دانشکده مهندسی کامپیوتر

پروژه مقطع کارشناسی مهندسی کامپیوتر

ساخت و آموزش مدل حرکتي ربات در مسیرهای پرچالش به کمک یادگیری تقویتي عمیق

پرهام نوران بخت

استاد راهنما:

دکتر سید حسین خواسته

[تاریخ دقیق روز، ماه و سال دفاع]


بسم الله الرحمن الرحیم

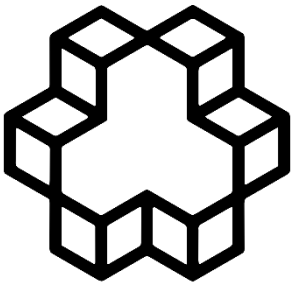
تأییدیّه هیات داوران

اعضای هیئت داوران، نسخه نهائی پروژه خانم / آقای: [نام دانشجو]

را با عنوان: [عنوان پروژه]

از نظر شکل و محتوی بررسی نموده و پذیرش آن را برای تکمیل درجه کارشناسی تأیید می‌کنند.

اعضای هیئت داوران	نام و نام خانوادگی	رتبه علمی	امضاء
۱- استاد راهنما			
۲- استاد داور			



دانشگاه صنعتی خواجه نصیرالدین طوسی

اظهارنامه دانشجو

اینجانب **پرهام نوران بخت** دانشجوی مقطع کارشناسی رشته **مهندسی کامپیوتر** گواهی می‌نمایم که مطالب ارائه شده در این پروژه با عنوان:

ساخت و آموزش مدل حرکتی ربات در مسیرهای پرچالش به کمک یادگیری تقویتی عمیق

با راهنمایی استاد محترم **دکتر سید حسین خواسته** توسط شخص اینجانب انجام شده است. صحت و اصالت مطالب نوشته شده در این پروژه تأیید می‌شود و در تدوین متن پروژه قالب مصوب دانشگاه را به طور کامل رعایت کرده‌ام.

امضاء دانشجو:

تاریخ:

حق طبع، نشر و مالکیت نتایج

تقدیم به:

تشکر و قدردانی

چکیده

همانطور که از عنوان پایان نامه مشخص است هدف این پروژه دستیابی به عاملی است که بتواند در مسیر های پرچالش بهترین عملکرد ممکن را داشته باشد؛ به عبارتی ربات مذکور باید بتواند در مسیر ها با چالش های گوناگون به حرکت خود آنگونه که برایش تعریف شده ادامه دهد. برای دستیابی به این هدف ابتدا باید محیط[۱] های فیزیکی موردنظر را بر روی یک موتور فیزیک پیاده کنیم، سپس با استفاده از الگوریتم[۲] های مناسب یادگیری تقویتی عمیق، عاملی را ساخته و آموزش دهیم که بتواند در ارتباط با این محیط ها به امتیاز قبل قبولی دست یابد. این عامل باید با زبان پایتون[۳] و صرفا با استفاده از کتابخانه‌ی تنسورفلو[۴] و با کمک واسط کراس[۵] طراحی شده باشد.

.۱

در این پروژه یک کتابخانه جدید شامل چند محیط ساخته شده است که امکان برقراری ارتباط با آن ها از طریق واسط جیم[۶] و آموزش و آزمایش عامل ها در این محیط‌ها به کمک جیم و یا ابزار مشابه وجود دارد. با توجه به ماهیت ذاتی پروژه می توان این محیط ها را به دو دسته تقسیم کرد: ۱- محیط های مخصوص آموزش ۲- محیط های غیر قابل استفاده برای آموزش؛ در محیط های آموزشی ویژگی تصادفی بودن از پاداش حاصل از اعمال عامل تا جای ممکن باید حذف شود در ادامه در این باره به طور کامل توضیح داده می شود.

علاوه بر کتابخانه مذکور، یک کتابخانه شامل عامل های پیاده کننده الگوریتم های یادگیری تقویتی عمیق نیز ساخته شده است که در آن دو عامل پایه و ساده A2C و MPO و دو عامل مشتق از A2C یعنی PPO و TRPO وجود دارد. به طور کلی در مقایسه با مدل های مشابه در محیط های یکسان و یا شبیه محیط

---

[1] Environment
[2] Algorithm
[3] Python
[4] Tensorflow
[5] Keras
[6] Gym

های پر چالش طراحی شده، بهترین عملکرد تاکنون مربوط به مدل پیاده شده در بستر کتابخانه پای تورچ[۱] اســـت که الگوریتم SAC را با دو لایه میانی به ترتیب به لندازه های ۴۰۰ و ۳۰۰ پیاده می کند. مدل های طراحی شـده در این پروژه با دو لایه میانی به اندازه های ۶۴ برای الگوریتم های A2C و مشـتق از آن و دو لایه میانی به اندازه های ۲۵۶ برای الگوریتم MPO به نتایج مشـابه و گاها بهتر دسـت می یابد. با توجه به کوتاه بودن زمان آموزش که برای مدل ها در بستر پای تورچ و با سخت افزار مناسب در حدود نیم ساعت و برای مدل های پیاده شده در بستر تنسورفلو در حدود یک الی دو ساعت است از مقایسه این پارامتر چشم پوشی می کنیم.

قابلیت نصب و استفاده بدون مشکل از هردو کتابخانه و تمام ویژگی هایشان بر روی تمام سیستم عامل هایی که قابلیت نصب پایتون و پیپ[۲] را دارند به راحتی و از طریق اجرای دستور

pip install -e {Library’s Directory} وجود دارد.

یا.....؟

.۲

[1] Pytorch
[2] Pip

فهرست مطالب

عنوان **صفحه**

فهرست شکل‌ها

عنوان **صفحه**

فهرست جدول‌ها

عنوان **صفحه**

No table of figures entries found.

فهرست علامت‌ها و نشانه‌ها

عنوان	علامت اختصاری
مجموعه حالات	S
مجموعه اعمال	A
مجموعه تمام حالت ها در فرآیند تصمیمی گیری مارکو	S^+
مجموعه حالت های آغازین ، زیر مجموعه S^+	S^i
احتمال وقوع x	$P(x)$
احتمال وقوع x به شرط y	$P(x\|y)$
حالت عامل در گام زمانی t یا همان حالت فعلی عامل	S_t
عمل عامل در گام زمانی t یا همان عمل فعلی عامل	A_t
تابع ارزش حالت	$v(s)$
تابع ارزش اقدام	$q(s,a)$

فصل ۱-مقدمه

۱-۱- پیشگفتار

همانطور که در بخش تاریخچه مطرح می شـود، بسـیاری از محیط ها و عامل های طراحی شـده برای آنها بسـیار سـاده اند و پاسـخگوی چالش های اسـاسـی دنیای واقعی نیسـتند. هدف از این پروژه طراحی عاملی با کمک الگوریتم های یادگیری تقویتی عمیق اسـت که بتواند در محیطی با چالش های محیطی دنیای واقعی عملکرد مطلوبی با توجه به سـاختار فیزیکی ربات مورد اسـتفاده داشـته باشـد. برای این منظور علاوه بر پیاده سـازی و مقایسـه الگوریتم های مذکور برای یافتن بهترین الگوریتم پاسـخگوی این اهداف، نیازمند طراحی محیط هایی تا حد ممکن شبیه به محیط های موردنظر با چالش های مشابه در جهان واقعی هستیم.

۱-۲- تاریخچه

در این بخش به بررسـی کارهای انجام شـده در زمینه طراحی محیط های مورد اسـتفاده برای آموزش عامل ها در کنار الگوریتم های یادگیری عمیق تقویتی پیاده شـــده در این پروژه و پیاده ســـازی های مطرح و پراستفاده دیگر آنها می پردازیم.

۱-۲-۱-محیط ها

ابتدا به بررسی چند پروژه پرکاربرد در این زمینه می پردازیم.

بولت[۱]: بولت یک موتور فیزیک است که اثر تصادم و دینامیک اشیا نرم و سخت را شبیه سازی می کند. این موتور برای بازی های کامپیوتری و جلوه های بصری در فیلم ها مورد استفاده قرار می گیرد. نویسنده اصلی این پروژه، اروین کومنز، برنده یک جایزه علمی و تخصصی آکادمیک برای کارهایش بر روی بولت شده است.

کتابخانه فیزیک بولت یک نرم افزار رایگان و متن باز است. منبع کد از بعد از سال ۲۰۱۴ بر روی گیتهاب قرار گرفته ولی پیش از آن بر روی گوگل کد قرار داشت. بخش عمده و اصلی زبان های استفاده شده در این پروژه به ترتیب سی پلاس پلاس[۲] و سی[۳] هستند.

پای بولت [1][۴]: پای بولت یک ماژول ساده پایتون برای شبیه سازی فیزیکی، رباتیک و یادگیری تقویتی عمیق است که برمبنای کتابخانه فیزیک بولت پیاده سازی شده است. با کمک این کتابخانه امکان بارگذاری بدنه بند بند ربات ها از طریف فایل های یوآردی اف[۵]، اس دی اف[۶]، اکس ام ال[۷] و دیگر فرمت های مشابه وجود دارد. از دیگر امکانات این کتابخانه می توان به شبیه سازی دینامیک[۸] رو به جلو، محاسبات دینامیک معکوس، فیزیک جنبش جلو رونده و معکوس و تشخیص برخورد اجسام اشاره کرد.

جیم [2][۹]: جیم یک کتابخانه متن باز پایتون برای توسعه و مقایسه الگوریتم های یادگیری تقویتی از طریق ارائه یک واسط استاندارد برای ارتباط بین الگوریتم های یادگیری و محیط هاست.

محیط های موجوکو[2] [۱۰]: موجوکو سروند "دینامیک چند مفصله با تصادم"[**معادل**] است. این کتاب خانه یک موتور فیزیکی برای تسهیل تحقیقات، توسعه رباتیکز[۱۱]، بیومکانیکز[۱۲]، گرافیک[۱۳] و انیمیشن[۱۴] است.

---

[1] Bullet
[2] C++
[3] C
[4] pybullet
[5] URDF
[6] SDF
[7] XML
[8] Dynamics
[9] Gym
[10] MuJoCo Environments: MuJoCo stands for Multi-joint dynamics with Contact
[11] Robotics
[12] Biomechanics
[13] Graphics
[14] Animation

تمام محیط های موجوکو نیازمند نصب موتور موجوکو هستند. درحال حاضر ۱۰ محیط در کتابخانه موجوکو وجود دارد.

از سال ۲۰۲۱ دیپ مایند[1]، موجوکو را به دست آورده است و از سال ۲۰۲۲ آن را متن باز خواهد نمود که در نتیجه آن برای همگان رایگان خواهد شد.

پای بولت جیم پریوم[3] [2]: برای رفع مشکل متن باز نبودن موجوکو و تا قبل از متن باز شدن آن یک تیم از توسعه دهندگان برای آسان تر کردن کار های توسعه و تحقیق در زمینه یادگیری تقویتی، تصمیم به توسعه پای بولت جیم گرفتند که بعد ها به پای بولت جیم پریوم تغییر نام داد.

این کتابخانه یک توسعه متن باز از محیط های موجوکوی اپن ای آی جیم است. برای آسان‌تر کردن روند توسعه و نشان دادن کارکرد و استفاده محیط های این کتابخانه، چند عامل از کتابخانه یادگیری تقویتی تنسورفورس[3] برای این محیط ها تنظیم و آموزش داده شده اند بنابراین برای هر محیط چندین عامل از پیش آموزش داده شده به عنوان تست واحد و خطوط محوری برای توسعه های بعدی وجود دارد.

رکس-جیم [4][4]: هدف این پروژه آموزش یک ربات چهارپای پرینت ۳ بعدی شده با پویش در الگوریتم های یادگیری تقویتی و محیط های اپن ای آی جیم است. مقصود این است که عامل طراحی شده وظایف خانگی و عمومی را در محیط شبیه سازی یادبگیرد و سپس بدون نیاز به هیچ میزان سازی و تنظیمات دیگری این دانش یا به عبارتی سیاست آموخته شده را بدون هیچ مشکلی بر روی ربات واقعی به کار بگیرد.

در این پروژه تعدادی وظایف زیر توسط عامل طراحی شده انجام می شوند:

۱- کنترل پایه

a. ژست گرفتن ربات طبق دستورات کنترلی

[1] DeepMind
[2] Pybullet Gymperium
[3] Tensorforce
[4] Rex-Gym

b. دویدن

c. راه رفتن

d. چرخیدن در یک نقطه

e. ایستادن از حالت خوابیده روی زمین

۲- مسیر یابی در زمین های نا هموار

a. مسیر یابی بر روی تپه و زمین های ناهموار ایجاد شده به صورت تصادفی

دی ام کنترل[5] [1]:

۱-۲-۲-الگوریتم ها

بیس لاینز [6]: اپن ای آی[2] بیس لاینز یک مجموعه با کیفیت از پیاده سازی الگوریتم های یادگیری تقویتی اســت. این الگوریتم ها کار را برای جامعه محقق آســان تر می کنند تا به کمک این خطوط پایه به بهبود و حتی پیاده سازی ایده های جدید بپردازند.

در این کتابخانه زبان پایتون، الگوریتم های زیر با نسخه ۱.۱۴ تنسورفلو (بدون واسط کراس) پیاده شده اند:

۱- ای ۲ سی[3]

۲- ای سر[4]

۳- اکتر[5]

۴- دیدی پیجی[6]

۵- دیکیو ان[7]

[1] DM_Control
[2] OpenAi
[3] A2C: Advantage Actor Critic
[4] ACER: Sample Efficient Actor-Critic with Experience Replay
[5] ACKTR: Actor Critic with Kronecker-factored Trust Region
[6] DDPG: Deep Deterministic Policy Gradient
[7] DQN: Deep Q-Network

۶- گیل[1]

۷- هر[۲]

۸- پیپیو۱ [3]

۹- پیپیو۲

۱۰- تی آر پیو[4]

استیبل بیس لاینز [7][۵]: استیبل بیس لاینز یک مجموعه از پیاده سازی های بهبودیافته الگوریتم های یادگیری تقویتی است که بر پایه پیاده سازی های اپن ای آی بیس لاینز است. در حال حاضر این کتابخانه در حال تعمیر است و در صورت نیاز به این الگوریتم ها باید از پیاده سازی دیگر این تیم یعنی استیبل بیس لاینز ۳ استفاده شود. شایان ذکر است که این کتابخانه تنسورفلوی نسخه ۱.۸.۰ تا نسخه ۱.۱۴.۰ را پشتیبانی می کند و پیاده سازی با تنسورفلوی ۲ در حال برنامه ریزی است.

از تفاوت های عمده این پیاده سازی با پیاده سازی اپن ای آی می توان به موارد زیر اشاره کرد:

- ساختار یکپارچه برای تمام الگوریتم ها
- سبک یکپارچه کد
- کلاس ها و توابع مستند نویسی شده
- تست های بیشتر و پوشش بیشتر کد با تست ها
- الگوریتم های جدید پیاده شده شامل سک[۶] و تی دی ۳[۷]

[1] GAIL: Generative Adversarial Imitation Learning
[2] HER
[3] PPO: Proximal Policy Optimization
[4] TRPO: Trust Region Policy Optimization
[5] Stable Baselines
[6] SAC: Soft Actor Critic
[7] TD3: Twin-Delayed Deep Deterministic Policy gradient

استیبل بیس لاینز۳[1][8] : استیبل بیس لاینز ۳ یک مجموعه از پیاده سازی های قابل اطمینان از الگوریتم های یادگیری تقویتی در بستر پای تورچ است که دومین پیاده سازی اصلی بهره گرفته از استیبل بیس لاینز اپن ای آی محسوب می شود. این کتابخانه یک واسط تمیز و ساده برای استفاده از الگوریتم های یادگیری تقویتی ارائه می دهد.

این کتابخانه کاملا مستند نویسی شده و آزموده، امکان آموزش عامل ها را تنها در چند خط فراهم می کند.

عامل های تنسورفلو[2][9] : یک کتابخانه مقیاس پذیر، مطمئن و ساده برای استفاده نوشته شده در بستر تنسورفلو برای الگوریتم های بندیتز متنی[3] و یادگیری تقویتی است.

این کتابخانه پیاده سازی، توسعه و آزمودن الگوریتم های یادگیری تقویتی را آسان تر می کند. این کتابخانه همچنین اجزا آزموده و قابل تغییری را در اختیار می گذارد که امکان دوره های سریع کد و محک آن را فراهم می آورد.

در این کتابخانه هسته اصلی الگوریتم های یادگیری تقویتی به عنوان عامل ها پیاده شده اند.هر عامل دو مسئولیت اصلی را بر عهده می گیرد: ۱- تعریف یک سیاست برای ارتباط با محیط و ۲- چگونگی تعلیم (بهبود) این سیاست به کمک تجربه کسب شده

این کتابخانه از تمام نسخه های تنسورفلو بعد از تنسورفلوی ۱.۱۵.۰ پشتیبانی می کند.

۱-۳- آنچه باید انجام شود

۱-۳-۱- کوچک تر کردن ساختار شبکه های عصبی

در بسیاری از پیاده سازی های بررسی شده در بخش تاریخچه برای دستیابی عامل به نتایج قابل قبول نیاز به آموزش شبکه های عمیق و یا پهنی است که علاوه بر مصرف حافظه برای نگه داری مدل آموزش دیده به

[1] Stable Baselines3
[2] Tensorflow Agents
[3] Contextual Bandits

رم[1] با حجم زیاد برای بارگذاری و استفاده از عامل هم نیاز است. علاوه بر این موارد، آموزش اینچنین مدل هایی به طور کلی نیازمند زمان بیشتری است؛ زیرا محاسبات لازم برای چنین شبکه هایی بسیار بیشتر است. بسته به سخت افزار مورد استفاده این زمان ممکن است از چند ساعت تا چند روز متفاوت باشد؛ به عبارت دیگر در این موارد برای آموزش و استفاده مطلوب از عامل طراحی شده نیازمند حافظه و رم با حجم بیشتر و پردازنده های قوی تری هستیم که ممکن است برای تولید انبوه در دسترس نباشد یا باعث زیاد شدن هزینه ها شود.

۱-۳-۲- کاهش زمان آموزش

یکی از عوامل مهمی که می تواند در ارزیابی و مقایسه عامل های مختلف استفاده شود زمان مورد نیاز برای آموزش آنها (مشخصا در یک محیط یکسان) می باشد.

همانطور که در بخش قبل اشاره شد یکی از راه های کاهش زمان آموزش کوچکتر کردن شبکه عصبی مورد نیاز برای عامل طراحی شده می باشد. علاوه بر این همانطور که در فصول بعد مطرح می شود برخی الگوریتم های یادگیری تقویتی عمیق باوجود عملکرد مشابه سایر الگوریتم ها نیاز به شبکه های عصبی کوچک تر و زمان آموزش کمتری دارند.

۱-۳-۳- کنترل بیشتر بر روی ربات

در بیشتر محیط ها و عامل های طراحی شده هدف اصلی آموزش راه رفتن و یا دویدن به ربات های طراحی شده است و در مواردی که جهت گیری در محیط به ربات آموزش داده می شود ربات امکان تغییر مسیر و حرکت در مسیر جدید را ندارد و صرفا جهت گیری جدیدی به سمت هدف انجام می دهد به عبارتی در این موارد یک یا چندین مدل برای وظایف مختلف آموزش داده می شود و امکان استفاده یک جا از تمام قابلیت ها برای مثال برای جهت گیری و حرکت، تنها از طریق تغییر دستی مدل در حین اجرا ممکن است که

[1] RAM

عملا کاری غیر ممکن است و حتی در صورت امکان به دلیل نیاز این روش به ذخیره وزن های چندین مدل و در برخی موارد بارگذاری لحظه ای مدل ها نیازمند منابع سخت افزاری بیشتری هستیم.

در بسیاری دیگر از مواردی که امکان کنترل ربات از طریق مشخص کردن هدف و مسیر حرکت برای آن فراهم است، ربات مورد استفاده بسیار ساده است؛ به عبارتی عامل طراحی شده به دلیل کم بودن عوامل کنترلی ربات یا به عبارتی کم بودن تعداد عمل[1] های ممکن برای عامل، کار مسیر یابی را به راحتی انجام می دهد. استفاده از اینگونه ربات ها می تواند در بسیاری از موارد پاسخگوی نیاز ما باشد اما در مواردی که نیاز به مسیر یابی در محیط های دشوار و پرچالش هست استفاده از این ربات ها و عامل ها نمی تواند مناسب باشد.

۱-۳-٤- بهینه کردن مصرف توان ربات

با توجه به موارد استفاده از اینگونه ربات ها نیازمند این هستیم که این ربات ها تا حد ممکن کم ترین مصرف توان را با وجود بهترین عملکرد ممکن داشته باشند. برای این منظور علاوه بر درنظر داشتن موارد ذکر شده در بخش های قبل مثل کوچک تر کردن ساختار شبکه عصبی، پیاده سازی بهتر الگوریتم و یا استفاده از الگوریتم های بهینه، باید تا حد امکان عملکرد ربات را به عملکرد مطلوب نزدیک کنیم؛ به عبارتی برای رسیدن به هدف موردنظر راه های زیادی وجود دارد ولی تنها یک راه بهینه است.

اختصاص زمان بیشتر به آموزش ربات طراحی شده می تواند متوسط امتیاز عامل را برای یک اپیزود تا حد مشخصی بالا ببرد و بعد از آن عملا رشد نمودار امتیاز به گام های آموزش، متوقف شده و به خط صاف میل می کند.

برای پیدا کردن راه بهینه، باید بیشترین امتیاز یا متوسط آن در یک اپیزود حاصل شود. برای این منظور باید با تنظیم فراپارامتر های هر الگوریتم بهترین عملکرد هر الگوریتم را بیابیم و از بین الگوریتم های موجود با توجه به امکانات سخت افزاری و دیگر ترجیحات یکی را انتخاب کنیم.

[1] Action

۱-۳-۵- حل چالش های مشابه جهان واقعی

عامل طراحی شـده باید بتواند در محیط هایی که به دلایلی امکان اسـتفاده از دیگر روش های حمل و نقل در آنها وجود ندارد مورد اسـتفاده قرار بگیرد. برای این منظور علاوه بر درنظر داشـتن موارد ذکر شـده در بخش های قبل، عامل باید بتواند در محیط هایی مانند پله، زمین های نا هموار، شیب دار و یا پر پیچ و خم بدون مشکل حرکت کند.

۱-٤- نوآوری های ارائه شده

نوآوری های انجام شده در این پروژه را در قالب سه بخش از پیاده سازی های انجام شده بررسی می کنیم.

۱-٤-۱- کتابخانه شامل محیط های مورد نظر

در این پروژه یک کتابخانه شامل دو محیط اصلی و سه محیط مشابه فرعی پیاده سازی شده است. در این پروژه هدف اصلی آموزش ربات چهار پای مورچه[1] است اما در کنار آن محیط هایی برای آموزش ربات های چهار پای دیگر نیز قرار گرفته اسـت. دو محیط اصـلی پیاده شـده یکی محیط آموزشـی و دیگری محیط نمایش امکانات ربات است.

در این محیط ها ۷ چالش مطرح می شود که با توجه به قابلیت های فیزیکی ربات مورد نظر حل ۶ مورد از آنها ممکن است. این چالش ها در دو دسته چالش های محیط آموزشی و محیط نمایشی مطرح می شوند و به شرح زیر اند:

۱-٤-۱-۱- محیط آموزشی

۱- حرکت در جهت رسیدن به یک نقطه در فاصله یک کیلومتری روی محور ایکس: این چالش کلاسیک در تمام محیط های شامل این ربات و ربات های چهارپای دیگر مطرح می شود. هدف این مسئله این است که عامل بتواند با اعمال عمل های مناسب بر روی ربات آنرا در جهت رسیدن به هدفی که معمولا در فاصله یک کیلومتری از محل شروع ربات قرار می گیرد و جهت آن معمولا ثابت و به

[1] Ant

سمت جهت مثبت محور ایکس[1] است به حرکت درآورد. شایان ذکر است که در طی فرآیند آموزش عامل، ربات تمام مسیر را طی نمی کند بلکه تعداد مشخصی گام زمانی به آن فرصت داده می شود تا به سمت هدف حرکت کند که این تعداد مشخص گام زمانی همان یک اپیزود[2] از بازی است که بعد از اتمام آن محیط به حالت همیشه یکسان اولیه بازگردانی می شود و با توجه به تابع پاداش، امتیاز عامل در میزان موفقیت در این محیط سنجده می شود که معمولا این پاداش به صورت متوسط پاداش های عامل طی چند اپیزود در نظر گرفته می شود.

۲- حرکت در جهت رسیدن به یک نقطه در فاصله یک کیلومتری: در این چالش یک نقطه در فاصله یک کیلومتری محل شروع ربات به صورت تصادفی انتخاب می شود؛ بنابراین هرکدام از نقاط موجود بر روی دایره ای به شعاع ۱ کیلومتر و مرکزیت محل شروع ربات ممکن است به عنوان مقصد ربات مشخص شود. در نتیجه آموزش ربات در این محیط، ربات باید بتواند در زمان حرکت از هر نقطه در جهت جدید مشخص شده تغییر مسیر داده و حرکت کند.

۳- حرکت بر روی زمین ناهموار: در این محیط پستی بلندی هایی به صورت تصادفی در محیط حرکت ربات ایجاد می شود و ربات باید بتواند در جهت مورد نظر بدون مشکل حرکت کند.

۱-٤-۱-۲- محیط نمایشی

۱- رسیدن به نقاط رندم هدف بر روی نقشه: در این چالش با رسیدن ربات به نقطه هدف مشخص شده بر روی نقشه، یک نقطه مقصد جدید به صورت تصادفی ایجاد می شود و این روند تا توقف برنامه ادامه می یابد.

۲- انجام دستورات کنترلی صفحه کلید: در این محیط عامل مسیر حرکت ربات را طبق دستورات صفحه کلید تعیین می کند.

[1] X axis
[2] Episode

۳- حرکت در مارپیچ: عامل باید بتواند بدون مشکل ربات را در مسیر های مارپیچ تصادفی ایجاد شده به حرکت درآورد. برای این منظور امکان مشخص کردن مارپیچ موردنظر از طریق عکس مارپیچ نیز وجود دارد. برای حل مارپیچ های بارگذاری شده از این طریق از الگوریتم راست گرد استفاده می شود. برای ایجاد و حل مارپیچ های تصادفی از کد پای میز[1] [10] استفاده مجدد شده و با اعمال تغییراتی برای استفاده در برنامه تنظیم شده است.

٤- حرکت در محیط هایی با تپه و کوه و پستی بلندی: در این محیط پستی بلندی و تپه هایی از طریق یک فایل عکس یا متنی بارگذاری می شوند که مشابه پستی و بلندی های واقعی هستند. بسته به اصطکاک سطح و ابعاد زمین و ناهمواری ایجاد شده ممکن است آموختن حرکت در این محیط برای ربات ممکن یا ناممکن باشد ولی در غالب موارد با توجه به نا همگون بودن شیب ها و ساختار ربات های ۴ پای ساده امکان، حرکت و آموزش در این محیط ها برای این ربات ها وجود ندارد و ربات به سادگی واژگون می شود. راه حل این مشکل در محیط های دی ام کنترل ارائه شده است. در این محیط ها ربات مورچه دارای پاهای سه بند به جای دو بند است. این ویژگی به ربات های مورچه این قابلیت را می دهد که با انعطاف بیشتر در مسیر های بسیار ناهموار به راحتی به حرکت خود ادامه دهند. همانطور که پیش تر اشاره شد، برای استفاده از این محیط ها نیاز به نصب موتور و محیط های موجوکو است تا سال ۲۰۲۲ متن باز نخواهد شد.

۱-٤-۲- کتابخانه شامل چند الگوریتم ساده یادگیری تقویتی عمیق

۱-٤-۳- پیاده سازی، آموزش و تنظیم عامل ها برای حل چالش های محیطی موردنظر

برای پیدا کردن بهترین راه حل ممکن برای حل چالش های مطرح شـــده در بخش قبل کار های زیر انجام شد:

[1] Pymaze

۱- بررسی الگوریتم های مختلف یادگیری تقویتی مناسب برای وظایف در محیط های موردنظر با فضای عملکردی پیوسته

۲- بررسی پیاده سازی های مختلف هر الگوریتم و مقایسه آنها

۳- آموزش تمام پیاده سازی های انجام شده در مرحله ۲ از تمام الگوریتم های مرحله ۱ و بررسی اثر فراپارامتر های مختلف روی فرآیند آموزش

شایان ذکر است که در مرحله ۳ پیاده سازی های مختلف از الگوریتم های شناخته شده و معمول برای وظایف مشابه با وظایف مورد نظر، انجام شده و با فراپارامتر ها با مقادیر مشابه راه حل این وظایف آموزش داده شده اند و بسیاری از پیاده سازی ها در مراحل ابتدایی و پس از چند ساعت آموزش عملا ناکارآمد تشخیص داده شده و از روند تحقیق حذف شده اند. برای آموزش عامل ها از امکانات سایت های کگل[1] و کولب[2] شامل واحد های پردازنده عادی و گرافیکی[3] استفاده شده است.

۱-۵- ساختار پایان نامه

در فصل دوم به بررسی الگوریتم های یادگیری عمیق تقویتی و مقایسه آنها پرداخته می شود. بررسی محیط آموزش و آزمایش عامل های طراحی شده، نحوه ساخت و شبیه سازی عوامل دنیای واقعی در طی این فرآیند ها و همچنین نحوه تحقیق در باره الگوریتم های یادگیری تقویتی و انتخاب آنها در فصل سوم انجام می شود. در فصل چهارم نتایج به دست آمده از پروژه و شبیه سازی های انجام شده در قالب اعداد و نمودار بیان و مقایسه می شود و در نهایت در فصل پنجم نتیجه گیری کلی از پروژه و به خصوص نتایج حاصل از فصل چهارم ارائه می شود و همچنین پیشنهاداتی در باره ادامه پروژه مطرح می شود.

[1] Kaggle
[2] Colab
[3] CPU and GPU

فصل ۲- پیاده سازی الگوریتم های یادگیری تقویتی

۲-۱- مقدمه

در این فصل ابتدا به بررسی مفاهیم پایه و ابتدایی در زمینه پیاده سازی الگوریتم های یادگیری تقویتی عمیق می پردازیم. پس از آن، در بخش بعد، با استفاده از این فرمول ها و مفاهیم ابتدایی به تعریف عامل های پیاده سازی شده در پروژه می پردازیم. در نهایت مقایسه بین مفاهیم و عامل ها و نکات خاص را در بخش نتیجه گیری بررسی می کنیم. شایان ذکر است که مفاهیم بیان شده در این فصل برگرفته از دو کتاب مقدمه ای بر یادگیری تقویتی[1] [11] ویرایش دوم و غر زدن یادگیری تقویتی عمیق[2] [12] ویرایش اول می باشد.

۲-۲- مفاهیم پایه

دو هسته اصلی یادگیری تقویتی را می توان عامل و محیط دانست. عامل تصمیم گیرنده و راه حل است و محیط نمود مسئله است. یکی از تفاوت های عمده یادگیری تقویتی با دیگر الگوریتم های یادگیری ماشین همین ارتباط بین عامل و محیط است. عامل از طریق فعالیت های خود بر محیط تاثیر می گذارد و محیط از این فعالیت ها تاثیر می پذیرد.

محیط قطعی[3]: یک محیط را قطعی می دانیم در صورتی که احتمال قرار گرفتن در موقعیت بعدی s' با داشتن محیط فعلی s و عمل a همیشه یک باشد.

---

[1] Reinforcement Learning: An Introduction, 2018

[2] Grokking Deep Reinforcement Learning, 2020

[3] Deterministic Environment

محیط تصادفی[1]: یک محیط را تصادفی یا غیر قطعی می دانیم در صورتی که احتمال قرار گرفتن در موقعیت بعدی s’ با داشتن محیط فعلی s و عمل a کمتر از یک باشد. در این حالت بیش از یک موقعیت بعدی ممکن s’ بعد از هر موقعیت s وجود دارد.

فرآیند های تصمیم گیری مارکو: در ریاضیات، یک فرآیند تصمیم گیری مارکو یک روند کنترلی اتفاقی گسسته در زمان است که یک چارچوب ریاضی برای مدل کردن تصمیم گیری در شرایطی فراهم می آورد که نتایج تا حدی تصادفی هستند.

برای مدل کردن تصمیم گیری عامل در محیط های یادگیری تقویتی که تا حدی تصادفی هستند از این شیوه استفاده می شود.

خاصیت مارکو[2]: خاصیت مارکو بیان می دارد که احتمال موقعیت[3] بعدی با دانستن موقعیت و عمل فعلی، مستقل از تاریخچه فعل و انفعالات است.

$$P(S_{t+1}|S_t, A_t) = P(S_{t+1}|S_t, A_t, S_{t-1}, A_{t-1}, \dots) \quad (۲-۱)$$

در معادله (۲-۱) نشان داده می شود که احتمال قرار گرفتن در حالت s در لحظه t+1 در صورت در نظر گرفتن حالت و عمل در لحظه t با احتمال قرار گرفتن در حالت s در لحظه t+1 در صورت در نظر گرفتن تمام فعل و انفعالات قبلی مساوی است.

تابع گذار[4]: به شیوه ای که محیط در واکنش به فعالیت های عامل تغییر پیدا می کند، احتمال گذار حالت می گوییم. به این معنی که در محیط قطعی با احتمال ۱ به یک حالت قطعی مشخص می رویم و در محیط غیر قطعی با احتمال کمتر از یک به یکی از حالات ممکن با توجه به حالت و عمل فعلی منتقل می شویم.

$$p(s'|s, a) = P(S_t = s'|S_{t-1} = s, A_{t-1} = a) \quad (۲-۲)$$

$$\sum_{s' \epsilon S} p(s'|s, a) = 1, \forall s \in S, \forall a \in A(s) \quad (۲-۳)$$

در معادله (۲-۲) تابع گذار به صورت احتمال گذار به حالت s’ در زمان t در صورت قرار داشتن در حالت s و با عمل a در زمان t-1 تعریف شده است.

---

[1] Stochastic Environment
[2] Markov Property
[3] State
[4] Transition Function

شرط معادله (۳-۲) به معنای این می باشد که این برابری برای تمام حالات موجود در مجاوعه حالات و تمام اعمال موجود در مجموعه اعمال برقرار است.

سه تایی انتقال[1]: سه تایی حالت فعلی، عمل فعلی و حالت بعدی را با s, a, s’ نشان داده و به آن سه تایی گذار یا انتقال می گوییم.

تابع پاداش[2]: به تابعی که سه تایی گذار را به یک عدد حقیقی نگاشت می کند تابع پاداش می گوئیم.

سیگنال پاداش: به مقدار عددی حاصل از تابع پاداش سیگنال پاداش گفته می شود که در واقع معیاری برای سنجش میزان خوب بودن یک گذار است. زمانی که این مقدار مثبت باشد می توان آن را درآمد یا پاداش می گویند. بیشتر محیط ها حداقل یک سیگنال پاداش با مقدار مثبت دارند. همچنین این مقدار می تواند منفی باشد که در این صورت می توان آن را تنبیه یا جریمه خواند؛ با این حال مقدار این سیگنال همیشه پاداش خوانده می شود.

$$r(s,a) = \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a] \quad (۲-۴)$$

$$r(s,a,s') = \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a, S_t = s'] \quad R_t \in \boldsymbol{\mathcal{R}} \subset \mathbb{R} \quad (۲-۵)$$

مطابق معادله (۴-۲) می توان تابع پاداش را به صورت انتظار پاداش در لحظه t با در نظر گرفتن زوج حالت-عمل در گام زمانی قبلی در نظر گرفت. همچنین طبق معادله (۵-۲) می توان این تابع را به صورت انتظار در لحظه t با در نظر گرفتن سه تا یی انتقال تعریف کرد. در نهایت سیگنال پاداش مقدار پاداش مورد انتظار است که عضو مجموعه پاداش های ممکن است که این مجموعه خود زیر مجموعه اعداد حقیقی است.

در فرآیندهای تصمیم گیری مارکو امکان نشان دادن زمان نیز وجود دارد. یک گام زمانی ممکن است با هر یک از اصطلاحات دوره[3]، چرخه[4]، تکرار[5] و یا حتی فعل و انفعال[6] بیان شود. در صورت تعریف گام زمانی در این فرآیندها امکان تقسیم بندی محیط ها بر این اساس به دو دسته فراهم می شود.

وظایف اپیزودیک[7]: وظیفه اپیزودیک به وظیفه ای می گویند که در آن تعداد متناهی گام زمانی وجود دارد؛ زیرا پس از یک تعداد گام زمانی عامل به یک حالت پایانی می رسد یا حداکثر تعداد گام های مجاز آن تمام می شود.

---

[1] Transition Tuple
[2] Reward Function
[3] Epoch
[4] Cycle
[5] Iteration
[6] Interaction
[7] Episodic

وظایف ادامه دار[1]: به وظایفی گفته می شود که در آنها هیچ حالت پایانی وجود ندارد بنابراین تعداد گام های زمانی مجاز عامل نا متناهی است.

افق برنامه ریزی[2]: در صورتی که وظایف اپیزودیک و ادامه دار را از دیدگاه عامل تعریف کنیم به آن افق برنامه ریزی می گویند. در این صورت یک افق که برای تعداد متناهی گام زمانی باشد را افق محدود[3] می گوییم نوع خاصی از این نوع افق را که تعداد گام های زمانی در آن یکی باشد را افق حریصانه[4] گوییم. از طرفی دیگر زمانی که عامل تعداد متناهی گام زمانی نداشته باشد و برای تعداد نا متناهی گام زمانی برنامه ریزی کند ممکن است در حال حل یک وظیفه اپیزودیک باشد اما از دیدگاه عامل این وظیفه، یک وظیفه افق نا متناهی[5] است.

عامل تخفیف[6]: در مواقعی که افق برنامه ریزی نا متناهی باشد، چه به علت ادامه دار بودن محیط و چه از دیدگاه عامل، احتمال نا متناهی بودن دنباله پاداش ها مطرح می شود. برای اینکه به عامل بفهمانیم که دریافت پاداش هرچه زودتر بهتر است از یک عدد حقیقی مثبت کمتر از ۱ استفاده می کنیم که توان هایی از آن در دنباله پاداش ها ضرب می شود و مقدار پاداش های آتی را کاهش می دهد. به این عدد حقیقی مثبت کمتر از یک گاما یا عامل تخفیف می گوییم. این عامل با کم کردن ارزش پاداش های آتی در فرآیند محاسبه تابع پاداش می تواند در پایدار کردن فرآیند یادگیری بسیاری از عامل ها موثر باشد.

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T \quad (۲-۶)$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots + \gamma^{T-1} R_T \quad (۲-۷)$$

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (۲-۸)$$

$$G_t = R_{t+1} + \gamma G_{t+1} \quad (۲-۹)$$

در معادله (۲-۶) تابع محاسبه برگشت یک اپیزود نشان داده شده است. در این معادله عامل تخفیف به کار نرفته است که در صورت میل T به بی نهایت مقدار برگشت[7] نیز نا متناهی می شود. در این معادله T دوره زمانی یا تعداد گام های زمانی یک اپیزود است. در معادله (۲-۷) عامل تخفیف اثر داده شده و تنها تفاوت آن با معادله قبلی همین است. این معادله در ادامه به صورت جمع وزن دار پاداش ها در هر گام زمانی در معادله

---

[1] Continuing Tasks
[2] Planning Horizon
[3] Finite Horizon
[4] Greedy Horizon
[5] Infinite Horizon
[6] Discount Factor
[7] Return

(۸-۲) نشان داده شده است. در ادامه معادله بازگشتی این معادلات با اعمال عامل تخفیف در معادله (۹-۲) نشان داده شده است.

سیاست[1]: با توجه به تصادفی بودن بیشتر محیط ها، عامل ها نیازمند پیدا کردن یک سیاست برای تصمیم گیری هستند؛ بنابراین می توان سیاست را به صورت تابعی که حالت فعلی را به اقدام (بعدی) عامل نگاشت می کند در نظر گرفت. سیاست را با π[2] نشان می دهیم. باید در نظر داشت که سیاست می تواند تصادفی باشد. این تصادفی بودن می تواند در خصوص یک اقدام باشد یا سیاست تعریف شده به صورت توزیع احتمال تمام اقدامات ممکن باشد. باید توجه داشت که هر عامل برای حل محیط (مسائل) مختلف سیاست های بهینه متفاوتی می تواند داشته باشد به عبارتی سیاست بهینه یک عامل در مواجه با یک محیط مشخص یکتا نیست.

تابع ارزش حالت[3]: این تابع ارزش هر حالت را در صورت استفاده از سیاست مشخص π بیان می دارد. از این تابع همچنین با عنوان تابع v[4] نیز یاد می شود.

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \qquad (۲-۱۰)$$

بنا براین ارزش هر حالت s در صورت پیروی از سیاست π برابر مقدار برگشت مورد انتظار در صورت قرار داشتن در این حالت و استفاده از سیاست مذکور می باشد.

در صورتی که معادله فوق را با نوشتن برگشت به صورت جمع وزن دار و سپس به صورت بازگشتی تعریف کنیم به معادله بلمن[5] می رسیم.

$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')], \forall s \in S \qquad (۲-۱۱)$$

در این معادله ما ابتدا مقدار حاصل از سیاست را به ازای یک حالت خاص به دست می آوریم که می تواند نشان دهنده مقدار یک یا چند اقدام باشد و یک جمع وزن دار را به ازای این مقادیر حساب می کنیم. همچنین وزن احتمال حالت و پاداش بعدی را در این جمع تاثیر می دهیم.

در معادله فوق ابتدا حاصل جمع پاداش و ارزش تخفیف خورده (با تاثیر دادن عامل تخفیف) حالت بعدی را حساب کرده و در احتمال این گذار ضرب می کنیم. در نتیجه به ازای تمام گذار های ممکن از حالت s یک جمع وزن دار محاسبه می شود. در محاسبه گذار های ممکن باید عامل اقدام هم در نظر گرفته شود؛ به عبارتی برای گذار از هر حالت s تعدادی اقدام ممکن وجود دارد که به ازای هر کدام مقدار حاصل از سیاست

---

[1] Policy

[2] Pie

[3] State-Value Function

[4] V-Function

[5] Bellman Equation

را در آن حالت محاسبه می کنیم و جمع وزن داری از ضرب این مقدار در حاص جمع وزن دار پیشین به دست می آوریم که این حاصل ارزش هر حالت را طبق معادله بلمن به ما می دهد.

معادله ارزش عمل/اقدام[1]: در این معادله، ارزش اقدامی مشخص، در صورت قرار داشتن در یک حالت معین محاسبه می شود. این تابع امکان مقایسه سیاست های مختلف را برای رسیدن به سیاست بهینه با مقایسه بین اقدام های ممکن برای گذار از یک حالت مشخص فراهم می کند. این تابع، همان تابع Q[2] می باشد.

$$q_\pi(s,a) = \mathbb{E}[G_t|S_t = s, A_t = a] \quad (۲-۱۲)$$

$$q_\pi(s,a) = \mathbb{E}[R_t + \gamma\, G_{t+1}|S_t = s, A_t = a] \quad (۲-۱۳)$$

$$q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma v_\pi(s')], \forall s \in S, \forall a \in A(s) \quad (۲-۱۴)$$

در معادله اول از سه معادله فوق ارزش اقدام a در صورت قرار داشتن در حالت s در صورت استفاده از سیاست π برابر مقدار برگشتی مورد انتظار تحت این شرایط است. در معادله دوم تعریف بازگشتی مقدار برگشت را جایگزین کرده ایم و در معادله سوم که همان معادله بلمن برای ارزش اقدام عامل است، این مقدار برابر جمع وزن دار روی مقدار حاصل از جمع پاداش حاصل یعنی r و ارزش کاهش یافته حالت بعدی یعنی $\gamma\, v_\pi(s')$ تحت سیاست مذکور مشخص شده است که وزن های این جمع، احتمال گذار از حالت s و با عمل a به حالت s' و با پاداش r می باشد. این جمع وزن دار به ازای تمام حالت های بعدی و پاداش های احتمالی حاصل از این گذار محاسبه می شود.

تابع سود اقدام[3]: نوع دیگری از تابع های ارزش گذاری سیاست، تابع سود یا سود اقدام یا تابع A می باشد. مقدار این تابع برابر تفاضل مقدار تابع ارزش حالت از مقدار تابع ارزش اقدام با عمل a در حالت s می باشد.

$$a_\pi(s,a) = q_\pi(s,a) - v_\pi(s) \quad (۲-۱۵)$$

مقدار حاصل از این تابع بیان می دارد که در هر مقطعی سود انتخاب عمل a نسبت به پیروی از سیاست پیش فرض π چقدر بیش تر است.

بهینگی[4]: سیاست ها، تابع ارزش حالت، تابع ارزش عمل و تابع سود عمل اجزایی برای مقایسه ارزیابی و بهبود رفتار عامل هستند. یک سیاست بهینه سیاستی است که به ازای هر حالت می تواند برگشت مورد انتظاری بزرگتر یا مساوی هر سیاست دیگری برگرداند. یک تابع ارزش حالت بهینه، یک تابع با مقدار حداکثری به ازای تمام سیاست ها و در تمام حالت هاست. به طور مشابه یک تابع ارزش عمل بهینه، یک تابع با مقدار

---

[1] Action-Value Function
[2] Q-Function
[3] Action-Advantage Function
[4] Oprimality

حداکثری به ازای تمام سیاست ها و به ازای تمام زوج های حالت و عمل است. یک تابع سود بهینه به ازای تمام زوج های حالت و عمل کوچکتر یا مساوی صفر است زیرا هیچ عملی نمی تواند سودی از تابع بهینه ارزش حالت داشته باشد.

$$v_*(s) = \max_{\pi} v_\pi(s), \forall s \in S \quad (۲-۱۶)$$

$$q_*(s,a) = \max_{\pi} q_\pi(s,a), \forall s \in S, \forall a \in A(s) \quad (۲-۱۷)$$

$$v_*(s) = \max_{a} \sum_{s',r} p(s',r|s,a)[r + \gamma v_*(s')] \quad (۲-۱۸)$$

$$q_*(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'} q_*(s',a')] \quad (۲-۱۹)$$

$$v_* = q(a_*) = \max_{a \in A} q(a) \quad (۲-۲۰)$$

$$a_* = argmax_{a \in A} q(a) \quad (۲-۲۱)$$

$$q(a_*) = v_* \quad (۲-۲۲)$$

همانطور که دیده می شود، در دو معادله اول از ۷ معادله بالا مقدار بهینه توابع V و Q برابر بیشترین مقدار ممکن در بین تمام سیاست ها ست. در دو معادله بعدی، معادلات این توابع جایگزین شده است و عملیات حداکثر گیری بر روی این معادلات انجام شده است. در معادله پنجم نشان داده می شود که تابع V بهینه برابر مقدار بیشینه تابع Q است که به ازای اقدام بهینه به دست می آید؛ بنابر این عمل بهینه عملیست که مقدار تابع Q بهینه را بیشینه می کند.

الگوریتم ارزیابی سیاست[1]: در این الگوریتم با بررسی کلیه مقادیر فضای حالات، مقدار تابع V را برای یک سیاست مشخص محاسبه می کنیم. اینگونه الگوریتم ها که به ازای یک سیاست ورودی مقدار یک تابع ارزیاب را خروجی می دهند الگوریتم های حل کننده مسئله پیش گوئی[2] هستند.

$$v_{k+1}(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)[r + \gamma v_k(s')] \quad (۲-۲۳)$$

که این معادله مشابه معادله (۲-۱۱) می باشد.

الگوریتم بهبود سیاست[3]: برای بهبود یک سیاست ما از تابع ارزیاب حالت و یک فرآیند تصمیم گیری مارکو برای پیش بینی و پیدا کردن اینکه چه عملی منجر به بیشترین مقدار ارزیابی می شود، استفاده می کنیم.

---

[1] Policy Evaluation/ Iterative Policy Evaluation

[2] Prediction Problem

[3] Policy Improvement Algorithm

$$\pi'(s) = argmax_a \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \quad (۲-۲۴)$$

مسئله کاوش و بهره برداری: یکی از مهمترین مسائل در زمینه یادگیری تقویتی و به طور کلی یادگیری ماشین، تنظیم درست نسبت کاوش و بهره برداری عامل می باشد. این مسئله این سوال اساسی را بیان می کند که در هر مقطع زمانی عامل بهتر است که به کاوش برای یافتن راه حل های جدید بپردازد یا آنکه از طریق بهترین راه حلی که به آن رسیده به دریافت پاداش بیشتری بپرداز. ممکن است بهترین روش پیدا شده بهترین نباشد و روش های بهتری با پاداش بالاتر هنوز توسط عامل کشف نشده باشند؛ همچنین ممکن است روش پیدا شده بهترین روش باشد و با کاوش بیشتر به راه حل بهتری نرسیم و صرفا زمان مفید عامل، بدون دستیابی به پاداش مثبتی تلف شود. روشی که درآن عامل به طور مطلق به انجام عملی که از آن بیشترین پاداش حاصل می شود می پردازد را روش حریصانه و روشی که در آن عامل به طور مطلق به دنبال به دست آوردن دانش جدید از محیط است و هرگز بهره برداری نمی کند را روش جستوجوی همیشگی[1] می گوییم. همچنین اپسیلن-حریصانه[2] یکی از روش های پر استفاده در مواجه با این مسئله است که در آن بیشتر مواقع به بهره برداری حریصانه می پردازیم و در اندک درصدی از مواقع به جستوجو و کاوش می پردازیم.

برازش عصبی Q[3]: تکرار Q برازش عصبی یا تکرار NFQ یکی از نخستین الگوریتم هایی است به طور موفق شبکه های عصبی را به عنوان تقریب تابع برای حل مسائل یادگیری تقویتی به کار گرفته است.

در ادامه چند جزء مهم در خصوص طراحی الگوریتم های یادگیری عمیق تقویتی مبتنی بر مقدار را بررسی می کنیم.

۱- انتخاب یک تابع ارزش برای تقریب: در استفاده از شبکه های عصبی توابع ارزش زیادی وجود دارد که برای تقریب توسط شبکه های عصبی می توانند استفاده شوند. در بیان تابع تقریب زده شده توسط شبکه های عصبی از حروف بزرگ لاتین استفاده می شود برای مثال تابع ارزش اقدام یعنی q(s,a) را به صورت Q(s, a: θ) می نویسیم که یعنی تقریب Q دارای پارامتر های θ یا وزن های شبکه عصبی، یک حالت s و یک اقدام a می باشد.

[1] Always Explore

[2] Epsilon-greedy

[3] NFQ: Neural Fitted Q

۲- انتخاب یک ساختار برای شبکه عصبی: یک شبکه عصبی مورد استفاده برای تقریب توابع ارزش را می توان متشکل از سه بخش اصلی دانست: ۱) لایه ورودی[۱] ۲) لایه پنهان[۲] ۳)لایه خروجی[۳] برای انتخاب تعداد نورون های لایه ورودی و خروجی دو رویکرد معمول وجود دارد: ۱) متغیر های حالت و عمل هر دو به عنوان ورودی شبکه عصبی باشند. که در این روش اندازه لایه ورودی برابر مجموع تعداد متغیر های حالت و عمل می باشد. در این حالت شبکه صرفا یک نورون در لایه خروجی دارد و صرفا یک مقدار به عنوان خروجی می دهد که برابر مقدار تابع Q برای زوج حالت-عمل ورودی شبکه است. ۲)تنها متغیر های حالت را به عنوان ورودی به شبکه عصبی بدهیم و به تعداد اقدام های ممکن در محیط یا به عبارتی تعداد متغیر های عمل مربوط به محیط نورون خروجی داشته باشیم که در این حالت شبکه به ازای تمام اقدام های ممکن، مقدار تابع Q را برای حالت ورودی شبکه حساب می کند. این روش نسبت به روش اول به صرفه تر است.

۳- انتخاب تابع مورد نظر برای بهبود: در صورتی که یادگیری تقویتی را به صورت یک مسئله یادگیری تحت نظارت[۴] ببینیم برای دستیابی به تابع بهینه ارزش عمل، نیازمند کمینه کردن فاصله مقادیر این تابع با مقادیر بهینه تابع Q هستیم.

۴- انتخاب یکی از روش های بهینه سازی: در ادامه به بررسی چند روش پرکاربرد در زمینه بهینه سازی تابع هدف می پردازیم.

یکی از روش های شناخته شده بهینه سازی، گرادیان کاهشی[۵] است. این روش نیازمند برقرار بودن دو شرط است: ۱) داده باید مستقل و به طور یکسان توزیع شده باشد. ۲) اهداف باید ایستا باشند. در یادگیری تقویتی نمی توان اطمینان حاصل کرد که این دو شرط برقرار باشند بنابراین برای کمینه کردن تابع هزینه باید از روش های دیگر و مشابه استفاده کرد. یکی دیگر از روش های مورد استفاده

---

[1] Input layer
[2] Hidden layer
[3] Output layer
[4] Supervised learning
[5] Gradient Decent

در یادگیری تحت نظارت، گرادیان کاهشی دسته ای[۱] می باشد. در این روش کل مجموعه داده در آن واحد در نظر گرفته شده و گرادیان آن محاسبه شده و کمی در جهت این گرادیان پیش می رویم. این فرآیند به صورت تکراری و تا زمان همگرایی انجام می شود. تکرار این فرآیند در صورت نزدیک شدن مقادیر حاصل از تابع هزینه در طی اعمال مکرر و مجدد گرادیان کاهشی متوقف می شود که در این نقطه از زمان همگرایی اتفاق می افتد. طبیعتا این روش در مواقعی که اندازه مجموعه داده زیاد باشد بسیار کند عمل می کند و نمی تواند مناسب باشد. در مورد مسائل یادگیری تقویتی به طور کلی هیچ گونه مجموعه داده ای وجود ندارد بنابراین این روش عملا کاربردی ندارد. برای رفع این مشکل از روش گرادیان کاهشی با دسته های کوچک[۲] استفاده می شود. در این روش صرفا از بخشی از مجموعه داده در هر مقطع زمانی استفاده می شود. در هر مقطع تابع هزینه را برای این دسته کوچکتر محاسبه می کنیم سپس با پس انتشار[۳] گرادیان مقدار تابع هزینه را به دست می آوریم و در ادامه، طبق روش گرادیان کاهشی، وزن های شبکه را به گونه ای تغییر می دهیم(اصلاح می کنیم) که در پیش بینی مقادیر این دسته کوچک بهتر عمل کند. با استفاده از این روش امکان تنظیم اندازه دسته های کوچکتر وجود دارد که در نتیجه آن می توانیم از مجموعه داده با اندازه های بیشتر استفاده کنیم. با تنظیم اندازه دسته های کوچکتر می توان به دو الگوریتم دیگر نیز رسید. در صورتی که اندازه این دسته های کوچک را برابر اندازه مجموعه داده هایمان در نظر بگیریم همان الگوریتم گرادیان کاهشی دسته ای را خواهیم داشت و در صورتی که اندازه دسته ها را یک بگیریم در هر مقطع صرفا یک داده را پردازش می کنیم که به این الگوریتم گرادیان کاهشی تصادفی[۴] می گوییم.

در روش گرادیان کاهشی با دسته های کوچک، هرچه اندازه دسته ها بزرگتر باشد میزان تغییر در گام های گرادیان کمتر می شود و با روند یکنواخت تری همگرایی اتفاق می افتد. هرچند که در

---

[1] Batch Gradient Decent
[2] Mini-Batch Gradient Descent
[3] Backpropagation
[4] Stochastic Gradient Descent

صورت زیاد کردن اندازه دسته ها سرعت هر دوره و در نتیجه سرعت الگوریتم کاهش می یابد. از طرفی دیگر در صورت کاهش این مقدار ضمن زیاد شدن تغییر ها در گام های گرادیان کاهشی، سرعت این الگوریتم به نوعی دیگر، به شدت کاهش می یابد.

یکی دیگر از این روش های، روش گرادیان کاهشی با گشتاور[1] می باشد. در این روش که مشابه گرادیان کاهشی با دسته های کوچک می باشد، وزن های جدید شبکه، در جهت حرکت متوسط گرادیان ها تغییر می کند و نه با مقدار خود گرادیان.

در روش دیگری مشابه به روش گرادیان کاهشی با گشتاور، وزن ها در جهت حرکت متوسط اندازه گرادیان ها تغییر می کنند. این روش با عنوان انتشار ریشه درجه دوم میانگین مربعات[2] شناخته می شود. در نهایت روش پرکاربرد در بیشتر مسائل روش آدام یا تخمین لحظه ای تطبیقی[3] است که ترکیبی از دو روش قبل می باشد.

چندتایی تجربه[4]: به مجموعه حالت فعلی S_t، اقدام کنونی A_t، پاداش حاصل R_{t+1} و حالت بعدی S_{t+1} چندتایی تجربه می گوییم. یک سری چندتایی تجربه یک مسیر[5] را تشکیل می دهند.

روش پیش بینی مونته کارلو[6]: پیش بینی مونته کارلو یا MC روشی است که در آن عامل با سیاست π تا زمان رسیدن به حالت پایانی S_T، طی یک اپیزود، با محیط در فعل و انفعال است و در طی این فعل و انفعال یک مسیر تولید می شود. از این مسیر می توان برای محاسبه برگشت $G_{t:T}$ برای هر حالت S_t استفاده کرد. برای این کار از جمع وزن دار پاداش های به دست آمده در طی مسیر با وزن توانی γ در بازه گام زمانی t تا گام زمانی نهایی T استفاده می شود. به همین ترتیب برای تمام گام های زمانی t تا T امکان محاسبه S_t با همین روش وجود دارد.

روش یادگیری اختلاف زمانی[7]: یکی از اشکالات روش مونته کارلو این است که عامل باید تا انتهای یک اپیزود منتظر بماند تا مقدار درست برگشت را به دست آورد پیش از آنکه بتواند تابع ارزش حالت را به روز کند. در

---

[1] Gradient Descent with Momentum
[2] Root Mean Square Propagation or RMSprop
[3] Adam or Adaptive Moment Estimation
[4] Experience tuple
[5] Trajectory
[6] Monte Carlo prediction
[7] Temporal-difference learning or TD

این روش ها به جای انتظار برای اتمام یک اپیزود از پاداش یک گام زمانی استفاده می شود. برای تخمین مقدار برگشت در گام بعد از تخمین های تابع ارزش حالت یعنی $V(S_{t+1})$ استفاده می شود.

$$R_{t+1} + \gamma\, V_t(S_{t+1}) \qquad (۲-۲۵)$$

در معادله فوق تابع هدف روش TD یا همان اختلاف زمانی مطابق توضیحات این روش بیان شده است.

$$G_{t:T} = R_{t+1} + \gamma\, G_{t+1:T} \qquad (۲-۲۶)$$

در معادله فوق فرم بازگشتی تابع برگشت برای استفاده در معادله بعد بیان شده است.

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1}) | S_t = s] \qquad (۲-۲۷)$$

که معادله فوق به معنای این است ما می توانیم مقدار تابع ارزش حالت را در هر گام زمانی تخمین بزنیم.

$$V_{t+1}(S_t) = V_t(S_t) + \alpha_t[R_{t+1} + \gamma\, V_t(S_{t+1}) - V_t(S_t)] \qquad (۲-۲۸)$$

در معادله فوق بخش داخل کروشه خطای اختلاف زمانی(TD) است که اختلاف بین تابع هدف TD و تخمین در حالت فعلی است. در صورت استفاده از روش TD، جای استفاده از روش مونته کارلو، عامل طراحی شده یک عامل سارساست[1] که در این عامل ها، سیاست بعد از هر گام زمانی بهبود می یابد.

روش یادگیری اختلاف زمانی چند گامه[2]: همانطور که در قسمت های قبل مشاهده شد برای پردازش نتایج به دست آمده از محیط در طی مسیر و تغیر تابع ارزش حالت دو راه وجود دارد: ۱) استفاده از روش مونته کارلو که در این حالت بعد از اتمام هر اپیزود نتایج پردازش مسیر روی تابع ارزش حالت اعمال می شود. ۲) روش اختلاف زمانی که درآن در هر گام زمانی تغیییرات حاصل از پردازش پاداش همان گام روی تابع ارزش حالت اعمال می شود. هر دوی این موارد بسته به موقعیت ممکن است بهتر از دیگری کار کنند ولی به طور کلی عملکرد هر دو روش ضعیف است. برای دست یابی به عملکرد بهتر از روش اختلاف زمانی چند گامه استفاده می کنیم که در آن بعد از هر چند گام زمانی نتایج حاصل از محیط را پردازش کرده و تابع ارزش حالت را بر اساس آن تغییر می دهیم.

$$V_{t+n}(S_t) = V_{t+n-1}(S_t) + \alpha_t[G_{t:t+n} - V_{t+n-1}(S_t] \qquad (۲-۲۹)$$

در فرمول بالا $G_{t:t+n}$ را هدف چند گامه می گوییم.

---

[1] SARSA

[2] N-step TD Learning Method

روش نمای رو به جلو اختلاف زمانی(ג)[1]: در این روش عامل، جمع وزن دار تمام چند گام را به عنوان یک هدف استفاده می کند. در این روش مجموع اختلاف زمانی یک گام، دو گام و ... تا بی نهایت گام[2] را با یک وزن توانی کم تر از یک حساب می کنیم.

بهینگی در روش های مبتنی بر مقدار[3]: یکی از اهداف ایده آل در یادگیری عمیق می تواند کم کردن تابع هزینه باشد. این تابع میزان تفاوت تابع تخمین زده شده با تابع بهینه q یعنی q_* را بیان می کند. تابع هزینه در روش های مبتنی بر مقدار را می توان به روش زیر تعریف کرد.

$$L_i(\theta_i) = \mathbb{E}_{s,a}\left[\left(q_*(s,a) - Q(s,a;\theta_i)\right)^2\right] \quad (۲-۳۰)$$

در این روش ها نیازمند راهی برای نمود تابع q_* هستیم. عملا داشتن چنین تابعی تلاش برای دستیابی به Q بهینه را نقض می کند و در عین حال دست یابی به چنین تابعی غیر ممکن است. در عوض به جای این تابع از مقادیر برگشت در روش مونته کارلو یا بوت استرپینگ[4] استفاده می کنیم.

حافظه بازپخش[5]:

روش های مبتنی بر سیاست[6]: در این روش ها هدف بیشینه کردن نتیجه یک سنجه عملکرد است که همان

تابع ارزش واقعی سیاست مورد استفاده برای تمام حالات ابتدایی می باشد.

$$J(\theta) = \mathbb{E}_{s0\sim p0}[v_{\pi\theta}(s0)] \quad (۲-۳۱)$$

معادله فوق نشان دهنده تابع هدف این روش هاست. می دانیم هدف پیدا کردن گرادیان این سنجه است. بنابراین از دو طرف معادله فوق گرادیان می گیریم. برای آسان تر کردن فرمول نگاری های آتی، برابری های زیر را به ترتیب برای نشان دادن یک مسیر و مقدار برگشت یک مسیر در نظر می گیریم.

$$\tau = S_0, A_0, R_1, S_1, \dots, S_{T-1}, A_{T-1}, R_T, S_T \quad (۲-۳۲)$$

$$\mathrm{G}(\tau) = R_1 + \gamma\, R_2 + \cdots + \gamma^{T-1} R_T \quad (۲-۳۳)$$

همچنین می توان احتمال وقوع یک مسیر را در صورت استفاده از سیاست مشخص طبق فرمول زیر تخمین زد.

---

[1] Forward-view TD(ג)

[2] Infinite-step

[3] Optimality in Value-Based Methods

[4] Bootstrapping

[5] Replay Memory

[6] Policy-Based Methods

$$p(\tau|\pi_\theta) = p_0(S_0)\pi(A_0|S_0;\theta)P(S_1,R1|S_0,A_0)\dots P(S_T,R_T|S_{T-1},A_{T-1}) \quad (۲-۳۴)$$

$$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[G(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}[\nabla_\theta \log\, \mathrm{p}(\tau|\pi_\theta)G(\tau)] \quad (۲-۳۵)$$

بنابراین با جایگزینی فرمول احتمال وقوع یک مسیر، تبدیل ضرب به جمع و مشتق گیری نسبت به θ وابستگی به تابع گذار از بین می رود و به تابع زیر می رسیم که همان گرادیان تابع هدف است.

$$\nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[G(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}[\sum_{t=0}^{T} G_t(\tau)\nabla_\theta \log\pi(A_t|S_t;\pi_\theta)] \quad (۲-۳۶)$$

در قسمت جمع فرمول فوق، به ازای تمام گام های زمانی در طول یک مسیر، مقدار برگشت از آن گام محاسبه می شود و از حاصل این مقدار به عنوان وزنی برای احتمال لگاریتمی عمل انتخاب شده در آن گام زمانی استفاده می شود. در نهایت حاصل جمع وزن دار گرادیان این احتمالات لگاریتمی محاسبه می شود.

ساختار دو شبکه ای: در این ساختار از دو شبکه عصبی استفاده می شود، یکی به اسم شبکه سیاست[۱] و یکی شبکه مقدار[۲] (مقدارده، ارزش یا ارزیاب). شبکه سیاست یا تخمین سیاست به تعداد متغیر های حالت نورون لایه ورودی و به تعداد متغیر های اعمال قابل انتخاب، نورون در لایه خروجی دارد. در شبکه مقدار به تعداد متغیر های حالت نورون ورودی داریم و تنها یک نورون در لایه خروجی قرار داده می شود که ارزش حالت ورودی شبکه را خروجی می دهد.

روش های عامل و ناقد[۳]: در این روش های هم یک سیاست و هم یک تابع ارزشگذاری آموزش داده می شود.

۲-۳- پیاده سازی

برای پیاده ســـازی از کمک پروژه های دیپ مایند[۴] [13]، پای بولت-جیم پریوم، پای بولت، بیس لاینز، استیبل بیس لاینز، استیبل بیس لاینز۳ و پیاده سازی های دکتر فیل تابر[14] استفاده شده است.

۲-۳-۱-بازپخش

حافظه های بازپخش مختلفی می توانند برای عامل ها با الگوریتم های مختلف به کار روند. برای مثال از یک بافر می توان برای ذخیره تصـادفی گذار های گذشــته یک مسـیر برای اسـتفاده در الگوریتم های off-

[1] Policy Network
[2] Value Network
[3] Actor-Critic Methods
[4] DeepMind

policy و از یک قطعه[1] برای ذخیره گذار های پیوسته یک مسیر استفاده کرد. کلاس های مربوط به این حافظه های باز پخش به صورتی طراحی شده اند که به راحتی امکان آزمایش آنها با مقدار های متفاوت مثل عامل تخفیف و تعداد گام های روش اختلاف زمانی چند گامه وجود دارد. این کلاس های مسئول ذخیره دسته گذار ها برای استفاده در روش های گرادیان کاهشی دسته ای و روش های مشابه هستند.

۲-۳-۲- مدل ها

همانطور که در بخش مفاهیم پایه توضیح داده شد شبکه های عصبی تخمین زننده توابع ارزش را می توان متشکل از سه بخش دانست: ۱) بخش اول معمولا به نام کدکننده[2] شناخته می شود که در صورت استفاده از شبکه های وابسته به اقدام، تعداد نورون های آن برابر مجموع اندازه متغیر های حالت و متغیر های عمل است و در غیر این صورت صرفا تعداد متغیر های حالت. این بخش همچنین وظیفه نرمال سازی متغیر های محیط را با توجه به متغیر های ورودی قبلی به عهده دارد. ۲) بخش دوم به عنوان بدنه شبکه معمولا یک پرسپترون[3] یا پرسپترون چند لایه است که در واقع یک زیر شبکه چند لایه است که ساختار آن به عنوان فراپارامتر ساختار شبکه قابل تنظیم است و به عامل داده می شود. ۳) در لایه آخر معمولا یا به تعداد متغیر های اقدام های ممکن در محیط برای خروجی یک توزیع احتمال نورون وجود دارد و یا فقط یک نورون دارد که برای خروجی مقدار تابع ارزیاب می باشد.

۳-۳-۲- به روز رسانی متغیر های شبکه عصبی

عامل های مختلف روش های متفاوتی را برای به روز رسانی متغیر هایشان دارند. یک به روز رسان[4] معمولا دسته هایی از داده را به عنوان ورودی دریافت می کند و یک تابع هزینه و گرادیان این تابع را محاسبه می کند که در این محاسبه برخی پارامتر های مربوط به مدل را تاثیر می دهد و در نهایت این پارامتر ها را نیز به روزرسانی می کند و در برخی موارد مثل الگوریتم MPO پارامتر های جدیدی ایجاد می کند.

---

[1] Segment
[2] Encoder
[3] Perceptron
[4] Updater

۲-۳-٤- آموزش دهنده[1]

کلاس آموزش دهنده وظیفه پیاده سازی حلقه آموزش را بر عهده دارد. این کلاس برقراری ارتباط بین عامل و محیط، آزمودن عامل، ذخیره برخی نتایج در هنگام آموزش و ذخیره دوره ای وزن های دو شـــبکه را به عهده دارد.

آموزش توزیع شده سرعت یادگیری عامل را به شدت افزایش می دهد. در این روش عامل به جای ارتباط با یک محیط با چند محیط یکسان و با احتمالات متفاوت به طور همزمان ارتباط برقرار می کند. در گام زمانی t، مشاهدات O_t را که همان حالت فعلی یعنی S_t می باشد از طریق تابع گام[2] عامل به آن منتقل می کنیم. در پاسخ عامل توزیع احتمالات اقدامات ممکن را به صورت تنسور[3] At بر می گرداند و همچنین کار ذخیره برخی اطلاعات مثل احتمال لگاریتمی اقدامات و مشـــاهدات این گام زمانی را انجام می دهد. در ادامه با اعمال A_t بر محیط از طریق تابع گام محیط چندین مقدار بازگردانده می شـــود: ۱)مقادیر توصـــیف کننده گذار فعلی با اعمال A_t ۲) تنسور O'_t از مشاهدات بعدی ۳) بردار r_t از پاداش ها ۳) بردار پایانه های محیط و ۴) بردار ρ_t از بازنشـانی[4] ها. بردار بازنشـانی ها پایان اپیزود ها را مشـخص می کند. عامل از این بردار ها برای تعیین زمان بوت استرپینگ و محاسبات برگشت (G_t با کمک لامبدا) استفاده می کند. به کمک پایانه ها عامل می تواند دلیل بازنشانی را متوجه شود که از این اطلاعات برای بوت استپینگ استفاده می کند.

۲-۳-٥- عامل ها

A2C: این عامل همچنین با عنوان VPG[5] شـــناخته می شـــود. این عامل از مزیت برگشـــت ها و توابع ارزشگذاری برای به روز رسانی یک سیاست تصادفی به کمک گرادیان سیاست استفاده می کند. این عامل پایدار اســت ولی از آخرین گذارهایش فقط یک بار برای به روز رســانی عامل اســتفاده می کند و در نتیجه فرآیند آموزش آن بسیار کند است.

---

[1] Trainer
[2] Step
[3] Tensor
[4] Reset
[5] Vanilla Policy Gradient

TRPO: این عامل از همان ساختار A2C به عنوان ساختار پایه استفاده می کند ولی از یک بهینه ساز گرادیان پیوسته برای برداشتن یک گام بزرگ در جهت به روز رسانی سیاست ضمن حفظ یک شرط KL بین سیاست جدید و قدیم استفاده می کند.

PPO: این عامل TRPO را با اعمال نسبت بریده شده[1] بین سیاست قبلی که اخیر ترین گذار را تولید کرده و سیاست بهینه شده جدید تقریب می زند.

MPO: این عامل از یک هدف آنتروپی پیچیده نسبی با بهره مندی از مزیت دوگانگی بین کنترل و تقریب استفاده می کند. این عامل در صورت میزان سازی مناسب می تواند بسیار قدرتمند باشد و تنها چالش آن پیاده سازی دشوار است.

۲-٤- نتیجه گیری

در این بخش ابتدا با تعریف مفاهیم پایه و پیوسته به مفاهیم پیچیده تر رسیدیم و در نهایت با استفاده از این تعاریف به بررسی نکات مهم و قابل توجه پیاده سازی پرداختیم. همانطور که در بخش قبل ذکرشد، پیاده سازی و آموزش عامل A2C که پایه دو عامل PPO و TRPO است از پیچیدگی کمتری برخوردار است ولی همین سادگی باعث زمانبر شدن فرآیند آموزش می شود. به همین ترتیب TRPO آموزش ساده تر و پیچیدگی کمتری نسبت به A2C دارد و این مسئله در رابطه بین عامل های PPO و TRPO و عامل های PPO و MPO هم صادق است. در نهایت نتایج فصل ۴ گواه تاثیرات مثبت این پیچیدگی در نتیجه نهایی حاصل از آموزش عامل و خود فرآیند آموزش است.

[1] Clipped Ratio

فصل ۳- شبیه سازی و تحقیق

۳-۱- مقدمه

در این فصل به بررسی روش های به کار برده شده برای شبیه سازی و تحقیق، مقایسه و توسعه الگوریتم ها و محیط شبیه سازی فیزیکی، در قالب دو کتابخانه اصلی حاصل از پروژه می پردازیم.

۳-۲- شبیه سازی

همانطور که در فصل ۱ اشاره شد، برای شبیه سازی از محیط های مبتنی بر موتور فیزیکی بولت استفاده می کنیم. برای آماده سازی محیط شبیه سازی ابتدا محیط موردنظر را با استفاده از موتور فیزیک بولت[1] طراحی می کنیم. برای آموزش و آزمایش عامل از کتابخانه جیم[2] و برای رفع مشکلات احتمالی محیط طراحی شده و اطمینان از عملکرد درست این محیط از یک مدل از پیش طراحی شده در بستر کتابخانه پای تورچ[3] بهره می گیریم؛ این عامل ابتدا صرفا بر روی یک محیط ساده و بدون چالش آموزش داده شده و بعد از تنظیم عامل و میزان سازی فراپارامتر های مدل، کار را با فراپارامتر های مشابه بر روی محیط های پر چالش ادامه می دهیم. بعد از آزمایش و اطمینان از صحت عملکرد محیط های طراحی شده به پیاده سازی و آموزش مدل ها و عامل ها با استفاده از مقالات مربوطه و در بستر کتابخانه تنسورفلو می پردازیم.

درنهایت عامل ها را بر روی محیط طراحی شده در مرحله قبل آموزش داده و با توجه به نتایج حاصل و مقایسه این نتایج با نتایج مقالات و مدل بی نقص آموزش داده شده قبلی به رفع ایرادات مدل های طراحی

---

[1] Bullet
[2] Gym
[3] Pytorch

شـده جدید می پردازیم؛ به عبارتی عملکرد مدل بی نقص را معادل عملکرد سـطح انسـانی در نظر گرفته و سعی در دستیابی به دقتی بهتر از دقت انسانی می کنیم. در نهایت با توجه به محدودیت تعداد گام های هر دوره، پاداش متوسط حاصل آزمودن عامل در هر دوره به یک مقدار حداکثری میل کرده و در برخی مدل ها در طی تکرار های بعدی از این مقدار کمتر می شود.

۳-۲-۱- محیط ها

در رابطه با بحث شـبیه سـازی و محیط باید به این نکته توجه داشـت که شـیوه اصـلی آموزش عامل ها تنها، طراحی محیط های فیزیکی بسـیار دقیقی اسـت که محیط فیزیکی واقعی را شـبیه سـازی می کنند و امکان آموزش دادن عامل حداقل از ابتدای کار به دلایل زیر امکان پذیر نیست:

- زمانبر بودن آموزش عامل در دنیای واقعی
- نبود معیار درستی برای سنجش عملکرد عامل در بسیاری از چالش ها
- مصرف بسیار زیاد انرژی برای حصول نتیجه قابل قبول
- سخت شدن رفع مشکلات احتمالی کد
- غیر ممکن شدن میزان سازی

و بسـیاری مشـکلات دیگر که عملا آموزش عامل در محیط های واقعی را غیر عملی می کند. در صـورتی که عامل پس از آموزش در محیط شـبیه سـازی نتواند در دنیای واقعی به عملکرد مورد انتظار و مشـابه شـبیه سـازی دسـت یابد باید به این نکته توجه داشـت که پارامتر های زیادی برای شـبیه سـازی دنیای واقعی در موتور های معمول و پر اسـتفاده فیزیکی وجود دارند که با تنظیم درسـت می توانند نمود درسـت تری از دنیای واقعی مورد نظر پروژه ارائه دهند. همچنین در صـورت نیاز می توان به اسـتفاده از محیط های بهتر و پیچیده تر با پارامتر های حائز اهمیت بیشـتری پرداخت که حاصـل شـبیه سازی را به نتیجه نهایی در دنیای واقعی نزدیک تر می کند. استفاده از محیط های شبیه سازی شده با کمک موتور های فیزیک امکان آموزش

در زمان بسیار کم تر، با انعطاف بسیار بیشتر و با صرف انرژی کم تر را در کنار مزایای بسیار دیگر فراهم می کنند.

همانطور که اشاره شد، برای ساخت محیط شبیه سازی از یک موتور فیزیکی بهره می گیریم و برای برقراری ارتباط استاندارد عامل ها با این محیط از یک واسط مثل جیم استفاده می کنیم. محیط آموزش و آزمایش عامل را می توان متشکل از دو بخش در نظر گرفت: ۱) ربات ۲)محیط فیزیکی منفک از ربات

به طور کلی می توان گفت هرچیزی که عامل با انتخاب اقدامات ممکن در محیط به آن نیرو وارد می کند و در هر اعمال عمل عامل، این نیرو به آن وارد می شود جزو ربات به حساب می آید و محیطی که نماینده دنیای فیزیکی ای است که ربات در آن قرار می گیرد و هر اعمال عملی لزوما بر آن تاثییر نمی گذارد، محیط منفک از ربات به حساب می آید. در ادامه به بررسی نحوه ساخت و شبیه سازی این دو بخش می پردازیم.

۳-۲-۱-۱- ساخت ربات ها

برای ساخت ربات ها از فایل های یو آردی اف، اکس ام ال و اس دی اف که به کمک ابزار هایی مثل راس[1]، سالید ورکز[2] و اتودسک[3] ساخته می شود استفاده می کنیم. در نهایت می توان این ربات ها را در محیط هایی مانند گزبو[4] شبیه سازی و آزمایش کرد. در ادامه به بررسی چند ابزار مرتبط با بحث شبیه سازی می پردازیم.

اکس ام ال[5] [15] : اکس ام ال یا زبان نشانه گذاری توسعه پذیر یک زبان نشانه گذاری است که برای ذخیره، انتقال و ساخت موقت داده ها به کار می رود. این نوع فایل ها یک مجموعه از قواعد را برای کد کردن فایل ها اعریف می کنند که هم برای انشان و هم برای ماشین قابل فهم است.

---

[1] ROS: Robot Operating System
[2] Solidworks
[3] Autodesk
[4] Gazebo
[5] Extensible Markup Language

یو آر دی اف[1] [16]: یو آر دی اف یا فرمت شرح یکپارچه ربات ها یک تبیین به فرمت اکس ام ال از ربات هاست که در موارد آموزشی و صنعتی برای مدل کردن سیستم های چندجسمه[2] مانند دست های رباتیک مورد استفاده در خطوط تولید و ربات های مورد استفاده در پارک های کودکان به کار می رود. مانند انواع دگیر فایل های اکس ام ال فایل های یو آر دی اف از چندین برچسب[3] اکس ام ال مانند ربات، لینک و مفصل که به صورت سلسله مراتبی درون همدیگر قرار گرفته اند و درختان اکس ام ال[4] را می سازند تشکیل شده اند.

راس[17] : سیستم عامل ربات یا راس، ابزار و کتابخانه هایی را برای کمک به توسعه نرم افزار های رباتیک در اختیار توسعه دهندگان قرار می دهد. این نرم افزار تمام وظایف یک سخت افزار مانند تجرید سخت افزار، مدیریت بسته های اطلاعاتی، انتقال پیام بین پردازه ها و کتابخانه های مربوطه را در اختیار می گذارد. این نرم افزار متن باز و تحت لیسانس بی اس دی[5] است.

گزبو[18] : یک شبیه ساز متن باز ۳ بعدی ربات می باشد که موتور فیزیک او دی ای[6]، اپن جی ال[7] و کد پشتیبان برای شبیه سازی سنسور ها و عملگر ها ارائه می دهد. گزبو امکان استفاده از چندین موتور فیزیک با قدرت اجرای بالا مثل اودی ای و بولت را در اختیار می گذارد.

جهت ساخت ربات ها نیازمند ساخت و بارگذاری فایل اکس ام ال آنها هستیم. برای این منظور می توان از ابزار معرفی شده استفاده کرد؛ همچنین باید در نظر داشت که در طراحی ربات باید به مواردی مانند اثر تصادم اجزای ربات با محیط و با دیگر اجزای ربات، وزن، نیروی جاذبه، قدرت بازوی ربات، میزان الکتریسیته مصرف شده در حرکات ربات و اصطکاک سطوح ربات در تماس با محیط خارجی توجه شود؛ همچنین تعیین موقعیت اولیه ربات در محیط یعنی مختصات مرکز ثقل ربات و ارتفاع اولیه در فایل اکس ام ال مشخص می شود و در هنگام بارگذاری در برنامه امکان تغییر در این مشخصات صرفا برای فایل های یو آر دی اف وجود دارد. به طور کلی وجود و حرکت ربات در محیط یک فرآیند پیوسته است که امکان تقسیم این فرآیند به چند اپیزود از طریق مشخص کردن یک وضعیت اولیه ثابت و محدوده تعداد حرکت های ربات وجود دارد. بنابراین در هر اپیزود عامل صرفا تعدادی عمل بر روی ربات اعمال کرده و بعد از آن ربات به وضعیت اولیه باز می گردد که این وضعیت اولیه در واقع همان رها شدن ربات از ارتفاع اولیه در نقطه مبدا مشخص شده

---

[1] Unified Robotics Description Format
[2] Multibody
[3] Tag
[4] XML trees
[5] BSD
[6] ODE
[7] OpenGL

در فایل اکس ام ال می باشد. برای ساخت ربات و شبیه سازی در حین ساخت جهت جلوگیری از وقوع اشتباه نیازمند استفاده از ابزاری مشابه راس و گزبو می باشیم.

۳-۲-۱-۲- ساخت چالش ها

برای ساخت چالش ها نیازمند ایجاد زمین[1] با پستی بلندی هستیم. برای ایجاد زمین و پستی بلندی دو راه داریم: ۱) آپلود اشیا سازنده زمین و پستی بلندی از طریق فایل شی[2] با پسوند فایل آبجکت[3] و فایل های مشابه و ساخت جزء جزء زمین از طریق این فایل ها ۲) بارگذاری تمام پستی بلندی های موردنظر از طریق یک فایل متنی یا عکس که درآن میزان بلندی یک نقطه برابر مقدار پیکسل تصویر یا عدد متناظر آن نقطه در فایل متنی می باشد و ارتفاع نهایی هر نقطه به طور نسبی با تعیین ابعاد پستی بلندی بارگذاری شده در محیط تعیین می شود. در صورت بارگذاری جزء جزء اشیا از انعطاف بیشتری برای شکل دهی چالش برخورداریم ضمن اینکه توان محاسباتی بیشتری برای بارگذاری محیط ایجاد شده نیاز می شود و پردازنده مورد استفاده باید قدرت محاسباتی بیشتری داشته باشد. در صورتی که بارگذاری پستی و بلندی از طریق تصویر انجام شود بار محاسباتی محیط ایجاد شده کمتر می شود، به طوری که پستی بلندی های وسیع نیز به راحتی و از طریق یک واحد پردازشی نه چندان قدرتمند در محیط به راحتی ایجاد شده و نمایش داده می شوند.

۳-۳- تحقیق

برای پیدا کردن عامل های مناسب برای حل چالش های مطرح شده در بخش های قبل، به ترتیب مراحل زیر انجام شده اند.

انتخاب الگوریتم ها

در بین الگوریتم های مبتنی بر مقدار و گرادیان-سیاست، ترجیح بر استفاده از الگوریتم های گرادیان-سیاست است که دلیل آن مزایای زیر می باشد:

۱- سیاست ها می توانند هر تابع قابل آموزشی انتخاب شوند.

۲- آموزش راحت تر سیاست های تصادفی

---

[1] Terrain

[2] Object file

[3] .obj

۳- عملکرد بهتر در محیط های تا اندازه ای قابل مشاهده[1]

۴- نمود راحت تر سیاست ها در مقایسه با توابع مقدار با کمک توابع تخمین

۵- در روش های مبتنی بر مقدار از عملگر های خصمانه[2] برای تغییر تابع مقدار استفاده می شود ولی در روش های مبتنی بر سیاست از گرادیان مبتنی بر سیاست های تصادفی استفاده می شود که اعمال منتخب عامل را به آرامی تغییر می دهند و در صورت دنبال کردن گرادیان سیاست، حداقل به یک بهینه محلی همگرا می شوند.

۶- و ...

بین الگوریتم های مبتنی بر مقدار و مبتنی بر سیاست برخی الگوریتم ها مشترک هستند که به آنها الگوریتم های عامل و ناقد می گوییم. این عامل ها ضمن پیچیدگی بیشتر نسبت به دو دسته دیگر از الگوریتم ها از پتانسیل و پیچیدگی بالاتری برای حل مسائل برخوردارند.
در بین الگوریتم های عامل و ناقد، از ساده ترین عامل ها، عامل A3C است که پایه الگوریتم A2C می باشد. به عبارتی با توجه به مطالب فصل ۲ می توان مجموعه عامل های A3C، A2C، TRPO و PPO را عضو یک مجموعه عامل های هم خانواده دانست؛ بنابراین با پیاده سازی الگوریتم A2C عملا پیاده سازی الگوریتم های دیگر این خانواده بسیار آسان خواهد بود. از طرف دیگر می توان الگوریتم های DDPG، TD3، SAC و TD4 را هم در دسته مشابهی از الگوریتم های هم خانواده قرار داد.

پیاده سازی

در این پروژه پیاده سازی بیشتر الگوریتم های پر استفاده یادگیری تقویتی عمیق انجام شده است. الگوریتم های پیاده شده در این پروژه شامل: ۱) A2C ۲) PPO ۳) TRPO ۴) MPO ۵) SAC ۶) DDPG ۸) TD3 می باشد. شایان ذکر است که در این پیاده سازی ها صرفا الگوریتم پیاده شده مورد نظر است و پیاده سازی برای محیط های مختلف موجود در پروژه پای بولت اعم از پیوسته و گسسته انجام شده است. برای اطمینان از صحت پیاده سازی، به ازای هر الگوریتم، پیاده سازی های متفاوت با استفاده از روش های گوناگون انجام شده است. این پیاده سازی ها گاها هم به با استفاده از کتابخانه تنسورفلو و هم با استفاده از کتابخانه پای تورچ انجام شده اند.

---

[1] Partially Observable

[2] Aggressive

آموزش

بعد از پیاده سازی هر عامل، ساعت ها آموزش جهت تنظیم فراپارامتر ها، بهبود و رفع نقص های احتمالی لازم است. شایان ذکر است که رفع نقص الگوریتم های یادگیری تقویتی به دلیل زمان بر بودن آموزش و نداشتن یک خط مشی ثابت بسیار زمانبر و دشوار است و گاها بسیاری از نقص ها عملا یافت نمی شوند؛ به همین جهت فراپارامتر های بیشتر الگوریتم ها در ابتدا مشابه عامل های موفق پیاده شده در پروژه های دیگر و برای محیط های ساده تر و مشابه مقدار دهی شده و به مرور با توجه به نحوه عملکرد عامل تغییر داده شده اند.

دست چین الگوریتم های نتیجه بخش

از بین الگوریتم های پیاده شده، صرفا بخشی از آنها که از پیچیدگی کافی برخوردارند در محیط های پرچالش پیاده شده امکان آموزش موثر را دارند و بسیاری از آنها به دلیل پیچیدگی های زیاد الگوریتم و تنظیم بسیار زمانبر فراپارامتر ها، عملا حتی برای محیط های بدون چالش نیز پاسخگو نیستند.

۳-٤- نتیجه گیری

در نهایت با توجه به نتایج حاصل از چرخه آموزش و میزان سازی و بهبود الگوریتم ها، خانواده مشتق از الگوریتم A3C برای حل ۶ چالش اصلی پیاده شده انتخاب شدند.

فصل ۴- نتایج حاصل از تحقیق و شبیه سازی

٤-١- مقدمه

در این فصل به بررسی نتایج نهایی حاصل از آموزش عامل ها می پردازیم.

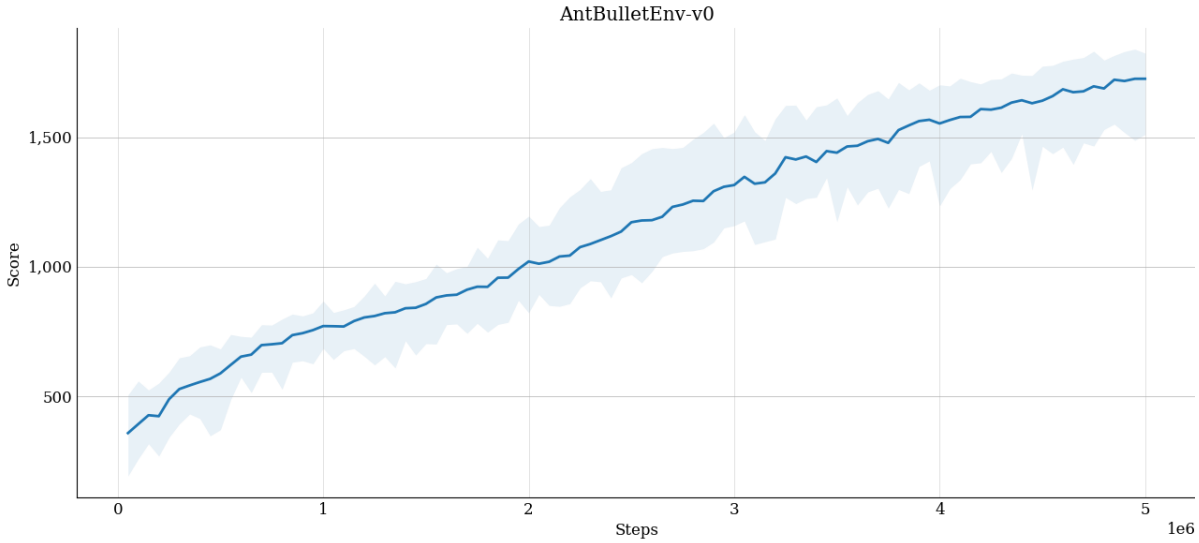
٤-٢- نتایج به دست آمده

همانطور که در فصل سوم بررسی شد، از بین پیاده سازی های مختلف انجام شده، تنها پیاده سازی خانواده A3C در محیط های با و بدون چالش نتیجه بخش بود. با توجه به اینکه هدف پروژه طراحی عامل هایی برای حل محیط های پر چالش بود و در زمینه محیط AntBulletEnv-v0 پروژه های زیادی به نتیجه رسیده اند، در ادامه صرفا عامل های منتخب در فصل سوم بررسی می شوند.

٤-٢-١- محیط بدون چالش

در این زیربخش به بررسی نتایج عامل های منتخب، در محیط ساده و بدون چالش AntBulletEnv-v0 می پردازیم. شایان ذکر است که متوسط امتیاز مورد انتظار که در نتیجه آن حرکت ربات در محیط قابل قبول خواهد بود در بازه ۲۱۰۰ تا ۲۵۰۰ می باشد.

۱-۱-۲-۴- A2C



شکل ۱ نمودار امتیاز کسب شده توسط عامل A2C نسبت به گام های آموزش

همانطور که در فصل دوم ذکر شد، این عامل صرفا به عنوان پایه ای برای بنای دیگر الگوریتم های مشابه یعنی TRPO و PPO پیاده شده است و به دلیل محدودیت برای استفاده یکباره از آخرین گذار ها بسیار کند می باشد در ادامه الگوریتم های مشتق از این الگوریتم بررسی می شوند که با ارائه روش هایی سرعت یادگیری بسیار بهتری نسبت به این الگوریتم دارند.

فراپارامتر های پیدا شده برای حصول این نتیجه به شرح زیر است:

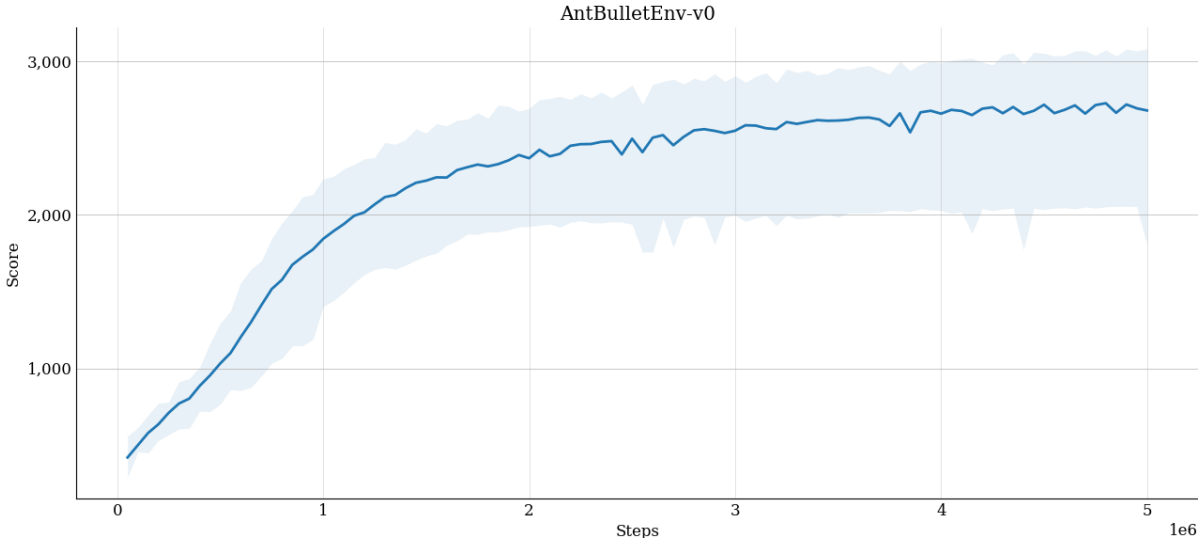
- ابعاد لایه های مخفی هر دو شبکه: ۶۴ و ۶۴
- تابع فعال ساز لایه های بدنه : تانژانت هذلولوی[1]
- تابع فعال ساز لایه خروجی(شبکه عامل): تانژانت هذلولوی
- نرخ آموزش[2] (بهینه ساز آدام): ۰.۰۰۰۳
- حداقل مقدار برش: ۰.۰۰۰۱
- حداکثر مقدار برش: ۱
- توزیع احتمال لایه خروجی: توزیع نرمال چند متغیره
- اپسیلن(بهینه ساز آدام): ۱ در ۱۰ به توان منفی هشت (1e-8)

[1] Tanh: hyperbolic tangent

[2] Learning Rate: alpha

- اعمال تاثیر آنتروپی: خیر

۲-۱-۲-۴- TRPO



شکل ۲ نمودار امتیاز کسب شده توسط عامل TRPO نسبت به گام های آموزش

همانطور که از بررسی های فصل ۲ انتظار می رفت، سرعت آموزش عامل TRPO نسبت به A2C بهبود یافته ولی همچنان از حد مطلوب فاصله دارد. با توجه به سرعت اجرای الگوریتم ها برای روی CPU می توان گفت که اجرای الگوریتم به مدت دو ساعت برای حصول نتیجه مورد نظر می تواند مطلوب باشد که در این مورد، طبق نمودار، نتیجه حاصل در بین گام های زمانی ۱ تا دو میلیون می باشد که از حد مطلوب کمی فاصله دارد.

فراپارامتر های پیدا شده برای حصول این نتیجه به شرح زیر است:

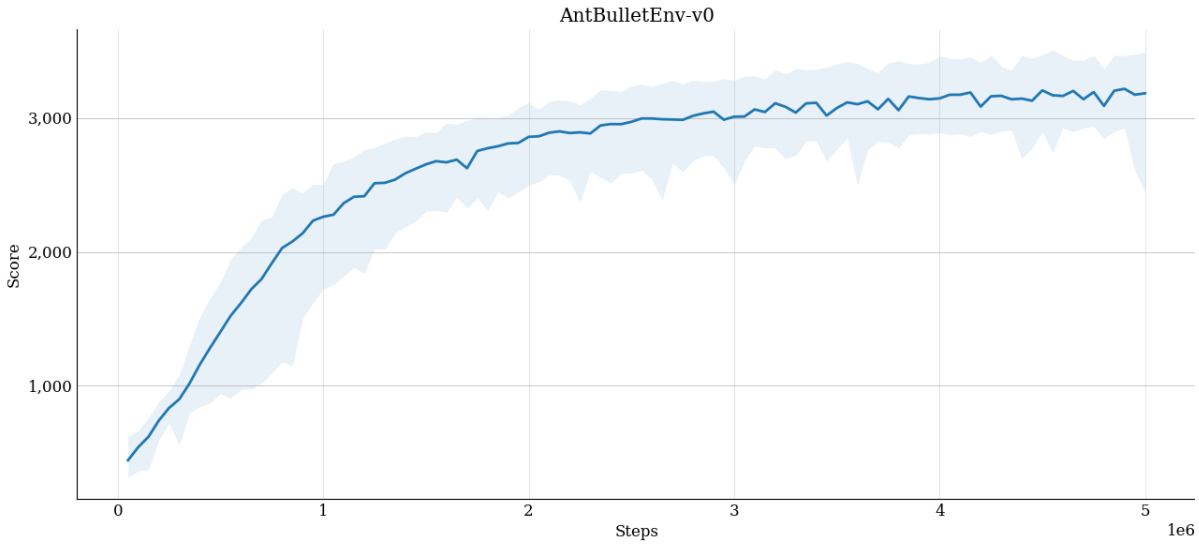
- ابعاد لایه های مخفی هر دو شبکه: ۶۴ و ۶۴
- تابع فعال ساز لایه های بدنه : تانژانت هذلولوی[1]
- تابع فعال ساز لایه خروجی(شبکه عامل): تانژانت هذلولوی
- نرخ آموزش[2] (بهینه ساز آدام): ۰.۰۰۰۳
- حداقل مقدار برش: ۰.۰۰۰۱
- حداکثر مقدار برش: ۱

[1] Tanh: hyperbolic tangent

[2] Learning Rate: alpha

- توزیع احتمال لایه خروجی: توزیع نرمال چند متغیره
- اپسیلن(بهینه ساز آدام): ۱ در ۱۰ به توان منفی هشت (1e-8)
- اعمال تاثیر آنتروپی: خیر

۳-۱-۲-۴- PPO



شکل ۳ نمودار امتیاز کسب شده توسط عامل PPO نسبت به گام های آموزش

می توان عامل PPO را نقطه عطفی برای حصول نتایج مطلوب دانست. همانطور که دیده می شود در طی ۳ میلیون گام، این عامل متوسط امتیاز ۳۰۰۰ را رد کرده است که در مقایسه با دو عامل قبل نتیجه قابل توجهی است.

فراپارامتر های پیدا شده برای حصول این نتیجه به شرح زیر است:

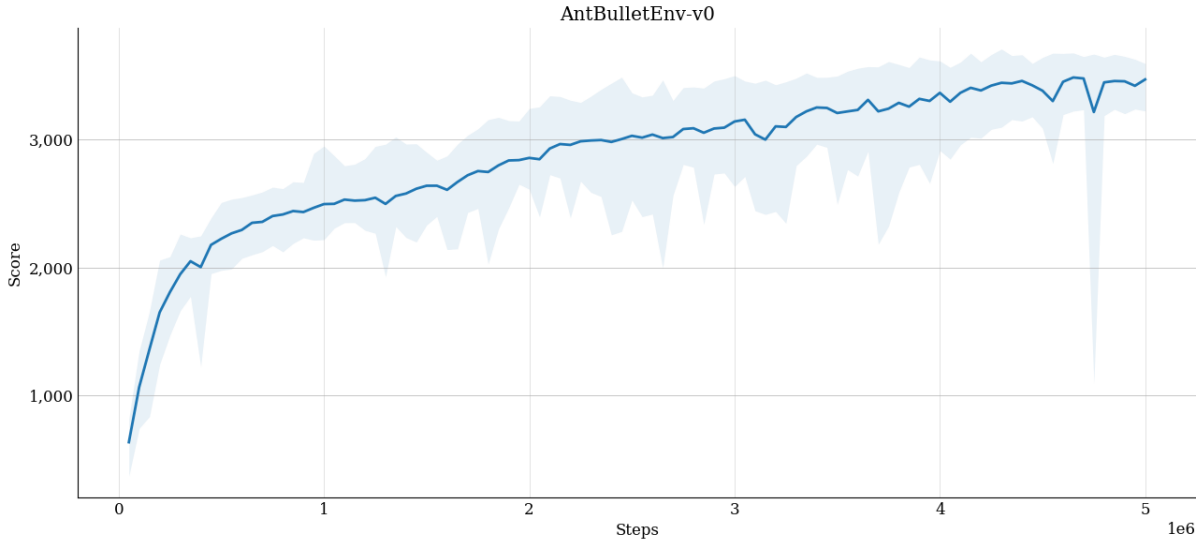
- ابعاد لایه های مخفی هر دو شبکه: ۶۴ و ۶۴
- تابع فعال ساز لایه های بدنه : تانژانت هذلولوی[1]
- تابع فعال ساز لایه خروجی(شبکه عامل): تانژانت هذلولوی
- نرخ آموزش[2] (بهینه ساز آدام): ۰.۰۰۰۳
- حداقل مقدار برش: ۰.۰۰۰۱
- حداکثر مقدار برش: ۱

[1] Tanh: hyperbolic tangent

[2] Learning Rate or lr or alpha

- توزیع احتمال لایه خروجی: توزیع نرمال چند متغیره
- اپسیلن(بهینه ساز آدام): ۱ در ۱۰ به توان منفی هشت (1e-8)
- اعمال تاثیر آنتروپی: خیر
- حد آستانه KL: ۰.۰۱۵
- نرخ برش: ۰.۲
- فعال ساز مقیاس: سافت پلاس[1]

۴-۱-۲-۴- MPO



شکل ۴ نمودار امتیاز کسب شده توسط عامل MPO نسبت به گام های آموزش

در مقایسه با الگوریتم PPO سرعت رشد امتیاز این الگوریتم بیشتر است و الگوریتم MPO تقریبا بعد از ۲ میلیون گام به امتیاز ۳۰۰۰ می رسد.

همانطور که در نمودار این دو الگوریتم قابل مشاهده است، الگوریتم PPO بعد از رسیدن به امتیاز ۳۰۰۰ با سرعت کمتری رشد می کند درصورتی که سرعت رشد الگوریتم MPO نشان می دهد که پتانسیل این الگوریتم برای کسب امتیازات بالاتر بیشتر است.

فراپارامتر های پیدا شده برای حصول این نتیجه به شرح زیر است:

- ابعاد لایه های مخفی هر دو شبکه: ۲۵۶ و ۲۵۶

[1] SoftPlus

- تابع فعال ساز لایه های بدنه : رلو[1]
- تابع فعال ساز لایه خروجی(شبکه عامل): رلو
- تعداد نمونه ها: ۲۰
- اپسیلن: ۰.۱
- خطای اپسیلن: ۰.۰۰۱
- میانه اپسیلن: ۰.۰۰۱
- واریانس اپسیلن: ۰.۰۰۰۰۰۱
- میانگین اولیه لگاریتمی آلفا: ۱
- میانه اولیه لگاریتمی آلفا: ۱۰
- حداقل لگاریتم دوتایی: منفی هجده
- برش گرادیان: ۰ (در ابتدا بدون تاثیر)
- درجه اولیه لگاریتم: ۱

٤-۲-۲-محیط های چالشی

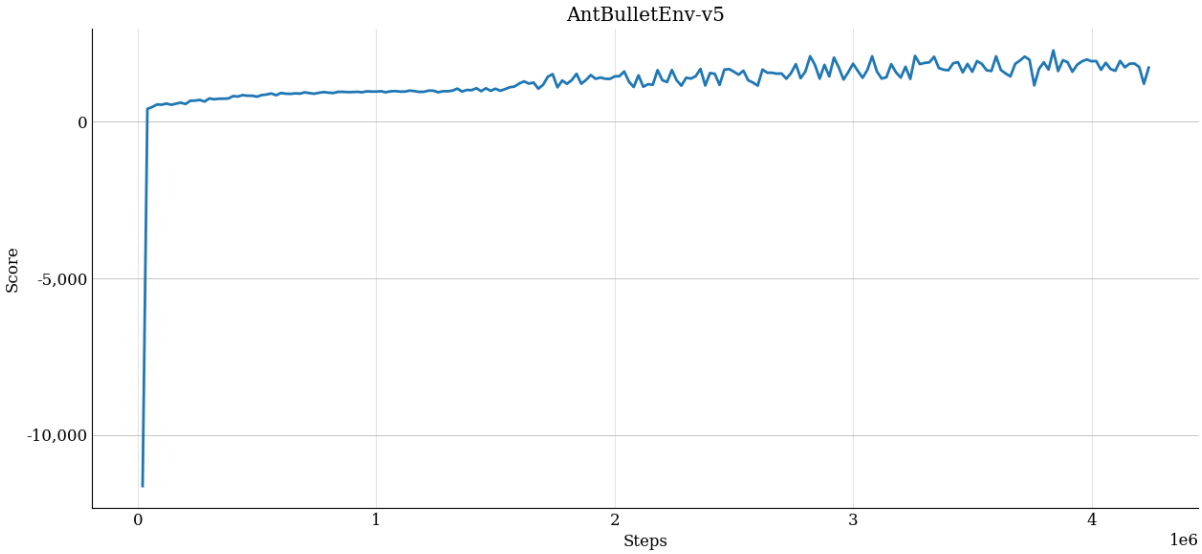
در این زیربخش به بررسی نتایج عامل های منتخب، در محیط های چالشی طراحی شده با عنوان TrainingAntBulletEnv-v0 و AntBulletEnv-v5 می پردازیم. در این محیط ها مقادیر فراپارامتر ها با حالت قبل یکی است.

۱-۲-۲-۴- A2C

همانطور که در فصل دوم بیان شد، روند آموزش الگوریتم A2C بسیار طولانی است به طوری که در طی ۷ میلیون گام، در محیط TrainingAntBulletEnv-v0، متوسط امتیاز به ۱۰۰۰ می رسد.

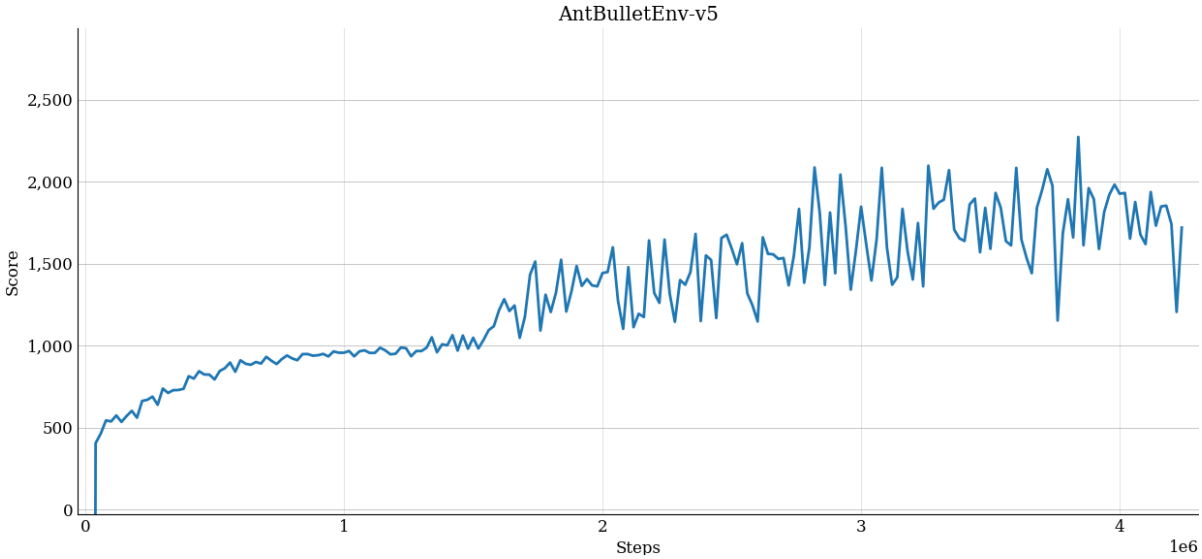
[1] Relu

۲-۲-۲-۴- TRPO



شکل ۵ نمودار امتیاز کسب شده توسط عامل TRPO نسبت به گام های آموزش

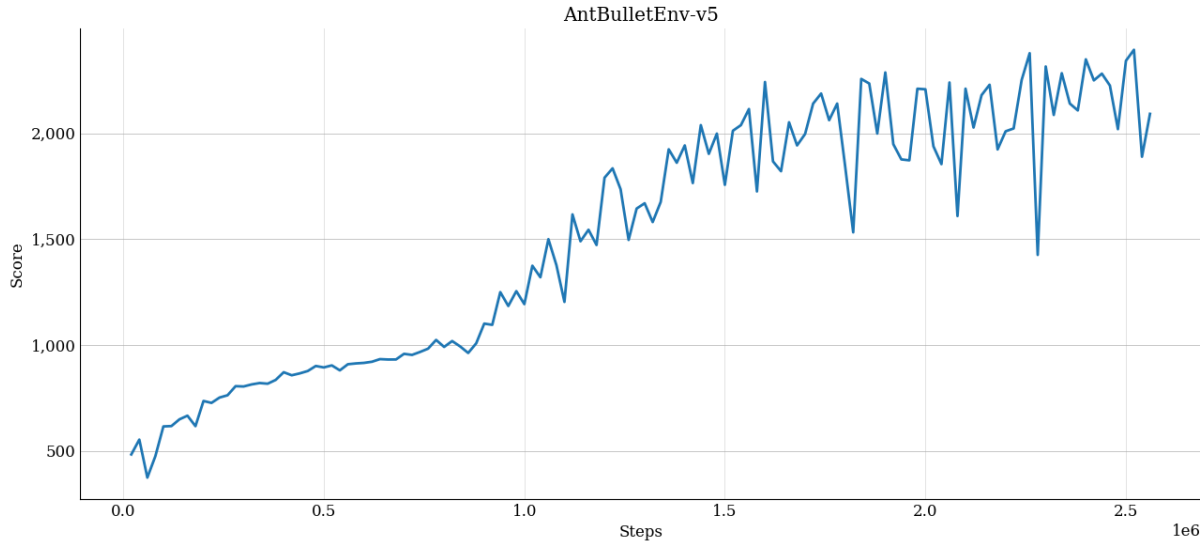
همانطور که در نمودار بالا دیده می شود، در گام های ابتدایی آموزش، امتیاز منفی کسب شده که دلیل آن می تواند عامل نویز و یا دیگر عوامل تصادفی محیط و یا عامل باشد. این امتیاز در روند کلی آموزش تاثیر بسزایی ندارد.



شکل ۶ نمودار امتیاز کسب شده توسط عامل TRPO نسبت به گام های آموزش با تمرکز بر قسمت نا منفی

در نمودار فوق صرفا بخش نامنفی نمودار اصلی نمایش داده شده است. همانطور که در بخش قبلی تعریف شد و در این نمودار مشاهده می شود، امتیاز این عامل در بازه زمانی مطلوب به حد مورد نظر نرسیده است.

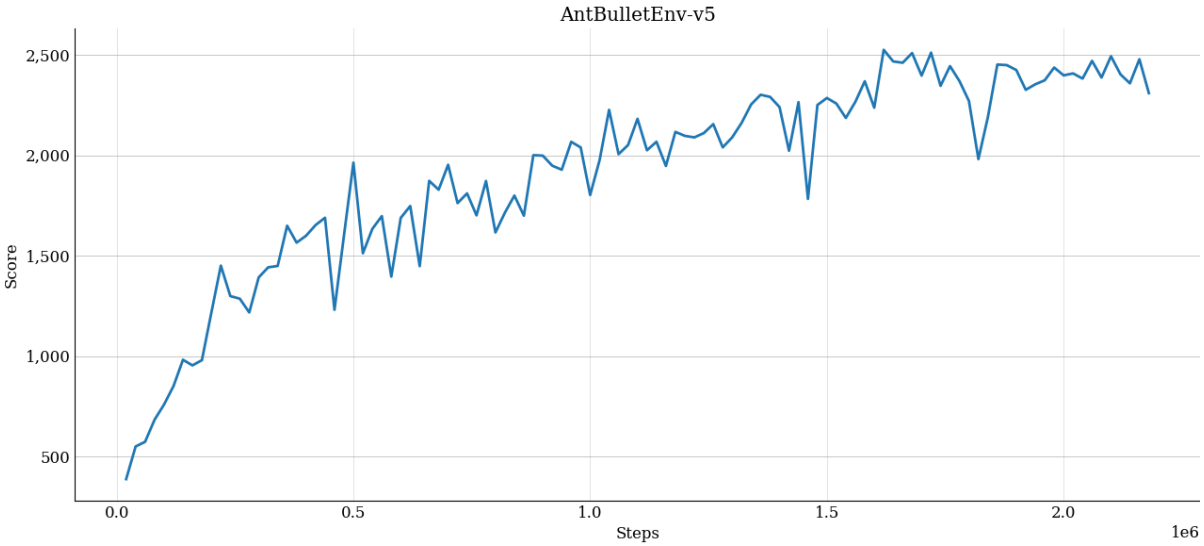
۳-۲-۲-۴- PPO



شکل ۷ نمودار امتیاز کسب شده توسط عامل PPO نسبت به گام های آموزش

در اینجا برخلاف دو مورد قبل در بازه زمانی مورد نظر امتیاز مطلوب کسب شده است. همچنین توجه به این نکته ضروری است که ضمن افزایش سرعت رشد بعد از گام ۱ میلیون ام، میزان تغییرات و افت و خیز های پاداش های کسب شده بیشتر شده است که دلیل آن می تواند پارامتر های متغیر مدل باشد.

۴-۲-۲-۴- MPO

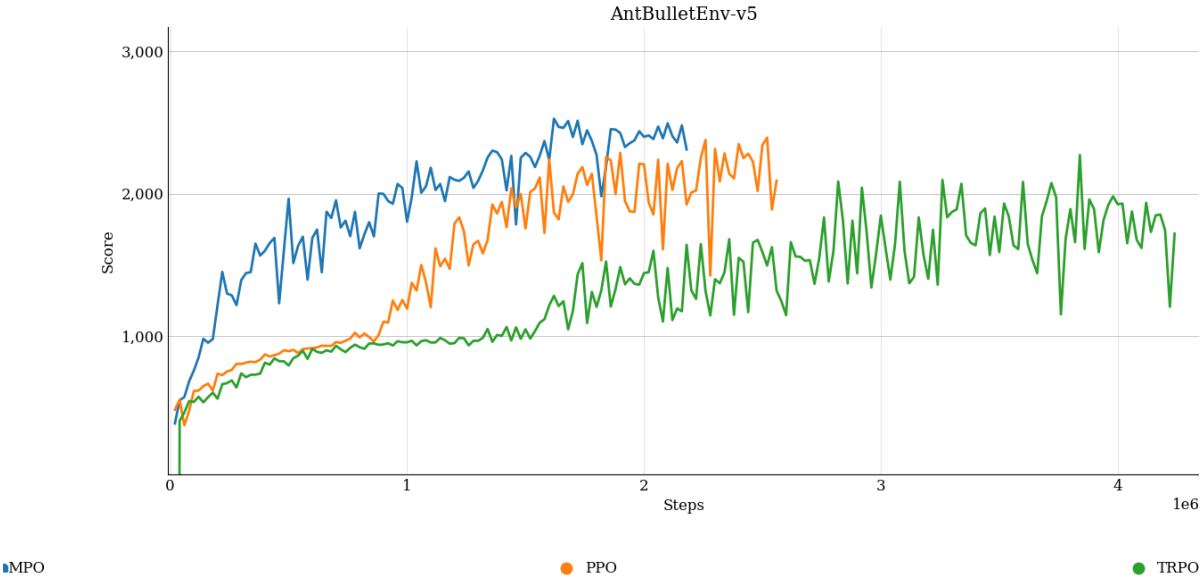


شکل ۸ نمودار امتیاز کسب شده توسط عامل MPO نسبت به گام های آموزش

مشابه محیط بدون چالش، همچنان عملکرد MPO از هر سه الگوریتم دیگر بیشتر است و با سرعت رشد یکنواخت، در زمانی کمتر از زمان مطلوب، به امتیاز مورد نظر دست یافته است.

٤-٣- نتیجه گیری

از مقایسه دو زیر بخش اصلی بخش قبل می توان به این نتیجه رسید که طراحی محیط آموزشی چالشی به گونه ای بوده اســت که عامل ها بدون تغییر فراپارامتر ها توانســته اند در بازه زمانی مشــابه به نتایج تقریبا یکسانی در دو محیط دست یابند.



شکل ۹ مقایسه سه الگوریتم برتر در محیط چالشی

در نمودار فوق سرعت آموزش، میزان تغییرات در سطح جایزه دریافتی و روند افزایش این تغییرات با افزایش گام های آموزش و سرعت و شتاب یادگیری عامل ها به خوبی مقایسه می شود. همانطور که دیده می شود در تمام موارد ذکر شده عامل MPO بهتر از سایر عامل ها عمل می کند.

فصل ۵- نتیجه گیری و پیشنهادات

۵-۱- نتیجه گیری

بهترین عامل ارائه شـده تاکنون، برای حل محیط AntBulletEnv-v0، عامل SAC در پروژه اسـتیبل بیس لاینز۳ می باشـد که از سـاختار میانی ۴۰۰ و ۳۰۰ به ترتیب اسـتفاده می کند. این عامل به کمک کتابخانه پای تورچ و در زبان پایتون پیاده سازی شده است. در اینجا با توجه به ثابت بودن تعداد متغیر های حالت و عمل که برابر تعداد متغیر های حالت و عمل محیط AntBulletEnv-v0 می باشـد از مقایسـه لایه ورودی و خروجی عامل های مختلف در محیط های چالشـــی و بدون چالش خودداری می کنیم. بهترین عامل پیاده شده در این پروژه، MPO می باشد که با ساختار میانی ۶۴ و ۶۴ به نتایج مشابهی دست می یابد. در مورد مقایسـه سـرعت آموزش می توان گفت که عوامل مختلفی در این زمینه می توانند تاثیر گذار باشـند ولی به طور کلی می توان گفت که در صـورت اسـتفاده از سـخت افزار یکسـان، سـرعت آموزش عاملSAC،۳ برابر سرعت آموزش عامل MPO طراحی شده در این پروژه است.

۵-۲- پیشنهادات

همانطور که بررسـی شـد، کار های انجام شـده در پروژه را می توان در دو شـاخه اصـلی که همان دو کتابخانه حاصـــل از پروژه می باشـــد دســـته بندی کرد. در زمینه ادامه فعالیت بر روی پروژه نیز امکان فعالیت در دو زمینه وجود دارد؛ بنابراین در ادامه پیشـنهادات برای ادامه فعالیت بر روی پروژه را در دو قسـمت بررسـی می کنیم.

۵-۲-۱- محیط ها

در زمینه محیط ها، چالش های زیر از جمله چالش های محیط های صنعتی و طبیعی به حساب می آیند که کار کردن روی آنها می تواند برای بسیاری از کاربرد های آینده این پروژه و پروژه های مشابه مفید باشد:

۱- ایجاد محیط پله کانی: برای محیط آموزشی باید پله ها به صورت مخروطی و طبقه طبقه در اطراف نقطه شروع ربات قرار بگیرند. در این صورت امتیاز نهایی ربات فارغ از جهت تصادفی مشخص شده محاسبه خواهد شد.

۲- استفاده از ساختار های دیگر برای ربات: همانطور که در فصل های قبل بررسی شد، ربات مورچه از یک کره و چهارپای دو بند ساخته شده است که برای پیمودن مسیر های بسیار دشوار نامناسب می باشد. یکی از فعالیت های بسیار مفید می تواند آموزش عامل ها با ربات هایی با ساختار های دیگر باشد. برای مثال تعداد بند های هر پا، تعداد پاها و یا ساختار کروی بدنه می تواند تغییر کند.

۳- مقید کردن ربات های چهارپای دیگر: علاوه بر دو محیط اصلی برای ربات مورچه، محیط های دیگری نیز که برای آموزش ربات های چهارپای دیگر به کار می روند در کتابخانه محیط ها قرار گرفته اند ولی در این محیط ها بازه حرکت اجزای متحرک ربات ها بیشتر از حد مورد نیاز است. این بازه گسترده، آموزش ربات ها را دشوار می کند و ممکن است نتیجه نهایی مطلوب، بهینه و به صرفه نباشد. همچنین باید توجه داشت که محدود سازی بیش از حد بازه حرکت ربات ها عملا مفهوم آموزش عامل برای حرکت دادن ربات ها را از بین می برد. تنظیم دقیق این بازه برای هر ربات متفاوت است.

در کنار دو کتابخانه اصلی پروژه، یک کتابخانه کوچکتر با استفاده مجدد از پروژه رکس-جیم [4] طراحی شده که به خوبی اثر محدود سازی بیش از حد ربات های چهارپا در این پروژه و عدم تاثیر چشم گیر عامل طراحی شده بر روی نتیجه حاصل را نشان می دهد.

۵-۲-۱- عامل ها

در این پروژه پایه مناسبی برای الگوبرداری و تکرار ارائه شده است که می تواند برای پیاده سازی الگوریتم های دیگری که در بخش تاریخچه در پروژه های مشابه پیاده شده اند استفاده شود. برای این بخش پیاده سازی الگوریتم های دیگر مخصوص فضای عمل پیوسته و مقایسه برای یافتن الگوریتمی با عملکرد بهتر از ام پی او مد نظر است؛ همچنین به طور مشابه می توان به پیاده سازی الگوریتم های مخصوص فضای عمل گسسته مثل آونگ[1] و آونگ معکوس یا کارت پل[2] پرداخت.

بهبود چهار الگوریتم پیاده شده از طریق صرف زمان بیشتر برای تنظیم فراپارامتر ها و دستیابی به عملکرد بهتر در زمینه مورد نظر با توجه به کاربرد نیز می تواند درجهت آسان کردن استفاده های صنعتی از پروژه مفید باشد.

[1] Pendulum
[2] CartPole

فهرست مرجع‌ها

[1] E. C. a. Y. Bai, "PyBullet, a Python module for physics simulation for games, robotics and machine learning," 2016--2021. [Online]. Available: http://pybullet.org.

[2] G. B. a. V. C. a. L. P. a. J. S. a. J. S. a. J. T. a. W. Zaremba, "OpenAI Gym," 2016. [Online]. Available: https://github.com/openai/gym.

[3] B. Ellenberger, "PyBullet Gymperium," 2018--2019. [Online]. Available: https://github.com/benelot/pybullet-gym.

[4] N. Russo, "rex-gym," 2018. [Online]. Available: https://github.com/nicrusso7/rex-gym.

[5] S. Tunyasuvunakool, A. Muldal , Y. Doron , S. Liu , S. Bohez , J. Merel , T. Erez , T. Lillicrap , N. Heess and Y. Tassa, "dm_control: Software and tasks for continuous control," *Software Impacts,* p. 100022, 2020.

[6] P. a. H. C. a. K. O. a. N. A. a. P. M. a. R. A. a. S. J. a. S. S. a. W. Y. a. Z. P. Dhariwal, "OpenAI Baselines," GitHub, 2017.

[7] A. a. R. A. a. E. M. a. G. A. a. K. A. a. T. R. a. D. P. a. H. C. a. K. O. a. N. A. a. P. M. a. R. A. a. S. J. a. Hill, "Stable Baselines," *GitHub repository,* 2018.

[8] A. R. a. A. H. a. A. G. a. A. K. a. M. E. a. N. Dormann, "Stable-Baselines3: Reliable Reinforcement Learning Implementations," *Journal of Machine Learning Research,* vol. 22, no. 268, pp. 1-8, 2021.

[9] S. G. a. A. K. a. O. R. a. P. C. a. E. H. a. S. F. a. K. W. a. E. G. a. N. W. a. E. K. a. L. S. a. J. S. a. G. B. a. J. B. a. C. Harri, "{TF-Agents}: A library for Reinforcement Learning in TensorFlow," 2018. [Online]. Available: https://github.com/tensorflow/agents. [Accessed 25 June 2019].

[10] J. Brændshøi, . A. Shevale, T. Thelen and K. , "Pymaze," Github, 2016. [Online]. Available: https://github.com/jostbr/pymaze.

[11] R. S. S. a. A. G. Barto, Reinforcement Learning An Introduction, 2018.

[12] M. Morales, Grokking Deep Reinforcement Learning, 2022.

[13] M. H. a. B. S. a. J. A. a. G. B.-M. a. F. B. a. T. N. a. A. A. a. A. C. a. F. Y. a. K. B. a. S. H. a. A. N. a. S. G. C. and, "Acme: A Research Framework for Distributed Reinforcement Learning," *arXiv preprint arXiv:2006.00979,* 2020.

[14] P. Tabor, "Actor Critic Methods, Paper to Code," Github, 2020. [Online]. Available: https://github.com/philtabor/Actor-Critic-Methods-Paper-To-Code.

[15] "XML," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/XML. [Accessed 2022].

[16] "URDF Primer," Mathworks, [Online]. Available: https://www.mathworks.com/help/physmod/sm/ug/urdf-model-import.html. [Accessed 2022].

[17] "Wiki ROS," Ros Noetic, 5 20 2022. [Online]. Available: http://wiki.ros.org/. [Accessed 2022].

[18] "Gazebo Simulator," Wikipedia, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Gazebo_simulator.