دانشگاه صنعتی شریف
دانشکده‌ی مهندسی کامپیوتر

یادگیری تقویتی
نیم‌سال دوم ۰۳-۰۲
مدرس: محمدحسین رهبان

کوییز دوم — زمان آزمون: ۴۵ دقیقه

۱. سوال اول (۱۵ نمره)

به سوالات زیر پاسخ کوتاه دهید.

۱. تفاوت بین الگوریتم‌های TRPO و PPO چیست؟ برای بررسی آنها میتوانید از تابع هدف شان استفاده کنید.

۲. مشکل اصلی روش‌های random shooting و cross entropy در یادگیری model-based چیست؟ به عبارت دیگر، در چه شرایطی نمی‌توان از آن‌ها بهره برد؟

۳. تفاوت میان الگوریتم‌های open-loop و close-loop چیست؟ مقایسه‌ای کوتاه بین این دو ارائه دهید.

حل:

۱. الگوریتم TRPO با استفاده از روش Trust Region Optimization و یک بهینه سازی مقید که میزان KL Divergence را محدود میکند پالیسی بهینه را می یابد. **(۲.۵ نمره)**
در مقابل الگوریتم PPO با کلیپ کردن مستقیم نسبت پالیسی جدید به پالیسی فعلی در حین آپدیت، مابین $[1-\epsilon, 1+\epsilon]$ به شکل بهتر و بهینه تری پالیسی بهینه را می یابد. **(۲.۵ نمره)**

۲. این دو روش در ابعاد بالا دچار نفرین ابعاد شده و کارامد نیستند. **(۵ نمره)**

۳. روش Open Loop با دریافت اولین state دسته ای از action های پیش رو را مشخص میکند .**(۲.۵ نمره)**
اما در الگوریتم های Close Loop پس از محاسبه action متناسب با اولین State و دریافت Feedback، به ازای state های بعدی نیز همین ارتباط با محیط حفظ می شود. **(۲.۵ نمره)**

۲. سوال دوم (۲۰ نمره)

نشان دهید رابطه زیر در الگوریتم Critic Actor Soft برقرار است. برای این کار میتوانید از روش های معرفی شده برای بهینه سازی محدب که در ابتدای درس معرفی شدند استفاده کنید. [راهنمایی: روابط مورد نیاز در انتهای سوالات قابل مشاهده هستند.]

$$\pi^*(a|s) = \frac{\exp Q(s,a)}{\sum_{a'} \exp Q(s,a')} \tag{۱}$$

حل:

ابتدا از محل پیوست ها فرمول V را به شکل زیر باز میکنیم **(۵ نمره)**:

$$V^{\pi}(s) = \sum_{a}[\pi(a \mid s) \cdot Q(s,a) - \alpha \cdot \pi(a \mid s) \cdot \log \pi(a \mid s)] \tag{۲}$$

در ادامه با توجه به اینکه مجموع احتمالات اکشن بعدی در یک استیت باید ۱ باشد، فرم لاگرانژین زیر را تشکیل میدهیم **(۵ نمره)**:

$$\text{Max}_{\pi}\left[\sum_{a}[\pi(a \mid s) \cdot Q(s,a) - \alpha \cdot \pi(a \mid s) \cdot \log \pi(a \mid s)]\right] \quad \text{S.T: } \sum_{a} \pi(a \mid s) = 1 \tag{۳}$$

$$L = \sum_{a}[\pi(a \mid s) \cdot Q(s,a) - \alpha \cdot \pi(a \mid s) \cdot \log \pi(a \mid s)] - \lambda \cdot \left(\sum_{a} \pi(a \mid s) - 1\right) \tag{۴}$$

سپس برای حل این معادیه لاگراژ، نسبت به پالیسی مشتق گرفته و برابر با صفر میگذاریم **(۵ نمره)**:

$$\nabla_{\pi(a_0|s)} L = Q\left(s, a_0\right) - \log \pi^{*}\left(a_0 \mid s\right) - \lambda - 1 = 0 \tag{۵}$$

$$Q\left(s, a_0\right) = \log \pi^{*}\left(a_0 \mid s\right) + c \tag{۶}$$

در ادامه دو طرف معادله به دست آمده را به توان e میرسانیم **(۲.۵ نمره)**:

$$\Rightarrow \pi^{*}(a \mid s) = ce^{Q(s,a)} \tag{۷}$$

در نهایت مقدار c باید طوری تعیین شود تا قید ابتدایی مساله ارضا شود **(۲.۵ نمره)**:

$$\sum_{a} ce^{Q(s,a)} = c \cdot \sum_{a} e^{Q(s,a)} = 1 \Rightarrow c = \frac{1}{\sum_{a} e^{Q(s,a)}} \tag{۸}$$

بنابراین داریم:

$$\Rightarrow \pi^{*}(a \mid s) = \frac{e^{Q(s,a)}}{\sum_{a} e^{Q(s,a)}} \tag{۹}$$

۳. سوال سوم (۱۰ نمره)

فرض کنید وظیفه ترین کردن یک بازو رباتیک بر عهده شماست. محیط آزمایش بسیار داینامیک است و پاداش ها نیز پراکنده (sparse) هستند. چه الگوریتم Off-Policy RL ای را برای این مساله پیشنهاد می‌دهید؟ لطفا دلیل خود را توضیح دهید.

حل:

الگوریتم Soft Actor-Critic انتخالب بهتری است **(۵ نمره)**. ماهیت این الگوریتم در ماکسیمایز کردن آنتروپی Agent را به exploration تشویق کرده و به سازگاری آن با تغییرات کمک میکند **(۵ نمره)**. همچنین همین تشویق

کمک میکند تا با وجود پراکندگی در ریوارد ها، Agent بتواند اکشن ها و مسیر های ارزشمند تر را بیابد. **(۵ نمره)**

۴. سوال چهارم (۳۰ نمره)

فرض کنید شما مسئول توسعه سیستم توصیه‌گر (recommender system) در وب‌سایت ویرگول هستید. وظیفه‌ی این سیستم، پیشنهاد مقاله‌های مناسب به افراد است تا زمان حضور آنها در سایت بیشتر شود. در لحظه‌ی ثبت‌نام یک کاربر جدید در سایت، الگوریتم شما مسئول چیدمان صفحه اصلی برای او است.

۱. مساله exploration-exploitation را تعریف کرده و نقش آن را در این مساله توصیف کنید.

۲. این مساله را با چه مساله مشابهی مدل سازی میکنید و چرا؟ از درس یادگیری تقویتی کدام الگوریتم ریگورالاریزیشن را برای حل آن پیشنهاد میدهید؟ رابطه پاداش آن را نوشته، اجزا و شهود پشت آن ها را شرح دهید (چگونه به حل این مشکل کمک میکنند).

۳. روش UCB را با epsilon-greedy مقایسه کرده و به معایا و مزایا هر یک اشاره کنید

حل:

۱. هاAgent در حالی که از دانش فعلی خود برای کسب حداکثر ریوارد استفاده می کنند، باید محیط را برای کشف action های جدید و بالقوه بهتر نیز جست و جو کنند.**(۲ نمره)** در این مساله ممکن است سیستم با چک کردن اولین پیشنهاد و گرفتن ریوارد تا ابد به همان بچسبد و دیگر موضوعاتی که ممکن است حتی فرد علاقه بیشتری به آنها نشان دهد را امتحان نکند. پس این تریدآف این گونه رخ میدهد که مدل ما علاوه بر پیشنهاد دادن موضوعات با سابقه مثبت نباید شانس را از مابقی موضوعات کمتر امتحان شده بگیرد. **(۳ نمره)**

۲. این مساله را میتوان با Multi-Arm Bandit مدل سازی کرد **(۲ نمره)**. زیرا امتحان هر موضوع حکم فشردن یکی از بازو ها را دارد و باید پر سود ترین آن ها را بیابیم **(۳ نمره)**. الگوریتم مناسب برای حل این مساله UCB یا Upper Confidence Bounds می باشد **(۲ نمره)**.

$$a_t = arg\max_a \left(\hat{r}_a + \sqrt{\frac{2 * \log T}{n_a}} \right)$$

(۲ نمره)
این فرمول ریوارد اصلی را برای جست و جو بهتر ریگولارایزد میکند. در مقدار اضافه شده، n در مخرج کمک میکند تا اکشن های کمتر امتحان شده ریوارد بیشتری را همراه داشته باشند تا شانس انتخاب شدنشان بالاتر رود. **(۳ نمره)** همچنین مقدار log(t) در صورت کسر کمک میکند تا اگر با گذشت زمان همچنان یک اکشن دفعات کمی انتخاب شده بود، بتواند شانسش را افزایش دهد، مگر اینکه مقدار ریوارد خالص آن بسیار کم باشد. **(۳ نمره)**

۳. در روش Epsilon-Greedy عموما با مقداری ثابت و یا کاهشی در طول زمان مواجه هستیم که به صورت کلی برای محیط مشخص می شود. اما روش UCB این امکان را به ما می دهد تا Exploration را با توجه به شرایط هر State و به صورت خودکار Adjust کنیم. **(۷ نمره)** اما این فرایند نیازمند محاسبه و نگه داری چند متغییرو ثابت c و n, t می باشد که این روش را نسبت به Epsilon-Greedy پیچیده تر می کند. **(۳ نمره)**

روابط مورد نیاز

1. $V^\pi(s) = E_{a\sim\pi(.|s)}\left[Q(s,a) + \alpha H(\pi(.|s))\right]$
2. $Q^\pi(s,a) = r(s,a) + \gamma E_{s'\sim P(.|s,a)}\left[V^\pi(s')\right]$