دانشگاه صنعتی شریف
دانشکده‌ی مهندسی هوافضا

پروژه کارشناسی ارشد
مهندسی فضا

عنوان:

هدایت یادگیری تقویتی مقاوم مبتنی بر بازی دیفرانسیلی در محیط‌های پویای چندجسمی با پیشران کم

نگارش:

علی بنی اسد

استاد راهنما:

دکتر هادی نوبهاری

شهریور ۱۴۰۴

بسم الله الرحمن الرحیم

به نام خدا

دانشگاه صنعتی شریف

دانشکده‌ی مهندسی هوافضا

پروژه کارشناسی ارشد

عنوان:

هدایت یادگیری تقویتی مقاوم مبتنی بر بازی دیفرانسیلی در محیط‌های پویای چندجسمی با پیشران کم

نگارش: علی بنی اسد

کمیته‌ی ممتحنین

استاد راهنما: دکتر هادی نوبهاری امضاء:

استاد ممتحن: دکتر سیدعلی امامی خوانساری امضاء:

استاد ممتحن: دکتر علیرضا باصحبت نوین زاده امضاء:

تاریخ:

سپاس

از استاد بزرگوارم، جناب آقای دکتر هادی نوبهاری، به پاس زحمات فراوانی که برای این شاگرد کوچکشان کشیدند، صمیمانه سپاسگزارم. در عرصه‌های گوناگون علمی، اخلاقی و انسانی از ایشان بسیار آموختم و بی‌گمان توفیق شاگردی ایشان برای من لطفی الهی بوده است که به آن می‌بالم. بخش چشمگیری از موفقیت‌هایم در دوره‌های کارشناسی و کارشناسی‌ارشد را مرهون راهنمایی‌ها و همراهی‌های این بزرگوار هستم. برای ایشان و خانواده‌ی گرامیشان سلامتی، عزت و سربلندی روزافزون در همه‌ی مراحل زندگی آرزو می‌کنم و امیدوارم بتوانم دینِ شاگردی را به‌خوبی ادا کنم. همچنین، از جناب آقای مهندس میثم علی‌زاد که نظرات ارزشمند او همواره راهگشای مشکلات بنده بود، تشکر می‌کنم. از پدر دلسوزم ممنونم که در انجام این پروژه مرا یاری نمود. در نهایت در کمال تواضع، با تمام وجود بر دستان مادرم بوسه می‌زنم که اگر حمایت بی‌دریغش، نگاه مهربانش و دستان گرمش نبود برگ برگ این دست نوشته و پروژه وجود نداشت.

چکیده

در این پژوهش، یک چارچوب هدایت مقاوم برای فضاپیماهای کم‌پیشران در محیط‌های دینامیکی چندجسمی (سامانه سه جسمی زمین–ماه) ارائه شده‌است. مسئله به‌صورت بازی دیفرانسیلی مجموع‌صفر بین عامل هدایت (فضاپیما) و عامل مزاحم (عدم قطعیت‌های محیطی) فرمول‌بندی شده و با رویکرد آموزش متمرکز و اجرای توزیع‌شده پیاده‌سازی گردیده است. در این راستا، چهار الگوریتم یادگیری تقویتی پیوسته DDPG، TD3، SAC و PPO به نسخه‌های چندعاملی مجموع‌صفر گسترش یافته‌اند (MA-DDPG، MA-TD3، MA-SAC و MA-PPO) و جریان آموزش آن‌ها همراه با ساختار شبکه‌ها در قالب اطلاعات کامل تشریح شده‌است. ارزیابی الگوریتم‌ها در سناریوهای متنوع عدم قطعیت شامل شرایط اولیه تصادفی، اغتشاش عملگر، نویز حسگر، تأخیر زمانی و عدم تطابق مدل روی مسیر مدار لیاپانوف زمین–ماه انجام گرفت. نتایج به‌وضوح نشان می‌دهد که نسخه‌های مجموع‌صفر در تمامی معیارهای ارزیابی بر نسخه‌های تک‌عاملی برتری دارند. به‌ویژه الگوریتم MA-TD3 با حفظ پایداری سیستم، کمترین انحراف مسیر و مصرف سوخت بهینه را حتی در سخت‌ترین سناریوهای آزمون از خود نشان داد. در نهایت، چارچوب پیشنهادی نشان می‌دهد که یادگیری تقویتی چندعاملی مبتنی بر بازی دیفرانسیلی مجموع‌صفر می‌تواند بدون نیاز به مدل‌سازی دقیق، هدایت تطبیقی و مقاوم فضاپیماهای کم‌پیشران را در نواحی ناپایدار سیستم‌های سه‌جسمی تضمین کند.

کلیدواژه‌ها : یادگیری تقویتی عمیق، بازی دیفرانسیلی، سامانه‌های چندعاملی، هدایت کم‌پیشران، بازی مجموع‌صفر، مسئله محدود سه‌جسمی، کنترل مقاوم.

فهرست مطالب

فهرست جداول

فهرست تصاویر

فهرست الگوریتم‌ها

فصل ۱

مقدمه

در سال‌های آغازین عصر فضا، فرایند هدایت فضاپیماها عمدتاً بر مبانی دینامیک کلاسیک و کنترل خطی استوار بوده است. با این حال، پیچیدگی روزافزون مأموریت‌های کنونی مانند سفرهای میان‌سیاره‌ای با پیشران‌کم و شبکه‌های انبوه ماهواره‌ای در مدار زمین موجب دوچندان شدن ضرورت بهره‌گیری از روش‌های هوشمند و تطبیق‌پذیر شده است. در ادامه، انگیزه‌ی پژوهش در بخش ۱-۱ و تعریف دقیق مسئله در بخش ۱-۲ آمده است. سپس، مروری کوتاه بر مبانی یادگیری تقویتی و نسخه‌ی چندعاملی آن در بخش‌های ۱-۳ و ۱-۴ ارائه شده و در نهایت، ساختار کل گزارش در بخش ۱-۵ تشریح شده است.

۱-۱ انگیزه پژوهش

در دو دهه‌ی اخیر، به‌دلیل کوچک‌سازی سامانه‌ها، توسعه‌ی الکترونیک مقرون‌به‌صرفه و افزایش ظرفیت‌های پرتاب، تحولات بنیادینی در مأموریت‌های فضایی تجربه شده‌است. از پروژه‌های علمیِ بین‌سیاره‌ای تا منظومه‌های انبوه ماهواره‌ای در مدارهای پایین زمین، مواجهه با چالشِ فراگیرِ هدایتِ بهینه در حضور عدم‌قطعیت‌ها به‌طور گسترده گزارش شده‌است. در مسیرهای فرا-قمری[۱] و به‌طور خاص در ناحیه‌های ناپایدارِ نقاط لاگرانژ در چارچوب مسئله‌ی سه‌جسمیِ کرویِ محدودِ دایروی[۲]، طراحی سامانه‌ی کنترل مستلزم تضمین هم‌زمانِ پایداریِ ایستا و بهره‌وریِ سوخت با پیشران‌کم[۳] است.

هم‌راستا با این تحولات، ظهور و گسترشِ الگوریتم‌های یادگیری تقویتیِ عمیق[۴] امکانات نوینی برای طراحیِ

[1] Trans-lunar
[2] Circular Restricted Three-Body Problem (CRTBP)
[3] Low-thrust
[4] Deep Reinforcement Learning (DRL)

کنترل‌کننده‌های تطبیقی فراهم آورده است؛ با این حال، غالبِ رویکردهای رایج بر سناریوهای تک‌عاملی و اتکا به مدل‌های دینامیکیِ دقیق استوار شده‌اند. غیابِ یک راهبردِ مقاوم در برابر اغتشاشاتِ مدل و تغییرات محیطی—از جمله خطای تراستِ پیشران و تأخیر حسگر—به ایجادِ فاصله‌ی معنادار میان عملکرد واقعی و پیش‌بینی‌های شبیه‌سازیِ ایده‌آل منجر شده‌است. در این پژوهش، این شکاف با بهره‌گیری از چارچوبِ یادگیریِ تقویتیِ چندعاملیِ مقاوم پُر می‌شود و اطمینانِ هدایتِ پیشران‌کم در CRTBP ارتقا داده می‌شود. در ادامه، تعریف دقیق مسئله و سپس اهداف و نوآوری‌های پژوهش ارائه می‌شود.

۱-۲ تعریف مسئله

در سال‌های اخیر، پیشرفت‌های فناوری در کنترل پرواز، پردازش و هوش مصنوعی به گسترشِ کاربردِ فضاپیماهای پیشران‌کم در منظومه‌ی زمین–ماه انجامیده است؛ از تعقیب و انتقال مداری تا استقرار و نگه‌داری. روش‌های هدایتِ بهینه‌ی کلاسیک، هرچند قدرتمند، عموماً به ساده‌سازی‌های بسیار، منابعِ محاسباتیِ زیاد و شرایط اولیه‌ی مناسب متکی بوده‌اند؛ در مقابل، بخشی از این محدودیت‌ها با الگوریتم‌های مبتنی بر یادگیری تقویتی و تکیه بر تعامل و امکان محاسباتِ درون‌برد[5] برطرف می‌شود.

هدف، طراحیِ سیاستِ کنترلی برای فضاپیمایی با جرم m در میدانِ گرانشِ سامانه‌ی زمین–ماه (مدل دوبعدی در چارچوبِ چرخان) است. ویژگی‌ها به‌اختصار:

- **پویایی‌ها:** معادلاتِ حرکت در چارچوب مرجعِ چرخان به‌صورت $\dot{\boldsymbol{x}} = f(\boldsymbol{x}) + g(\boldsymbol{x})\,\boldsymbol{a}$ با $\boldsymbol{x} = [x,\, y,\, \dot{x},\, \dot{y}]^\top$ و کنترلِ پیوسته‌ی $\boldsymbol{a} \in \mathcal{A}$ تعریف می‌شود، به‌طوری‌که کران $|\boldsymbol{a}| \leqslant a_{\max}$ برقرار است.
- **عدم‌قطعیت‌ها:** شرایطِ اولیه‌ی تصادفی، اغتشاش‌های عملگر، عدم‌تطابقِ مدل (در پارامترهای جرم)، مشاهده‌ی ناقص، نویز حسگر و تأخیر زمانی، که بر پایداری و کارایی اثرگذارند.
- **صورت‌بندیِ بازیِ دیفرانسیلی (جمع‌صفر):** فضاپیما و طبیعت (اغتشاشات) به‌ترتیب به‌عنوان عاملِ کنترل و حریفِ مزاحم مدل می‌شوند؛ با افق زمانیِ محدود t_f، هدف، دستیابی به سیاستی مقاوم در برابر بدترین سناریو است.

صورتِ فشرده‌ی بهینه‌سازی به‌صورت کمینه–بیشینه است:

$$\min_{\pi}\ \max_{\omega}\ \mathbb{E}_{p,\pi,\omega}\left[\ \sum_{t=0}^{T} r\big(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{\delta}_t\big)\ \right], \tag{۱-۱}$$

[5] On-board Computing

که در آن، پاداشِ r به‌عنوان تابعی از مصرفِ سوخت، انحراف از مسیر یا مدار نامی و قیودِ مسئله تعریف می‌شود. خروجیِ مورد انتظار، سیاستی سبک و غیرمتمرکز برای اجرای درون‌برد مدنظر است.

۱–۳ یادگیری تقویتی

یادگیری تقویتی[۶] شاخه‌ای از یادگیریِ ماشین است که در آن توالیِ اقدام‌ها $\boldsymbol{a}_t \in \mathcal{A}$ به‌گونه‌ای انتخاب می‌شود که بازدهِ تجمعیِ آینده بیشینه شود. یک فرایندِ تصمیم‌گیریِ مارکوف[۷] به‌صورت $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$ تعریف می‌شود که در آن:

- $\mathcal{S}$: مجموعه‌ی حالات،
- $p(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a})$: دینامیکِ انتقال،
- $r(\boldsymbol{s}, \boldsymbol{a})$: پاداشِ آنی،
- $\gamma \in [0, 1)$: ضریبِ تنزیل.

سیاست[۸] $\pi(\boldsymbol{a}|\boldsymbol{s})$ به‌عنوان احتمالِ انتخابِ اقدامِ $\boldsymbol{a}$ در وضعیتِ $\boldsymbol{s}$ بیان می‌شود. هدف، بیشینه‌سازیِ برگشت[۹] است:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{۱–۲}$$

روش‌های RL معمولاً در دو دسته‌ی ارزش‌محور (مانند Q-learning و DQN) و سیاست‌محور (مانند Reinforce) جای می‌گیرند؛ ترکیبِ این دو به چارچوبِ Actor–Critic منتهی می‌شود که در آن، یک بازیگر (Actor) سیاست را به‌روزرسانی می‌کند و یک منتقد (Critic) ارزش یا Q برآورد می‌شود [۱].

در حضورِ فضاهای پیوسته‌ی حالت–عمل، الگوریتم‌های DDPG، TD3، SAC و PPO با تکیه بر شبکه‌های عصبی به‌عنوان تقریب‌گر توابع، کاراییِ بالایی نشان داده‌اند. در این پژوهش، خانواده‌ی Actor–Critic به‌عنوان پایه‌ی توسعه‌ی کنترل‌کننده پیشنهاد شده است و در ادامه، به نسخه‌ی چندعاملیِ آن در بخش ۱–۴ پیوند داده می‌شود.

[6] Reinforcement Learning (RL)
[7] Markov Decision Process (MDP)
[8] Policy
[9] Return

۱-۴ یادگیری تقویتی چندعاملی

در یادگیری تقویتی چندعاملی[۱۰]، فضای تصمیم‌گیری به‌صورت یک بازی مارکفی[۱۱] با مجموعه‌ی عامل‌ها $\mathcal{N} = \{1, \ldots, N\}$ مدل شده که در آن هر عامل با سیاست π_i به‌دنبال بیشینه‌سازی بازده تجمعی خود است. در سناریوهای رقابتیِ دونفره‌ی جمع‌صفر[۱۲]، مفهوم تعادل نش[۱۳] به‌عنوان معیار پایداری سیاست‌ها در نظر گرفته می‌شود.

رویکرد آموزشِ متمرکز، اجرایِ توزیع‌شده[۱۴] با جدا کردن مرحله‌ی آموزش که در آن اطلاعات خصوصیِ همه‌ی عامل‌ها برای منتقدها در دسترس است و مرحله‌ی اجرا در آن هر عامل صرفاً بر مشاهده‌ی محلی اتکا می‌کند که باعث تعادل میان کارایی، مقیاس‌پذیری و هزینه‌ی ارتباطی برقرار شده‌است.

در این پایان‌نامه، یک صورت‌بندیِ دوعاملیِ جمع‌صفر اتخاذ شده‌است که در آن سیاست هدایت توسط عاملِ کنترل آموخته می‌شود و اغتشاشات یا نامعینی‌ها توسط عاملِ مزاحم مدل‌سازی می‌شوند تا سیاستی مقاوم حاصل شود.

- DDPG: الگوریتم مبتنی بر گرادیان سیاستِ قطعی برای فضاهای کنشِ پیوسته،
- TD3: نسخه‌ی بهبودیافته‌ی **DDPG** با برآورد دوسویه‌ی Q برای کاهش تورشِ بیش‌براورد[۱۵]،
- PPO: الگوریتم سیاست احتمالیِ پایدار با قیود نسبت احتمال و بهبود تدریجی سیاست،
- SAC: الگوریتم حداکثرسازی آنتروپی که تعادل میان بهره‌برداری و اکتشاف به‌طور ذاتی برقرار می‌شود.

تابع پاداشِ طراحی‌شده، مصالحه‌ی سوخت، انحراف و قیود منعکس می‌شود و مبنایی برای ارزیابیِ کیفیتِ سیاست‌های مختلف فراهم می‌گردد.

۱-۵ ساختار گزارش

در فصل ۲ مروری انتقادی بر کارهای مرتبط در هدایت پیشران‌کم و یادگیری تقویتی تک‌عاملی و چندعاملی ارائه می‌شود. در فصل ۳ مدل‌سازی محیط آزمایش بر پایه‌ی CRTBP ارائه می‌گردد. فصل ۴ به مبانی

[10] Multi-Agent Reinforcement Learning (MARL)
[11] Markov Games (MG)
[12] Zero-Sum
[13] Nash Equilibrium
[14] Centralized Training with Decentralized Execution (CTDE)
[15] Overestimation Bias

یادگیری تقویتی اختصاص دارد و الگوریتم‌های DDPG، TD3، SAC و PPO در بخش‌های ۴-۲ تا ۴-۵ مرور می‌شوند. در فصل ۵ طراحی شبیه‌سازی شامل تعریف عامل‌ها، فضای حالت، عمل و تابع پاداش توضیح داده می‌شود. در فصل ۶ چارچوب یادگیری تقویتی چندعاملی تشریح شده و پیوند آن با بازی‌های جمع‌صفر و تعادل نش در بخش ۶-۲ بیان می‌شود. در فصل ۷ نتایج و مقایسه با معیارهای مرجع ارائه می‌شود. در نهایت، فصل ۸ جمع‌بندی دستاوردها و پیشنهادهای پژوهش‌های آینده را ارائه می‌دهد.

فصل ۲

پیشینه پژوهش

این فصل تصویری منسجم از ادبیات مأموریت‌های بین‌مداری، مبانی یادگیری تقویتی و یادگیری تقویتی چندعاملی ارائه می‌کند. تمرکز بر تبیین مفاهیم کلیدی، چالش‌های رایج، و روندهای پژوهشی مؤثر برای طراحی و هدایت درون‌سفینه‌ای است؛ به‌گونه‌ای که زمینه‌ی نظری لازم برای روش‌ها و مسائل مورد استفاده در ادامه‌ی پژوهش فراهم شود. ساختار فصل به این صورت است که ابتدا در بخش ۲-۱ مروری بر روش‌های طراحی مسیر و هدایت درون‌سفینه‌ای و کاربردهای یادگیری ماشین ارائه می‌شود، سپس در بخش ۲-۲ مبانی و الگوریتم‌های اصلی یادگیری تقویتی و خطوط پژوهشی مرتبط مرور می‌گردد و در پایان بخش ۲-۳ به رویکردها و چالش‌های یادگیری تقویتی چندعاملی پرداخته می‌شود.

۲-۱ ماموریت‌های بین‌مداری

هدایت فضاپیماها معمولاً با استفاده از ایستگاه‌های زمینی انجام می‌شود. با این حال، این تکنیک‌ها دارای محدودیت‌هایی از جمله حساسیت به قطع ارتباطات، تأخیرهای زمانی و محدودیت‌های منابع محاسباتی هستند. الگوریتم‌های یادگیری تقویتی و بازی‌های دیفرانسیلی می‌توانند برای بهبود قابلیت‌های هدایت فضاپیماها، از جمله مقاومت در برابر تغییرات محیطی، کاهش تأخیرهای ناشی از ارتباطات زمینی و افزایش کارایی محاسباتی، مورد استفاده قرار گیرند.

هدایت فضاپیماها معمولاً پیش از پرواز انجام می‌شود. این روش‌ها می‌توانند از تکنیک‌های بهینه‌سازی فراگیر [۲] یا برنامه‌نویسی غیرخطی برای تولید مسیرها و فرمان‌های کنترلی بهینه استفاده کنند. با این حال، این روش‌ها معمولاً حجم محاسباتی زیادی دارند و برای استفاده درون‌سفینه‌ای نامناسب هستند [۳]. یادگیری ماشین می‌تواند برای بهبود قابلیت‌های هدایت فضاپیماها استفاده شود. کنترل‌کننده شبکه عصبی حلقه‌بسته

می‌تواند برای محاسبه سریع و خودکار تاریخچه کنترل استفاده شود. یادگیری تقویتی نیز می‌تواند برای یادگیری رفتارهای هدایت بهینه استفاده شود.

روش‌های هدایت و بهینه‌سازی مسیر فضاپیماها به‌طورکلی به راه‌حل‌های اولیه مناسب نیاز دارند. در مسائل چند جسمی، طراحان مسیر اغلب حدس‌های اولیه کم‌هزینه‌ای برای انتقال‌ها با استفاده از نظریه سیستم‌های دینامیکی و منیفولدهای ثابت [۴،۵] ایجاد می‌کنند.

شبکه‌های عصبی قابلیت‌های منحصر به فردی برای انجام هدایت در فضاپیما دارند. به‌عنوان مثال، شبکه‌های عصبی می‌توانند به‌طور مستقیم از تخمین‌های وضعیت به دستورهای پیشران کنترلی که با محدودیت‌های مأموریت سازگار است، برسند. عملکرد مناسب هدایت شبکه‌های عصبی در مطالعاتی مانند فرود بر سیارات [۶]، عملیات نزدیکی به سیارات [۷] و کنترل فضاپیما با پیشران از دست‌رفته [۸] نشان داده شده‌است. تازه‌ترین پیشرفت‌های تکنیک‌های یادگیری ماشین در مسائل خودکارسازی درونی به‌طور گسترده‌ای مورد مطالعه قرار گرفته‌اند؛ از پژوهش‌های اولیه تا توانایی‌های پیاده‌سازی. به‌عنوان مثال، الگوریتم‌های یادگیری ماشین ابتدایی در فضاپیماهای مریخی‌نورد برای کمک به شناسایی ویژگی‌های زمین‌شناسی تعبیه شده‌اند. الگوریتم AEGIS توانایی انتخاب خودکار هدف توسط یک دوربین در داخل فضاپیماهای Spirit، Opportunity و Curiosity را دارد [۹]. در کامپیوتر پرواز اصلی، فرآیند دقت افزایی[1] نیاز به ۹۴ تا ۹۶ ثانیه دارد [۱۰]، که به طور قابل توجهی کمتر از زمان مورد نیاز برای ارسال تصاویر به زمین و انتظار برای انتخاب دستی توسط دانشمندان است. برنامه‌های آینده برای کاربردهای یادگیری ماشین درون‌سفینه شامل توانایی‌های رباتیکی درون‌سفینه برای فضاپیمای Perseverance [۱۱،۱۲] و شناسایی عیب برای Europa Clipper [۱۳] می‌شود. الگوریتم‌های یادگیری ماشین پتانسیل انجام نقش مهمی در مأموریت‌های خودکار آینده را دارند.

علاوه بر رباتیک سیاره‌ای، پژوهش‌های مختلفی به استفاده از تکنیک‌های مختلف یادگیری ماشین در مسائل نجومی پرداخته‌اند. در طراحی مسیر عملکرد رگرسیون معمولاً مؤثرتر هست. به عنوان مثال، از یک شبکه عصبی[2] در بهینه‌سازی مسیرهای رانشگر کم‌پیشران استفاده شده‌است [۱۴]. پژوهش‌های جدید شامل شناسایی انتقال‌های هتروکلینیک [۱۵]، اصلاح مسیر رانشگر کم‌پیشران [۱۶] و تجزیه و تحلیل مشکلات ازدست‌رفتن رانشگر [۸] می‌شود.

تکنیک‌های یادگیری نظارتی می‌توانند نتایج مطلوبی تولید کنند؛ اما، دارای محدودیت‌های قابل توجهی هستند. یکی از این محدودیت‌ها این است که این رویکردها بر وجود دانش پیش از فرآیند تصمیم‌گیری متکی هستند. این امر مستلزم دقیق‌بودن داده‌های تولیدشده توسط کاربر برای نتایج مطلوب و همچنین وجود تکنیک‌های موجود برای حل مشکل کنونی و تولید داده است.

[1] Refinement Process
[2] Neural Network

در سال‌های اخیر، قابلیت یادگیری تقویتی[۳] در دستیابی به عملکرد بهینه در بخش‌هایی با ابهام محیطی قابل توجه، به اثبات رسیده است [۱۷، ۱۸]. هدایت انجام‌شده توسط یادگیری تقویتی را می‌توان به‌صورت گسترده بر اساس فاز پرواز دسته‌بندی کرد. مسائل فرود [۱۹، ۲۰] و عملیات در نزدیکی اجسام کوچک [۶،۷]، از حوزه‌های پژوهشی هستند که از یادگیری تقویتی استفاده می‌کنند. تحقیقات دیگر شامل مواجهه تداخل خارجی جوی [۲۱]، نگهداری ایستگاهی [۲۲] و هدایت به‌صورت جلوگیری از شناسایی [۲۳] است. مطالعاتی که فضاپیماهای رانشگر کم‌پیشران را در یک چارچوب دینامیکی چندبدنی با استفاده از یادگیری تقویتی انجام شده‌است، شامل طراحی انتقال با استفاده از Q-learning [۲۴]، Proximal Policy Optimization [۲۵] و هدایت نزدیکی مدار [۲۶] است.

۲-۲ یادگیری تقویتی

از نخستین صورت‌بندی‌های فرایند تصمیم‌گیری مارکُفی در یادگیری تقویتی، پژوهش بر آن بوده‌است که عامل بتواند با اجرای عمل‌ها و دریافت پاداش، سیاستی برای بیشینه‌سازی بازگشت بیاموزد. تبیین جامع این چارچوب و الگوریتم‌های بنیادین در کتاب سوتون و بارتو به‌مثابه مرجع کلاسیک این حوزه ارائه شده و همچنان مبنای بسیاری از آثار معاصر است [۱].

دهه‌ی ۱۹۹۰ میلادی شاهد شکل‌گیری روش‌هایی بر پایه‌ی ارزش[۴] نظیر Q-learning و نخستین رویکردهای گرادیانِ سیاست بود؛ با وجود این، محدودیت توان محاسباتی و فقدان داده‌ی فراوان، سرعت رشد را کند می‌کرد. ورود شبکه‌های عصبی عمیق[۵] نقطه‌ی عطفی بود: مقاله‌ی معروف دیپ‌مایند[۶] نشان داد که شبکه‌ی Q عمیق[۷] می‌تواند صرفاً از پیکسل‌های بازی آتاری سیاستی نزدیک به انسان بیاموزد [۲۷].

موفقیت DQN نگاه‌ها را به‌سوی گرادیانِ سیاستِ مقیاس‌پذیر معطوف ساخت. بهینه‌سازی ناحیه‌ی اطمینان[۸] تضمین بهبود یکنواخت سیاست را فراهم کرد [۲۸] و روش A3C با موازی‌سازی بازیگران، سرعت یادگیری را چند برابر افزایش داد [۲۹]. کمی بعد، DDPG اولین بار گرادیان سیاست قطعی را به فضاهای عمل پیوسته وارد کرد [۳۰]. سپس PPO با ساده‌سازی قیود TRPO و کاهش فراپارامترهای حساس، به انتخاب پیش‌فرض بسیاری از کاربردهای مهندسی بدل شد [۳۱].

با گسترش دامنه‌ی مسائل، پایداری و کاراییِ داده به چالش اصلی بدل گشت. TD3 نشان داد که کمینه‌کردن

[3] Reinforcement Learning (RL)
[4] Value
[5] Deep Neural Network (DNN)
[6] DeepMind
[7] Deep Q Network (DQN)
[8] Trust Region Policy Optimization (TRPO)

میان دو منتقد می‌تواند برآورد بیش‌ازحد Q را مهار کند [۳۲]، و SAC با افزودن بند آنتروپی، همزمان اکتشاف و بازده را بهبود داد [۳۳].

در محیط‌های پرخطر یا گران، جمع‌آوری داده‌ی برخط ناممکن است؛ ازاین‌رو یادگیری تقویتی غیربرخط مطرح شد. روش CQL با برقراری کران محافظه‌کارانه بر Q-value از گرایشِ خارج از توزیع جلوگیری می‌کند [۳۴] و مرور اخیر پراودِنسیو و همکاران طبقه‌بندی جامعی از چالش‌های باز این حوزه ارائه داده است [۳۵].

همزمان، دغدغه‌ی ایمنی و مقاومت در سامانه‌های واقعی پررنگ شد. مرور سال ۲۰۲۲ نشان می‌دهد که ترکیب قیدهای سخت، توابع جریمه‌ی ریسک و شبیه‌سازی محیط‌های بدبینانه سه خط اصلی ایمنی دریادگیری تقویتی هستند [۳۶]. سلسله‌مراتب نیز با هدف انتقال دانش و تسریع یادگیری مورد توجه قرار گرفت و یک مطالعه‌ی جامع در ACM Computing Surveys چهار چالش کشف زیرکار، یادگیری اشتراک‌پذیر، انتقال و مقیاس‌پذیری را برجسته می‌کند [۳۷].

وقتی چند عامل به‌طور همزمان یاد می‌گیرند، پویایی محیط از دید هر عامل غیرایستا می‌شود. مرور جامع ۲۰۲۴ نشان می‌دهد که چارچوب ناظر متمرکز ـ بازیگر توزیع‌شده[۹] راهکاری موثر برای این چالش است و مباحثی چون تخصیص اعتبار جمعی و کشف تعادل را معرفی می‌کند [۳۸].

پیشرفت‌های یادشده در نهایت به دستاوردهای نمادینی چون AlphaGo [۳۹] و AlphaStar [۴۰] انجامیدند که در بازی‌های Go و StarCraft II از انسان پیشی گرفتند و معماری توزیع‌شده‌ی IMPALA نشان داد که چگونه می‌توان هزاران شبیه‌ساز را با به‌روزرسانی وزن‌های مهم ادغام کرد [۴۱].

به‌رغم این جهش‌ها، سه شکاف اساسی پابرجا مانده است: ۱) تضمین ایمنی سخت‌گیرانه در سناریوهای نزدیک‌برخورد، ۲) کاهش وابستگی به داده‌ی پرهزینه یا نایاب از طریق روش‌های مدل‌مبنا و غیربرخط و ۳) مقیاس‌پذیری یادگیری چندعاملی برای سامانه‌های رباتیکی یا فضاپیمای چندگانه.

۲-۳ پیشینه‌ی پژوهش یادگیری تقویتی چندعاملی

امروز یادگیری تقویتی چندعاملی[۱۰] به‌عنوان بنیاد اصلی سامانه‌های هوشمند مشارکتی شناخته می‌شود؛ مسیری که از آزمون‌های ساده‌ی دوعاملی در دهه‌ی ۱۹۹۰ آغاز شد و اکنون به معماری‌های توزیع‌شده‌ی در مقیاس هزاران بازیگر رسیده است. این بخش، به بررسی اینکه چگونه ایده‌ی آموزش متمرکز ـ اجرای توزیع‌شده (CTDE) به پاسخ غالب برای چالش‌های غیرایستایی و انفجار بُعدی[۱۱] بدل شد و چه گام‌هایی هنوز برای ایمنی، ناهمگونی

[۹] Centralized Training with Decentralized Execution (CTDE)
[۱۰] Multi-Agent Reinforcement Learning (MARL)
[۱۱] Curse of Dimensionality

و مقیاس‌پذیری باقی مانده است.

دهه‌ی ۱۹۹۰ با مقاله‌ی [۴۲] آغاز شد؛ جایی که برای نخستین‌بار مقایسه‌ی عامل‌های مستقل با عامل‌های همکار انجام شد و سود ارتباط و اشتراک تجربه به‌صورت تجربی نشان داده شد. در میانه‌ی دهه‌ی بعد، مرور جامع پانایت و لوک [۴۳] چشم‌اندازی از مسائل تخصیص اعتبار و غیرایستایی ترسیم کرد و دو موضوع یادگیری تیمی و یادگیری هم‌زمان را صورت‌بندی نمود. هم‌زمان، بوشونیو و همکاران [۴۴] ادبیات MARL را در قالب اهداف پایداری دینامیک یادگیری و انطباق با رفتار سایر عامل‌ها جمع‌بندی کردند و راه را برای تحلیل‌های بازی‌محور هموار ساختند.

ورود شبکه‌های عمیق در سال‌های ۲۰۱۶ و ۲۰۱۷ نقطه‌ی عطف بعدی بود؛ منتقد متمرکز- بازیگر توزیع‌شده در MA-DDPG [۴۵] نشان داد که می‌توان از حالت سراسری در فاز آموزش بهره برد، اما سیاست نهایی را صرفاً بر اساس مشاهدات محلی اجرا کرد. در همان سال، Value-Decomposition Networks [۴۶] ایده‌ی تجزیه‌ی خطی پاداش را برای همکاری عامل‌ها مطرح کرد و راه را برای تقسیم بندی‌های پیش‌رفته پاداش گشود.

۲۰۱۸ شاهد جهش مهمی با QMIX بود؛ این روش با اعمال قید تک‌نوا[۱۲] بر ترکیب مقادیر منفرد، هم امکان بهینه‌سازی غیرسیاست‌محور را فراهم کرد و هم تضمین سازگاری سیاست‌های محلی با ارزش مشترک را برقرار ساخت [۴۷].

سال ۲۰۱۹ به گسترش بسترهای آزمایش اختصاص یافت. چالش استاندارد StarCraft Multi-Agent Challenge (SMAC) بر مبنای StarCraft II معرفی شد و معیار مشترکی برای مقایسه‌ی الگوریتم‌ها را مهیا کرد [۴۸]. هم‌زمان، QTRAN [۴۹] نشان داد که می‌توان بدون قید خطی یا تک‌نوا، تابع ارزش مشترک را به فضای قابل تجزیه تبدیل کرد. از سوی دیگر، MAVEN با افزودن متغیر نهفته‌ی مشترک، کاوش هماهنگ و سلسله‌مراتبی را امکان‌پذیر ساخت [۵۰]. نقطه‌ی اوج همان سال، سامانه‌ی AlphaStar بود که نشان داد ترکیب خودبازی و معماری توزیع‌شده می‌تواند به رتبه‌ی استاد بزرگ[۱۳] انسان برساند [۴۰].

در ۲۰۲۰ مفهوم نقش‌های در حال ظهور با ROMA [۵۱] معرفی شد تا عامل‌ها بر اساس شباهت رفتاری به‌طور خودکار خوشه‌بندی و اشتراک دانش کنند؛ رویکردی که در نقشه‌های پرتراکم SMAC برتری محسوسی نشان داد. پژوهش‌های متا در ۲۰۲۱، از مرور نظری زانگ و بشار [۵۲] تا محک[۱۴] تطبیقی پاپوداکیس و همکاران [۵۳]، شکاف‌های باقی‌مانده در تضمین همگرایی و مقیاس را فهرست کردند.

آخرین موج مطالعات بر ناهمگونی و ایمنی تمرکز دارد. مرور جامع [۵۴] نشان می‌دهد که تفاوت در قابلیت‌ها و اطلاعات عامل‌ها، مسائلی نظیر تخصیص اعتبار و تعادل را پیچیده‌تر می‌سازد و به الگوریتم‌های سازگار با نقش‌های پویا نیاز دارد.

[12] Monotonic
[13] Grandmaster
[14] Benchmark

به‌طور خلاصه، مسیر تاریخی MARL از الگوهای مستقل دهه‌ی ۱۹۹۰ به سامانه‌های توزیع‌شده‌ی امروزی، همواره با سه دغدغه‌ی اصلی هدایت شده‌است: کنترل انفجار بُعدی توابع ارزش، مقابله با غیرایستایی ناشی از یادگیری هم‌زمان، و انتقال مؤثر تجربه میان عامل‌ها. علی‌رغم پیشرفت‌های شتابان، تضمین ایمنی سخت‌گیرانه در محیط‌های شکست‌پذیر، مدیریت نقش‌های پویا در تیم‌های ناهمگون و کاهش نیاز به داده‌ی شبیه‌سازی پرهزینه همچنان چالش‌های باز باقی می‌مانند؛ چالش‌هایی که در این پژوهش با رویکرد ترکیبی مدل‌مبنا، مقاوم و چندعاملی پیگیری می‌شوند.

فصل ۳

مدل‌سازی محیط یادگیری سه جسمی

مسیرهای فضایی در بسیاری از مأموریت‌ها نه‌تنها تحت تأثیر گرانش یک جسم مرکزی (مانند خورشید یا زمین)، بلکه به‌طور هم‌زمان تحت نفوذ دست‌کم یک جسم دیگر نیز هستند. در این وضعیت، مدل‌های دوجسمی با اختلالات جسم سوم دقت کافی ندارند و باید دینامیک دو جسمِ اصلی و اثرات آن‌ها به‌صورت هم‌زمان در نظر گرفته شود. مسئله‌ی سه‌جسمیِ محدود با دو جسم اصلی و یک جسم سوم با جرم ناچیز (فضاپیما) چارچوبی طبیعی برای مطالعه‌ی این پدیده‌ها و نیز یک محیط مناسب برای به‌کارگیری روش‌های یادگیری تقویتی است؛ زیرا دینامیک غیرخطی و پیچیده‌ی آن ویژگی‌های غنی (مانند نقاط تعادلِ لاگرانژ) ایجاد می‌کند.

در این فصل ابتدا در بخش ۳-۱ دستگاهِ بی‌بُعد و چارچوبِ چرخان تعریف شده و در بخش ۳-۲ معادلاتِ حرکت در مسئله‌ی سه‌جسمیِ محدودِ دایره‌ای استخراج می‌شود.

۳-۱ مسئله‌ی سه‌جسمیِ محدودِ دایره‌ای (CRTBP)

دو جرمِ اصلی (زمین با جرم m_1 و ماه با جرم m_2) روی مدارهایی دایره‌ای و هم‌صفحه پیرامونِ مرکزِ جرمِ مشترک حرکت می‌کنند. جرمِ سوم (فضاپیما با جرمِ ناچیز m_3) چنان کوچک فرض می‌شود که تأثیرِ گرانشیِ آن بر حرکتِ دو جسمِ اصلی قابلِ صرفِ نظر است؛ بدین ترتیب، مسئله‌ی سه‌جسمیِ محدودِ دایره‌ای شکل می‌گیرد.

جدول ۳-۱: مقادیر عددی برای مسئله سه‌جسمی محدود (سامانه زمین–ماه)

پارامتر	توصیف	مقدار عددی
m_1	جرم زمین	$5.972 \times 10^{24}\,\mathrm{kg}$
m_2	جرم ماه	$7.348 \times 10^{22}\,\mathrm{kg}$
μ	نسبت جرمی	0.0121505856
ω	سرعت زاویه‌ای سامانه	$2.6617 \times 10^{-6}\,\mathrm{rad/s}$

دستگاهِ مختصاتِ چرخانی هم‌دوران با دو جرم اصلی انتخاب می‌شود؛ مبدأ در مرکزِ جرمِ سامانه است، محور x خطِ واصلِ دو جرم و محور y بر آن عمود (در صفحه‌ی مدارها) است. واحدِ طول برابر فاصله‌ی ثابتِ میان دو جرم و واحدِ زمان چنان تعریف می‌شود که دوره‌ی مداریِ سامانه 2π (و در نتیجه $\omega = 1$) گردد. همچنین جرم‌ها به‌گونه‌ای مقیاس می‌شود که مجموع دو جرم برابر با یک شود:

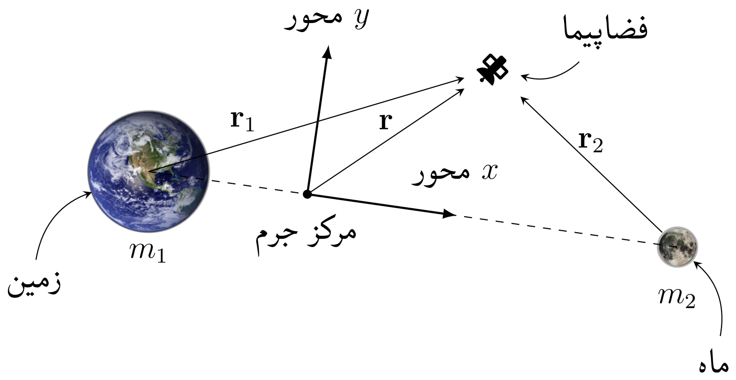
$$m_1 + m_2 = 1. \tag{۳-۱}$$

با نسبتِ جرمی

$$\mu \equiv \frac{m_2}{m_1 + m_2}, \tag{۳-۲}$$

داریم $m_1 = 1 - \mu$ و $m_2 = \mu$ و مکانِ دو جرم در دستگاهِ بی‌بُعد به صورت

$$\mathbf{r}_{\mathrm{Earth}} = (-\mu, 0), \qquad \mathbf{r}_{\mathrm{Moon}} = (1 - \mu, 0). \tag{۳-۳}$$



شکل ۳-۱: هندسه‌ی مسئله‌ی سه‌جسمیِ محدود در چارچوبِ چرخان

۳-۱-۱ لاگرانژ و معادلات حرکت

با در نظر گرفتن $G = 1$ در حالت بی‌بُعد، تابعِ لاگرانژِ جرمِ سوم در دستگاهِ چرخان برابر است با [۵۵]

$$L = \tfrac{1}{2}\big(\dot{x}^2 + \dot{y}^2 + \dot{z}^2\big) + (1-\mu)\,\frac{1}{r_1} + \mu\,\frac{1}{r_2} + \tfrac{1}{2}\big(x^2 + y^2\big), \tag{۳-۴}$$

که در آن

$$r_1 = \sqrt{(x+\mu)^2 + y^2 + z^2}, \qquad r_2 = \sqrt{\big(x-1+\mu\big)^2 + y^2 + z^2}. \tag{۳-۵}$$

با به‌کارگیری رابطه‌ی اویلر–لاگرانژ

$$\frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = 0, \qquad q_i \in \{x, y, z\},$$

معادلاتِ بی‌بُعدِ حرکتِ جرمِ سوم به دست می‌آید:

$$\ddot{x} - 2\dot{y} = x - \frac{1-\mu}{r_1^3}(x+\mu) - \frac{\mu}{r_2^3}\big(x-1+\mu\big), \tag{۳-۶}$$

$$\ddot{y} + 2\dot{x} = y - \frac{1-\mu}{r_1^3}\,y - \frac{\mu}{r_2^3}\,y, \tag{۳-۷}$$

$$\ddot{z} = -\frac{1-\mu}{r_1^3}\,z - \frac{\mu}{r_2^3}\,z. \tag{۳-۸}$$

یا به نگاشتِ برداری به‌صورت زیر است.

$$\ddot{\mathbf{r}} + 2\,\boldsymbol{\omega} \times \dot{\mathbf{r}} = \nabla\Omega(\mathbf{r}), \qquad \Omega(x,y,z) = \tfrac{1}{2}\big(x^2 + y^2\big) + \frac{1-\mu}{r_1} + \frac{\mu}{r_2}. \tag{۳-۹}$$

که در آن Ω پتانسیلِ مؤثر است و در بخش ۳-۲ برای یافتنِ نقاطِ تعادل از شرطِ $\nabla\Omega = 0$ استفاده می‌شود.

۳-۲ نقاط تعادلِ لاگرانژ

نقطه‌ی تعادل مکانی است که در چارچوبِ چرخان، جرمِ سوم بی‌حرکت می‌ماند. این شرط با صفر شدن مؤلفه‌های سرعت و شتاب حاصل می‌شود؛ ازاین‌رو در معادلاتِ حرکتِ بخش ۳-۱ قرار می‌دهیم $\dot{x} = \dot{y} = \dot{z} = \ddot{x} = \ddot{y} = \ddot{z} = 0$. در نتیجه دستگاهِ جبریِ زیر برای مختصاتِ نقطه‌ی تعادل به‌دست می‌آید:

$$0 = x - \frac{1-\mu}{r_1^3}(x+\mu) - \frac{\mu}{r_2^3}\big(x-1+\mu\big), \tag{۳-۱۰}$$

$$0 = y\Big[1 - \frac{1-\mu}{r_1^3} - \frac{\mu}{r_2^3}\Big], \tag{۳-۱۱}$$

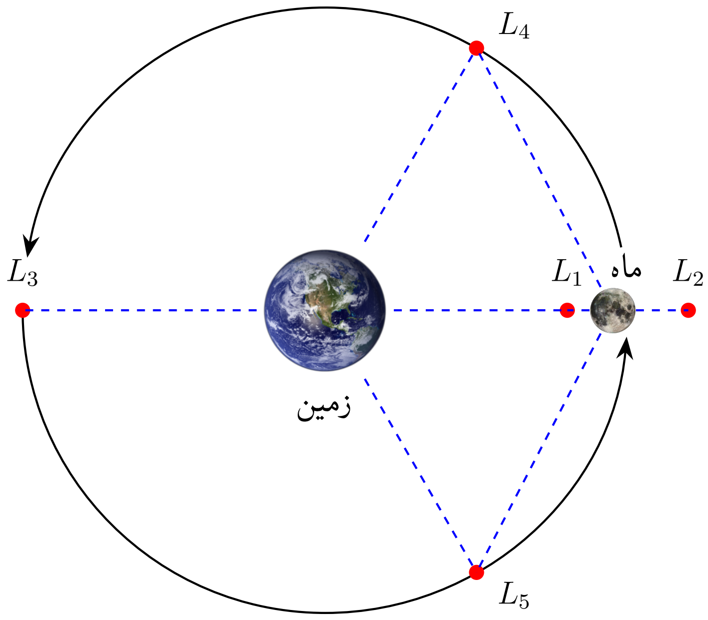
$$0 = -\frac{1-\mu}{r_1^3}\,z - \frac{\mu}{r_2^3}\,z. \tag{۳-۱۲}$$

معادله‌ی سوم نشان می‌دهد در حالتِ عمومی باید $z = 0$ باشد؛ بنابراین، نقاطِ تعادل همگی در صفحه‌ی مدار قرار می‌گیرند.

دسته‌بندی کلی

۱. نقاطِ هم‌خط[۱]: سه نقطه‌ی L_1، L_2 و L_3 روی خطِ واصلِ دو جرم قرار دارند و لذا $y = 0$ است.

۲. نقاطِ سه‌گوش[۲]: . دو نقطه‌ی L_4 و L_5 رأس‌های مثلثِ متساوی‌الاضلاع با دو جرمِ اصلی را تشکیل می‌دهند و در آن‌ها $y \neq 0$.



شکل ۳-۲: نقاطِ لاگرانژ در سامانه‌ی زمین–ماه

نقاطِ هم‌خط (L_1, L_2, L_3)

با اعمال $y = 0$، تنها معادله‌ی زیر باقی می‌ماند:

$$x - \frac{1-\mu}{|x+\mu|^3}(x+\mu) - \frac{\mu}{|x-1+\mu|^3}(x-1+\mu) = 0. \qquad (۱۳-۳)$$

این معادله در سه ناحیه‌ی مجزا—بین دو جرم، بیرونِ جرمِ کوچک و بیرونِ جرمِ بزرگ—دارای یک ریشه است که به‌ترتیب نقاطِ L_1، L_2 و L_3 را تعیین می‌کند.

[1] Collinear
[2] Triangular

برای $\mu \ll 1$ (همچون سامانه‌ی خورشید–زمین یا زمین–ماه) می‌توان تقریب‌های شناخته‌شده را نوشت:

$$x_{L_1} \simeq (1-\mu) - \left(\tfrac{\mu}{3}\right)^{1/3},$$
$$x_{L_2} \simeq (1-\mu) + \left(\tfrac{\mu}{3}\right)^{1/3},$$
$$x_{L_3} \simeq -1 - \tfrac{5}{12}\,\mu; \qquad y_{L_i} = 0.$$

در عمل، ریشه‌ی دقیقِ معادله‌ی (۳-۱۳) با یک روشِ عددی (نیوتن–رافسون) محاسبه می‌شود.

نقاطِ سه‌گوش (L_4, L_5)

در این نقاط $r_1 = r_2 = 1$ و شرطِ $1-(1-\mu)/r_1^3 - \mu/r_2^3 = 0$ به‌طور طبیعی برقرار است. مختصات به عبارت‌اند از

$$x_{L_4} = x_{L_5} = \tfrac{1}{2} - \mu, \qquad y_{L_4} = +\tfrac{\sqrt{3}}{2}, \qquad y_{L_5} = -\tfrac{\sqrt{3}}{2}. \tag{۳-۱۴}$$

پایداری این نقاط مستلزم نسبتِ جرمِ کافی است؛ شرطِ کلاسیک $m_1/m_2 > 24.96$ (یا معادل آن $\mu < \mu_{\mathrm{R}} \approx 0.03852$) در سامانه‌های خورشید–سیاره یا زمین–ماه برقرار است و سببِ وجودِ خانواده‌ی سیارک‌های تروجان حول L_4 و L_5 می‌شود. در مقابل، نقاطِ هم‌خط ناپایدارند و معمولاً مأموریت‌های فضایی روی مدارهای هاله‌ای یا لیساژور در پیرامونِ آن‌ها قرار می‌گیرند.

برای سامانه‌ی زمین–ماه، $\mu \simeq 0.01215$ است. جدولِ زیر مختصاتِ بی‌بُعدِ هر پنج نقطه را نشان می‌دهد (واحدِ طول: فاصله‌ی زمین–ماه). موقعیتِ زمین در $(-\mu, 0)$ و ماه در $(1-\mu, 0)$ است.

جدول ۳-۲: مقادیر عددی نقاط لاگرانژ برای مسئله‌ی سه‌جسمیِ محدودِ سیستمِ زمین–ماه

نقطه‌ی لاگرانژ	x (بی‌بعد)	y (بی‌بعد)
L_1	$+0.83692$	0
L_2	$+1.15568$	0
L_3	-1.00506	0
L_4	$+0.48785$	$+0.86603$
L_5	$+0.48785$	-0.86603

این نتایج نشان می‌دهد که L_1 در حدودِ 0.84 فاصله‌ی زمین–ماه از زمین قرار دارد (فاصله‌ی آن تا ماه در حدود 0.16 واحد طول است) و L_2 بیرونِ مدارِ ماه است. نقطه‌ی L_3 تقریباً یک واحدِ طول در سوی مقابلِ ماه نسبت به زمین قرار دارد. دو نقطه‌ی L_4 و L_5 در مختصات $(0.488, \pm 0.866)$ قرارگرفته و با زمین و ماه مثلثِ متساوی‌الاضلاع می‌سازند.

این نتایج نشان می‌دهد که L_1 در حدودِ 0.84 فاصله‌ی زمین–ماه از زمین قرار دارد (فاصله‌ی آن تا ماه در حدود 0.16 واحد طول است) و L_2 بیرونِ مدارِ ماه است. نقطه‌ی L_3 تقریباً یک واحدِ طول در سوی مقابلِ ماه نسبت به زمین قرار دارد. دو نقطه‌ی L_4 و L_5 در مختصات $(0.488, \pm 0.866)$ قرارگرفته و با زمین و ماه مثلثِ متساوی‌الاضلاع می‌سازند.

فصل ۴

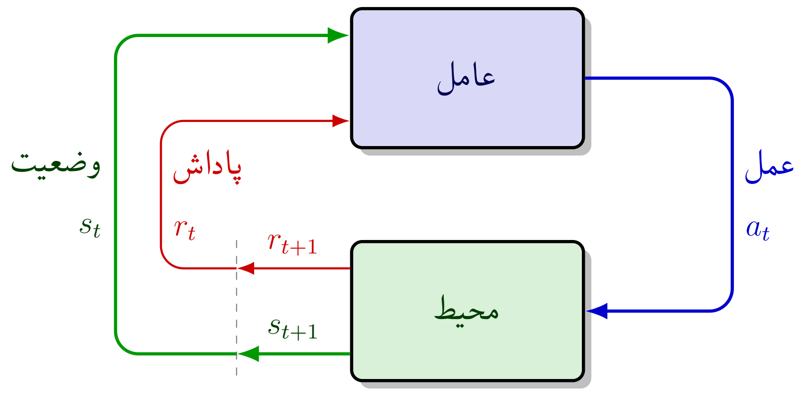
یادگیری تقویتی

در این فصل به بررسی یادگیری تقویتی پرداخته شده‌است. ابتدا در فصل ۴-۱ مفاهیم اولیه یادگیری تقویتی ارائه شده‌است. در ادامه عامل‌های گرادیان سیاست عمیق قطعی ۴-۲، گرادیان سیاست عمیق قطعی تاخیری دوگانه ۴-۳، عملگر نقاد نرم ۴-۴ و بهینه‌سازی سیاست مجاور ۴-۵ توضیح داده شده‌است.

۴-۱ مفاهیم اولیه

دو بخش اصلی یادگیری تقویتی[1] شامل عامل[2] و محیط[3] است. عامل در محیط قرار دارد و با آن در تعامل است. در هر مرحله از تعامل بین عامل و محیط، عامل یک مشاهده جزئی از وضعیت محیط انجام می‌دهد و سپس در مورد اقدامی که باید انجام دهد، تصمیم می‌گیرد. وقتی عامل روی محیط عمل می کند، محیط تغییر می‌کند؛ اما، ممکن است محیط به تنهایی نیز تغییر کند. عامل همچنین یک سیگنال پاداش[4] از محیط دریافت می‌کند؛ سیگنالی که به عامل می‌گوید وضعیت تعامل فعلی آن با محیط چقدر خوب یا بد است. هدف عامل بیشینه‌کردن پاداش انباشته خود است که برگشت[5] نام دارد. یادگیری تقویتی به روش‌هایی گفته می‌شود که در آن‌ها عامل رفتارهای مناسب برای رسیدن به هدف خود را می‌آموزد. در شکل ۴-۱ تعامل بین محیط و عامل نشان داده شده‌است.

[1] Reinforcement Learning (RL)
[2] Agent
[3] Environment
[4] Reward
[5] Return



شکل ۴-۱: حلقه تعامل عامل و محیط

۴-۱-۱ حالت و مشاهدات

حالت[۶] (s) توصیف کاملی از وضعیت محیط است. همه‌ی اطلاعات محیط در حالت وجود دارد. مشاهده[۷] (o) یک توصیف جزئی از حالت است که ممکن است شامل تمامی اطلاعات نباشد. در این پژوهش مشاهده توصیف کاملی از محیط هست؛ در نتیجه، حالت و مشاهده برابر هستند.

۴-۱-۲ فضای عمل

فضای عمل (a) در یادگیری تقویتی، مجموعه‌ای از تمام اقداماتی است که یک عامل می‌تواند در محیط انجام دهد. این فضا می‌تواند گسسته[۸] یا پیوسته[۹] باشد. در این پژوهش فضای عمل پیوسته و محدود به یک بازه مشخص است.

۴-۱-۳ سیاست

سیاست[۱۰] قاعده‌ای است که یک عامل برای تصمیم‌گیری در مورد اقدامات خود استفاده می‌کند. در این پژوهش به تناسب الگوریتم پیاده‌سازی‌شده از سیاست قطعی[۱۱] یا تصادفی[۱۲] استفاده شده‌است که به دو صورت زیر نشان

[6] State
[7] Observation
[8] Discrete
[9] Continuous
[10] Policy
[11] Deterministic
[12] Stochastic

داده می‌شود:

$$a_t = \mu(s_t) \tag{۴-۱}$$

$$a_t \sim \pi(\cdot|s_t) \tag{۴-۲}$$

که زیروند t بیانگر زمان است. در یادگیری تقویتی عمیق از سیاست‌های پارامتری‌شده‌استفاده می‌شود. خروجی این سیاست‌ها تابعی پارامترهای سیاست (وزن‌ها و بایاس‌های یک شبکه عصبی) هستند که می‌توان از الگوریتم‌های بهینه‌سازی جهت تعیین مقدار بهینه این پارامترها استفاده‌کرد. در این پژوهش پارامترهای سیاست با θ نشان داده شده‌است و سپس نماد آن به‌عنوان زیروند سیاست مانند معادله (۴-۳) نشان داده شده‌است.

$$\begin{aligned} a_t &= \mu_\theta(s_t) \\ a_t &\sim \pi_\theta(\cdot|s_t) \end{aligned} \tag{۴-۳}$$

۴-۱-۴ مسیر

یک مسیر[۱۳] یک توالی از حالت‌ها و عمل‌ها در محیط است.

$$\tau = (s_0, a_0, s_1, a_1, \cdots) \tag{۴-۴}$$

گذار حالت[۱۴] به اتفاقاتی که در محیط بین زمان t در حالت s_t و زمان $t+1$ در حالت s_{t+1} رخ می‌دهد، گفته می‌شود. این گذارها توسط قوانین طبیعی محیط انجام می‌شوند و تنها به آخرین اقدام انجام‌شده توسط عامل (a_t) بستگی دارند. گذار حالت را می‌توان به‌صورت زیر تعریف کرد.

$$s_{t+1} = f(s_t, a_t) \tag{۴-۵}$$

۴-۱-۵ تابع پاداش و برگشت

تابع پاداش[۱۵] در حالت کلی به حالت فعلی محیط، آخرین عمل انجام‌شده و حالت بعدی محیط بستگی دارد. تابع پاداش را می‌توان به‌صورت زیر تعریف کرد.

$$r_t = R(s_t, a_t, s_{t+1}) \tag{۴-۶}$$

[13] Trajectory
[14] State Transition
[15] Reward Function

در این پژوهش، پاداش تنها تابعی از جفتِ حالت-عمل $(r_t = R(s_t, a_t))$ فرض شده‌است. هدف عامل این است که مجموع پاداش‌های به‌دست‌آمده در طول یک مسیر را به حداکثر برساند. در این پژوهش مجموع پاداش‌ها در طول یک مسیر را با نماد $R(\tau)$ نشان داده شده‌است و به آن تابع برگشت[۱۶] گفته می‌شود. یکی از انواع برگشت، برگشت بدون تنزیل[۱۷] با افق محدود[۱۸] است که مجموع پاداش‌های به‌دست‌آمده در یک بازه زمانی ثابت و از مسیر τ است که در معادله (۴-۷) نشان داده شده‌است.

$$R(\tau) = \sum_{t=0}^{T} r_t \tag{۴-۷}$$

نوع دیگری از برگشت، برگشت تنزیل‌شده با افق نامحدود[۱۹] است که مجموع همه پاداش‌هایی است که تا به حال توسط عامل به‌دست آمده‌است. اما، فاصله زمانی تا دریافت پاداش باعث تنزیل ارزش آن می‌شود. این معادله برگشت (۴-۸) شامل یک فاکتور تنزیل[۲۰] با نماد γ است که عددی بین صفر و یک است.

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t \tag{۴-۸}$$

۴-۱-۶ ارزش در یادگیری تقویتی

در یادگیری تقویتی، دانستن ارزش[۲۱] یک حالت یا جفتِ حالت-عمل ضروری است. منظور از ارزش، برگشت مورد انتظار[۲۲] است. یعنی اگر از آن حالت یا جفت حالت-عمل شروع شود و سپس برای همیشه طبق یک سیاست خاص عمل شود، به‌طور میانگین چه مقدار پاداش دریافت خواهد شد. توابع ارزش تقریباً در تمام الگوریتم‌های یادگیری تقویتی به کار می‌روند. در اینجا به چهار تابع مهم اشاره شده‌است.

۱. تابع ارزش تحت سیاست[۲۳] $(V^{\pi}(s))$: خروجی این تابع برگشت مورد انتظار است در صورتی که از حالت s شروع شود و همیشه طبق سیاست π عمل شود و به‌صورت زیر بیان می‌شود:

$$V^{\pi}(s) = \mathop{\mathbb{E}}_{\tau \sim \pi} [R(\tau) | s_0 = s] \tag{۴-۹}$$

۲. تابع ارزش-عمل تحت سیاست[۲۴] $(Q^{\pi}(s, a))$: خروجی این تابع برگشت مورد انتظار است در صورتی که از حالت s شروع شود، یک اقدام دلخواه a (که ممکن است از سیاست π نباشد) انجام شود و سپس

[16] Return
[17] Discount
[18] Finite-Horizon Undiscounted Return
[19] Infinite-Horizon Discounted Return
[20] Discount Factor
[21] Value
[22] Expected Return
[23] On-Policy Value Function
[24] On-Policy Action-Value Function

برای همیشه طبق سیاست π عمل شود و به‌صورت زیر بیان می‌شود:

$$Q^\pi(s,a) = \underset{\tau\sim\pi}{\mathbb{E}}[R(\tau)|s_0 = s, a_0 = a] \tag{۱۰-۴}$$

۳. تابع ارزش بهینه[۲۵] ($V^*(s)$): خروجی این تابع برگشت مورد انتظار است در صورتی که از حالت s شروع شود و همیشه طبق سیاست بهینه در محیط عمل شود و به‌صورت زیر بیان می‌شود:

$$V^*(s) = \max_\pi(V^\pi(s)) \tag{۱۱-۴}$$

۴. تابع ارزش–عمل بهینه[۲۶] ($Q^*(s,a)$): خروجی این تابع برگشت مورد انتظار است در صورتی که از حالت s شروع شود، یک اقدام دلخواه a انجام شود و سپس برای همیشه طبق سیاست بهینه در محیط عمل شود و به‌صورت زیر بیان می‌شود:

$$Q^*(s,a) = \max_\pi(Q^\pi(s,a)) \tag{۱۲-۴}$$

۴-۱-۷ معادلات بلمن

توابع ارزش اشاره‌شده از معادلات خاصی که به آن‌ها معادلات بلمن گفته می‌شود، پیروی می‌کنند. ایده اصلی پشت معادلات بلمن این است که ارزش نقطه شروع برابر است با پاداشی است که انتظار دارید از آنجا دریافت کنید، به علاوه ارزش مکانی که بعداً به آنجا می‌رسید. معادلات بلمن برای توابع ارزش سیاست محور به شرح زیر هستند:

$$V^\pi(s) = \underset{\substack{a\sim\pi \\ s'\sim P}}{\mathrm{E}}[r(s,a) + \gamma V^\pi(s')] \tag{۱۳-۴}$$

$$Q^\pi(s,a) = r(s,a) + \underset{\substack{a\sim\pi \\ s'\sim P}}{\mathrm{E}}\left[\gamma \underset{a'\sim\pi}{\mathrm{E}}[Q^\pi(s',a')]\right] \tag{۱۴-۴}$$

که در آن $V^\pi(s)$ تابع ارزش حالت s تحت سیاست π است؛ $Q^\pi(s,a)$ تابع ارزش عمل a در حالت s تحت سیاست π است؛ $R(s,a)$ پاداش دریافتی پس از انجام عمل a در حالت s است؛ γ ضریب تنزیل است که ارزش پاداش‌های آینده را کاهش می‌دهد؛ $s' \sim P(\cdot|s,a)$ نشان می‌دهد که حالت بعدی s' از توزیع انتقال محیط P با شرط‌های s و a نمونه‌برداری می‌شود؛ و $a' \sim \pi(\cdot|s')$ نشان می‌دهد که عمل بعدی a' از سیاست

[25] Optimal Value Function
[26] Optimal Action-Value Function

π با شرط حالت جدید s' نمونه‌برداری می‌شود. این معادلات بیانگر این هستند که ارزش یک حالت یا عمل، مجموع پاداش مورد انتظار آن و ارزش حالت بعدی است که بر اساس سیاست فعلی تعیین می‌شود. معادلات بلمن برای توابع ارزش بهینه به شرح زیر هستند:

$$V^*(s) = \max_a \mathop{\mathrm{E}}_{s'\sim P}\left[r(s,a) + \gamma V^*(s')\right] \tag{۱۵-۴}$$

$$Q^*(s,a) = r(s,a) + \mathop{\mathrm{E}}_{s'\sim P}\left[\gamma \max_{a'} Q^*(s',a')\right] \tag{۱۶-۴}$$

تفاوت حیاتی بین معادلات بلمن برای توابع ارزش سیاست محور و توابع ارزش بهینه، عدم حضور یا حضور عملگر max بر روی اعمال است. حضور آن منعکس‌کننده این است که هرگاه عامل بتواند عمل خود را انتخاب کند، برای عمل بهینه، باید هر عملی را که منجر به بالاترین ارزش می‌شود انتخاب کند.

۴-۱-۸ تابع مزیت

گاهی در یادگیری تقویتی، نیازی به توصیف میزان خوبی یک عمل به صورت مطلق نیست، بلکه تنها می‌خواهیم بدانیم که چه مقدار بهتر از سایر اعمال به‌طور متوسط است. به عبارت دیگر، مزیت نسبی آن عمل مورد بررسی قرار می‌گیرد. این مفهوم با تابع مزیت[۲۷] توضیح داده می‌شود.

تابع مزیت $A^\pi(s,a)$ که مربوط به سیاست π است، توصیف می‌کند که انجام یک عمل خاص a در حالت s چقدر بهتر از انتخاب تصادفی یک عمل بر اساس $\pi(\cdot|s)$ است، با فرض اینکه شما برای همیشه پس از آن مطابق با π عمل می‌کنید. به صورت ریاضی، تابع مزیت به صورت زیر تعریف می‌شود:

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

که در آن $A^\pi(s,a)$ تابع مزیت برای عمل a در حالت s است. $Q^\pi(s,a)$ تابع ارزش عمل a در حالت s تحت سیاست π است و $V^\pi(s)$ تابع ارزش حالت s تحت سیاست π است. این تابع مزیت نشان می‌دهد که انجام عمل a در حالت s نسبت به میانگین اعمال تحت سیاست π چقدر مزیت دارد. اگر $A^\pi(s,a)$ مثبت باشد، نشان‌دهنده این است که عمل a بهتر از میانگین اعمال است و اگر منفی باشد، نشان‌دهنده کمتربودن عملکرد آن نسبت به میانگین است.

[27] Advantage Function

۴-۲ عامل گرادیان سیاست عمیق قطعی

گرادیان سیاست عمیق قطعی[۲۸] الگوریتمی است که همزمان یک تابع Q و یک سیاست را یاد می‌گیرد. این الگوریتم برای یادگیری تابع Q از داده‌های غیرسیاست محور[۲۹] و معادله بلمن استفاده می‌کند. این الگوریتم برای یادگیری سیاست نیز از تابع Q استفاده می‌کند.

این رویکرد وابستگی نزدیکی به یادگیری Q دارد. اگر تابع ارزش–عمل بهینه مشخص باشد، در هر حالت داده‌شده عمل بهینه را می‌توان با حل معادله (۴-۱۷) به دست آورد.

$$a^*(s) = \arg\max_a Q^*(s,a) \tag{۴-۱۷}$$

الگوریتم DDPG ترکیبی از یادگیری تقریبی برای $Q^*(s,a)$ و یادگیری تقریبی برای $a^*(s)$ است و به‌صورتی طراحی شده‌است که برای محیط‌هایی با فضاهای عمل پیوسته مناسب باشد. آنچه این الگوریتم را برای فضای عمل پیوسته مناسب می‌کند، روش محاسبه $a^*(s)$ است. فرض می‌شود که تابع $Q^*(s,a)$ نسبت به آرگومان عمل مشتق‌پذیر است. مشتق‌پذیری این امکان را می‌دهد که یک روش یادگیری مبتنی بر گرادیان برای سیاست $\mu(s)$ استفاده شود. سپس، به‌جای اجرای یک بهینه‌سازی زمان‌بر در هر بار محاسبه $\max_a Q(s,a)$، می‌توان آن را با رابطه $\max_a Q(s,a) \approx Q(s,\mu(s))$ تقریب زد.

۴-۲-۱ یادگیری Q در DDPG

معادله بلمن که تابع ارزش عمل بهینه $(Q^*(s,a))$ را توصیف می‌کند، در پایین آورده شده‌است.

$$Q^*(s,a) = r(s,a) + \mathop{\mathrm{E}}_{s'\sim P}\left[\gamma \max_{a'} Q^*(s',a')\right] \tag{۴-۱۸}$$

عبارت $s' \sim P$ به این معنی است که وضعیت بعدی یعنی s' از توزیع احتمال $P(\cdot|s,a)$ نمونه گرفته می‌شود. در معادله بلمن نقطه شروع برای یادگیری $Q^*(s,a)$ یک مقداردهی تقریبی است. پارامترهای شبکه عصبی $Q_\phi(s,a)$ با علامت ϕ نشان داده شده‌است. مجموعه $\mathcal{D}$ شامل اطلاعات جمع‌آوری شده تغییر از یک حالت به حالت دیگر (s,a,r,s',d) (که d نشان می‌دهد که آیا وضعیت s' پایانی است یا خیر) است. در بهینه‌سازی از تابع خطای میانگین مربعات بلمن[۳۰] (MSBE) استفاده شده‌است که معیاری برای نزدیکی Q_ϕ به حالت بهینه برای برآورده کردن معادله بلمن است.

[28] Deep Deterministic Policy Gradient (DDPG)
[29] Off-Policy
[30] Mean Squared Bellman Error

$$L(\phi, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}} \left[\left(Q_\phi(s,a) - \left(r + \gamma(1-d) \max_{a'} Q_\phi(s',a') \right) \right)^2 \right] \qquad (۱۹-۴)$$

در الگوریتم DDPG دو ترفند برای عمکرد بهتر استفاده شده‌است که در ادامه به بررسی آن پرداخته شده‌است.

- بافرهای تکرار بازی

الگوریتم‌های یادگیری تقویتی جهت آموزش یک شبکه عصبی عمیق برای تقریب $Q^*(s,a)$ از بافرهای تکرار بازی[۳۱] تجربه‌شده‌استفاده می‌کنند. این مجموعه $\mathcal{D}$ شامل تجربیات قبلی عامل است. برای داشتن رفتار پایدار در الگوریتم، بافر تکرار بازی باید به اندازه کافی بزرگ باشد تا شامل یک دامنه گسترده از تجربیات شود. انتخاب داده‌های بافر به دقت انجام شده‌است چرا که اگر فقط از داده‌های بسیار جدید استفاده شود، بیش‌برازش[۳۲] رخ می‌دهید و اگر از تجربه بیش از حد استفاده شود، ممکن است فرآیند یادگیری کند شود.

- شبکه‌های هدف

الگوریتم‌های یادگیری Q از شبکه‌های هدف استفاده می‌کنند. اصطلاح زیر به‌عنوان هدف شناخته می‌شود.

$$r + \gamma(1-d) \max_{a'} Q_\phi(s',a') \qquad (۲۰-۴)$$

در هنگام کمینه کردن تابع خطای میانگین مربعات بلمن، سعی شده‌است تا تابع Q شبیه‌تر به هدف یعنی رابطه (۴-۲۰) شود. اما مشکل این است که هدف بستگی به پارامترهای در حال آموزش ϕ دارد. این باعث ایجاد ناپایداری در کمینه کردن تابع خطای میانگین مربعات بلمن می‌شود. راه حل آن استفاده از یک مجموعه پارامترهایی است که با تأخیر زمانی به ϕ نزدیک می‌شوند. به عبارت دیگر، یک شبکه دوم ایجاد می‌شود که به آن شبکه هدف گفته می‌شود. شبکه هدف پارامترهای شبکه اول را با تاخیر دنبال می‌کند. پارامترهای شبکه هدف با نشان ϕ_{targ} نشان داده می‌شوند. در الگوریتم DDPG، شبکه هدف در هر به‌روزرسانی شبکه اصلی، با میانگین‌گیری پولیاک[۳۳] به‌صورت زیر به‌روزرسانی می‌شود.

$$\phi_{\text{targ}} \leftarrow \rho\phi_{\text{targ}} + (1-\rho)\phi \qquad (۲۱-۴)$$

در رابطه بالا ρ یک ابرپارامتر[۳۴] است که بین صفر و یک انتخاب می‌شود. در این پژوهش این مقدار نزدیک به یک درنظر گرفته شده‌است.

[31] Replay Buffers
[32] Overfit
[33] Polyak Averaging
[34] Hyperparameter

الگوریتم DDPG نیاز به یک شبکه سیاست هدف ($\mu_{\theta_{\text{targ}}}$) برای محاسبه عمل‌هایی که به‌طور تقریبی بیشینه $Q_{\phi_{\text{targ}}}$ را حاصل کند، را دارد. برای رسیدن به این شبکه سیاست هدف از همان روشی که تابع Q به دست می‌آید یعنی با میانگین‌گیری پولیاک از پارامترهای سیاست در طول زمان آموزش استفاده می‌شود.

با درنظرگرفتن موارد اشاره‌شده، یادگیری Q در DDPG با کمینه‌کردن تابع خطای میانگین مربعات بلمن (MSBE) یعنی معادله (۴-۲۲) با استفاده از کاهش گرادیان تصادفی[۳۵] انجام می‌شود.

$$L(\phi, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}} \left[\left(Q_{\phi}(s,a) - \left(r + \gamma(1-d) Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))\right) \right)^2 \right] \qquad (۴-۲۲)$$

۴-۲-۲ سیاست در DDPG

در این بخش یک سیاست تعیین‌شده $\mu_{\theta}(s)$ یاد گرفته می‌شود تا عملی را انجام می‌دهد که بیشینه $Q_{\phi}(s,a)$ رخ دهد. از آنجا که فضای عمل پیوسته است و فرض شده‌است که تابع Q نسبت به عمل مشتق‌پذیر است، رابطه زیر با استفاده از صعود گرادیان[۳۶] (تنها نسبت به پارامترهای سیاست) بیشینه می‌شود.

$$\max_{\theta} \underset{s\sim\mathcal{D}}{\mathrm{E}} \left[Q_{\phi}(s, \mu_{\theta}(s)) \right] \qquad (۴-۲۳)$$

۴-۲-۳ اکتشاف و بهره‌برداری در DDPG

برای بهبود اکتشاف[۳۷] در سیاست‌های DDPG، در زمان آموزش نویز به عمل‌ها اضافه می‌شود. نویسندگان مقاله DDPG [۵۶] توصیه کرده‌اند که نویز OU[۳۸] با همبندی زمانی[۳۹] اضافه شود. در زمان بهره‌برداری[۴۰] سیاست، از آنچه یاد گرفته است، نویز به عمل‌ها اضافه نمی‌شود.

۴-۲-۴ شبه‌کد DDPG

در این بخش، شبه‌کد الگوریتم DDPG پیاده‌سازی‌شده آورده شده‌است. در این پژوهش الگوریتم ۱ در محیط پایتون با استفاده از کتابخانه TensorFlow [۵۷] پیاده‌سازی شده‌است.

[35] Stochastic Gradient Descent
[36] Gradient Ascent
[37] Exploration
[38] Ornstein–Uhlenbeck
[39] Time-Correlated
[40] Exploitation

الگوریتم ۱ گرادیان سیاست عمیق قطعی

ورودی: پارامترهای اولیه سیاست (θ)، پارامترهای تابع Q (ϕ)، بافر تکرار بازی خالی ($\mathcal{D}$)

۱: پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید $\theta_{\text{targ}} \leftarrow \theta$، $\phi_{\text{targ}} \leftarrow \phi$

۲: تا وقتی همگرایی رخ دهد:

۳: وضعیت s را مشاهده کرده و عمل $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{\text{Low}}, a_{\text{High}})$ را انتخاب کنید به‌طوری که $\epsilon \sim \mathcal{N}$ است.

۴: عمل a را در محیط اجرا کنید.

۵: وضعیت بعدی s'، پاداش r و سیگنال پایان d را مشاهده کنید تا نشان دهد آیا s' پایانی است یا خیر.

۶: اگر s' پایانی است، وضعیت محیط را بازنشانی کنید.

۷: اگر زمان به‌روزرسانی فرا رسیده است:

۸: به ازای هر تعداد به‌روزرسانی:

۹: یک دسته تصادفی گذر از یک حالت به حالت دیگر، $B = \{(s, a, r, s', d)\}$، از $\mathcal{D}$ نمونه‌گیری شود.

۱۰: هدف را محاسبه کنید:

$$y(r, s', d) = r + \gamma(1 - d) Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))$$

۱۱: تابع Q را با یک مرحله از نزول گرادیان با استفاده از رابطه زیر به‌روزرسانی کنید:

$$\nabla_\phi \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_\phi(s, a) - y(r, s', d)\right)^2$$

۱۲: سیاست را با یک مرحله از صعود گرادیان با استفاده از رابطه زیر به‌روزرسانی کنید:

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_\phi(s, \mu_\theta(s))$$

۱۳: شبکه‌های هدف را با استفاده از معادلات زیر به‌روزرسانی کنید:

$$\phi_{\text{targ}} \leftarrow \rho \phi_{\text{targ}} + (1 - \rho)\phi$$

$$\theta_{\text{targ}} \leftarrow \rho \theta_{\text{targ}} + (1 - \rho)\theta$$

۴-۳ عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه

عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه[۴۱] یکی از الگوریتم‌های یادگیری تقویتی است که برای حل مسائل کنترل در محیط‌های پیوسته طراحی شده‌است. این الگوریتم بر اساس الگوریتم DDPG توسعه یافته و با استفاده از تکنیک‌های مختلف، پایداری و کارایی یادگیری را بهبود می‌بخشد. در حالی که DDPG گاهی اوقات می‌تواند عملکرد بسیار خوبی داشته باشد، اما اغلب نسبت به ابرپارامترها و سایر انواع تنظیمات یادگیری حساس است. یک حالت رایج شکستِ عامل DDPG در یادگیری این است که تابع Q یادگرفته‌شده شروع به بیش‌برآورد مقادیر Q می‌کند که منجر به واگرایی سیاست می‌شود. واگرایی به این دلیل رخ می‌دهد که در فرآیند یادگیری سیاست از تخمین تابع Q استفاده می‌شود که افزایش خطای تابع Q منجر به ناپایداری در یادگیری سیاست می‌شود.

الگوریتم (Twin Delayed DDPG) TD3 از دو ترفند زیر جهت بهبود مشکلات اشاره‌شده استفاده می‌کند.

- یادگیری دوگانه‌ی محدودشده[۴۲]: الگوریتم TD3 به جای یک تابع Q، دو تابع Q_{ϕ_1} و Q_{ϕ_2} را یاد می‌گیرد (از این رو دوگانه[۴۳] نامیده می‌شود) و از کوچک‌ترین مقدار این دو Q_{ϕ_1} و Q_{ϕ_2} در تابع بلمن استفاده می‌شود. نحوه محاسبه هدف بر اساس دو تابع Q اشاره‌شده در رابطه (۴-۲۴) آورده شده‌است.

$$y(r, s', d) = r + \gamma(1-d) \min_{i=1,2} Q_{\phi_{i,\text{targ}}}(s', a'(s')) \tag{۴-۲۴}$$

سپس، در هر دو تابع Q_{ϕ_1} و Q_{ϕ_2} یادگیری انجام می‌شود.

$$L(\phi_1, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}} \left(Q_{\phi_1}(s,a) - y(r,s',d) \right)^2 \tag{۴-۲۵}$$

$$L(\phi_2, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}} \left(Q_{\phi_2}(s,a) - y(r,s',d) \right)^2 \tag{۴-۲۶}$$

- به‌روزرسانی‌های تاخیری سیاست[۴۴]: الگوریتم TD3 سیاست را با تاخیر بیشتری نسبت به تابع Q به‌روزرسانی می‌کند. در مرجع [۵۸] توصیه شده‌است که برای هر دو به‌روزرسانی تابع Q، یک به‌روزرسانی سیاست انجام شود.

[41] Twin Delayed Deep Deterministic Policy Gradient (TD3)
[42] Clipped Double-Q Learning
[43] twin
[44] Delayed Policy Updates

این دو ترفند منجر به بهبود قابل توجه عملکرد TD3 نسبت به DDPG پایه می‌شوند. در نهایت سیاست با بیشینه‌کردن Q_{ϕ_1} آموخته می‌شود:

$$\max_{\theta} \underset{s\sim\mathcal{D}}{\mathrm{E}}\left[Q_{\phi_1}(s,\mu_\theta(s))\right] \tag{۴-۲۷}$$

۴-۳-۱ اکتشاف و بهره‌برداری در TD3

الگوریتم TD3 یک سیاست قطعی را به‌صورت غیرسیاست‌محور آموزش می‌دهد. از آنجایی که سیاست قطعی است، در ابتدا عامل تنوع کافی از اعمال را برای یافتن روش‌های مفید امتحان نمی‌کند. برای بهبود اکتشاف سیاست‌های TD3، در زمان آموزش نویز به عمل‌ها اضافه می‌شود. در این پژوهش، نویز گاوسی با میانگین صفر بدون هم‌بندی زمانی اعمال شده‌است. شدت نویز جهت بهره‌برداری بهتر در طول زمان کاهش می‌یابد.

۴-۳-۲ شبه‌کد TD3

در این بخش الگوریتم TD3 پیاده‌سازی شده آورده شده‌است. در این پژوهش الگوریتم ۴ در محیط پایتون با استفاده از کتابخانه PyTorch [۵۹] پیاده‌سازی شده‌است.

الگوریتم ۲ عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه

ورودی: پارامترهای اولیه سیاست (θ)، پارامترهای تابع Q (ϕ_1, ϕ_2)، بافر بازی خالی ($\mathcal{D}$)

پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید $\theta_{\text{targ}} \leftarrow \theta$، $\phi_{\text{targ},1} \leftarrow \phi_1$، $\phi_{\text{targ},2} \leftarrow \phi_2$

تا وقتی همگرایی رخ دهد:

وضعیت (s) را مشاهده کرده و عمل $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{\text{Low}}, a_{\text{High}})$ را انتخاب کنید، به‌طوری که $\epsilon \sim \mathcal{N}$ است.

عمل a را در محیط اجرا کنید.

وضعیت بعدی s'، پاداش r و سیگنال پایان d را مشاهده کنید تا نشان دهد آیا s' پایانی است یا خیر.

اگر s' پایانی است، وضعیت محیط را بازنشانی کنید.

اگر زمان به‌روزرسانی فرا رسیده است:

به ازای j در هر تعداد به‌روزرسانی:

یک دسته تصادفی گذر از یک حالت به حالت دیگر، $B = \{(s, a, r, s', d)\}$، از $\mathcal{D}$ نمونه‌گیری شود.

هدف را محاسبه کنید:

$$y(r, s', d) = r + \gamma(1-d) \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', a'(s'))$$

تابع Q را با یک مرحله از نزول گرادیان با استفاده از رابطه زیر به‌روزرسانی کنید:

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_{\phi_i}(s,a) - y(r,s',d)\right)^2 \quad \text{for } i = 1, 2$$

اگر باقیمانده j بر تاخیر سیاست برابر 0 باشد :

سیاست را با یک مرحله از صعود گرادیان با استفاده از رابطه زیر به‌روزرسانی کنید:

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_{\phi_1}(s, \mu_\theta(s))$$

شبکه‌های هدف را با استفاده از معادلات زیر به‌روزرسانی کنید:

$$\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1-\rho)\phi_i \quad \text{for } i = 1, 2$$

$$\theta_{\text{targ}} \leftarrow \rho \theta_{\text{targ}} + (1-\rho)\theta$$

۴-۴ عامل عملگر نقاد نرم

عملگرد نقاد نرم[۴۵] الگوریتمی است که یک سیاست تصادفی را به‌صورت غیرسیاست‌محور بهینه می‌کند و پلی بین بهینه‌سازی سیاست تصادفی و رویکردهای غیرسیاست‌محور مانند DDPG ایجاد می‌کند. این الگوریتم جانشین مستقیم TD3 نیست (زیرا تقریباً همزمان منتشر شده‌است)؛ اما، ترفند یادگیری دوگانه محدودشده را در خود جای داده است و به‌دلیل سیاست تصادفی SAC، از روشی به نام صاف‌کردن سیاست هدف[۴۶] استفاده شده‌است. یکی از ویژگی های اصلی SAC، تنظیم آنتروپی است. آنتروپی معیاری از تصادفی‌بودن انتخاب عمل در سیاست است. آموزش سیاست در جهت تعادل بهینه بین آنتروپی و بیشنه‌سازی بازده مورد انتظار است. این شرایط ارتباط نزدیکی با تعادل اکتشاف–بهره‌برداری دارد. افزایش آنتروپی منجر به اکتشاف بیشتر می‌شود که می‌تواند یادگیری را در مراحل بعدی تسریع کند. همچنین، می‌تواند از همگرایی زودهنگام سیاست به یک بهینه محلی بد جلوگیری کند. برای توضیح SAC، ابتدا باید به بررسی یادگیری تقویتی تنظیم‌شده با آنتروپی[۴۷] پرداخته شود. در RL تنظیم‌شده با آنتروپی، روابط تابع ارزش کمی متفاوت است.

۴-۴-۱ یادگیری تقویتی تنظیم‌شده با آنتروپی

آنتروپی معیاری برای سنجش میزان عدم قطعیت یا تصادفی بودن یک متغیر تصادفی یا توزیع احتمال آن است. به عبارت دقیق‌تر، آنتروپی برای یک توزیع احتمال، میانگین اطلاعات حاصل از نمونه‌برداری از آن توزیع را اندازه‌گیری می‌کند. در زمینه یادگیری تقویتی، تنظیم با آنتروپی تکنیکی است که با افزودن یک ترم متناسب با آنتروپی سیاست به تابع هدف، عامل را تشویق به اکتشاف بیشتر و اتخاذ سیاست‌های تصادفی‌تر می‌کند. این امر می‌تواند به بهبود پایداری فرآیند یادگیری و جلوگیری از همگرایی زودهنگام به بهینه‌های محلی کمک کند.

فرض کنید X یک متغیر تصادفی پیوسته با تابع چگالی احتمال $p(x)$ باشد. آنتروپی $H(X)$ این متغیر تصادفی به صورت امید ریاضی لگاریتم منفی چگالی احتمال آن تعریف می‌شود:

$$H(X) = \mathrm{E}_{x\sim p}\left[-\log p(x)\right] \tag{۴-۲۸}$$

۴-۴-۲ سیاست در SAC

در یادگیری تقویتی تنظیم‌شده با آنتروپی، عامل در هر مرحله زمانی متناسب با آنتروپی سیاست در آن مرحله زمانی پاداش دریافت می‌کند. بر اساس توضیحات اشاره شده روابط یادگیری تقویتی به‌صورت زیر می‌شود.

[45] Soft Actor Critic (SAC)
[46] Target Policy Smoothing
[47] Entropy-Regularized Reinforcement Learning

$$\pi^* = \arg\max_{\pi} \mathop{\mathrm{E}}_{\tau\sim\pi} \sum_{t=0}^{\infty} \gamma^t \Big(R(s_t, a_t, s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right) \Big) \tag{۴-۲۹}$$

که در آن $(\alpha > 0)$ ضریب مبادله[۴۸] است.

۴-۴-۳ تابع ارزش در SAC

اکنون می‌توان تابع ارزش کمی متفاوت را بر اساس این مفهوم تعریف کرد. V^π به گونه‌ای تغییر می‌کند که پاداش‌های آنتروپی را از هر مرحله زمانی شامل می‌شود.

$$V^\pi(s) = \mathop{\mathrm{E}}_{\tau\sim\pi} \left[\sum_{t=0}^{\infty} \gamma^t \Big(R(s_t, a_t, s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right) \Big) \middle| s_0 = s \right] \tag{۴-۳۰}$$

۴-۴-۴ تابع Q در SAC

تابع Q^π به گونه ای تغییر می‌کند که پاداش های آنتروپی را از هر مرحله زمانی به جز مرحله اول شامل می‌شود.

$$Q^\pi(s,a) = \mathop{\mathrm{E}}_{\tau\sim\pi} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) + \alpha \sum_{t=1}^{\infty} \gamma^t H\left(\pi(\cdot|s_t)\right) \middle| s_0 = s, a_0 = a \right] \tag{۴-۳۱}$$

با این تعاریف رابطه V^{π} و Q^π به‌صورت زیر است.

$$V^\pi(s) = \mathop{\mathrm{E}}_{a\sim\pi} \left[Q^\pi(s,a) \right] + \alpha H\left(\pi(\cdot|s)\right) \tag{۴-۳۲}$$

۴-۴-۵ معادله بلمن در SAC

معادله بلمن در حالت تنظیم‌شده با آنتروپی به‌صورت زیر ارائه می‌شود.

$$Q^\pi(s,a) = \mathop{\mathrm{E}}_{\substack{s'\sim P \\ a'\sim\pi}} \left[R(s,a,s') + \gamma \left(Q^\pi(s',a') + \alpha H\left(\pi(\cdot|s')\right) \right) \right] \tag{۴-۳۳}$$

$$= \mathop{\mathrm{E}}_{s'\sim P} \left[R(s,a,s') + \gamma V^\pi(s') \right] \tag{۴-۳۴}$$

[48] Trade-Off

۴-۴-۶ یادگیری Q

با درنظرگرفتن موارد اشاره‌شده، یادگیری Q در SAC با کمینه‌کردن تابع خطای میانگین مربعات بلمن (MSBE) یعنی معادله (۴-۳۵) با استفاده از کاهش گرادیان انجام می‌شود.

$$L(\phi_i, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}}\left[\left(Q_{\phi_i}(s,a) - y(r,s',d)\right)^2\right] \tag{۴-۳۵}$$

در معادله (۴-۳۵) تابع هدف برای روش یادگیری تقویتی SAC به صورت زیر تعریف می‌شود.

$$y(r,s',d) = r + \gamma(1-d)\left(\min_{j=1,2} Q_{\phi_{\text{targ},j}}(s',\tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s')\right), \qquad \tilde{a}' \sim \pi_\theta(\cdot|s') \tag{۴-۳۶}$$

نماد عمل بعدی را به جای a' به $\tilde{a}'$ تغییر داده شده تا مشخص شود که عمل‌های بعدی باید از آخرین سیاست نمونه‌برداری شوند در حالی که r و s باید از بافر تکرار بازی آمده باشند.

۴-۴-۷ سیاست در SAC

سیاست باید در هر وضعیت برای به حداکثر رساندن بازگشت مورد انتظار آینده به همراه آنتروپی مورد انتظار آینده عمل کند. یعنی باید $V^\pi(s)$ را به حداکثر برساند، بسط تابع ارزش در ادامه آمده است.

$$V^\pi(s) = \underset{a\sim\pi}{\mathrm{E}}\left[Q^\pi(s,a)\right] + \alpha H\left(\pi(\cdot|s)\right) \tag{۴-۳۷}$$

$$= \underset{a\sim\pi}{\mathrm{E}}\left[Q^\pi(s,a) - \alpha \log \pi(a|s)\right] \tag{۴-۳۸}$$

در بهینه‌سازی سیاست از ترفند پارامترسازی مجدد[۴۹] استفاده می‌شود، که در آن نمونه ای از $\pi_\theta(\cdot|s)$ با محاسبه یک تابع قطعی از وضعیت، پارامترهای سیاست و نویز مستقل استخراج می‌شود. در این پژوهش مانند نویسندگان مقاله SAC [۶۰]، از یک سیاست گاوسی فشرده[۵۰] استفاده شده‌است. بر اساس این روش نمونه‌ها مطابق با رابطه زیر بدست می‌آیند:

$$\tilde{a}_\theta(s,\xi) = \tanh\left(\mu_\theta(s) + \sigma_\theta(s) \odot \xi\right), \qquad \xi \sim \mathcal{N} \tag{۴-۳۹}$$

در رابطه بالا $\odot$ نماد ضرب داخلی است. تابع tanh در سیاست SAC تضمین می‌کند که اعمال در یک محدوده متناهی محدود شوند. این مورد در سیاست‌های VPG، TRPO و PPO وجود ندارد. همچنین اعمال این تابع توزیع را از حالت گاوسی تغییر می‌دهد.

[49] Reparameterization

[50] Squashed Gaussian Policy

در الگوریتم SAC با استفاده از ترفند پارامتری‌سازی مجدد، عمل‌ها از یک توزیع نرمال به‌وسیله نویز تصادفی تولید شده و به این ترتیب امکان محاسبه مشتق‌ها به‌طور مستقیم از طریق تابع توزیع فراهم می‌شود، که باعث ثبات و کارایی بیشتر در آموزش می‌شود. اما در حالت بدون پارامتری‌سازی مجدد، عمل‌ها مستقیماً از توزیع سیاست نمونه‌برداری می‌شوند و محاسبه گرادیان نیازمند استفاده از ترفند نسبت احتمال[۵۱] است که معمولاً باعث افزایش واریانس و ناپایداری در آموزش می‌شود.

$$\underset{a\sim\pi_\theta}{\mathrm{E}}\left[Q^{\pi_\theta}(s,a)-\alpha\log\pi_\theta(a|s)\right]=\underset{\xi\sim\mathcal{N}}{\mathrm{E}}\left[Q^{\pi_\theta}(s,\tilde{a}_\theta(s,\xi))-\alpha\log\pi_\theta(\tilde{a}_\theta(s,\xi)|s)\right] \tag{۴۰-۴}$$

برای به‌دست آوردن تابع هزینه سیاست، گام نهایی این است که باید Q^{π_θ} را با یکی از تخمین‌زننده‌های تابع خود جایگزین کنیم. برخلاف TD3 که از Q_{ϕ_1} (فقط اولین تخمین‌زننده Q) استفاده می‌کند، SAC از $\min_{j=1,2} Q_{\phi_j}$ (کمینه‌ی دو تخمین‌زننده Q) استفاده می‌کند. بنابراین، سیاست طبق رابطه زیر بهینه می‌شود:

$$\max_\theta \underset{\substack{s\sim\mathcal{D}\\ \xi\sim\mathcal{N}}}{\mathrm{E}}\left[\min_{j=1,2} Q_{\phi_j}(s,\tilde{a}_\theta(s,\xi))-\alpha\log\pi_\theta(\tilde{a}_\theta(s,\xi)|s)\right] \tag{۴۱-۴}$$

که تقریباً مشابه بهینه‌سازی سیاست در DDPG و TD3 است، به جز ترفند min-double-Q، تصادفی‌بودن و عبارت آنتروپی.

۴-۴-۸ اکتشاف و بهره‌برداری در SAC

الگوریتم SAC یک سیاست تصادفی با تنظیم‌سازی آنتروپی آموزش می‌دهد و به صورت سیاست محور به اکتشاف می‌پردازد. ضریب تنظیم آنتروپی α به طور صریح تعادل بین اکتشاف و بهره‌برداری را کنترل می‌کند، به‌طوری‌که مقادیر بالاتر α به اکتشاف بیشتر و مقادیر پایین‌تر α به بهره‌برداری بیشتر منجر می‌شود. مقدار بهینه α (که به یادگیری پایدارتر و پاداش بالاتر منجر می‌شود) ممکن است در محیط‌های مختلف متفاوت باشد و نیاز به تنظیم دقیق داشته باشد. در زمان آزمایش، برای ارزیابی میزان بهره‌برداری سیاست از آنچه یاد گرفته است، تصادفی بودن را حذف کرده و از عمل میانگین به جای نمونه‌برداری از توزیع استفاده می‌کنیم. این روش معمولاً عملکرد را نسبت به سیاست تصادفی بهبود می‌بخشد.

[51] Likelihood Ratio Trick

۹-۴-۴ شبه‌کد SAC

در این بخش الگوریتم SAC پیاده‌سازی شده آورده شده‌است. در این پژوهش الگوریتم ۳ در محیط پایتون با استفاده از کتابخانه PyTorch [۵۹] پیاده‌سازی شده است.

الگوریتم ۳ عامل عملگرد نقاد نرم

ورودی: پارامترهای اولیه سیاست (θ)، پارامترهای تابع Q (ϕ_1, ϕ_2)، بافر بازی خالی $(\mathcal{D})$

پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید $\theta_{\text{targ}} \leftarrow \theta$، $\phi_{\text{targ},1} \leftarrow \phi_1$، $\phi_{\text{targ},2} \leftarrow \phi_2$

تا وقتی همگرایی رخ دهد:

وضعیت (s) را مشاهده کرده و عمل $a \sim \pi_\theta(\cdot|s)$ را انتخاب کنید.

عمل a را در محیط اجرا کنید.

وضعیت بعدی s'، پاداش r و سیگنال پایان d را مشاهده کنید تا نشان دهد آیا s' پایانی است یا خیر.

اگر s' پایانی است، وضعیت محیط را بازنشانی کنید.

اگر زمان به‌روزرسانی فرا رسیده است:

به ازای j در هر تعداد به‌روزرسانی:

یک دسته تصادفی گذر از یک حالت به حالت دیگر، $B = \{(s, a, r, s', d)\}$، از $\mathcal{D}$ نمونه‌گیری شود.

هدف را محاسبه کنید:

$$y(r, s', d) = r + \gamma(1 - d) \left(\min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s') \right), \quad \tilde{a}' \sim \pi_\theta(\cdot|s')$$

تابع Q را با یک مرحله از نزول گرادیان با استفاده از رابطه زیر به‌روزرسانی کنید:

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} \left(Q_{\phi_i}(s, a) - y(r, s', d) \right)^2 \quad \text{for } i = 1, 2$$

سیاست را با یک مرحله از صعود گرادیان با استفاده از رابطه زیر به‌روزرسانی کنید:

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left(\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta \left(\tilde{a}_\theta(s) \middle| s \right) \right)$$

شبکه‌های هدف را با استفاده از معادلات زیر به‌روزرسانی کنید:

$$\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1 - \rho) \phi_i \quad \text{for } i = 1, 2$$

۵-۴ عامل بهینه‌سازی سیاست مجاور

الگوریتم بهینه‌سازی سیاست مجاور[52] یک الگوریتم بهینه‌سازی سیاست مبتنی بر گرادیان است که برای حل مسائل کنترل مسئله‌های یادگیری تقویتی استفاده می‌شود. این الگوریتم از الگوریتم TRPO[53] الهام گرفته شده‌است و با اعمال تغییراتی بر روی آن، سرعت و کارایی آن را افزایش داده است. در این بخش به بررسی این الگوریتم و نحوه عملکرد آن می‌پردازیم. الگوریتم PPO همانند سایر الگوریتم‌های یادگیری تقویتی، به‌دنبال یافتن بهترین گام ممکن برای بهبود عملکرد سیاست با استفاده از داده‌های موجود است. این الگوریتم تلاش می‌کند تا از گام‌های بزرگ که می‌توانند منجر به افت ناگهانی عملکرد شوند، اجتناب کند. برخلاف روش‌های پیچیده‌تر مرتبه دوم مانند TRPO، PPO از مجموعه‌ای از روش‌های مرتبه اول ساده‌تر برای حفظ نزدیکی سیاست‌های جدید به سیاست‌های قبلی استفاده می‌کند. این سادگی در پیاده‌سازی، PPO را به روشی کارآمدتر تبدیل می‌کند، در حالی که از نظر تجربی نشان داده شده‌است که عملکردی حداقل به اندازه TRPO دارد. از جمله ویژگی‌های مهم این الگوریتم می‌توان به سیاست محور بودن آن اشاره کرد. این الگوریتم برای عامل‌های یادگیری تقویتی که سیاست‌های پیوسته و گسسته دارند، مناسب است.

الگوریتم PPO دارای دو گونه اصلی PPO-Clip و PPO-Penalty است. در ادامه به بررسی هر یک از این دو گونه پرداخته شده‌است.

- روش **PPO-Penalty:** روش PPO-Penalty به‌دنبال حل تقریبی و به‌روزرسانی با محدودیت واگرایی کولباک–لیبلر[54] است، مشابه روشی که در الگوریتم TRPO استفاده شده‌است. با این حال، به جای اعمال یک محدودیت سخت[55]، PPO-Penalty واگرایی KL را در تابع هدف جریمه می‌کند. این جریمه به طور خودکار در طول آموزش تنظیم می‌شود تا از افت ناگهانی عملکرد جلوگیری کند.

- روش **PPO-Clip:** در این روش، هیچ عبارت واگرایی KL در تابع هدف وجود ندارد و هیچ محدودیتی اعمال نمی‌شود. در عوض، PPO-Clip از یک عملیات بریدن[56] خاص در تابع هدف استفاده می‌کند تا انگیزه سیاست جدید برای دور شدن از سیاست قبلی را از بین ببرد.

در این پژوهش از روش PPO-Clip برای آموزش عامل‌های یادگیری تقویتی استفاده شده‌است.

---
[52]Proximal Policy Optimization (PPO)
[53]Trust Region Policy Optimization
[54]Kullback-Leibler (KL) Divergence
[55]Hard Constraint
[56]Clipping

۴-۵-۱ سیاست در الگوریتم PPO

تابع سیاست در الگوریتم PPO به صورت یک شبکه عصبی پیاده‌سازی شده‌است. این شبکه عصبی ورودی‌های محیط را دریافت کرده و اقدامی را که باید عامل انجام دهد را تولید می‌کند. این شبکه عصبی می‌تواند شامل چندین لایه پنهان با توابع فعال‌سازی مختلف باشد. در این پژوهش از یک شبکه عصبی با سه لایه پنهان و تابع فعال‌سازی ReLu استفاده شده‌است. تابع سیاست در الگوریتم PPO به صورت زیر به‌روزرسانی می‌شود:

$$\theta_{k+1} = \arg\max_{\theta} \mathop{\mathrm{E}}_{s,a\sim\pi_{\theta_k}} \left[L(s,a,\theta_k,\theta)\right] \tag{۴-۴۲}$$

در این پژوهش برای به حداکثر رساندن تابع هدف، چندین گام بهینه‌سازی گرادیان کاهشی تصادفی[۵۷] اجرا شده‌است. در معادله بالا L به‌صورت زیر تعریف شده‌است:

$$L(s,a,\theta_k,\theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}A^{\pi_{\theta_k}}(s,a),\ \ \mathbf{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\right)A^{\pi_{\theta_k}}(s,a)\right) \tag{۴-۴۳}$$

که در آن ϵ یک ابرپارامتر است که مقدار آن معمولا کوچک است. این ابرپارامتر مشخص می‌کند که چقدر اندازه گام بهینه‌سازی باید محدود شود. در این پژوهش مقدار $\epsilon = 0.2$ انتخاب شده‌است. جهت سادگی در پیاده‌سازی معادله (۴-۴۳) به معادله تغیر داده شده‌است.

$$L(s,a,\theta_k,\theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}A^{\pi_{\theta_k}}(s,a),\ \ g(\epsilon, A^{\pi_{\theta_k}}(s,a))\right) \tag{۴-۴۴}$$

که تابع g به صورت زیر تعریف شده‌است.

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geqslant 0 \\ (1-\epsilon)A & A < 0 \end{cases} \tag{۴-۴۵}$$

در حالی که این نوع محدود کردن (PPO-Clip) تا حد زیادی به اطمینان از به‌روزرسانی‌های معقول سیاست کمک می‌کند، همچنان ممکن است سیاستی به‌دست آید که بیش از حد از سیاست قدیمی دور باشد. برای جلوگیری از این امر، پیاده‌سازی‌های مختلف PPO از مجموعه‌ای از ترفندها استفاده می‌کنند. در پیاده‌سازی این پژوهش، از روشی ساده به نام توقف زودهنگام[۵۸] استفاده شده‌است. اگر میانگین واگرایی کولباک–لیبلر (KL) خط‌مشی جدید از خط‌مشی قدیمی از یک آستانه فراتر رود، گام‌های گرادیان (بهینه‌سازی) را متوقف می‌شوند.

[57] Stochastic Gradient Descent (SGD)
[58] Early Stopping

۴-۵-۲ اکتشاف و بهره‌برداری در PPO

الگوریتم PPO از یک سیاست تصادفی به‌صورت سیاست‌محور برای آموزش استفاده می‌کند. این به این معنی است که اکتشاف محیط با نمونه‌گیری عمل‌ها بر اساس آخرین نسخه از این سیاست تصادفی انجام می‌شود. میزان تصادفی بودن انتخاب عمل به شرایط اولیه و فرآیند آموزش بستگی دارد.

در طول آموزش، سیاست به طور کلی به تدریج کمتر تصادفی می‌شود، زیرا قانون به‌روزرسانی آن را تشویق می‌کند تا از پاداش‌هایی که قبلاً پیدا کرده است، بهره‌برداری کند. البته این موضوع می‌تواند منجر به رسیدن سیاست به بهینه‌های محلی[۵۹] شود.

۴-۵-۳ شبه‌کد PPO

در این بخش الگوریتم PPO پیاده‌سازی شده آورده شده‌است. در این پژوهش الگوریتم ۴ در محیط پایتون با استفاده از کتابخانه PyTorch [۵۹] پیاده‌سازی شده است.

[59] Local Optima

الگوریتم ۴ بهینه‌سازی سیاست مجاور (PPO-Clip)

ورودی: پارامترهای اولیه سیاست (θ_0)، پارامترهای تابع ارزش (ϕ_0)

۱: به ازای $k = 0, 1, 2, ...$:

۲: مجموعه‌ای از مسیرها به نام $\mathcal{D}_k = \{\tau_i\}$ با اجرای سیاست $\pi_k = \pi(\theta_k)$ در محیط جمع‌آوری شود.

۳: پاداش‌های باقی‌مانده ($\hat{R}_t$) محاسبه شود.

۴: برآوردهای مزیت را محاسبه کنید، $\hat{A}_t$ (با استفاده از هر روش تخمین مزیت) بر اساس تابع ارزش فعلی V_{ϕ_k}.

۵: سیاست را با به حداکثر رساندن تابع هدف PPO-Clip به‌روزرسانی کنید:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} A^{\pi_{\theta_k}}(s_t, a_t), \quad g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right)$$

معمولاً از طریق گرادیان افزایشی تصادفی Adam.

۶: برازش تابع ارزش با رگرسیون بر روی میانگین مربعات خطا:

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^{T} \left(V_\phi(s_t) - \hat{R}_t \right)^2$$

معمولاً از طریق برخی از الگوریتم‌های کاهشی گرادیان.

فصل ۵

شبیه‌سازی عامل درمحیط سه جسمی

در این فصل، فرآیند شبیه‌سازی عامل هوشمند کنترل‌کننده فضاپیما در محیط دینامیکی سه جسمی بررسی شده است. در بخش ۵-۱ به طراحی و در بخش ۵-۲ به شبیه‌سازی عامل هدایت‌کننده مبتنی بر یادگیری تقویتی است پرداخته شده است. این عامل طراحی و شبیه‌سازی شده باید توانایی این را داشته باشد که فضاپیما را به‌طور مؤثر به سمت اهداف تعیین‌شده هدایت کند، در حالی که محدودیت‌هایی نظیر مصرف سوخت و وجود اغتشاش دارد.

۵-۱ طراحی عامل

در این زیربخش، معماری عامل هوشمند کنترل‌کننده فضاپیما در محیط سه‌جسمی شرح داده شده‌است. این معماری شامل تعریف فضای حالت، عمل و تابع پاداش است.

۵-۱-۱ فضای حالت

فضای حالت[1] در این پژوهش به‌گونه‌ای طراحی شده‌است که وضعیت دینامیکی فضاپیما را نسبت به یک مسیر و سرعت مرجع مشخص می‌کند. این فضا شامل اختلاف‌های موقعیت و سرعت از مسیر و سرعت مرجع است و به‌صورت زیر تعریف شده‌است:

$$S = \{\delta x, \delta y, \delta \dot{x}, \delta \dot{y}\}$$

که در آن:

[1] State Space

- $\delta x, \delta y$: اختلاف موقعیت فضاپیما نسبت به مسیر مرجع در محورهای x, y .
- $\delta \dot{x}, \delta \dot{y}$: اختلاف سرعت فضاپیما نسبت به سرعت مرجع در محورهای x, y .

هر یک از این متغیرها به‌طور مستقل وضعیت فضاپیما را در یک جهت خاص توصیف می‌کنند و امکان تحلیل دقیق انحرافات را فراهم می‌سازند. استفاده از اختلاف‌های موقعیت و سرعت به جای مقادیر مطلق، به دلایل زیر انجام شده‌است:

- **تمرکز بر انحرافات:** هدف اصلی سیستم کنترلی، کاهش انحرافات از مسیر و سرعت مطلوب است. با استفاده از اختلاف‌ها، کنترلر می‌تواند به طور مستقیم بر این انحرافات اثر بگذارد و نیازی به محاسبه مقادیر مطلق موقعیت و سرعت ندارد.
- **سازگاری با یادگیری تقویتی:** در الگوریتم‌های یادگیری تقویتی، فضاهای حالت مبتنی بر اختلاف معمولاً دامنه محدودتری دارند که فرآیند یادگیری را سریع‌تر و پایدارتر می‌کند.

۵-۱-۲ فضای عمل

فضای عمل[۲] فضاپیما با پیشران‌کم مجموعه‌ای از عمل‌های پیوسته است که فضاپیما می‌تواند در محیط شبیه‌سازی انجام دهد. این فضا به‌گونه‌ای طراحی شده که امکان اعمال نیرو در جهت‌های مشخص و با مقادیر متناسب با توان واقعی فضاپیماها فراهم شود. به‌طور خاص، فضای اقدام شامل موارد زیر است:

- **نیروی اعمال‌شده در جهت x:** این متغیر پیوسته، مقدار نیرویی را که در جهت محور x به فضاپیما وارد می‌شود، تعیین می‌کند. دامنه این نیرو بر اساس توان پیشرانه‌های موجود در فضاپیماهای واقعی انتخاب شده‌است. به عبارت دیگر، اگر حداکثر نیروی قابل اعمال در جهت x برابر با $f_{x,\max}$ باشد، این متغیر می‌تواند مقادیری در بازه $[-f_{x,\max}, f_{x,\max}]$ داشته باشد.
- **نیروی اعمال‌شده در جهت y:** این متغیر پیوسته، مقدار نیرویی را که در جهت محور y به فضاپیما وارد می‌شود، مشخص می‌کند. مشابه جهت x، دامنه این نیرو نیز بر اساس توان پیشرانه‌های موجود تعیین شده و می‌تواند در بازه $[-f_{y,\max}, f_{y,\max}]$ قرار گیرد.

انتخاب این نیروها بر اساس ویژگی‌های واقعی فضاپیماها، به‌ویژه توان و محدودیت‌های پیشرانه‌های آن‌ها، صورت گرفته است. این امر اطمینان می‌دهد که شبیه‌سازی تا حد ممکن به شرایط واقعی نزدیک باشد و نتایج

[2] Action Space

به‌دست‌آمده قابلیت تعمیم به کاربردهای عملی را داشته باشند. همچنین، تعریف فضای اقدام به‌صورت پیوسته، امکان کنترل دقیق و انعطاف‌پذیر بر حرکت فضاپیما را فراهم می‌کند، که برای دستیابی به اهداف کنترلی در محیط‌های دینامیکی پیچیده ضروری است. به‌طور خلاصه، فضای اقدام به‌صورت زیر تعریف می‌شود:

$$a = \{f_x, f_y \mid f_x \in [-f_{x,\max}, f_{x,\max}],\ f_y \in [-f_{y,\max}, f_{y,\max}]\}$$

انطباق بازه‌ی فضای عمل با داده‌های واقعی

برای هم‌تراز کردن شبیه‌سازی با سخت‌افزارهای واقعی، از بیشینه‌ی نیروی بی‌بُعدِ پیشران‌ها استفاده می‌شود. جدول زیر نمونه‌هایی از فضاپیماهای مجهز به پیشران‌های یونی/الکتریکی را نشان می‌دهد که مبنای انتخاب بازه‌ی نیروی عمل قرارگرفته شده‌اند. با توجه به برداری‌بودن عمل $\boldsymbol{a} = [f_x\ f_y]$، کران‌ها را به دو صورت اعمال شده‌است:

$$|a| \leqslant f_{\text{nondim max}}, \quad \text{یا} \quad f_{x,\max} = f_{y,\max} = f_{\text{nondim max}}.$$

با استناد به جدول ۵-۱، مقدار نمونه‌ی 4×10^{-2} شبیه‌سازی شده با Psyche هم‌مرتبه و کمتر از DS1 است که باعث شده‌است بازه‌ی عمل را در چارچوب پیشران‌های کم‌تراست واقع‌گرایانه نگه داشته شود.

جدول ۵-۱: قابلیت‌های بی‌بعد پیشران‌کم‌تراستِ فضاپیماهای مختلف در سامانه‌ی زمین–ماه [۶۱].

نام اختصار	نام فضاپیما	$f_{\text{max, nondim}}$	$M_{3,0}$ (kg)	$F_{\max}$ (mN)
DS1	Deep Space 1	$6.940 \cdot 10^{-2}$	486.3	92.0
Psyche	Psyche	$4.158 \cdot 10^{-2}$	2464	279.3
Dawn	Dawn	$2.741 \cdot 10^{-2}$	1217.8	91.0
LIC	Lunar IceCube	$3.276 \cdot 10^{-2}$	14	1.25
H1	Hayabusa 1	$1.640 \cdot 10^{-2}$	510	22.8
H2	Hayabusa 2	$1.628 \cdot 10^{-2}$	608.6	27.0
s/c	فضاپیمای نمونه	$4 \cdot 10^{-2}$	–	–

۵-۱-۳ تابع پاداش

تابع پاداش[۳] به‌منظور هدایت رفتار عامل طراحی شده و شامل سه بخش اصلی در طول شبیه‌سازی و یک پاداش نهایی در هنگام پایان است:

- **پاداش نهایی برای دستیابی به هدف:** در صورت رسیدن به مدار هدف، شبیه‌سازی پایان یافته و یک پاداش بزرگ مثبت به عامل داده می‌شود.
- **جریمه نهایی برای دور شدن بیش‌ازحد:** اگر عامل از محدوده مجاز فاصله بگیرد، شبیه‌سازی خاتمه یافته و یک جریمه بزرگ منفی اعمال می‌گردد.
- **جریمه برای مصرف سوخت:** در طول مسیر، استفاده بیش‌ازحد از پیشرانه با جریمه همراه است.
- **جریمه برای انحراف از مسیر مرجع:** در طول مسیر، انحراف از مسیر مرجع باعث دریافت جریمه متناسب می‌شود.

تابع پاداش به‌صورت زیر تعریف می‌شود:

$$r(s,a) = r_{\text{thrust}}(a) + r_{\text{reference}}(s) + r_{\text{terminal}}(s)$$

که در آن مؤلفه‌ها عبارتند از:

$$r_{\text{thrust}}(a) = -k_1 \cdot |a| \tag{۵-۱}$$

$$r_{\text{reference}}(s) = -k_2 \cdot d(s, s_{\text{reference}}) \tag{۵-۲}$$

$$r_{\text{terminal}}(s) = \begin{cases} +R_{\text{goal}} & \text{if } s \in S_{\text{goal}} \\ -R_{\text{fail}} & \text{if } d(s, s_{\text{reference}}) > \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{۵-۳}$$

در این رابطه:

۱. R_{goal}: یک پاداش بزرگ مثبت برای دستیابی به هدف است.

۲. R_{fail}: یک جریمه بزرگ منفی برای خروج از محدوده مجاز است.

۳. $d(s,s')$: فاصله بین دو وضعیت بوده و به‌صورت فاصله اقلیدسی محاسبه می‌شود.

[3] Reward Function

ضرایب k_1, k_2 برای تنظیم تعادل بین بهینه‌سازی مصرف سوخت و حفظ نزدیکی به مسیر مرجع استفاده می‌شوند. انتخاب مناسب مقادیر این ضرایب نقش کلیدی در سرعت همگرایی و پایداری الگوریتم یادگیری تقویتی دارد.

۵-۲ شبیه‌سازی عامل

در این زیربخش، فرآیند شبیه‌سازی و آموزش عامل با استفاده از الگوریتم‌های یادگیری تقویتی پیشرفته ارائه شده‌است. تمرکز بر طراحی شبکه‌ها، منطق انتخاب الگوریتم‌ها، فراپارامترهای کلیدی و ملاحظات پایداری در حین آموزش است تا تکرارپذیری و دقت نتایج تضمین شود.

۵-۲-۱ پارامترهای یادگیری و منطق انتخاب الگوریتم‌ها

الگوریتم‌های DDPG، TD3، SAC و PPO به دلیل کارایی در فضاهای کنش پیوسته و عملکرد پایدار در محیط‌های پیچیده انتخاب شده‌اند. به‌طور خلاصه:

- DDPG: سیاست قطعی با شبکه‌های هدف و میانگین پلیاک؛ مناسب محیط‌های پیوسته با هزینه محاسباتی پایین‌تر، اما حساس به نویز.

جدول ۵-۲: جدول پارامترها و مقادیر پیش‌فرض الگوریتم DDPG [۶۲]

نام پارامتر	مقدار	نام پارامتر	مقدار
گام در هر دوره یادگیری	30 000	تعداد دوره‌های یادگیری	100
اندازه‌ی مخزنِ تجربه	10^6	ضریب تنزیل (γ)	0.99
ضریب میانگین پلیاک	0.995	نرخِ یادگیریِ سیاست	10^{-3}
نرخِ یادگیریِ Q	10^{-3}	اندازه‌ی دسته	1024
گام شروعِ استفاده از سیاست	5 000	گام شروعِ به‌روزرسانی	1 000
فاصله‌ی به‌روزرسانی	2 000	نویز عمل	0.1
حداکثر طولِ رخداد	6 000	دستگاه	Cuda
اندازه شبکه‌ی Actor	$(2^5, 2^5)$	تابع فعال‌سازی Actor	ReLU
اندازه شبکه‌ی Critic	$(2^5, 2^5)$	تابع فعال‌سازی Critic	ReLU

- TD3: بهبود DDPG با دو Critic، هموارسازی سیاست هدف و به‌روزرسانی تأخیری سیاست؛ کاهش بیش‌براوردی Q و پایداری بیشتر.

جدول ۵-۳: جدول پارامترها و مقادیر پیش‌فرض الگوریتم TD3 [۶۲]

نام پارامتر	مقدار	نام پارامتر	مقدار
گام در هر دوره یادگیری	30 000	تعداد دوره‌های یادگیری	100
اندازه‌ی مخزنِ تجربه	10^6	ضریب تنزیل (γ)	0.99
ضریب میانگین پلیاک	0.995	نرخ یادگیریِ سیاست	10^{-3}
نرخ یادگیریِ Q	10^{-3}	اندازه‌ی دسته	1024
گام شروعِ استفاده از سیاست	5 000	گام شروعِ به‌روزرسانی	1 000
فاصله‌ی به‌روزرسانی	2 000	نویز عمل	0.1
نویز هدف	0.2	برش نویز	0.5
تأخیر در به‌روزرسانی سیاست	2	حداکثر طولِ رخداد	30 000
اندازه شبکه‌ی Actor	$(2^5, 2^5)$	تابع فعال‌سازی Actor	ReLU
اندازه شبکه‌ی Critic	$(2^5, 2^5)$	تابع فعال‌سازی Critic	ReLU

- SAC: سیاست تصادفی بیشینه‌ساز آنتروپی با دمای α؛ کاوش مؤثرتر و همگرایی پایدارتر در محیط‌های نویزی.

جدول ۵-۴: جدول پارامترها و مقادیر پیش‌فرض الگوریتم SAC [۶۲]

نام پارامتر	مقدار	نام پارامتر	مقدار
گام در هر دوره یادگیری	30 000	تعداد دوره‌های یادگیری	100
اندازه‌ی مخزنِ تجربه	10^6	ضریب تنزیل (γ)	0.99
ضریب میانگین پلیاک	0.995	نرخِ یادگیری	10^{-3}
نرخِ دمای آلفا	0.2	اندازه‌ی دسته	1024
گام شروعِ استفاده از سیاست	5 000	گام شروعِ به‌روزرسانی	1 000
تعداد به‌روزرسانی در هر مرحله	10	فاصله‌ی به‌روزرسانی	2 000
تعداد اپیزودهای آزمون	10	حداکثر طولِ رخداد	30 000
اندازه شبکه‌ی Actor	$(2^5, 2^5)$	تابع فعال‌سازی Actor	ReLU
اندازه شبکه‌ی Critic	$(2^5, 2^5)$	تابع فعال‌سازی Critic	ReLU

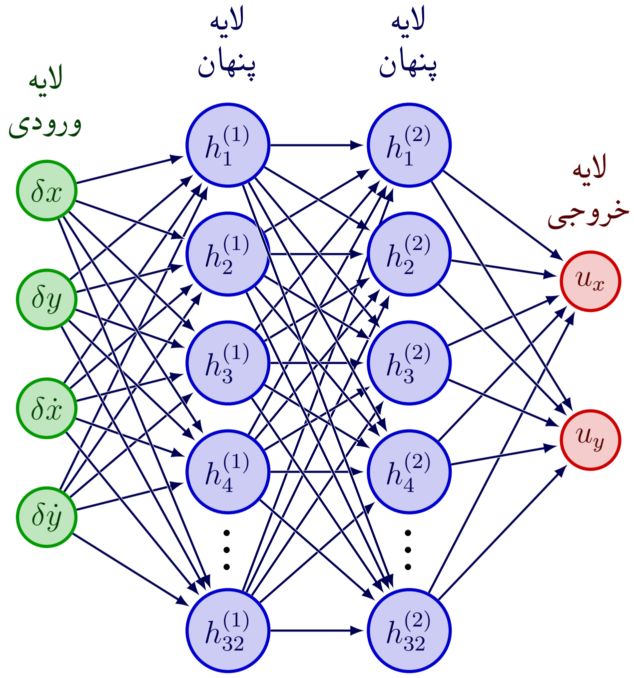
• PPO: روش مبتنی بر سیاست با برش نسبت احتمال؛ به‌روزرسانی‌های ایمن و پیاده‌سازی ساده با کارایی تجربی بالا.

جدول ۵-۵: جدول پارامترها و مقادیر پیش‌فرض الگوریتم PPO [۶۲]

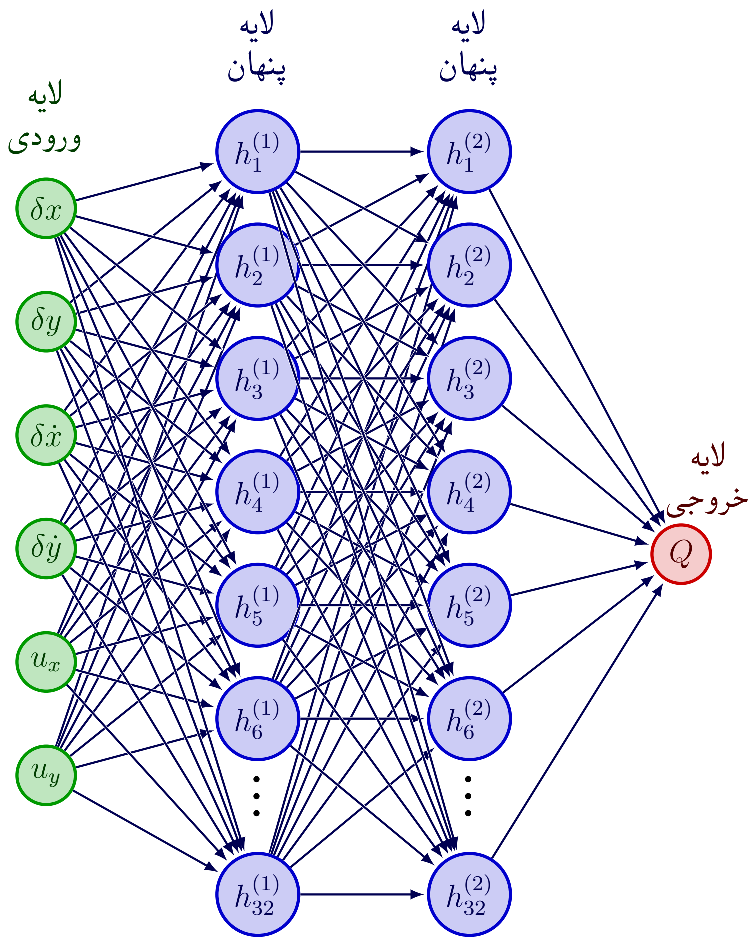
نام پارامتر	مقدار	نام پارامتر	مقدار
گام در هر دوره یادگیری	30 000	تعداد دوره‌های یادگیری	100
ضریب تنزیل (γ)	0.99	ضریب برش ratio clip	0.2
نرخِ یادگیریِ سیاست	3×10^{-4}	نرخِ یادگیریِ تابع ارزش	10^{-3}
تعداد تکرار آموزش سیاست	80	تعداد تکرار آموزش ارزش	80
اندازه شبکه‌ی Actor	$(2^5, 2^5)$	تابع فعال‌سازی Actor	ReLU
اندازه شبکه‌ی Critic	$(2^5, 2^5)$	تابع فعال‌سازی Critic	ReLU

این الگوریتم‌ها به دلیل توانایی در مدیریت فضاهای پیوسته و عملکرد مؤثر در محیط‌های پیچیده انتخاب شده‌اند.

در شکل‌های ۵-۱ و ۵-۲ ساختار شبکه‌های Actor و Critic آورده شده‌است.



شکل ۵-۱: ساختار شبکه عصبی سیاست



شکل ۵-۲: ساختار شبکه عصبی نقاد

۵-۲-۲ فرآیند آموزش

رویه آموزش با PyTorch و اجرای Cuda به‌صورت زیر انجام شده‌است:

۱. گردآوری تجربه‌ی اولیه با سیاست تصادفی تا رسیدن به گام شروع به‌روزرسانی برای پرشدن اولیه‌ی مخزن تجربه.

۲. حلقه‌ی یادگیری: در هر گام، اجرای کنش، ذخیره‌ی چهارتایی‌ها (s, a, r, s') (و در صورت نیاز d برای پایان اپیزود) در مخزن تجربه با ظرفیت 10^6.

۳. نمونه‌گیری دسته داده و به‌روزرسانی Criticها با هدف‌های حاوی شبکه‌های هدف و میانگین پلیاک؛ در TD3 استفاده از دو شبکه Q مستقل و هدف‌های کمینه‌شده.

۴. به‌روزرسانی Actor: در TD3/DDPG بیشینه‌سازی $\mathbb{E}_s[Q(s, \pi_\theta(s))]$ و در SAC بیشینه‌سازی بازگشت

انتروپی‌دار؛ در PPO به‌روزرسانی برش‌خورده با نسبت احتمال.

۵. تکنیک‌های پایداری: Target networks با پلیاک، reward/observation normalization، هموارسازی هدف TD3، gradient clipping در صورت نیاز، و بذردهی ثابت برای تکرارپذیری.

۶. ارزیابی دوره‌ای: اجرای چند اپیزود آزمون بدون نویز کنش و ثبت بازگشت، نرخ موفقیت و واریانس.

برای جلوگیری از بیش‌برازش و همگرایی زودرس، از نویز کاوش کنش و هموارسازی سیاست هدف (در TD3) استفاده شده‌است. معیار توقف زمانی فعال می‌شود که نرخ موفقیت آزمون در چند پنجره‌ی پیاپی از ۹۰٪ عبور کند و واریانس بازگشت کاهش یابد.

بهینه‌سازی و پس‌انتشار گرادیان

محاسبه‌ی گرادیان‌ها با autograd انجام شده‌است. به‌روزرسانی پارامترها با Adam [۶۳] بوده است که در عمل نسبت به گرادیان نزولی ساده پایدارتر است:

$$\begin{aligned} &g_t = \nabla_w L_t, \quad m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \quad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ &\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}, \quad w_{t+1} = w_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon} \end{aligned} \tag{۵-۴}$$

که در آن η نرخ یادگیری، β_1, β_2 ضرایب مومنتوم $(0.9, 0.999)$ و ϵ برای پایدارسازی عددی است. به‌صورت مفهومی، زنجیره گرادیان نیز برقرار است:

$$\nabla_w L = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w} \tag{۵-۵}$$

در این رابطه:

- L_t: مقدار تابع هزینه (Loss) در گام زمانی t.
- w_t: بردار وزن‌ها یا پارامترهای مدل در گام t.
- $g_t = \nabla_w L_t$: گرادیان تابع هزینه نسبت به پارامترها در زمان t.
- m_t: میانگین نمایی گرادیان‌ها (مومنتوم مرتبه اول) که حافظه‌ای از جهت گرادیان‌ها ایجاد می‌کند.
- v_t: میانگین نمایی مربعات گرادیان‌ها (مومنتوم مرتبه دوم) که بزرگی تغییرات گرادیان را ثبت می‌کند.
- $\hat{m}_t, \hat{v}_t$: نسخه‌های اصلاح‌شده‌ی بایاس برای m_t و v_t به‌منظور پایدارسازی در مراحل اولیه.

- η: نرخ یادگیری (Learning Rate) که اندازه‌ی گام به‌روزرسانی وزن‌ها را مشخص می‌کند.

- β_1, β_2: ضرایب کاهش (Decay Rates) برای میانگین‌گیری نمایی؛ مقادیر معمول آن‌ها به‌ترتیب 0.9 و 0.999 است.

- ϵ: یک مقدار بسیار کوچک (معمولاً 10^{-8}) برای جلوگیری از تقسیم بر صفر و افزایش پایداری عددی.

الگوریتم Adam به این صورت عمل می‌کند که همزمان از میانگین مرتبه‌ی اول (m_t) برای جهت حرکت و از میانگین مرتبه‌ی دوم (v_t) برای تنظیم نرخ یادگیری هر پارامتر استفاده می‌کند. در نتیجه هم از نوسانات شدید جلوگیری می‌شود و هم فرآیند همگرایی سرعت می‌گیرد.

از دیدگاه محاسبه‌ی گرادیان، زنجیره‌ی مشتق‌گیری (قاعده‌ی زنجیره‌ای) نیز برقرار است:

$$\nabla_w L = \frac{\partial L}{\partial y} \frac{\partial y}{\partial w} \tag{۵-۶}$$

که در آن y خروجی لایه یا شبکه است. این فرمول مبنای پس‌انتشار خطا (Backpropagation) در شبکه‌های عصبی محسوب می‌شود و باعث می‌گردد که گرادیان تابع هزینه نسبت به تمامی پارامترها به‌صورت کارآمد محاسبه شود.

فصل ۶

یادگیری تقویتی چندعاملی

کاربردهای پیچیده در یادگیری تقویتی نیازمند اضافه کردن چندین عامل[1] برای انجام همزمان وظایف مختلف هستند. با این حال، افزایش تعداد عامل‌ها چالش‌هایی در مدیریت تعاملات میان آن‌ها به همراه دارد. در این فصل، بر اساس مسئله بهینه‌سازی برای هر عامل، مفهوم تعادل نش[2] معرفی شده تا رفتارهای توزیعی چندعاملی را تنظیم کند. رابطه رقابت میان عامل‌ها در سناریوهای مختلف تحلیل شده و آن‌ها با الگوریتم‌های معمول یادگیری تقویتی چندعاملی ترکیب شده‌اند. بر اساس انواع تعاملات، یک چارچوب نظریه بازی برای مدل‌سازی عمومی در سناریوهای چندعاملی استفاده شده است. با تحلیل بهینه‌سازی و وضعیت تعادل برای هر بخش از چارچوب، سیاست بهینه یادگیری تقویتی چندعاملی برای هر عامل بررسی شده است. در این فصل ابتدا در بخش ۶-۱ مفاهیم اولیه‌ی یادگیری تقویتی چندعاملی معرفی می‌شوند، سپس در بخش ۶-۲ انواع بازی‌ها و تعادل نش مورد بررسی قرار می‌گیرند. الگوریتم‌های مختلف یادگیری تقویتی چندعاملی شامل MA-DDPG در بخش ۶-۳، MA-TD3 در بخش ۶-۴، MA-SAC در بخش ۶-۵ و MA-PPO در بخش ۶-۶ معرفی و بررسی شده‌اند.

۶-۱ تعاریف و مفاهیم اساسی

یادگیری تقویتی چندعاملی[3] به بررسی چگونگی یادگیری و تصمیم‌گیری چندین عامل مستقل در یک محیط مشترک پرداخته می‌شود. مفاهیم پایه‌ای یادگیری تقویتی در بخش ۴-۱ ارائه شده‌اند و در اینجا تنها مباحث کلی و موردنیاز برای MARL بیان می‌شوند. برای تحلیل دقیق و درک بهتر این حوزه، اجزای اصلی آن شامل

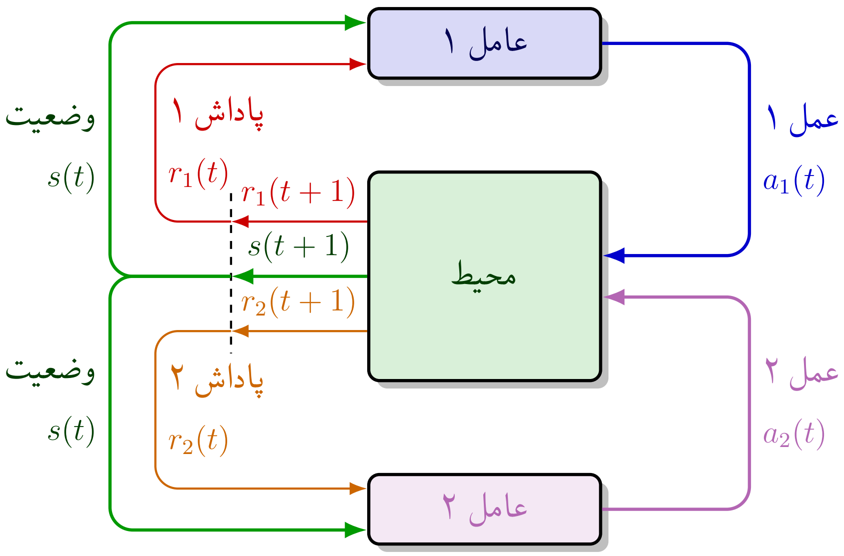
[1] Multi-Agent
[2] Nash Equilibrium
[3] Multi-Agent Reinforcement Learning (MARL)

عامل، سیاست و مطلوبیت[۴] در نظر گرفته می‌شوند که در ادامه به صورت مختصر و منسجم تشریح می‌گردند.

- عامل: یک موجودیت مستقل به عنوان عامل تعریف می‌شود که به صورت خودمختار با محیط تعامل کرده و بر اساس مشاهدات رفتار سایر عامل‌ها، سیاست‌هایش انتخاب می‌گردند تا سود حداکثر یا ضرر حداقل حاصل شود. در سناریوهای مورد بررسی، چندین عامل به صورت مستقل عمل می‌کنند؛ اما اگر تعداد عامل‌ها به یک کاهش یابد، MARL به یادگیری تقویتی معمولی تبدیل می‌شود.
- سیاست: برای هر عامل در MARL، سیاستی خاص در نظر گرفته می‌شود که به عنوان روشی برای انتخاب اقدامات بر اساس وضعیت محیط و رفتار سایر عامل‌ها تعریف می‌گردد. این سیاست‌ها با هدف به حداکثر رساندن سود و به حداقل رساندن هزینه طراحی شده و تحت تأثیر محیط و سیاست‌های دیگر عامل‌ها قرار می‌گیرند.
- مطلوبیت: مطلوبیت هر عامل بر اساس نیازها و وابستگی‌هایش به محیط و سایر عامل‌ها تعریف شده و به صورت سود منهای هزینه، با توجه به اهداف مختلف محاسبه می‌شود. در سناریوهای چندعاملی، از طریق یادگیری از محیط و تعامل با دیگران، مطلوبیت هر عامل بهینه می‌گردد.

در این چارچوب، برای هر عامل در MARL تابع مطلوبیت خاصی در نظر گرفته شده و بر اساس مشاهدات و تجربیات حاصل از تعاملات، یادگیری سیاست به صورت مستقل انجام می‌شود تا ارزش مطلوبیت به حداکثر برسد، بدون اینکه مستقیماً به مطلوبیت سایر عامل‌ها توجه شود. این فرآیند ممکن است به رقابت یا همکاری میان عامل‌ها منجر گردد. با توجه به پیچیدگی تعاملات میان چندین عامل، تحلیل نظریه بازی‌ها به عنوان ابزاری مؤثر برای تصمیم‌گیری در این حوزه به کار گرفته می‌شود.

[4]Utility



شکل ۶-۱: حلقه تعامل عامل‌های یادگیری تقویتی چند عاملی با محیط

۶-۲ نظریه بازی‌ها

نظریه بازی‌ها شاخه‌ای از ریاضیات است که به مطالعه تصمیم‌گیری در موقعیت‌هایی می‌پردازد که نتیجه انتخاب‌های هر فرد به تصمیمات دیگران وابسته است. این نظریه چارچوبی برای تحلیل تعاملات میان بازیکنان ارائه می‌دهد و در حوزه‌های مختلفی مانند اقتصاد، علوم سیاسی، زیست‌شناسی و علوم کامپیوتر کاربرد دارد. در این بخش، دو مفهوم کلیدی نظریه‌ی بازی‌ها، یعنی تعادل نش و بازی‌های مجموع‌صفر، بررسی می‌شوند.

۶-۲-۱ تعادل نش

تعادل نش[5] یکی از بنیادی‌ترین مفاهیم در نظریه‌ی بازی‌ها است که توسط جان نش در سال ۱۹۵۰ معرفی شد. این مفهوم به ترکیب[6] سیاست‌ها اشاره دارد که در آن هیچ بازیکنی نمی‌تواند با تغییر یک‌جانبه‌ی سیاست خود، سود بیشتری به دست آورد (در حالی‌که سیاست‌های سایر بازیکنان ثابت است).

- **تعریف تعادل نش:** فرض کنید یک بازی با n بازیکن داریم. هر بازیکن i دارای مجموعه‌ی سیاست‌های Π_i و تابع مطلوبیت $u_i : \Pi_1 \times \Pi_2 \times \cdots \times \Pi_n \to \mathbb{R}$ است. یک ترکیب سیاست $\pi^* = (\pi_1^*, \pi_2^*, \ldots, \pi_n^*)$ تعادل نش نامیده می‌شود اگر برای هر بازیکن i و هر سیاست $\pi_i \in \Pi_i$ در وضعیت

[5] Nash Equilibrium
[6] Profile

s داشته باشیم:

$$u_i(\pi_i^*, \pi_{-i}^*, \boldsymbol{s}) \geqslant u_i(\pi_i, \pi_{-i}^*, \boldsymbol{s}) \qquad (۱-۶)$$

در اینجا، π_{-i}^* نشان‌دهنده‌ی سیاست‌های همه‌ی بازیکنان به جز بازیکن i است. در ادامه‌ی این پژوهش و به‌منظور به‌کارگیری چارچوب نظریه‌ی بازی در یادگیری تقویتی، مطلوبیت هر عامل به‌صورت برابر با تابع ارزش او در حالت $\boldsymbol{s}$ در نظر گرفته می‌شود: $u_i(\pi_i, \pi_{-i}, \boldsymbol{s}) = V_i^{\pi_i, \pi_{-i}}(\boldsymbol{s})$.

۶-۲-۲ بازی مجموع صفر

بازی‌های مجموع صفر[7] دسته‌ای از بازی‌ها هستند که در آن‌ها تابع ارزش یک بازیکن دقیقاً برابر با ضرر بازیکن دیگر است؛ ازاین‌رو، مجموع ارزش‌های همه‌ی بازیکنان در هر وضعیت صفر خواهد بود.

- **تعریف بازی مجموع صفر:**

در یک بازی دو نفره، اگر تابع ارزشِ حالت (value) بازیکن اوّل $V_1^{(\pi_1,\pi_2)}(s)$ و بازیکن دوم $V_2^{(\pi_1,\pi_2)}(s)$ برای هر مجموعه سیاست (π_1, π_2) به‌گونه‌ای باشند که:

$$V_1^{(\pi_1,\pi_2)}(s) + V_2^{(\pi_1,\pi_2)}(s) = 0 \implies V_1^{(\pi_1,\pi_2)}(s) = -V_2^{(\pi_1,\pi_2)}(s), \qquad (۲-۶)$$

آنگاه آن بازی را بازی مجموع صفر می‌نامیم.

به‌طور مشابه، اگر تابع ارزش–عمل برای دو بازیکن را با $Q_1^{(\pi_1,\pi_2)}(s, a_1, a_2)$ و $Q_2^{(\pi_1,\pi_2)}(s, a_1, a_2)$ نشان دهیم، باید برقرار باشد:

(۳-۶)

$$Q_1^{(\pi_1,\pi_2)}(s, a_1, a_2) + Q_2^{(\pi_1,\pi_2)}(s, a_1, a_2) = 0 \implies Q_1^{(\pi_1,\pi_2)}(s, a_1, a_2) = -Q_2^{(\pi_1,\pi_2)}(s, a_1, a_2).$$

- **سیاست بهینه در بازی مجموع صفر:**

در این بازی‌ها، هر بازیکن سیاستی را برمی‌گزیند که تابع ارزش خود را در برابر بهترین پاسخِ حریف بیشینه کند؛ این انتخاب در نهایت به تعادل نش منجر می‌شود.

به‌صورت تابع ارزشِ حالت:

$$V_1^*(s) = \max_{\pi_1} \min_{\pi_2} V_1^{(\pi_1,\pi_2)}(s), \qquad (۴-۶)$$

$$V_2^*(s) = \max_{\pi_2} \min_{\pi_1} V_2^{(\pi_1,\pi_2)}(s). \qquad (۵-۶)$$

[7] Zero-Sum Games

و به‌صورت تابع ارزش–عمل:

$$Q_1^*(s,a_1,a_2) = \max_{\pi_1} \min_{\pi_2} \ Q_1^{(\pi_1,\pi_2)}(s,a_1,a_2), \tag{۶-۶}$$

$$Q_2^*(s,a_1,a_2) = \max_{\pi_2} \min_{\pi_1} \ Q_2^{(\pi_1,\pi_2)}(s,a_1,a_2). \tag{۶-۷}$$

- **تابع پاداش:** تابع پاداش در بازی‌های دوسویه مجموع‌صفر باید به‌گونه‌ای طراحی شود که پاداش لحظه‌ای دو عامل در هر گام جمعاً صفر باشد. در ادامه ساختار پاداش عاملِ ۱ مشابه قالب تک‌عاملی تعریف می‌شود و پاداش عاملِ ۲ به‌صورت منفی آن اخذ می‌گردد.

  - **پاداش نهایی برای دستیابی به هدفِ عامل ۱:** در صورت رسیدن به هدف عامل ۱، شبیه‌سازی پایان یافته و پاداش بزرگ مثبت به او داده می‌شود.
  - **جریمه نهایی برای دور شدنِ عامل ۱:** اگر عامل ۱ از محدوده مجاز خود خارج شود، شبیه‌سازی خاتمه یافته و جریمه بزرگ منفی اعمال می‌گردد.
  - **جریمه برای مصرف سوختِ عامل ۱:** استفاده بیش‌ازحد از پیشرانه برای عامل ۱ با جریمه همراه است.
  - **جریمه برای انحراف از مسیر مرجعِ عامل ۱:** انحراف از مسیر مرجع عامل ۱ باعث دریافت جریمه متناسب می‌شود.

تابع پاداش عامل ۱ به‌صورت زیر تعریف می‌شود:

$$r_1(s,a_1,a_2) = r_{\text{thrust},1}(a_1) + r_{\text{thrust},1}(a_2) + r_{\text{reference},1}(s) + r_{\text{terminal},1}(s)$$

که در آن مؤلفه‌ها عبارتند از:

$$r_{\text{thrust},1}(a_1) = -k_1 \cdot |a_1| \tag{۶-۸}$$

$$r_{\text{thrust},1}(a_2) = -k_2 \cdot |a_2| \tag{۶-۹}$$

$$r_{\text{reference},1}(s) = -k_3 \cdot d_1\big(s, s_{\text{ref},1}\big) \tag{۶-۱۰}$$

$$r_{\text{terminal},1}(s) = \begin{cases} +R_{\text{goal},1} & \text{if } s \in S_{\text{goal},1} \\ -R_{\text{fail},1} & \text{if } d_1\big(s, s_{\text{ref},1}\big) > \epsilon_1 \\ 0 & \text{otherwise} \end{cases} \tag{۶-۱۱}$$

برای تضمین خاصیت مجموع‌صفر، پاداشِ عامل ۲ را در هر گام به‌صورت زیر تعریف می‌کنیم:

$$r_2(s, a_1, a_2) = -\, r_1(s, a_1, a_2),$$

بنابراین با افق و ضریب تنزیل یکسان، روابط (۶-۲) و (۶-۳) نیز برقرار خواهند بود.

در این رابطه:

۱. $R_{\text{goal},1}$: پاداش بزرگ مثبت برای دستیابی عامل ۱ به هدف.

۲. $R_{\text{fail},1}$: جریمه بزرگ منفی برای خروج عامل ۱ از محدوده مجاز.

۳. $d_1(s, s')$: فاصله مرتبط با عامل ۱ اقلیدسی بین دو وضعیت.

ضرایب k_1, k_2, k_3 برای تنظیم تعادل بین جریمهٔ پیشرانهٔ عامل ۱، جریمهٔ پیشرانهٔ عامل ۲، و جریمهٔ انحراف از مسیر مرجع استفاده می‌شوند. به‌دلیل تعریف $r_2 = -r_1$، جمع پاداش‌ها در هر گام صفر بوده و مقدار بازی یکتا و با تعادل نش در راهبردهای مختلط سازگار است.

بر پایه‌ی قضیه‌ی کمینه‌بیشینه‌ی فون‌نویمان، در بازی‌های دوسویه‌ی مجموع‌صفرِ متناهی داریم:

$$\max_{\pi_1} \min_{\pi_2} V_1^{(\pi_1,\pi_2)}(s) = \min_{\pi_2} \max_{\pi_1} V_1^{(\pi_1,\pi_2)}(s),$$

که وجود تعادل نش در راهبردهای مختلط و یکتایی مقدار بازی را تضمین می‌کند.

۶-۳ گرادیان سیاست عمیق قطعی چندعاملی

گرادیان سیاست عمیق قطعی چندعاملی[۸] توسعه‌ای از الگوریتم DDPG برای محیط‌های چندعاملی است. در این بخش، به بررسی این الگوریتم در چارچوب بازی‌های دوعاملیِ مجموع‌صفر می‌پردازیم که در آن مجموع پاداش‌های دو عامل همواره صفر است (آنچه یک عامل به دست می‌آورد، عامل دیگر از دست می‌دهد).

۶-۳-۱ چالش‌های یادگیری تقویتی در محیط‌های چندعاملی

در محیط‌های چندعاملی، سیاست هر عامل مدام در حال تغییر است، که باعث می‌شود محیط از دید هر عامل غیرایستا[۹] شود. این مسئله چالش بزرگی برای الگوریتم‌های یادگیری تقویتی تک‌عاملی مانند DDPG ایجاد می‌کند، زیرا فرض ایستایی محیط را نقض می‌کند.

[8] Multi-Agent Deep Deterministic Policy Gradient (MA-DDPG)
[9] Non-stationary

MA-DDPG با استفاده از رویکرد آموزش متمرکز، اجرای غیرمتمرکز[10] این مشکل را حل می‌کند. در این رویکرد، هر عامل در زمان آموزش به اطلاعات کامل محیط دسترسی دارد، اما در زمان اجرا تنها از مشاهدات محلی خود استفاده می‌کند.

۶-۳-۲ معماری MA-DDPG در بازی‌های مجموع‌صفر

در یک بازی دوعاملیِ مجموع‌صفر، دو عامل با نمادهای ۱ و ۲ نشان داده می‌شوند. هر عامل دارای شبکه‌های منحصر به فرد خود است:

- **شبکه‌های بازیگر:** $\mu_{\theta_1}(o_1)$ و $\mu_{\theta_2}(o_2)$ که مشاهدات محلی o_1 و o_2 را به اعمال a_1 و a_2 نگاشت می‌کنند.
- **شبکه‌های منتقد:** $Q_{\phi_1}(o_1, a_1, a_2)$ و $Q_{\phi_2}(o_2, a_2, a_1)$ که ارزش حالت–عمل را با توجه به مشاهدات و اعمال تمام عامل‌ها تخمین می‌زنند.
- **شبکه‌های هدف:** مشابه DDPG، برای پایدارکردن آموزش از شبکه‌های هدف استفاده می‌شود.

در بازی‌های مجموع‌صفر، پاداش‌ها رابطه $r_1 + r_2 = 0$ دارند که در آن r_1 و r_2 پاداش‌های دریافتی عامل‌ها هستند. در نتیجه، $r_2 = -r_1$ است که نمایانگر تضاد کامل منافع بین عامل‌هاست.

۶-۳-۳ آموزش MA-DDPG در بازی‌های مجموع‌صفر

فرایند آموزش MA-DDPG برای بازی‌های مجموع‌صفر به شرح زیر است:

یادگیری تابع Q

برای هر عامل $i \in \{1, 2\}$، تابع Q با کمینه کردن خطای میانگین مربعات بلمن به‌روزرسانی می‌شود:

$$L(\phi_i, \mathcal{D}) = \underset{(\boldsymbol{o},\boldsymbol{a},r_i,\boldsymbol{o}',d)\sim\mathcal{D}}{\mathrm{E}}\left[\left(Q_{\phi_i}(o_i, a_1, a_2) - y_i\right)^2\right] \tag{۶-۱۲}$$

که در آن $\boldsymbol{o} = (o_1, o_2)$ بردار مشاهدات، $\boldsymbol{a} = (a_1, a_2)$ بردار اعمال، و y_i هدف برای عامل i است:

$$y_i = r_i + \gamma(1-d)Q_{\phi_{i,\text{targ}}}(o_i', \mu_{\theta_{1,\text{targ}}}(o_1'), \mu_{\theta_{2,\text{targ}}}(o_2')) \tag{۶-۱۳}$$

[10] Centralized Training, Decentralized Execution

در این پژوهش منتقد هر عامل به اعمال همه عامل‌ها دسترسی دارد. در بازی‌های مجموع‌صفر، عامل شماره ۲ جهت مخالف هدف عامل ۱ را دنبال می‌کند.

یادگیری سیاست

سیاست هر عامل با بیشینه کردن تابع Q مربوط به آن عامل به‌روزرسانی می‌شود:

$$\max_{\theta_i} \mathop{\mathrm{E}}_{\boldsymbol{o}\sim\mathcal{D}} \left[Q_{\phi_i}(o_i, \mu_{\theta_i}(o_i), \mu_{\theta_{-i}}(o_{-i})) \right] \tag{۱۴-۶}$$

که در آن $i-$ نشان‌دهنده‌ی عامل مقابل است. با توجه به ماهیت بازی مجموع‌صفر، هر عامل تلاش می‌کند تا مطلوبیت خود را افزایش دهد، در حالی که مطلوبیت عامل دیگر به‌طور همزمان کاهش می‌یابد.

شبکه‌های هدف و بافر تجربه

مشابه DDPG، برای پایدار کردن آموزش، شبکه‌های هدف با میانگین‌گیری پولیاک به‌روزرسانی می‌شوند:

$$\phi_{i,\text{targ}} \leftarrow \rho\phi_{i,\text{targ}} + (1-\rho)\phi_i$$

$$\theta_{i,\text{targ}} \leftarrow \rho\theta_{i,\text{targ}} + (1-\rho)\theta_i$$

همچنین، از یک بافر تکرار بازی مشترک برای ذخیره تجربیات استفاده می‌شود که شامل وضعیت‌ها، اعمال و پاداش‌های همه عامل‌هاست.

۶-۳-۴ اکتشاف در MA-DDPG

اکتشاف در MA-DDPG مشابه DDPG است، اما برای هر عامل به طور جداگانه اعمال می‌شود. در طی آموزش، به اعمال هر عامل نویز اضافه می‌شود:

$$a_i = \textbf{clip}(\mu_{\theta_i}(o_i) + \epsilon_i, a_{\text{Low}}, a_{\text{High}}) \tag{۱۵-۶}$$

که در آن ϵ_i نویز اضافه شده به عامل i است.

۶-۳-۵ شبکه‌کد MA-DDPG برای بازی‌های دوعاملیِ مجموع‌صفر

در این بخش، شبه‌کد الگوریتم MA-DDPG پیاده‌سازی‌شده آورده شده‌است. در این پژوهش الگوریتم ۵ در محیط پایتون با استفاده از کتابخانه PyTorch [۵۹] پیاده‌سازی شده‌است.

الگوریتم ۵ عامل گرادیان سیاست عمیق قطعی چندعاملی

ورودی: پارامترهای اولیه سیاست عامل‌ها (θ_1, θ_2)، پارامترهای تابع Q (ϕ_1, ϕ_2)، بافر تکرار بازی خالی $(\mathcal{D})$

۱: پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید: $\theta_{i,\text{targ}} \leftarrow \theta_i$، $\phi_{i,\text{targ}} \leftarrow \phi_i$ برای $i \in \{1, 2\}$

۲: تا وقتی همگرایی رخ دهد:

۳: مشاهدات (o_1, o_2) را دریافت کنید

۴: برای هر عامل i، عمل $a_i = \text{clip}(\mu_{\theta_i}(o_i) + \epsilon_i, a_{\text{Low}}, a_{\text{High}})$ را انتخاب کنید، به‌طوری که $\epsilon_i \sim \mathcal{N}$ است

۵: اعمال (a_1, a_2) را در محیط اجرا کنید

۶: مشاهدات بعدی (o'_1, o'_2)، پاداش‌ها $(r_1, r_2 = -r_1)$ و سیگنال پایان d را دریافت کنید

۷: تجربه $(o_1, o_2, a_1, a_2, r_1, r_2, o'_1, o'_2, d)$ را در بافر $\mathcal{D}$ ذخیره کنید

۸: اگر $d = 1$ است، وضعیت محیط را بازنشانی کنید

۹: اگر زمان به‌روزرسانی فرا رسیده است:

۱۰: به ازای هر تعداد به‌روزرسانی:

۱۱: یک دسته تصادفی از تجربیات، $B = \{(\boldsymbol{o}, \boldsymbol{a}, r_1, r_2, \boldsymbol{o}', d)\}$، از $\mathcal{D}$ نمونه‌گیری کنید اهداف را محاسبه کنید:

$$y_1 = r_1 + \gamma(1-d) Q_{\phi_{1,\text{targ}}}(o'_1, \mu_{\theta_{1,\text{targ}}}(o'_1), \mu_{\theta_{2,\text{targ}}}(o'_2))$$

$$y_2 = r_2 + \gamma(1-d) Q_{\phi_{2,\text{targ}}}(o'_2, \mu_{\theta_{2,\text{targ}}}(o'_2), \mu_{\theta_{1,\text{targ}}}(o'_1))$$

۱۲: توابع Q را با نزول گرادیان به‌روزرسانی کنید:

$$\nabla_{\phi_1} \frac{1}{|B|} \sum_{(\boldsymbol{o}, \boldsymbol{a}, r_1, r_2, \boldsymbol{o}', d) \in B} \left(Q_{\phi_1}(o_1, a_1, a_2) - y_1\right)^2$$

$$\nabla_{\phi_2} \frac{1}{|B|} \sum_{(\boldsymbol{o}, \boldsymbol{a}, r_1, r_2, \boldsymbol{o}', d) \in B} \left(Q_{\phi_2}(o_2, a_2, a_1) - y_2\right)^2$$

۱۳: سیاست‌ها را با صعود گرادیان به‌روزرسانی کنید:

$$\nabla_{\theta_1} \frac{1}{|B|} \sum_{\boldsymbol{o} \in B} Q_{\phi_1}(o_1, \mu_{\theta_1}(o_1), a_2)$$

$$\nabla_{\theta_2} \frac{1}{|B|} \sum_{\boldsymbol{o} \in B} Q_{\phi_2}(o_2, \mu_{\theta_2}(o_2), a_1)$$

۱۴: شبکه‌های هدف را به‌روزرسانی کنید:

$$\phi_{1,\text{targ}} \leftarrow \rho \phi_{1,\text{targ}} + (1-\rho)\phi_1$$

$$\phi_{2,\text{targ}} \leftarrow \rho \phi_{2,\text{targ}} + (1-\rho)\phi_2$$

$$\theta_{1,\text{targ}} \leftarrow \rho \theta_{1,\text{targ}} + (1-\rho)\theta_1$$

$$\theta_{2,\text{targ}} \leftarrow \rho \theta_{2,\text{targ}} + (1-\rho)\theta_2$$

۶-۳-۶ مزایای MA-DDPG در بازی‌های مجموع‌صفر

MA-DDPG چندین مزیت برای یادگیری در بازی‌های دوعاملیِ مجموع‌صفر ارائه می‌دهد:

- **مقابله با غیرایستایی:** با استفاده از منتقدهایی که به اطلاعات کامل دسترسی دارند، مشکل غیرایستایی محیط از دید هر عامل حل می‌شود.
- **همگرایی بهتر:** در بازی‌های مجموع‌صفر، MA-DDPG معمولاً همگرایی بهتری نسبت به آموزش مستقل عامل‌ها با DDPG نشان می‌دهد.
- **یادگیری استراتژی‌های متقابل:** عامل‌ها می‌توانند استراتژی‌های متقابل پیچیده را یاد بگیرند که در آموزش مستقل امکان‌پذیر نیست.

در بازی‌های دوعاملیِ مجموع‌صفر، این رویکرد به رقابت کامل بین عامل‌ها منجر می‌شود، که هر یک تلاش می‌کند بهترین استراتژی را در برابر استراتژی رقیب پیدا کند.

۶-۴ عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه چندعاملی

عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه چندعاملی[11] توسعه‌ای از الگوریتم TD3 برای محیط‌های چندعاملی است. در این بخش، به بررسی این الگوریتم در چارچوب بازی‌های چندعاملیِ مجموع‌صفر می‌پردازیم که در آن ترکیب ویژگی‌های TD3 با رویکرد چندعاملی MA-DDPG به پایداری و کارایی بیشتر در یادگیری منجر می‌شود.

۶-۴-۱ چالش‌های یادگیری تقویتی در محیط‌های چندعاملی و راه‌حل MA-TD3

در محیط‌های چندعاملی، عامل‌ها همزمان سیاست‌های خود را تغییر می‌دهند که باعث غیرایستایی محیط از دید هر عامل می‌شود. علاوه بر این، بیش‌برآورد تابع Q که در DDPG دیده می‌شود، در محیط‌های چندعاملی می‌تواند تشدید شود.

MA-TD3 هر دو چالش را با ترکیب رویکردهای زیر حل می‌کند:

- **آموزش متمرکز، اجرای غیرمتمرکز:** مشابه MA-DDPG، از منتقدهایی استفاده می‌کند که به اطلاعات کامل دسترسی دارند.

[11] Multi-Agent Twin Delayed Deep Deterministic Policy Gradient (MA-TD3)

- **منتقدهای دوگانه:** برای هر عامل، از دو شبکه منتقد استفاده می‌کند تا بیش‌برآورد تابع Q را کاهش دهد.

- **به‌روزرسانی‌های تاخیری سیاست:** سیاست‌ها را با تواتر کمتری نسبت به منتقدها به‌روزرسانی می‌کند.

۶-۴-۲ معماری MA-TD3 در بازی‌های مجموع‌صفر

در یک بازی چندعاملیِ مجموع‌صفر، هر عامل دارای شبکه‌های زیر است:

- **شبکه بازیگر:** $\mu_{\theta_i}(o_i)$ که مشاهدات محلی o_i را به اعمال a_i نگاشت می‌کند.
- **شبکه‌های منتقد دوگانه:** $Q_{\phi_{i,1}}(o_i, a_1, a_2)$ و $Q_{\phi_{i,2}}(o_i, a_1, a_2)$ که ارزش حالت–عمل را تخمین می‌زنند.
- **شبکه‌های هدف:** برای پایدارسازی آموزش، از نسخه‌های هدف بازیگر و منتقدها استفاده می‌شود.

۶-۴-۳ آموزش MA-TD3

فرایند آموزش MA-TD3 به شرح زیر است:

یادگیری تابع Q

برای هر عامل $i \in \{1,2\}$ و هر منتقد $j \in \{1,2\}$، تابع Q با کمینه کردن خطای میانگین مربعات بلمن به‌روزرسانی می‌شود:

$$L(\phi_{i,j}, \mathcal{D}) = \underset{(\boldsymbol{o},\boldsymbol{a},r_i,\boldsymbol{o}',d)\sim\mathcal{D}}{\mathrm{E}}\left[\left(Q_{\phi_{i,j}}(o_i, a_1, a_2) - y_i\right)^2\right] \tag{۶-۱۶}$$

که در آن y_i هدف برای عامل i است:

$$y_i = r_i + \gamma(1-d)\min_{j=1,2} Q_{\phi_{i,j,\text{targ}}}(o_i', \mu_{\theta_{1,\text{targ}}}(o_1'), \mu_{\theta_{2,\text{targ}}}(o_2')) \tag{۶-۱۷}$$

استفاده از عملگر حداقل روی دو منتقد، بیش‌برآورد را کاهش می‌دهد که منجر به تخمین‌های محتاطانه‌تر و پایدارتر می‌شود.

یادگیری سیاست با تاخیر

سیاست هر عامل با تاخیر (معمولاً پس از هر دو به‌روزرسانی منتقدها) و با بیشینه کردن تابع Q اول به‌روزرسانی می‌شود:

$$\max_{\theta_i} \mathop{\mathrm{E}}_{\boldsymbol{o}\sim\mathcal{D}} \left[Q_{\phi_{i,1}}\left(o_i, \mu_{\theta_i}(o_i), \mu_{\theta_{-i}}(o_{-i})\right)\right] \tag{۱۸-۶}$$

به‌روزرسانی تاخیری سیاست اجازه می‌دهد تا منتقدها قبل از تغییر سیاست به مقادیر دقیق‌تری همگرا شوند.

شبکه‌های هدف

مشابه TD3، شبکه‌های هدف با میانگین‌گیری پولیاک به‌روزرسانی می‌شوند.

۶-۴-۴ اکتشاف در MA-TD3

اکتشاف در MA-TD3 با افزودن نویز به اعمال هر عامل انجام می‌شود:

$$a_i = \mathrm{clip}(\mu_{\theta_i}(o_i) + \epsilon_i, a_{\mathrm{Low}}, a_{\mathrm{High}}) \tag{۱۹-۶}$$

که در آن $\epsilon_i \sim \mathcal{N}(0, \sigma_i)$ است و مقدار σ_i به مرور زمان کاهش می‌یابد.

۶-۴-۵ شبه‌کد MA-TD3 برای بازی‌های چندعاملیِ مجموع‌صفر

در این بخش، شبه‌کد الگوریتم MA-TD3 پیاده‌سازی‌شده آورده شده‌است. در این پژوهش الگوریتم ۶ در محیط پایتون با استفاده از کتابخانه PyTorch [۵۹] پیاده‌سازی شده‌است.

الگوریتم ۶ عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه چندعاملی

ورودی: پارامترهای اولیه سیاست عامل‌ها (θ_1, θ_2)، پارامترهای توابع Q $(\phi_{1,1}, \phi_{1,2}, \phi_{2,1}, \phi_{2,2})$، بافر تکرار بازی خالی $(\mathcal{D})$

پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید:

$\theta_{i,\text{targ}} \leftarrow \theta_i$، $\phi_{i,j,\text{targ}} \leftarrow \phi_{i,j}$ برای $i \in \{1,2\}$ و $j \in \{1,2\}$

تا وقتی همگرایی رخ دهد:

مشاهدات (o_1, o_2) را دریافت کنید

برای هر عامل i، عمل $a_i = \textbf{clip}(\mu_{\theta_i}(o_i) + \epsilon_i, a_{\text{Low}}, a_{\text{High}})$ را انتخاب کنید، به‌طوری که $\epsilon_i \sim \mathcal{N}(0, \sigma_i)$ است

اعمال (a_1, a_2) را در محیط اجرا کنید

مشاهدات بعدی (o'_1, o'_2)، پاداش‌ها $(r_1, r_2 = -r_1)$ و سیگنال پایان d را دریافت کنید

تجربه $(o_1, o_2, a_1, a_2, r_1, r_2, o'_1, o'_2, d)$ را در بافر $\mathcal{D}$ ذخیره کنید

اگر $d = 1$ است، وضعیت محیط را بازنشانی کنید

اگر زمان به‌روزرسانی فرا رسیده است:

به ازای j در هر تعداد به‌روزرسانی:

یک دسته تصادفی از تجربیات، $B = \{(\boldsymbol{o}, \boldsymbol{a}, r_1, r_2, \boldsymbol{o}', d)\}$، از $\mathcal{D}$ نمونه‌گیری کنید.

اهداف را محاسبه کنید:

$$y_1 = r_1 + \gamma(1-d) \min_{k=1,2} Q_{\phi_{1,k,\text{targ}}}(o'_1, \mu_{\theta_{1,\text{targ}}}(o'_1), \mu_{\theta_{2,\text{targ}}}(o'_2))$$

$$y_2 = r_2 + \gamma(1-d) \min_{k=1,2} Q_{\phi_{2,k,\text{targ}}}(o'_2, \mu_{\theta_{2,\text{targ}}}(o'_2), \mu_{\theta_{1,\text{targ}}}(o'_1))$$

توابع Q را با نزول گرادیان به‌روزرسانی کنید:

$$\nabla_{\phi_{1,k}} \frac{1}{|B|} \sum_B \left(Q_{\phi_{1,k}}(o_1, a_1, a_2) - y_1\right)^2 \quad \text{برای} k = 1,2$$

$$\nabla_{\phi_{2,k}} \frac{1}{|B|} \sum_B \left(Q_{\phi_{2,k}}(o_2, a_2, a_1) - y_2\right)^2 \quad \text{برای} k = 1,2$$

اگر باقیمانده j بر تاخیر سیاست برابر 0 باشد:

سیاست‌ها را با صعود گرادیان به‌روزرسانی کنید:

$$\nabla_{\theta_1} \frac{1}{|B|} \sum_{\boldsymbol{o} \in B} Q_{\phi_{1,1}}(o_1, \mu_{\theta_1}(o_1), a_2)$$

$$\nabla_{\theta_2} \frac{1}{|B|} \sum_{\boldsymbol{o} \in B} Q_{\phi_{2,1}}(o_2, \mu_{\theta_2}(o_2), a_1)$$

شبکه‌های هدف را به‌روزرسانی کنید:

$$\phi_{i,k,\text{targ}} \leftarrow \rho\phi_{i,k,\text{targ}} + (1-\rho)\phi_{i,k} \quad \text{برای} i,k \in \{1,2\}$$

$$\theta_{i,\text{targ}} \leftarrow \rho\theta_{i,\text{targ}} + (1-\rho)\theta_i \quad \text{برای} i \in \{1,2\}$$

۶-۴-۶ مزایای MA-TD3 در بازی‌های مجموع‌صفر

MA-TD3 مزایای زیر را نسبت به MA-DDPG در بازی‌های چندعاملیِ مجموع‌صفر ارائه می‌دهد:

- **پایداری بیشتر:** با استفاده از منتقدهای دوگانه، بیش‌برآورد تابع Q که در محیط‌های غیرایستای چند-عاملی شدیدتر است، کاهش می‌یابد.
- **یادگیری کارآمدتر:** به‌روزرسانی‌های تاخیری سیاست اجازه می‌دهد منتقدها به تخمین‌های دقیق‌تری دست یابند، که منجر به بهبود کیفیت یادگیری سیاست می‌شود.
- **مقاومت در برابر نویز:** ترکیب منتقدهای دوگانه با رویکرد آموزش متمرکز، مقاومت الگوریتم در برابر نویز و تغییرات محیط را افزایش می‌دهد.
- **همگرایی بهتر:** بهبودهای TD3 در کنار رویکرد چندعاملی، به همگرایی سریع‌تر و پایدارتر در بازی‌های رقابتی منجر می‌شود.

در مجموع، MA-TD3 ترکیبی از بهترین ویژگی‌های TD3 و MA-DDPG را ارائه می‌دهد که آن را به گزینه‌ای مناسب برای یادگیری سیاست‌های پیچیده در بازی‌های چندعاملیِ مجموع‌صفر تبدیل می‌کند.

۵-۶ عامل عملگر نقاد نرم چندعاملی

عامل عملگر نقاد نرم دوعاملی[12] توسعه‌ای از الگوریتم SAC برای محیط‌های چندعاملی است. در این بخش، به بررسی این الگوریتم در چارچوب بازی‌های چندعاملیِ مجموع‌صفر می‌پردازیم که در آن ترکیب ویژگی‌های SAC با رویکرد چندعاملی به پایداری و کارایی بیشتر در یادگیری منجر می‌شود.

۱-۵-۶ چالش‌های یادگیری تقویتی در محیط‌های چندعاملی و راه‌حل MA-SAC

در محیط‌های چندعاملی، عامل‌ها همزمان سیاست‌های خود را تغییر می‌دهند که باعث غیرایستایی محیط از دید هر عامل می‌شود. علاوه بر این، چالش‌های مربوط به تعادل اکتشاف-بهره‌برداری در محیط‌های چندعاملی پیچیده‌تر است.

MA-SAC این چالش‌ها را با ترکیب رویکردهای زیر حل می‌کند:

[12] Multi-Agent Soft Actor-Critic (MA-SAC)

- **آموزش متمرکز، اجرای غیرمتمرکز:** مشابه MA-DDPG، از منتقدهایی استفاده می‌کند که به اطلاعات کامل دسترسی دارند.

- **سیاست‌های تصادفی:** برخلاف MA-DDPG و MA-TD3 که سیاست‌های قطعی دارند، MA-SAC از سیاست‌های تصادفی استفاده می‌کند.

- **تنظیم آنتروپی:** با استفاده از تنظیم آنتروپی، اکتشاف و همگرایی به سیاست‌های بهتر را بهبود می‌بخشد.

- **منتقدهای دوگانه:** برای هر عامل، از دو شبکه منتقد استفاده می‌کند تا بیش‌برآورد تابع Q را کاهش دهد.

۶-۵-۲ معماری MA-SAC در بازی‌های مجموع‌صفر

در یک بازی چندعاملیِ مجموع‌صفر، هر عامل دارای شبکه‌های زیر است:

- **شبکه بازیگر:** $\pi_{\theta_i}(a_i|o_i)$ که توزیع احتمال اعمال را با توجه به مشاهدات محلی تعیین می‌کند.

- **شبکه‌های منتقد دوگانه:** $Q_{\phi_{i,1}}(o_i, a_1, a_2)$ و $Q_{\phi_{i,2}}(o_i, a_1, a_2)$ که ارزش حالت–عمل را تخمین می‌زنند.

- **شبکه‌های هدف:** برای پایدارسازی آموزش، از نسخه‌های هدف منتقدها استفاده می‌شود.

۶-۵-۳ آموزش MA-SAC

فرایند آموزش MA-SAC به شرح زیر است:

یادگیری تابع Q

برای هر عامل $i \in \{1,2\}$ و هر منتقد $j \in \{1,2\}$، تابع Q با کمینه کردن خطای میانگین مربعات بلمن به‌روزرسانی می‌شود:

$$L(\phi_{i,j}, \mathcal{D}) = \mathop{\mathrm{E}}_{(\boldsymbol{o},\boldsymbol{a},r_i,\boldsymbol{o}',d)\sim\mathcal{D}} \left[\left(Q_{\phi_{i,j}}(o_i, a_1, a_2) - y_i \right)^2 \right] \tag{۶-۲۰}$$

که در آن y_i هدف برای عامل i است:

$$y_i = r_i + \gamma(1-d)\Big(\min_{j=1,2} Q_{\phi_{i,j,\text{targ}}}(o'_i, \tilde{a}'_1, \tilde{a}'_2) - \alpha_i \log \pi_{\theta_i}(\tilde{a}'_i|o'_i)\Big) \tag{۶-۲۱}$$

که در آن $\tilde{a}'_i \sim \pi_{\theta_i}(\cdot|o'_i)$ است. استفاده از عملگر حداقل روی دو منتقد، بیش‌برآورد را کاهش می‌دهد که منجر به تخمین‌های محتاطانه‌تر و پایدارتر می‌شود.

یادگیری سیاست

سیاست هر عامل با بیشینه کردن ترکیبی از تابع Q و آنتروپی به‌روزرسانی می‌شود:

$$\max_{\theta_i} \mathop{\mathrm{E}}_{\boldsymbol{o}\sim\mathcal{D}} \left[\min_{j=1,2} Q_{\phi_{i,j}}(o_i, \tilde{a}_i, a_{-i}) - \alpha_i \log \pi_{\theta_i}(\tilde{a}_i|o_i)\right] \tag{۶-۲۲}$$

که در آن $\tilde{a}_i \sim \pi_{\theta_i}(\cdot|o_i)$ است و از ترفند پارامترسازی مجدد برای استخراج گرادیان استفاده می‌شود:

$$\tilde{a}_{i,\theta_i}(o_i, \xi_i) = \tanh\left(\mu_{\theta_i}(o_i) + \sigma_{\theta_i}(o_i) \odot \xi_i\right), \qquad \xi_i \sim \mathcal{N}(0, I) \tag{۶-۲۳}$$

شبکه‌های هدف

مشابه SAC، شبکه‌های هدف منتقد با میانگین‌گیری پولیاک به‌روزرسانی می‌شوند:

$$\phi_{i,j,\text{targ}} \leftarrow \rho\phi_{i,j,\text{targ}} + (1-\rho)\phi_{i,j} \qquad \text{برای}\, j = 1, 2 \tag{۶-۲۴}$$

تنظیم ضریب آنتروپی

یکی از مزایای MA-SAC، توانایی تنظیم خودکار ضریب آنتروپی α_i برای هر عامل است که می‌تواند با استفاده از یک تابع هزینه مجزا بهینه شود:

$$\min_{\alpha_i} \mathop{\mathrm{E}}_{\boldsymbol{o}\sim\mathcal{D}, \tilde{a}_i\sim\pi_{\theta_i}} \left[-\alpha_i\Big(\log \pi_{\theta_i}(\tilde{a}_i|o_i) + H_{\text{target}}\Big)\right] \tag{۶-۲۵}$$

که در آن H_{target} آنتروپی هدف است که به عنوان یک ابرپارامتر تعیین می‌شود.

۶-۵-۴ اکتشاف در MA-SAC

اکتشاف در MA-SAC به صورت ذاتی از طریق سیاست‌های تصادفی و تنظیم آنتروپی انجام می‌شود. برخلاف MA-DDPG و MA-TD3 که به افزودن نویز به اعمال نیاز دارند، MA-SAC اعمال را مستقیماً از توزیع احتمال سیاست نمونه‌گیری می‌کند:

$$a_i \sim \pi_{\theta_i}(\cdot|o_i) \tag{۶-۲۶}$$

این رویکرد امکان اکتشاف ساختاریافته‌تر و کارآمدتر را فراهم می‌کند که در محیط‌های چندعاملی پیچیده مفید است.

۶-۵-۵ شبه‌کد MA-SAC برای بازی‌های چندعاملیِ مجموع‌صفر

در این بخش، شبه‌کد الگوریتم MA-SAC پیاده‌سازی‌شده آورده شده‌است. در این پژوهش الگوریتم ۷ در محیط پایتون با استفاده از کتابخانه PyTorch [۵۹] پیاده‌سازی شده‌است.

الگوریتم ۷ عامل عملگر نقاد نرم چندعاملی

ورودی: پارامترهای اولیه سیاست عامل‌ها (θ_1, θ_2)، پارامترهای توابع Q $(\phi_{1,1}, \phi_{1,2}, \phi_{2,1}, \phi_{2,2})$، ضرایب آنتروپی (α_1, α_2)، بافر تکرار بازی خالی $(\mathcal{D})$

پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید:

$\phi_{i,j,\text{targ}} \leftarrow \phi_{i,j}$ برای $i \in \{1,2\}$ و $j \in \{1,2\}$

تا وقتی همگرایی رخ دهد:

مشاهدات (o_1, o_2) را دریافت کنید

برای هر عامل i، عمل $a_i \sim \pi_{\theta_i}(\cdot|o_i)$ را انتخاب کنید

اعمال (a_1, a_2) را در محیط اجرا کنید

مشاهدات بعدی (o'_1, o'_2)، پاداش‌ها $(r_1, r_2 = -r_1)$ و سیگنال پایان d را دریافت کنید

تجربه $(o_1, o_2, a_1, a_2, r_1, r_2, o'_1, o'_2, d)$ را در بافر $\mathcal{D}$ ذخیره کنید

اگر $d = 1$ است، وضعیت محیط را بازنشانی کنید

اگر زمان به‌روزرسانی فرا رسیده است:

به ازای هر تعداد به‌روزرسانی:

یک دسته تصادفی از تجربیات، $B = \{(\boldsymbol{o}, \boldsymbol{a}, r_1, r_2, \boldsymbol{o}', d)\}$، از $\mathcal{D}$ نمونه‌گیری کنید.

اهداف را محاسبه کنید:

$$y_1 = r_1 + \gamma(1-d)\Big(\min_{j=1,2} Q_{\phi_{1,j,\text{targ}}}(o'_1, \tilde{a}'_1, \tilde{a}'_2) - \alpha_1 \log \pi_{\theta_1}(\tilde{a}'_1|o'_1)\Big)$$

$$y_2 = r_2 + \gamma(1-d)\Big(\min_{j=1,2} Q_{\phi_{2,j,\text{targ}}}(o'_2, \tilde{a}'_2, \tilde{a}'_1) - \alpha_2 \log \pi_{\theta_2}(\tilde{a}'_2|o'_2)\Big)$$

توابع Q را با نزول گرادیان به‌روزرسانی کنید:

$$\nabla_{\phi_{1,j}} \frac{1}{|B|} \sum_{B} \big(Q_{\phi_{1,j}}(o_1, a_1, a_2) - y_1\big)^2 \qquad \text{برای } j = 1,2$$

$$\nabla_{\phi_{2,j}} \frac{1}{|B|} \sum_{B} \big(Q_{\phi_{2,j}}(o_2, a_2, a_1) - y_2\big)^2 \qquad \text{برای } j = 1,2$$

سیاست‌ها را با صعود گرادیان به‌روزرسانی کنید:

$$\nabla_{\theta_1} \frac{1}{|B|} \sum_{\boldsymbol{o} \in B} \Big[\min_{j=1,2} Q_{\phi_{1,j}}(o_1, \tilde{a}_{1,\theta_1}(o_1, \xi_1), a_2) - \alpha_1 \log \pi_{\theta_1}(\tilde{a}_{1,\theta_1}(o_1, \xi_1)|o_1)\Big]$$

$$\nabla_{\theta_2} \frac{1}{|B|} \sum_{\boldsymbol{o} \in B} \Big[\min_{j=1,2} Q_{\phi_{2,j}}(o_2, \tilde{a}_{2,\theta_2}(o_2, \xi_2), a_1) - \alpha_2 \log \pi_{\theta_2}(\tilde{a}_{2,\theta_2}(o_2, \xi_2)|o_2)\Big]$$

ضرایب آنتروپی را با نزول گرادیان به‌روزرسانی کنید (اختیاری):

$$\nabla_{\alpha_1} \frac{1}{|B|} \sum_{\boldsymbol{o} \in B} -\alpha_1 \Big(\log \pi_{\theta_1}(\tilde{a}_{1,\theta_1}(o_1, \xi_1)|o_1) + H_{\text{target}}\Big)$$

$$\nabla_{\alpha_2} \frac{1}{|B|} \sum_{\boldsymbol{o} \in B} -\alpha_2 \Big(\log \pi_{\theta_2}(\tilde{a}_{2,\theta_2}(o_2, \xi_2)|o_2) + H_{\text{target}}\Big)$$

شبکه‌های هدف را به‌روزرسانی کنید:

$$\phi_{i,j,\text{targ}} \leftarrow \rho \phi_{i,j,\text{targ}} + (1-\rho)\phi_{i,j} \qquad \text{برای } i, j \in \{1,2\}$$

۶-۵-۶ مزایای MA-SAC در بازی‌های مجموع‌صفر

MA-SAC مزایای زیر را نسبت به سایر الگوریتم‌های چندعاملی در بازی‌های چندعاملیِ مجموع‌صفر ارائه می‌دهد:

- **اکتشاف بهتر:** استفاده از سیاست‌های تصادفی و تنظیم آنتروپی، اکتشاف فضای حالت–عمل را بهبود می‌بخشد که برای یافتن راه‌حل‌های بهینه در بازی‌های دوعاملی ضروری است.
- **ثبات بیشتر:** ترکیب منتقدهای دوگانه با تنظیم آنتروپی، یادگیری را پایدارتر می‌کند و از همگرایی زودهنگام به سیاست‌های ضعیف جلوگیری می‌کند.
- **سازگاری با محیط‌های پیچیده:** توانایی تنظیم خودکار تعادل بین اکتشاف و بهره‌برداری، MA-SAC را برای محیط‌های چندعاملی پیچیده مناسب می‌سازد.
- **عملکرد بهتر در مسائل با چندین بهینه محلی:** سیاست‌های تصادفی می‌توانند از دام‌های بهینه محلی فرار کنند و به راه‌حل‌های بهتر برسند.

در مجموع، MA-SAC ترکیبی از ویژگی‌های مثبت SAC و رویکردهای چندعاملی را ارائه می‌دهد که آن را به گزینه‌ای قدرتمند برای یادگیری سیاست‌های پیچیده در بازی‌های چندعاملیِ مجموع‌صفر تبدیل می‌کند، به‌ویژه در محیط‌هایی که اکتشاف کارآمد و سیاست‌های تصادفی اهمیت دارند.

۶-۶ عامل بهینه‌سازی سیاست مجاور چندعاملی

عامل بهینه‌سازی سیاست مجاور دوعاملی[13] توسعه‌ای از الگوریتم PPO برای محیط‌های چندعاملی است. در این بخش، به بررسی این الگوریتم در چارچوب بازی‌های چندعاملیِ مجموع‌صفر می‌پردازیم که در آن ترکیب ویژگی‌های PPO با رویکرد چندعاملی به پایداری و کارایی بیشتر در یادگیری منجر می‌شود.

۶-۶-۱ چالش‌های یادگیری تقویتی در محیط‌های چندعاملی و راه‌حل MA-PPO

در محیط‌های چندعاملی، عامل‌ها همزمان سیاست‌های خود را تغییر می‌دهند که باعث غیرایستایی محیط از دید هر عامل می‌شود. این چالش با پیچیدگی‌های ذاتی الگوریتم‌های مبتنی بر گرادیان سیاست مانند PPO ترکیب می‌شود.

[13] Multi-Agent Proximal Policy Optimization (MA-PPO)

MA-PPO این چالش‌ها را با ترکیب رویکردهای زیر حل می‌کند:

- **آموزش متمرکز، اجرای غیرمتمرکز:** مشابه سایر الگوریتم‌های چندعاملی، از منتقدهایی استفاده می‌کند که به اطلاعات کامل دسترسی دارند، اما بازیگران تنها به مشاهدات محلی خود دسترسی دارند.
- **به‌روزرسانی کلیپ‌شده:** استفاده از مکانیسم کلیپ شده PPO برای محدود کردن به‌روزرسانی‌های سیاست، که به پایداری بیشتر در یادگیری چندعاملی کمک می‌کند.
- **بافر تجربه مشترک:** استفاده از یک بافر تجربه مشترک که تعاملات بین عامل‌ها را ثبت می‌کند.

۶-۶-۲ معماری MA-PPO در بازی‌های مجموع‌صفر

در یک بازی چندعاملیِ مجموع‌صفر، هر عامل دارای شبکه‌های زیر است:

- **شبکه بازیگر:** $\pi_{\theta_i}(a_i|o_i)$ که توزیع احتمال اعمال را با توجه به مشاهدات محلی تعیین می‌کند.
- **شبکه منتقد:** $V_{\phi_i}(\boldsymbol{o})$ که ارزش حالتِ متمرکز را (با دسترسی به مشاهدات همهٔ عامل‌ها) تخمین می‌زند و برای محاسبهٔ تابع مزیت استفاده می‌شود.

۶-۶-۳ آموزش MA-PPO

فرایند آموزش MA-PPO به شرح زیر است:

جمع‌آوری تجربیات

در هر تکرار، عامل‌ها با استفاده از سیاست‌های فعلی خود در محیط تعامل می‌کنند و مجموعه‌ای از مسیرها را جمع‌آوری می‌کنند:

$$\mathcal{D}_k = \{(o_1^t, o_2^t, a_1^t, a_2^t, r_1^t, r_2^t, o_1^{t+1}, o_2^{t+1})\} \qquad (۶-۲۷)$$

محاسبه مزیت

برای هر عامل $i \in \{1, 2\}$، تابع مزیت با استفاده از تابع ارزش فعلی محاسبه می‌شود. روش‌های مختلفی برای محاسبه مزیت وجود دارد؛ یک روش متداول استفاده از تخمین‌زننده مزیت تعمیم‌یافته (GAE) است:

$$\hat{A}_i^t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{i,t+l} \tag{۶-۲۸}$$

که در آن $\delta_{i,t} = r_i^t + \gamma V_{\phi_i}(\boldsymbol{o}^{t+1}) - V_{\phi_i}(\boldsymbol{o}^t)$ است.

به‌روزرسانی سیاست

سیاست هر عامل با بیشینه کردن تابع هدف PPO-Clip به‌روزرسانی می‌شود:

$$\max_{\theta_i} \mathop{\mathrm{E}}_{(o_i,a_i)\sim\mathcal{D}_k} \left[\min\left(\frac{\pi_{\theta_i}(a_i|o_i)}{\pi_{\theta_{i,k}}(a_i|o_i)} \hat{A}_i, \;\; \textbf{clip}\left(\frac{\pi_{\theta_i}(a_i|o_i)}{\pi_{\theta_{i,k}}(a_i|o_i)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_i \right) \right] \tag{۶-۲۹}$$

یا با استفاده از همان فرمول‌بندی ساده‌تر:

$$\max_{\theta_i} \mathop{\mathrm{E}}_{(o_i,a_i)\sim\mathcal{D}_k} \left[\min\left(\frac{\pi_{\theta_i}(a_i|o_i)}{\pi_{\theta_{i,k}}(a_i|o_i)} \hat{A}_i, \;\; g(\epsilon, \hat{A}_i) \right) \right] \tag{۶-۳۰}$$

که تابع g به صورت زیر تعریف شده‌است:

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geqslant 0 \\ (1-\epsilon)A & A < 0 \end{cases} \tag{۶-۳۱}$$

به‌روزرسانی منتقد

تابع ارزش هر عامل با کمینه کردن خطای میانگین مربعات به‌روزرسانی می‌شود:

$$\min_{\phi_i} \mathop{\mathrm{E}}_{(o_i,\hat{R}_i)\sim\mathcal{D}_k} \left[\left(V_{\phi_i}(o_i) - \hat{R}_i \right)^2 \right] \tag{۶-۳۲}$$

که در آن $\hat{R}_i$ بازده تنزیل‌شده برای عامل i است.

۶-۶-۴ اکتشاف در MA-PPO

اکتشاف در MA-PPO به صورت ذاتی از طریق سیاست‌های تصادفی انجام می‌شود. برخلاف الگوریتم‌های مبتنی بر DDPG که به افزودن نویز به اعمال نیاز دارند، MA-PPO از توزیع احتمال سیاست برای اکتشاف استفاده می‌کند:

$$a_i \sim \pi_{\theta_i}(\cdot|o_i) \tag{۶-۳۳}$$

این رویکرد اکتشاف سیاست‌محور، در ترکیب با مکانیسم کلیپ PPO که از به‌روزرسانی‌های بزرگ سیاست جلوگیری می‌کند، به ثبات بیشتر در یادگیری چندعاملی کمک می‌کند.

۶-۶-۵ شبه‌کد MA-PPO برای بازی‌های چندعاملیِ مجموع‌صفر

در این بخش، شبه‌کد الگوریتم MA-PPO پیاده‌سازی‌شده آورده شده‌است. در این پژوهش الگوریتم ۸ در محیط پایتون با استفاده از کتابخانه PyTorch [۵۹] پیاده‌سازی شده‌است.

الگوریتم ۸ عامل بهینه‌سازی سیاست مجاور چندعاملی

ورودی: پارامترهای اولیه سیاست عامل‌ها (θ_1, θ_2)، پارامترهای تابع ارزش (ϕ_1, ϕ_2)

۱: به ازای $k = 0, 1, 2, ...$:

۲: مجموعه‌ای از مسیرها به نام $\mathcal{D}_k = \{(o_1^t, o_2^t, a_1^t, a_2^t, r_1^t, r_2^t, o_1^{t+1}, o_2^{t+1})\}$ با اجرای سیاست‌های π_{θ_1} و π_{θ_2} در محیط جمع‌آوری شود.

۳: برای هر عامل i، پاداش‌های باقی‌مانده $\hat{R}_i^t$ را محاسبه کنید.

۴: برای هر عامل i، برآوردهای مزیت $\hat{A}_i^t$ را با استفاده از تابع ارزش فعلی V_{ϕ_i} محاسبه کنید.

۵: برای هر عامل i، سیاست را با به حداکثر رساندن تابع هدف PPO-Clip به‌روزرسانی کنید:

$$\theta_{i,k+1} = \arg\max_{\theta_i} \frac{1}{|\mathcal{D}_k|} \sum_{(o_i,a_i)\in\mathcal{D}_k} \min\left(\frac{\pi_{\theta_i}(a_i|o_i)}{\pi_{\theta_{i,k}}(a_i|o_i)}\hat{A}_i, \quad g(\epsilon, \hat{A}_i)\right)$$

۶: برای هر عامل i، تابع ارزش را با رگرسیون بر روی میانگین مربعات خطا به‌روزرسانی کنید:

$$\phi_{i,k+1} = \arg\min_{\phi_i} \frac{1}{|\mathcal{D}_k|} \sum_{(o_i)\in\mathcal{D}_k} \left(V_{\phi_i}(o_i) - \hat{R}_i\right)^2$$

۶-۶-۶ مزایای MA-PPO در بازی‌های مجموع‌صفر

MA-PPO مزایای زیر را نسبت به سایر الگوریتم‌های چندعاملی در بازی‌های چندعاملیِ مجموع‌صفر ارائه می‌دهد:

- **پایداری یادگیری:** مکانیسم کلیپ PPO از به‌روزرسانی‌های بزرگ سیاست جلوگیری می‌کند که به پایداری بیشتر در محیط‌های غیرایستای چندعاملی منجر می‌شود.
- **کارایی نمونه:** به عنوان یک روش درون‌سیاست، MA-PPO معمولاً کارایی نمونهٔ کمتری نسبت به روش‌های برون‌سیاست مانند MA-TD3 و MA-SAC دارد، اما پایداری بهتری در به‌روزرسانی‌ها ارائه می‌کند.
- **اکتشاف سیاست‌محور:** اکتشاف ذاتی از طریق سیاست‌های تصادفی به جای افزودن نویز به اعمال، به اکتشاف کارآمدتر فضای حالت–عمل کمک می‌کند.

- **مقیاس‌پذیری:** MA-PPO به‌راحتی به سیستم‌های با تعداد بیشتری از عامل‌ها قابل گسترش است، اگرچه در این پژوهش بر بازی‌های دوعاملی تمرکز شده‌است.

در مجموع، MA-PPO ترکیبی از سادگی و کارایی PPO با رویکردهای چندعاملی را ارائه می‌دهد که آن را به گزینه‌ای قدرتمند برای یادگیری در بازی‌های چندعاملیِ مجموع‌صفر تبدیل می‌کند.

فصل ۷

ارزیابی و نتایج یادگیری

در این فصل، چارچوب ارزیابی و نتایج تجربی چهار الگوریتم شاخص یادگیری تقویتی برای کنترل فضاپیما در میدان گرانشی سه‌جسمی برای روش‌های DDPG، PPO، SAC و TD3 ارائه می‌شود. تحلیل‌ها در دو قسمت انجام می‌گیرد: حالت تک‌عاملی استاندارد و حالت چندعاملیِ بازی مجموع‌صفر. تمرکز ارزیابی بر سه محور اصلی است: سنجش مقاومت در برابر آشفتگی‌های محیطی و سامانه‌ای (شرایط اولیه‌ی تصادفی، اغتشاش عملگر، عدم‌تطابق مدل، مشاهده‌ی ناقص، نویز حسگر و تأخیر زمانی)، ارزیابی کیفیت مسیر و پروفایل فرمان پیشران، و گزارش شاخص‌های کمی شامل پاداش تجمعی، خطای مسیر، تلاش کنترلی و احتمال شکست.

به‌منظور هدایت خواننده و تضمین بازتولیدپذیری، ابتدا پروتکل و سناریوهای ارزیابی مقاومت همراه با جزئیات پیاده‌سازی و پارامترگذاری در بخش ۷-۱ معرفی می‌شود. سپس نتایج هر یک از الگوریتم‌ها به‌صورت نظام‌مند ارائه می‌گردد؛ بدین‌ترتیب که مسیر طی‌شده، فرمان‌های پیشران و توزیع پاداش در سناریوهای مختلف تحلیل می‌شود. نتایج DDPG در بخش ۷-۲، نتایج PPO در بخش ۷-۵، نتایج SAC در بخش ۷-۴ و نتایج TD3 در بخش ۷-۳ گزارش شده‌اند. در پایان، جمع‌بندی مقایسه‌ای برای نسخه‌های تک‌عاملی در بخش ۷-۶ و چندعاملیِ مجموع‌صفر در بخش ۷-۷ ارائه می‌شود تا تصویر روشنی از عملکرد نسبی روش‌ها فراهم گردد. در این مقایسه‌ها علاوه بر شاخص‌های عددی، مبادله‌های کارایی–پایداری و حساسیت نسبت به اغتشاش‌ها نیز مورد بحث قرار می‌گیرد.

۷-۱ ارزیابی مقاومت الگوریتم‌ها

در این بخش، مقاومت الگوریتم‌های یادگیری در برابر شرایط مختلف اختلال مورد بررسی قرارگرفته است. این ارزیابی شامل شش سناریوی چالش‌برانگیز می‌شود: (۱) شرایط اولیه تصادفی، (۲) اغتشاش در عملگرها، (۳)

عدم تطابق مدل، (۴) مشاهده ناقص، (۵) نویز حسگر و (۶) تأخیر زمانی. هدف، بررسی توانایی الگوریتم‌ها در حفظ کارایی خود در شرایط غیرایده‌آل و نزدیک به واقعیت است.

۷-۱-۱ سناریوهای ارزیابی مقاومت

به‌منظور ایجاز، مشخصات هر سناریو به‌صورت فشرده فهرست شده است:

۱. شرایط اولیه تصادفی: به هر مؤلفه حالت اولیه نویز گوسی با انحراف معیار σ=0.1 افزوده می‌شود:

$$x_0 \leftarrow x_0 + \mathcal{N}(0,\ 0.1^2)$$

۲. اغتشاش در عملگرها: نویز افزایشی روی ورودی‌ها و نویز کوچک روی سنسورها:

$$u_t \leftarrow u_t + \mathcal{N}(0,\ 0.05^2)$$

$$y_t \leftarrow y_t + \mathcal{N}(0,\ 0.02^2)$$

۳. عدم تطابق مدل: پارامترهای دینامیک در طول انتقال با نویز گوسی مختل می‌شوند:

$$\theta \leftarrow \theta + \mathcal{N}(0,\ 0.05^2)$$

۴. مشاهده ناقص: در هر گام، به‌صورت تصادفی 50% از مؤلفه‌های مشاهده ماسک شده و مقدارشان صفر می‌شود:

$$m_t^{(i)} \sim \mathrm{Bernoulli}(0.5), \quad y_t \leftarrow y_t \circ m_t$$

۵. نویز حسگر: نویز گوسی ضربی با σ=0.05 روی هر مؤلفه مشاهده اعمال می‌شود:

$$y_t \leftarrow y_t \circ \big(1 + \mathcal{N}(0,\ 0.05^2)\big)$$

۶. تأخیر زمانی: اعمال عامل با تأخیر 10 گام زمانی اعمال می‌شود و روی عملِ تاخیردار نویز افزایشی افزوده می‌گردد:

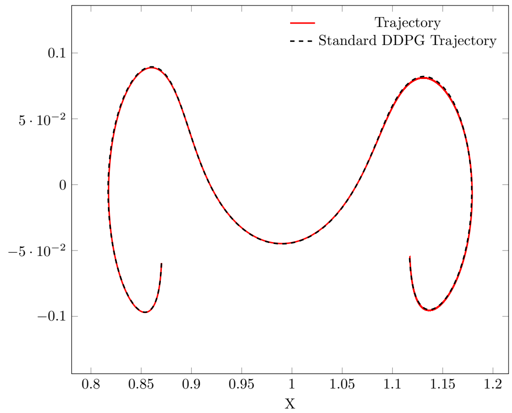
$$u_t^{\text{applied}} \leftarrow u_{t-10} + \mathcal{N}(0,\ 0.05^2)$$

۷-۲ الگوریتم DDPG

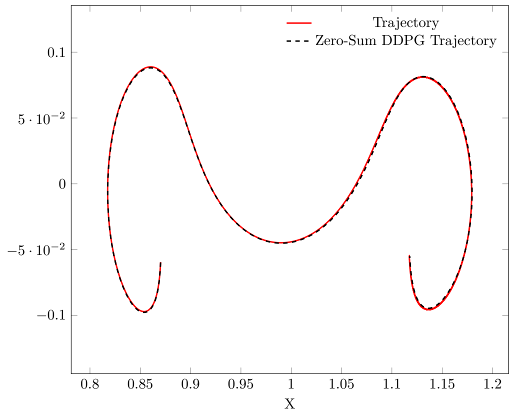
الگوریتم DDPG از جمله روش‌های یادگیری خارج از سیاست است که از دو شبکه عصبی برای بازیگر و منتقد استفاده می‌کند. در اینجا، عملکرد نسخه استاندارد و نسخه مبتنی بر بازی مجموع‌صفر این الگوریتم در کنترل فضاپیما مقایسه شده است.

۷-۲-۱ مسیر طی‌شده

این بخش مسیر طی‌شده فضاپیما را برای نسخه استاندارد و نسخه بازی مجموع‌صفر DDPG نشان می‌دهد.



(آ) DDPG استاندارد

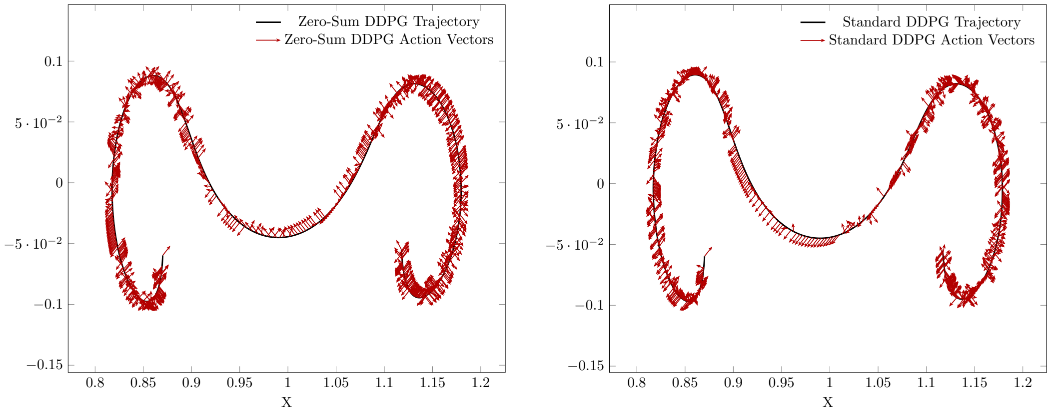


(ب) MA-DDPG بازی مجموع‌صفر

شکل ۷-۱: مسیر طی‌شده فضاپیما با DDPG استاندارد و نسخه بازی مجموع‌صفر MA-DDPG.

۷-۲-۲ مسیر و فرمان پیشران

این بخش مسیر و پروفایل فرمان پیشران در طول زمان را برای هر دو نسخه DDPG ارائه می‌کند.



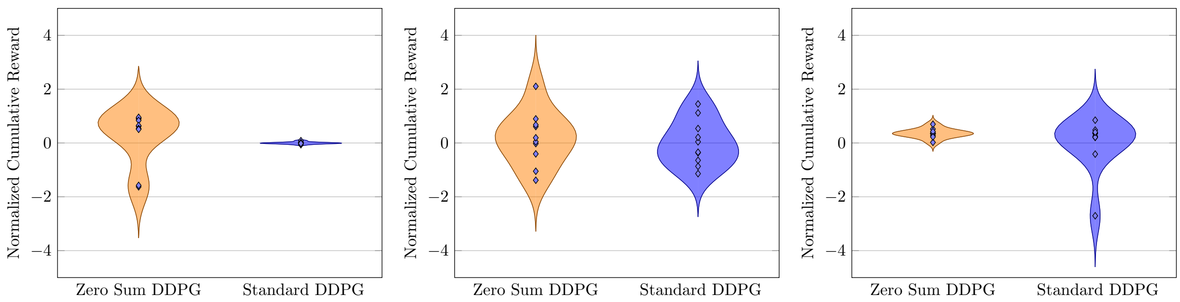
(آ) DDPG استاندارد

(ب) MA-DDPG بازی مجموع‌صفر

شکل ۷-۲: مسیر و فرمان پیشران فضاپیما در DDPG استاندارد و نسخه بازی مجموع‌صفر MA-DDPG.

۷-۲-۳ توزیع پاداش تجمعی

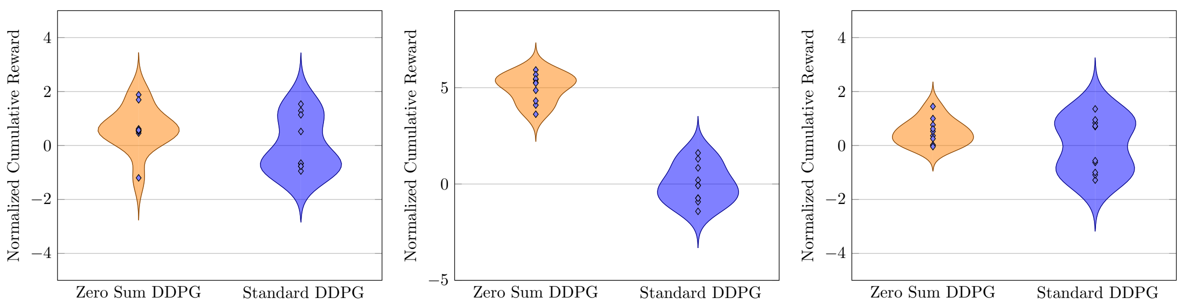
این بخش نمودارهای ویولن توزیع پاداش تجمعی را در سناریوهای مختلف برای DDPG و MA-DDPG نمایش می‌دهد.



(آ) شرایط اولیه تصادفی

(ب) اغتشاش در عملگرها

(ج) عدم تطابق مدل



(د) مشاهده ناقص

(ه) نویز حسگر

(و) تأخیر زمانی

شکل ۷-۳: مقایسه توزیع پاداش تجمعی در سناریوهای مختلف برای DDPG و MA-DDPG.

۴-۲-۷ مقایسه عددی

این بخش شاخص‌های عددی را گزارش می‌کند؛ نتایج بر اساس ۱۰۰ اجرای مستقل شبیه‌سازی برای هر سناریو به‌دست آمده‌اند.

سناریو	پاداش تجمعی		مجموع خطای مسیر		مجموع تلاش کنترلی		احتمال شکست	
	DDPG	MA-DDPG	DDPG	MA-DDPG	DDPG	MA-DDPG	DDPG	MA-DDPG
شرایط اولیه تصادفی	-4.17	-3.63	0.40	0.63	5.60	5.60	1.00	1.00
اغتشاش در عملگرها	-1.93	-1.96	7.56	7.94	5.60	5.59	0.90	0.30
عدم تطابق مدل	-3.24	-2.70	0.70	0.76	5.57	5.57	1.00	1.00
مشاهده ناقص	-3.28	-2.89	0.68	0.75	5.57	5.57	0.60	0.80
نویز حسگر	-1.07	-0.47	0.10	0.15	5.54	5.54	0.00	0.00
تأخیر زمانی	-3.20	-1.91	1.74	2.43	5.61	5.61	0.70	0.70

جدول ۷-۱: مقایسه عملکرد DDPG و MA-DDPG در سناریوهای مختلف مقاومت

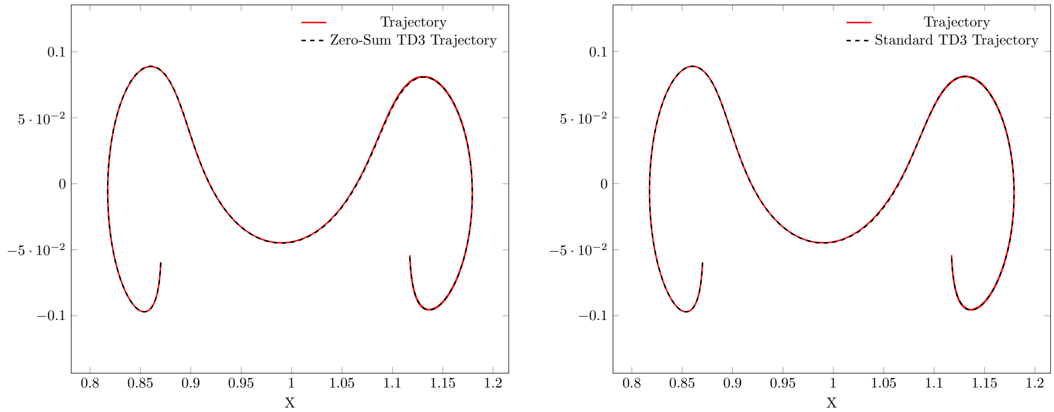
در جمع‌بندی بر اساس داده‌های جدول، MA-DDPG در پنج سناریو پاداش تجمعی بهتری از DDPG دارد و تنها در اغتشاش در عملگرها DDPG اندکی بهتر است. از نظر مجموع خطای مسیر، DDPG در همه سناریوها مقدار کمتری ثبت کرده است. تلاش کنترلی در همه موارد تقریباً برابر است. در احتمال شکست، MA-DDPG در اغتشاش در عملگرها بهتر است (۰/۳۰ در برابر ۰/۹۰)، DDPG در مشاهده ناقص بهتر است (۰/۶۰ در برابر ۰/۸۰) و در سایر سناریوها دو روش برابر هستند.

۳-۷ الگوریتم TD3

الگوریتم TD3 (یادگیری تفاضل زمانی سه‌گانه عمیق) نسخه بهبودیافته DDPG است که با استفاده از تکنیک‌های جدید مانند شبکه‌های دوگانه منتقد و تأخیر در بروزرسانی سیاست، مشکلات تخمین بیش از حد را کاهش می‌دهد.

۷-۳-۱ مسیر طی‌شده

این بخش مسیر طی‌شده فضاپیما را برای نسخه استاندارد و نسخه بازی مجموع‌صفر TD3 نشان می‌دهد.



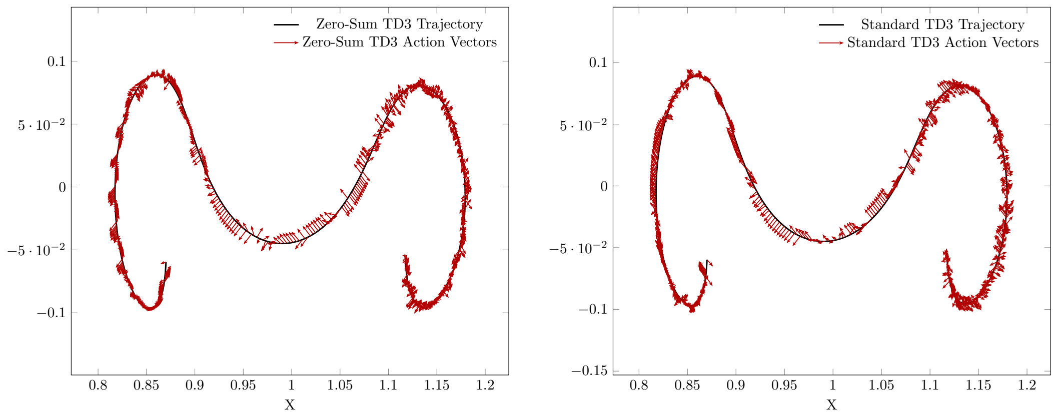
(آ) TD3 استاندارد

(ب) MA-TD3 بازی مجموع‌صفر

شکل ۷-۴: مسیر طی‌شده فضاپیما با TD3 استاندارد و نسخه بازی مجموع‌صفر MA-TD3.

۷-۳-۲ مسیر و فرمان پیشران

این بخش مسیر و پروفایل فرمان پیشران در طول زمان را برای هر دو نسخه TD3 ارائه می‌کند.



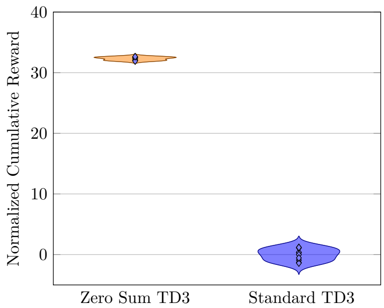
(آ) TD3 استاندارد

(ب) MA-TD3 بازی مجموع‌صفر

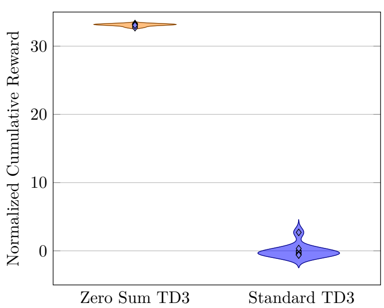
شکل ۷-۵: مسیر و فرمان پیشران فضاپیما در TD3 استاندارد و نسخه بازی مجموع‌صفر MA-TD3.

۷-۳-۳ توزیع پاداش تجمعی

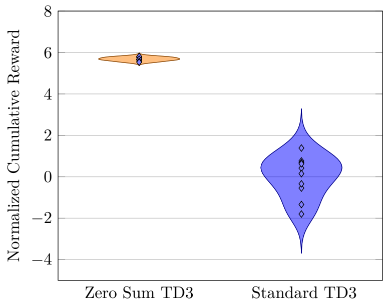
این بخش نمودارهای ویولن توزیع پاداش تجمعی را در سناریوهای مختلف برای TD3 و MA-TD3 نمایش می‌دهد.



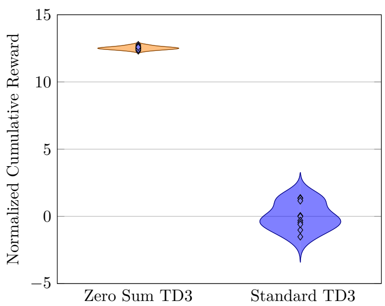
(آ) شرایط اولیه تصادفی



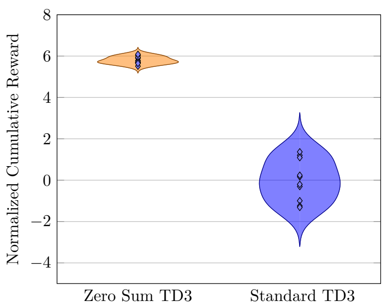
(ب) اغتشاش در عملگرها



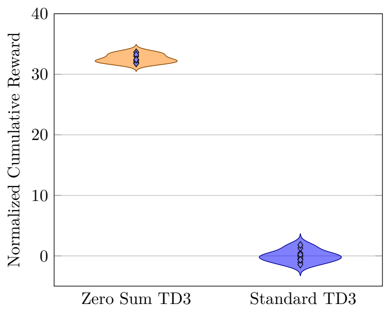
(ج) عدم تطابق مدل



(د) مشاهده ناقص



(ه) نویز حسگر



(و) تأخیر زمانی

شکل ۷-۶: مقایسه توزیع پاداش تجمعی در سناریوهای مختلف برای TD3 و MA-TD3.

۷-۳-۴ مقایسه عددی

این بخش شاخص‌های عددی را گزارش می‌کند؛ نتایج بر اساس ۱۰۰ اجرای مستقل شبیه‌سازی برای هر سناریو به‌دست آمده‌اند.

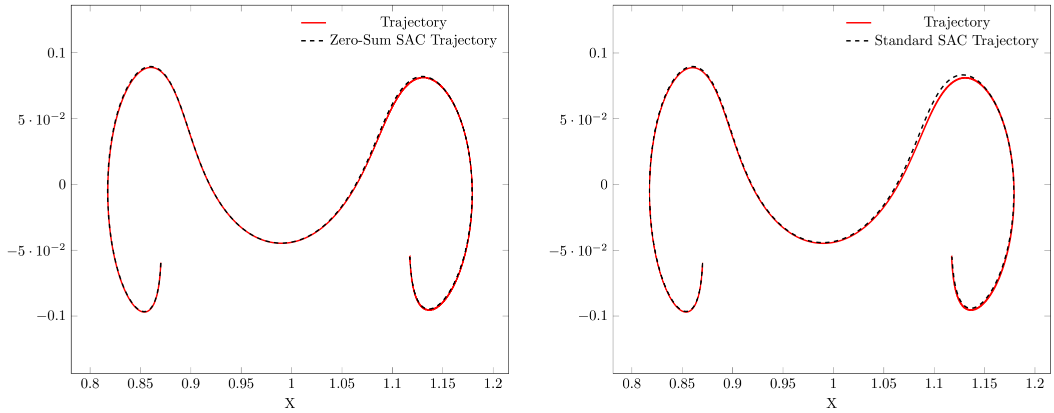
سناریو	پاداش تجمعی		مجموع خطای مسیر		مجموع تلاش کنترلی		احتمال شکست	
	TD3	MA-TD3	TD3	MA-TD3	TD3	MA-TD3	TD3	MA-TD3
شرایط اولیه تصادفی	−2.95	−0.26	0.39	0.14	4.57	4.57	1.00	0.30
اغتشاش در عملگرها	0.56	0.73	0.02	0.00	2.66	2.66	0.00	0.00
عدم تطابق مدل	−4.73	−3.30	0.47	0.73	5.41	5.41	1.00	1.00
مشاهده ناقص	0.21	0.71	0.02	0.01	3.18	3.18	0.00	0.00
نویز حسگر	−0.08	−2.93	0.11	3.19	5.50	5.50	0.00	1.00
تأخیر زمانی	0.55	0.67	0.01	0.01	4.57	4.57	0.00	0.00

جدول ۷-۲: مقایسه عملکرد TD3 و MA-TD3 در سناریوهای مختلف مقاومت

الگوریتم TD3 در هر دو حالت عملکرد قابل توجهی دارد، اما نسخه بازی مجموع‌صفر آن بهبودهای معناداری در کیفیت مسیر و مصرف سوخت نشان می‌دهد. ثبات بیشتر این الگوریتم در مقایسه با DDPG در هر دو نسخه قابل مشاهده است.

۷-۴ الگوریتم SAC

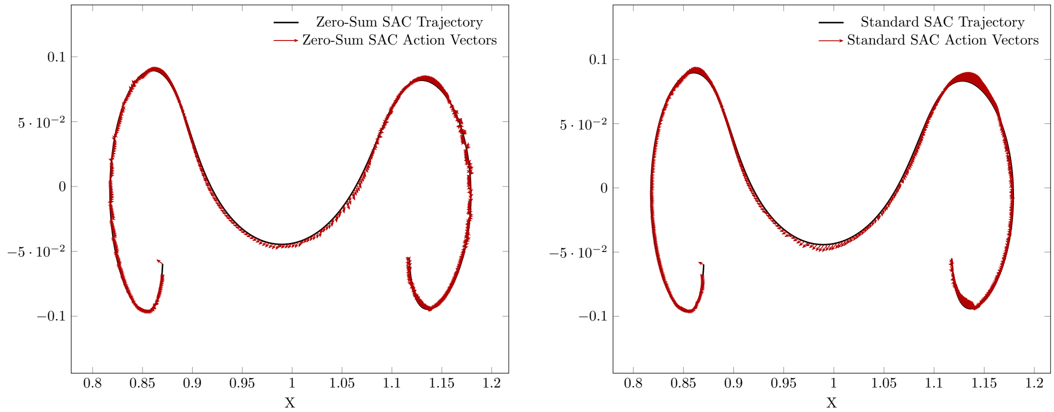
الگوریتم SAC از روش‌های نوین یادگیری تقویتی است که با استفاده از مفهوم آنتروپی، تعادل بهتری بین اکتشاف و بهره‌برداری ایجاد می‌کند. این الگوریتم در شرایط فضاهای پیوسته عملکرد قابل توجهی دارد.



(آ) SAC استاندارد

(ب) SAC بازی مجموع‌صفر

شکل ۷-۷: مسیر طی‌شده فضاپیما با SAC استاندارد و نسخه بازی مجموع‌صفر MA-SAC.



(آ) SAC استاندارد

(ب) SAC بازی مجموع‌صفر

شکل ۷-۸: مسیر و فرمان پیشران فضاپیما در SAC استاندارد و نسخه بازی مجموع‌صفر MA-SAC.

الگوریتم SAC در هر دو حالت عملکرد قابل قبولی ارائه می‌دهد. ویژگی خاص این الگوریتم در تنظیم خودکار پارامتر آنتروپی باعث می‌شود که بتواند تعادل مناسبی بین اکتشاف و بهره‌برداری ایجاد کند، اما نسخه بازی مجموع‌صفر آن در شرایط سخت‌تر مقاومت بیشتری نشان می‌دهد.

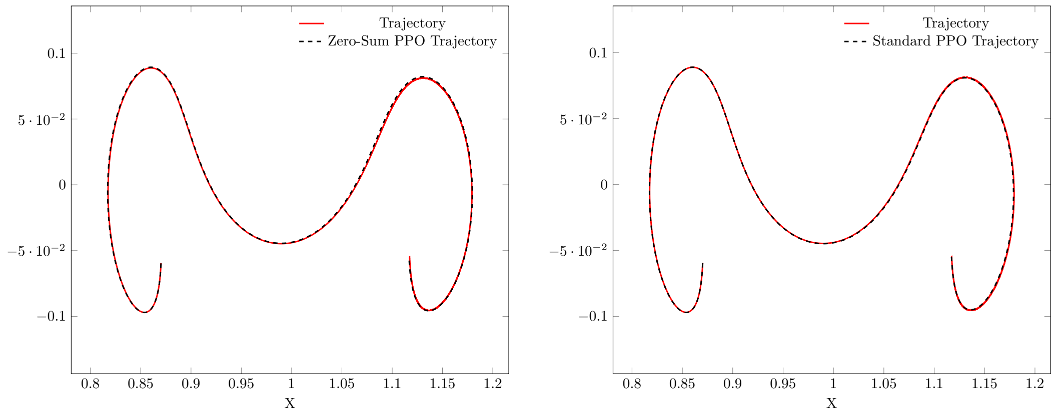
سناریو	پاداش تجمعی		مجموع خطای مسیر		مجموع تلاش کنترلی		احتمال شکست	
	SAC	MA-SAC	SAC	MA-SAC	SAC	MA-SAC	SAC	MA-SAC
شرایط اولیه تصادفی	−4.69	−2.98	0.29	0.26	1.37	1.37	1.00	1.00
اغتشاش در عملگرها	−1.95	−1.93	8.02	7.72	3.09	3.09	1.00	1.00
عدم تطابق مدل	−4.89	−4.35	0.38	0.26	1.16	1.16	1.00	1.00
مشاهده ناقص	−3.63	−0.44	1.95	0.07	1.99	1.99	1.00	0.00
نویز حسگر	−0.89	0.12	0.12	0.12	1.86	1.86	0.00	0.00
تأخیر زمانی	−4.14	−0.05	1.87	0.01	1.25	1.25	1.00	0.00

جدول ۷-۳: مقایسه عملکرد SAC و MA-SAC در سناریوهای مختلف مقاومت

۷-۵ الگوریتم PPO

الگوریتم PPO از روش‌های نوین سیاست گرادیان است که با محدودسازی میزان تغییرات در هر بروزرسانی، پایداری بیشتری در فرآیند یادگیری ایجاد می‌کند. در ادامه، عملکرد این الگوریتم در دو حالت مورد بررسی قرار گرفته است.

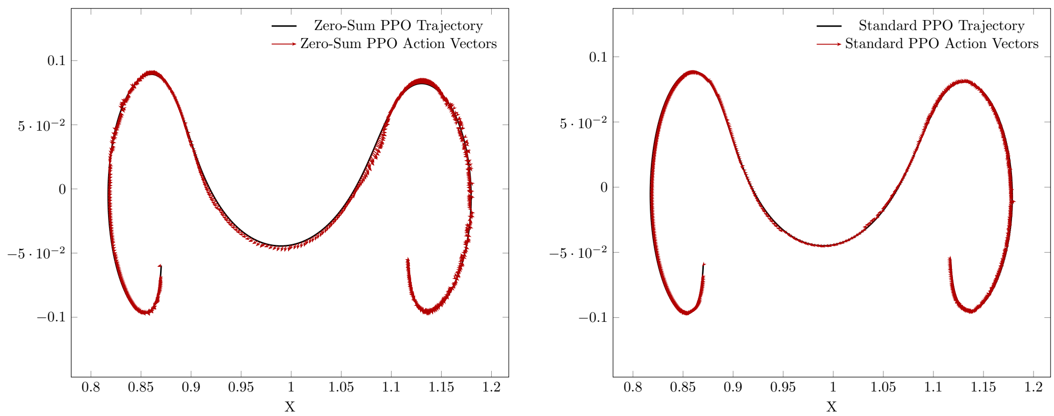
۷-۵-۱ مسیر طی‌شده



(ب) MA-PPO بازی مجموع‌صفر　　　(آ) PPO استاندارد

شکل ۷-۹: مسیر طی‌شده فضاپیما با PPO استاندارد و نسخه بازی مجموع‌صفر MA-PPO.

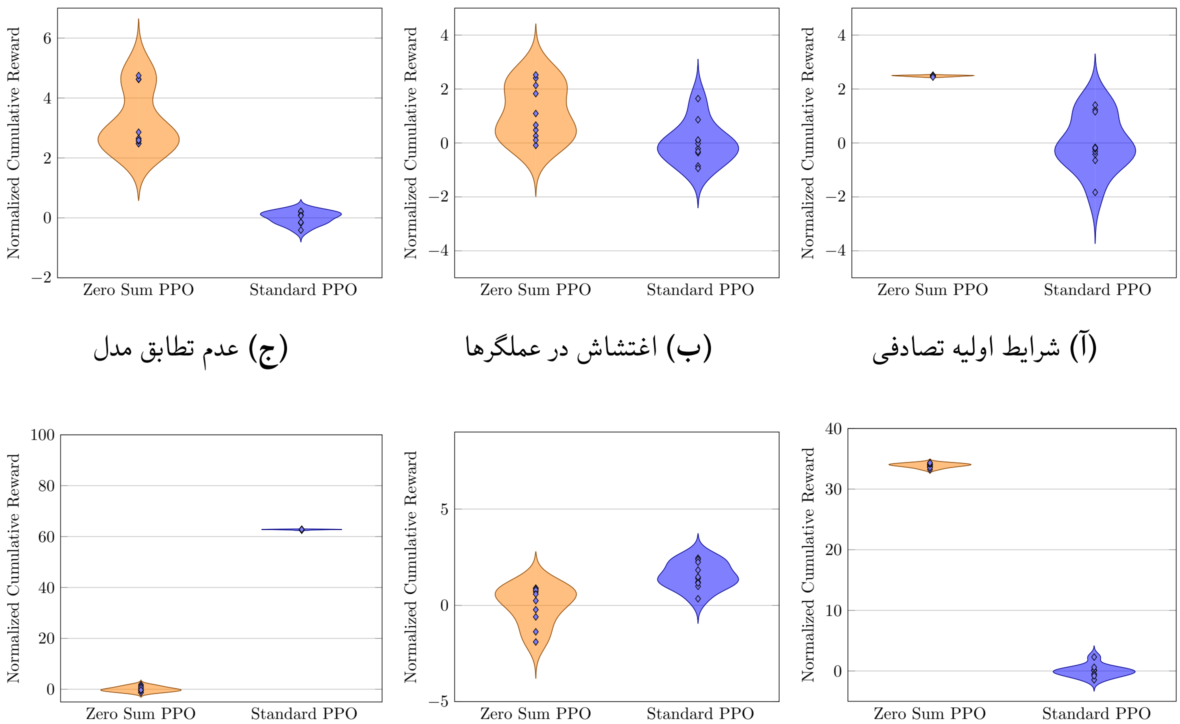
۷-۵-۲ مسیر و فرمان پیشران



(ب) MA-PPO بازی مجموع‌صفر　　　(آ) PPO استاندارد

شکل ۷-۱۰: مسیر و فرمان پیشران فضاپیما در PPO استاندارد و نسخه بازی مجموع‌صفر MA-PPO.

۷-۵-۳ توزیع پاداش تجمعی



(آ) شرایط اولیه تصادفی

(ب) اغتشاش در عملگرها

(ج) عدم تطابق مدل

(د) مشاهده ناقص

(ه) نویز حسگر

(و) تأخیر زمانی

شکل ۷-۱۱: مقایسه توزیع پاداش تجمعی برای PPO و MA-PPO در سناریوهای مختلف.

۷-۵-۴ مقایسه عددی

سناریو	پاداش تجمعی		مجموع خطای مسیر		مجموع تلاش کنترلی		احتمال شکست	
	PPO	MA-PPO	PPO	MA-PPO	PPO	MA-PPO	PPO	MA-PPO
شرایط اولیه تصادفی	-1.85	0.46	0.22	0.14	1.98	1.98	0.70	0.00
اغتشاش در عملگرها	-1.97	-1.91	8.33	7.50	3.42	3.42	1.00	1.00
عدم تطابق مدل	0.46	0.30	0.07	0.08	1.13	1.13	0.00	0.00
مشاهده ناقص	-3.60	-1.81	2.34	2.06	2.15	2.15	1.00	1.00
نویز حسگر	0.52	0.48	0.13	0.15	2.08	2.08	0.00	0.00
تأخیر زمانی	0.58	-2.44	0.03	2.49	2.56	2.56	0.00	1.00

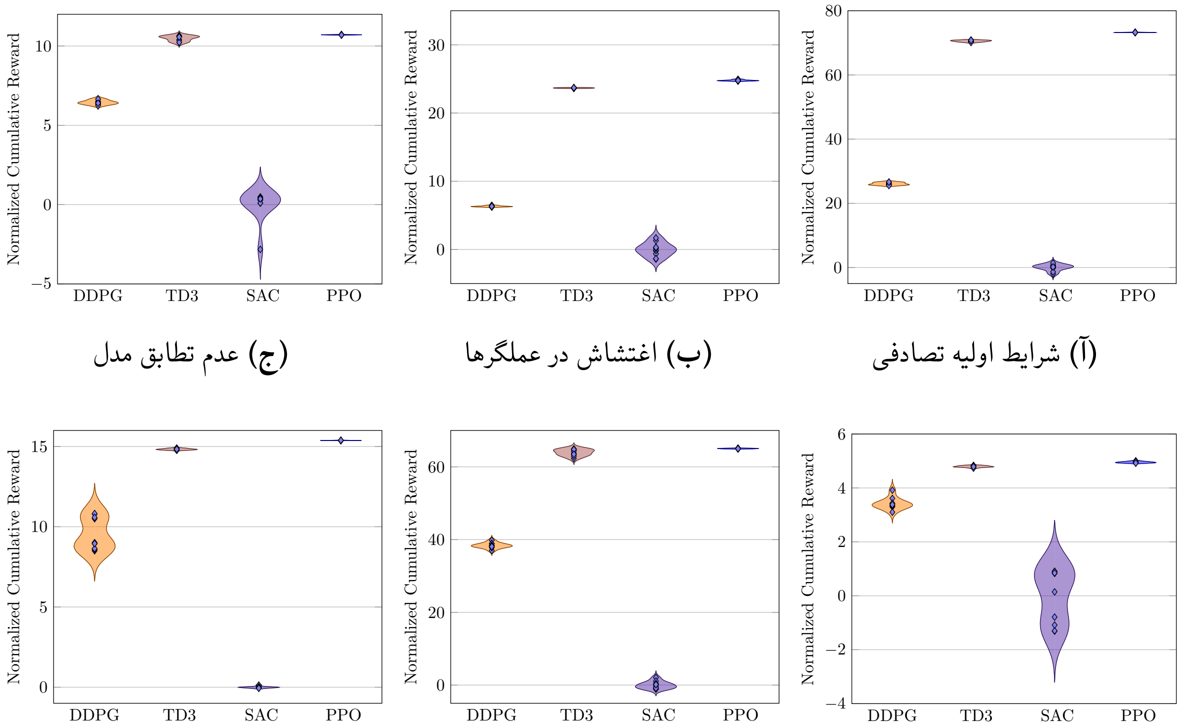
جدول ۷-۴: مقایسه عملکرد PPO و MA-PPO در سناریوهای مختلف مقاومت

نتایج نشان می‌دهد که الگوریتم PPO در حالت بازی مجموع‌صفر عملکرد قابل توجهی دارد، اما تفاوت آن با نسخه استاندارد کمتر از DDPG است. این می‌تواند به دلیل ماهیت ذاتی PPO در ایجاد تعادل بین اکتشاف و بهره‌برداری باشد که آن را در حالت استاندارد نیز نسبتاً مقاوم می‌سازد.

۷-۶ نتایج نسخه استاندارد

در این بخش، نتایج نسخه‌های تک‌عاملی الگوریتم‌ها در سناریوهای مقاومت مختلف ارائه و تحلیل می‌شود.

۷-۶-۱ توزیع پاداش تجمعی



(آ) شرایط اولیه تصادفی
(ب) اغتشاش در عملگرها
(ج) عدم تطابق مدل
(د) مشاهده ناقص
(ه) نویز حسگر
(و) تأخیر زمانی

شکل ۷-۱۲: مقایسه توزیع پاداش تجمعی برای نسخه‌های تک‌عاملی در سناریوهای مختلف.

۷-۶-۲ مقایسه عددی

سناریو	DDPG	PPO	SAC	TD3
شرایط اولیه تصادفی	−0.41	0.34	−0.02	0.74
اغتشاش در عملگرها	−0.44	0.35	−0.02	0.73
عدم تطابق مدل	−0.63	0.38	−0.13	0.75
مشاهده ناقص	−1.52	0.40	−0.44	0.71
نویز حسگر	−0.60	0.37	−0.12	0.75
تأخیر زمانی	−1.19	0.17	−0.05	0.67

پاداش تجمعی

سناریو	DDPG	PPO	SAC	TD3
شرایط اولیه تصادفی	4.42	4.30	4.02	1.22
اغتشاش در عملگرها	4.39	4.38	4.01	1.26
عدم تطابق مدل	8.85	3.57	4.78	1.25
مشاهده ناقص	9.65	2.44	5.17	1.09
نویز حسگر	9.12	3.58	4.66	1.25
تأخیر زمانی	6.73	4.53	4.12	1.21

مجموع خطای مسیر

سناریو	DDPG	PPO	SAC	TD3
شرایط اولیه تصادفی	5.11	0.77	1.76	3.31
اغتشاش در عملگرها	4.89	0.77	1.71	3.07
عدم تطابق مدل	5.48	0.86	2.37	4.32
مشاهده ناقص	5.37	1.03	2.33	4.10
نویز حسگر	5.48	0.86	2.37	4.30
تأخیر زمانی	5.51	0.76	2.11	5.12

مجموع تلاش کنترلی

سناریو	DDPG	PPO	SAC	TD3
شرایط اولیه تصادفی	0.00	0.00	0.00	0.00
اغتشاش در عملگرها	0.00	0.00	0.00	0.00
عدم تطابق مدل	0.00	0.00	1.00	0.00
مشاهده ناقص	0.00	0.00	1.00	0.00
نویز حسگر	0.00	0.00	1.00	0.00
تأخیر زمانی	0.00	0.00	1.00	0.00

احتمال شکست

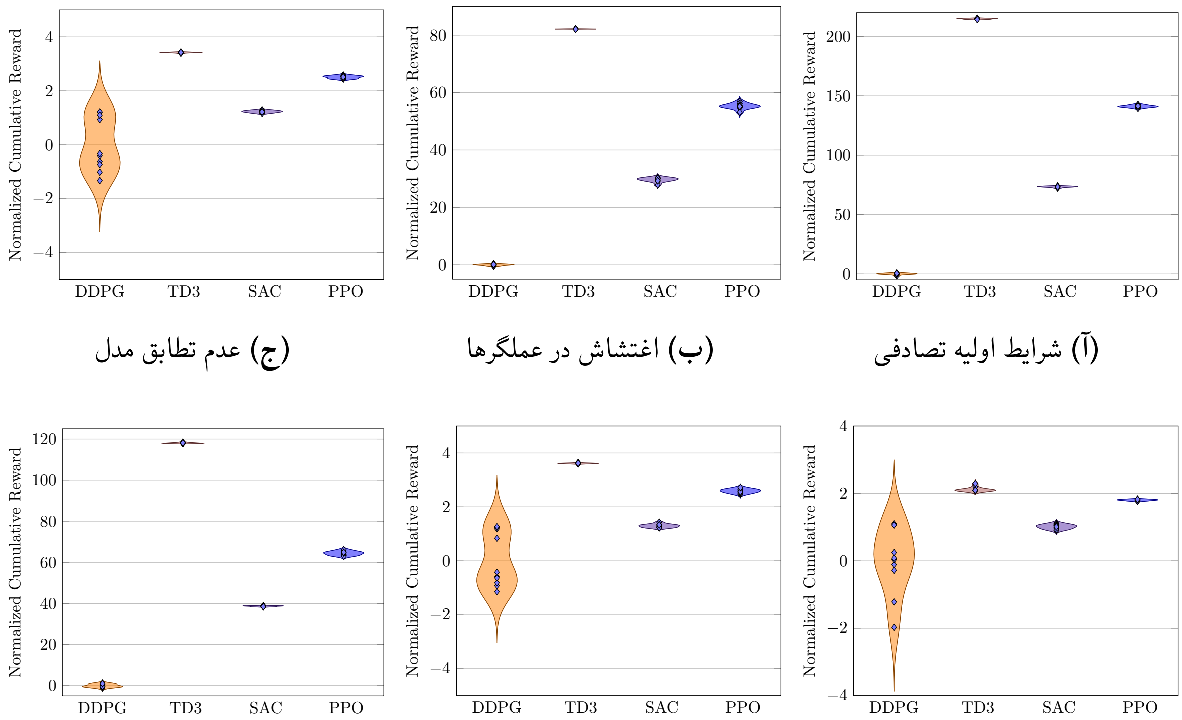
جدول ۷-۵: مقایسه الگوریتم‌های تک‌عاملی در سناریوهای مختلف مقاومت

بر اساس داده‌ها، TD3 به‌طور پایدار بالاترین پاداش و کمترین خطای مسیر را ثبت می‌کند، درحالی‌که PPO کمترین تلاش کنترلی را دارد. SAC در برخی سناریوهای دشوار (عدم تطابق مدل، مشاهده ناقص، نویز حسگر، تأخیر زمانی) نرخ شکست بالاتری نشان می‌دهد و DDPG عموماً از نظر پاداش و خطا ضعیف‌تر از PPO و TD3 است.

۷-۷ نتایج نسخه چندعاملی

در این بخش، عملکرد الگوریتم‌ها در حالت چندعاملیِ بازی مجموع‌صفر ارائه و تحلیل می‌شود.

۷-۷-۱ توزیع پاداش تجمعی



(آ) شرایط اولیه تصادفی (ب) اغتشاش در عملگرها (ج) عدم تطابق مدل

(د) مشاهده ناقص (ه) نویز حسگر (و) تأخیر زمانی

شکل ۷-۱۳: مقایسه توزیع پاداش تجمعی برای الگوریتم‌ها در حالت چندعاملی در سناریوهای مختلف.

۲-۷-۷ مقایسه عددی

سناریو	DDPG	PPO	SAC	TD3
شرایط اولیه تصادفی	−0.41	0.34	−0.02	0.74
اغتشاش در عملگرها	−0.44	0.35	−0.02	0.73
عدم تطابق مدل	−0.63	0.38	−0.13	0.75
مشاهده ناقص	−1.52	0.40	−0.44	0.71
نویز حسگر	−0.60	0.37	−0.12	0.75
تأخیر زمانی	−1.19	0.17	−0.05	0.67

پاداش تجمعی

سناریو	DDPG	PPO	SAC	TD3
شرایط اولیه تصادفی	4.42	4.30	4.02	1.22
اغتشاش در عملگرها	4.39	4.38	4.01	1.26
عدم تطابق مدل	8.85	3.57	4.78	1.25
مشاهده ناقص	9.65	2.44	5.17	1.09
نویز حسگر	9.12	3.58	4.66	1.25
تأخیر زمانی	6.73	4.53	4.12	1.21

مجموع خطای مسیر

سناریو	DDPG	PPO	SAC	TD3
شرایط اولیه تصادفی	5.11	0.77	1.76	3.31
اغتشاش در عملگرها	4.89	0.77	1.71	3.07
عدم تطابق مدل	5.48	0.86	2.37	4.32
مشاهده ناقص	5.37	1.03	2.33	4.10
نویز حسگر	5.48	0.86	2.37	4.30
تأخیر زمانی	5.51	0.76	2.11	5.12

مجموع تلاش کنترلی

سناریو	DDPG	PPO	SAC	TD3
شرایط اولیه تصادفی	0.00	0.00	0.00	0.00
اغتشاش در عملگرها	0.00	0.00	0.00	0.00
عدم تطابق مدل	0.00	0.00	0.20	0.00
مشاهده ناقص	0.00	0.00	0.20	0.00
نویز حسگر	0.00	0.00	0.20	0.00
تأخیر زمانی	0.00	0.00	0.20	0.00

احتمال شکست

جدول ۷-۶: مقایسه الگوریتم‌های چندعاملی در سناریوهای مختلف مقاومت

در حالت چندعاملی، TD3 به‌طور پایدار پاداش بالاتر و خطای مسیر کمتر ثبت می‌کند، در حالی‌که PPO کمترین تلاش کنترلی را نشان می‌دهد. عملکرد SAC و DDPG در برخی سناریوهای دشوار ضعیف‌تر است، هرچند نرخ‌های شکست عمدتاً پایین باقی می‌ماند.

فصل ۸

نتیجه‌گیری و پیشنهادها

در این پایان‌نامه، مسأله‌ی هدایتِ مقاومِ فضاپیماهای کم‌پیشران در دینامیکِ چندجسمیِ مدلِ CRTBP زمین–ماه به‌صورت یک بازی دیفرانسیلی مجموع‌صفر میان عامل هدایت و عامل مزاحم صورت‌بندی شد و با الگوی آموزش متمرکز–اجرای توزیع‌شده دنبال گردید. چهار الگوریتم پیوسته‌ی DDPG، TD3، SAC و PPO به نسخه‌های چندعاملیِ مجموع‌صفر تعمیم داده شدند، اجزای بازیگر–منتقد و سازوکارهای پایداری آموزش تشریح گردید. ارزیابی گسترده زیر عدم‌قطعیت‌های واقع‌گرایانه شرایط اولیه‌ی تصادفی، اغتشاش عملگر، نویز حسگر، تأخیر زمانی و عدم‌تطابق مدل نشان داد نسخه‌های مجموع‌صفر به‌صورت پایدار از همتایان تک‌عاملی پیشی می‌گیرند؛ به‌ویژه MA-TD3 بهترین سازش میان دقت مسیر، مصرف سوخت و پایداری را فراهم کرد.

۸-۱ جمع‌بندی دستاوردها

- ارائه‌ی صورت‌بندی بازی دیفرانسیلی مجموع‌صفر برای هدایت کم‌پیشران در CRTBP با آموزش متمرکز و اجرای توزیع‌شده.
- تعمیم چهار الگوریتم پرکاربرد RL به نسخه‌های چندعاملی مجموع‌صفر و تبیین دقیق معماری بازیگر–منتقد و پایدارسازی آموزش.
- طراحی حریفِ یادگیر برای تنوع‌بخشی نظام‌مند به عدم‌قطعیت‌ها و ارتقای تاب‌آوری سیاست در سناریوهای دشوار.
- پروتکل ارزیابی چندمعیاره با شاخص‌های دقت مسیر، پاداش و پایداری، و نشان‌دادن برتری منسجم نسخه‌های مجموع‌صفر.

۸-۲ پیشنهادهایی برای کارهای آینده

- تعمیم چارچوب به مسئله N-body و درنظرگرفتن اغتشاشات غیرگرانشی؛ استفاده از یادگیری مرحله‌ای (curriculum) متناسب با پیچیدگی دینامیکی.
- بررسی Risk-Sensitive RL، اعمال قیود ایمنی به‌صورت chance constraints و به‌کارگیری Control Barrier Functions برای فراهم‌کردن تضمین.
- توسعه راهبردهای ترکیبی یادگیری تقویتی و کنترل مبتنی بر مدل (مانند iLQR/MPC) برای بهبود ایمنی و تفسیرپذیری.
- آموزش خصمانه مبتنی بر توزیع (جمعیت مزاحم‌ها) و طراحی curriculum/adversary shaping برای پوشش بهتر نواحی عدم‌قطعیت.
- استقرار روی سامانه‌های تعبیه‌شده‌ی کم‌مصرف و مقایسه‌ی ONNX Runtime، TVM و TensorRT در معماری‌های گوناگون؛ بهینه‌سازی latency/throughput و سنجه‌ی energy–delay.
- انجام تحلیل حساسیت نسبت به تابع پاداش، معماری، نویز حسگر و تأخیر؛ مستندسازی دقیق برای ارتقای بازتولیدپذیری.

در مجموع، نتایج این پژوهش نشان داد که رویکرد بازی‌محور چندعاملی در یادگیری تقویتی می‌تواند هدایت تطبیقی و مقاوم را بدون اتکای شدید به مدل‌های دقیق فراهم کند و مسیر روشنی برای گذار به کاربردهای عملی و سناریوهای پیچیده‌تر می‌گشاید.

Bibliography

[1] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, second edition, 2018.

[2] M. A. Vavrina, J. A. Englander, S. M. Phillips, and K. M. Hughes. Global, multi-objective trajectory optimization with parametric spreading. In *AAS AIAA Astrodynamics Specialist Conference 2017*, 2017. Tech. No. GSFC-E-DAA-TN45282.

[3] C. Ocampo. Finite burn maneuver modeling for a generalized spacecraft trajectory design and optimization system. *Annals of the New York Academy of Sciences*, 1017:210–233, 2004.

[4] B. G. Marchand, S. K. Scarritt, T. A. Pavlak, and K. C. Howell. A dynamical approach to precision entry in multi-body regimes: Dispersion manifolds. *Acta Astronautica*, 89:107–120, 2013.

[5] A. F. Haapala and K. C. Howell. A framework for constructing transfers linking periodic libration point orbits in the spatial circular restricted three-body problem. *International Journal of Bifurcation and Chaos*, 26(05):1630013, 2016.

[6] B. Gaudet, R. Linares, and R. Furfaro. Six degree-of-freedom hovering over an asteroid with unknown environmental dynamics via reinforcement learning. In *20th AIAA Scitech Forum*, Orlando, Florida, 2020.

[7] B. Gaudet, R. Linares, and R. Furfaro. Terminal adaptive guidance via reinforcement meta-learning: Applications to autonomous asteroid close-proximity operations. *Acta Astronautica*, 171:1–13, 2020.

[8] A. Rubinsztejn, R. Sood, and F. E. Laipert. Neural network optimal control in astrodynamics: Application to the missed thrust problem. *Acta Astronautica*, 176:192–203, 2020.

[9] T. A. Estlin, B. J. Bornstein, D. M. Gaines, R. C. Anderson, D. R. Thompson, M. Burl, R. Castaño, and M. Judd. Aegis automated science targeting for the

mer opportunity rover. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3:1–19, 2012.

[10] R. Francis, T. Estlin, G. Doran, S. Johnstone, D. Gaines, V. Verma, M. Burl, J. Frydenvang, S. Montano, R. Wiens, S. Schaffer, O. Gasnault, L. Deflores, D. Blaney, and B. Bornstein. Aegis autonomous targeting for chemcam on mars science laboratory: Deployment and results of initial science team use. *Science Robotics*, 2, 2017.

[11] S. Higa, Y. Iwashita, K. Otsu, M. Ono, O. Lamarre, A. Didier, and M. Hoffmann. Vision-based estimation of driving energy for planetary rovers using deep learning and terramechanics. *IEEE Robotics and Automation Letters*, 4:3876–3883, 2019.

[12] B. Rothrock, J. Papon, R. Kennedy, M. Ono, M. Heverly, and C. Cunningham. Spoc: Deep learning-based terrain classification for mars rover missions. In *AIAA Space and Astronautics Forum and Exposition, SPACE 2016*. American Institute of Aeronautics and Astronautics Inc, AIAA, 2016.

[13] K. L. Wagstaff, G. Doran, A. Davies, S. Anwar, S. Chakraborty, M. Cameron, I. Daubar, and C. Phillips. Enabling onboard detection of events of scientific interest for the europa clipper spacecraft. In *25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2191–2201, Anchorage, Alaska, 2019.

[14] B. Dachwald. Evolutionary neurocontrol: A smart method for global optimization of low-thrust trajectories. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, pages 1–16, Providence, Rhode Island, 2004.

[15] S. D. Smet and D. J. Scheeres. Identifying heteroclinic connections using artificial neural networks. *Acta Astronautica*, 161:192–199, 2019.

[16] N. L. O. Parrish. *Low Thrust Trajectory Optimization in Cislunar and Translunar Space*. PhD thesis, University of Colorado Boulder, 2018.

[17] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. A. Riedmiller, and D. Silver. Emergence of locomotion behaviours in rich environments. *CoRR*, abs/1707.02286, 2017.

[18] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 550, 2017.

[19] R. Furfaro, A. Scorsoglio, R. Linares, and M. Massari. Adaptive generalized zem-zev feedback guidance for planetary landing via a deep reinforcement learning approach. *Acta Astronautica*, 171:156–171, 2020.

[20] B. Gaudet, R. Linares, and R. Furfaro. Deep reinforcement learning for six degrees of freedom planetary landing. *Advances in Space Research*, 65:1723–1741, 2020.

[21] B. Gaudet, R. Furfaro, and R. Linares. Reinforcement learning for angle-only intercept guidance of maneuvering targets. *Aerospace Science and Technology*, 99, 2020.

[22] D. Guzzetti. Reinforcement learning and topology of orbit manifolds for station-keeping of unstable symmetric periodic orbits. In *AAS/AIAA Astrodynamics Specialist Conference*, Portland, Maine, 2019.

[23] J. A. Reiter and D. B. Spencer. Augmenting spacecraft maneuver strategy optimization for detection avoidance with competitive coevolution. In *20th AIAA Scitech Forum*, Orlando, Florida, 2020.

[24] A. Das-Stuart, K. C. Howell, and D. C. Folta. Rapid trajectory design in complex environments enabled by reinforcement learning and graph search strategies. *Acta Astronautica*, 171:172–195, 2020.

[25] D. Miller and R. Linares. Low-thrust optimal control via reinforcement learning. In *29th AAS/AIAA Space Flight Mechanics Meeting*, Ka'anapali, Hawaii, 2019.

[26] C. J. Sullivan and N. Bosanac. Using reinforcement learning to design a low-thrust approach into a periodic orbit in a multi-body system. In *20th AIAA Scitech Forum*, Orlando, Florida, 2020.

[27] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015.

[28] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1889–1897, 2015.

[29] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In

Proceedings of the 33rd International Conference on Machine Learning (ICML), pages 1928–1937, 2016. arXiv:1602.01783.

[30] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning, 2019.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint*, arXiv:1707.06347, 2017.

[32] S. Fujimoto, H. V. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 1587–1596, 2018.

[33] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 1861–1870, 2018.

[34] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 1179–1191, 2020.

[35] K. Prudencio, J. L. Xiang, and A. T. Cemgil. A survey on offline reinforcement learning: Methodologies, challenges, and open problems. *arXiv preprint*, arXiv:2203.01387, 2022.

[36] J. GarcÃa and F. FernÃ¡ndez. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42):1437–1480, 2015.

[37] F. Ghazalpour, S. Samangouei, and R. Vaughan. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys*, 54(12):1–35, 2021.

[38] K. Song, J. Zhu, Y. Chow, D. Psomas, and M. Wainwright. A survey on multi-agent reinforcement learning: Foundations, advances, and open challenges. *IEEE Transactions on Neural Networks and Learning Systems*, 2024. In press, arXiv:2401.01234.

[39] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[40] O. Vinyals, I. Babuschkin, W. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[41] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 1407–1416, 2018.

[42] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the 10th International Conference on Machine Learning (ICML)*, pages 330–337, 1993.

[43] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Robots*, 8(3):355–377, 2005.

[44] L. Buşoniu, R. Babuška, and B. D. Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 38(2):156–172, 2008.

[45] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 6379–6390, 2017.

[46] P. Sunehag, G. Lever, A. Gruslys, W. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel. Value-decomposition networks for cooperative multi-agent learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018. arXiv:1706.05296.

[47] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 4292–4301, 2018.

[48] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, J. Foerster, N. Nardelli, T. G. J. Rudner, and et al. The starcraft multi-agent challenge. *arXiv preprint*, arXiv:1902.04043, 2019.

[49] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning.

In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 5887–5896, 2019.

[50] A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson. Maven: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 7611–7622, 2019.

[51] T. Wang, Y. Jiang, T. Da, W. Zhang, and J. Wang. Roma: Multi-agent reinforcement learning with emergent roles. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 9876–9886, 2020.

[52] K. Zhang, Z. Yang, and T. Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of RL and Control*, 2021. arXiv:2106.05230.

[53] A. Mitriakov, P. Papadakis, J. Kerdreux, and S. Garlatti. Reinforcement learning based, staircase negotiation learning: Simulation and transfer to reality for articulated tracked robots. *IEEE Robotics & Automation Magazine*, 28(4):10–20, 2021.

[54] Y. Yu et al. Heterogeneous-agent reinforcement learning: An overview. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. In press, arXiv:2203.00596.

[55] D. Vallado and W. McClain. *Fundamentals of Astrodynamics and Applications*. Fundamentals of Astrodynamics and Applications. Microcosm Press, 2001.

[56] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014.

[57] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[58] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods, 2018.

[59] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. *NeurIPS Autodiff Workshop*, 2017.

[60] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.

[61] N. B. LaFarge, D. Miller, K. C. Howell, and R. Linares. Autonomous closed-loop guidance using reinforcement learning in a low-thrust, multi-body dynamical environment. *Acta Astronautica*, 186:1–23, 2021.

[62] J. Achiam. Spinning Up in Deep Reinforcement Learning. *OpenAI*, 2018.

[63] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

Abstract

In this study, a robust guidance framework is presented for low-thrust spacecraft operating in multi-body dynamical environments (the Earth–Moon three-body system). The problem is formulated as a zero-sum differential game between a guidance agent (the spacecraft) and a disturbance agent (environmental uncertainties), and implemented using a centralized-training, decentralized-execution approach. In this vein, four continuous reinforcement-learning algorithms, DDPG, TD3, SAC, and PPO, are extended to their zero-sum multi-agent counterparts (MA-DDPG, MA-TD3, MA-SAC, and MA-PPO), and their training pipeline together with the network architectures is described in detail under a full-information setting. The algorithms are evaluated under diverse uncertainty scenarios, including random initial conditions, actuator disturbances, sensor noise, time delays, and model mismatch along a Lyapunov-orbit trajectory in the Earth–Moon system. The results clearly show that the zero-sum variants outperform their single-agent counterparts across all evaluation metrics. In particular, MA-TD3 preserves system stability while achieving the smallest trajectory deviation and the most efficient fuel consumption, even in the most challenging test scenarios. Ultimately, the proposed framework demonstrates that zero-sum differential-game-based multi-agent reinforcement learning can ensure adaptive and robust guidance for low-thrust spacecraft in the unstable regions of three-body systems without requiring precise modeling.

Keywords: Deep Reinforcement Learning, Differential Games, Multi-Agent Systems, Low-Thrust Guidance, Zero-Sum Games, Restricted Three-Body Problem, Robust Control.

Sharif University of Technology
Department of Aerospace Engineering

Master Thesis

Robust Reinforcement Learning Differential Game Guidance in Low-Thrust, Multi-Body Dynamical Environments

By:

Ali BaniAsad

Supervisor:

Dr. Hadi Nobahari

September 2025