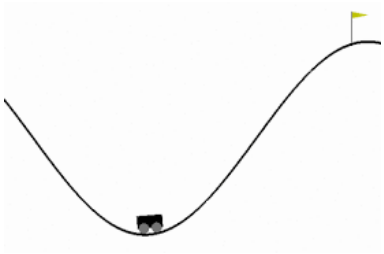
هوش مصنوعی و یادگیری خودرو

تمرین ششم (یادگیری تقویتی)

مرداد ماه ۱۴۰۴

هدف این تمرین آشنایی با الگوریتم‌های گوناگون یادگیری تقویتی و ارزیابی مقایسه‌ای عملکرد آن‌هاست.

به خاطر داریم که بر خلاف یادگیری بامربی که نیازمند یک مجموعه دادهٔ برچسب‌دار است، یادگیری تقویتی به محیطی نیاز دارد که پیاده‌سازی اقدامات عامل هوشمند و دریافت پاداش در آن صورت می‌گیرد. برای این منظور در این تمرین از یکی از محیط‌های شبیه‌سازی gym استفاده خواهید کرد که بستر مناسبی برای پیاده‌سازی الگوریتم‌های کنترلی است (لینک دسترسی به محیط‌های gym).



محیط مورد نظر در این تمرین MountainCar-v0 است که در آن یک خودرو در دره‌ای بین دو تپه قرار دارد. توان موتور این خودرو به اندازه‌ای نیست که بتواند با یک حرکت از تپه بالا برود و ناگزیر است که انرژی لازم برای بالا رفتن از تپه را طی چندین حرکت رفت و برگشتی بدست آورد. برای مقایسهٔ کارآیی روش‌های یادگیری تقویتی با اقدام‌های گسسته و پیوسته، مساله را در این دو حالت بررسی خواهیم کرد. در هر دو حالت، هدف یافتن سیاستی است که در کمترین زمان، خودرو را به پرچم بالای تپهٔ سمت راست برساند.

الف: پیاده سازی محیط

برای استفاده از محیط MountainCar-v0 مراحل زیر را به ترتیب انجام دهید.

مرحله اول:

اگر از محیط های آناکوندا[1] استفاده میکنید، محیط مورد نظر خود را با استفاده از دستور زیر در shell یا bash فعال کنید:

```
conda activate [name of your environment]
```

بعد از آن با استفاده از دستور زیر کتابخانه مورد نیاز را نصب کنید:

```
pip install gymnasium
```

پس از نصب میتوانید با استفاده از دستور زیر محیط مورد نظر را وارد کنید:

```
env = gym.make("your environment name")
```

(در صورت نیاز به راهنمایی بیشتر می‌توانید از لینک یاد شده در ابتدای تمرین کمک بگیرید.)

[1] Anaconda

ب: پیاده سازی الگوریتم‌ها

حالت اول: محیط با اقدام‌ها (و فضای حالت)[۲] گسسته

در این حالت اقدام‌ها گسسته فرض می‌شوند و می‌توانند با یکی از مقادیر زیر بیان شوند: 0 به معنای حرکت به سمت چپ، 1 به معنای حرکت نکردن و 2 به معنای حرکت به سمت راست. اما فضای حالت (موقعیت و سرعت خودرو) پیوسته است و با یک آرایهٔ دو متغیره تعریف می‌شود که متغیر اول آن نشان دهنده موقعیت خودرو بر حسب متر (بین 1.2- تا 0.6) و متغیر دوم آن نشان دهندهٔ سرعت خطی خودرو بر حسب متر بر ثانیه (بین 0.07- تا 0.07) است. در الگوریتم‌هایی که با فضای حالت گسسته کار می‌کنند برای گسسته‌سازی هر یک از متغیرهای فضای حالت می‌توان بازهٔ آن متغیر را برای مثال به ۳۰ بخش تقسیم کرد.

در هر گام زمانی، موقعیت و سرعت خودرو با روابط زیر بروزآوری می‌شود:

$$velocity_{t+1} = velocity_t + (action - 1) * force - \cos(3 * position_t) * gravity$$

$$position_{t+1} = position_t + velocity_{t+1}$$

که در آن مقدار نیروی پیشرانه (force) برابر 0.001 و مقدار نیروی جاذبه (gravity) برابر 0.0025 می‌باشد. همچنین در صورت رسیدن خودرو به ابتدا یا انتهای بازهٔ موقعیت (1.2- یا 0.6)، سرعت خودرو بلافاصله به صفر کاهش می‌یابد (مشابه برخورد صلب به دیوار). در ابتدای حرکت، سرعت خودرو صفر است اما موقعیت آن بصورت تصادفی دربازهٔ 0.4- و 0.6- انتخاب می‌شود. توجه داشته باشید که مقادیر سرعت و موقعیت خودرو همواره باید در بازهٔ مجاز خود قرار داشته باشند.

محاسبهٔ پاداش برای هر اقدام عامل هوشمند از اصلی‌ترین چالش‌های یادگیری تقویتی است. در مسالهٔ حاضر برای این کار می‌توان از معیارهای مختلفی استفاده کرد. برای مثال می‌توان به ازای هر واحد زمانی که از شروع حرکت می‌گذرد اگر خودرو هنوز به پرچم نرسیده باشد یک امتیاز منفی برای عامل هوشمند در نظر گرفت.

در دو صورت زیر فرآیند یادگیری متوقف می‌شود و محیط بازنشانی میشود:

۱. موقعیت خودرو از عدد 0.5 فراتر رود که به این معنی است که خودرو به پرچم رسیده است.
۲. تعداد گام‌های حرکتی خودرو از 200 بیشتر شود (حداکثر تعداد گام در یک مرتبه یادگیری).

پس از آماده‌سازی محیط، الگوریتم‌های زیر را روی آن پیاده کنید:

الگوریتم یادگیری Q

این الگوریتم برای مسایلی با فضای حالت و اقدام‌های گسسته مناسب است. برای پیاده‌سازی آن روش‌های گوناگونی وجود دارد، از جمله با برداشتن گام‌های زیر:

۱. وارد کردن محیط از طریق کتابخانهٔ gym؛
۲. گسسته‌سازی فضای حالت. ساده‌ترین روش برای این کار، روش خطی است که برای اجرای آن می‌توانید از np.linspace استفاده کنید؛

[2] Actions and state space

۳. تشکیل جدول Q ؛
۴. نوشتن حلقهٔ یادگیری Q. این حلقه دارای هایپرپارامترهای متعددی است، از جمله نرخ یادگیری، ضریب کاهش، نرخ جستجو، ضریب کاهشِ نرخ جستجو، حد پایین نرخ جستجو، حداکثر تعداد مراحل و حداکثر تعداد تلاش‌ها در هر مرحله. مقادیر زیر به عنوان نمونه پیشنهاد می‌شوند و طبعا قابل تغییر هستند.
   - نرخ یادگیری: alpha=0.1
   - ضریب کاهش: gamma=0.99
   - نرخ جستجو: epsilon=1
   - ضریب کاهش نرخ جستجو: epsilon_decay=0.995
   - حد پایین نرخ جستجو: epsilon_min=0.1
   - حداکثر تعداد مراحل: n_episodes=10000
   - حداکثر تعداد تلاش‌ها در هر مرحله: max_steps=200
۵. اجرای الگوریتم با استفاده از جدول Q (در پایان یادگیری) و مشاهدهٔ نتایج بصورت انیمیشن با استفاده از زیر تابع render .

الگوریتم یادگیری Q دوگانه (Double Q-Learning)

این الگوریتم یک نسخهٔ بهبود یافته از الگوریتم کلاسیک یادگیری Q است که هدف آن پیشگیری از تخمین بیش‌ازحد مقدار Q در الگوریتم کلاسیک است. در یادگیری Q کلاسیک، مقدار Q با استفاده از "بیشینهٔ Q برای حالت بعدی" به‌روزرسانی می‌شود، اما چون همان Q بطور همزمان برای انتخاب اقدام‌ها و ارزیابی آن‌ها استفاده می‌شود، مقدار Q بصورت خوش‌بینانه‌ای زیاد تخمین زده می‌شود، چرا که گاه ممکن است دریافت پاداشِ کمتر در لحظهٔ فعلی، پاداشِ بیشتری را در آینده در پی داشته باشد. در الگوریتم Q دوگانه، **دو** جدول Q ساخته می‌شود: یکی برای انتخاب اقدام بعدی و دیگری برای ارزیابی آن. در هر گام، به‌صورت تصادفی یکی از دو جدول Q به‌روزرسانی می‌شود و از دیگری برای محاسبهٔ مقدار هدف استفاده می‌شود. این جداسازی باعث کاهش سوگیری و بهبود پایداری آموزش مدل می‌شود. برای پیاده‌سازی این الگوریتم نیز روش‌های گوناگونی وجود دارد، از جمله با برداشتن گام‌های زیر:

۱. گام‌های ۱ و ۲ الگوریتم قبلی؛
۲. ساخت دو جدول Q ؛
۳. نوشتن حلقهٔ یادگیری Q. تفاوت این مرحله فقط در مرحلهٔ بروزآوری جدول هاست که برای انجام آن می‌توانید از الگوریتم معرفی شده در این مقاله استفاده کنید. این حلقه نیز دارای هایپرپارامترهای متعددی است، از جمله نرخ یادگیری، ضریب کاهش، نرخ جستجو، ضریب کاهشِ نرخ جستجو، حد پایین نرخ جستجو، حداکثر تعداد مراحل و حداکثر تعداد تلاش‌ها در هر مرحله. به عنوان نمونه می‌توانید از مقادیر پیشنهاد شده در الگوریتم Q استفاده کنید.
۴. اجرای الگوریتم با استفاده از دو جدول Q (در پایان یادگیری) و مشاهدهٔ نتایج بصورت انیمیشن با استفاده از زیر تابع render .

الگوریتم DQN (Deep Q-Network)

الگوریتم DQN نسخه‌ای از الگوریتم یادگیری Q است که به‌جای استفاده از جدول Q ، از یک شبکهٔ عصبی عمیق برای تقریب تابع Q استفاده می‌کند. این روش برای محیط‌هایی با فضای حالت پیوسته یا بسیار بزرگ مناسب است که در آن‌ها ذخیرهٔ Q برای هر حالت-اقدام ممکن نیست.

در DQN ورودی شبکهٔ عصبی وضعیت محیط (حالت) و خروجی، مقدار Q برای اقدام‌های ممکن در آن حالت است. برای آموزش شبکه، از حافظهٔ بازپخش[3] استفاده می‌شود که شامل داده‌هایی از تجربه‌های گذشتهٔ عامل در قالب (state, action, reward, next_state, done) است. ضرایب وزنی شبکه با کمینه‌سازی میانگین مجذور خطا (MSE) بین مقدار پیش‌بینی‌شدهٔ Q و مقدار هدف که با عبارت زیر تعریف می‌شود بروزآوری می‌شوند:

$$Target = Q_{target}(s', a) \max_{a}.\gamma + r$$

برای پیاده‌سازی این الگوریتم روش‌های گوناگونی وجود دارد، از جمله با برداشتن گام‌های زیر:

۱. وارد کردن محیط از طریق کتابخانهٔ gym؛
۲. وارد کردن کتابخانه‌های مورد نیاز برای ساختن شبکهٔ عصبی مورد نظر (برای مثال کتابخانه‌های تورچ یا تنسورفلو)؛
۳. تعیین ساختار شبکه. برای مثال یک شبکهٔ سه لایه، شامل:
   - یک لایه خطی با اندازه ۱۲۸
   - یک لایه خطی با اندازه ۶۴
   - یک لایه خطی با اندازهٔ خروجی شبکه
   - تابع فعال‌سازی ReLU در تمامی لایه‌ها
۴. طراحی حافظهٔ بازپخش. برای مثال:

```python
class ReplayBuffer:
    def __init__(self, capacity=100000):
        self.buffer = deque(maxlen=capacity)

    def push(self, transition):
        self.buffer.append(transition)

    def sample(self, batch_size):
        transitions = random.sample(self.buffer, batch_size)
        return map(np.array, zip(*transitions))

    def __len__(self):
        return len(self.buffer)
```

۵. نوشتن حلقهٔ یادگیری Q. این حلقه نیز دارای هایپرپارامترهای متعددی است، از جمله نرخ یادگیری، ضریب کاهش، نرخ جستجو، ضریب کاهشِ نرخ جستجو، حد پایین نرخ جستجو، حداکثر تعداد مراحل، حداکثر تعداد تلاش‌ها در هر مرحله، الگوریتم بهینه‌سازی و نوع تابع خطا. به عنوان نمونه می‌توانید از مقادیر پیشنهاد شده در زیراستفاده کنید.
   - نرخ یادگیری: alpha=0.1
   - ضریب کاهش: gamma=0.99

[3] Replay Buffer

- نرخ جستجو: epsilon=1
- ضریب کاهش نرخ جستجو: epsilon_decay=0.995
- حد پایین نرخ جستجو: epsilon_min=0.01
- حداکثر تعداد مراحل: n_episodes=500
- حداکثر تلاش ها در هر مرحله: max_steps=200
- استفاده از تابع Adam به عنوان الگوریتم بهینه‌سازی با نرخ یادگیری 0.001
- استفاده از تابع هزینهٔ MSE

برای بررسی تاثیر **نوع تابع هزینه**، **الگوریتم بهینه‌سازی** و **تابع پاداش** بر عملکرد عامل هوشمند، برای هر یک از این سه مورد از دو نوع مختلف استفاده کنید (جمعا شش حالت) و نتایج را در جدولی ارایه کنید.

۶. اجرای الگوریتم با استفاده از شبکهٔ تربیت شده و مشاهدهٔ نتایج بصورت انیمیشن با استفاده از زیر تابع render .

حالت دوم: محیط با اقدام‌های پیوسته

در این حالت اقدام‌ها پیوسته هستند و هر اقدام با یک عدد که در بازهٔ مشخصی قرار دارد معرفی می‌شود. در مسالهٔ حاضر، **اقدام** عبارتست از نیروی وارد بر خودرو. این نیرو به جای آنکه مانند حالت قبل تنها ۳ مقدار گسستهٔ ۰ و ۱ و ۲ را بپذیرد یک مقدار پیوسته در بازهٔ صفر تا ۱ را می‌پذیرد که همان نیروی وارد بر خودرو است. فضای حالت (موقعیت و سرعت خودرو) نیز پیوسته است و با یک آرایهٔ دو متغیره تعریف می‌شود که متغیر اول آن نشان دهنده موقعیت خودرو بر حسب متر (بین 1.2- تا 0.6) و متغیر دوم آن نشان دهندهٔ سرعت خطی خودرو بر حسب متر بر ثانیه (بین 0.07- تا 0.07) است.

در هر گام زمانی، موقعیت و سرعت خودرو با روابط زیر بروزآوری می‌شود:

$$velocity_{t+1} = velocity_t + force * 0.0015 - 0.0025 * cos(3 * position_t)$$

$$position_{t+1} = position_t + velocity_{t+1}$$

که در آن مقدار نیروی پیشرانه (force) همان عدد اقدام (عددی پیوسته در بازهٔ صفر تا ۱) است. همچنین در صورت رسیدن خودرو به ابتدا یا انتهای بازهٔ موقعیت (1.2- یا 0.6)، سرعت خودرو بلافاصله به صفر کاهش می‌یابد (مشابه برخورد صلب به دیوار). در ابتدای حرکت، سرعت خودرو صفر است اما موقعیت آن بصورت تصادفی دربازهٔ 0.4- و 0.6- انتخاب می‌شود. توجه داشته باشید که مقادیر سرعت و موقعیت خودرو همواره باید در بازهٔ مجاز خود قرار داشته باشند.

در اینجا نیز برای محاسبهٔ پاداش می‌توان از معیارهای مختلفی استفاده کرد. برای مثال می‌توان به ازای هر واحد زمانی که از شروع حرکت می‌گذرد اگر خودرو هنوز به پرچم نرسیده باشد یک امتیاز منفی برای عامل هوشمند در نظر گرفت.

در دو صورت زیر فرآیند یادگیری متوقف می‌شود و محیط بازنشانی میشود:

۱. موقعیت خودرو از عدد 0.5 فراتر رود که به این معنی است که خودرو به پرچم رسیده است.
۲. تعداد گام‌های حرکتی خودرو از 200 بیشتر شود (حداکثر تعداد گام در یک مرتبه یادگیری).

پس از آماده‌سازی محیط، الگوریتم‌های زیر را روی آن پیاده کنید:

الگوریتم DQN

فرآیند پیاده‌سازی DQN دقیقا مانند حالت قبل است، با این تفاوت که برای کاربرد این الگوریتم در مسایلی که اقدام آن‌ها ذاتا پیوسته است باید ابتدا اقدام‌ها گسسته‌سازی شوند. برای مثال با تقسیم بازهٔ اقدام به ۱۰ قسمت، به جای ۳ اقدام گسسته (مانند حالت قبل) این بار ۱۰ اقدام خواهید داشت.

الگوریتم DDPG (Deep Deterministic Policy Gradient)

این الگوریتم که ترکیبی از روش‌های Actor-Critic و یادگیری تقویتی عمیق مانند DQN است یکی از الگوریتم‌های پیشرفتهٔ یادگیری تقویتی برای محیط‌هایی با اقدام‌های پیوسته (مانند MountainCarContinuous-v0) به شمار می‌آید. اجزای تشکیل دهندهٔ این الگوریتم عبارتند از:

- شبکه‌ای که یک سیاست قطعی یاد می‌گیرد و در هر حالت، یک اقدام پیوسته تولید می‌کند (Actor)

- شبکه‌ای که مقدار Q را برای جفت‌های (state, action) تقریب می‌زند (Crtitic)
- حافظهٔ بازپخش (Replay Buffer) برای ذخیرهٔ تجربه‌ها جهت آموزش پایدار
- افزونهٔ نویز به خروجی شبکهٔ actor (مثلاً نویز گوسی یا Ornstein-Uhlenbeck) برای اکتشاف در فضای اقدام‌های پیوسته
- یک شبکهٔ هدف (Target Network) برای هر یک از شبکه‌های actor و critic که بطور جداگانه تعریف شده و به آرامی به‌روزرسانی می‌گردند (soft update)

برای پیاده‌سازی این الگوریتم روش‌های گوناگونی وجود دارد، از جمله با برداشتن گام‌های زیر:

۱. وارد کردن محیط از طریق کتابخانهٔ gym؛
۲. وارد کردن کتابخانه های مورد نیاز برای ساختن شبکهٔ عصبی مورد نظر (برای مثال کتابخانه‌های تورچ یا تنسورفلو)؛
۳. تعیین ساختار شبکه‌های Actor و Critic. برای مثال:

شبکهٔ Actor

- یک لایه خطی با اندازه ۲۵۶ با تابع ReLU
- یک لایه خطی با اندازه ۱۲۸ با تابع ReLU
- یک لایه خطی با اندازه خروجی شبکه با تابع Tanh
- تابع فوروارد مخصوص Actor

شبکهٔ Critic

- یک لایه خطی با اندازه ۲۵۶ با تابع ReLU
- یک لایه خطی برای سر هم کردن خروجی لایه قبلی و خروجی Actor
- یک لایه خطی با اندازه ۱ با تابع ReLU
- تابع فوروارد مخصوص Critic

۴. طراحی حافظهٔ بازگشتی مربوط به شبکهٔ عصبی (مانند DQN)
۵. تعریف تابع نویز (برای مثال تابع نویز Ornstein-Uhlenbeck با mu=0.15 و sigma=0.2)
۶. نوشتن حلقهٔ یادگیری. این حلقه نیز دارای هایپرپارامترهای متعددی است، از جمله ضریب کاهش، ظرفیت حافظه، تعداد مراحل، حداکثر تعداد تلاش‌ها در هر مرحله، الگوریتم بهینه‌سازی و نوع تابع خطا. به عنوان نمونه می‌توانید از مقادیر پیشنهاد شده در زیراستفاده کنید.

- ضریب کاهش: gamma=0.99
- ظرفیت حافظه: buffer_capacity=100000
- تعداد مراحل: n_episodes=1000
- حداکثر تعداد تلاش‌ها در هر مرحله: max_steps=200
- استفاده از تابع Adam به عنوان بهینه‌ساز با نرخ یادگیری 0.001
- استفاده از تابع خطای MSE
- ضریب تاو در soft update: Tau=0.001

ج: ارزیابی عملکرد مدل‌ها

۱- نمودارها و مقایسهٔ بصری

نمودار پاداش به ازای هر مرحله را در طول مدت یادگیری برای هر الگوریتم رسم کنید. (رسم و توضیح نمودارهای دیگری که بتواند به شناخت کاستی‌های فرآیند یادگیری کمک کند نمرهٔ امتیازی خواهد داشت.)

۲- معیارهای ارزیابی مدل

برای هر مدلی که آموزش داده‌اید معیارهای زیر را محاسبه کنید و نتایج را به صورت جدول ارائه دهید تا به مقایسهٔ مدل‌ها کمک کند:

- **میانگین تعداد مرحله‌ها تا رسیدن خودرو به پرچم:**
  بعد از آموزشِ هر مدل ۱۰ بار محیط را با مدلِ آموزش داده شده اجرا کنید و میانگین تعداد گام‌های طی شده تا رسیدن به هدف را برای هر مدل گزارش کنید.

- **زمان آموزش مدل:**
  زمان صرف شده برای آموزش هر مدل را گزارش کنید.

استفاده از معیارهای تکمیلی برای مقایسهٔ عملکرد مدل‌های مختلف نمرهٔ امتیازی خواهد داشت.

۳- تحلیل عملکرد (خلاصه)

به سوالات زیر پاسخ دهید:

- در این مساله کدام مدل دقیق‌ترین پیش‌بینی را ارائه می‌دهد؟ به نظر شما دلیل عملکرد بهتر این مدل چیست؟
- درباره تعادل بین دقت (Accuracy) و زمان آموزش بحث کنید.

چند تذکر:

- برای یادگیری مفاهیمی که در تمرین مطرح شده و در کلاس تدریس نشده‌اند از منابع موجود در اینترنت استفاده کنید.
- برای انجام بخش‌های مختلف تمرین می‌توانید از کتابخانه‌های آمادهٔ پایتون استفاده کنید.
- تحویل گزارش این تمرین ضروری است و به تمرین بدون گزارش نمره‌ای تعلق نمی‌گیرد. حجم گزارش معیاری برای ارزیابی نخواهد بود و لزومی به توضیحات کد نیست؛ اما تحلیل نتایج الزامی است، حتی اگر در صورت پرسش اشاره‌ای به آن نشده باشد.
- در فرآیند ارزیابی گزارش، کدهای شما لزوما اجرا نخواهد شد. بنابراین همهٔ نتایج و تحلیل‌های خود را به طور کامل ارائه کنید.
- شباهت بیش از حد گزارش‌ها و کدها باعث از دست دادن نمرهٔ تمرین خواهد شد. همچنین گزارش‌هایی که در آن‌ها از کدهای آماده استفاده شده باشد پذیرفته نخواهد شد.
- تنها زبان برنامه نویسی مجاز Python است.
- کدها می‌توانند در قالب نوت بوک یا به صورت ماژولار در قالب فایل‌های پایتون تهیه شوند. در پایان کار باید تمامی کدها اجرا شده و خروجی هر سلول در فایلی ذخیره و به همراه گزارش ارسال شود. برای مثال اگر خروجی یک سلول یک نمودار است، این نمودار باید هم در گزارش و هم در نوت‌بوک کدها وجود داشته باشد.
- لطفا گزارش، کدها و سایر پیوست‌ها را در یک پوشه با نام زیر قرار داده و پس از فشرده سازی، در سامانهٔ Elearn بارگذاری کنید.

HW6_Last Name_Student Number.zip

- پرسش‌های خود را از طریق ایمیل یا تلگرام از دستیار آموزشی مربوطه بپرسید:

	ایمیل	تلگرام
محمد مهدی عابدی	mohammadabedi@ut.ac.ir	@mohammadabedi1179