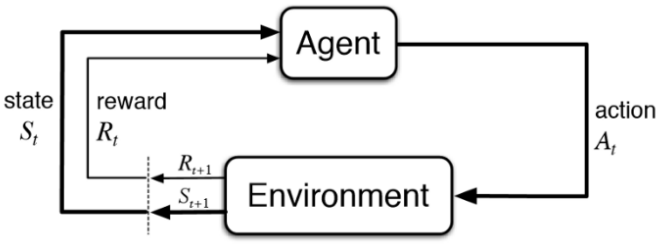
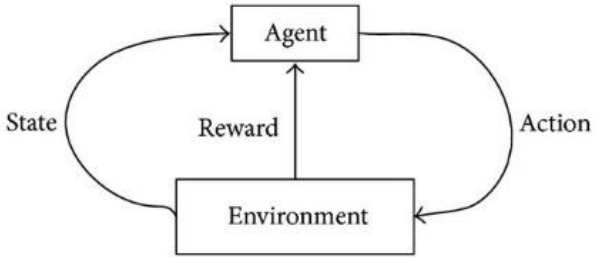
Reinforcment Learning Progect Documentation

Mohammad Andalibi & Mohammadmahdi Mohammadi – spring 2020

هوش مصنوعی، علم مطالعه و بررسی ماشین‌هایی است که رفتار هوشمندانه دارند. مفهوم هوشمندی، چیزی است که به سختی قابل تعریف است. اما معمولا هوشمندی با توانایی یادگیری از طریق تجربه، مرتبط دانسته می‌شود. یادگیری ماشینی، زمینه‌ای مطالعاتی در هوش مصنوعی است که به دنبال ایجاد عامل‌هایی همچون برنامه‌های کامپیوتری است که بتوانند با استفاده از تجربیات خود یاد بگیرند. ایده‌ی ایجاد ماشین‌هایی که رفتار هوشمندانه‌ی انسان را تقلید می‌کنند، بدون الهام گرفتن از هوشمندی انسان‌ها، هیچ‌گاه کامل نخواهد بود و به ثمر نخواهد نشست.ایجاد چنین ماشین‌هایی، بسیار زیاد است. زیرا ما اطلاع دقیقی از منبع هوشمندی خودمان نداریم و طبعا نمی‌توانیم به راحتی، ماشین‌هایی هوشمند و نظیر خودمان را بسازیم.

مهم ترین کاری که غالبا یک سیستم هوشمند انجام میدهد، امکان یادگیری است و اگر یک سیستم هوشمند بتواند یادگیری انسان را شبیه سازی کند، تا حدود زیادی در پیاده سازی هوش مصنوعی، موفق خواهد بود. هدف اصلی از یادگیری یافتن شیوه‌ای برای عملکرد در حالات مختلف است که این شیوه در مقایسه با سایرین، با در نظر گرفتن معیارهایی، بهتر است. معمولا این شیوه‌ی عملکرد، از نظر ریاضی، به صورت نگاشتی از فضای حالات به فضای اعمال، قابل بیان است. هنگامی می‌توان گفت یادگیری اتفاق افتاده است که، عاملی بر اساس تجربیاتی که کسب می‌کند به نحوی دیگر، و به احتمال زیاد بهتر، عمل کند. در این صورت می‌بایست نحوه‌ی عملکرد عامل در اثر کسب اطلاعات جدید، متفاوت از نحوه‌ی عملکرد در زمان قبل از کسب این اطلاعات و تجارب باشد.





فرایندهای تصمیم‌گیری مارکوف (Markov Decision Process)

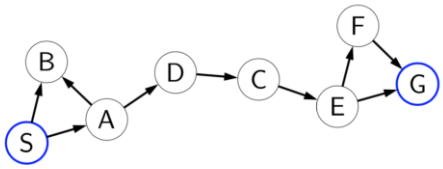
معمولا یک MDP به صورت چندتایی (S , A , T , R) نشان داده می‌شود.

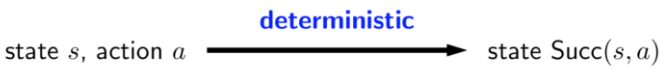
متغیر S مجموعه‌ای از همه حالت‌های ممکن است. در بازی ساده tic-tac-toe ، هر مربع(خانه) می‌تواند یکی از سه مقدار: خالی، X و یا O را داشته باشد. با این شرایط تعداد حالت‌های ممکن، بیش از 3^9 نمی‌تواند باشد. با این وجود این مقدار، تعدادی از حالت‌هایی که در بازی tic-tac-toe معتبر نیست را نیز دربر می‌گیرد.

متغیر A مجموعه‌ای از فعالیت‌هایی است که برای هر عامل در دسترس است. این موضوع می‌تواند از راه‌های مختلفی توضیح داده شود. برای مثال: مسیر یک کاراکتر(شخصیت) در یک بازی ویدیوئی در یک صفحه مشبک (Grid)می‌تواند اعمال : *رفتن به بالا، چپ، راست و پایین* باشد. به همین صورت، کاراکتر می‌تواند عمل‌هایی مثل :*چرخش ۹۰ درجه و رفتن به جلو* را برای تولید نتایج یکسان انجام دهد. عامل‌ها در بسیاری از موارد می‌توانند هیچ عملی انجام ندهند.

متغیر T تابع گذار است که مشخص می‌کند یک حالت چطور به وجود می‌آید. به عنوان ورودی، حالت فعلی، عملی که روی آن انجام شده و حالت بعدی را دریافت می‌کند و احتمال این‌که چقدر حالت بعدی پیشنهادشده، حالت بعدی مطلوب باشد را به عنوان خروجی برمی‌گرداند.

متغیر R تابع تشویق(جایزه) است. این تابع به عنوان ورودی، حالت فعلی و حرکتی که انجام شده را گرفته و امتیازی که عامل با انجام‌دادن آن حرکت به دست می‌آورد را نمایش می‌دهد و معمولا همه مقدارهای گویا را می‌پذیرد





معمولا، زمانی که شما یک MDP را تعریف کردید، می‌خواهید که آن را حل کنید. به‌ویژه این که شما علاقه‌مند هستید که استراتژی بهینه‌ای را پیدا کنید که عامل باید با آن منطبق شود تا بیشترین امتیاز را زمان بازی کردن به‌دست آورد. این استراتژی بهینه همچنین سیاست بهینه نیز نامیده می‌شود. یک سیاست، هر فرآیند تصمیم‌گیری است که هر گذشته ممکن را به یک انتخاب درباره حرکت بعدی که انجام می‌شود، نگاشت می‌کند. از این راه، یک MDP به صورت مستقیم به یک مدل عامل نگاشت می‌شود.

معرفی بر الگوریتم های تقویتی(RL) :

در حالت کلی در حوزه یادگیری ماشین سه دسته الگوریتم وجود دارند که عبارت هستند از الگوریتم های supervised یا یادگیری با نظارت، الگوریتم های unsupervised یا یادگیری بدون نظارت و الگوریتم های Reinforcement learning یا یادگیری تقویتی. در الگوریتم های یادگیری با نظارت هدف یافتن یک تابع است که داده های ورودی را به برچسب ها نگاشت می‌کند برای مثال فرض کنید از روی یک تصویر می‌خواهید تشخیص دهید که در آن تصویر پراید وجود دارد یا نه؟ داده های ورودی شما در حالتی که به دنبال آموزش از طریق یادگیری با نظارت باشید عبارت است از عکس و برچسب های آن پراید یا غیر پراید می‌باشد. در نوع دوم یادگیری که تحت عنوان یادگیری بدون نظارت مطرح است داده ها برچسب ندارند و به دنبال یافتن الگو های پنهان در داده ها هستیم. **اما نوع سوم یادگیری که به یادگیری تقویتی معروف است. در این الگوریتم دیگر با برچسب ها سرکار نداریم و برعکس با سیگنال های تنبیه و تشویق مواجهیم. این نوع از الگوریتم که از یادگیری انسان ها الگو گرفته است مبنا را بر این قرار می‌دهد که در صورتی که الگوریتم به درستی عمل کند باید تشویق شود و برعکس. حال به بررسی بیشتر این الگوریتم می‌پردازیم.**

Agent	عامل تصمیم گیرنده تحت عنوان agent شناخته می‌شود.
Action	تمام حرکت های ممکنی که عامل می‌تواند انجام دهد.
Reward	سیگنال جایزه که بازخوردی است از این که action انتخاب شده توسط عامل چه مقدار خوب یا بد است.
State	شرایط حال حاضری که محیط آن را برمی‌گرداند. (شرایطی که در هر لحظه عامل با آن روبرو است.)
Policy	یک تابع وظیفه نگاشت state به action ها را بر عهده دارد.
Environment	محیطی که عامل تصمیم گیرنده در آن عمل میکند و از آن ها بازخورد می‌گیرد. هرآنچه به غیراز عامل جزء محیط محسوب می‌شود.
Value function	سیگنال reward برای بازه زمانی کوتاهی استفاده می‌شود حال آن که برای بازی مثل بازی شطرنج حالت نهایی مهم است. به عبارتی ممکن است در حرکت های کوتاه مدت عامل جایزه زیادی دریافت نکند و حتی reward منفی بگیرد اما برای بلند مدت نتیجه مناسبی داشته باشد.

یادگیری تقویتی [Reinforcement learning] روشی است که در آن عامل[Agent] با در نظر گرفتن حالت [State]محیط، از بین همه اعمال [Action]ممکن یکی را انتخاب می کند و محیط [Environment]در ازای انجام آن عمل، یک سیگنال عددی به نام پاداش [Reward]به عامل باز می گرداند .هدف عامل این است که از طریق سعی و خطا سیاستی [Policy]را بیابد که با دنبال کردن آن به بیشترین پاداش ممکن برسد .در این پروژه سعی داریم به عاملی که در محیط پر پیچ و خمی قرار دارد را به هدف برسانیم.

گفتیم **Value function** سیگنال reward است و برای بازه زمانی کوتاهی استفاده می‌شود حال آن که برای بازی مثل بازی شطرنج حالت نهایی مهم است. به عبارتی ممکن است در حرکت های کوتاه مدت عامل جایزه زیادی دریافت نکند و حتی reward منفی بگیرد اما برای بلند مدت نتیجه مناسبی داشته باشد. (مثلا الگوریتم وزیر را فدا می‌کند تا حریف را کیش و مات کند!)

$$\dots + V(s0, s1, \dots) = r(s0) + \gamma * r(s1) + \gamma 2 * r(s2)$$

$$Q^{\pi}(s,a) = \mathbb{E}[r_{t+1} + \lambda r_{t+2} + \lambda^2 r_{t+3} + \dots | s, a]$$

$$= \mathbb{E}_{s'}[r + \lambda Q^{\pi}(s', a') | s, a]$$

$$Q^{\pi}(s,a) = \mathbb{E}[r_{t+1} + \lambda r_{t+2} + \lambda^2 r_{t+3} + \dots | s, a]$$

$$= \mathbb{E}_{s'}[r + \lambda Q^{\pi}(s', a') | s, a]$$

Action value function همانند value function است با این تفاوت که تابع value function ارزش یک state را نشان می‌دهد حال آن که تابع action value میزان ارزش بودن در یک state و انتخاب یک action را نمایان می‌کند.

یک مفهوم دیگر تحت عنوان optimal policy وجود دارد. این مفهوم بیان کننده policy است که تحت این policy مقدار Q ماکزیمم می‌شود.

حال پس از این معرفی مختصر از مفاهیم موجود به بررسی دو مسئله می‌پردازیم که عبارت هستند از الگوریتم تقویتی مبتنی بر مدل و الگوریتم های تقویتی بدون مدل. **الگوریتم های تقویتی مبتنی بر مدل الگوریتم هایی هستند که در آن ها فهمی از رفتار محیط وجود دارد به این گونه که عامل می‌داند که احتمال رفتن از یک state به state دیگر چیست. به عبارتی بدون آنکه آزمایش و تجربه کرده باشد از آینده state ها آگاه است. اما مدل هایی که مبتنی بر مدل نیستند به وسیله آزمایش و خطا پیش می‌روند. به دلیل آنکه برای تمام محیط ها نمیتوان مدل ساخت بیشتر الگوریتم های کاربردی مبتنی بر الگوریتم هایی هستند که هیچ پیش فرضی راجب محیط ندارند.**

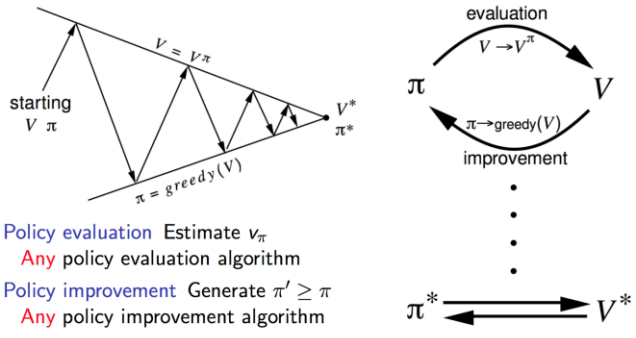
الگوریتم های Policy Iteration

این الگوریتم از دو قسمت شکل گرفته است. درقسمت اول ارزیابی policy اتفاق می‌افتد و در قسمت دوم بهبود policy اتفاق می‌افتد. در حالت ارزیابی policy ، تابع V که همان value function است از روی policy قبلی تخمین زده میشود. حال در حالت بهبود policy از روی تابع تخمین زده شده در مرحله قبل آن policy یافته میشود که value function را بیشینه کند. این حلقه تا زمانی که الگوریتم همگرا شود ادامه خواهد یافت. در ادامه یک شبه کد از این الگوریتم را مشاهده می‌کنید.

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Repeat
   $\Delta \leftarrow 0$
   For each $s \in \mathcal{S}$:
   $v \leftarrow V(s)$
   $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s))\left[r + \gamma V(s')\right]$
   $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $a \leftarrow \pi(s)$
   $\pi(s) \leftarrow \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
   If $a \neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return V and π; else go to 2



ارزیابی سیاست:

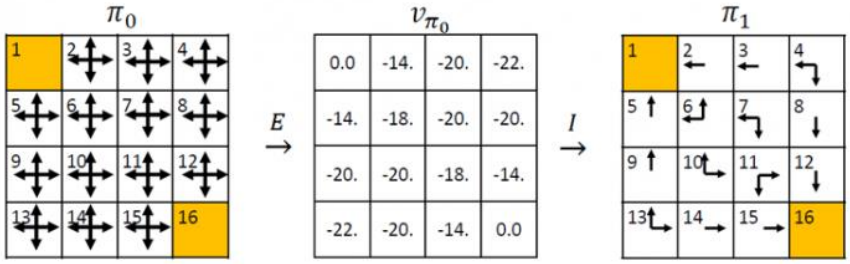
همیشه مستقیم برو

همیشه به راست برو

$$V_0^{\pi}(s) \leftarrow 0$$

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_k^{\pi}(s')]$$

$$\pi_0 \xrightarrow{E} v_{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} v_{\pi_1} \xrightarrow{I} \pi_2 \xrightarrow{E} \dots \xrightarrow{I} \pi_* \xrightarrow{E} v_*$$



الگوریتم Value Iteration

در این الگوریتم فقط تابع Value function آپدیت می شود. شبه کد این الگوریتم در ادامه می‌آید .

Initialize array V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)

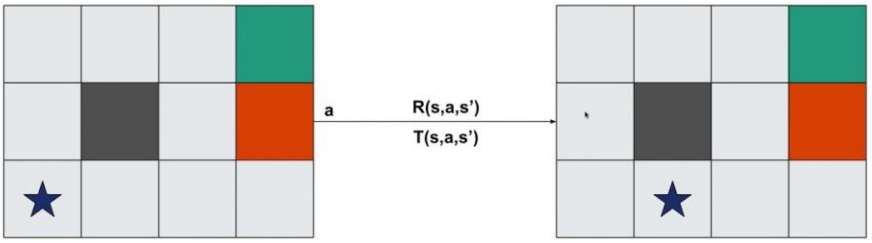
Repeat
  $\Delta \leftarrow 0$
  For each $s \in \mathcal{S}$:
    $v \leftarrow V(s)$
    $V(s) \leftarrow \max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
until $\Delta < \theta$ (a small positive number)

Output a deterministic policy, π, such that
  $\pi(s) = \arg\max_a \sum_{s',r} p(s', r|s, a)\left[r + \gamma V(s')\right]$

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

$$V_{k+1}(s) \leftarrow \max_a \sum_{s\prime} T(s,a,s')[R(s,a,s') + \gamma V_k(s')]$$

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



الگوریتم Q Learning

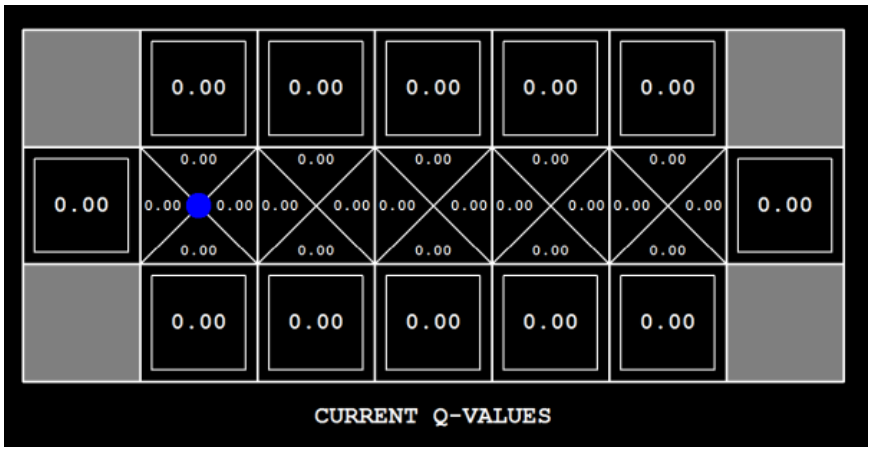
اگر به مدل های قبلی توجه داشته باشید همه این الگوریتم های مبتنی بر مدل بودند که در عمل چنین مسئله ای کمتر اتفاق می‌افتد. راه حل این مسئله استفاده از الگوریتم q learning هست. نحوه آپدیت کردن در این الگوریتم به شکل زیر است :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]$$

Q-learning: Learn function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$

Require:

Sates $\mathcal{X} = \{1, \ldots, n_x\}$
Actions $\mathcal{A} = \{1, \ldots, n_a\}$, $A : \mathcal{X} \Rightarrow \mathcal{A}$
Reward function $R : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$
Black-box (probabilistic) transition function $T : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$
Learning rate $\alpha \in [0, 1]$, typically $\alpha = 0.1$
Discounting factor $\gamma \in [0, 1]$
procedure QLEARNING($\mathcal{X}$, A, R, T, α, γ)
    Initialize $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ arbitrarily
    while Q is not converged **do**
        Start in state $s \in \mathcal{X}$
        while s is not terminal **do**
            Calculate π according to Q and exploration strategy (e.g. $\pi(x) \leftarrow \arg\max_a Q(x, a)$)
            $a \leftarrow \pi(s)$
            $r \leftarrow R(s, a)$ ▷ Receive the reward
            $s' \leftarrow T(s, a)$ ▷ Receive the new state
            $Q(s', a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot (r + \gamma \cdot \max_{a'} Q(s', a'))$
            $s \leftarrow s'$
    return Q



قسمت پایانی – پاسخ سوالات طرح پروژه

مشکلاتی که به آنها برخوردیم ...

- مهمترین مشکلی که در فرایند پیاده سازی این پروژه برای گروه مشکل ساز شد، بحث همگرایی میباشد به گونه ای که طبق طراحی های اولیه همگرایی را حالتی درنظرگرفتیم که در طی دو iteration متوالی مقدار تمامی V ها ثابت بماند که طی تحقیقات انجام شده متوجه شدیم که ممکن است در این مدلسازی سیستم هیچگاه متوقف نشود و به همین دلیل باید مقدار مشخصی را درنظر گرفت که اگر تعداد خانه های تغییر یافته از این مقدار مشخص کمتر بود را به عنوان همگرایی پذیرفتیم.
- مشکل دیگر در طراحی ساختمان داده بود که به دلیل بی توجهی به خروجی های عددی سیستم مقادیر از نوع int طراحی شده بودند که در مواردی گروه با خروجی صفر روبرو میشد که در ادامه متوجه شدیم این عدد صفر نیست بلکه به دلیل داشتن مقدار اعشاری خروجی صفر داشتییم و با جایگزین کردن آن با متغیر float این مشکل نیز برطرف شد.
- توجه نداشتن اعضا گروه نسبت به بهترین سیاست که سیاستی است توانمند برای ماکسیمم کردن مقدار V که در ابتدا سیاست تقریبا بهترین اعمال شد و در بعضی حالات مساله مشکل ساز میشد.
- موقعیت هایی پیشروی گروه قرار میگرفت که احتمال رفتن به State های مختلف مقدار برابری بودند که استفاده از حالت تصادفی برای حل این مشکل راهگشا شد.
- در الگوریتم value Iteration برای انجام محاسبات باید از Data Value های قبلی استفاده شود درحالیکه ابتدا گروه به اشتباه از مقادیری که در همان لحظه درحال آپدیت کردن آنها بود استفاده میکرد.

پاسخ سوالات مطرح شده در داکیومنت طرح پروژه:

پاسخ قسمت های الف تا ج : درمورد بحث بهینگی به بررسی حالت های مختلف و حرکت در محیط عامل ما تجارب بیشتری را به مرور از محیط کسب میکرد و مقادیری را که تخمین زد بود با گذر زمان دقیق و دقیقتر میشد پس میتوان گفت دیگر بعد از بحث همگرایی مسیری که انتخاب میشود بهینه میباشد. به صورت کلی بحث بسیار مهمی درمورد سایز مساله وجود دارد که البته با شرایط مساله تعیین میشود به عنوان مثال فضای بالای ۱۰*۱۰ در این مساله را دیگر نمیتوان ساده گفت و این حالتها به سمت مسائل سایز متوسط حرکت میکنند چراکه ماتریس gameboard آنها گسترش میابد ولی به صورت کلی متوجه شدیم که الگوریتم Value Iteration نسبت به Policy ساده تر میباشد اما این درحالیست که Policy سرعت بالاتری نسبت به Value دارا میباشد. موضوعی که اینجا باید مدنظر قرارگیر بحث بیشتر بودن محاسبات در Policy است که البته طی تحقیقات ابتدای پروژه به مقالاتی بابت ارایه هیورستیک برای Policy برخوردیم .نهایتا باید گفت برای مسایل کوچک استفاده از Policy Iteration انتخاب بهتری میباشد ولی برای مسایل بزرگ اگرچه تعداد iteration ها زیاد است اما میتوان گفت در حالت نگرانی بابت محاسبات و کنترل State نداریم.

از آنجایی که در حالت اولیه مقداردهی و Initial به صورت رندوم صورت میگیرد ممکن است مجموعه معادلاتی را به وجود بیاورد که به راحتی حل نشود و اینقدر اسپارس میشود که حل آن به راحتی امکان پذیر نمیباشد و این مهمترین ایراد الگوریتم Policy Iteration و گلوگاه بزرگ این الگوریتم علارغم سرعت بالای آن میباشد.

قسمت چ) درمورد الگوریتم Q Learning : در این حالت محیط برای ما شناخته شده نیست ولی این الگوریتم سعی کرده است از ایده های Valueو Policy نشات گرفته و الگوریتم بسیار معروفیست و ورژن های مختلفی را دارا میباشد. نهایتا میتوان گفت این الگوریتم برعکس دو الگوریتم دیگر برای محیط های شناخته نشده میباشد. در مورد بحث بهینگی هم باتوجه به اینکه در ابتدا به صورت رندوم State و Action را انتخاب میکنیم اما نهایتا با اطلاعاتی که درجدول بدست می آید همواره مسیر با امتیاز ماکسیمم را انتخاب میکند پس بهینه است.

قسمت د) طبیعتا طی این الگوریتم انتخاب State و Action در ابتدا برای بدست آمدن مقادیر جدول ارزش ها به صورت رندوم صورت میگیرد ولی پس از تکمیل شدن جدول دیگر فرایند به صورت رندوم نیست بلکه فرایندی انتخاب میشود که انجام آن مسبب بیشترین امتیاز برای عامل شود و علت بهینه بودن این الگوریتم نیز همین مساله میباشد.

منابع:

- **https://pathmind.com/wiki/deep-reinforcement-learning**
- **https://www.geeksforgeeks.org/what-is-reinforcement-learning/**
- **https://www.coursera.org/specializations/reinforcement-learning**
- **https://deepsense.ai/what-is-reinforcement-learning-the-complete-guide/**