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 نوبهاری که با کمک‌ها و راهنمایی‌های بی‌دریغشان، بنده را در انجام این پروژه یاری داده‌اند، تشکر و قدردانی 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 (مدل CRTBP زمین–ماه) ارائه شده است. مسئله به‌صورت بازی دیفرانسیلی مجموع‌صفر بین عامل هدایت (فضاپیما) و عامل مزاحم (عدم قطعیت‌های محیطی) فرمول‌بندی شده و با رویکرد آموزش متمرکز–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، MATD3، MASAC و MAPPO) و جریان آموزش آن‌ها همراه با ساختار شبکه‌ها در قالب ارزش–سیاست مشترک تشریح شده است.

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 MATD3 با حفظ پایداری سیستم، کمترین انحراف مسیر و مصرف سوخت بهینه را حتی در سخت‌ترین سناریوهای آزمون از خود نشان داد.

به‌منظور تسهیل استقرار عملی، سیاست‌های آموخته‌شده روی بستر ROS 2 با بهره‌گیری از کوانتیزاسیون INT8 و تبدیل به فرمت ONNX پیاده‌سازی شدند. این بهینه‌سازی‌ها زمان استنتاج را به ۵٫۸ میلی‌ثانیه و مصرف حافظه را به ۹٫۲ مگابایت کاهش داد که به‌ترتیب بهبود ۴۷ درصدی و ۵۳ درصدی نسبت به مدل FP32 را نشان می‌دهد، در حالی‌که چرخه کنترل ۱۰۰ هرتز بدون هیچ‌گونه نقض زمانی حفظ شد.

در مجموع، چارچوب پیشنهادی نشان می‌دهد که یادگیری تقویتی چندعاملی مبتنی بر بازی دیفرانسیلی می‌تواند بدون نیاز به مدل‌سازی دقیق، هدایت تطبیقی و مقاوم فضاپیماهای کم‌پیشران را در نواحی ذاتاً ناپایدار سیستم‌های سه‌جسمی تضمین کند و برای پیاده‌سازی روی سخت‌افزار در حلقه آماده باشد.

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

فهرست مطالب

فهرست جداول

فهرست تصاویر

فهرست الگوریتم‌ها

فصل ۱

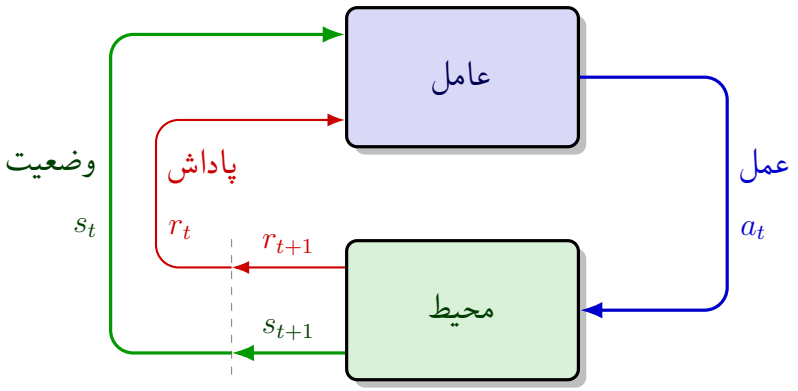
یادگیری تقویتی

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

۱-۱ مفاهیم اولیه

دو بخش اصلی یادگیری تقویتی[۱] شامل عامل[۲] و محیط[۳] 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[۴] 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[۵] نام دارد. یادگیری تقویتی به روش‌هایی گفته می‌شود که در آن‌ها عامل رفتارهای مناسب برای رسیدن به هدف خود را می‌آموزد. در شکل ۱-۱ تعامل بین محیط و عامل نشان داده شده‌است.

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



شکل ۱-۱: حلقه تعامل عامل و محیط

۱-۱-۱ 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.

۱-۱-۲ 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.

۱-۱-۳ 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:

$$a_t = \mu(s_t) \tag{۱-۱}$$

$$a_t \sim \pi(\cdot|s_t) \tag{۱-۲}$$

که زیروند t بیانگر زمان است. در یادگیری تقویتی عمیق از سیاست‌های پارامتری‌شده استفاده 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 استفاده کرد. در این پژوهش پارامترهای سیاست با θ نشان داده شده‌است و سپس نماد آن به‌عنوان زیروند سیاست مانند معادله (۱-۳) نشان داده شده‌است.

$$\begin{aligned} a_t &= \mu_\theta(s_t) \\ a_t &\sim \pi_\theta(\cdot|s_t) \end{aligned} \tag{۱-۳}$$

۱-۱-۴ مسیر

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

$$\tau = (s_0, a_0, s_1, a_1, \cdots) \tag{۱-۴}$$

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.

$$s_{t+1} = f(s_t, a_t) \tag{۱-۵}$$

۱-۱-۵ 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.

$$r_t = R(s_t, a_t, s_{t+1}) \tag{۱-۶}$$

[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 (۱-۷) نشان داده شده‌است.

$$R(\tau) = \sum_{t=0}^{T} r_t \tag{۱-۷}$$

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 (۱-۸) شامل یک فاکتور تنزیل[۲۰] با نماد γ است که عددی بین صفر و یک است.

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t \tag{۱-۸}$$

۱-۱-۶ 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:

$$V^\pi(s) = \mathop{\mathbb{E}}_{\tau\sim\pi}[R(\tau)|s_0 = s] \tag{۱-۹}$$

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:

$$Q^{\pi}(s,a) = \underset{\tau\sim\pi}{\mathbb{E}}[R(\tau)|s_0 = s, a_0 = a] \qquad (۱۰-۱)$$

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:

$$V^*(s) = \max_{\pi}(V^{\pi}(s)) \qquad (۱۱-۱)$$

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:

$$Q^*(s,a) = \max_{\pi}(Q^{\pi}(s,a)) \qquad (۱۲-۱)$$

۱-۱-۷ 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:

$$V^{\pi}(s) = \underset{\substack{a\sim\pi \\ s'\sim P}}{\mathrm{E}}[r(s,a) + \gamma V^{\pi}(s')] \qquad (۱۳-۱)$$

$$Q^{\pi}(s,a) = r(s,a) + \underset{\substack{a\sim\pi \\ s'\sim P}}{\mathrm{E}}\left[\gamma \underset{a'\sim\pi}{\mathrm{E}}[Q^{\pi}(s',a')]\right] \qquad (۱۴-۱)$$

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:

$$V^*(s) = \max_a \mathop{\mathrm{E}}_{s' \sim P} [r(s,a) + \gamma V^*(s')] \qquad (۱۵-۱)$$

$$Q^*(s,a) = r(s,a) + \mathop{\mathrm{E}}_{s' \sim P} \left[\gamma \max_{a'} Q^*(s',a') \right] \qquad (۱۶-۱)$$

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.

۱-۱-۸ 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

۱-۲ 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 (۱-۱۷) به دست آورد.

$$a^*(s) = \arg\max_a Q^*(s,a) \tag{۱-۱۷}$$

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.

۱-۲-۱ یادگیری Q در DDPG

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

$$Q^*(s,a) = r(s,a) + \underset{s'\sim P}{\mathrm{E}}\left[\gamma \max_{a'} Q^*(s',a')\right] \tag{۱-۱۸}$$

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

$$L(\phi, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}} \left[\left(Q_\phi(s,a) - \left(r + \gamma(1-d) \max_{a'} Q_\phi(s',a') \right) \right)^2 \right] \qquad (۱۹-۱)$$

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] تجربه‌شده استفاده 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.

$$r + \gamma(1-d) \max_{a'} Q_\phi(s',a') \qquad (۲۰-۱)$$

در هنگام کمینه کردن تابع خطای میانگین مربعات بلمن، سعی شده‌است تا تابع Q شبیه‌تر به هدف یعنی رابطه (۱-۲۰) 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.

$$\phi_{\text{targ}} \leftarrow \rho\phi_{\text{targ}} + (1-\rho)\phi \qquad (۲۱-۱)$$

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 (۱-۲۲) با استفاده از کاهش گرادیان تصادفی[۳۵] انجام می‌شود.

$$L(\phi, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\text{E}} \left[\left(Q_\phi(s,a) - \left(r + \gamma(1-d) Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s')) \right) \right)^2 \right] \qquad (۱-۲۲)$$

۱-۲-۲ 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.

$$\max_\theta \underset{s\sim\mathcal{D}}{\text{E}} \left[Q_\phi(s, \mu_\theta(s)) \right] \qquad (۱-۲۳)$$

۱-۲-۳ 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.

۱-۲-۴ 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:

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

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

۱-۳ 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 TD3 (Twin Delayed DDPG) 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 (۱-۲۴) آورده شده‌است.

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{i,\text{targ}}}(s', a'(s')) \tag{۱-۲۴}$$

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

$$L(\phi_1, \mathcal{D}) = \mathop{\mathrm{E}}_{(s,a,r,s',d)\sim\mathcal{D}} \left(Q_{\phi_1}(s, a) - y(r, s', d) \right)^2 \tag{۱-۲۵}$$

$$L(\phi_2, \mathcal{D}) = \mathop{\mathrm{E}}_{(s,a,r,s',d)\sim\mathcal{D}} \left(Q_{\phi_2}(s, a) - y(r, s', d) \right)^2 \tag{۱-۲۶}$$

- 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:

$$\max_{\theta} \underset{s\sim\mathcal{D}}{\mathrm{E}} \left[Q_{\phi_1}(s, \mu_\theta(s))\right] \tag{۲۷-۱}$$

۱-۳-۱ 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.

۱-۳-۲ 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 ($\mathcal{D}$)

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

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 j در 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:

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

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

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

۱۲: اگر باقیمانده j بر تاخیر سیاست برابر 0 باشد :

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

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

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

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

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

۱-۴ 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.

۱-۴-۱ 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:

$$H(X) = \mathrm{E}_{x\sim p}\left[-\log p(x)\right] \qquad (۲۸-۱)$$

۱-۴-۲ 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

$$\pi^* = \arg\max_{\pi} \mathop{\mathrm{E}}_{\tau\sim\pi} \sum_{t=0}^{\infty} \gamma^t \Big(R(s_t, a_t, s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right) \Big) \tag{۲۹-۱}$$

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

۳-۴-۱ تابع ارزش در SAC

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

$$V^\pi(s) = \mathop{\mathrm{E}}_{\tau\sim\pi} \left[\sum_{t=0}^{\infty} \gamma^t \Big(R(s_t, a_t, s_{t+1}) + \alpha H\left(\pi(\cdot|s_t)\right) \Big) \middle| s_0 = s \right] \tag{۳۰-۱}$$

۴-۴-۱ 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.

$$Q^\pi(s, a) = \mathop{\mathrm{E}}_{\tau\sim\pi} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) + \alpha \sum_{t=1}^{\infty} \gamma^t H\left(\pi(\cdot|s_t)\right) \middle| s_0 = s, a_0 = a \right] \tag{۳۱-۱}$$

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

$$V^\pi(s) = \mathop{\mathrm{E}}_{a\sim\pi} \left[Q^\pi(s, a) \right] + \alpha H\left(\pi(\cdot|s)\right) \tag{۳۲-۱}$$

۵-۴-۱ معادله بلمن در SAC

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

$$Q^\pi(s, a) = \mathop{\mathrm{E}}_{\substack{s'\sim P \\ a'\sim\pi}} \left[R(s, a, s') + \gamma \left(Q^\pi(s', a') + \alpha H\left(\pi(\cdot|s')\right) \right) \right] \tag{۳۳-۱}$$

$$= \mathop{\mathrm{E}}_{s'\sim P} \left[R(s, a, s') + \gamma V^\pi(s') \right] \tag{۳۴-۱}$$

[48] Trade-Off

۱-۴-۶ یادگیری Q

با درنظرگرفتن موارد اشاره‌شده، یادگیری Q در SAC با کمینه‌کردن تابع خطای میانگین مربعات بلمن (MSBE) یعنی معادله (۱-۳۵) با استفاده از کاهش گرادیان انجام می‌شود.

$$L(\phi_i, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}} \left[\left(Q_{\phi_i}(s,a) - y(r,s',d) \right)^2 \right] \tag{۱-۳۵}$$

در معادله (۱-۳۵) تابع هدف برای روش یادگیری تقویتی SAC به صورت زیر تعریف می‌شود.

$$y(r,s',d) = r + \gamma(1-d)\left(\min_{j=1,2} Q_{\phi_{\text{targ},j}}(s',\tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s') \right), \qquad \tilde{a}' \sim \pi_\theta(\cdot|s') \tag{۱-۳۶}$$

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.

۱-۴-۷ 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.

$$V^\pi(s) = \underset{a\sim\pi}{\mathrm{E}} \left[Q^\pi(s,a) \right] + \alpha H\left(\pi(\cdot|s)\right) \tag{۱-۳۷}$$

$$= \underset{a\sim\pi}{\mathrm{E}} \left[Q^\pi(s,a) - \alpha \log \pi(a|s) \right] \tag{۱-۳۸}$$

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:

$$\tilde{a}_\theta(s,\xi) = \tanh\left(\mu_\theta(s) + \sigma_\theta(s) \odot \xi\right), \qquad \xi \sim \mathcal{N} \tag{۱-۳۹}$$

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.

$$\underset{a\sim\pi_\theta}{\mathrm{E}}[Q^{\pi_\theta}(s,a)-\alpha\log\pi_\theta(a|s)]=\underset{\xi\sim\mathcal{N}}{\mathrm{E}}[Q^{\pi_\theta}(s,\tilde{a}_\theta(s,\xi))-\alpha\log\pi_\theta(\tilde{a}_\theta(s,\xi)|s)] \qquad (۴۰-۱)$$

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:

$$\max_\theta \underset{\substack{s\sim\mathcal{D}\\ \xi\sim\mathcal{N}}}{\mathrm{E}}\left[\min_{j=1,2} Q_{\phi_j}(s,\tilde{a}_\theta(s,\xi))-\alpha\log\pi_\theta(\tilde{a}_\theta(s,\xi)|s)\right] \qquad (۴۱-۱)$$

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

۱-۴-۸ 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

۹-۴-۱ 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 $(\mathcal{D})$

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

۲: تا وقتی همگرایی رخ دهد:

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

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 j در 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:

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

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

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

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

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

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

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

۵-۱ عامل بهینه‌سازی سیاست مجاور

الگوریتم بهینه‌سازی سیاست مجاور[۵۲] یک الگوریتم بهینه‌سازی سیاست مبتنی بر گرادیان است که برای حل مسائل کنترل مسئله‌های یادگیری تقویتی استفاده می‌شود. این الگوریتم از الگوریتم TRPO[۵۳] 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 داری 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[۵۴] است، مشابه روشی که در الگوریتم TRPO استفاده شده‌است. با این حال، به جای اعمال یک محدودیت سخت[۵۵]، 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[۵۶] 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

۱-۵-۱ 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:

$$\theta_{k+1} = \arg\max_{\theta} \underset{s,a\sim\pi_{\theta_k}}{\mathrm{E}} [L(s, a, \theta_k, \theta)] \tag{۴۲-۱}$$

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:

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a),\ \ \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_k}}(s, a)\right) \tag{۴۳-۱}$$

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 (۴۳-۱) به معادله تغیر داده شده‌است.

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a),\ \ g(\epsilon, A^{\pi_{\theta_k}}(s, a))\right) \tag{۴۴-۱}$$

که تابع g به صورت زیر تعریف شده‌است.

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geqslant 0 \\ (1-\epsilon)A & A < 0 \end{cases} \tag{۴۵-۱}$$

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

۱-۵-۲ 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.

۱-۵-۳ 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 ۲

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 ۲-۱ مفاهیم اولیه‌ی یادگیری تقویتی چندعاملی معرفی می‌شوند، سپس در بخش ۲-۲ انواع بازی‌ها و تعادل نش مورد بررسی قرار می‌گیرند. الگوریتم‌های مختلف یادگیری تقویتی چندعاملی شامل MA-DDPG در بخش ۲-۳، MA-TD3 در بخش ۲-۴، MA-SAC در بخش ۲-۵ و MA-PPO در بخش ۲-۶ معرفی و بررسی شده‌اند.

۲-۱ 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 ۲-۱ 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

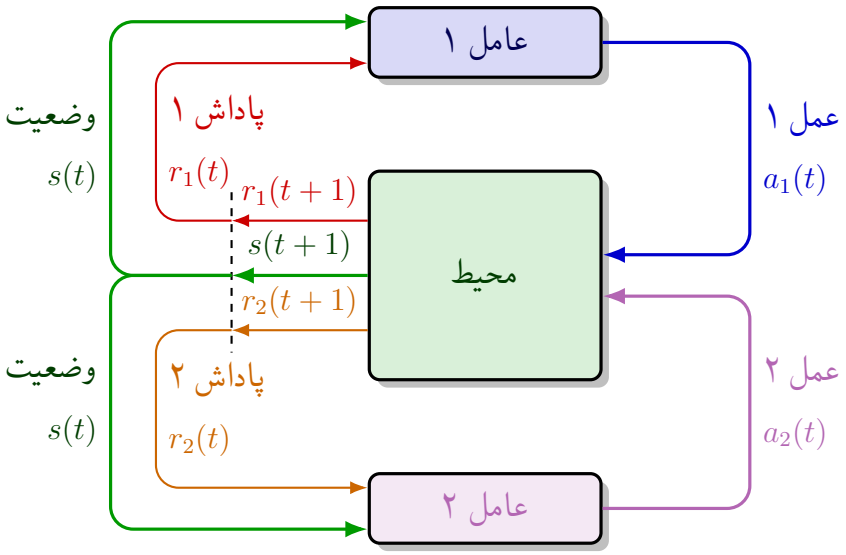
[1] Multi-Agent
[2] Nash Equilibrium
[3] Multi-Agent Reinforcement Learning (MARL)

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 نظرگرفته 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 ۲-۱: حلقه تعامل عامل‌های یادگیری تقویتی چند عاملی با محیط

۲-۲ 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.

۲-۲-۱ تعادل نش

تعادل نش[۵] یکی از بنیادی‌ترین مفاهیم در نظریه‌ی بازی‌ها است که توسط جان نش در سال ۱۹۵۰ معرفی شد. این مفهوم به ترکیب[۶] 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:

$$u_i(\pi_i^*, \pi_{-i}^*, \boldsymbol{s}) \geqslant u_i(\pi_i, \pi_{-i}^*, \boldsymbol{s}) \qquad (۱-۲)$$

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: $u_i(\pi_i, \pi_{-i}, \boldsymbol{s}) = V_i^{\pi_i, \pi_{-i}}(\boldsymbol{s})$.

۲-۲-۲ بازی مجموع صفر

بازی‌های مجموع صفر[۷] 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 تابعِ 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:

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

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:

(۳-۲)

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

- **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:

$$V_1^*(s) = \max_{\pi_1} \min_{\pi_2} V_1^{(\pi_1,\pi_2)}(s), \qquad (۴-۲)$$

$$V_2^*(s) = \max_{\pi_2} \min_{\pi_1} V_2^{(\pi_1,\pi_2)}(s). \qquad (۵-۲)$$

[7] Zero-Sum Games

و به‌صورت تابع ارزش–عمل:

$$Q_1^*(s,a_1,a_2) = \max_{\pi_1} \min_{\pi_2} Q_1^{(\pi_1,\pi_2)}(s,a_1,a_2), \tag{۶-۲}$$

$$Q_2^*(s,a_1,a_2) = \max_{\pi_2} \min_{\pi_1} Q_2^{(\pi_1,\pi_2)}(s,a_1,a_2). \tag{۷-۲}$$

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.

۲-۳ 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).

۲-۳-۱ 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.

MA-DDPG با استفاده از رویکرد آموزش متمرکز، اجرای غیرمتمرکز[۱۰] 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.

۲-۳-۲ 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:

[8] Multi-Agent Deep Deterministic Policy Gradient (MA-DDPG)
[9] Non-stationary
[10] Centralized Training, Decentralized Execution

- **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 پایدار کردن 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.

۲-۳-۳ 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:

$$L(\phi_i, \mathcal{D}) = \underset{(\boldsymbol{o}, \boldsymbol{a}, r_i, \boldsymbol{o}', d) \sim \mathcal{D}}{\mathrm{E}} \left[\left(Q_{\phi_i}(o_i, a_1, a_2) - y_i \right)^2 \right] \tag{۸-۲}$$

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:

$$y_i = r_i + \gamma(1 - d) Q_{\phi_{i,\text{targ}}}(o_i', \mu_{\theta_{1,\text{targ}}}(o_1'), \mu_{\theta_{2,\text{targ}}}(o_2')) \tag{۹-۲}$$

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:

$$\max_{\theta_i} \underset{\boldsymbol{o} \sim \mathcal{D}}{\mathrm{E}} \left[Q_{\phi_i}(o_i, \mu_{\theta_i}(o_i), \mu_{\theta_{-i}}(o_{-i})) \right] \tag{۱۰-۲}$$

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 پایدارکردن 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.

۲-۳-۴ 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:

$$a_i = \text{clip}(\mu_{\theta_i}(o_i) + \epsilon_i, a_{\text{Low}}, a_{\text{High}}) \qquad (۱۱-۲)$$

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

۲-۳-۵ شبه‌کد 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:

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

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

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

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

۲-۳-۶ 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.

۲-۴ 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.

۲-۴-۱ 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.

۲-۴-۲ 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.

۲-۴-۳ 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:

$$L(\phi_{i,j}, \mathcal{D}) = \underset{(\boldsymbol{o},\boldsymbol{a},r_i,\boldsymbol{o'},d)\sim\mathcal{D}}{\mathrm{E}} \left[\left(Q_{\phi_{i,j}}(o_i, a_1, a_2) - y_i \right)^2 \right] \qquad (۱۲-۲)$$

که در آن y_i هدف برای عامل i است:

$$y_i = r_i + \gamma(1-d) \min_{j=1,2} Q_{\phi_{i,j,\text{targ}}}(o'_i, \mu_{\theta_{1,\text{targ}}}(o'_1), \mu_{\theta_{2,\text{targ}}}(o'_2)) \qquad (۱۳-۲)$$

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:

$$\max_{\theta_i} \mathop{\mathrm{E}}_{\boldsymbol{o}\sim\mathcal{D}} \left[Q_{\phi_{i,1}}\left(o_i, \mu_{\theta_i}(o_i), \mu_{\theta_{-i}}(o_{-i})\right)\right] \qquad (۱۴-۲)$$

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.

۴-۴-۲ اکتشاف در MA-TD3

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

$$a_i = \mathrm{clip}(\mu_{\theta_i}(o_i) + \epsilon_i, a_{\mathrm{Low}}, a_{\mathrm{High}}) \qquad (۱۵-۲)$$

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

۵-۴-۲ 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 $(\mathcal{D})$

۱: 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 $j \in \{1,2\}$

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 $\epsilon_i \sim \mathcal{N}(0, \sigma_i)$ 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 j در 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.

۱۲: اهداف را محاسبه کنید:

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

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

۱۳: توابع Q را با نزول گرادیان به‌روزرسانی کنید:

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

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

۱۴: اگر باقیمانده j بر تاخیر سیاست برابر 0 باشد:

۱۵: سیاست‌ها را با صعود گرادیان به‌روزرسانی کنید:

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

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

۱۶: شبکه‌های هدف را به‌روزرسانی کنید:

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

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

۲-۴-۶ 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.

۲-۵ عامل عملگر نقاد نرم چندعاملی

عامل عملگر نقاد نرم دوعاملی[۱۲] 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.

۲-۵-۱ 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.

۲-۵-۲ 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.

۲-۵-۳ 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:

$$L(\phi_{i,j}, \mathcal{D}) = \underset{(\boldsymbol{o}, \boldsymbol{a}, r_i, \boldsymbol{o}', d) \sim \mathcal{D}}{\mathrm{E}} \left[\left(Q_{\phi_{i,j}}(o_i, a_1, a_2) - y_i \right)^2 \right] \qquad (۱۶-۲)$$

که در آن y_i هدف برای عامل i است:

$$y_i = r_i + \gamma(1-d)\Big(\min_{j=1,2} Q_{\phi_{i,j,\text{targ}}}(o'_i, \tilde{a}'_1, \tilde{a}'_2) - \alpha_i \log \pi_{\theta_i}(\tilde{a}'_i|o'_i)\Big) \qquad (۱۷-۲)$$

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:

$$\max_{\theta_i} \mathop{\mathrm{E}}_{\boldsymbol{o}\sim\mathcal{D}} \left[\min_{j=1,2} Q_{\phi_{i,j}}(o_i, \tilde{a}_i, a_{-i}) - \alpha_i \log \pi_{\theta_i}(\tilde{a}_i|o_i)\right] \qquad (۱۸-۲)$$

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

$$\tilde{a}_{i,\theta_i}(o_i, \xi_i) = \tanh\left(\mu_{\theta_i}(o_i) + \sigma_{\theta_i}(o_i) \odot \xi_i\right), \quad \xi_i \sim \mathcal{N}(0, I) \qquad (۱۹-۲)$$

شبکه‌های هدف

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

$$\phi_{i,j,\text{targ}} \leftarrow \rho\phi_{i,j,\text{targ}} + (1-\rho)\phi_{i,j} \quad \text{برای} j = 1,2 \qquad (۲۰-۲)$$

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:

$$\min_{\alpha_i} \mathop{\mathrm{E}}_{\boldsymbol{o}\sim\mathcal{D}, \tilde{a}_i\sim\pi_{\theta_i}} \left[-\alpha_i\Big(\log \pi_{\theta_i}(\tilde{a}_i|o_i) + H_{\text{target}}\Big)\right] \qquad (۲۱-۲)$$

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

۲-۵-۴ 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:

$$a_i \sim \pi_{\theta_i}(\cdot|o_i) \tag{۲۲-۲}$$

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

۲-۵-۵ 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 $(\mathcal{D})$

۱: پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید:

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

۲: تا وقتی همگرایی رخ دهد:

۳: مشاهدات (o_1, o_2) را دریافت کنید

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

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.

۱۲: اهداف را محاسبه کنید:

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

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

۱۳: توابع Q را با نزول گرادیان به‌روزرسانی کنید:

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

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

۱۴: سیاست‌ها را با صعود گرادیان به‌روزرسانی کنید:

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

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

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

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

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

۱۶: شبکه‌های هدف را به‌روزرسانی کنید:

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

۲-۵-۶ 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.

۲-۶ عامل بهینه‌سازی سیاست مجاور چندعاملی

عامل بهینه‌سازی سیاست مجاور دوعاملی[۱۳] 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.

۲-۶-۱ 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.

۲-۶-۲ 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.

۲-۶-۳ 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:

$$\mathcal{D}_k = \{(o_1^t, o_2^t, a_1^t, a_2^t, r_1^t, r_2^t, o_1^{t+1}, o_2^{t+1})\} \qquad (۲۳-۲)$$

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:

$$\hat{A}_i^t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{i,t+l} \tag{۲۴-۲}$$

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:

$$\max_{\theta_i} \mathop{\mathrm{E}}_{(o_i,a_i)\sim\mathcal{D}_k} \left[\min \left(\frac{\pi_{\theta_i}(a_i|o_i)}{\pi_{\theta_{i,k}}(a_i|o_i)} \hat{A}_i, \quad \text{clip} \left(\frac{\pi_{\theta_i}(a_i|o_i)}{\pi_{\theta_{i,k}}(a_i|o_i)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_i \right) \right] \tag{۲۵-۲}$$

یا با استفاده از همان فرمول‌بندی ساده‌تر:

$$\max_{\theta_i} \mathop{\mathrm{E}}_{(o_i,a_i)\sim\mathcal{D}_k} \left[\min \left(\frac{\pi_{\theta_i}(a_i|o_i)}{\pi_{\theta_{i,k}}(a_i|o_i)} \hat{A}_i, \quad g(\epsilon, \hat{A}_i) \right) \right] \tag{۲۶-۲}$$

که تابع g به صورت زیر تعریف شده‌است:

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geqslant 0 \\ (1-\epsilon)A & A < 0 \end{cases} \tag{۲۷-۲}$$

به‌روزرسانی منتقد

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

$$\min_{\phi_i} \mathop{\mathrm{E}}_{(o_i,\hat{R}_i)\sim\mathcal{D}_k} \left[\left(V_{\phi_i}(o_i) - \hat{R}_i \right)^2 \right] \tag{۲۸-۲}$$

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

۲-۶-۴ 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:

$$a_i \sim \pi_{\theta_i}(\cdot|o_i) \tag{۲۹-۲}$$

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.

۲-۶-۵ 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:

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

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

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

۲-۶-۶ 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.

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

This thesis proposes a robust guidance framework for low-thrust spacecraft operating in multi-body dynamical environments modeled by the Earth–Moon circular restricted three-body problem (CRTBP). The guidance task is cast as a zero-sum differential game between a controller agent (spacecraft) and an adversary agent (environmental disturbances), implemented under a centralized-training/ decentralized-execution paradigm. Four continuous-control reinforcement-learning algorithms—DDPG, TD3, SAC, and PPO—are extended to their multi-agent zero-sum counterparts (MA-DDPG, MATD3, MASAC, MAPPO); their actor–critic network structures and training pipelines are detailed.

The policies are trained and evaluated on transfers to the Earth–Moon lyapunov orbit under five uncertainty scenarios: random initial states, actuator perturbations, sensor noise, communication delays, and model mismatch. Zero-sum variants consistently outperform their single-agent baselines, with MATD3 delivering the best trade-off between trajectory accuracy and propellant consumption while maintaining stability in the harshest conditions.

The results demonstrate that the proposed multi-agent, game-theoretic reinforcement-learning framework enables adaptive and robust low-thrust guidance in unstable three-body regions without reliance on precise dynamics models, and is ready for hardware-in-the-loop implementation.

Keywords: Deep Reinforcement Learning; Differential Game; Multi-Agent; Low-Thrust Guidance; Three-Body Problem; Robustness.

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