پردیس علوم
دانشکده ریاضی، آمار و علوم کامپیوتر

به‌کارگیري یادگیري تقویتي عمیق براي آموزش عامل هوشمند جهت بازي Snake

نگارنده

رامتین راهوریان

استاد راهنما: دکتر هدیه ساجدي

پایان‌نامه برای دریافت درجه کارشناسی
در رشته علوم کامپیوتر

بهمن ۱۴۰۰

چکیده

یادگیری تقویتی شامل خانواده ای از الگوریتم ها است که برای کنترل و تصمیم گیری استفاده می شوند. این رویکردها مسائلی را حل می کنند که می توانند به عنوان فرایند تصمیم گیری مارکوف (MDP)[1] بیان شوند. بدان معنی که مساله را باید با مجموعه ای از حالات S (به عنوان مثال بر اساس موقعیت مار)، مجموعه ای از اقدامات A (به عنوان مثال، بالا، پایین، راست، چپ)، یک عملکرد پاداش R (به عنوان مثال $+10$ هنگامی که مار سیب می خورد، -10 وقتی مارها به دیوار برخورد می کنند) و یک تابع گذار T که تغییرات بین حالت ها را توصیف می کند،بیان کرد. برای استفاده از یادگیری تقویتی، باید مساله را با استفاده از این چهار جزء فرموله کنیم.

در این پایان نامه، این فرضیه بررسی می‌شود که کاربردهای یادگیری تقویتی برای انجام تجزیه و تحلیل دقیق مناسب نیستند. با ترکیب یادگیري عمیق تقویتي و شبکه عصبی، عامل بازی Snake آموزش داده می شود. وابستگی های این بازی به پایتون، کراس، تصویر کیت علمی و جریان تانسور است. برای آموزش عامل در بازی، از SARSA[2] برای کنترل قابلیت های حرکتی مار طبق میل کاربر استفاده می شود. یادگیری عمیق Q در ماژول یادگیری استفاده می شود و نرخ یادگیری الگوریتم SARSA نتایج موثر را در تکرارهای مختلف عامل نشان می دهد.

[1] Markov Decision Process

[2] State–action–reward–state–action

پیشگفتار

با بزرگتر شدن و پیچیده تر شدن دنیای بازی، اطمینان از قابل بازی بودن و بدون اشکال بودن آنها برای توسعه دهندگان سخت تر می شود و شرکت های بازی به دنبال ابزارهای جدیدی از جمله هوش مصنوعی هستند تا به غلبه بر چالش های گسترده آزمایش محصولات خود کمک کنند. هوش مصنوعی و بازی برخلاف تصور عموم ، با هم خوب کنار نمی آیند چرا که بین هوش مصنوعی و رفتار مصنوعی تفاوت وجود دارد. ما نمی خواهیم عوامل بازی های ما از بازیکنان پیشی بگیرند. ما می خواهیم آنها به همان اندازه که برای ارائه سرگرمی و تعامل لازم است باهوش باشند. ما نمی خواهیم از حداکثر محدودیت ربات ML[۳] خود استفاده کنیم ، همانطور که معمولا در صنایع مختلف نیز اینگونه انجام می دهیم. حریف نباید کامل باشد و از رفتاری شبیه به انسان تقلید کند. از آنجایی که با وجود چالش برانگیز بودن بازی های ویدیویی به راحتی قابل رسمی کردن هستند، برای مدت طولانی یکی از حوزه های محبوب تحقیقات هوش مصنوعی بوده است. برای دهه ها، توسعه دهندگان بازی سعی کرده اند عاملی را با هوش شبیه سازی بسازند، به ویژه برای ساخت بازیکنی با هوش مصنوعی که بتواند بازی را بر اساس تجربه بازی اش یاد بگیرد، نه اینکه صرفا از یک استراتژی ثابت پیروی کند. برنامه نویسی پویا می تواند مشکل را با تعداد کمی از حالات و ساختار تصادفی زیرین ساده حل کند، اما برای حالت پیچیده کارآمد نمی باشد. البته بازی ها فقط سرگرمی نیستند. آموزش یک عامل مجازی برای عملکرد بهتر از بازیکنان انسانی می تواند به ما بیاموزد که چگونه فرایندهای مختلف را در زمینه های فرعی مختلف و هیجان انگیز بهینه کنیم. در این پایان نامه ، یک عامل هوش مصنوعی بررسی می شود که قادر به یادگیری نحوه بازی کردن بازی محبوب Snake از ابتدا است.

---

[۳] Machine Learning

فهرست مطالب

فصل ۱

معرفي و پيش‌زمينه

۱.۱ معرفي

انجام بازی های ویدیویی چالش برانگیز و سرگرم کننده است. بازی ها به عنوان جالب ترین حوزه هوش مصنوعی (AI)[۱] در نظر گرفته می شوند. با هوش شبیه سازی، توسعه دهندگان بازی در حال آموزش عوامل هستند و بازی هایی را برای کمک به یادگیری بسیاری از مفاهیم جدید ارائه می دهند. استراتژی های مختلفی دنبال می شود و از زبان های برنامه پویا در توسعه بازی ها استفاده می شود. ترکیبی از شبکه های عصبی و یادگیری عمیق برای فرمول بندی بازی ها استفاده می شود. چنین ترکیبی فرصت هایی را برای درک واضح چیزها و یادگیری موثر رفتار تقویتی فراهم می کند.

این روزها ترکیبی از یادگیری ماشینی و یادگیری تقویتی در توسعه بازی ها استفاده می شود. عوامل آموزش می بینند، و تفسیرها برای آزمایش کارایی بازی گرفته می شود. قوانین خاصی در بازی طراحی شده و عوامل بازی برای کمک به حل مشکلات خاص آموزش می بینند. در یادگیری ماشینی، یادگیری تقویتی یکی از جذاب ترین فناوری هاست که در آن اقدامات بهینه در هر حالتی با آزمون و خطا آموخته می شود. این فرآیند آزمون و خطا به حل موثر مشکلات کمک می کند و دیدگاه منحصر به فردی از جنبه بررسی عملکرد یادگیری تقویتی ارائه می دهد.

در این پایان نامه بازی Snake برای بررسی حالات مختلف یادگیری تقویتی اصلاح شده است. از یادگیری عمیق برای آموزش حالت های بازی و ارزیابی عملکرد مار در بازی استفاده می شود. هدف اصلی این پایان نامه آموزش عامل بازی Snake با استفاده از یادگیری عمیق و هوش مصنوعی

[۱] Artificial Intelligence

است. اجزا جدید اضافه شده به تجزیه و تحلیل بازی، کار اصلی یادگیری تقویتی است.

با تلفن های همراه نوکیا، بازی مار به یکی از محبوب ترین بازی ها تبدیل شده است. بازی Snake نوکیا به گونه ای طراحی شده است که یک بازیکن می تواند وضعیت کار مار را کنترل کند. مراحل مختلفی در بازی طراحی شده است تا به تدریج بر پیچیدگی آن افزوده شود. برخی از قوانین باید در بازی رعایت شود. به عنوان مثال، مار سعی می کند چیزی را بخورد و به دنبال غذا می دود. خوردن مواد غذایی مختلف باعث می شود اندازه مار بزرگتر شود و در نتیجه امتیاز جمع شود. با افزایش اندازه مار، پیچیدگی بازی نیز افزایش می‌یابد، بنابراین کسب امتیاز بالا دشوارتر می‌شود.

در بازی Snake که تغییر یافته است، آب نبات های تصادفی ظاهر می شوند و مار هر بار که یک آب نبات می خورد رشد می کند. تفاوت این است که یک آب نبات‌ها سمی است. اگر مار آب نبات سمی را بخورد، می میرد. امتیاز نهایی به اندازه مار در زمان مرگ بستگی دارد یا مانند حالت کلاسیک بازی، امتیازات با تعداد آب نبات های خورده شده توسط مار محاسبه می شود. الگوریتم Q-learning با استفاده از Keras به عنوان مدل شبکه عصبی وارد بازی شده است. استفاده از یادگیری عمیق در طراحی جدید Game Snake قابلیت های بالقوه Q-learning را افزایش می دهد. این بازی در Pygame طراحی شده است که به مدیریت موثر کد کمک می کند تا عوامل بتوانند برای ارائه کنترل کامل به کاربر آموزش ببینند. SARSA در آموزش عواملي كه منجر به کنترل کاربر برای حرکت قابلیت های مار مي شود استفاده شد. الگوریتم SARSA از الگوریتم on-policy و استفاده مي كند و برای انجام عمل بر مبناي Q-value است. الگوریتم SARSA به جای استفاده از خط مشی حریصانه، از خط مشی متداول استفاده می کند. با استفاده از الگوریتم on-policy عملکرد بازی موثرتر و اجرای فرآیند سریعتر می شود.

۲.۱ یادگیري ماشيني

یادگیری ماشینی (ML) نوعی هوش مصنوعی (AI) است که به برنامه‌های نرم‌افزاری اجازه می‌دهد تا در پیش‌بینی نتایج دقیق‌تر شوند، بدون اینکه به صراحت برای این کار برنامه‌ریزی شده باشند. الگوریتم های یادگیری ماشین از داده های گذشته به عنوان ورودی برای پیش بینی مقادیر خروجی جدید استفاده می کنند. موتورهای توصیه یک مورد رایج برای یادگیری ماشین هستند (سیستم‌های توصیه‌کننده کلاس مهمی از الگوریتم‌های یادگیری ماشین هستند که پیشنهادات «مرتبط» را به کاربران ارائه می‌دهند.) سایر کاربردهای محبوب عبارتند از: تشخیص تقلب، فیلتر هرزنامه

(اسپم)، شناسایی تهدید بدافزار، اتوماسیون فرآیند کسب و کار (BPA)[۲] و پیش‌بینی تعمیرات.

یادگیری ماشینی مهم است زیرا به شرکت ها دیدی از روند رفتار مشتری و الگوهای عملیاتی تجاری می دهد و همچنین از توسعه محصولات جدید پشتیبانی می کند. بسیاری از شرکت‌های پیشرو امروزی، مانند فیس‌بوک، گوگل و اوبر، یادگیری ماشینی را به بخش مرکزی عملیات خود تبدیل کرده‌اند. یادگیری ماشینی به یک تمایز رقابتی مهم برای بسیاری از شرکت ها تبدیل شده است.

یادگیری ماشینی کلاسیک اغلب بر اساس نحوه یادگیری الگوریتم در پیش‌بینی دقیق‌تر، طبقه‌بندی می‌شود. چهار رویکرد اساسی وجود دارد: یادگیری با نظارت، یادگیری بدون نظارت، یادگیری نیمه نظارتی و یادگیری تقویتی. نوع الگوریتمی که دانشمندان داده‌های الگوریتمی انتخاب می‌کنند، بستگی به نوع داده‌هایی دارد که می‌خواهند پیش‌بینی کنند.

۳.۱ یادگیری تقویتی

یادگیری تقویتی (RL) [۳] حوزه ای از یادگیری ماشین است. این حوزه این که چگونه عوامل هوشمند باید در یک محیط اقداماتی را انجام دهند تا مفهوم پاداش تجمعی را به حداکثر برسانند، مربوط می‌شود. یادگیری تقویتی یکی از سه نمونه اصلی یادگیری ماشین در کنار یادگیری تحت نظارت و یادگیری بدون نظارت است.

یادگیری تقویتی در عدم نیاز به ارائه جفت های ورودی/خروجی برچسب دار و عدم نیاز به اصلاح صریح اقدامات زیر بهینه (اپتیمال) با یادگیری نظارت شده متفاوت است. در عوض تمرکز بر یافتن تعادل بین شناسایی (چیزهای ناشناخته) و بهره برداری (دانش فعلی) است [۱]. الگوریتم‌های RL با نظارت جزئی می‌توانند مزایای الگوریتم‌های نظارت شده و RL را ترکیب کنند [۲].

محیط معمولاً در قالب فرآیند تصمیم گیری مارکوف (MDP) بیان می شود، زیرا بسیاری از الگوریتم های یادگیری تقویتی برای این متن از تکنیک های برنامه نویسی پویا استفاده می کنند. تفاوت اصلی بین روش های برنامه نویسی پویا کلاسیک و الگوریتم های یادگیری تقویتی در این است که دومی دانش یک مدل ریاضی دقیق از MDP را فرض نمی‌کند و هایMDP بزرگ را در جایی که روش‌های دقیق غیرممکن می‌شوند هدف قرار می‌دهند.

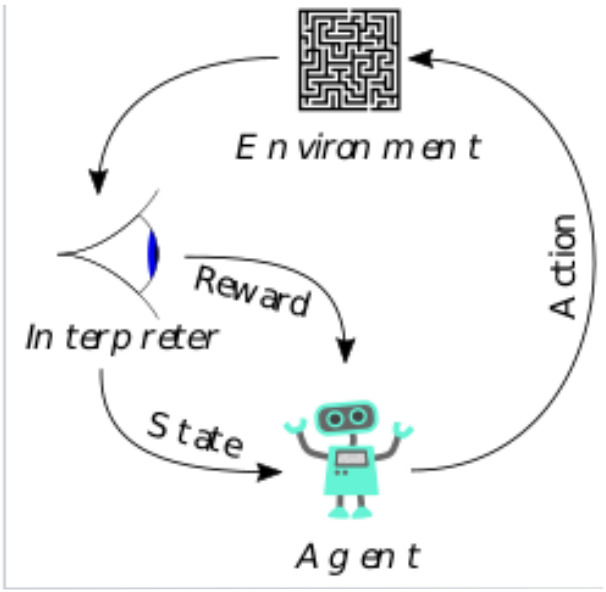
[۲] Business Process Automation
[۳] Reinforcement Learning

به دلیل عمومیت آن، یادگیری تقویتی در بسیاری از رشته ها مانند نظریه بازی، نظریه کنترل، تحقیق در عملیات، نظریه اطلاعات، بهینه سازی مبتنی بر شبیه سازی، سیستم های چند عاملی، هوش گروه و آمار مورد مطالعه قرار می گیرد. در ادبیات تحقیق و کنترل عملیات، یادگیری تقویتی، برنامه‌ریزی پویا تقریبی یا برنامه‌ریزی عصبی پویا نامیده می‌شود. مسائل مورد علاقه در یادگیری تقویتی نیز در تئوری کنترل بهینه بررسی شده است. که بیشتر به وجود و توصیف راه‌حل‌های بهینه و الگوریتم‌هایی برای محاسبه دقیق آن‌ها می‌پردازد و کمتر به یادگیری یا تقریب، به‌ویژه در غیاب یک مدل ریاضی از محیط می‌پردازد. در اقتصاد و تئوری بازی‌ها، یادگیری تقویتی ممکن است برای توضیح چگونگی ایجاد تعادل در عقلانیت محدود استفاده شود.

تقویت اساسی به عنوان یک فرآیند تصمیم گیری مارکوف (MDP) مدل می شود:

- مجموعه ای از حالت های محیط و عامل، S.
- مجموعه ای از اقدامات، A، از عامل.
- $P_a(s, s') = \Pr(s_{t+1} = s' s_t = s, a_t = a)$ احتمال انتقال (در زمان t) از حالت s به حالت s' تحت عمل a است.

R پاداش فوری پس از انتقال از s به s' با عمل a است.



شکل ۱.۱: چارچوب بندی معمول سناریوی یادگیری تقویتی (RL): یک عامل اقداماتی را در یک محیط انجام می دهد که به پاداش و نمایش وضعیت تفسیر می شود که به عامل بازخورد داده می شود.

هدف از یادگیری تقویتی این است که عامل یک خط مشی بهینه یا تقریباً بهینه را بیاموزد که "عملکرد پاداش" یا سایر سیگنال های تقویتی ارائه شده توسط کاربر را که از پاداش های فوری جمع می شود، به حداکثر می رساند. این شبیه به فرآیندهایی است که به نظر می رسد در روانشناسی حیوانات رخ می دهد. به عنوان مثال، مغزهای بیولوژیکی برای تفسیر سیگنال‌هایی مانند درد و گرسنگی به عنوان تقویت‌کننده‌های منفی، و لذت و دریافت غذا به عنوان تقویت‌کننده‌های مثبت طراحی شده‌اند. در برخی شرایط، حیوانات می توانند یاد بگیرند که در رفتارهایی شرکت کنند که این پاداش ها را بهینه می کند. این نشان می دهد که حیوانات قادر به یادگیری تقویتی هستند.

یک عامل یادگیری تقویتی اساسی هوش مصنوعی با محیط خود در مراحل زمانی گسسته تعامل دارد. در هر زمان t، عامل وضعیت فعلی s_t و پاداش r_t را دریافت می کند. سپس یک اقدام a_t را از مجموعه اقدامات موجود انتخاب می کند که متعاقباً به محیط ارسال می شود. محیط به حالت جدید s_{t+1} حرکت می‌کند و پاداش r_{t+1} مرتبط با انتقال (s_t, a_t, s_{t+1}) تعیین می‌شود. هدف یک عامل یادگیری تقویتی یادگیری یک خط مشی است:

که $\pi : A \times S \to [0, 1], \pi(a, s) = \Pr(a_t = a|s_t = s)$ پاداش تجمعی مورد انتظار را به حداکثر می رساند.

فرمول‌بندی مسئله به‌عنوان یک MDP فرض می‌کند که عامل مستقیماً وضعیت محیطی فعلی را مشاهده می‌کند. در این مورد گفته می شود که مشکل قابل مشاهده کامل است. اگر عامل فقط به زیرمجموعه ای از حالت ها دسترسی داشته باشد، یا اگر حالت های مشاهده شده توسط نویز خراب شده باشند، گفته می شود که عامل دارای قابلیت مشاهده جزئی است و به طور رسمی مشکل باید به عنوان فرآیند تصمیم گیری مارکوف تا حدی قابل مشاهده فرموله شود. در هر دو مورد، مجموعه اقدامات در دسترس برای عامل می تواند محدود شود. به عنوان مثال، وضعیت موجودی حساب می تواند مثبت باشد. اگر مقدار فعلی حالت ۳ باشد و انتقال حالت سعی کند مقدار را تا ۴ کاهش دهد، انتقال مجاز نخواهد بود.

وقتی عملکرد عامل با عملکردی که بهینه عمل می کند مقایسه می شود، تفاوت در عملکرد باعث ایجاد مفهوم پشیمانی می شود. به منظور عمل تقریباً بهینه، عامل باید در مورد پیامدهای بلندمدت اقدامات خود (یعنی به حداکثر رساندن درآمد آینده) استدلال کند، اگرچه پاداش فوری مرتبط با این مسئله، ممکن است منفی باشد.

بنابراین، یادگیری تقویتی به‌ویژه برای مشکلاتی که شامل مبادله پاداش بلندمدت در مقابل کوتاه‌مدت است، مناسب است. با موضوع موفقیت براي مشکلات مختلف، از جمله کنترل ربات، برنامه ریزی آسانسور، مخابرات، تخته نرد، checkers و Go (AlphaGo) اعمال شده است.

دو عنصر یادگیری تقویتی را قدرتمند می کند: استفاده از نمونه ها برای بهینه سازی عملکرد و استفاده از تقریب تابع برای مقابله با محیط های بزرگ. به لطف این دو جزء کلیدی، یادگیری تقویتی را می توان در محیط های بزرگ در شرایط زیر استفاده کرد:

- مدلی از محیط شناخته شده است، اما یک راه حل تحلیلی در دسترس نیست.
- فقط یک مدل شبیه سازی از محیط ارائه شده است (موضوع بهینه سازی مبتنی بر شبیه سازی)؛
- تنها راه جمع آوری اطلاعات در مورد محیط، تعامل با آن است.

دو مورد اول از این مسائل را می توان مسائل برنامه ریزی در نظر گرفت (از آنجایی که نوعی مدل در دسترس است)، در حالی که آخرین مورد را می توان یک مساله یادگیری واقعی در نظر گرفت. با این حال، یادگیری تقویتی هر دو مساله برنامه ریزی را به مسائل یادگیری ماشین تبدیل می کند.

۴.۱ یادگیري Q

یادگیري Q یک الگوریتم یادگیری تقویتی بدون مدل برای یادگیری ارزش یک عمل در یک حالت خاص است. نیازی به مدلی از محیط ندارد (بنابراین «بدون مدل»)، و می‌تواند مشکلات مربوط به تحولات و پاداش‌های تصادفی را بدون نیاز به سازگاری حل کند.

برای هر فرآیند تصمیم‌گیری محدود مارکوف (FMDP)[۴]، یادگیری Q یک خط‌مشی بهینه به معنای به حداکثر رساندن ارزش مورد انتظار کل پاداش در تمام مراحل متوالی، با شروع از وضعیت فعلی، پیدا می‌کند. یادگیری Q می تواند یک خط مشی انتخاب عمل بهینه را، برای هر FMDP مشخص، با توجه به زمان شناسايي نامحدود و یک خط مشی نیمه تصادفی شناسایی کند. "Q" به تابعی اشاره دارد که الگوریتم پاداش های مورد انتظار برای یک اقدام انجام شده در یک وضعیت خاص را محاسبه مي‌کند.

۱.۴.۱ الگوریتم یادگیري Q

در اینجا مدل مسئله تشکیل شده از یک عامل، وضعیت‌ها S و مجموعه از اقدامات A برای هر وضعیت.با انجام یک اقدام $a \in A$، عامل از یک وضعیت به وضعیت بعدی حرکت کرده و هر

[۴] Factored Markov Decision Processes

وضعیت پاداشی به عامل می‌دهد. هدف عامل حداکثر کردن پاداش دریافتی کل خود است. این کار با یادگیری اقدام بهینه برای هر وضعیت انجام می گردد. الگوریتم دارای تابعی است که ترکیب حالت اقدام را محاسبه می نماید :

$$Q : S \times A \to \mathbb{R}$$

قبل از شروع یادگیری، Q مقدار ثابتی را که توسط طراح انتخاب شده برمی گرداند. سپس هر بار که به عامل پاداش داده می‌شود، مقادیر جدیدی برای هر ترکیب وضعیت اقدام محاسبه می گردد. هسته الگوریتم از یک بروز رسانی تکراری ساده تشکیل شده‌است. به این ترتیب که بر اساس اطلاعات جدید مقادیر قبلی اصلاح می‌شود.

$$Q(s_t, a_t) \leftarrow \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{learning rate}} \times \left[\overbrace{\underbrace{R(s_t)}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_{a_{t+1}} Q(s_{t+1}, a_{t+1})}_{\text{max future value}}}^{\text{learned value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right]$$

که $R(s_t)$ پاداش s_t و $\alpha_t(s, a)$ است. نرخ یادگیری $(0 < \alpha \leq 1)$ ممکن است برای همه زوج‌ها یکسان باشد. مقدار عامل تخفیف γ بگونه است که $0 \leq \gamma < 1$.

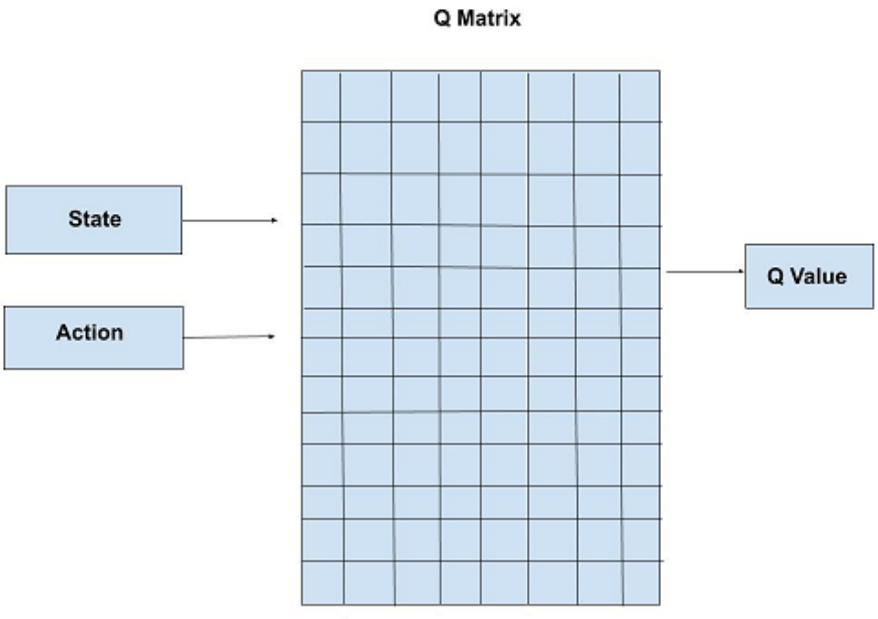
فرمول فوق معادل عبارت زیر است:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t)(1 - \alpha_t(s_t, a_t)) + \alpha_t(s_t, a_t)[R(s_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})]$$

یک اپیزود الگوریتم وقتی s_{t+1} به وضعیت نهایی می سد پایان می یابد. توجه کنید که برای همه وضعیت‌های نهایی s_f و $Q(s_f, a)$ مربوطه هیچگاه بروز نمی‌شود و مقدار اولیه خود را حفظ می‌کند.

۲.۴.۱ یادگیري Q عمیق

فرآیند یادگیري Q یک ماتریس دقیق برای عامل کار ایجاد می کند که می تواند به آن رجوع کند تا پاداش خود را در دراز مدت به حداکثر برساند. اگرچه این رویکرد به خودی خود اشتباه نیست، اما فقط برای محیط های بسیار کوچک عملی است و با افزایش تعداد حالت ها و اقدامات در



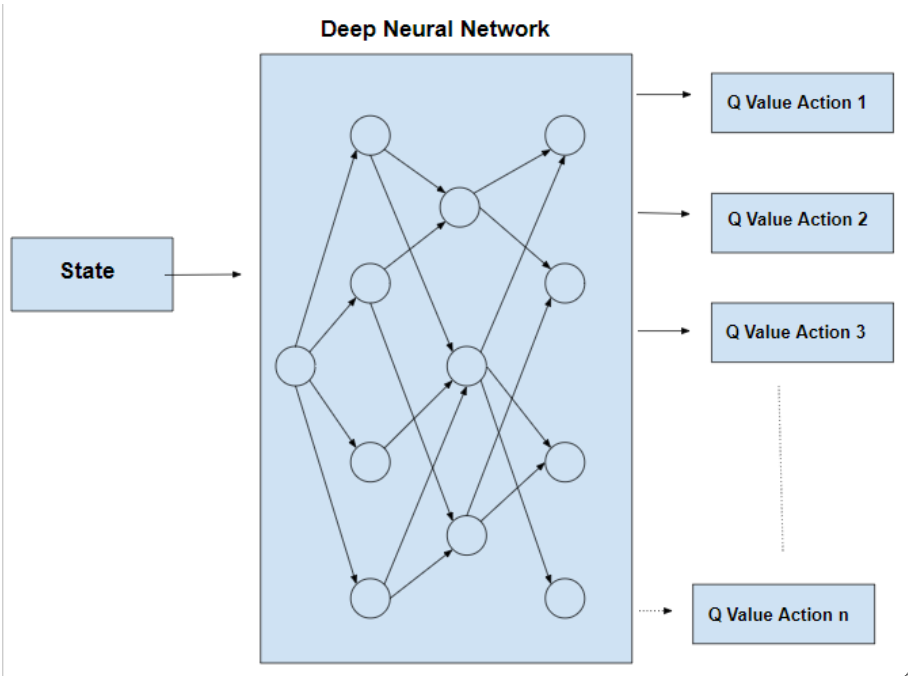
شکل ۲.۱: در این شکل State و Action به Q-matrix داده می شود و Q-value بدست می اید

محیط، به سرعت ممکن بودن خود را از دست می دهد.

راه حل مشکل فوق از درک این موضوع ناشی می شود که مقادیر موجود در ماتریس فقط اهمیت نسبی دارند، یعنی مقادیر فقط با توجه به سایر مقادیر اهمیت دارند. بنابراین، این تفکر ما را به یادگیري Q عمیق هدایت می کند که از یک شبکه عصبی عمیق برای تقریب مقادیر استفاده می کند. تا زمانی که اهمیت نسبی حفظ شود، این تقریب مقادیر ضرری ندارد.

مرحله کار اساسی برای یادگیري Q عمیق این است که حالت اولیه به شبکه عصبی وارد می شود و Q-value تمام اقدامات ممکن را در خروجی برمی گرداند.

تفاوت بین یادگیري Q و یادگیري Q عمیق را می توان به صورت زیر نشان داد:



شکل ۳.۱: در این شکل State به Network Neural Deep داده می شود و تعدادی Action Q-value بدست می اید

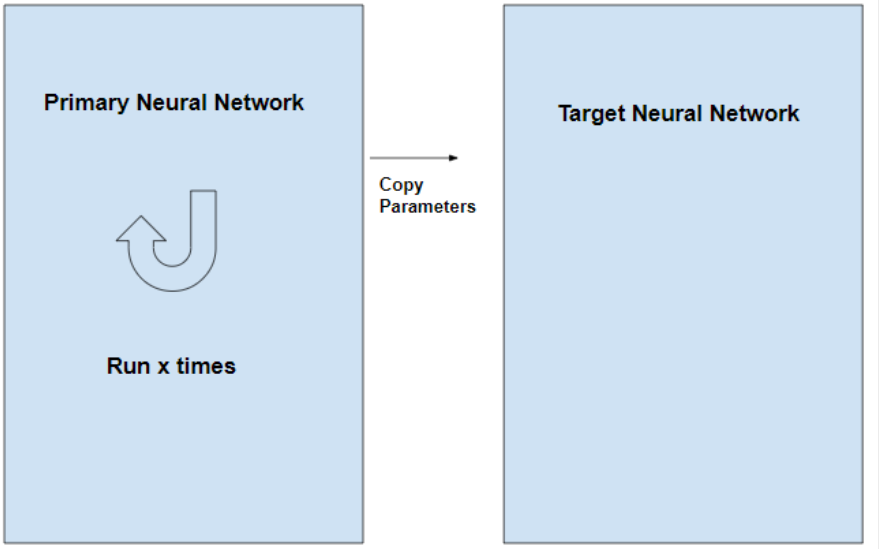
کد شبه برنامه:

```
Initialize Q_0(s,a) for all pairs (s,a)
s = initial state
k = 0
while(convergence is not achieved)
{
    simulate action a and reach state s'
    if(s' is a terminal state)
    {
        target = R(s,a,s')
    }
    else
    {
        target = R(s,a,s') + γmax_a' Q_k(s',a')
    }
    θ_{k+1} = θ_k − αΔ_θ E_{s' P(s'|s,a)}[(Q_θ(s,a) − target(s'))^2]|_{θ=θ_k}
    s = s'
}
```

شکل ۴.۱: شبه کد برای Q-Learning Deep

توجه داشته باشید که در معادله هدف $R(s,a,s')+\gamma max_{a'}Q_k(s',a') = \gamma max_{a'}Q_k(s',a')$، عبارت یک عبارت متغیر است. بنابراین در این فرآیند، هدف برای شبکه عصبی بر خلاف سایر فرآیندهای یادگیری عمیق معمولی که در آن هدف ثابت است، متغیر است.

این مساله با داشتن دو شبکه عصبی به جای یکی برطرف می شود. یک شبکه عصبی برای تنظیم پارامترهای شبکه و دیگری برای محاسبه هدف استفاده می شود که دارای معماری مشابه با شبکه اول است اما دارای پارامترهای ثابت است.



شکل ۵.۱: پس از تعداد x تکرار در شبکه اصلی، پارامترها در شبکه هدف کپی می شوند.

فصل ۲

پیشینه پژوهش

کارهای قابل توجهی در زمینه هوش مصنوعی و یادگیری ماشین انجام شده است. تکنیک ها و روش های متعددی برای کمک به حل مسائل محاسباتی پیشنهاد شده است. با استفاده از یادگیري تقويتي، بازی‌های مختلفی برای حل مسائل چالش برانگیز و ارائه درک عمیق از مدل‌های عصبی مختلف اجرا می‌شوند. با استفاده از زبان های مختلف، بازی ها با استفاده از مفهوم یادگیری تقویتی توسعه می یابند. مدل های ترکیبی با استفاده از یادگیری عمیق و یادگیری تقویتی در بازی ها تهیه شده اند.

با استفاده از نمایش نمادین، بازی های مختلفی از جمله بازی های کارتی و بازی های رومیزی تهیه می شود. بازی های ورق و بازی های رومیزی پیچیدگی خاصی ندارند. تکنیک‌های هوش محاسباتی (CI)[۱] برای کمک به حل مشکلات چالش برانگیز در این بازی‌ها استفاده شده است، اما روش‌های اختیار داشتن با استفاده از تکنیک CI شکست خوردند [۱].

تخته نرد شناخته شده ترین بازی است که با استفاده از یادگیري تقويتي ساخته شده است. در تخته نرد از الگوریتم مدل - آزاد، که یک الگوریتم تقويتي است، استفاده شد. این الگوریتم مبتنی بر لایه های چندگانه و پرسپترون چند لایه ای است که در این بازی برای ارزیابی موثر مقدار عملکرد، استفاده شده است. الگوریتم مدل - آزاد، همان عملکردهایی را فراهم می کند که یادگیری Q ارائه می کند.

مشارکت دیگری که توسط Tsitisklis نشان داده شد، ترکیب الگوریتم مدل - آزاد بود که باعث واگرایی در Qnetworking می شود. در یادگیری تقویتی، اکثر کارها بر اساس تقریبگرهای

[۱] Continuous Integratio

تابع خطی است. برای تخمین محیط می توان از شبکه عصبی عمیق برای ارائه تخمین بهتر نتایج مطابق با یادگیری تقویتی استفاده کرد [۲]. برای تخمین مقدار تابع، می توان از ماشین های بولتزمن برای ارائه نتایج دقیق در ترکیب با یادگیری تقویتی استفاده کرد.

تکامل عصبي برای کمک به تولید پارامتر شبکه عصبی مورد استفاده در بازی ها پیشنهاد شد. تکامل عصبي بخشی از الگوریتم یادگیری تقویتی است که به الگوریتم های تکاملی کمک می کند. تکامل عصبي بر اساس توپولوژی است و به عنوان یکی از بهترین روش های مورد استفاده در یادگیری تقویتی در نظر گرفته می شود. این کمک می کند تا وظایف یادگیری تقویتی را کارآمدتر و سریعتر انجام دهید. در مقایسه با روش تکامل عصبي قراردادي (معمولی) (CNE)[۲]، تکامل عصبي بهترین گزینه است، زیرا CNE را فقط می توان بر روی توپولوژی های ثابت اعمال کرد و این روش در ارائه دسترسی سریع برای انجام وظایف یادگیری تقویتی شکست خورده است [۳].

با استفاده از یادگیري، بازی های مختلفی وجود دارند که مفهومی دقیق برای حل مسائل مختلف ارائه می دهند. برای پیاده سازی الگوریتم یادگیری تقویتی باید با چالش های متعددی مواجه شد که پیچیده ترین آنها یافتن محیط مناسبی است که در آن عامل به گونه ای آموزش ببیند که نتایج را به طور موثر ارزیابی کند [۴]. با تعمیم رفتار مناسب عامل در بازی، الگوریتم های مختلفی برای کمک به حل پیچیدگی محیط، با توجه به رفتار عامل پیشنهاد شده است.

روش ها و رویکردهای یادگیری تقویتی مختلف برای کمک به حل مسائل پیچیده محیطی پیشنهاد شده است. بیشتر رویکردها در بازی های آتاری استفاده شده است. رویکرد برنامه پویا برای کمک به حل استراتژی بهینه استفاده شده است [۵] . روش دیگر، شاخص تخصیص است. روش شاخص تخصیص به یافتن انتخاب عمل بهینه کمک می کند. با استفاده از این روش می توان اقدامات مقایسه ای انجام داد و می توان از آن برای دستکاری رباتیک استفاده کرد. با استفاده از روش شاخص تخصیص، مشکلات تقویت تاخیری یافت می شود که مقدار شاخص آنالوگ ها نمی تواند آنها را پیدا کند [۶].

در یادگیری تقویتی از تکنیک ها و استراتژی های Ad-Hoc مختلف از جمله راهبردهای حریصانه استفاده شده است. تکنیک های مختلفی در الگوریتم های یادگیری تقویتی استفاده می شود. استراتژی تصادفی یک استراتژی اکتشافی است که به انجام عمل کمک می کند [۷]. استراتژی دیگری که در الگوریتم یادگیری تقویتی استفاده می شود، اکتشاف بولتزمن است.

مدل‌های تقویتی مختلفی پیشنهاد شده‌اند که در آنها عامل با استفاده از اعمال و ادراک به محیط متصل می‌شود. کل اصطلاحات این بازی ها بر اساس محیط و عامل است، اما این امر منجر

[۲] Computer Network Exploitation

به کندی عملکرد می شود زیرا انجام اقدامات و ارزیابی نتایج زمان زیادی می برد.

رویکردهای مختلفی برای انجام عملکردهای مختلف و ساده کردن پیچیدگی یک مساله خاص استفاده می شود. از رویکرد پویا می توان برای حل یک برنامه ساده استفاده کرد و این شامل استفاده از استدلال بیزی می شود. استدلال بیزی به حل پیچیدگی یک مسئله ساده کمک می کند [۸]. علاوه بر این، برای بدست آوردن مقدار تقریبی، از روش‌های مدل - آزاد استفاده می‌شود که در آن محاسبات مطابق با تکرار خط مشی انجام می‌شود.

روش دیگری که اخیرا مورد استفاده قرار گرفته است، Critic Actor نام دارد. Actor Critic یک خروجی دو لایه برای استفاده از شبکه تکي در اجرای وظایف مجزا تولید می کند. برای انجام این کار، هم مقدار تابع کامل و هم مقدار گسسته را ارائه می کند. الگوریتم دیگری که استفاده می شود، الگوریتم تکرار سیاست است که در آن معادله خطی و پیچیدگی را حل می کند. الگوریتم تکرار سیاست (Policy Iteration) نتایج ارزیابی را ارائه می دهد و هر بار که یک عمل انجام می شود خط مشی را در بازی تغییر می دهد. با استفاده از این الگوریتم، محاسبه تعداد تکرارهای انجام شده در موقعیت های پیچیده تر دشوار است. در بدترین شرایط، الگوریتم به اندازه کافی برای محاسبه تکرارهایی که نتایج ارزیابی را تولید می کند، معتبر نیست.

۱.۲ بازي مار و یادگیري تقویتي

با استفاده از بازي مار مفهوم یادگیری تقویتی آشکار می شود. بازی قدیمي مار با استفاده از رویکردها و الگوریتم های تقویتی اصلاح شده است. علاوه بر این، در بازی قدیمي مار، تولید ایده‌های جدیدی تقویت شدند که در آن قوانین و مقررات به طور مؤثر تعریف می‌شدند. بازی مار به گونه ای بود که یک بازیکن طبق قوانین از پیش تعیین شده بازی مي کرد. آب نبات در بازی ظاهر می شود و اگر مار آن را بخورد، به اندازه ۱ واحد رشد می کند.

گسترش بازی مار، با معرفی آب نبات های سمی به بازی داده می شود. اگر مار آب نبات سمی را بخورد، فورا می میرد. نشانه‌ها به گونه ای در بازی طراحی شده اند که قبل از تمام شدن ساعت باید تعداد معینی آب نبات خورده شود. امتیاز نهایی بازی به تعداد کل آب نبات های خورده شده توسط مار بستگی دارد. قوانین بازی شامل ۲ روش دیگر برای مردن مار می شود: ۱) مار می تواند از دم خود عبور کند اما نه بیش از سه بار. در گذر چهارم خواهد مرد. ۲) مار وقتی سرش به ناحیه مرزی برخورد کند می میرد. علاوه بر این، از تکنیک های جدید نیز استفاده می شود و با استفاده از الگوریتم SARSA ، فرآیند اقدام انجام شده، قابل اعتمادتر می شود. قوانینی برای

افزایش پیچیدگی و افزایش علاقه بازیکن تعریف شده است.

۲.۲ آنالیز ریاضي بازي مار

- به طور انتزاعی، هر مرحله از بازی مار را می توان به عنوان یافتن، راه رفتن خود اجتنابی (بدون اجتناب) (SAW)[۳] از یک شبکه در $\mathbb{R}$ در نظر گرفت. تحقق بازی مار را می توان به صورت ریاضی نشان داد:
  ـ صفحه‌ي $m \times n$ به‌صورت $G = (V, E)$ معرفي مي‌شود كه در آن $V = \{v_i\}$ مجموعه ای از رئوس است، که در آن هر رأس مربوط به یک مربع در تخته است، و $E = \{e_{ij} | v_i$ در مجاورت $v_j\}$.
  ـ مار به‌صورت يك مسير $\{u_1, \ldots, u_k\}$ معرفي مي‌شود كه در آن u_1 و u_k ابتدا و انتها مي‌باشند.

- $NP - Hard$: Viglietta (۲۰۱۳) [۶] ثابت کرد که هر بازی شامل آیتم کلکسیونی، شامل پیمایش مکان و مسیر یک طرفه NP-hard است. مار یک بازی است که شامل عبور از یک مسیر یک طرفه است، یعنی مار نمی تواند از خود عبور کند. بنابراین، اگر اطلاعات قبلی در مورد توالی ظهور غذا نداشته باشیم، انتخاب کوتاهترین SAW برای هر مرحله در یک قسمت NP-hard است.

۳.۲ الگوريتم تقريبي مار

با توجه به سختی NP (NP-hardness) در یافتن راه حل بهینه برای مار، یک الگوریتم اکتشافی توسعه داده شده است که به طور قابل توجهی برای مسئله مار، خوب عمل می کند و از آن به عنوان معیار برای الگوریتم های یادگیری تقویتی که بعداً اشاره شده می شود استفاده شده است. این الگوریتم، در الگوریتم ۱ نشان داده شده است.

[۳] Simple Additive Weighting

الگوریتم ۱: الگوریتم ابتکاری قطعی برای مار:

```
Build the graph G = (V, E), each square v_ij ∈ V
while Snake S= {s_1, ··· , s_k} ∈ V and Food F= {f} ∈ V do
    build {v_ij, v_kl} ∈ E, where v_ij, v_kl ∉ S with (i = ±k, j = l) or (i = k, j = ±l)
    initialize m × n array D
    assign D[i][j] = length of path between v_ij and f or MAX_NUM if no path exists
    find the min{D[i][j]} with {v_ij, s_1} ∈ E
    if D[i][j] ≠ MAX_NUM then
        find Path P = {v_ij, p_2, ··· , p_{n-1}, f}
        assign s̄_1 = f, s̄_2 = p_{n-2}, ··· s̄_k = p_{n-k}
        if s̄_1 and s̄_k is connected then
            assign s_1 = v_ij, s_2 = s_1, ···
            continue to step 2
        end
        else
            Find the longest Path between s_1 and s_k,
            P = {s_1, p_2, p_3 ··· , p_{n-1}, s_k}
            assign s_1 = p_2, s_2 = s_1, ···
            continue to step 2
        end
    end
end
```

۴.۲ کاهش فضای حالت برای یادگیری تقویتی

برای انجام و پیاده سازی الگوریتم های یادگیری تقویتی موفق برای انجام بازی، یکی از موانع اساسی، اندازه عظیم فضای حالت است. به عنوان مثال، اندازه صفحه بازی را به‌صورت $n \times n$ مشخص کنید. خیلی ساده، با استفاده از موقعیت دقیق مار و غذا، هر سلول را می‌توان با یکی از چهار شرط پارامتر کرد: { حاوی غذا، حاوی سر مار، حاوی بدن مار، خالی }. با اصل شمارش

ساده ، اندازه فضا ي حالت $|S|$ در $|S| > n^8$ صدق مي‌كند، به‌طوري كه وقتي n بسيار بزرگ است و يادگيري آن در فضاي حالت از چنين سايزي مشكل است، بسيار بزرگ است. از این رو، برای تسریع سرعت یادگیری، یک روش کاهش ساده این است که فقط موقعیت نسبی مار و غذا را با وضعیت فعلی مار ثبت کنید. دقیقاً، هر حالت در فضای کاهش یافته به شکل

$$\{w_s, w_t, w_r, q_f, q_t\}$$

است. در عبارت فوق، w_s، w_t و w_r توابع نشانگر هستند که آیا دیواری در مجاورت سر به ترتیب در جهت جلو، چپ و راست وجود دارد. q_f و q_t موقعیت نسبی غذا و دم نسبت به سر است. با استفاده از چنین نقشه‌اي، اندازه کل فضای حالت تنها $2^3 \cdot 4^2 = 128$ است، و عملکرد به طور برجسته در فرآیند یادگیری بهبود می یابد.

فصل ۳

پیاده سازی بازی

بازی مار با استفاده از پایتون ۶.۳ با برخی وابستگی های دیگر پیاده سازی شده است. از الگوریتم یادگیری Q با کراس به عنوان مدل شبکه عصبی استفاده شده است. وابستگی های این بازی به پایتون، کراس، تصویر کیت علمی و جریان تانسور است. بخشی از کد در این بازی با CNN[۱] ساخته می شود. الگوریتم یادگیري Q تا جايي ادامه دارد که در آن پارامتر گاما حل می شود و ماتریس اولیه Q به صفر می رسد.

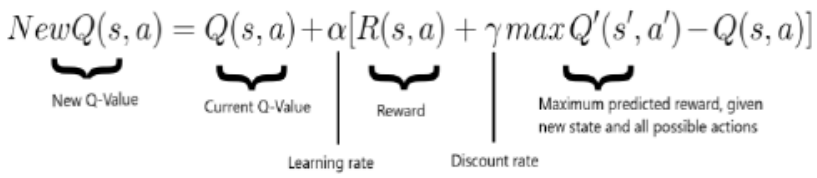
الگوریتم یادگیري تقويتي عمیق با استفاده از Keras انجام می شود. بازی مار در پایتون با استفاده از Pygame نوشته شده است. یادگیری D-Q به جای استفاده از رویکرد سنتی یادگیری ماشین انجام می شود. از معادله بلمن استفاده می شود که در آن مقادیر Q به روز می شوند. از ۱۲۰ نورون پنهان استفاده می شود و یک لایه حذفی برای بهینه سازی استفاده می شود.

برای آموزش بازی، موقعیت - عمل - جایزه - موقعیت، اعمال می شود. پس از آموزش داده ها، تست بازی اعمال می شود. در تست بازی، قابلیت حرکت مار تا پایان بازی تست و آزمایش می شود.

یادگیري Q عمیق برای کمک به افزایش کارایی یادگیري Q و ارائه یک جریان ثابت برای کنترل موقعیت مار در بازی استفاده می شود [۹]. چندین حالت در بازی استفاده می شود که در آن مار حرکت می کند و طبق میل کاربر قدم بعدی را پیش بینی می کند. برای پیش بینی حالت ها، جدول Q به طور مداوم به روز می شود و مقادیر به روز رسانی ها با استفاده از معادله بلمن ساخته

[۱] Convolutional Neural Network

می شوند.



این الگوریتم طوری عمل می کند که بازی با "s" شروع می شود و مقدار "Q" مقداردهی اولیه می شود. بر اساس وضعیت سیستم ،"s" یک عمل را اجرا می کند و انجام اقدامات تصادفی را انتخاب می کند. همانطور که Q مقدار به روز می شود و اقدامات بیشتری در بازی انجام می شود. اقدامات لازم برای جمع آوری پاداش و ارزیابی نهایی پس از خاتمه حالات انجام می شود. این نوع عملیات به عنوان حافظه رله برچسب گذاری شده است.

عامل به گونه ای آموزش می بیند که اقدام بعدی انجام شده در بازی بر اساس الگوریتم SARSA [۱۰] باشد. این الگوریتم یک الگوریتم مبتنی بر خط مشی است که به تخمین ارزش و امتیازات پیش بینی شده بر اساس آن کمک می کند. عامل در بازی با الگوریتم SARSA [۱۱] آموزش دیده است. در بازی، الگوریتم آموزشی بر اساس الگوریتم SARSA با استفاده از کد زیر مي‌باشد:

```
def train_short_memory(self, state, action, reward, next_state, done):
     target = reward
     if not done:
          target = reward + self.gamma *
np.amax(self.model.predict(next_state.reshape((1, state_dim)))[0])
     target_f = self.model.predict(state.reshape((1, state_dim)))
     target_f[0][np.argmax(action)] = target
     self.model.fit(state.reshape((1, state_dim)), target_f, epochs=1, verbose=0)
```

با استفاده از کد پایتون در بازی، الگوریتم SARSA برای آموزش عاملی که در آن اقدام بعدی انجام می شود، به‌کار مي‌رود. الگوریتم SARSA طوری کار می کند که وقتي عامل در حالت "S" است و اقدامات انجام شده توسط عامل منجر به پاداش "R" می شود. عامل به حالت "S" برمی گردد تا اقدام بعدی "A" را انجام دهد.

۱.۳ الگوریتم State-Action-Reward-State-Action (SARSA) و یادگیری تقویتی

در یادگیری تقویتی، الگوریتم SARSA به تخمین خط مشی ارزشی که دنبال می شود کمک می کند. این یک الگوریتم روی خط مشی است که در آن عامل اقدام را انجام می دهد و برای انجام اقدام بعدی به حالت نهایی باز می گردد. الگوریتم SARSA به گونه ای عمل می کند که عامل حالت اول را انجام می دهد، اولین اقدام را برای دریافت پاداش انجام می دهد و برای تصمیم گیری در مورد اقدام بعدی به حالت نهایی باز می گردد.

$Q(s,a)$ را به صورت دلخواه انتخاب کنید.
تکرار (براي هر دور)
s را مقداردهي کنید.
با استفاده از خط مشی مشتق شده از Q، a را از s انتخاب کنید
(براي مثال ϵ حریص)
تکرار (برای هر مرحله از قسمت)
اقدام a را انجام دهید، r، s را مشاهده کنید.
با استفاده از خط مشی مشتق شده از Q، a' را از s' انتخاب کنید.
(براي مثال ϵ حریص)

$$A(s,a) \leftarrow Q(s,a) + \alpha[r + 8Q(s',a') - Q(s,a)]$$
$$s \leftarrow s' \; ; \;\; a \leftarrow a'$$

با استفاده از الگوریتم SARSA، عامل اقدام می کند و با محیط تعامل می کند. این الگوریتم روی خط مشی روی فرآیندهای حالت و عمل کار می کند و با تغییر عملکرد به روز می شود. حالت نشان دهنده موقعیتی است که عامل در آن قرار می گیرد. عامل در بازی موقعیت خاص حالت را بر اساس اقدامات کاربر دریافت می کند. مقدار Q در الگوریتم SARSA مطابق با وضعیت فعلی عامل به روز می شود.

$$Q(S,A) \leftarrow Q(S,A) + a(R + \gamma(S',A') - Q(S,A))$$

معادله SARSA بر اساس تغییرات عامل در عمل به روز می شود. ،"S" "A" وضعیت اولیه عامل را نشان می دهد و پاداش "R" مطابق با عمل داده می شود. عامل به حالت نهایی می رسد و مطابق با محیط عمل می شود.

SARSA الگوریتم روی خط مشی است که به تصمیم گیری موثر کمک می کند. با توجه به رفتار عامل، الگوریتم SARSA مقدار جایی که عامل عمل خاصی را در حالت S انجام داده است، تخمین می زند. عملکرد الگوریتم SARSA ساده است و با هر تکرار عملکرد عامل، a نشان دهنده نرخ یادگیری است. γ پاداش با تخفیف را نشان می دهد. $Q(S, A)$ زمانی که عمل توسط عامل از هر حالت خاص S انجام می شود مقدار را ذخیره می کند.

فصل ۴

نتایج

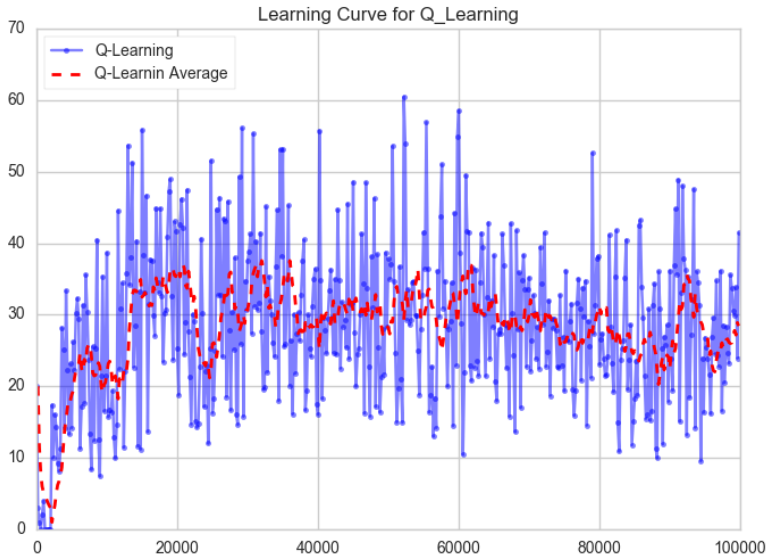
۱.۴ پارامترهاي تنظيم

ضریب تخفیف $0/95$ تنظیم شد، نرخ یادگیری نسبتاً کاهش می یابد که از $\alpha = 0/1$ شروع می شود و پاداش ها همانطور که در جدول ۱ نشان داده شده است تعیین شد.

ما می خواهیم که عامل کنترل، مار را به سرعت به سمت غذا ببرد و آخرین ستون در جدول ۱ ، برای انجام یک حرکتي است که در آن عامل را تشویق می کند تا مسیر کوتاه تری را به سمت غذا طی کند. این پاداش های منفی مانند عملکردی مشابه ضریب تخفیف γ عمل می کند. در این پایان نامه آزمون و خطا را روی ترکیب‌های مختلف پاداش برای موارد مختلف انجام شده است تا ترکیب فعلی مقادیر پاداش به عنوان یک ترکیب بهینه در بین همه آزمایش‌ها به‌دست آورده شود.

حالت	خوردن غذا	برخورد با دیوار	برخورد با خود مار	موارد دیگر
پاداش	$+500$	-100	-100	-10

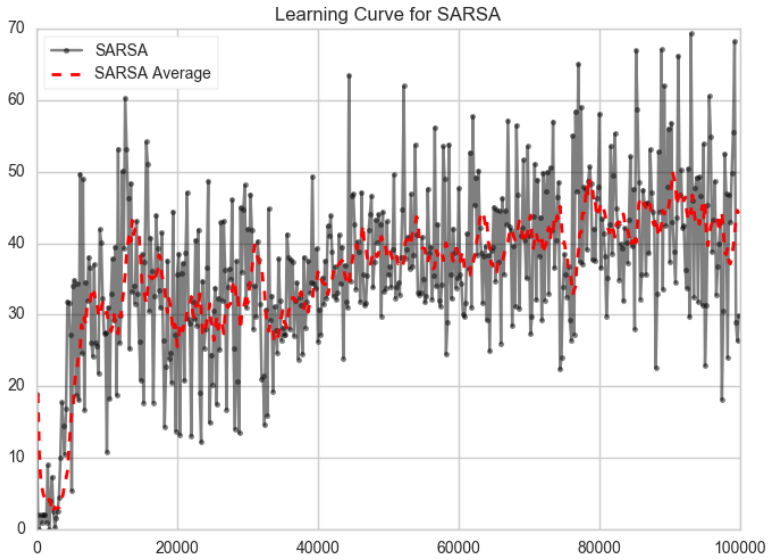
جدول ۱.۴: جدول پاداش



شکل ۱.۴ : منحنی های یادگیری از یادگیری Q

۲.۴ منحنی یادگیری

منحنی های یادگیری برای یادگیری Q (شکل ۱.۵) و SARSA (شکل ۲.۵) به ترتیب در زیر نشان داده شده است. خطوط خط قرمز نشان دهنده میانگین منحنی های یادگیری است.



شکل ۲.۴: منحنی های یادگیری از SARSA

ما به راحتی می توانیم مشاهده کنیم که در همان ابتدا عملکرد عامل با یادگیری Q سریعتر از عامل با SARSA بهبود می یابد، یعنی در کوتاه مدت، عامل با الگوریتم یادگیري Q از عامل با SARSA بهتر عمل می کند. اما با افزایش تعداد تکرارهای آموزشی، عملکرد عامل با یادگیری Q بهبود چندانی پیدا نمی کند، در حالی که عملکرد عامل با SARSA هنوز به طور قابل توجهی بهبود می یابد. یادگیری Q زمانی که دوره آموزشی کوتاه است به خوبی انجام می شود (در مقایسه با SARSA). اما در دوره طولانی، SARSA بهتر عمل مي‌کند.

دلیل اینکه عامل یادگیری Q در برخی موارد در دوره طولانی آموزش خوب عمل نمی کند این است که الگوریتم یادگیري Q ارزش Q خود درست بيني را حتی با نرخ یادگیری بسیار کمی تقویت می کند. این منجر به عملکرد قابل توجهی نوسان می شود. اگرچه به نظر می رسد عامل با SARSA در یک دوره آموزشی طولانی از عامل با الگوریتم یادگیری Q بهتر عمل می کند، اما در موارد بررسي در این جا، منحنی یادگیری نسبتاً کندی دارد.

روش			
یادگیری Q	SARSA	بهینه	تکرارها
$18/023$	$36/858$	$77/504$	10^4
$25/789$	$51/994$	$77/504$	10^5
$36/567$	$61/830$	$77/504$	10^6

۴.۴ مقایسه عملکرد

۳.۴ مقایسه عملکرد

عملکرد الگوریتم حل بهینه تقریبی به عنوان معیار در نظر گرفته شده است. همانطور که قبلاً اشاره کردیم، با توجه به اینکه مار یک مساله NP-hard است، بهترین معیاری که می‌توانیم در اینجا بدست آوریم، راه‌حل بهینه تقریبی است. و بعداً می‌توان نشان داد که عوامل ما با الگوریتم‌های یادگیری تقویتی نمی‌توانند این راه‌حل تقریبی را حتی با یک دوره آموزشی طولانی مدت شکست دهند.

سپس عملکرد عامل ها بر اساس دو الگوریتم یادگیری تقویتی با معیار - عملکرد راه حل بهینه تقریبی‌مان مقایسه شده است. در عین حال، تأثیرات دوره آموزشی بر عملکرد عوامل مار در نظر گرفتیم. 10^4، 10^5 و 10^6 تکرار آموزشی بر اساس الگوریتم‌های یادگیری تقویتی مختلف بر روی عوامل انجام داده شده است و به ترتیب عملکرد آنها ارزیابی شده است.

محدودیت زمانی برای بازی ۱ دقیقه تعیین شده است، ۱۰۰۰ تست برای عوامل با الگوریتم های مختلف اجرا می شود و میانگین امتیازی که عوامل مختلف می توانند به دست آورند محاسبه شده است. دلیل انتخاب ۱ دقیقه به عنوان محدودیت زمانی این است که در عرض ۱ دقیقه، عواملی با آن الگوریتم‌های یادگیری تقویتی می‌توانند زنده ماندن مار را کنترل کنند، بنابراین می‌توان مطمئن شد که تفاوت فقط در این است که آیا عوامل مختلف می‌توانند مسیر کوتاه‌تری برای غذا پیدا کنند. ۱ دقیقه یک دوره زمانی نسبتا طولانی برای نشان دادن تفاوت عملکرد قابل توجه بین الگوریتم های مختلف است در حالی که یک دوره زمانی نسبتا کوتاه است که ۱۰۰۰ تست را می توان با رایانه شخصی در زمان اجرای معقول انجام داد. نتایج در شکل ۲.۵ و جدول ۲ نشان داده شده است. در اینجا شایان ذکر است که عملکرد الگوریتم حل بهینه تقریبی به تعداد تکرارهای آموزشی مربوط نمی شود. فقط عملکرد عوامل با الگوریتم SARSA و یادگیری Q ارتباط مستقیمی با تعداد

تکرارهای آموزشی دارد. عملکرد راه حل بهینه تقریبی به عنوان یک معیار در همان جدول و شکل عملکرد عوامل با الگوریتم های یادگیری تقویتی مختلف نشان داده شده است.

نتایج نشان می‌دهد که در محدوده 10^4 تا 10^6 تکرار آموزشی، تعداد بیشتری از تکرارهای آموزشی به میانگین نمرات بهتری برای الگوریتم یادگیري Q و SARSA منجر می‌شود. با توجه به تعداد تکرارهای آموزشی یکسان، عامل آموزش دیده در SARSA عملکرد بهتری نسبت به آموزش یادگیري Q دارد.

۵.۴ نتیجه گیری

در این پروژه بازی مار با قوانین جدید تعریف شده پیاده سازی شده است. گسترش بازی مار با معرفی آب نبات های سمی در بازی ایجاد می شود. با استفاده از الگوریتم SARSA ، بازی در پایتون نوشته شده است. عملکرد الگوریتم SARSA با یادگیری تقویتی عمیق مشاهده می شود. با استفاده از الگوریتم SARSA کارایی بازی به دلیل عملکرد سریع کار افزایش می یابد. نتایج نسبتاً بهتری را در مقایسه با تکنیک های موجود ارائه می دهد. این روش که در آن فرآیند اولیه سازی کنش عامل شکل می گیرد و سپس اقدامات انجام می شود تا به مرحله بعد می رسند، کار مي‌کند. عملکرد بازی با توجه به عملکرد عامل و امتیاز نهایی مشاهده می شود و سپس ارزیابی می شود. الگوریتم یادگیري تقویتي عمیق با استفاده از Keras انجام می شود. میانگین امتیاز ۵۰ و امتیاز ثبت شده بازی مار ۱۵۰ است.

کتاب‌نامه

[1] S. Mukhopadhyay, O. Tilak and S. Chakrabarti, "Reinforcement Learning Algorithms for Uncertain, Dynamic, Zero-Sum Games," 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), vol. 5, no. 3, pp. 66-70, 2018 .

[2] B. Tsitsiklis, "An Analysis of Temporal-Difference Learning with Function Approximation," IEEE TRANSACTIONS ON AUTOMATIC CONTROL, vol. 42, no. 5, pp. 5-6, 1997.

[3] A. Notsu, K. Honda, H. Ichihashi and Y. Komori, "Simple Reinforcement Learning for Small-Memory Agent," 2011 10th International Conference on Machine Learning and Applications and Workshops, vol. 1, pp. 78-80, 2011.

[4] A. Jeerige, D. Bein and A. Verma, "Comparison of Deep Reinforcement Learning Approaches for Intelligent Game Playing," 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), vol. 6, no. 2, pp. 55-60, 2019.

[5] s. Çalişir and m. K. Pehlivanoğlu, "Model-Free Reinforcement Learning Algorithms: A Survey," 2019 27th Signal Processing And Communications Applications Conference (siu), vol. 4, no. 2, pp. 67-70, 2019

[6] G. Fenza, F. Orciuoli and D. G. Sampson, "Building Adaptive Tutoring Model Using Artificial Neural Networks and Reinforcement Learning," 2017 IEEE 17th International Conference on Advanced Learning Technologies (ICALT), vol. 8, no. 6, pp. 56-67, 2017 .

[7] M. Wang, L. Wang and T. Yue, "An Application of Continuous Deep Reinforcement Learning Approach to Pursuit-Evasion Differential Game," 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), vol. 3, no. 2, pp. 4-7, 2019.

[8] B. Gökçe and H. L. Akın, "Implementation of Reinforcement Learning by transfering sub-goal policies in robot navigation," 2013 21st Signal Processing and Communications Applications Conference (SIU), vol. 6, no. 4, pp. 7-9, 2013.

[9] Z. Wu, N. M. Khan, L. Gao and L. Guan, "Deep Reinforcement Learning with Parameterized Action Space for Object Detection," 2018 IEEE International Symposium on Multimedia (ISM), 2018.

[10] X. Wang, Y. Gu, Y. Cheng, A. Liu and C. L. P. Chen, "Approximate Policy-Based Accelerated Deep Reinforcement Learning," IEEE Transactions on Neural Networks and Learning Systems, 2019.

[11] G. Surma, "Towards data science," 10 2018. [Online]. Available: https://towardsdatascience.com/slitherin-solving-the-classicgame-of-snake-with-ai-part-3-genetic-evolution-33186e6be110.

[12] Risto Miikkulainen, Bobby Bryant, Ryan Cornelius, Igor Karpov, Kenneth Stanley, and Chern Han Yong. Computational Intelligence in Games. URL:ftp://ftp.cs.utexas.edu/pub/neural-nets/papers/miikkulainen.wcci06.pdf

[13] Gerald Tesauro. Temporal difference learning and td-gammon. Communications of the ACM, 38(3):5868, 1995.

[14] John N Tsitsiklis and Benjamin Van Roy. analysis of temporaldifference learning with function approximation. Automatic Control, IEEE Transactions on, 42(5):674690, 1997.

[15] Brian Sallans and Geoffrey E. Hinton. Reinforcement learning with factored states and actions. Journal of Machine Learning Research, 5:10631088, 2004.

[16] Lucas Jen An application of SARSA temporal difference learning to Super Mario URL: http://x3ro.de/downloads/MarioSarsa.pdf

[17] van Seijen, H., van Hasselt, H., Whiteson, S. and Wiering, M. (2009). A theoretical and empirical analysis of Expected Sarsa, 2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning pp. 177184. URL: http://goo.gl/Oo1lu

Abstract

In this research, we explore the hypothesis that Reinforcement Learning applications are not amenable to conduct close analysis. With the combination of Deep Reinforcement Learning and neural network, the snake game agent is trained. The dependences of this game are on python, Keras, sci-kit image, and tensor flow. For the training of the agent in the game, SARSA is used to control the moving capabilities of the snake as per the user's desire. Deep Q-learning is used in the learning module and the learning rate of the SARSA algorithm shows the effective results in the different agent iterations.

College of Science
School of Mathematics, Statistics, and Computer Science

Exploration of Reinforcement Learning to Play Snake Game

Ramtin Rahvarian

Supervisor: Dr. Hedie Sajedi

A thesis submitted in partial fulfillment of the requirements for
the degree of B.Sc. in Computer Science

February 2022