به نام خدا

دانشگاه تهران
دانشکده ی مهندسی برق و کامپیوتر

Intelligent Systems

Assignment 4

علیرضا محمدی	نام و نام خانوادگی
۸۱۰۱۹۵۴۷۱	شماره دانشجویی
۱۲/۱۰/۱۳۹۸	تاریخ ارسال گزارش

۱ سوالات

سوال اول پاداش در الگوریتم یادگیری تقویتی در واقع یک مقدار عددی است که عامل از محیط به عنوان یک پاسخ مستقیم به اکشنی که انتخاب کرده است، دریافت می‌کند. هدف عامل این است که پاداش کلی که در یک اپیزود دریافت می‌کند را بیشینه کند. این پاداش‌ها در واقع یک محرک برای عامل در مسیر یادگیری هستند که عامل برای رفتار در مسیر مطلوب به آن‌ها نیاز دارد. همه‌ی اکشن‌ها، پاداشی را در پیش دارد که این پاداش به سه نوع است، نوع اول پاداش مثبت است که نشان می‌دهد که رفتار انجام شده مطلوب و در جهت رسیدن به هدف است، پاداش منفی که نشان‌دهنده‌ی آن است که این اکشن در خلاف جهت رسیدن به هدف است و باید از آن دوری شود و پاداش صفر که به معنای این است که عامل هیچ کار خاصی انجام نداده است.

در واقع این پاداش است که مشخص می‌کند که عامل در هر وضعیت چه اقدامی را باید انجام دهد. عامل در حین فرآیند یادگیری متوجه می‌شود که پاداش‌های مثبت نشان‌دهنده‌ی حرکت به سمت هدف است و پاداش منفی، حرکت بر خلاف هدف را مدل می‌کند. عامل به ازای هر اکشنی که انجام می‌دهد، یک پاداشی را از محیط دریافت می‌کند که به آن پاداش لحظه‌ای [۱] می‌گویند. از طرفی عامل باید اکشنی را انتخاب کند که پاداش آینده را بیشینه می‌کند، اما بعضی از این اکشن‌ها عواقب بلند مدتی دارند. در واقع به خاطر همین عواقب بلند مدت است که علاوه بر پاداش لحظه‌ای، باید پاداش آینده را نیز در نظر بگیریم. زیرا ممکن است در بعضی مسائل، انتخاب یک حرکت از سوی عامل، در لحظه پاداش زیادی داشته باشد اما در بلند مدت، این اکشن عامل را به سمتی ببرد که در نهایت، پاداش کلی کمتر از حالت بهینه شود یا اینکه حتی در بدترین حالت، عامل اصلا به هدف خود نرسد. این موضوع در مساله‌های مختلف مصادیق متفاوتی دارد اما آنچه مشترک است این است که همیشه پیروی از اکشنی که بیشترین پاداش را دارد مطلوب نیست، زیرا این روش و به طور کلی، الگوریتم‌های حریصانه، می‌تواند به راحتی در یک نقطه‌ی بهینه‌ی محلی گیر بیافتد و نتواند به هدف اصلی خود برسد. در بعضی مسائل گاهی لازم است که عامل بتواند به میزان کمی خود را فدا کند و حتی اکشن با امتیاز منفی را برگزیند تا اینکه بتواند در نهایت، پاداش کلی را در پایان اپیزود، بیشینه کند.

برای مثال یک محیط را در نظر بگیرید که عامل قصد دارد که از نقطه‌ی شروع به پایان برسد، همچنین در این نقشه، بین نقطه‌ی شروع و پایان، مسیری وجود دارد که اگر عامل به آن وارد شود ممکن است آسیب ببیند پس به ازای ورود به این مسیر پاداش لحظه‌ای منفی دریافت می‌کند، اما این مسیر کوتاه‌ترین مسیر ممکن است و اگر عامل هر مسیر دیگری را انتخاب کند، پاداش نهایی کم‌تر خواهد بود، پس اگر هدف عامل این باشد که در کوتاه‌ترین زمان ممکن به نقطه‌ی پایان برسد، باید از پاداش لحظه‌ای خود بگذرد تا بتواند مجموع پاداش در پایان اپیزود را بیشینه کند زیرا انتخاب بعضی از اکشن‌ها دارای پاداش با تاخیر [۲] است، به همین دلیل است که عامل باید در انتخاب اکشن در هر وضعیت، این پاداش را نیز در نظر بگیرد.

در انتخاب اکشن بهینه در هر استیت، پارامتری به نام ضریب تخفیف [۳] تعریف می‌شود که با γ نشان داده می‌شود، این ضریبی است که در پاداش مورد انتظار آینده ضرب می‌شود و مقداری بین صفر و یک دارد. این پارامتر اهمیت پاداش تاخیری را در مقابل پاداش لحظه‌ای مشخص می‌کند.

$$R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \ldots + \gamma^n R_{t+n}$$

اگر این پارامتر به صفر نزدیک تر باشد، پاداش لحظه‌ای در مقابل پاداش تاخیری اهمیت بالاتری دارد، در حالی که اگر به یک نزدیک تر باشد، پاداش تاخیری اهمیت بالاتری پیدا می‌کند و از این جهت عامل آینده‌نگرتر می‌شود.

سوال دوم الگوریتم‌های وابسته به مدل با کمک یک transition table کار می‌کنند، این جدول در واقع یک کتاب راهنما برای عامل است که همه‌ی آن دانشی را که عامل برای موفق بودن در محیط نیاز دارد را در خود دارد.

[۱] Immediate reward
[۲] Delayed reward
[۳] Discount factor

طبیعتا نوشتن همچین کتابی سخت و در بعضی موارد غیرممکن است. به این دلیل است که الگوریتم‌های مستقل از مدل کاربرد عملی بیشتری دارند.

در الگوریتم‌های مستقل از مدل، عامل از طریق تجربه‌ی واقعی در محیط یاد میگیرد به جای آنکه صرفا از یک کتاب مرجع استفاده کند. این موضوع ما را قادر می‌سازد تا عناصر احتمالاتی و دنباله‌های طولانی از action-state ها را در مساله تعریف کنیم.

سوال سوم در بحث یادگیری تقویتی، برای انتخاب اکشن در هر وضعیت می‌توان دو سیاست را در پیش گرفت، سیاست اول این است که بر اساس دانشی که از محیط کسب کرده‌ایم، بهترین اکشن را انتخاب کنیم و سیاست دوم این است که به صورت تصادفی یک حرکت جدید را برگزینیم. برای مثال ما وقتی می‌خواهیم به رستوران برویم، یکی از انتخاب‌های ما می‌تواند بهترین رستورانی باشد که می‌شناسیم، ولی یکی دیگر از انتخاب ها این است که یک رستوران جدید را انتخاب کنیم و آن را برای اولین بار تست کنیم. این انتخاب می‌تواند مطلوب باشد یا نباشد، اما آنچه مهم است این است که انتخاب بهترین اکشن در هر استیت در واقع جست‌وجو در یک فضای محلی است و امتحان کردن انتخاب‌های رندوم این امکان را به ما می‌دهد که فضای تصمیم گیری خود را گسترش دهیم. باید به خاطر داشته باشیم که یادگیری در این روش مبتنی بر تعامل با محیط و کسب تجربه از آن است، پس طبیعتا تنها دنبال کردن بهترین حرکت در فضایی که تاکنون تجربه کرده‌ایم نمی‌تواند ما را به بهینه‌ی سراسری برساند.

در همان مثالی که در مورد رستوران زدیم، اگر تنها بهترین رستورانی را که می‌شناسیم انتخاب کنیم، طبیعتا ممکن است رستوران بهتری وجود داشته باشد که در فضای تجربه‌ی ما نباشد و تنها با یک حرکت رندوم است که می‌توانیم این فضا را گسترش دهیم.

در واقع هدف کلی این است که تا جایی که می‌توانیم اطلاعات کافی را با کسب تجربه از محیط به دست آوریم تا بتوانیم بهترین تصمیم را در دامنه‌ی این اطلاعات اتخاذ کنیم.

یکی از مهم‌ترین مسائلی که در یادگیری بدون exploration داریم این است که اگر بخواهیم همیشه بهترین حرکت را در هر وضعیت انتخاب کنیم، یعنی به صورت کاملا حریصانه‌ای به دنبال جواب بهینه باشیم، به راحتی در یک نقطه‌ی بهینه‌ی محلی که در دامنه‌ی تجربه‌های کسب شده از محیط است گیر می‌افتیم.

روشی که می‌توانیم از این مشکل رهایی پیدا کنیم greedy ϵ نام دارد. این روش به این صورت است که ابتدا یک عدد رندوم را تولید می‌کنیم و سپس در صورتی که این عدد رندوم کمتر از مقدار ϵ باشد سیاست حریصانه را در پیش می‌گیریم و در غیر این صورت به سراغ کاوش در محیط مساله می‌رویم و یک حرکت تصادفی را انتخاب می‌کنیم.

یکی دیگر از روش‌ها یا سیاست‌هایی که می‌توانیم برای فرار از این نقطه‌ی بهینه‌ی محلی استفاده کنیم، سیاست softmax است. این سیاست به صورت زیر تعریف می‌شود:

$$P(a) = \frac{e^{\beta * Q(a)}}{\sum_{a_i \in A} e^{\beta Q(a_i)}}$$

در این معادله β حکم پارامتر دما را دارد و عملکرد آن مشابه دمایی است که در روش تبرید شبیه‌سازی شده استفاده می‌کنیم. در ابتدا که دما بالاست، احتمال انتخاب همه‌ی اکشن‌ها تقریبا با هم برابر است و کاوش در محیط بیشتری انجام می‌شود در حالی که با کاهش دما در فرآیند آموزش، احتمال انتخاب اکشن برتر افزایش پیدا می‌کند و الگوریتم به این سمت می‌رود که به جای انتخاب‌های تصادفی، بهترین گزینه را بیشتر انتخاب کند و در پیش بگیرد.

۲ شبیه‌سازی

در این مساله ما یک ماز داریم و می‌خواهیم که عامل خود را از نقطه‌ی شروع به نقطه‌ی پایان برسانیم، در این مسیر خانه‌های ماز هزینه‌های متفاوتی دارند به این صورت که اگر به هر خانه‌ای وارد شویم، با هزینه‌ی متفاوتی روبرو می‌شویم و در نهایت می‌خواهیم به عامل خود یاد دهیم تا بتواند با کمترین هزینه به خانه‌ی هدف برسد، این خانه شامل امتیاز زیاد است، پس در حقیقت عامل قصد دارد تا از این مساله بیشترین امتیاز را به دست آورد.

برای حل این مساله از روش یادگیری تقویتی Q-Learning استفاده می‌کنیم. این روش در واقع یک روش مستقل از مدل است و عامل برای انتخاب حرکت‌های خود مدلی از محیط نمی‌سازد یا مدل از پیش تعریف شده‌ای را دنبال نمی‌کند. یک معیاری که می‌توانیم با استفاده از آن وابسته بودن یا نبودن الگوریتم را به مدل بررسی کنیم این است که اگر عامل در پایان یادگیری بدون اینکه حرکتی را انجام دهد بتواند وضعیت بعدی محیط و امتیاز آن را پیش‌بینی کند، این یادگیری وابسته به مدل است، اما همانطور که در این مساله مشخص است، عامل پس از یادگیری نمی‌تواند بدون انجام دادن حتی یک حرکت، وضعیت‌های بعدی محیط را تخمین بزند، بلکه باید حرکت‌ها به ترتیب انجام شوند تا عامل بتواند وضعیت‌های بعدی خود را متوجه شود.

در واقع آنچه در انتهای یادگیری با این روش داریم این است که در هر وضعیتی که قرار داریم، انجام چه اکشنی از همه بهتر است و عامل را در نهایت به بیشترین پاداش می‌رساند.

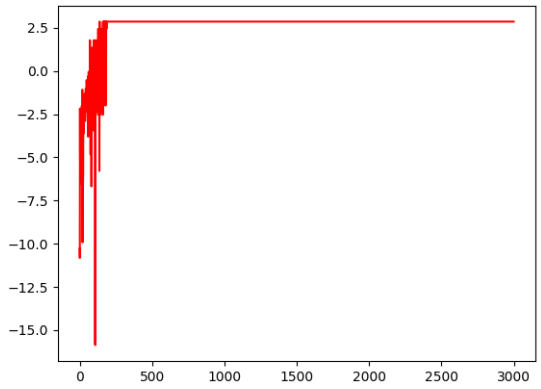
در این مساله از روش یادگیری Q-Learning مبتنی بر روش td-error استفاده می‌شود، روش آپدیت ماتریس Q به صورت زیر است:

$$Q[s][a] = Q[s][a] + \alpha * (R_t + \gamma * Q[\bar{s}][\bar{a}] - Q[s][a])$$

در عبارت بالا، مقدار $R_t + \gamma * Q[\bar{s}][\bar{a}]$ نام دارد و مقدار $R_t + \gamma * Q[\bar{s}][\bar{a}] - Q[s][a]$ TD Target نام دارد.

نتایج شبیه‌سازی به صورت زیر است:

الف در فرآیند یادگیری تعداد قدم‌های مجاز برای عامل در هر اپیزود برابر با ۱۰۰ قرار داده شده است. به این صورت که در هر اپیزود، عامل تنها می‌تواند ۱۰۰ قدم را بردارد و پس از این عامل اگر به خانه‌های هدف نرسیده باشد شکست می‌خورد. در هر قدم، مقدار پاداشی را که عامل از محیط کسب می‌کند به دست می‌آوریم و روی تعداد قدم‌ها میانگین می‌گیریم، نتیجه به صورت زیر است:



ب پس از پایان یادگیری، عامل می‌تواند تنها با استفاده از سیاست آموخته شده، در هر استیت، حرکت مناسب را انتخاب کند، به همین منظور در هر قدم با استفاده از معادله‌ی زیر، حرکت بهینه را در هر استیت انتخاب می‌کنیم:

$$argmax_{a_i} Q[s]$$

پس از این فرآیند یادگیری، سیاست بهینه و مسیر انتخاب شده مطابق با آن چیزی است که در فایل optimal.policy ذخیره شده است و به پیوست پروژه ارسال می‌شود.

ج پشیمانی در الگوریتم یادگیری تقویتی کاملا مشابه با آن چیزی است که یک انسان در تصمیمات خود دارد، در همان مثال رستورانی که در قسمت قبل به آن اشاره شد، وقتی یک رستوران جدید را در مساله کاوش می‌کنیم، ممکن است از آن راضی باشیم که در این صورت دامنه‌ی اطلاعات ما از محیط افزایش یافته و می‌توانیم حرکت‌های بهتری را انتخاب کنیم، یا اینکه از انتخاب این رستوران پشیمان می‌شویم و تصمیم می‌گیریم که دیگر از آن استفاده نکنیم، میزان پشیمانی عامل در این الگوریتم به صورت عکس پاداش تعریف می‌شود به این معنا که در نمودار بخش الف، قسمت بالای نمودار تا مقدار بیشنیه‌ی پاداش برابر با میزان پشیمانی عامل است.