به نام خدا

دانشکده‌ی مهندسی کامپیوتر

یادگیری ژرف

نیم‌سال اول ۰۳-۱۴۰۲

مدرس: دکتر حمید بیگی

تمرین سری ششم (۳۰+۱۰۰ نمره) | **یادگیری تقویتی** | زمان تحویل: ۲۵ دی (نظری)، ۶ بهمن (عملی)

لطفا نکات زیر را رعایت کنید:

- سوالات خود را از طریق پست مربوط به تمرین در Quera مطرح کنید.
- در هر کدام از سوالات، اگر از منابع خارجی استفاده کرده‌اید باید آن را ذکر کنید. در صورت همفکری با افراد دیگر هم باید نام ایشان را در سوال مورد نظر ذکر نمایید.
- پاسخ ارسالی واضح و خوانا باشد. در غیر این صورت ممکن است منجر به از دست دادن نمره شود.
- پاسخ ارسالی باید توسط خود شما نوشته شده باشد. به اسکرین‌شات از منابع یا پاسخ افراد دیگر نمره‌ای تعلق نمی‌گیرد.
- در صورتی که بخشی از سوال‌ها را جای دیگری آپلود کرده و لینک آن را قرار داده باشید، حتما باید تاریخ آپلود مشخص و قابل اعتنا باشد.
- تمام پاسخ‌های خود را در یک فایل با فرمت HW#_[SID]_[Fullname].zip روی کوئرا قرار دهید.

سوال ۱: (نظری) Q-learning (۱۵ نمره)

در جدول زیر یک ربات با شروع از خانه (۰،۰) در سمت پایین ـ چپ می‌خواهد به خانه (۳،۳) در بالا ـ راست برسد. به این منظور درهر گام می‌تواند یکی از حرکات بالا، پایین، چپ و راست را انجام دهد. درصورت موفقیت پاداش 1 و در صورت افتادن به چاله وسط جدول، پاداش -1 را دریافت می‌کند.

		+1
	-1	
Start		

(آ) برای سه اپیزودی که در زیر آمده‌اند به‌روزرسانی‌هایی که الگوریتم *Q-learning* انجام می‌دهد را به‌دست آورید. فرض کنید $\alpha = 0.5$ و $\gamma = 1$ و مقدار اولیه جدول Q صفر است. مواردی که هیچ تغییری نمی‌کنند نیاز نیست ذکر شوند.

Episode 1: (0,0), (1,0), (2,0), (2,1), (1,1)
Episode 2: (0,0), (0,1), (0,2), (1,2), (2,2)
Episode 3: (0,0), (1,0), (2,0), (2,1), (2,2)

(ب) فرض کنید از الگوریتم *Approximate Q-learning* استفاده می‌کنیم و تابع Q را به‌صورت تابع خطی از چهار ویژگی "در ستون اول هستیم"، "در ستون دوم هستیم"، "در ستون سوم هستیم" و "در سطر بالایی هستیم" (مقدار هریک از این چهار ویژگی‌ها ۰ یا ۱ است) نمایش داده‌ایم. به‌روزرسانی‌هایی که الگوریتم در وزن ویژگی‌ها در دو اپیزود اول انجام می‌دهد را مشخص کنید. مواردی که هیچ تغییری نمی‌کنند نیاز نیست ذکر شوند.

سوال ۲: (نظری) مقایسه روش های Value-based (۲۰ نمره)

همان طور که می‌دانیم در روش های یادگیری Temporal Difference و Monte Carlo نیازی به داشتن مدلی از محیط نداریم، یعنی از قبل مشخص نیست که با انجام یک عمل، با چه احتمالی به کدام حالت می‌رویم و چه پاداشی می‌گیریم و به همین دلیل به این روش‌ها model-free می‌گویند. در مورد این دو روش به سوالات زیر پاسخ دهید.

(آ) به نظر شما کدام یک از این دو روش برای مسائل گسسته[1] و کدام یک برای مسائل پیوسته[2] مناسب است؟

[1] episodic
[2] continuous

(ب) تعداد دفعات به روز رسانی ارزش حالت‌ها در کدام روش بیشتر است؟ تشریح دهید.

(ج) فرض کنید نتایج حاصل از ۳ اپیزود یک مسئله اپیزودیک به صورت زیر بوده است و دنباله حالات و پاداش های زیر تاکنون به دست آمده است: (حروف نشان‌دهنده حالت‌ها هستند و پس از آن‌ها پاداش به دست آمده به صورت یک عدد نوشته شده است)

Episode1: A, 0, B, 0, C, 0, D, 1, T
Episode2: B, 1, C, 1, T
Episode3: D, 0, T

فرض کنید ارزش اولیه همه حالت‌ها صفر و $\alpha = 0.2$ ، $\gamma = 0.9$ باشند ارزش حالات A, B, C, D را پس از این اپیزودها با دو روش TD(0) و Monte Carlo بدست آورید.

سوال ۳: (نظری) Deep Q-Learning (۲۵ نمره)

ساختار جدولی: اگر فضای حالت و تعداد action ها در Q-Learning به اندازه کافی کم باشد، به سادگی می‌توان برای تمامی جفت (s, a) های موجود Q*(s, a) را در یک جدول Q(s, a) نگهداری کرد. در این حالت با داشتن یک نمونه تجربه (s, a, r, s') ، قانون بروزرسانی جدول به صورت زیر است:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a' \in \mathcal{A}} Q(s',a') - Q(s,a) \right) \quad (۱)$$

به طوری که: $\alpha > 0$ ، $\gamma \in [0,1)$

تقریب تابع: اغلب به علت فضای حالت بزرگ و action های زیاد ما منطقا توانایی یادگیری و ذخیره یک Q-value برای هر جفت state و action را نداریم. و به جای آن ما Q value را با تابع $\hat{q}(s,a;w)$ تخمین می‌زنیم، w در این تابع پارامتر قابل یادگیری است (معمولا پارامترهای وزن‌ها و بایاس شبکه‌های عصبی) در این حالت قانون بروزرسانی به صورت زیر است:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \left(r + \gamma \max_{a' \in \mathcal{A}} \hat{q}(s',a';\mathbf{w}) - \hat{q}(s,a;\mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{q}(s,a;\mathbf{w}). \quad (۲)$$

به عبارت دیگر هدف ما کمینه کردن تابع هدف زیر است:

$$L(\mathbf{w}) = \mathop{\mathbb{E}}_{s,a,r,s' \sim \mathcal{D}} \left[\left(r + \gamma \max_{a' \in \mathcal{A}} \hat{q}(s',a';\mathbf{w}) - \hat{q}(s,a;\mathbf{w}) \right)^2 \right] \quad (۳)$$

Target Network: مقاله DeepMind (۱ و ۲) دو مجموعه پارامتر w (برای محاسبه $\hat{q}(s,a)$) و w^- (برای محاسبه $\hat{q}(s',a')$، target network) را استفاده می‌کند و قانون بروزرسانی به صورت زیر بازنویسی می‌شود:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha \left(r + \gamma \max_{a' \in \mathcal{A}} \hat{q}(s',a';\mathbf{w}^-) - \hat{q}(s,a;\mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{q}(s,a;\mathbf{w}) \quad (۴)$$

و همچنین تابع هدف نیز به صورت زیر بازنویسی می‌شود:

$$L^-(\mathbf{w}) = \mathop{\mathbb{E}}_{s,a,r,s' \sim \mathcal{D}} \left[\left(r + \gamma \max_{a' \in \mathcal{A}} \hat{q}(s',a';\mathbf{w}^-) - \hat{q}(s,a;\mathbf{w}) \right)^2 \right] \quad (۵)$$

در هر C تا iteration پارامتر های target network بروززسانی شده و با پارامترهای Q-network یکسان میشود (کپی) و در طول این C تا iteration ثابت و بدون تغییر نگهداری می‌شوند.

Replay Memory: در زمان بازی (مثلا آتاری)، هر تجربه (s, a, r, s') در بافر D ذخیره‌می‌شود و در حین ذخیره تجربه‌های جدید، قدیمی‌ها حذف می‌شوند، برای بروزرسانی پارامترها ما یک minibatch از بافر نمونه می‌گیریم و یک بروزرسانی به روش گرادیان کاهشی تصادفی[3] انجام می‌دهیم.

ϵ-Greedy Exploration Strategy: با هدف اکتشاف[4]، از روش ϵ-greedy استفاده می‌کنیم و با احتمال ϵ به صورت رندوم یک action از مجموعه action های A انتخاب می‌شود و با احتمال $1-\epsilon$ به صورت حریصانه[5] یک action انتخاب می‌شود ($argmax_{a \in \mathcal{A}} \hat{q}(s,a,w)$) مقاله DeepMind در طول یک میلیون step اولیه مقدار ϵ را بین 1 به 0.1 تغییر و انعطاف میدهد. در زمان آزمون، عامل با احتمال $\varepsilon_{\text{soft}} = 0.05$ یک عمل تصادفی انتخاب می‌کند.

ساختار شبکه: شبکه Deep Q network مقاله DeepMind به عنوان ورودی حالت s را ورودی گرفته و به عنوان خروجی یک بردار به ابعاد $|A|$ را خروجی میدهد ($\hat{q}(s;w) \in R^{|A|}$).

[3] stochastic gradient descent
[4] exploration
[5] greedy

در ادامه قصد داریم برخی از ملاحظات تئوری مرتبط با تنظیم هایپرپارامتر C را بررسی کنیم (هدف بررسی حد افراط و حد تفریط در مقدار C است)، C فرکانس بازنگری وزن‌های w^- متعلق به target network جهت بروزرسانی و تطابق با وزن‌های w متعلق به Q network است، در حالت اول، هر بار که Q network بروزرسانی می‌شود، target network نیز بروزرسانی می‌شود؛ واضح است که این حالت منجر به بدون استفاده شدن شبکه target network خواهد شد. در حالت دوم، وزن‌های target network می‌تواند طی تمامی مراحل و iteration های آموزش ثابت بماند.

یادآوری: گرادیان کاهشی تصادفی تابع هدف به فرم $J(\mathbf{w}) = \mathbb{E}_{x\sim\mathcal{D}}[l(x, \mathbf{w})]$ را توسط بروزرسانی روی نمونه داده‌ها به صورت زیر کمینه می‌کند:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha\nabla_{\mathbf{w}} l(x, \mathbf{w}) \tag{۶}$$

گرادیان کاهشی تصادفی ویژگی های تئوری مطلوبی دارد. به طور خاص، تحت فرضیاتی، نشان داده شده است که به یک بهینه محلی همگرا می شود. در سؤالات زیر شرایطی را بررسی خواهیم کرد که تحت آن Q-Learning یک بروزرسانی گرادیان کاهشی تصادفی را تشکیل می دهد.

(آ) مزیت اصلی استفاده از Q Function به صورت $\hat{q}(s; w) \in R^{|A|}$ چیست؟

(ب) اولین مورد از این دو حد را در نظر بگیرید: Q learning استاندارد، بدون target network، که به روز رسانی وزن آن در رابطه ۲ بالا ارائه شده است. آیا این بروزرسانی وزن نمونه ای از گرادیان کاهشی تصادفی (تا حداکثر ضریب ثابت ۲) روی تابع هدف L(w) داده شده توسط رابطه ۳ است؟ پاسخ بله یا خیر خود را به صورت ریاضی استدلال کنید.

(ج) اکنون دومین مورد از این دو حالت افراطی را در نظر بگیرید: استفاده از یک target network که هرگز بروزرسانی نمی‌شود (یعنی در طول آموزش ثابت نگه داشته می‌شود). در این مورد، بروزرسانی وزن در رابطه ۴ بالا داده شده است، و w^- را به عنوان یک ثابت در نظر می گیرد. آیا این بروزرسانی وزن، نمونه ای از گرادیان کاهشی تصادفی (تا حداکثر ضریب ثابت ۲) روی هدف $L^-(w)$ است که در رابطه ۵ ارائه شده است؟ پاسخ بله یا خیر خود را به صورت ریاضی استدلال کنید.

(د) یک نقطه ضعف آشکار برای ثابت نگه داشتن وزن‌های target network در طول آموزش این است که بستگی به دانستن وزن های خوب برای target network از قبل دارد (priori). اما اگر اینطور بود، ما اصلاً نیازی به آموزش Q network نداشتیم! با توجه به این موضوع، همراه با بحث بالا در مورد همگرایی گرادیان کاهشی تصادفی و پاسخ های شما به دو بخش قبلی، trade off اساسی در تعیین یک انتخاب خوب برای C را شرح دهید.

(ه) در یادگیری بانظارت، هدف معمولاً به حداقل رساندن خطای یک مدل پیش‌بینی در داده‌های نمونه‌گیری شده از برخی توزیع‌ها است. اگر یک مسئله رگرسیون را با یک خروجی تک بعدی حل کنیم و از میانگین مربع خطا برای ارزیابی عملکرد استفاده کنیم و تابع هدف به صورت زیر نوشته میشود:

$$L(\mathbf{w}) = \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}}\left[(y - f(\mathbf{x}; \mathbf{w}))^2\right] \tag{۷}$$

که در آن x ورودی است، y خروجی است که باید از x پیش‌بینی شود، D مجموعه‌ای از نمونه‌ها از توزیع مشترک (ناشناخته) x و y است، و f (· ; w) یک مدل پیش‌بینی‌کننده با پارامتر w است.

این تابع هدف بسیار شبیه به تابع هدف DQN ذکر شده در بالا است. این دو سناریو چقدر متفاوت هستند؟ راهنمایی: این مجموعه داده D چه تفاوتی با replay buffer، D استفاده شده در بالا دارد؟

سوال ۴: (عملی) الگوریتم PPO (۴۰ نمره)

تابع هدف در الگوریتم TRPO[۶] به شرح زیر است:

$$maximize_\theta \quad \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\right]$$

$$subject\,to \quad \hat{\mathbb{E}}_t[KL[\pi_{\theta old}(\cdot|s_t), \pi_\theta(\cdot|s_t)]] \leq \delta$$

ایده‌ای که نویسندگان TRPO برای حل تابع فوق پیشنهاد کردند، استفاده از جریمه به جای حل مسئله بهینه سازی با محدودیت بود:

$$maximize_\theta \quad \hat{\mathbb{E}}_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}\hat{A}_t\right] - \beta KL[\pi_{\theta old}(\cdot|s_t), \pi_\theta(\cdot|s_t)]$$

اما انتخاب β که عملکرد خوبی در مسائل مختلف یا حتی در یک مسئله خاص داشته باشد، چالش‌برانگیز است. برای حل این چالش نویسندگان الگوریتم PPO[۷] پیشنهاداتی را ارائه کردند:

- Clipped Surrogate Objective ($\epsilon = 0.2$)
- Fixed KL Penalty Coefficient ($\beta = 3$)
- Adaptive KL Penalty Coefficient ($d_{targ} = 0.01$)

با توجه به موارد مذکور، هریک از موارد بالا (با هایپرپارامترهای داده شده) را برای محیط Lunar Lander[۸] پیاده سازی کنید و نتایج را باهم مقایسه

[6] Trust region policy optimization
[7] Proximal Policy Optimization Algorithms
[8] https://www.gymlibrary.dev/environments/box2d/lunar_lander/

کنید. در صورتی که منابع محاسباتی مناسبی در اختیار دارید آزمایش را به ازای هرکدام چندبار تکرار کنید. در ساخت محیط، پارامتر enable_wind را True در نظر بگیرید.

یکی از معایب الگوریتم PPO، حساسیت به انتخاب هایپرپارامترها است. روشی پیشنهاد دهید تا مقدار ϵ در Clipped Surrogate Objective را به صورت تطبیقی در طول آموزش تعیین کند. روش خود را توضیح داده و سپس پیاده‌سازی کنید. نتیجه بدست آمده را با نتایج قبلی مقایسه کنید. نمودار تغییرات ϵ را در طول آموزش رسم کنید.

سوال ۵: (عملی امتیازی) الگوریتم DQN (۳۰ نمره)

در این سوال قرار است به پیاده‌سازی Deep Q-Learning بپردازید و از آن برای حل چالش‌های موجود در محیط Lunar Lander استفاده کنید. با مراجعه به نوتبوک مورد نظر بخش‌های مشخص شده را تکمیل نمایید و به سوالات پاسخ دهید.