریعتر انجام شوند و میزان زمان مورد نیاز برای آپدیت Q کاهش یابد. این بهبود در عملکرد می‌تواند به دلیل استفاده بهینه از منابع سخت افزاری و بهره‌برداری از قدرت پردازش موازی رخ دهد.

هرچند، باید توجه داشت که استفاده از روش دسته‌ای نیز ممکن است با مشکلاتی همراه باشد. به عنوان مثال، زمانی که تعداد بزرگی از داده‌ها و عناصر Q در دسترس است، انجام عملیات بروزرسانی همزمان بر روی همه این داده‌ها ممکن است زمان زیادی را به طول انجامد. بنابراین، برای استفاده بهینه از روش دسته‌ای، ممکن است نیاز به بهبود الگوریتم و بهینه‌سازی مناسب داشته باشیم.
به طور خلاصه، روش دسته‌ای با تمامیت عملیات بروزرسانی و بهره‌برداری از پردازش موازی، می‌تواند بهبود قابل توجهی در کارایی و عملکرد الگوریتم‌ها و محاسبات ماشینی به همراه داشته باشد.

۶-
وقتی که مقدار Q برای اقداماتی که عامل قبلاً تجربه نکرده، از ارزش‌های موجود بسیار دور یا نزدیک باشد، ممکن است سیاستی که عامل بر اساس آن اقدام می‌کند، به طور ناخواسته اشتباه باشد. به عبارت دیگر، ممکن است عامل بر اساس ارزش‌های نادرستی که به آن دست یافته، تصمیم بگیرد و عملی را در یک حالت خاص انجام دهد، در حالی که اگر ارزش‌ها به درستی بروزرسانی شده بودند، تصمیمی متفاوت می‌گرفت.
روش Q-learning یک روش off-policy و مبتنی بر ارزش (value-based) است. این روش از رویکرد off-policy استفاده می‌کند، به این معنی که سیاستی که در حین یادگیری بروز می‌دهد، مستقیماً متکی بر مشاهدات جاری نیست و می‌تواند بر اساس سیاستی دیگر که بهترین ارزش‌ها را دارد، اقدام کند. همچنین، این روش مبتنی بر ارزش است، به این معنی که تلاش می‌کند تا تابع Q را برای هر حالت و عمل بروزرسانی کند.
به عنوان روش‌های دیگری از یادگیری تقویتی، می‌توان به TD-learning و Montecarlo اشاره کرد. در روش TD-learning، بروزرسانی ارزش‌ها براساس هر گام از فرایند یادگیری صورت می‌گیرد و قادر است از اپیزودهای ناکامل نیز یاد بگیرد و در هر گام اقدام به بروزرسانی کند. اما در روش Monte Carlo، بروزرسانی ارزش‌ها پس از پیمایش تمام حالات و پایان هر اپیزود انجام می‌شود. این روش نیازمند استفاده از تمام اپیزودها است تا بتواند ارزش‌های صحیح را محاسبه کند.

۷-
یکی از مزایای استفاده از روش Q-learning و رویکرد off-policy این است که عامل در حین فرایند یادگیری می‌تواند به صورت تصادفی اقدامات جدید را بررسی کند و در عین

حال از اقداماتی که در حال حاضر بهترین عملکرد را دارند، بهره‌برداری کند. این به عامل این اختیار را می‌دهد که به صورت بیشتری از جستجوی عملکردهای جدید و اکتشاف محیط استفاده کند، در حالی که همزمان از دانش و تجربه‌ای که تاکنون به دست آورده، بهره‌برداری کند.
یکی از عوامل مهم در روش Q-learning، مقدار اپسیلون است. اپسیلون یک عدد بین 0 و 1 است که مشخص می‌کند که عامل در چه میزان از جستجوی تصادفی و اکتشاف استفاده کند و در چه میزان به بهره‌برداری از عملکردهای بهتر تمایل داشته باشد. اگر مقدار اپسیلون بزرگ باشد، احتمال بررسی اقدامات تصادفی بیشتر می‌شود و عامل بیشتر از جستجوی عملکردهای جدید و کاوش محیط استفاده می‌کند. اما اگر مقدار اپسیلون کوچک باشد، احتمال بهره‌برداری از عملکردهای بهتر بالاتر می‌رود و عامل بیشتر به تلاش برای بهینه‌سازی عملکرد خود می‌پردازد.
با تنظیم مناسب مقدار اپسیلون، عامل می‌تواند به صورت تعادلی بین اکتشاف و بهره‌برداری عمل کند. در ابتدای فرایند یادگیری، ممکن است مقدار اپسیلون بزرگتر باشد تا عامل بیشتر از اکتشاف محیط استفاده کند و تجربه‌های جدیدی کسب کند. با گذشت ز
مان و بهبود عملکرد عامل، معمولاً مقدار اپسیلون به تدریج کاهش می‌یابد تا عامل بیشتر به بهره‌برداری از عملکردهای بهتر و افزایش بهره‌وری مشتاق باشد.
استفاده از مقدار مناسب اپسیلون در Q-learning می‌تواند به بهبود عملکرد عامل کمک کند. اما باید توجه داشت که این تنظیم پارامتر به شدت وابسته به محیط و وظیفه‌ای است که عامل با آن سروکار دارد. لذا، نیاز است تا آزمایش‌های متعددی با مقادیر مختلف اپسیلون انجام شود تا مقدار بهینه آن برای هر وظیفه مشخص شود.

-۸
مقدار اپسیلون در Q-learning نقش بسیار مهمی در تعادل بین اکتشاف و بهره‌برداری دارد. با تنظیم اپسیلون به نزدیکی از صفر، عامل تلاش می‌کند به بیشترین حد ممکن از عملکردهای بهتر و قبلی خود بهره‌برداری کند. در این حالت، احتمال انتخاب عملی که در حال حاضر بهترین عملکرد را دارد، بسیار بالاست. به عبارت دیگر، عامل تمایل دارد به عملکردهایی که تاکنون به خوبی عمل کرده‌اند و در بهبود خود موثر بوده‌اند، وفادار بماند.
با نزدیک به یک بودن اپسیلون، عامل تلاش می‌کند به صورت فعال اقدامات جدید را کشف کند و از عملکردهای جدید بهره‌برداری کند. در این حالت، احتمال انتخاب عمل تصادفی و

بدون در نظر گرفتن عملکرد قبلی بسیار بالاست. عامل تمایل دارد تا بیشترین اطلاعات ممکن را از فضای عمل جمع‌آوری کرده و عملکرد بهتری را کشف کند.
با توجه به این تفاوت‌ها، تنظیم مقدار اپسیلون در Q-learning بسیار مهم است. این تنظیم باید بر اساس ویژگی‌های وظیفه و محیط، هدف عامل و نیاز به اکتشاف یا بهره‌برداری بیشتر، انجام شود. برای دستیابی به بهترین عملکرد، نیاز است تا مقدار اپسیلون به صورت تجربی و با آزمایش‌های متعدد تنظیم شود تا تعادل مطلوبی بین اکتشاف و بهره‌برداری حاصل شود.

۹-
در این بخش، عملکرد عامل در تمامی حالات مورد نیاز به خوبی انجام شد و نیازی به تغییر کد نبود. از این که خروجی کد در بالا قرار گرفته است، مشخص است که عامل پس از ۲۰۰۰ اپیزود آموزش داخل smallGrid به عملکرد مطلوبی دست یافت و با میانگین امتیاز بالایی بازی‌ها را برنده شد.
این نتایج نشان می‌دهد که عامل با گذشت زمان و تجربه بیشتر، بهبود قابل توجهی در عملکرد خود داشته است. با ادامه آموزش و اکتشاف در فضای عمل، عامل قادر به انجام اقدامات بهینه‌تر و کسب امتیاز بیشتر می‌شود.
این نتایج حاکی از این است که روش یادگیری Q-learning که مبتنی بر ارزش (value-based) و بدون مدل (model-free) است، باعث بهبود عملکرد عامل در تعامل با محیط می‌شود. توانایی عامل در تصمیم‌گیری بهتر و انتخاب اقدامات بهینه، در نتیجه‌ای که بهره‌برداری از عملکردهای بهتر و کشف عملکردهای جدید به ترتیب با اهتمام انجام می‌شود، بهبود می‌یابد.
بر اساس این داده‌ها و نتایج، می‌توان نتیجه گرفت که روش Q-learning با استفاده از الگوریتم بروزرسانی TD-learning و با تنظیم درست مقدار اپسیلون، عملکرد عامل را در محیط‌های پیچیده و تعاملی بهبود می‌بخشد. این روش به عامل این اختیار را می‌دهد تا به صورت همزمان اقدامات جدید را کشف کند و از عملکردهای بهتر استفاده کند، همچنین به اندازه کافی اکتشاف و کشف در محیط انجام دهد تا به بهینگی بیشتری دست یابد.
با توجه به توانایی عامل در بهبود عملکرد خود و برنده شدن در بازی‌ها، می‌توان نتیجه گرفت که این روش یادگیری مناسبی برای تقویت تصمیم‌گیری عامل در محیط‌های پویا و پیچیده است. این موضوع نشان می‌دهد که با استفاده از Q-learning و تنظیم مناسب

پارامترها، می‌توان عامل را آماده بهترین عملکرد در برابر تغییرات محیطی کرد و به بهره‌برداری از عملکردهای بهتر دست یافت.

-۱۰

Deep Q-Learning (DQL) یک الگوریتم تقریبی از Q-Learning است که برای حل مسائلی با فضای عمل وضعیت بزرگ‌تر استفاده می‌شود. DQL از شبکه‌های عصبی عمیق به عنوان تقریب‌گر تابع Q استفاده می‌کند و به وسیله تقریب زدن تابع Q از فضای وضعیت-عمل بزرگ‌تر بهره می‌برد.

استفاده از DQL به جای Q-Learning سنتی در موارد زیر می‌تواند مناسب باشد:

1. مسائل با فضای وضعیت-عمل بزرگ: Q-Learning سنتی در مسائل با فضای وضعیت-عمل بزرگ‌تر می‌تواند با مشکل معروف به "curse of dimensionality" مواجه شود. به این معنی که زمانی که تعداد وضعیت‌ها و عمل‌ها بسیار زیاد است، جدول Q به صورت کامل نمی‌تواند نگهداری شود. در چنین مواردی، استفاده از تقریب های تابع Q مانند شبکه‌های عصبی عمیق در DQL می‌تواند کمک کننده باشد.
2. حالت‌های پیوسته: اگر مسئله شما حالت‌های پیوسته را شامل می‌شود، مانند حرکت ربات در یک محیط پیوسته، استفاده از Q-Learning سنتی که با جدول Q کار می‌کند دشوار خواهد بود. در چنین مواردی، تقریب گر تابع Q در DQL می‌تواند از مشکل تعامل با حالت‌های پیوسته خلاصه شود.

دو الگوریتم DQL و Approximate Q-Learning هر دو مشکلاتی را که در Q-Learning وجود دارد حل می‌کنند. این مشکلات عبارتند از:

1. Curse of Dimensionality: این مشکل در Q-Learning به وجود می‌آید زمانی که فضای وضعیت-عمل بسیار بزرگ است و جدول Q کامل نمی‌تواند نگهداری شود. در DQL و
Approximate Q-Learning، از تقریب های تابع Q استفاده می‌شود که می‌توانند فضای بزرگ‌تری را پوشش دهند.
2. حالت‌های پیوسته: Q-Learning سنتی با استفاده از جدول Q، تنها برای مسائل با حالت‌های گسسته قابل استفاده است. Approximate Q-Learning و DQL با استفاده از تقریب های تابع Q می‌توانند با مسائلی که حالت‌های پیوسته دارند نیز سازگاری داشته باشند.

به طور کلی، استفاده از DQL و Approximate Q-Learning برای مسائلی که فضای وضعیت-عمل بزرگی دارند و یا حالت‌های پیوسته را شامل می‌شوند، مناسب است. با این روش‌ها می‌توان تقریبی از تابع Q را یادگیری کرده و برای تصمیم‌گیری بهینه در مسائل پیچیده استفاده نمود.