مدرس: دکتر فاطمه منصوری

نیم‌سال اول ۱۴۰۲-۱۴۰۳

تهیه کننده : فاطمه حسین زاده

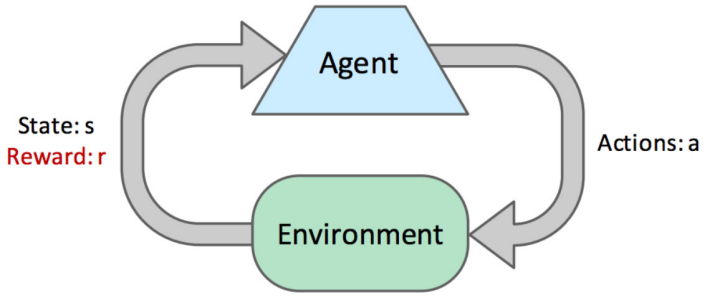
هوش مصنوعی

جزوه نهم

یادگیری تقویتی[1]

[یادگیری نیروافزوده یا یادگیری تقویتی یا یادگیری پاداش و تاوان یکی از گرایش‌های یادگیری ماشینی است که از روانشناسی رفتارگرایی الهام می‌گیرد. این روش بر رفتارهایی تمرکز دارد که ماشین باید برای بیشینه کردن پاداشش انجام دهد. این مسئله، با توجه به گستردگی‌اش، در زمینه‌های گوناگونی بررسی می‌شود. مانند: نظریه بازی‌ها، نظریه کنترل، تحقیق در عملیات، نظریه اطلاعات، سامانه چندعامله، هوش ازدحامی، آمار، الگوریتم ژنتیک، بهینه‌سازی بر مبنای شبیه‌سازی. یادگیری تقویتی در اقتصاد و نظریه بازیها بیشتر به بررسی تعادل‌های ایجاد شده تحت عقلانیت محدود می‌پردازد. در یادگیری ماشینی با توجه به این که بسیاری از الگوریتم‌های یادگیری نیروافزوده از تکنیک‌های برنامه‌نویسی پویا استفاده می‌کنند معمولاً مسئله تحت عنوان یک فرایند تصمیم‌گیری مارکف مدل می‌شود. تفاوت اصلی بین روش‌های سنتی و الگوریتم‌های یادگیری تقویتی این است که در یادگیری تقویتی نیازی به داشتن اطلاعات راجع به فرایند تصمیم‌گیری ندارد و این که این روش روی فرایندهای مارکف بسیار بزرگی کار می‌کند که روش‌های سنتی در آنجا ناکارآمدند.]

در یادداشت قبلی، درباره فرآیندهای تصمیم مارکوف بحث کردیم، که با استفاده از تکنیک‌هایی مانند تکرار ارزش و تکرار سیاست برای محاسبه مقادیر بهینه حالت‌ها و استخراج سیاست‌های بهینه، آن‌ها را حل کردیم. حل فرآیندهای تصمیم مارکوف نمونه‌ای از برنامه‌ریزی آفلاین است، که در آن عوامل، اطلاعات کاملی از تابع انتقال و تابع پاداش دارند، تمامی اطلاعاتی که برای پیش‌محاسبه اقدامات بهینه در جهان نیاز دارند، که توسط MDP کدگذاری شده‌اند، بدون اینکه هیچ اقدامی انجام دهند در دسترس آنهاست. در این یادداشت، درباره برنامه‌ریزی آنلاین بحث می‌کنیم که طی آن یک نماینده هیچ دانش قبلی از پاداش‌ها یا انتقال‌ها در جهان ندارد(که هنوز آنرا به عنوان یک MDP نشان می‌دهیم).در برنامه‌ریزی آنلاین، یک عامل باید جستجویی[2] را امتحان کند، که در طی آن اقداماتی را انجام می‌دهد و بازخوردی را در قالب حالت‌های جانشینی که وارد می‌شود و پاداش‌های مربوطه دریافت می‌کند، می‌گیرد .عامل از این بازخورد برای تخمین یک خط مشی بهینه، از طریق فرآیندی به نام یادگیری تقویتی قبل از استفاده از این خط مشی تخمین زده شده برای بهره برداری یا به حداکثر رساندن پاداش استفاده می کند.



با برخی از اصطلاحات اولیه شروع می‌کنیم .در هر مرحله زمانی در طول برنامه ریزی آنلاین، یک عامل در یک حالت s شروع می کند، سپس یک عمل a را انجام می دهد و در حالت جانشین s′ تمام می‌کند و پاداش r را دریافت می‌کند. هر (s,a, s′, r) به عنوان نمونه[3] شناخته می شود. اغلب، یک عامل به

[1] Reinforcement Learning
[2] exploration
[3] sample

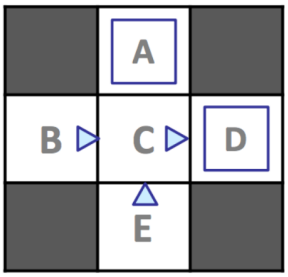
انجام اقدامات و جمع آوری نمونه های متوالی تا رسیدن به حالت پایانی ادامه می دهد. چنین مجموعه ای از نمونه ها به عنوان یک قسمت[4] شناخته می شوند. عوامل معمولاً در حین اکتشاف قسمت های زیادی را پشت سر می گذارند تا داده های کافی مورد نیاز برای یادگیری را جمع آوری کنند.

دو نوع یادگیری تقویتی داریم: یادگیری مبتنی بر مدل[5] و یادگیری بدون مدل[6]. یادگیری مبتنی بر مدل تلاش می‌کند تا توابع انتقال و پاداش را با نمونه‌های به‌دست‌آمده در طول اکتشاف، قبل از استفاده از این تخمین‌ها برای حل MDP به طور معمول با تکرار ارزش یا تکرار خط‌مشی، تخمین بزند. از سوی دیگر، یادگیری بدون مدل، تلاش می‌کند تا مقادیر یا مقادیر Q حالت‌ها را مستقیماً تخمین بزند، بدون اینکه از هیچ حافظه‌ای برای ساخت مدلی از پاداش‌ها و انتقال‌ها در MDP استفاده کند.

یادگیری مبتنی بر مدل[7]

[در یادگیری تقویتی مبتنی بر مدل، عامل یادگیری تقویتی که مدلی از محیط را یاد می گیرند و از آن مدل برای برای ایجاد یک قانون کنترلی بدون آزمون و خطای مستقیم استفاده می نماید.]

در یادگیری مبتنی بر مدل، عامل تقریبی از تابع انتقال ($\hat{T}(s,a,s')$) را با نگه داشتن تعداد دفعاتی که پس از رسیدن به هر حالت Q (s,a) به حالت s'می‌رسد محاسبه می‌کند. سپس عامل می‌تواند تابع انتقال تقریبی $\hat{T}$ را با عادی‌سازی شمارش‌هایی که جمع کرده است محاسبه کند، سپس تعداد هر (s,a, s′) مشاهده‌شده را بر مجموع شمارش‌ها برای همه مواردی که عامل در حالت Q (s,a) قرار داشته است تقسیم می‌کند . عادی سازی شمارش ها آنها را به گونه ای مقیاس می کند که مجموع آنها به یک برسد و بتوان آنها را به عنوان احتمالات تفسیر کرد .مثال MDP زیر را با حالت های S = {A,B,C,D,E,x} در نظر بگیرید که x نشان دهنده حالت پایانی است و ضریب کاهش $\gamma = 1$ است:



فرض کنید به عامل اجازه بدهیم تا MDP را برای چهار قسمت تحت خط مشی π explore که در بالا مشخص شده است کاوش کند (مثلث جهت دار حرکت در جهتی را که مثلث نشان می دهد و مربع های آبی نشان دهنده خروج به عنوان اقدام انتخابی است) را نشان می دهد و نتایج زیر را به دست می‌آید:

[4] episode
[5] model-based learning
[6] model-free learning
[7] Model-Based Learning

Episode 1

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2

B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3

E, north, C, -1
C, east, D, -1
D, exit, x, +10

Episode 4

E, north, C, -1
C, east, A, -1
A, exit, x, -10

اکنون ۱۲ نمونه جمعی داریم، ۳ نمونه از هر بخش با شمارشش به شرح زیر است:

s	a	s′	count
A	*exit*	x	1
B	*east*	C	2
C	*east*	A	1
C	*east*	D	3
D	*exit*	x	3
E	*north*	C	2

می دانیم که $T(s,a,s') = P(s'|a,s)$، پس می توانیم تابع انتقال را با این شمارش ها با تقسیم تعداد هر (s,a,s')بر تعداد کل مواقعی که در حالت $Q(s,a)$ قرار داشتیم و تابع پاداش مستقیماً از پاداش هایی که در طول اکتشاف به دست آوردیم، تخمین بزنیم:

- **Transition Function**: $\hat{T}(s,a,s')$
  - $\hat{T}(A,exit,x) = \frac{\#(A,exit,x)}{\#(A,exit)} = \frac{1}{1} = 1$
  - $\hat{T}(B,east,C) = \frac{\#(B,east,C)}{\#(B,east)} = \frac{2}{2} = 1$
  - $\hat{T}(C,east,A) = \frac{\#(C,east,A)}{\#(C,east)} = \frac{1}{4} = 0.25$
  - $\hat{T}(C,east,D) = \frac{\#(C,east,D)}{\#(C,east)} = \frac{3}{4} = 0.75$
  - $\hat{T}(D,exit,x) = \frac{\#(D,exit,x)}{\#(D,exit)} = \frac{3}{3} = 1$
  - $\hat{T}(E,north,C) = \frac{\#(E,north,C)}{\#(E,north)} = \frac{2}{2} = 1$
- **Reward Function**: $\hat{R}(s,a,s')$
  - $\hat{R}(A,exit,x) = -10$
  - $\hat{R}(B,east,C) = -1$
  - $\hat{R}(C,east,A) = -1$
  - $\hat{R}(C,east,D) = -1$
  - $\hat{R}(D,exit,x) = +10$
  - $\hat{R}(E,north,C) = -1$

طبق قانون اعداد بزرگ[8]، هرچقدر که عامل نمونه‌های بیشتری را با تجربه کردن قسمت های بیشتر جمع‌آوری کند، مدل‌های $\hat{T}$ و $\hat{R}$ ما بهبود می‌یابند، و $\hat{T}$ به سمت T و $\hat{R}$ همگرا می‌شود، و با کشف (s,a,s') های جدید، دانش پاداش‌هایی قبلاً کشف‌نشده اند به دست می‌آیند. هر زمان که بخواهیم، می‌توانیم آموزش عامل خود را برای ایجاد یک سیاست exploit π با اجرای روش تکرار ارزش یا تکرار خط مشی با مدل‌های فعلی خود برای $\hat{T}$ و $\hat{R}$ پایان دهیم و از exploit π برای بهره‌برداری استفاده کنیم، و از عامل بخواهیم تا MDP را طی کند و اقداماتی را برای به حداکثر رساندن پاداش انجام دهد به جای اینکه به دنبال یادگیری باشد. به زودی روش هایی را برای چگونگی تخصیص زمان موثر بین اکتشاف و بهره برداری را ارائه می‌کنیم. یادگیری مبتنی بر مدل بسیار

[8] law of large numbers

ساده و شهودی و در عین حال بسیار مؤثر است و $\hat{T}$ و $\hat{R}$ را با شمارش و عادی‌سازی تولید می‌کند. با این حال، حفظ شمارش برای هر (s, a, s') که بررسی می‌شود می‌تواند هزینه‌بر باشد، و بنابراین در بخش بعدی در مورد یادگیری بدون مدل، روش‌هایی را برای دور زدن کل حفظ شمارش و جلوگیری از سربار حافظه مورد نیاز روش یادگیری مبتنی بر مدل شرح خواهیم داد.

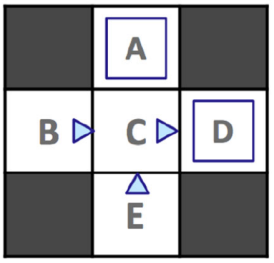
یادگیری بدون مدل[9]

[در یادگیری تقویتی بدون مدل، عامل یادگیری تقویتی که به مدلی از محیط متکی نیستند. آنها از تجربه مستقیم در تعامل با محیط یادگیری را انجام می دهند.]

چندین الگوریتم یادگیری بدون مدل وجود دارد، و ما به سه مورد از آنها خواهیم پرداخت: ارزیابی مستقیم(direct evaluation)، یادگیری تفاوت زمانی(temporal difference learning) و یادگیری Q (Q-learning). ارزیابی مستقیم و یادگیری تفاوت زمانی تحت دسته‌ای از الگوریتم‌ها قرار می‌گیرند که به عنوان یادگیری تقویتی غیرفعال[10] شناخته می‌شوند. در یادگیری تقویتی غیرفعال، به یک عامل خط مشی داده می شود تا دنبال کند و ارزش ارزش حالت های تحت آن خط مشی را در حین تجربه قسمت ها بیاموزد، این دقیقاً همان کاری است که با ارزیابی خط مشی برای MDP ها زمانی که T و R را داشتیم، انجام می شد. یادگیری Q تحت دسته دومی از الگوریتم‌های یادگیری بدون مدل قرار می‌گیرد که به عنوان یادگیری تقویتی فعال[11] شناخته می‌شوند، که طی آن عامل می‌تواند از بازخوردی که دریافت می‌کند برای به‌روزرسانی مکرر خط‌مشی خود در حین یادگیری استفاده کند تا سرانجام پس از کاوش کافی، خط‌مشی بهینه را تعیین کند.

ارزیابی مستقیم

اولین تکنیک یادگیری تقویتی غیرفعال که به آن می پردازیم، ارزیابی مستقیم است، روشی که مانند نامش خسته کننده و ساده است. تنها کاری که ارزیابی مستقیم انجام می دهد این است که برخی از خط مشی های π را اصلاح می کند و نماینده چندین قسمت را حین پیروی از π تجربه می کند. همانطور که عامل از طریق این قسمت ها نمونه ها را جمع آوری می کند، تعداد کل سودمندی به دست آمده از هر حالت و تعداد دفعاتی که از هر حالت بازدید کرده است را نیز حفظ می کند. در هر نقطه، می‌توانیم ارزش تخمینی هر حالت s را با تقسیم کل سودمندی به‌دست‌آمده از حالت s بر تعداد دفعاتی که حالت sبازدید شده است، محاسبه کنیم. اکنون ارزیابی مستقیم را بر روی مثال قبلی خود اجرا می کنیم، به یاد داشته باشید که $\gamma = 1$ است.



Episode 1
B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 2
B, east, C, -1
C, east, D, -1
D, exit, x, +10

Episode 3
E, north, C, -1
C, east, D, -1
D, exit, x, +10

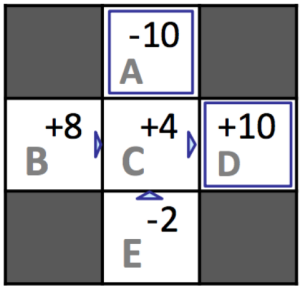
Episode 4
E, north, C, -1
C, east, A, -1
A, exit, x, -10

با گذر از قسمت اول، می‌بینیم که از حالت D تا پایان، پاداش کلی ۱۰، از وضعیت C پاداش کلی9 = 10+(−1) ، و از حالت B پاداش کلی (8 = 10+(−1)+(−1 را به دست می آوردیم. با تکمیل این فرآیند، پاداش کل در هر قسمت برای هر حالت و مقادیر تخمینی حاصل به شرح زیر به دست می‌آیند:

[9] Model-Free Learning
[10] passive reinforcement learning
[11] active reinforcement learning

s	Total Reward	Times Visited	$V^{\pi}(s)$
A	-10	1	-10
B	16	2	8
C	16	4	4
D	30	3	10
E	-4	2	-2

اگرچه ارزیابی مستقیم در نهایت مقادیر حالت را برای هر حالت بررسی می‌کند، اما اغلب به کندی همگرا می‌شود زیرا اطلاعات مربوط به انتقال بین حالت‌ها را هدر می‌دهد.



در این مثال، $V^{\pi}(E) = -2$ و $V^{\pi}(B) = 8$ را محاسبه کردیم، اگرچه بر اساس بازخوردی که دریافت کردیم، هر دو حالت فقط C را به عنوان حالت جانشین دارند و در هنگام انتقال به C ، پاداش یکسان -1 را متحمل می‌شوند. با توجه به معادله بلمن، این بدان معنی است که B و E باید هر دو مقدار یکسانی را تحت خط مشی π داشته باشند. با این حال، از ۴ باری که عامل ما در وضعیت C بود، به D منتقل شد و سه بار پاداش ۱۰ را کسب کرد و به A منتقل شد و یک بار پاداش ۱۰- را کسب کرد. این کاملاً تصادفی بوده است که هنگامی که پاداش ۱۰- را دریافت کرد از حالت E به جای B شروع کرده بود، اما این به شدت ارزش تخمینی را برای E تغییر داده است. با تعداد قسمت های کافی، مقادیر B و E به مقادیر واقعی خود همگرا می‌شوند، اما مواردی مانند این باعث می‌شوند که فرآیند بیش از آنچه ما می‌خواهیم طول بکشد. این مشکل را می توان با استفاده از دومین الگوریتم یادگیری تقویتی غیرفعال، یعنی یادگیری تفاوت زمانی، کاهش داد.

یادگیری تفاوت زمانی

[یادگیری تفاوت زمانی(Temporal difference learning) یک روش پیش‌بینی است. این روش به صورت عمده برای حل مسائل یادگیری تقویتی مورد استفاده بود است. روش تفاوت زمانی ترکیبی از ایده‌های مونت کارلو و برنامه‌ریزی پویا است. این روش مشابه روش مونت کارلو است چرا که یادگیری در آن با استفاده از نمونه برداری از محیط با توجه به یک یا چند *سیاست خاص* انجام می‌شود. روش تفاوت زمانی به این دلیل به تکنیک‌های برنامه‌ریزی پویا شباهت دارد که این روش تخمین کنونی را بر اساس تخمین‌های یادگیری شده به دست می‌آورد. به عنوان یک روش پیش‌بینی، یادگیری تفاوت زمانی این واقعیت را در نظر می‌گیرد که پیش‌بینی‌های آینده نیز معمولاً از جهاتی دارای همبستگی هستند. در روش‌های یادگیری مبتنی بر پیش‌بینی نظارتی، عامل تنها از مقادیر دقیقاً مشاهده شده یاد می‌گیرد: یک پیش‌بینی انجام می‌شود، و زمانی که مشاهده ممکن باشد، پیش‌بینی به تطابق بهتری با مشاهده خواهد رسید. ایده اساسی یادگیری تفاوت زمانی این است که پیش‌بینی‌ها با پیش‌بینی‌هایی دقیق تر دیگری از آینده تنظیم شوند.]

یادگیری تفاوت زمانی (TD Learning) از ایده یادگیری از هر تجربه، به جای اینکه صرفاً کل پاداش ها و تعداد دفعات بازدید از حالات و یادگیری در پایان را مانند روش ارزیابی مستقیم پیگیری کند، استفاده می کند. در ارزیابی خط مشی، ما از سیستم معادلات تولید شده توسط خط مشی ثابت خود و معادله بلمن برای تعیین مقادیر حالت های تحت آن خط مشی استفاده کردیم (یا از به روز رسانی های تکراری مانند روش تکرار ارزش استفاده کردیم).

$$V^{\pi}(s) = \sum_{s'} T(s, \pi(s), s') [R(s, \pi(s), s') + \gamma V^{\pi}(s')]$$

هر یک از این معادلات، ارزش یک حالت را برابر با میانگین وزنی نسبت به مقادیر کاهش یافته جانشینان آن حالت به علاوه پاداش های به دست آمده در انتقال به آنها قرار می‌دهد. یادگیری تفاوت زمانی سعی می کند به این سوال پاسخ دهد که چگونه می توان این میانگین وزنی را بدون وزن محاسبه کرد و این کار را هوشمندانه با میانگین متحرک نمایی[12] انجام می دهد. با مقداردهی اولیه $\forall s, V^{\pi}(s) = 0$ شروع می کنیم. در هر گام زمانی، یک عامل یک عمل $\pi(s)$ را از حالت s انجام می دهد، به حالت s' منتقل می‌شود و پاداش $R(s, \pi(s), s')$ را کسب می‌کند. می‌توانیم با جمع کردن پاداش دریافتی با مقدار فعلی کاهش یافته s' تحت π، یک ارزش نمونه[13] بدست بیاوریم:

$$\text{sample} = R(s, \pi(s), s') + \gamma V^{\pi}(s')$$

این ارزش نمونه، تخمین جدیدی برای $V^{\pi}(s)$ است. گام بعدی این است که این تخمین نمونه را در مدل موجود خود برای $V^{\pi}(s)$ با میانگین متحرک نمایی، که به قانون به روز رسانی زیر پایبند است، بگنجانیم:

$$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha \cdot \text{sample}$$

در معادله بالا، α پارامتری است با محدوده $0 \leq \alpha \leq 1$ که به عنوان نرخ یادگیری شناخته می شود، که وزنی را که می خواهیم مدل موجود خود را برای $V^{\pi}(s)$، $1-\alpha$ اختصاص دهیم، و وزنی را که می خواهیم تخمین نمونه جدید خود یعنی α را به آن اختصاص دهیم، مشخص می کند. به طور معمول با نرخ یادگیری $\alpha = 1$ شروع می‌کنیم، بر این اساس $V^{\pi}(s)$ را به هر نمونه اولیه نسبت می دهیم، و به آرامی آن را به سمت ۰ کاهش می‌دهیم، در این مرحله تمام نمونه های بعدی صفر می شوند و دیگر تاثیری بر مدل ما یعنی $V^{\pi}(s)$ ندارند.

اکنون قانون به روز رسانی را تجزیه و تحلیل می‌کنیم. با تشریح وضعیت مدل خود در مقاطع مختلف زمانی با تعریف $V_k^{\Pi}(s)$ و sample_k به عنوان مقدار تخمینی حالت s پس از به‌روزرسانی k ام و k امین نمونه، می‌توانیم قانون به‌روزرسانی خود را دوباره بیان کنیم:

$$V_k^{\pi}(s) \leftarrow (1-\alpha)V_{k-1}^{\pi}(s) + \alpha \cdot \text{sample}_k$$

این تعریف بازگشتی را برای $V_k^{\Pi}(s)$ بسط می‌دهیم:

$$\begin{aligned}
V_k^{\pi}(s) &\leftarrow (1-\alpha)V_{k-1}^{\pi}(s) + \alpha \cdot \text{sample}_k \\
V_k^{\pi}(s) &\leftarrow (1-\alpha)[(1-\alpha)V_{k-2}^{\pi}(s) + \alpha \cdot \text{sample}_{k-1}] + \alpha \cdot \text{sample}_k \\
V_k^{\pi}(s) &\leftarrow (1-\alpha)^2 V_{k-2}^{\pi}(s) + (1-\alpha) \cdot \alpha \cdot \text{sample}_{k-1} + \alpha \cdot \text{sample}_k \\
&\vdots \\
V_k^{\pi}(s) &\leftarrow (1-\alpha)^k V_0^{\pi}(s) + \alpha \cdot [(1-\alpha)^{k-1} \cdot \text{sample}_1 + \ldots + (1-\alpha) \cdot \text{sample}e_{k-1} + \text{sample}_k] \\
V_k^{\pi}(s) &\leftarrow \alpha \cdot [(1-\alpha)^{k-1} \cdot \text{sample}_1 + \ldots + (1-\alpha) \cdot \text{sample}_{k-1} + \text{sample}_k]
\end{aligned}$$

از آنجایی که $0 \leq (1-\alpha) \leq 1$ است، با افزایش مقدار $(1-\alpha)$ به توان های بزرگتر، این مقدار به ۰ نزدیک تر و نزدیک تر می شود. با بسط قانون به روز رسانی ای که به دست آوردیم، این بدان معناست که به نمونه های قدیمی به طور تصاعدی وزن کمتری داده می شود، دقیقاً همان چیزی که می خواهیم زیرا این نمونه های قدیمی با استفاده از نسخه های قدیمی تر (و در نتیجه بدتر) مدل ما برای $V_k^{\Pi}(s)$ محاسبه می شوند! این خوبی روش یادگیری تفاوت های زمانی است، با یک قانون به روز رسانی ساده، ما می‌توانیم:

- در هر مرحله زمانی یاد می‌گیریم، از این رو از اطلاعات مربوط به انتقال حالت به هنگام دریافت آنها استفاده می کنیم، زیرا ما از نسخه های به روز شده مکرر $V^{\pi}(s')$ در نمونه های خود استفاده می کنیم به جای اینکه برای انجام هر محاسباتی تا پایان منتظر بمانیم.
- به نمونه‌های قدیمی‌تر و بالقوه کمتر، وزن کمتری می‌دهیم.

---

[12] Exponential moving average
[13] sample value

- با قسمت های کمتر نسبت به ارزیابی مستقیم، یادگیری مقادیر حالت واقعی خیلی سریعتر همگرا می شوند.

یادگیری Q

[یادگیری Q، یک تکنیک یادگیری تقویتی است که با یادگیری یک تابع اقدام/مقدار، سیاست مشخصی را برای انجام حرکات مختلف در وضعیت‌های مختلف دنبال می‌کند. یکی از نقاط قوت این روش، توانایی یادگیری تابع مذکور بدون داشتن مدل معینی از محیط می‌باشد.. یادگیری Q در تلاش است با توجه به شرایط فعلی، بهترین اقدامات را انجام دهد. این الگوریتم خارج‌از خط‌مشی در نظر گرفته می‌شود. چراکه، تابع یادگیری Q از اقداماتی خارج‌از خط‌مشی فعلی یادمی‌گیرد. به‌طور کلی می‌توان گفت که، یادگیری Q به‌دنبال یادگیری خط‌مشی است تا مجموع پاداش را بیشینه کند.]

هم ارزیابی مستقیم و هم یادگیری TD در نهایت ارزش واقعی همه حالت ها را تحت سیاستی که دنبال می کنند یاد خواهند گرفت. با این حال، هر دوی آنها یک مسئله ذاتی اصلی دارند، ما می خواهیم یک خط مشی بهینه برای عامل خود پیدا کنیم که نیاز به آگاهی از مقادیر Q حالت ها دارد. برای محاسبه مقادیر Q از مقادیری که داریم، به یک تابع انتقال و تابع پاداش که توسط معادله بلمن ایجاد می‌شود، نیاز داریم.

$$Q^*(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

در نتیجه، یادگیری TD یا ارزیابی مستقیم معمولاً همراه با برخی از یادگیری های مبتنی بر مدل برای بدست آوردن تخمین T و R به منظور به روز رسانی موثر خط مشی دنبال شده توسط عامل درحال یادگیری استفاده می شوند. این امر با ایده انقلابی جدید به نام یادگیری Q ، که یادگیری مستقیم مقادیر Q حالت‌ها را پیشنهاد می‌کرد، اجتناب‌پذیر شد و نیاز به دانستن هر گونه ارزش، توابع انتقال یا توابع پاداش را دور زد. در نتیجه، یادگیری Q کاملاً یک یادگیری بدون مدل است. یادگیری Q از قانون به روز رسانی زیر برای انجام آنچه به عنوان تکرار ارزش Q[14] شناخته می شود استفاده می کند:

$$Q_{k+1}(s,a) \leftarrow \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma \max_{a'} Q_k(s',a')]$$

توجه کنید که این به روز رسانی فقط یک تغییر جزئی در قانون به روز رسانی برای تکرار ارزش است. در واقع، تنها تفاوت این است که موقعیت حداکثر عملگر بر روی اقدامات تغییر کرده است، زیرا ما یک عمل را قبل از انتقال، زمانی که در یک حالت هستیم انتخاب می کنیم، اما قبل از انتخاب یک عمل جدید، زمانی که در یک حالت Q هستیم، انتقال را انجام می دهیم.

با این قانون به روز رسانی جدید تحت ، یادگیری Q اساساً به همان روش یادگیری TD ، با به دست آوردن نمونه های ارزش[15] Q مشتق می شود:

$$\text{sample} = R(s,a,s') + \gamma \max_{a'} Q(s',a')$$

و ترکیب آنها در یک میانگین متحرک نمایی به صورت زیر خواهد شد:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha \cdot \text{sample}$$

تا زمانی که زمان کافی را صرف کاوش کنیم و نرخ یادگیری α را با سرعت مناسب کاهش دهیم، یادگیری Q ارزش های Q بهینه را برای هر حالت Q می آموزد. این همان چیزی است که یادگیری Q را بسیار انقلابی می‌کند، در حالی که یادگیری TD و ارزیابی مستقیم ارزش‌های حالت‌های تحت یک خط‌مشی را با پیروی از خط‌مشی، قبل از تعیین بهینه بودن خط‌مشی از طریق تکنیک‌های دیگر یاد می‌گیرند، یادگیری Q می‌تواند مستقیماً خط‌مشی

[14] Q-value iteration
[15] Q-value samples

بهینه را حتی با استفاده از روش‌های زیر بهینه یا اقدامات تصادفی یاد بگیرد. این را یادگیری خارج از سیاست[16] می نامند(برخلاف ارزیابی مستقیم و یادگیری TD که نمونه هایی از یادگیری درون سیاست[17] هستند).

یادگیری Q تقریبی

یادگیری Q یک تکنیک یادگیری باورنکردنی است که همچنان در مرکز تحولات در زمینه یادگیری تقویتی قرار دارد. با این حال، هنوز هم جای پیشرفت دارد. یادگیری Q فقط تمام ارزش های Q را برای حالت ها به شکل جدول ذخیره می کند، که با توجه به اینکه اکثر برنامه های یادگیری تقویتی چندین هزار یا حتی میلیون ها حالت دارند، کارایی خاصی ندارد. این بدان معناست که ما نمی‌توانیم در طول آموزش از همه حالت‌ها بازدید کنیم و نمی‌توانیم تمام ارزش های Q را به دلیل کمبود حافظه ذخیره کنیم.

شکل ۱

شکل ۲

شکل ۳

در بالا، اگر پک من یاد گرفته بود که شکل ۱ پس از اجرای یادگیری Q نامطلوب است، باز هم متوجه نمی‌شد که شکل ۲ یا حتی شکل ۳ نیز نامطلوب هستند. یادگیری Q تقریبی تلاش می‌کند این مشکل را با یادگیری در مورد چند موقعیت کلی و برون‌یابی به بسیاری از موقعیت‌های مشابه توضیح دهد. کلید تعمیم تجارب یادگیری، نمایش حالت‌های مبتنی بر ویژگی[18] است که هر حالت را به عنوان یک بردار نشان می‌دهد که به عنوان بردار ویژگی[19] شناخته می‌شود. به عنوان مثال، یک بردار ویژگی برای پک من ممکن است به صورت های زیر رمزگذاری شود:

- فاصله تا نزدیکترین روح
- فاصله تا نزدیکترین گلوله غذا
- تعداد ارواح
- آیا پکمن به دام افتاده است؟ 0یا ۱

با بردارهای ویژگی، می‌توانیم ارزش حالت‌ها و حالت‌های Q را به عنوان توابع ارزش خطی[20] در نظر بگیریم:

$$V(s) = w_1 \cdot f_1(s) + w_2 \cdot f_2(s) + \ldots + w_n \cdot f_n(s) = \vec{w} \cdot \vec{f}(s)$$
$$Q(s,a) = w_1 \cdot f_1(s,a) + w_2 \cdot f_2(s,a) + \ldots + w_n \cdot f_n(s,a) = \vec{w} \cdot \vec{f}(s,a)$$

که در آن

$$\vec{f}(s) = \begin{bmatrix} f_1(s) & f_2(s) & \ldots & f_n(s) \end{bmatrix}^T$$

و

---

[16] off-policy learning
[17] on-policy learning
[18] feature-based representation
[19] feature vector
[20] linear value functions

$$\vec{f}(s,a) = \begin{bmatrix} f_1(s,a) & f_2(s,a) & \dots & f_n(s,a) \end{bmatrix}^T$$

به ترتیب نشان دهنده بردارهای ویژگی برای حالت s و حالت Q (s,a) و

$$\vec{w} = \begin{bmatrix} w_1 & w_2 & \dots & w_n \end{bmatrix}$$

یک بردار وزن را نشان می دهد. تفاوت به صورت زیر بیان می‌کنیم:

$$\text{difference} = [R(s,a,s') + \gamma \max_{a'} Q(s',a')] - Q(s,a)$$

یادگیری تقریبی Q ، تقریباً مشابه یادگیری Q با استفاده از قانون به روز رسانی زیر عمل می کند:

$$w_i \leftarrow w_i + \alpha \cdot \text{difference} \cdot f_i(s,a)$$

به جای ذخیره کردن ارزش های Q برای هر حالت، با یادگیری تقریبی Q فقط به ذخیره یک بردار وزنی نیاز داریم و می‌توانیم ارزش های Q را در صورت نیاز محاسبه کنیم. در نتیجه، این نه تنها نسخه تعمیم‌یافته‌تری از یادگیری Q را در اختیار ما قرار می‌دهد، بلکه نسخه‌ای از نظر حافظه کارآمدتر نیز به ما می‌دهد.

به عنوان آخرین نکته در مورد یادگیریQ ، می توانیم قانون به روز رسانی را برای یادگیری دقیق Q با استفاده از تفاوت به صورت زیر بیان کنیم:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot \text{difference}$$

این نماد دوم تفسیر کمی متفاوت اما به همان اندازه ارزشمند از به روز رسانی را به ما می دهد: این محاسبه تفاوت بین مدل تخمینی نمونه و مدل فعلی Q(s,a) است، و مدل را در جهت تخمین جابجا می کند تا بزرگی تغییر، متناسب با بزرگی تفاوت باشد.