Interactive Learning
Homework 4

Soleiman BashirGanji
810103077

Question 1.

دو رویکرد ذکر شده در سوال، رویکردهای اصلی یادگیری تقویتی هستند. در رویکرد policy-based ما مستقیم به یادگیری سیاست بهینه میپردازیم اما در رویکرد value-based هدف ما یادگیری یک تابع ارزش برای هر حالت یا هر جفت حالت-عمل است.

در رویکرد value-based ما ارزش هر عمل را در هر حالت یا ارزش کلی یک حالت را بر اساس پاداش‌تجمعی حاصل از انجام آن عمل در آن حالت (یا بودن در آن حالت) محاسبه میکنیم. اینجا سیاست ما یک سیاست حریصانه بر اساس مقادیر محاسبه شده خواهد بود (به صورت غیر مستقیم بدست می‌آید).

ولی در رویکرد policy-based به یادگیری تابعی می پردازیم که به انتخاب هر عمل در یک حالت احتمالی انتصاب می‌کند، درواقع اینجا خروجی یادگیری ما مستقیما یک سیاست است.

- رویکردهای value-based در مسائل گسسته به خوبی عمل می‌کنند. ولی در فضاهای پیچیده و پیوسته می توانند دچار مشکل شوند. از طرفی دیگر رویکردهای policy-based میتوانند گزینه‌ی مناسبی برای مسائل پیوسته و پیچیده باشند چرا که ذاتا stochastic هستند.

- ایجاد تعادل exploration-exploitation در رویکردهای value-based میتواند چالش بیشتری داشته باشد چرا که سیاست‌های حریصانه جستجو را محدود می‌کنند. اما برقراری این تعادل در رویکردهای policy-based آسان‌تر است چرا که ذاتا stochastic هستند و جستجو را تشویق می‌کنند.

- پیاده‌سازی رویکردهای value-based معمولا آسان‌تر است و همچنین معمولا به نمونه‌های کمتری برای یادگیری نیاز دارند.

با توجه به مطالب کلاس، نظر من در مورد فواید اصلی یادگیری مستقیم سیاست:

در مسائل پیوسته و پیچیده، تخمین‌گر ما معمولا یک شبکه‌ی عصبی‌است که ورودی آن می‌تواند حاوی اطلاعات متفاوتی باشد. یکی از اصلی‌ترین مزایای این رویکرد برای ما این است که می‌توانیم بخش زیادی از اهداف و خواسته‌های خود را با طراحی "۱.ورودی" و "۲.تابع هزینه" این شبکه عصبی در مسئله اعمال کنیم. اینجا عامل ما سعی در یادگیری بهترین سیاست با توجه به تمام اطلاعات موجود در مسئله دارد و لزوما تلاش در تخمین ارزش یک حالت یا عمل-حالت بر اساس پاداش را ندارد.

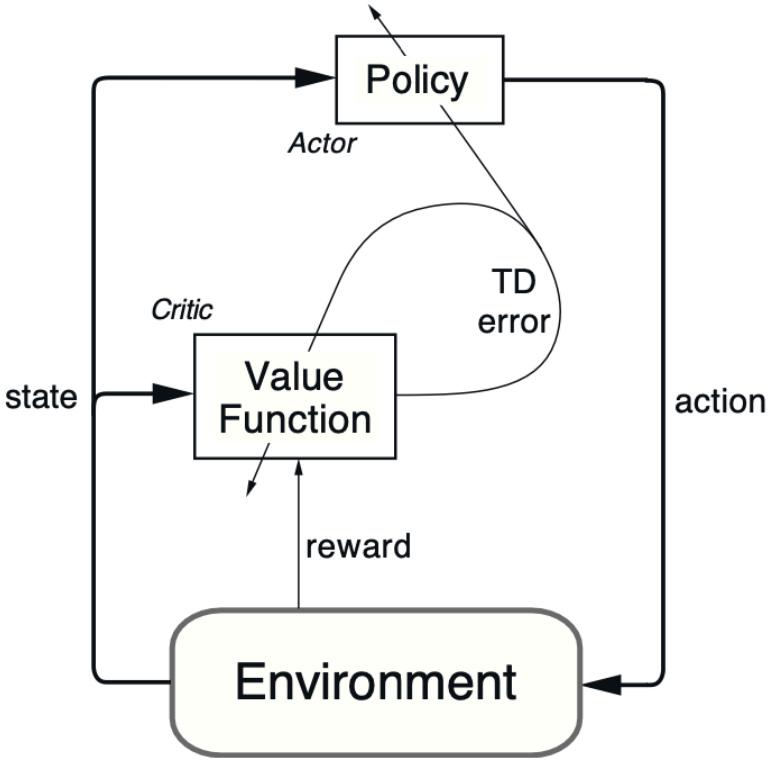
مثال‌هایی از این امر شامل کمینه کردن free energy یا پیاده‌سازی عامل‌هایی با پاداش‌های درونی متفاوت (prospect theory) است.

یکی دیگر از مزایای مهم تمرکز بر سیاست، توانایی مقایسه و سنجش مستقیم سیاست‌های مختلف است. برای مثال در یادگیری اجتماعی، ما می‌توانیم سیاست خود را مستقیما با سیاست دیگران مقایسه کنیم و از تجارب دیگران در یادگیری خود بهره‌مند شویم. این کار با رویکرد value-based میتواند بسیار دشوار باشد چرا که تمرکز ما یادگیری value است ولی در محیط سوشال معمولا فقط اعمال و حالات عامل‌های دیگر را می‌بینیم و از پاداش آن اعمال بی‌خبریم. در روش

policy-based با داشتن مدل محیط، می‌توان سیاست عامل‌های دیگر را صرفا با داشتن اعمال آن‌ها تخمین زد ولی تخمین value به این راحتی نیست.
در حالت کلی، بنظر میرسد در محیط‌های پیوسته و پیچیده، رویکردهای policy-based اکثرا گزینه‌ی مناسب‌تری باشند و رویکردهای value-based باید با بررسی محیط، هدف و مسئله انتخاب شوند تا گزینه‌ی مناسبی باشند.

روش ترکیبی:

در یادگیری تقویتی روش‌های ترکیبی وجود دارند که رویکردهای مبتنی بر سیاست (policy-based) و مبتنی بر ارزش (value-based) را با هم ترکیب می‌کنند. این روش‌ها تلاش می‌کنند تا از نقاط قوت هر دو رویکرد بهره ببرند و نقاط ضعف آن‌ها را کاهش دهند. یکی از کلاس‌های برجسته این روش‌ها چارچوب بازیگر-منتقد (Actor-Critic) است.
روش‌های بازیگر-منتقد (Actor–Critic) از جمله روش‌های TD هستند که دارای ساختار حافظه‌ای جداگانه برای نمایش صریح سیاست به‌طور مستقل از تابع ارزش می‌باشند. ساختار مربوط به سیاست "بازیگر" نامیده می‌شود، زیرا برای انتخاب اقدامات استفاده می‌شود، و تابع ارزش تخمینی "منتقد" نامیده می‌شود، زیرا اقدامات انجام‌شده توسط بازیگر را نقد می‌کند. یادگیری همیشه مبتنی بر سیاست است؛ منتقد باید درباره سیاستی که بازیگر در حال حاضر دنبال می‌کند، بیاموزد و آن را نقد کند. این نقد به شکل یک خطای TD بیان می‌شود. این سیگنال تنها خروجی منتقد است و تمام فرایند یادگیری در هر دو بازیگر و منتقد را هدایت می‌کند.
روش‌های بازیگر-منتقد گسترش طبیعی ایده Gradient-Bandit Methods هستند که به یادگیری TD و مسئله کامل یادگیری تقویتی تعمیم داده شده‌اند. معمولاً منتقد یک تابع ارزش-حالت است. پس از هر انتخاب اقدام، منتقد وضعیت جدید را ارزیابی می‌کند تا مشخص کند آیا شرایط بهتر یا بدتر از انتظارات پیش رفته است. این ارزیابی همان خطای TD است.



Question 2.

خروجی الگوریتم DQN یک تابع ارزش Q برای عمل a در حالت s است. خروجی الگوریتم DDQN هم همین است. تفاوت اصلی این دو الگوریتم در معماری داخلی شبکه‌ی آن‌ها (قبل از خروجی نهایی) است.
خروجی DDQN حاصل جمع (aggregate) دو عبارت متفاوت است. درواقع این دو عبارت هرکدام خروجی یک تخمین‌گر (estimator) مجزا هستند. برعکس DQN که فقط یک تخمین‌گر برای Q دارد، اینجا دو تخمین‌گر داریم یکی برای Q و یکی برای Advantage function که توضیحات Advantage function بدین شکل است:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$$

اینجا V همان ارزش بودن در حالت s و Q همان ارزش انجام عمل a در حالت s است. درواقع Advantage function ارزش حالت را از ارزش انجام هر عمل در آن حالت کم می‌کند تا یک "ارزش نسبی" برای هر عمل بدست بیاورد.
مسئله‌ای که اینجا مورد توجه است، این است که در بعضی حالات انجام عمل‌های متفاوت تاثیر چندانی در نتیجه ندارد (با دید ایده‌آل مقدار A آن‌ها برابر است). رساله‌ی پیوست شده در این مورد برای مثال عکس محیطی را آورده که در آن عامل ما یک ماشین است و باید از برخورد به موانع دوری کند.
در یک عکس، هیچ چیزی جلوی ماشین نیست و انتظار می‌رود که در این حالت، انجام اعمال مختلف فرق زیادی ایجاد نکند. اما در عکس دیگری، یک ماشین دیگر نزدیک ماشین ما وجود دارد و اینجا انجام عمل درست بسیار مهم است.
در اینجا A به ما کمک می‌کند که حالات ارزشمند را شناسایی کنیم. منظور از حالات ارزشمند حالاتی هستند که اعمال متفاوت تاثیرات مهمی میتوانند داشته باشند. همچنین وجود دو تخمین‌گر باعث می‌شود که شبکه بتواند به صورت مجزا و به خوبی به دو چیز متفاوت توجه بکند و تصمیم نهایی را بر اساس جمع خروجی‌های این دو تخمین‌گر انجام دهد.

Question 3.

رساله‌ی مذکور بستری برای یادگیری تقویتی به صورت هم‌روند ارائه می‌دهد. در این بستر، چندین عامل در کپی‌هایی از یک محیط و روی نخ‌های یک پردازنده شروع به یادگیری میکنند.
این بستر جایگزینی برای روش‌های مبتنی بر experience replay است، درواقع اینجا بجای اینکه یک عامل داشته باشیم که شبکه‌ی خود را با mini-batchهایی از replay buffer آموزش میدهد، چندین عامل داریم که mini-batchهایی از تجارب همه‌ی آن‌ها برای یادگیری استفاده می‌شوند.
رساله‌ی مورد بررسی از عامل‌های متفاوتی با الگوریتم‌های متفاوت استفاده کرده که یکی از این عامل‌ها از الگوریتم A3C پیروی می‌کند که الگوریتم مورد توجه این سوال است.
در سوال قبل، Advantage function برابر بود با:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s)$$

که تلاش در سنجش ارزش عمل a در حالت s، نسبت به ارزش کلی حالت s داشت.
در رساله‌ی مذکور advantage function بدین صورت است:

$$A(s_t, a_t; \theta, \theta_v) = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k}; \theta_v) - V(s_t; \theta_v),$$

اینجا هم نقش advantage function این است که محاسبه‌ی ای از ارزش انجام دادن عمل a در حالت s نسبت به کل ارزش مورد انتظار از حالت s به ما بدهد. اما نحوه‌ی اعمال این نقش تفاوت‌هایی دارد.
اینجا بجای Q از V_t+k استفاده شده‌است که همان ارزش حالت بعدی‌است. این دو نمونه از advantage function ارتباط نزدیکی باهم دارند چرا که ارزش حالت بعدی، همان ارزشی‌است که از انجام عمل a بدست می‌آید. در واقع اینجا تلاش این است که ارزش نسبی حالت بعد نسبت به ارزش کلی تمام حالاتِ بعدی ممکن را بسنجیم، که این به صورت غیر مستقیم Q را هم در بر می‌گیرد.
همچنین اینجا تخمینگری مجزا برای advantage function نداریم و تخمینگر ما یک critique است که V را تخمین میزند. از این V برای محاسبه‌ی advantage function و همچنین loss function استفاده می‌شود.

Question 4.

در این محیط هدف ایجاد تعادل در یک میله‌ی عمودی با حرکت دادن یک ماشین به سمت چپ یا راست است.
اعمال محیط گسسته هستند و میتوانند راست (یک) یا چپ (صفر) باشند. دریافت‌های (observations) ما از محیط شامل چهار المان متفاوت است:
۱. محل قرارگیری ماشین (بین -4.8 تا 4.8)
۲. سرعت ماشین (منفی تا مثبت بی‌نهایت)
۳. زاویه‌ی میله (از 0.418 (-24) rad - تا مثبتِ همین مقدار)
۴. سرعت زاویه‌دار میله (منفی تا مثبت بی‌نهایت)
با توجه به اینکه هدف در این محیط متعادل نگه داشتن میله است، به ازای هر عمل مادامی که میله متعادل باشد پاداش ۱ دریافت میکنیم. منظور از متعادل پایین‌تر ذکر شده است.
شروط پایان اپیزود:
۱. زاویه‌ی میله بیش از ۱۲ یا کمتر از ۱۲- باشد.
۲. مرکز ماشین به لبه‌ی نمایشگر/تصویر برسد.
۳. طول اپیزود بیش از ۵۰۰ بشود.

Question 6.

یکی از دلایل اصلی اینکه الگوریتم‌های کلاسیک برای این محیط مناسب نیستند پیوسته بودن حالات است. گرچه میتوان حالات پیوسته را به گسسته تبدیل کرد اما اینکار مشکلاتی از جمله دقت کم و تعداد حالات بسیار زیاد را دارند.

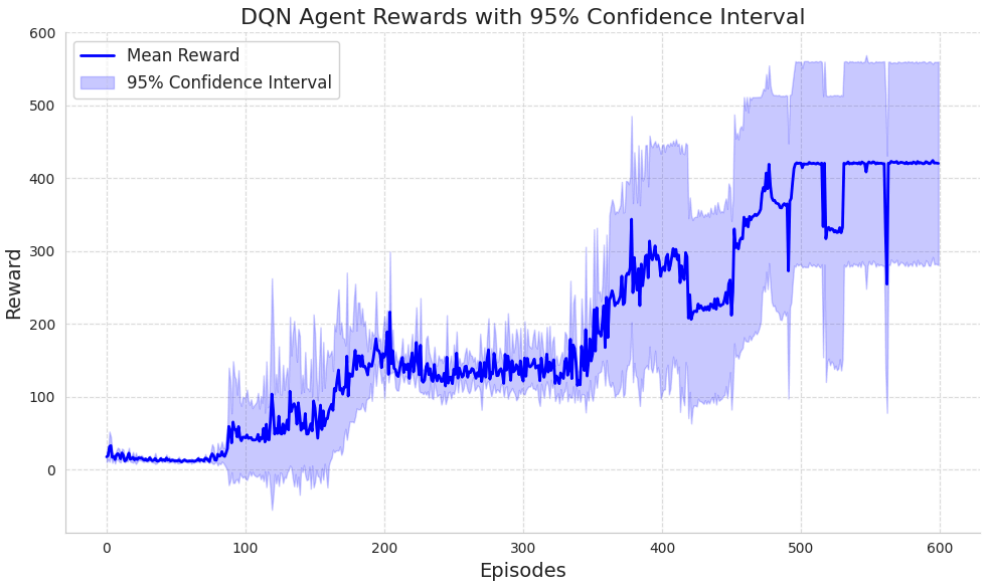
معمولا بجای اینکه سعی در استخراج ویژگی از حالات پیوسته داشته باشیم، بهتر است این کار را به یک function approximator مانند یک شبکه‌ی عصبی بسپاریم.

Question 6.

الف)

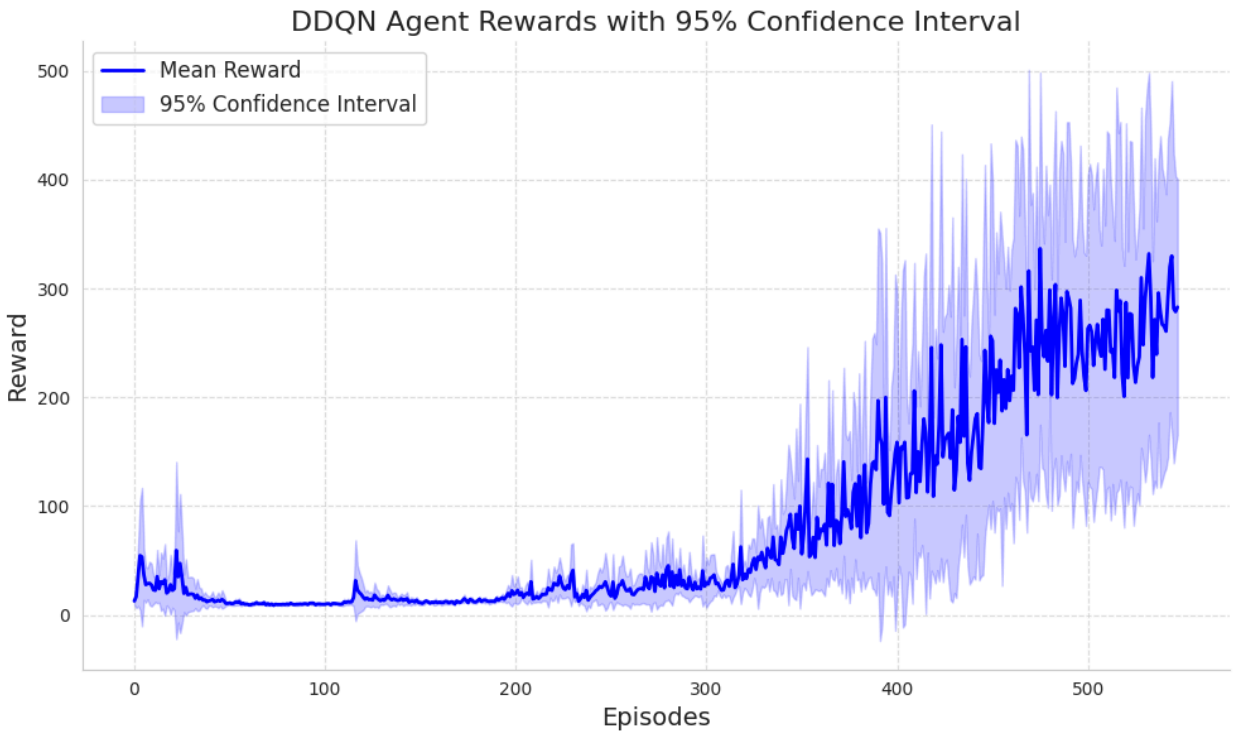
Parameter	Value	Description
BATCH_SIZE	128	Number of transitions sampled from the replay buffer
GAMMA	0.99	Discount factor
EPS_START	0.9	Starting value of epsilon
EPS_END	0.05	Final value of epsilon
EPS_DECAY	1000	Controls the rate of exponential decay of epsilon; higher means a slower decay
TAU	0.005	Update rate of the target network
LR	1e-4	Learning rate of the `AdamW` optimizer

ب)



Question 7.

الف)



ب)

Parameter	Value	Description
GAMMA	0.99	Discount factor for future rewards
EXPLORE	20000	Number of steps over which epsilon is linearly annealed
INITIAL_EPSILON	0.1	Starting value of epsilon (exploration rate)
FINAL_EPSILON	0.0001	Final value of epsilon (exploration rate)
REPLAY_MEMORY	50000	Size of the replay buffer
BATCH	16	Number of experiences sampled from replay memory for each training iteration

lr	1e-4	Learning rate for the optimizer

Question 8.

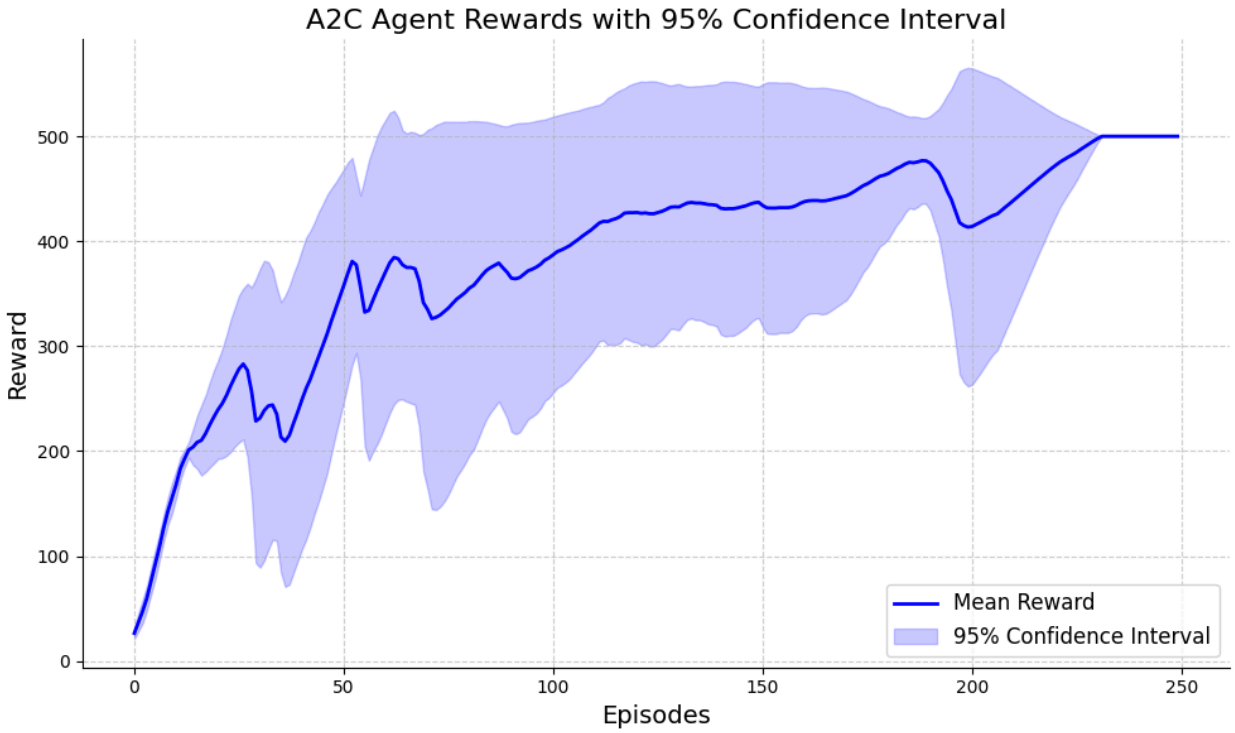
هردو الگوریتم توانسته‌اند تسک مورد نظر را تا مقدار قابل قبولی یاد بگیرند. سرعت همگرایی و میزان همگرایی در الگوریتم DQN بیشتر بوده است، اما بنظر میرسد با افزایش اپیزودهای آموزش، عامل DDQN هم به همگرایی خیلی خوبی برسد.
هردو عامل برای ۶۰۰ اپیزود آموزش دیده‌اند.
شبکه‌ی عامل DDQN لایه‌ها و وزن‌های بیشتری دارد. شاید بتوان گفت که به همین دلیل نیاز به آموزش طولانی‌تری نسبت به DQN دارد. روند سعودی پاداش در نمودار DDQN به وضوح قابل مشاهده‌است و بنظر میرسد فقط با کمی آموزش بیشتر میتواند به سیاست بهینه بسیار نزدیک شود.
همچنین پارامترهای دو عامل و نحوه‌ی کاهش دادن اپسیلون نیز در دوعامل باهم متفاوت هستند و این‌ها میتوانند تفاوت‌هایی را در نحوه‌ی یادگیری دو عامل ایجاد کنند.

Question 9.

الف)

Parameter	Value
learning_rate	0.0007
n_steps	5
gamma	0.99
gae_lambda	1.0
ent_coef	0.0
vf_coef	0.5
max_grad_norm	0.5
rms_prop_eps	1e-05

ب)



متاسفانه نتوانستم بفهمم چگونه می‌توانم پاداش‌های هر اپیزود را از کتابخانه‌ی stable baseline دریافت کنم. گویا کتابخانه تعداد اپیزود را ورودی نمیگیرد و تعداد اپیزودها به صورت غیر مستقیم با n_steps تایین میشود.

در هنگام آموزش اگر پارامتر verbose را یک بگذاریم، در حین آموزش یک سری لاگ پرینت میشود که در آن متوسط پاداش اپیزود نیز وجود دارد. من عامل را ۵ بار و به میزان ۵۰۰۰۰۰ قدم آموزش دادم و با کد خاصی verbose هارا جمع‌آوری کرده و پارس کردم تا پاداش را استخراج کنم:

```
learning_rate=0.0007
ı_steps=5
gamma=0.99
gae_lambda=1.0
ent_coef=0.0
vf_coef=0.5
max_grad_norm=0.5
rms_prop_eps=1e-05


def get_rewards(lines):
  def extract_floats(text):
      import re
      # Regular expression to match float numbers
      float_pattern = r'[-+]?\d*\.\d+|\d+'
      matches = re.findall(float_pattern, text)

      # Convert matches to float
      return [float(match) for match in matches]

  rewards = []
  for line in lines:
    if "ep_rew_mean" in line:
      rewards.append(extract_floats(line)[0])
  return rewards


runs_reward = [[], [], [], [], []]
for run in range(5):

  vec_env = make_vec_env("CartPole-v1", n_envs=4)
  output_buffer = io.StringIO()

  with redirect_stdout(output_buffer):
      model = A2C("MlpPolicy", vec_env, verbose=1, learning_rate=learning_rate, n_s
      model.learn(total_timesteps=500000, progress_bar=True, tb_log_name='A2C')

  captured_output = output_buffer.getvalue()

  lines = captured_output.splitlines()
  runs_reward[run] = get_rewards(lines)
```

Question 10.

میتوان مشاهده کرد که الگوریتم A2C در کمتر از ۲۵۰ اپیزود همگرا شده‌است. همچنین سرعت یادگیری بسیار زیاد بوده و در همان اپیزودهای اول میزان یادگیری قابل توجهی انجام شده‌است.

نتیجتا انتظار میرود میزان حسرت این الگوریتم بسیار کمتر باشد. همچنین از لحاظ سرعت و بهینگی، و با توجه به اینکه الگوریتم روی CPU اجرا شده است، تفاوت چشم‌گیری وجود دارد و میتوان گفت الگوریتم A2C بسیار سریع است و راهکار مناسبی برای یادگیری‌است.

به صورت کلی مشاهدات نشان می‌دهد که این الگوریتم بهترین الگوریتم از بین ۳ الگوریتم استفاده شده در این تمرین است.