پردیس علوم
دانشکده ریاضی، آمار و علوم کامپیوتر

بررسی انواع روش‌های یادگیری تقویتی

نگارنده

زهرا قاسمی

استاد راهنما : دکتر هدیه ساجدی

پروژه برای دریافت درجه کارشناسی
در رشته علوم کامپیوتر

تیر ۱۳۹۸

چکیده

یادگیری تقویتی [1] نوع مهمی از یادگیری ماشین است که در آن عامل می‌آموزد چگونه در محیط با انجام اقدامات و بررسی نتایج رفتار کند. وظیفه اصلی یادگیری تقویتی استفاده از مشوق های مشاهده‌شده به‌منظور یافتن یک خط مشی بهینه و یا تقریبا بهینه برای محیط است. آموزش مستقیم یک تابع ارزیابی از مثال‌ها، بسیار چالش برانگیز است. در عوض می‌توان به برنامه آموزش داد تا قادر به درک برد یا باخت خود باشد. همچنین، از این اطلاعات به‌منظور یادگیری یک تابع ارزیابی استفاده کند که پیش‌بینی های دقیق و معقول و احتمال برد را در هر شرایط مفروض، تامین کند. یادگیری تقویتی انواعی دارد. یادگیری خنثی، یادگیری فعال و یادگیری Q-learning یا Q از انواع آن است.

در یادگیری خنثی خط مشی عامل ثابت است و وظیفه‌ی اصلی یادگیر، بررسی سودمندی حالت‌ها است. در یادگیری فعال، عامل نوع عملیاتی را که باید انجام دهد، یاد می‌گیرد. مهم‌ترین موضوع یادگیری فعال کاوش است: یک عامل باید به اندازه‌ی کافی از محیط خود تجربه کسب نماید تا چگونگی رفتار خود را در محیط یاد گیرد. یادگیری Q-learning نوعی از یادگیری تقویتی بدون مدل است که بر پایه برنامه ریزی پویا عمل می‌کند. یادگیری Q یک تکنیک یادگیری تقویتی است که با یادگیری یک تابع، سیاست مشخصی را برای انجام حرکات مختلف در وضعیت‌های مختلف دنبال می‌کند. در این پروژه انواع روش‌های یادگیری تقویتی مرور و در انتها دو مثال بررسی می‌شود.

کلمات کلیدی : یادگیری تقویتی، reinforcment learning ، سیاست‌گذاری، خط مشی، q-learning ، پویش.

[1] reinforcement learning

پیشگفتار

در این پروژه انواع روش‌های یادگیری تقویتی مورد بررسی قرار گرفته‌است. در یادگیری تقویتی عامل به صورت خودآموز و با استفاده از مشوق‌های مشاهده شده، یک خط مشی بهینه در محیط پیدا می‌کند. می‌توان به برنامه آموزش داد تا قادر به درک برد یا باخت خود باشد و از این اطلاعات برای بهبود رفتار خود استفاده کند. در فصل اول به تعاریف مرسوم در یادگیری تقویتی پرداخته می‌شود. در فصل دوم انواع روش‌های یادگیری تقویتی مورد یررسی و شرح قرار می‌گیرد. در فصل سوم مثال تیک-تک-تو مورد بررسی قرار گرفته می‌شود. درفصل آخر مثالی از q-learning شرح داده می‌شود.

فهرست مطالب

فصل ۱

تعاریف

یادگیری تقویتی یک یادگیری برای تبدیل موقعیت‌ها به کنش‌ها است به‌طوری‌که میزان پاداش را به حداکثر برساند. یادگیرنده نمی‌گوید که چه اقداماتی باید انجام شود، بلکه باید کشف کند که کدام اقدامات با تلاش آن‌ها بیشترین امتیاز را به‌دنبال خواهد داشت. در جالب‌ترین و چالش‌برانگیزترین اقدامات ممکن است یک اقدام منحصربه‌فرد نه تنها پاداش فوری به‌همراه داشته باشد، بلکه وضعیت بعدی و از این طریق، تمامی پاداش‌های بعدی را تحت تاثیر قرار دهد. این دو ویژگی (بررسی و تحویل محتوا و پاداش و تنبیه به تعویق افتادن)، ویژگی‌های برجسته یادگیری تقویتی هستند.

یادگیری تقویتی، مانند یادگیری ماشین یک مسئله همزمان است: یک کلاسی از روش‌های راه حل که به خوبی بر روی مشکل کار می‌کنند به‌علاوه زمینه‌ای که این مسئله و روش‌های حل آن را بررسی می‌کند. استفاده از یک نام برای سه مفهوم ساده است اما در عین حال مفاهیم هر یک باید به‌طور جداگانه حفظ شود. به طور خاص، تمایز بین مشکلات و روش‌های حل در یادگیری تقویتی بسیار مهم است و ناکامی در ایجاد این تمایز منبع بسیاری از سردرگمی‌ها است. ما مشکل یادگیری تقویتی را با استفاده از ایده‌ها در نظریه سیستم‌های دینامیکی، بخصوص به عنوان کنترل بهینه فرآیندهای تصمیم‌گیری مارکف، رسمی می‌کنیم. ایده اصلی این است که به سادگی بتوانیم مهم‌ترین جنبه‌های مسئله واقعی را که یک عامل یادگیری در طی زمان با محیط خود برای رسیدن به یک هدف مواجه می‌شوند، دربیابیم. عامل یادگیری باید بتواند تا حدودی وضعیت محیط را درک کند و باید اقداماتی را که بر وضعیت‌ها تاثیر می‌گذارد، انجام دهد. عامل همچنین باید یک هدف یا اهداف مربوط به وضعیت‌های محیط را داشته باشد. فرایندهای تصمیم‌گیری مارکوف قصد دارند فقط سه جنبه‌ی حساسیت، عمل و هدف را در ساده‌ترین شکل ممکن خود بدون بی‌اهمیت دانستن

هر یک از آن‌ها در نظر بگیرند. هر روشی که برای حل این مشکلات مناسب است، به عنوان یادگیری تقویتی شناخته می‌شود.

یادگیری تقویتی متفاوت از یادگیری نظارت شده [۱] است، نوع یادگیری مورد مطالعه در بیشتر تحقیقات فعلی در زمینه یادگیری ماشین است. یادگیری تحت نظارت، یادگیری از یک مجموعه آموزش از نمونه‌های برچسب‌زده‌شده است که توسط یک ناظر خارجی شناخته‌شده ارائه می‌شود. هر مثال شرح یک وضعیت همراه با یک مشخصه عمل صحیح است که سیستم باید در آن موقعیت اتخاذ کند، که اغلب برای شناسایی یک دسته که به کدام وضعیت تعلق دارد، در نظر گرفته می‌شود. هدف از این نوع یادگیری آن است که سیستم پاسخ‌های خود را به‌طور کلی استخراج یا تعمیم دهد. به‌طوری که در شرایطی که در مجموعه، آموزش وجود ندارد، سیستم به درستی عمل کند. این نوع مهمی از یادگیری است، اما به تنهایی برای یادگیری از تعامل مناسب نیست. در مسائل تعاملی غالبا دستیابی به نمونه‌هایی از رفتار مطلوب که معمولا صحیح و نماینده‌ای از تمام شرایطی است که عامل باید عمل کند، غیرممکن است. در ناحیه ناشناخته که در آن می‌توان انتظار داشت که یادگیری بیشتر سودمند باشد، یک عامل باید بتواند از تجربه خود آموزش ببیند.

یادگیری تقویتی، از آنچه که محققان یادگیری ماشین، یادگیری بدون نظارت [۲] می‌نامند نیز متفاوت است. یادگیری بدون نظارت معمولا درباره یافتن ساختار پنهان در مجموعه داده‌های بدون برچسب‌گذاری است. به نظر می‌رسد که شرایط یادگیری تحت نظارت و یادگیری بدون نظارت به طور کامل دسته‌بندی نمونه‌های یادگیری ماشین را طبقه‌بندی می‌کنند، اما اینطور نیست. اگر چه ممکن است آموختن یادگیری تقویتی به عنوان یک نوع یادگیری بدون نظارت باشد (زیرا به نمونه‌هایی از رفتار صحیح تکیه نمی‌کند)، اما یادگیری تقویتی تلاش می‌کند تا به جای یافتن ساختار پنهان، سیگنال پاداش را به حداکثر برساند. کشف ساختار در یک عامل می‌تواند در یادگیری تقویتی مفید باشد، اما به خودی خود، مسئله یادگیری تقویتی برای به‌حداکثررساندن یک سیگنال پاداش نیست. بنابراین ما یادگیری تقویتی را به عنوان نمونه سوم یادگیری ماشین، همراه با یادگیری تحت نظارت و یادگیری بدون نظارت در نظر می‌گیریم.

یکی از چالش‌هایی که در یادگیری تقویتی، و نه در سایر انواع یادگیری به‌وجود می‌آید، تعادل بین پویش [۳] و انتفاع [۴] است. برای به‌دست‌آوردن پاداش زیاد، یک عامل یادگیری

[1] supervised learning
[2] unsupervised learning
[3] exploration
[4] exploitation

تقویت‌کننده باید اقداماتی را که قبلا تلاش کرده انجام دهد را ترجیح دهد و در موعد مقرر به اثبات برساند. اما برای کشف چنین اقداماتی، باید اقداماتی را که پیش از این انتخاب نکرده است، امتحان کند. عامل باید از آنچه که در حال حاضر برای به دست آوردن پاداش به‌دست‌آورده است انتفاع کند، اما همچنین باید به‌منظور انتخاب گزینه‌های بهتر در آینده استفاده شود. معضل این است که نه پویش و نه انتفاع نمی‌تواند به‌طور انحصاری بدون شکست انجام شود. عامل باید عمل‌های مختلفی را امتحان کند و به‌تدریج وضعیت‌هایی که به نظر می‌رسد بهترین هستند را ترجیح دهد. در یک کار تصادفی، هر عمل باید بارها مورد آزمایش قرار گیرد تا برآورد قابل اعتماد از پاداش بدست بیاد. معضل پویش-انتفاع توسط ریاضی‌دانان چندین دهه به شدت مورد مطالعه قرار گرفته‌است، با این حال همچنان حل نشده‌است. در حال حاضر ما توجه داشته باشیم که کل مسئله تعادل پویش و انتفاع حتی در یادگیری تحت نظارت و بدون نظارت، حداقل در خالص‌ترین شکل این نمونه‌ها، ایجاد نمی‌شود.

یکی دیگر از ویژگی‌های مهم یادگیری تقویتی این است که کل مسئله‌ی عامل هدفمند را در تعامل با یک محیط غیرقطعی در نظر می گیرد. این در مقایسه با بسیاری از رویکردهایی است که در معرض مشکلات بدون در نظر گرفتن اینکه چگونه ممکن است به تصویر بزرگتر متوسل شوند، در نظر گرفته شود. به عنوان مثال، ما اشاره کردیم که بسیاری از تحقیقات یادگیری ماشین مربوط به یادگیری تحت نظارت است و به صراحت مشخص نیست که چطور چنین توانایی در نهایت مفید خواهد بود. محققان، دیگر نظریه‌های برنامه ریزی با اهداف عمومی را توسعه داده اند، اما این‌ها بدون درنظر گرفتن نقش برنامه ریزی در تصمیم گیری در زمان واقعی، و یا مسئله ای که در آن مدل‌های پیش بینی شده برای برنامه ریزی ضروری است. اگر چه این رویکردها نتایج بسیار خوبی کسب کرده اند، اما تمرکز آنها بر زیرمجموعه‌های جداگانه محدودیت قابل توجهی دارد.

همه عوامل آموزش تقویتی دارای اهداف صریح هستند، می توانند جنبه‌های محیط‌هایشان را درک کنند و می توانند اقدامات را برای تاثیر گذاری بر محیط آنها انتخاب کنند. علاوه بر این، معمولا از ابتدا فرض می‌شود که عامل باید با وجود عدم اطمینان در مورد محیطی که با آن مواجه است کار کند. هنگامی که یادگیری تقویتی شامل برنامه ریزی است، باید ارتباط بین برنامه ریزی و انتخاب عمل در زمان واقعی و همچنین این مسئله که چگونه مدل‌های محیطی به دست می آید و بهبود می یابند، مورد توجه قرار گیرد. وقتی یادگیری تقویتی شامل یادگیری با نظارت می‌شود، به دلایل خاصی که تعیین می‌کند که چه ویژگی‌هایی مهم هستند و چه چیزی نمی باشد. برای یادگیری، زیرمجموعه‌های مهم باید جدا مورد مطالعه

قرار گیرند. آنها باید زیرمجموعه ای باشند که در واحدهای کامل، تعاملی و هدفمند نقش واضح داشته باشند.

با یک عامل کامل، تعاملی و هدفمند ما همیشه چیزی را مانند یک ربات نمی‌بینیم. این‌ها نمونه‌های واضح هستند، اما یک عامل کامل، تعاملی و هدفمند می تواند جزء یک رفتار سیستم بزرگتر نیز باشد. در این حالت، عامل به طور مستقیم با بقیه سیستم بزرگتر ارتباط برقرار می‌کند و به طور غیر مستقیم با محیط سیستم بزرگتر ارتباط برقرار می‌کند. یک مثال ساده یک عامل است که سطح شارژ باتری ربات را نظارت می‌کند و دستورات را به معماری کنترل ربات می‌فرستد. این محیط عامل، بقیه ربات به‌همراه محیط ربات است. باید فراتر از نمونه‌های واضح عامل‌ها و محیط‌های آن‌ها را در نظر گرفت تا ربات کلیه چارچوب یادگیری تقویتی را درک کند.

یکی از جنبه‌های هیجان انگیز یادگیری تقویتی مدرن، تعامل حقیقی و مثبت آن با سایر رشته‌های مهندسی و علمی است. یادگیری تقویتی بخشی از یک دهه طولانی در زمینه هوش مصنوعی و یادگیری ماشین به منظور ادغام بیشتر با آمار، بهینه سازی و دیگر موضوعات ریاضی است. به طور مشخص‌تر، یادگیری تقویتی نیز به شدت با روانشناسی و علوم اعصاب با مزایای قابل توجهی در هر دو تعامل داشته است. از تمامی انواع یادگیری ماشین، یادگیری تقویتی نزدیک‌ترین نوع یادگیری است که انسان و دیگر حیوانات انجام می دهند، و بسیاری از الگوریتم‌های اصلی یادگیری تقویتی در اصل توسط سیستم‌های یادگیری بیولوژیکی الهام شدند. یادگیری تقویتی با مدل روانشناختی یادگیری حیوانات، که با برخی از داده های تجربی سازگار است و با یک مدل تاثیرگذار از بخش های سیستم پاداش مغز، همخوانی دارد.

در نهایت، یادگیری تقویتی نیز بخشی از روند بزرگتر در هوش مصنوعی به سمت اصول کلی ساده است. از اواخر دهه ۱۹۶۰، بسیاری از محققان هوش مصنوعی معتقد بودند که هیچ اصول کلی برای کشف وجود ندارد، بلکه این اطلاعات به دلیل داشتن تعداد وسیعی از ترفندها، روش‌ها و پویش‌های خاص است. گاهی اوقات گفته می‌شود که اگر ما فقط بتوانیم حقایق مربوط به یک ماشین را به‌دست بیاوریم، آنگاه ماشین هوشمند خواهد شد. روش‌های مبتنی بر اصول کلی مانند جستجوی یا یادگیری، به عنوان "روشهای ضعیف" شناخته شده اند، در حالی که آن‌هایی که بر اساس دانش خاص هستند، "روش‌های قوی" نامیده می شوند. امروزه این دیدگاه رایج است اما غالب نیست. تلاش‌های بسیار کمی برای جستجوی اصول عمومی به منظور نتیجه گیری بود که هیچ کدام از آنها وجود نداشت. هوش مصنوعی مدرن اکنون شامل تحقیقات زیادی به دنبال اصول عمومی یادگیری، جستجو و تصمیم‌گیری است.

تحقیقات یادگیری تقویتی قطعا بخشی از چرخش به سمت هوش مصنوعی با اصول ساده‌تر و کمتر است.

فصل ۲

انواع روش‌های یادگیری تقویتی

۱.۲ یادگیری تقویتی خنثی

ابتدا با ساده‌ترین موضوع آغاز می‌کنیم و آن عبارت است از عامل یادگیری خنثی که از یک توصیف حالت بنیان در یک محیط کاملا مشاهده پذیر استفاده می‌کند. در روش یادگیری خنثی خط مشی عامل ثابت و تغییرناپذیر است. هدف آن به سادگی عبارت است از یادگیری میزان خوب بودن یک خط مشی و به بیان دیگر یادگیری تابع سودمندی هدف اصلی عامل. وظیفه ی یادگیری خنثی بسیار مشابه وظیفه‌ی ارزیابی خط مشی یا همان سیاست گذاری است. تفاوت عمده آن است که عامل یادگیر خنثی از الگوی انتقال یعنی: $P\left(s' \middle| s, a\right)$که احتمال رسیدن به حالت s' را از حالت s پس از انجام عملیات a توصیف می‌کند، ناآگاه است و نیز اطلاعاتی از **تابع مشوق** $R(s)$ که مشوق هر حالت را توصیف می‌کند، نیز در اختیار ندارد. عامل با بهره گیری از خط مشی π، مجموعه‌ای از آزمون‌ها را در محیط خود به اجرا می‌گذارد.

۲.۲ یادگیری تقویتی فعال

یک عامل یادگیری خنثی دارای یک سیاست گذاری یا خط مشی ثابت است که به تعریف رفتار آن می‌پردازد. از سوی دیگر یک عامل هوشمند فعال باید پیرامون نوع عملیاتی که مایل به انجام آن است؛ تصمیم گیری کند. در ابتدای مطالب این بخش با عامل برنامه نویسی تطبیق پذیر پویا آغاز می‌کنیم و این مهم تحلیل می‌شود: چگونه عامل باید دچار تغییر شود تا این آزادی عمل جدید را مورد انتفاع قرار دهد. در آغاز لازم است عامل یک الگوی کامل

را به همراه پیامد احتمالاتی برای همه عملیاتی که انجام می‌دهد، یادگیری نماید. به عبارت دیگر لازم نیست چنین عامل هوشمندی الگوگذاری برای سیاست گذاری ثابت را انجام دهد. سازوکار ساده‌ی یادگیری استفاده شده از طریق الگوریتم PDA این منظور ساده را به خوبی انجام می‌دهد. در گام بعدی توجه به این واقعیت ضروری است که عامل بهره‌مند از گزینه‌های عملیات است. سودمندی‌های مورد نیاز شامل آن گروهی می‌شود که از سوی خط مشی بهینه تعریف شده است. آنها از معادلات بلمن تبعیت می‌کنند:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|\, s, a) U(s') \qquad (۱.۲)$$

با حل این معادلات تابع سودمندی از طریق بهره گیری از تکرار ارزش محاسبه می‌شود. سرانجام نکته ی بنیادی نوع کاری است که باید در هر گام انجام داد. پس از محاسبه ی یک تابع سودمندی که برای یک الگوی یادگیری شده بهینه است، عامل می‌تواند یک عملیات بهینه را با یک گام که نگاه رو به جلو دارد گزینش کرده تا سودمندی مورد انتظار را بیشینه نماید. به صورت جایگزین اگر از روش تکرار سیاست گذاری استفاده کند، آنگاه خط مشی بهینه در همان لحظه در دسترس او قرار می‌گیرد و بنابراین کافیست عملیاتی را که سیاست گذاری بهینه به آن توصیه کرده است، انجام دهد.

۳.۲ پویش

عامل، سودمندی‌های واقعی یا خط مشی بهینه ی واقعی را یادگیری نمی‌کند. بلکه در تلاش خود موفق به یافتن یک خط مشی می‌شود که مشوق ۱+ را در امتداد مسیر کوتاه‌تر قابل دسترسی می‌سازد. پس از انجام آزمون‌هایی با تغییرات اندک، عامل تنها همان سیاست گذاری را تعقیب می‌کند و هرگز اقدام به یادگیری سودمندی دیگر حالت‌ها نمی‌کند و هرگز هم موفق به یافتن مسیر بهینه نمی‌شود. این عامل هوشمند اصطلاحاً عامل حریص خوانده می‌شود. آزمون‌های تکراری نشان می‌دهد عامل حریص بسیار به ندرت به سمت خط مشی بهینه برای این محیط همگرا می‌شود و پاره ای از اوقات همگرایی آن به سوی خط مشی‌های واقعا دهشتناک است.

آنچه عامل حریص از نظر خود دور نگه داشته است آن است که عملیات می‌تواند سبب تامین مشوق‌های بیشتر بر مبنای الگوی یادگیری شده‌ی جاری باشد. آنها هم‌چنین می‌توانند سبب تسهیل یادگیری الگوی حقیقی از طریق تاثیرگذاری برداشت‌هایی شوند که از محیط دریافت می‌کنند. بهبود و توسعه ی الگو سبب می‌شود عامل مشوق‌های بیشتر و بالاتری را در

آینده دریافت نماید. بنابراین یک عامل باید تعادلی را بین انتفاع برای بیشینه سازی مشوق خود که در تخمین سودمندی جاری آن انعکاس یافته است و پویش که منافع دراز مدت عامل را تامین می‌کند، برقرار نماید. انتفاع تنها زیان توقف در یک بهینه محلی را همراه دارد. اگر عامل هرگز از دانش خود در عمل بهره گیری نکند، تنها استفاده از پویش به منظور بهبود دانش عامل نیز سودمند نخواهد بود. در دنیای واقعی عامل باید همواره درحال تصمیم‌گیری پیرامون ادامه ی حیات راحت خود و یا وارد شدن به مکان مجهول با این امید باشد که یک حیات بهتر و جدید را کشف خواهد کرد. هرچه درک و آگاهی بیشتر باشد، پویش به مراتب کمتری نیاز دارد. آیا می‌توان دقیق تر از فرآیند بالا عمل کرد؟ آیا یک خط مشی پویش بهینه وجود دارد؟ این پرسش‌ها به طور عمیق در زیر شاخه ی نظریه‌ی تصمیم‌سازی آماری قرار می گیرد. این گونه تصمیم‌سازی‌ها معروف به چالش‌های راهزنی می‌باشند که در ادامه‌ی همین بخش مورد مطالعه قرار می‌گیرند.

اگرچه حل مسائل راهزنی به صورت دقیق آن برای دستیابی یک روش پویش بهینه فوق العاده دشوار است اما انتخاب یک طرح معقول که سرانجام سبب رفتار بهینه از سوی عامل می‌شود، دور از انتظار نیست. در قالب دقیق خود هر طرح مشابه نیاز به حرص و آز در محدوده ای از پویش بی نهایت است. به سخن دیگر GLIE دارد. یک طرح GLIE بایستی هر عملیات در هر حالت را به تعداد مرتبه ی نامحدودی مورد آزمایش قرار دهد تا از ظهور احتمال محدود که یک عملیات بهینه آن را دور از نظر نگه‌داشته است پرهیز نماید. فراموش شدن عملیات بهینه به دلیل یک مجموعه از پیامدهای ناخوشایند غیر معمول است. یک عامل ADP که از چنین طرحی استفاده می‌کند، سرانجام موفق به یادگیری الگوی محیط واقعی خود می‌شود. یک طرح GLIE نیز باید سرانجام حریص شود تا عملیات عامل نسبت به الگوی یادگیری شده و در نتیجه الگوی حقیقی بهینه گردد.

۴.۲ تعمیم یادگیری تقویتی

تا کنون فرض شده است هر دو تابع سودمندی و Q که از سوی عامل یادگیری شده‌اند، در قالب جدول بندی شده با یک ارزش خروجی برای هر چند ورودی هستند. چنین رویکردی برای فضاهای حالت کوچک عملکردی معقول و قابل قبول دارند، اما همگرایی و به ویژه برای ADP زمان لازم برای هر تکرار فرآیند به سرعت با بزرگتر شده فضا افزایش پیدا می‌کند. از سوی دیگر با روش‌های ADP تقریبی به دقت کنترل شده امکان تحلیل ۱۰۰۰۰ حالت و یا بیشتر وجود دارد. این مهم برای محیط‌های دوبعدی مارپیچ گونه (ماز) کافی است، اما در دنیاهای به مراتب واقعی تر کاملا مردود است. بازی شطرنج زیرمجموعه بسیار کوچکی

از دنیای واقعی است. در عین حال فضای آن شامل 10^{40} حالت است. تصور این موضوع کاملا نامعقول است که یک عامل باید همه ی این حالت‌ها را آن هم به تعداد مرتبه‌های زیاد مورد بازدید قرار دهد تا چگونگی انجام بازی را یاد بگیرد. یک روش برخورد با چنین مسائلی استفاده از تابع تقریب سازی است که به زبان ساده به مفهوم بهره‌گیری از گونه ای توصیف برای تابع Q به غیر از روش جدول جستجو است. این توصیف از این جهت تقریبی شناخته می‌شود که ممکن است امکان توصیف تابع سودمندی و یا تابع Q واقعی در قالب گزینش شده میسر نباشد. به عنوان مثال یک تابع ارزیابی برای بازی شطرنج در قالب یک تابع خطی توزین شده از مجموعه‌ای از ویژگی‌ها یا توابع پایه توصیف شده اند. معادله ی ذیل این مورد را به شرح ذیل توصیف می‌کند:

$$\hat{U}_\theta(s) = \theta_1 f_1(s) + \theta_2 f_2(s) + \ldots + \theta_\mathrm{n} f_\mathrm{n}(s) \qquad (۲.۲)$$

یک الگوریتم یادگیری تقویتی می‌تواند ارزش‌هایی را برای پارامترهای: $\theta = \theta_1, \ldots, \theta_\mathrm{n}$ را به گونه ای یادگیری کند که تابع ارزیابی $\hat{U}_\theta$ تقریب سازی به تابع سودمندی واقعی گردد. در عوض تعداد 10^{40} ارزش در جدول، این تابع تقریب ساز از طریق تعداد $n = 20$ پارامتر مشخصه سازی می‌شود. این موضوع به مفهوم فشرده سازی بسیار قابل ملاحظه است. اگرچه تابع سودمندی واقعی برای شطرنج شناخته شده است، اما کسی باور نمی‌کند که آن را بتوان در قالب دقیقا تعداد ۲۰ عدد توصیف کرد. اگر تقریب سازی به اندازه ی کافی مطلوب باشد، عامل ممکن است هنوز هم بازی شطرنج را در حد عالی به انجام رساند. تابع تقریب ساز امکان توصیف عملی توابع سودمندی برای فضاهای حالت بسیار بزرگ را فراهم می‌کند. اما این فرآیند به نظر مفید نمی رسد. فشردگی حاصل از طریق از یک تابع تقریب ساز اجازه می‌دهد عامل یادگیر خود را از حالت‌های ملاقات شده به حالت‌هایی تعمیم دهد که آن‌ها را تاکنون ملاقات نکرده است. به عبارت دیگر مهم‌ترین جنبه ی تقریب سازی تابع، نیاز آن به فضای کمتر نمی‌باشد بلکه قدرت ان در تامین شرایط لازم برای تعمیم القائی روی حالت‌های ورودی است.

از سوی دیگر این ضعف هم وجود دارد که در فضای فرض انتخابی که تابع سودمندی واقعی را به خوبی تقریب سازی می‌کند، ممکن است هیچ تابعی پدیدار نگردد. مانند تمام یادگیری‌های استنتاجی همواره یک تعادل و توازن بین بزرگی و فضای فرض و زمان لازم برای یادگیری تابع وجود دارد. یک فضای فرضیه ی بزرگتر سبب افزایش این احتمال می‌شود که یک تقریب سازی مناسب می‌تواند شناسایی گردد اما به این مفهوم هم هست که احتمالا سبب تاخیر همگرایی می‌شود.

برای یادگیری تقویتی استفاده از الگوریتم یادگیری آنلاین به مراتب بهتر است زیرا پارامترها را پس از هر سنجش و آزمون به روزرسانی می‌کند. اگر $u_j(s)$ مجموع امتیاز مشاهده شده از حالت s به طرف بالا در آزمون شماره‌ی j باشد، آنگاه خطای تعریف شده عبارت است از نیمی از مجذور تفاضل بین کل پیش بینی شده و کل واقعی و به عبارت دیگر معادله ی ذیل تعریف می‌شود:

$$E_j(s) = \left(\hat{U}_\theta(s) - u_j(s)\right)^2 \Big/ 2 \qquad (۳.۲)$$

نرخ تغییر خطا نسبت به هر پارامتر یعنی θ_{i} برابر است با $\frac{\partial E_j}{\partial \theta_i}$ ، بنابراین حرکت پارامتر در جهت کاهش خطا سبب نیاز به تعریف معادله ی زیر می‌شود:

$$\theta_i \leftarrow \theta_i - \alpha \frac{\partial E_j(s)}{\partial \theta_i(s)} = \theta_i + \alpha \left(u_j(s) - \hat{U}_\theta(s)\right) \frac{\partial \hat{U}_\theta(s)}{\partial \theta_i} \qquad (۴.۲)$$

این معادله معروف به قانون ویدرو وهاف یا به بیان دیگر قانون دلتا برای کوچک ترین مربعات آنلاین است. برای تابع تقریب ساز خطی: $\hat{U}_\theta(s)$ در معادله ی زیر تعداد سه قانون به روز رسانی ساده ی ذیل را می توان اظهار کرد:

$$\begin{aligned}
\theta_0 &\leftarrow \theta_0 + \alpha \left(u_j(s) - \hat{U}_\theta(s)\right), \\
\theta_1 &\leftarrow \theta_1 + \alpha \left(u_j(s) - \hat{U}_\theta(s)\right) x, \\
\theta_2 &\leftarrow \theta_2 + \alpha \left(u_j(s) - \hat{U}_\theta(s)\right) y.
\end{aligned} \qquad (۵.۲)$$

لازم به توضیح است که تغییر پارامترهای θ در واکنش به یک انتقال مشاهده شده بین دو حالت سبب تغییر ارزش‌ها برای هر حالت دیگر نیز می‌شود. که یعنی تابع تقریب سازی اجازه ی تعمیم تجارب را به عامل یادگیر تقویتی می‌دهد. انتظار می‌رود عامل یادگیری خود را در صورتی تسریع نماید که از یک تابع تقریب ساز مشروط بر آنکه فضای فرضیه بسیار بزرگ است، استفاده نماید. اما این فضا می‌تواند شامل برخی از توابع باشد که بهره‌مند از یک برازندگی مناسب نسبت به تابع سودمندی واقعی هستند. برای تابع تقریب ساز خطی آنچه اهمیت دارد، خطی بودن تابع در پارامترها است. ویژگی حالت‌ها می‌تواند شامل متغیرهای حالت از توابع غیرخطی اختیاری باشد. بنابراین عبارتی مانند آنچه در ذیل آمده است را

می‌توان اضافه نمود تا اندازه گیری فاصله از هدف را به انجام رساند:

$$\theta_3 f_3(x,y) = \theta_3 \sqrt{(x-x_g)^2 + (y-y_g)^2} \qquad (۶.۲)$$

۵.۲ جستجوی خط مشی (سیاست‌گذاری)

این جستجو به این مفهوم است که سیاست گذاری تا زمانی که کارایی آن بهینه شود در شرایط سرگرمی نگهداشت شده و سپس متوقف می‌گردد. ابتدا با خط مشی‌ها تحلیل را آغاز می‌کنیم. یادآوری این نکته ضرورت دارد که یک خط مشی π یک تابع است که حالت‌ها را به عملیات نگاشت می‌کند. آنچه در اصل اهمیت دارد توصیف‌های عامل بندی شده یا همان پارامتریک از π می باشد که تعداد بسیار اندکی پارامتر در مقایسه با تعداد حالت‌ها در فضای حالت را شامل می‌شود. به عنوان مثال می‌توان π را از طریق مجموعه ای از توابع Q که پارامترسازی شده است، توصیف کرد. بدیهی است برای هر عملیات یک تابع تعریف می‌شود و عملیات انجام شده بر اساس معادله زیر بهره‌مند از بالاترین ارزش پیش بینی شده است.

$$\pi(s) = \max_a \hat{Q}_\theta(s,a). \qquad (۷.۲)$$

هر تابع Q می‌تواند یک تابع خطی از پارامترهای θ و یا یک تابع غیرخطی مانند یک شبکه عصبی باشد. در این صورت جستجوی سیاست گذاری، خود را با پارامترهای θ به منظور بهبود خط مشی سازگار می‌کند. بدیهی است اگر خط مشی از طریق توابع Q توصیف شده باشد، آنگاه نتیجه ی جستجوی سیاست گذاری یک فرایند یادگیری از توابع Q خواهد بود. این فرآیند مشابه یادگیری Q نیست. در یادگیری Q به کمک توابع تقریب‌سازی، الگوریتم ارزش از θ را یافت می‌کند که نزدیک به تابع بهینه‌ی Q است. از طرف دیگر جستجوی خط مشی ارزشی از θ را پیدا می‌کند که نتیجه ی حاصل از آن عملکرد خوب و مطلوب است. ارزش‌های بدست آمده از طریق این دو روش ممکن است بسیار متفاوت باشند.

یک اشکال توصیف خط مشی آن است که سیاست گذاری یک تابع ناپیوسته از پارامترها، در لحظه‌ای است که عملیات حالت گسسته اختیار می‌کند. برای یک فضای عملیات پیوسته، خط مشی می‌تواند یک تابع هموار از پارامترها باشد. به سخن دیگر ارزش‌هایی از θ وجود دارد که یک تغییر بسیار جزئی در θ را سبب می‌شود تا خط مشی از یک عملیات به عملیاتی دیگر تغییر موضع دهد. این به معنی آن است که ارزش سیاست گذاری ممکن است به صورت پیوسته تغییر پیدا کند و این موضوع سبب می‌شود جستجوی تدریجی را با دشواری روبه‌رو نماید. به همین دلیل روش‌های جستجوی خط مشی در اغلب موارد از یک توصیف خط

مشی تصادفی بهره می‌گیرند که احتمال گزینش عملیات a در حالت s را مشخص می‌سازد. یک توصیف پرطرفدار اصطلاحا تابع بیشینه ی نرم است که در قالب معادله ی زیر تعریف می‌شود:

$$\pi_\theta(s) = e^{\hat{Q}_\theta(s,a)} \Big/ \sum_{a'} e^{\hat{Q}_\theta(s,a)}. \qquad (۸.۲)$$

تابع بیشینه ی نرم درصورتیکه یک عملیات به مراتب بهتر از عملیات دیگر باشد، تقریبا سبب قطعیت می‌شود. اما همواره یک تابع دیفرانسیل از θ را نیز سبب می‌شود و بنابراین ارزش خط مشی که وابسته به رویکردی پیوسته نسبت به احتمالات گزینش عملیات است باعث یک تابع دیفرانسیل از θ می‌شود. تابع بیشینه‌ی نرم تقریب یک تابع منطقی برای متغیرهای چندگانه است.

فصل ۳

مثال تیک-تک-تو

برای نشان دادن ایده کلی یادگیری تقویتی و مقایسه آن با رویکردهای دیگر، یک مثال واحد را در جزئیات بیشتر در نظر می‌گیریم. بازی تیک-تک-تو [۱] را در نظر بگیرید. دو بازیکن به طور مداوم در صفحه‌ی سه در سه بازی می‌کنند. یک بازیکن X و دیگری O بازی می‌کند تا یک بازیکن با قرار دادن سه علامت در یک ردیف افقی، عمودی یا مورب، برنده شود. از آنجایی که یک بازیکن ماهر می تواند جوری بازی کند تا هرگز نبازد، فرض می‌کنیم که ما در برابر یک بازیکن غیرحرفه‌ای بازی می‌کنیم، که بازی گاه گاهی نادرست است و ما را به پیروزی می‌رساند. برای لحظه‌ای، در واقع، به ما اجازه می‌دهد که قرعه کشی‌ها و زیان‌ها به همان اندازه برای ما بد باشد. چگونه می‌توان یک بازیکنی ایجاد کرد که نقص در بازی حریف خود را پیدا کند و یاد بگیرد تا شانس خود را برای برنده شدن به حداکثر برساند؟

اگر چه این یک مشکل ساده است، اما از طریق تکنیک‌های کلاسیک نمی توان به راحتی حل و فصل کرد. به عنوان مثال، راه حل "مینیمکس [۲] " کلاسیک از نظریه بازی در اینجا درست نیست، زیرا به یک روش خاص از بازی توسط حریف برمی‌گردد. به عنوان مثال، یک بازیکن مینیمکس هرگز به حالت بازی ای که از آن می‌تواند از دست برود، نمی‌رسد، حتی اگر در حقیقت همیشه از آن حالت به دلیل بازی نادرست توسط حریف برنده شده باشد.

روش‌های بهینه سازی کلاسیک برای مشکلات تصمیم‌گیری پیوسته، مانند برنامه نویسی دینامیکی، می‌تواند راه حل مطلوب برای هر حریف باشد، اما به عنوان ورودی نیاز به مشخصات کامل آن حریف را دارد. از جمله احتمال‌هایی که حریف هر حرکت را در هر حالت صفحه بازی انجام دهد. بگذارید فرض کنیم که این اطلاعات برای پیشینه این مشکل در دسترس

[1] TIC-TAC-TOE
[2] minimax

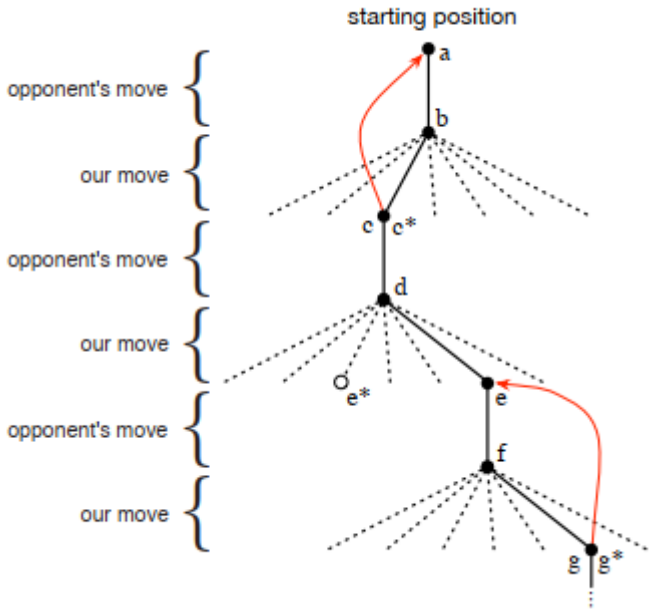
نیست، زیرا اکثریت قریب به اتفاق مشکلات عملی است. از سوی دیگر، این اطلاعات را می‌توان از تجربه، در این مورد با بازی‌های بسیاری در برابر حریف، برآورد کرد. بهترین کاری که می‌توان در این مسئله انجام داد این است که بازیکن برای اولین بار یک مدل رفتار حریف را یاد بگیرد تا به سطح اطمینان بیشتری برسد و سپس برنامه ریزی پویا را برای محاسبه یک راه‌حل بهینه با توجه به مدل حقیقی تقریبی اعمال کند. در نهایت این تفاوت چندانی با برخی از روش‌های یادگیری تقویتی ندارد.

یک روش تکاملی برای این مشکل این است که به طور مستقیم در فضای سیاست‌های ممکن برای یکی با احتمال برنده شدن بالا، در برابر حریف جستجو می‌کند. در اینجا یک سیاست، یک قاعده است که به بازیکن می‌گوید که چه چیزی برای هر وضعیت ممکن از X و O در یک صفحه‌ی سه‌درسه ایفا می‌کند. برای هر سیاست در نظر گرفته شده، برآورد احتمال پیروزی آن با بازی تعدادی از بازی‌ها در برابر حریف به دست می‌آید. این ارزیابی پس از آن تعیین می‌کند که کدام سیاست یا سیاست‌ها در آینده مورد توجه قرار گیرد. یک روش تکاملی معمولی می‌تواند در فضای سیاست مورد توجه قرار بگیرد و به دنبال آن تولید و ارزیابی سیاست‌ها در تلاش برای به‌دست آوردن پیشرفت‌های قدم‌به‌قدم است. یا شاید یک الگوریتم سبک ژنتیکی می‌تواند مورد استفاده قرار گیرد که تعداد سیاست‌ها را حفظ و ارزیابی می‌کند. به معنای واقعی کلمه صدها روش مختلف بهینه سازی می‌تواند مورد استفاده قرار گیرد.

در اینجا مسئله این است که چگونه با استفاده از یک تابع ارزش، مشکل تیک-تک-تو حل خواهد شد. اول ما یک جدول برای هر حالت ممکن بازی تنظیم می‌کنیم. هر عدد آخرین برآورد احتمالی برنده شدن از آن حالت خواهد بود. ما این برآورد را به عنوان ارزش حالت تلقی می‌کنیم و کل جدول تابع ارزش آموخته شده است. ما این برآورد را به عنوان ارزش حالت تلقی می‌کنیم و کل جدول تابع ارزش آموخته شده است. حالت A مقدار بالاتری نسبت به حالت B دارد یا بهتر از حالت B در نظر گرفته می‌شود، اگر برآورد فعلی احتمال برداشت ما از A بالاتر از B باشد. با فرض اینکه ما همیشه X را بازی می‌کنیم، برای همه حالت‌هایی که با سه X در یک ردیف هستند، احتمال برنده شدن ۱ است، زیرا ما قبلا برنده شده‌ایم. به طور مشابه، برای تمام حالت‌هایی که سه O در یک ردیف دارند و یا پر شده اند، احتمال صحیح ۰ است، زیرا ما نمی توانیم از آنها برآییم. مقادیر اولیه همه وضعیت‌های دیگر را به ۵.۰ می رسانیم، که نشان دهنده یک حدس است که ما نیمی از اوقات احتمال برنده شدن داریم.

ما پس از آن، بازی‌های بسیاری را در مقابل حریف بازی می‌کنیم. برای انتخاب حرکاتمان،

حالت‌هایی را که از هر یک از حرکت‌های ممکن (یک حرکت برای هر فضای خالی در صفحه) حاصل می‌شود بررسی می‌کنیم و مقادیر فعلی آنها را در جدول می‌بینیم. اغلب اوقات ما به طور مرتب حرکت می‌کنیم و حرکتی با بیشترین امتیاز و بالاترین احتمال برنده شدن را انتخاب می‌کنیم. گاهی اوقات ما به طور تصادفی از میان حرکت‌های دیگر انتخاب می‌کنیم. اینها حرکت‌های پویشی نامیده می‌شوند؛ زیرا باعث می‌شود بعضی از وضعیت‌ها تجربه کنند که ما هرگز نمی‌توانیم آن‌ها را ببینیم. یک دنباله ای از حرکات ساخته شده و در نظر گرفته شده در طول یک بازی می‌توان به شکل ۱.۳ نمودار نمود.



شکل ۱.۳: یک سکانس از حرکت Tic-Tac-Toe

هنگام بازی، ارزش‌های وضعیت‌هایی که در طول بازی خودمان پیدا کرده‌ایم را تغییر می‌دهیم. ما تلاش می‌کنیم تا برآوردهای دقیق تر از احتمال برنده شدن آنها را ارزیابی کنیم. برای انجام این کار، پس از هر حرکت به حالت قبل از حرکت، وضعیت پس از حرکت فلش در شکل ۱.۳ نشان داده شده است. به طور دقیق‌تر، مقدار فعلی وضعیت قبلی به روز می‌شود تا نسبت به مقدار وضعیت بعدی نزدیک‌تر شود. این را می‌توان با جابجایی مقدار وضعیت قبلی مقداری از راه به سمت ارزش وضعیت بعدی انجام داد. اگر ما قبل از حرکت به وضعیت نشان دهیم، و S_{t+1} وضعیت پس از حرکت باشد، سپس برآورد به ارزش برآورد شده S_t ،

نشان دهنده $V(S_t)$ می‌تواند به صورت زیر نوشته شود:

$$V(S_t) \leftarrow V(S_t) + \alpha [V(S_{t+1}) - V(S_t)], \qquad (۱.۳)$$

آلفا یک مقدار کوچک مثبتی به نام پارامتر گام اندازه است که بر میزان یادگیری تأثیر می‌گذارد. این قانون به روزرسانی نمونه ای از روش زمانبندی متفاوت آموزش است، زیرا تغییرات آن بر اساس یک تفاوت است. $V(S_{t+1})$ و $V(S_t)$ بین دو بار متوالی تخمین زده می‌شود. روشی که در فوق توضیح داده شد، در این کار بسیار خوب عمل می‌کند. به عنوان مثال، اگر پارامتر گام اندازه به درستی در طول زمان کاهش یابد، این روش برای هر حریف ثابت، برای احتمال واقعی برنده شدن از هر حالت، به‌دست می‌آید که به وسیله بازیکن ما بهینه شده است. علاوه بر این، حرکت‌هایی که بعدا انجام می‌شود (به غیر از حرکت پویشی) در واقع حرکت مطلوب در برابر این حریف است. به عبارت دیگر، این روش به یک سیاست مطلوب برای بازی در برابر این حریف همسو می‌شود. اگر پارامتر گام اندازه در طول زمان به صفر برسد، پس این بازیکن نیز به خوبی با حریف بازی می‌کند و به آرامی راه بازی خود را تغییر می‌دهد.

این مثال تفاوت‌های بین روش‌های تکاملی و روش‌هایی که توابع ارزش را یاد می‌گیرند را نشان می‌دهد. برای ارزیابی یک سیاست، یک روش تکاملی این است که سیاست را ثابت نگه دارد و بسیاری از بازی‌ها را در برابر حریف بازی می‌کند یا بازی‌های بسیاری را با استفاده از یک مدل از حریف شبیه‌سازی می‌کند. فراوانی برنده‌ها برآورد بی رویه احتمال برنده شدن با آن سیاست را می‌دهد و می‌تواند برای هدایت انتخاب سیاست بعدی استفاده شود. اما هر تغییر سیاست تنها پس از بسیاری از بازی‌ها ساخته می‌شود و تنها نتیجه نهایی هر بازی مورد استفاده قرار می‌گیرد و اتفاقی در طول بازی نادیده گرفته می‌شود. به عنوان مثال، اگر بازیکن برنده شود، پس از رفتار خود در بازی، مستقل از اینکه چگونه حرکات خاص ممکن است برای پیروزی مهم باشد، به او اعتبار داده می‌شود. اعتبار حتی به حرکت‌هایی که هرگز اتفاق نیافتاده است داده می‌شود! در مقابل، روش‌های اعتبارسنجی اجازه می‌دهد که وضعیت‌های فردی مورد ارزیابی قرار گیرند. در نهایت، روش‌های تابع تکاملی و تابع ارزش هر دو فضای سیاست‌ها را جستجو می‌کنند، اما یادگیری یک تابع ارزش، از اطلاعات موجود و مفید در طول بازی استفاده می‌کند.

این مثال ساده برخی از ویژگی‌های کلیدی روش یادگیری تقویت را نشان می‌دهد. اول، تاکید بر یادگیری در تعامل با یک محیط با یک بازیکن حریف وجود دارد. دوم، یک هدف مشخص وجود دارد و رفتار صحیح، نیازمند برنامه‌ریزی یا پیش‌بینی است که تأثیرات تاخیری در انتخاب افراد را در نظر می‌گیرد. برای مثال، بازیکن یادگیری تقویتی ساده یاد می‌گیرد تا

تله‌های چند حرکتی را برای یک حریف کوتاه‌مدت طراحی کند. این یک ویژگی قابل توجه از راه حل یادگیری تقویتی است که می‌تواند بدون استفاده از یک مدل از حریف و بدون انجام جستجوی صریح در مورد توالی‌های احتمالی وضعیت‌ها و اقدامات آینده، به اثرات برنامه‌ریزی و پیشروی دست یابد.

در حالی که این مثال برخی از ویژگی‌های کلیدی یادگیری تقویتی را نشان می‌دهد، این بسیار ساده است که ممکن است این نکته به وجود بیاید که یادگیری تقویتی محدودتر از آنچه هست باشد. اگرچه تیک-تک-تو یک بازی دو نفره است، یادگیری تقویتی نیز در مواردی که هیچ حریف خارجی وجود ندارد، به کار می‌رود. یادگیری تقویتی نیز به مشکلات محدود نمی‌شود، در حالی که رفتار در قسمت‌های جداگانه، مانند بازی‌های جداگانه تیک-تک-تو با پاداش تنها در انتهای هر قسمت، تجزیه می‌شود. این فقط زمانی کاربرد دارد که رفتار به طور نامحدود ادامه می یابد و زمانی که پاداش از مقادیر مختلف می‌تواند در هر زمان دریافت شود. همچنین یادگیری تقویتی برای مسائلی که حتی به مرحله‌های مجزا مانند بازی‌های تیک-تک-تو تجزیه نشده اند، قابل استفاده است. اصول کلی برای مشکلات متداول هم کاربرد دارند، گرچه نظریه پیچیده‌تر می‌شود و ما از این روش مقدماتی حذف می‌کنیم.

تیک-تک-تو دارای مجموعه ای نسبتا کوچک و محدود است، در حالی که یادگیری تقویتی می تواند مورد استفاده قرار گیرد زمانی که مجموعه‌ی وضعیت‌ها بسیار بزرگ و یا حتی بی نهایت است. به عنوان مثال، گری تسارو [۳] (۱۹۹۲-۱۹۹۵) الگوریتمی که در بالا توضیح داده شده با یک شبکه عصبی مصنوعی برای یادگیری بازی تخته نرد ترکیب کرد که حدود ۲۰ وضعیت دارد. با این وضعیت‌های زیاد، غیرممکن است بیشتر آنها را تجربه کند. شبکه عصبی مصنوعی این برنامه را قادر می‌سازد تا تجربیات خود را به طور کلی تعمیم دهد، به طوری که در حالت‌های جدید، حرکت‌ها را براساس اطلاعات ذخیره شده از حالت‌های مشابه که در گذشته صورت گرفته است، بر اساس شبکه تعیین می‌کند. اینکه چطور یک سیستم یادگیری تقویتی می‌تواند در مسائلی با چنین مجموعه ای از وضعیت‌های بزرگ کار کند، به شدت وابسته به این است که چگونه می‌تواند به درستی از تجربه گذشته تعمیم دهد که ما نیاز بیشتر به روش‌های یادگیری تحت نظارت با یادگیری تقویتی داریم. شبکه‌های عصبی مصنوعی و یادگیری عمیق تنها راه یا لزوما بهترین روش برای انجام این کار نیستند. اطلاعات پیشین می‌تواند در یادگیری تقویتی به شیوه‌های گوناگون ایجاب می‌کند که برای یادگیری مؤثر است. ما همچنین می‌توانیم به حالت واقعی در مثال تیک-تک-تو دسترسی پیدا کنیم، در حالی که یادگیری تقویتی نیز می‌تواند در زمانی که بخشی از

[3] Gerry Tesauro

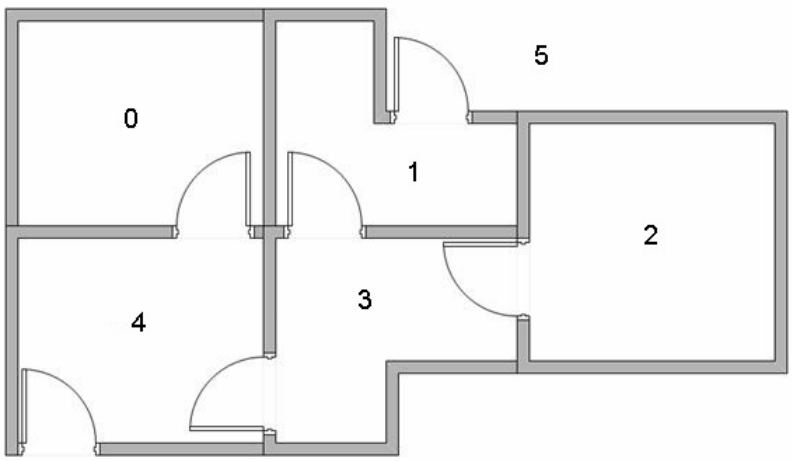
وضعیت پنهان شده باشد یا زمانی که حالت‌های مختلف برای یادگیرنده، یک شکل پدیدار می‌شوند. در نهایت، بازیکن تیک-تک-تو می‌تواند وضعیتی که حرکت‌های مجاز از یکدیگر نتیجه می‌گیرند را بداند. برای انجام این کار، باید یک مدل از بازی داشته باشیم که اجازه ی پیش‌بینی اینکه چگونه محیط آن در پاسخ به حرکت‌هایی که هرگز رخ نمی دهند، را داشته باشیم. بسیاری از مشکلات مانند این هستند. اما در برخی دیگر، حتی یک مدل کوتاه مدت، فاقد اثرات اقدامات آن است. یادگیری تقویتی می‌تواند در هر دو مورد استفاده شود. یک مدل ضروری نیست، اما مدلها اگر در دسترس باشند، می‌توانند یاد گرفته شوند و به راحتی می‌توانند مورد استفاده قرار بگیرند. از سوی دیگر، روش‌های یادگیری تقویتی وجود دارند که به هیچ وجه نیازی به تمام انواع مدل محیط ندارند. سیستم‌های بی‌مدل حتی نمی توانند در مورد چگونگی تغییر محیط آن‌ها در پاسخ به یک اقدام واحد فکر کنند. بازیکن تیک تاک-تو در این زمینه با توجه به حریف خود، بدون مدل است یعنی هیچ مدلی از حریف خود در همه انواع را ندارد.

از آنجایی که مدل‌ها باید منطقی و دقیق باشند، روش‌های بی‌مدل می‌توانند در مقایسه با روش‌های پیچیده محسنات بیشتری داشته باشند. زمانی که تنگنای واقعی در حل مشکل، ساختن یک مدل دقیق از محیط اطراف است، روش‌های بدون مدل نیز بلوک‌های مهم برای روش‌های مبتنی بر مدل هستند. یادگیری تقویتی می‌تواند در هر دو سطح بالا و پایین در یک سیستم استفاده شود. اگر چه بازیکن تیک-تک-تو فقط در مورد حرکات اصلی بازی می‌آموزد، هیچ چیز مانع تقویت یادگیری از کار در سطوح بالاتر نمی‌شود که هر کدام از اقدامات به خودی خود ممکن است کاربرد یک روش حل مسئله با جزئیات باشند. در سیستم‌های یادگیری سلسله مراتبی، یادگیری تقویتی می‌تواند به طور همزمان در سطوح مختلف کار کند.

فصل ۴

مثال q-learning

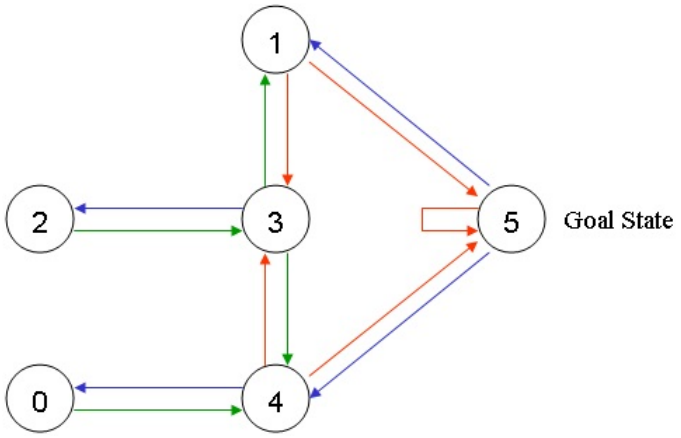
این آموزش مفهوم Q-learning را از طریق یک مثال ساده اما جامع عددی معرفی می‌کند. این مثال نمونه ای از آموزش بدون نظارت برای یادگیری در مورد یک محیط ناشناخته است. فرض کنید پنج اتاق در یک ساختمان که با درها بهم متصل شده‌اند به صورت شکل ۱.۴ داریم. شماره‌ی هر اتاق یکی از اعداد صفر تا چهار است. خارج از ساختمان را می‌توان به عنوان یک اتاق بزرگ با شماره‌ی پنج در نظر گرفت. توجه داشته باشید که درهای یک و چهار به درب شماره‌ی پنج متصل می‌شوند.



شکل ۱.۴: شمایی از وضعیت اتاق‌ها و درب‌ها

ما می‌توانیم مطابق شکل ۲.۴ اتاق‌ها را به صورت یک گراف معرفی کنیم، به‌طوریکه هر

اتاق به عنوان یک گره و هر درب به عنوان یک یال باشد.



شکل ۲.۴: گراف اتاق‌ها

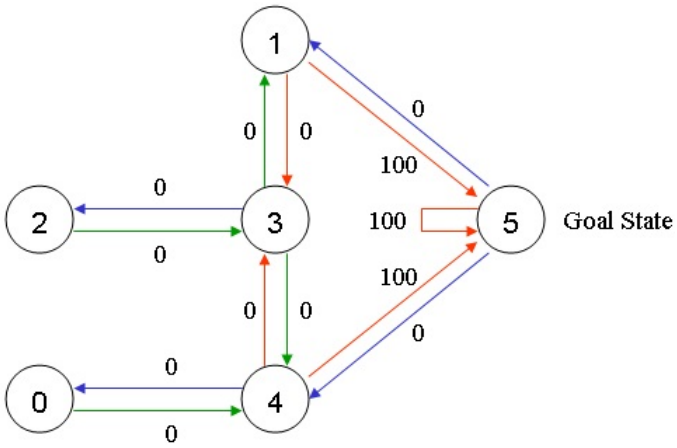
برای این مثال، ما می‌خواهیم عامل را در هر اتاق قرار دهیم، و عامل از آن اتاق به خارج از ساختمان برود و این اتاق هدف ما خواهد بود. به عبارت دیگر، اتاق هدف اتاق شماره پنج است. برای تعیین این اتاق به عنوان هدف، ما باید ارزش پاداش را برای هر درب (یعنی اتصال بین گره ها) مرتبط کنیم. درب هایی که مستقیما به هدف باز می‌شوند، پاداش ۱۰۰ دارند. درب‌های دیگر که مستقیما به اتاق هدف متصل نیستند، پاداش صفر دارند. درهای دو طرفه (صفر به چهار و چهار به صفر) دو فلش به هر اتاق اختصاص داده شده است. همانطور که در شکل ۳.۴ نشان داده شده است، هر فلش حاوی ارزش پاداش فوری است.

به طور واضح، اتاق پنج با پاداش ۱۰۰ به خود برمی‌گردد و تمام ارتباطات مستقیم با اتاق هدف، پاداش ۱۰۰ را می‌گیرند. در Q-learning هدف رسیدن به وضعیتی با بالاترین پاداش است، به طوریکه اگر عامل به هدف برسد، برای همیشه در آنجا باقی خواهد ماند. این نوع هدف "هدف جذب [1] " نامیده می‌شود.

عامل به عنوان روبات مجازی گنگ که می‌تواند از طریق تجربه یاد بگیرد، تصور می‌شود. عامل می‌تواند از یک اتاق به دیگری منتقل شود، اما هیچ دانشی از محیط نداشته باشد و نمی‌داند که کدام درب‌ها به محیط خارج منتهی می‌شوند.

برای مدل کردن خروج یک عامل از هر اتاق در ساختمان مطابق شکل ۴.۴ عمل می‌شود. عامل، ابتدا در اتاق دو قرار دارد وهدف ان رسیدن به خارج از خانه یعنی اتاق پنج است.

[1] absorbing goal



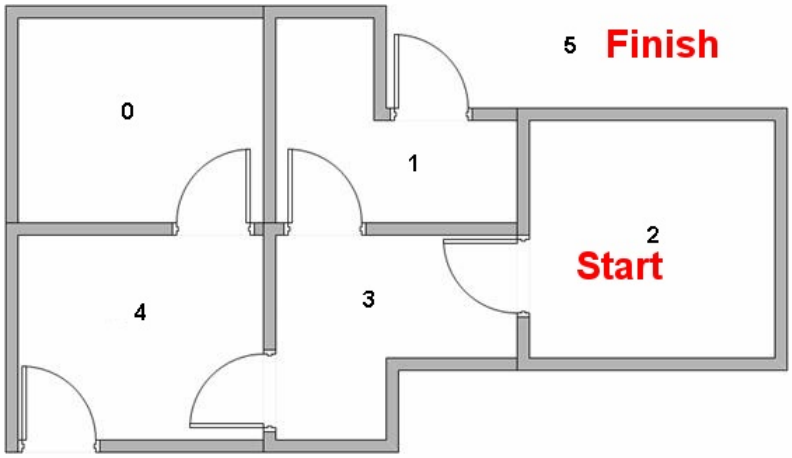
شکل ۳.۴: گراف اتاق‌ها به همراه ارزش آن‌ها

مجوعه‌ی اصلاحات در Q-learning شامل وضعیت و عمل است.

هر اتاق، از جمله محیط خارج را یک "وضعیت"، و حرکت عامل از یک اتاق به اتاق دیگری را "عمل" می‌نامیم. در گراف شکل ۵.۴ هر "وضعیت" به عنوان یک گره و هر "عمل" توسط یک فلش نشان داده شده است.

فرض می‌شود ابتدا عامل در وضعیت دو است. عامل از وضعیت دو می‌تواند به وضعیت سه برود زیرا وضعیت دو به وضعیت سه متصل است. عامل نمی‌تواند به طور مستقیم از وضعیت دو به وضعیت یک برود؛ زیرا اتاق دو دسترسی مستقیم به اتاق یک ندارد. عامل می‌تواند از وضعیت سه به وضعیت یک یا چهار یا به وضعیت دو برگردد. (به تمام فلش‌های مربوط به وضعیت سه نگاه کنید.) اگر عامل در وضعیت چهار باشد، سه حرکت ممکن برای او رفتن به وضعیت صفر، پنج یا سه است. اگر عامل در وضعیت یک باشد، می تواند به وضعیت پنج یا سه برگردد. از وضعیت صفر تنها می‌تواند به وضعیت چهار برگردد.

نمودار وضعیت و مقادیر پاداش فوری می‌تواند در ماتریس پاداش شکل ۱.۴ یعنی"ماتریس R" قرار داده‌شود.



شکل ۴.۴: اتاق شروع و پایان

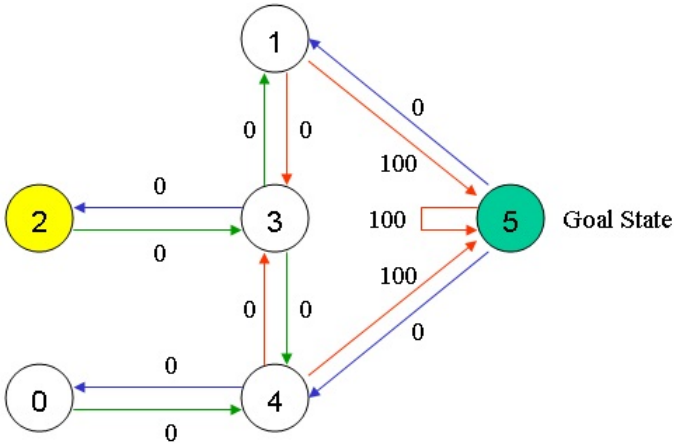
$$R = \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \quad (۱.۴)$$

حال یک ماتریس مشابه Q به مغز عامل اضافه می‌شود که نشانگر حافظه‌ی چیزی است که عامل از طریق تجربه آموخته است. ردیف ماتریس Q نشان دهنده وضعیت فعلی عامل و ستون آن نشان دهنده اقدامات احتمالی است که به حالت بعدی (یال بین گره ها) می‌انجامد.

هنگامی‌که عامل شروع به یادگیری می‌کند، ماتریس Q صفر است. در این مثال، برای ساده بودن توضیح، فرض می‌شود که تعداد وضعیت‌ها شناخته شده و به تعداد شش وضعیت است. اگر اطلاعی وجود نداشت که چند وضعیت درگیر هستند، ماتریس Q تنها با یک عنصر شروع می‌شد و با یافتن وضعیت‌های دیگر، به ردیف‌ها و ستون‌های آن افزوده می‌شد. قانون گذار در Q یک فرمول بسیار ساده است:

(۲.۴)

$$Q\left(state, action\right) = R\left(state, action\right) + \gamma \max[Q\left(nextstate, allactions\right)]$$



شکل ۵.۴: گراف اتاق‌ها به همراه گره شروع و پایان

با توجه به این فرمول، مقدار اختصاص داده شده به یک عنصر خاص از ماتریس Q ، برابر با مجموع مقدار متناظر در ماتریس R و پارامتر یادگیری گاما است، که در حداکثر مقدار Q برای تمام اقدامات ممکن در حالت بعدی ضرب می‌شود.

عامل از طریق تجربه و بدون معلم یاد می‌گیرد. عامل از وضعیتی به وضعیت دیگر تا زمانیکه به هدف برسد، کاوش می‌کند. ما هر کاوش را یک قسمت می‌نامیم. هر قسمت حرکت عامل از وضعیت اولیه به وضعیت هدف را شامل می‌شود. هر بار که عامل به وضعیت هدف برسد، برنامه به قسمت بعدی می‌رود.

الگوریتم Q-Learning به شرح زیر است:

۱. تنظیم پارامتر گاما و پاداش محیط در ماتریس R .

۲. صفر کردن ماتریس Q

۳. انتخاب یک حالت اولیه تصادفی برای هر قسمت:

آیا به وضعیت هدف رسیده است؟

از تمام اقدامات ممکن یکی را برای وضعیت فعلی انتخاب شود.

با استفاده از اقدامات ممکن، رفتن به وضعیت بعدی در نظر گرفته‌شود.

حداکثر مقدار Q برای این حالت بعدی را بر اساس تمام اقدامات ممکن به‌دست آورده‌شود.

محاسبه:

(۳.۴)

$$Q\left(state, action\right) = R\left(state, action\right) + \gamma \max[Q\left(nextstate, allactions\right)]$$

وضعیت بعدی را به عنوان وضعیت فعلی تنظیم شود.
پایان حلقه

الگوریتم بالا توسط عامل برای یادگیری از تجربه استفاده می‌شود. عامل محیط را بررسی می‌کند (ماتریس R) و درصورت وجود پاداش را دریافت می‌کند تا زمانیکه به وضعیت هدف برسد. هدف آموزش، ارتقاء مغز عامل است که توسط ماتریس Q ارائه شده است. آموزش بیشتر نتایج یک ماتریس Q را بهینه‌تر می کند. در این مورد، اگر ماتریس Q پیشرفت کرده باشد، به جای بررسی در اطراف و رفتن به عقب و بازگشت به همان اتاق، عامل سریع‌ترین مسیر را به وضعیت هدف پیدا می‌کند.

پارامتر گاما دارای دامنه‌ای از صفر تا یک است. اگر گاما نزدیک به صفر باشد، عامل تمایل دارد فقط پاداش فوری را در نظر بگیرد. اگر گاما نزدیک به یک باشد، عامل به پاداش‌های آینده توجه بیشتری خواهد کرد و مایل است پاداش را به تاخیر بیندازد.

برای استفاده از ماتریس Q عامل به سادگی دنباله‌ای از وضعیت‌ها را از وضعیت اولیه به وضعیت هدف دنبال می‌کند. الگوریتم اقداماتی را با بالاترین میزان پاداش ثبت شده در ماتریس Q برای حالت فعلی پیدا می‌کند:

الگوریتم برای استفاده از ماتریس Q :

۱. تنظیم حالت فعلی به حالت اولیه.
۲. پیدا کردن عمل با بالاترین مقدار Q از حالت فعلی.
۳. تنظیم حالت فعلی به حالت بعدی.
۴. تکرار مراحل دو و سه تا رسیدن وضعیت فعلی به وضعیت هدف.

الگوریتم بالا توالی وضعیت را از وضعیت اولیه به وضعیت هدف برمی‌گرداند.

برای درک اینکه چگونه الگوریتم Q-learning کار می‌کند، گام به گام از طریق چند قسمت آن را نشان داده شده است.

پارامتر یادگیری گاما به ۰.۸ تنظیم می‌شود و حرکت از اتاق یک آغاز می‌شود.
ماتریس Q را به صورت ماتریس ۴.۴ شروع می‌شود.

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{۴.۴}$$

با توجه به ردیف دوم (وضعیت یک) در ماتریس ۵.۴ ، دو حالت ممکن برای حالت فعلی وجود دارد: از وضعیت یک رفتن به وضعیت سه یا وضعیت پنج. رفتن به وضعیت پنج انتخاب می‌شود.

$$R = \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \tag{۵.۴}$$

حالا تصور شود اگر عامل در وضعیت پنج باشد، چه اتفاقی می‌افتد. با توجه به ردیف ششم ماتریس پاداش R (به عنوان مثال، وضعیت پنج) نگاه کنید. این سه عمل ممکن است: رفتن به وضعیت یک، چهار یا پنج.

(۶.۴)
$$Q\left(1,5\right) = R\left(1,5\right) + 0.8 * max[Q\left(5,1\right), Q\left(5,4\right), Q\left(5,5\right)] = 100 + 0.8 * 0 = 100$$

وضعیت بعدی، یعنی وضعیت پنج، در حال حاضر وضعیت فعلی است. از آنجا که پنج، وضعیت هدف است، یک قسمت به پایان رسیده‌است. مغز عامل ما در حال حاضر حاوی ماتریس به‌روز شده به صورت ۷.۴ است.

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{۷.۴}$$

برای قسمت بعدی، یک حالت اولیه تصادفی انتخاب می‌شود. این بار وضعیت سه به عنوان وضعیت اولیه انتخاب می‌شود.

با توجه به ردیف چهارم ماتریس R این اقدامات ممکن است: رفتن به وضعیت یک، دو یا چهار. با انتخاب تصادفی، بازگشت به وضعیت یک انتخاب می‌شود.

حال ما تصور می‌شود که عامل در وضعیت یک است. با توجه به ردیف دوم ماتریس پاداش R (به عنوان مثال وضعیت یک) این دو اقدام ممکن است: رفتن به وضعیت سه یا وضعیت پنج. سپس، ما مقدار Q محاسبه می‌شود.

$$[\mathrm{Q}(1,5) = \mathrm{R}(1,5) + 0.8 * \max[\mathrm{Q}(1,2), \mathrm{Q}(1,5)] = 80 \tag{۸.۴}$$

از ماتریس به روز شده Q از قسمت آخر استفاده کنید. ماتریس Q به صورت ماتریس شماره‌ی ۹.۴ است.

$$Q = \begin{bmatrix} ss0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{۹.۴}$$

وضعیت بعدی یعنی وضعیت یک، به وضعیت کنونی تبدیل می‌شود. حلقه‌ی درونی الگوریتم یادگیری Q تکرار می‌شود زیرا همانطور که در شکل
مشخص است، وضعیت یک، وضعیت هدف نیست.

بنابراین، با شروع حلقه جدید با وضعیت فعلی یک، دو عمل احتمالی وجود دارد: رفتن به وضعیت سه یا رفتن به وضعیت پنج. عمل انتخاب شده رفات به وضعیت پنج است.

حال فرض می‌شود که عامل در وضعیت پنج است. سه عمل ممکن است: رفتن به وضعیت یک یا چهار یا پنج. مقدار Q محاسبه می‌شود:

(۱۰.۴)
$$[\mathrm{Q}(1,5) = \mathrm{R}(1,5) + 0.8 * \max[\mathrm{Q}(5,1), \mathrm{Q}(5,4), \mathrm{Q}(5,5)] = 100$$

از آنجا که پنج وضعیت هدف است، این قسمت تمام می‌شود. مغز عامل در حال حاضر ماتریس به‌روز شده‌ی شماره‌ی
است.

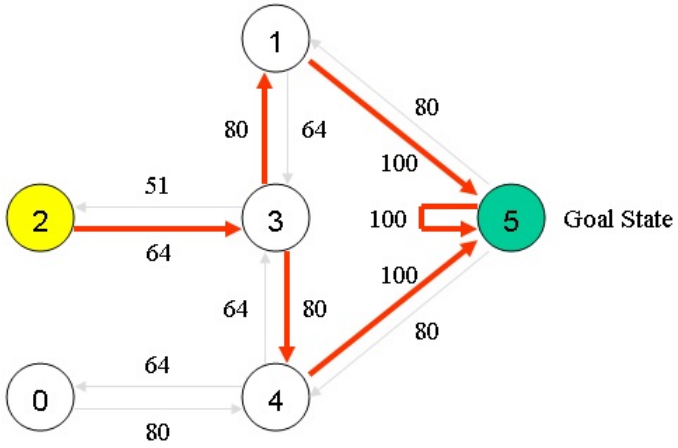
اگر عامل بیشتر از قسمت‌های دیگر یاد بگیرد، در نهایت به مقادیر همگرایی در ماتریس شماره‌ی
می‌رسد.

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \tag{۱۱.۴}$$

این ماتریس شماره‌ی ۱۱.۴ می‌تواند به وسیله تقسیم تمام مقادیر غیر صفر به بیشترین مقدار نرمالیزه شود و به صورت ماتریس شماره‌ی ۱۲.۴ به درصد تبدیل شود.

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \tag{۱۲.۴}$$

هنگامی که ماتریس Q به حالت همگرایی نزدیک می‌شود، یعنی عامل اکثر راه‌های رسیدن به وضعیت هدف را یاد گرفته‌است. برای یافتن بهترین دنباله‌های وضعیت لازم است که پیوندهای با بالاترین مقدار مطابق شکل ۶.۴ دنبال شوند.



شکل ۶.۴: بهترین دنباله وضعیت با شروع حرکت عامل از وضعیت دو

فصل ۵

جمع‌بندی

در این پروژه یادگیری تقویتی مورد مطالعه قرار گرفته است. یادگیری تقویتی یک رویکرد محاسباتی برای درک و خودکارسازی یادگیری و تصمیم گیری هدفمند است. یادگیری تقویتی، با تأکید بر یادگیری توسط یک عامل از تعامل مستقیم با محیط آن، بدون نیاز به نظارت یا داشتن مدل کاملی از محیط، از دیگر روش‌های محاسباتی متفاوت است. به نظر ما یادگیری تقویتی اولین زمینه برای جدی کردن مسائل محاسباتی است به طوری‌که هنگام یادگیری از تعامل با محیط زیست به منظور دستیابی به اهداف بلند مدت به‌وجود می‌آیند.

یادگیری تقویتی از چارچوب رسمی فرآیند تصمیم‌گیری مارکوف برای تعریف تعامل بین یک عامل یادگیری و محیط آن از نظر وضعیت‌ها، اعمال و پاداش استفاده می‌کند. این چارچوب یک روش ساده برای نشان دادن ویژگی های اساسی مسائل هوش مصنوعی است. این ویژگی ها عبارتند از علت و معلول، عدم قطعیت، نامشخص بودن و وجود اهداف صریح.

توابع ارزش برای جستجوی مؤثر در فضای سیاست ها مهم هستند. استفاده از توابع ارزش روش های تقویت یادگیری را از روش های تکاملی که مستقیما در فضای سیاست توسط ارزیابی کل سیاست ها راهنمایی می‌شوند، متمایز می‌کند.

طراحی عمومی عامل، نوع اطلاعاتی را که عامل باید یاد بگیرد را تعریف می‌کند. تعداد سه طرح بنیادی در رابطه با این موضوع تحلیل شدند. این سه عبارتند از: طرح‌های مبنی بر الگو که از یک الگوی p و یک تابع سودمندی U بهره می‌گیرد. طرح های براساس الگوی آزاد که از تابع عملیات-سودمندی Q استفاده می‌کند و سرانجام طرح‌های انعکاسی که از یک خط‌مشی یا سیاست‌گذاری π بهره می‌گیرند. یادگیری سودمندی از دو طریق میسر می‌شود:

۱- پیش‌بینی مستقیم سودمندی: که از کل امتیاز برای حرکت مشاهده شده برای

یادگیری سودمندی خود استفاده می‌کند.

۲- برنامه نویسی تطبیق‌پذیر پویا (ADP): که نسبت به یادگیری یک الگو و یک تابع مشوق از مشاهدات خود عمل می‌کند تا از طریق رویکرد تکرار ارزش و یا تکرار خط‌مشی به منظور رسیدن به سودمندی‌ها و یا خط‌مشی بهینه بهره‌گیری نماید. عامل ADP بهره‌گیری بهینه از محدودیت‌ها روی سودمندی حالت ها را در شرایطی انجام می‌دهد که این سودمندی‌ها در سراسر ساختار محیط تاثیرگذار هستند.

منابع

[1] Stuart Russell, Peter Norvig. ” Artificial Intelligence: A Modern Approach” 2009, Pearson; 3 edition
[2] Richard S.Sutton, Andrew G.Barto. ”Reinforcement learning an introduction” 2018, MIT press; second edition
[3] www.mnemstudio.org, accessed date: 12/7/2019

Abstract

Reinforcement learning is an important type of learning of the machine, in which the agent learns how to behave in the environment by taking actions and checking out the results. The main task of learning to enhance the use of incentives is to find an optimal or almost optimal policy for the environment. Direct training an evaluation function of the examples is very challenging. Instead, you can train the program to be able to understand your win or lose. Also, use this information to learn a valuation function that provides accurate and reasonable predictions and probability of a win in any given assumptions. There is a variety of learning enhancements. Neutral learning, active learning, and learning Q-learning of its types. Neutral learning is a policy of constant factors, and the main task of learning is to examine the utility of states. In active learning, the agent learns the type of operation to be done. The most important subject of active learning is exploring: An agent must gain enough of his environment to learn how to behave in the environment. Learning Q-learning is a form of non-model boost learning based on dynamic programming. Learning Q-learning is a reinforcement learning technique that learns a function that has a specific policy for doing different things in different situations. The project examines a variety of learning enhancement methods and finally, examines an example.

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

Revision of reinforcement learning methods

Zahra Ghasemi

Supervisor: Dr. Hedieh Sajedi

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

2018