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

دی ۱۴۰۳


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 مشاور: استاد مشاور امضاء:

استاد مدعو: استاد ممتحن 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، از یک روش مبتنی بر نظریه بازی[1] به‌منظور کنترل وضعیت استند سه درجه آزادی چهارپره استفاده شده‌است. در این روش بازیکن اول سعی در ردگیری ورودی مطلوب می‌کند و بازیکن دوم با ایجاد اغتشاش سعی در ایجاد خطا در ردگیری بازیکن اول می‌کند. در این روش انتخاب حرکت با استفاده از تعادل نش[2] که با فرض بدترین حرکت دیگر بازیکن است، انجام می‌شود. این روش نسبت به اغتشاش ورودی و همچنین نسبت به عدم قطعیت مدل‌سازی می‌تواند مقاوم باشد. برای ارزیابی عملکرد این روش ابتدا شبیه‌سازی‌هایی در محیط سیمولینک انجام شده‌است و سپس، با پیاده‌سازی روی استند سه درجه آزادی صحت عملکرد کنترل‌کننده تایید شده‌است.

کلیدواژه‌ها: چهارپره، بازی دیفرانسیلی، نظریه بازی، تعادل نش، استند سه درجه آزادی، مدل‌مبنا، تنظیم‌کننده مربعی خطی

[1] Game Theory
[2] Nash Equilibrium

فهرست مطالب

فهرست جداول

فهرست تصاویر

فهرست الگوریتم‌ها

فصل ۱

مقدمه

۱-۱ انگیزه پژوهش

۱-۲ تعریف مسئله

در سال‌های اخیر، پیشرفت‌های فناوری در زمینه‌های مختلف، از جمله کنترل پرواز، پردازش سیگنال و هوش مصنوعی، به افزایش کاربردهای ماهواره با پیشران کم در منظومه زمین–ماه کمک کرده است. ماهواره با پیشران کم می‌تواند برای تعقیب ماهواره‌ها، انتقال مداری و استقرار ماهواره‌ها استفاده شود. روش‌های هدایت بهینه قدیمی جهت کنترل ماهواره‌ها اغلب نیازمند فرضیات ساده‌کننده، منابع محاسباتی فراوان و شرایط اولیه مناسب هستند. الگوریتم‌های مبتنی بر یادگیری تقویتی این توانایی را دارند که بدون مشکلات اشاره‌شده هدایت ماهواره را انجام دهند. به همین دلیل، این الگوریتم‌ها می‌توانند امکان محاسبات درونی (On-board Computing) را فراهم می‌کنند.

۱-۳ اهداف و نوآوری

۱-۴ محتوای گزارش

فصل ۲

پیشینه پژوهش

۲-۱ ماموریت‌های بین مداری

هدایت فضاپیماها معمولاً با استفاده از ایستگاه‌های زمینی انجام می‌شود. با این حال، این تکنیک‌ها دارای محدودیت‌هایی از جمله حساسیت به قطع ارتباطات، تاخیرهای 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 تاخیرهای 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 [۱] یا برنامه‌نویسی غیرخطی برای تولید مسیرها و فرمان‌های کنترلی بهینه استفاده کنند. با این حال، این روش‌ها معمولا حجم محاسباتی زیادی دارند و برای استفاده درون‌سفینه نامناسب هستند [۲]. 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 به‌طور کلی 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 [۳، ۴] ایجاد می‌کنند.

شبکه‌های عصبی ویژگی‌های جذابی برای فعال‌سازی 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 هدایت شبکه‌های عصبی در مطالعاتی مانند فرود بر سیارات [۵]، عملیات نزدیکی به سیارات [۶] و کنترل فضاپیما با پیشران ازدست‌رفته [۷] نشان داده شده است. 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 مریخی نبرد 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 فعال دارد [۸]. در کامپیوتر پرواز اصلی، فرآیند دقت افزایی (Refinement Process) نیاز به ۹۴ تا ۹۶ ثانیه دارد [۹]، 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 [۱۰، ۱۱] و شناسایی عیب برای Europa Clipper [۱۲] می‌شود. الگوریتم‌های یادگیری ماشین پتانسیلی برای سهم مهمی در مأموریت‌های اتوماسیون آینده 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 (NN) در بهینه‌سازی مسیرهای رانشگر کم‌پیشران استفاده شده است [۱۳]. پژوهش‌های جدید شامل شناسایی انتقال‌های هتروکلینیک [۱۴]، اصلاح مسیر رانشگر کم‌پیشران [۱۵] و تجزیه و تحلیل مشکلات ازدست‌رفتن رانشگر [۷] 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.

در سال‌های اخیر، قابلیت یادگیری تقویتی (RL) در دستیابی به عملکرد بهینه در دامنه‌هایی با ابهام محیطی قابل توجه، به اثبات رسیده است [۱۶، ۱۷]. هدایت انجام‌شده توسط RL را می‌توان به‌صورت گسترده بر اساس فاز پرواز دسته‌بندی کرد. مسائل فرود [۱۸، ۱۹] و عملیات در نزدیکی اجسام کوچک [۶، ۵]، از حوزه‌های پژوهشی هستند که از RL استفاده می‌کنند. تحقیقات دیگر شامل مواجهه تداخل خارجی جوی [۲۰]، نگهداری ایستگاهی [۲۱] و هدایت به‌صورت جلوگیری از شناسایی [۲۲] است. مطالعاتی که فضاپیماهای رانشگر کم‌پیشران را در یک چارچوب دینامیکی چند بدنی با استفاده از RL انجام‌شده‌است، شامل طراحی انتقال با استفاده از Q-learning [۲۳]، Proximal Policy Optimization [۲۴] و هدایت نزدیکی مدار [۲۵] است.

۲-۲ بازی دیفرانسیلی

بازی دیفرانسیلی زیر مجموعه‌ای از نظریه بازی است. نظریه بازی با استفاده از مدل‌های ریاضی به تحلیل روش‌های همکاری یا رقابت موجودات منطقی و هوشمند می‌پردازد. نظریه بازی، شاخه‌ای از ریاضیات کاربردی است که در علوم اجتماعی و به ویژه در اقتصاد، زیست‌شناسی، مهندسی، علوم سیاسی، روابط بین‌الملل، علوم رایانه، بازاریابی و فلسفه مورد استفاده قرار می‌گیرد. نظریه بازی در تلاش است تا به وسیله‌ی ریاضیات، رفتار را در شرایط راهبردی یا در یک بازی که در آن موفقیت فرد در انتخاب کردن، وابسته به انتخاب دیگران می‌باشد، برآورد کند.. در سال ۱۹۹۴ جان فوربز نش به همراه جان هارسانی و راینهارد سیلتن به خاطر مطالعات خلاقانه‌ی خود در زمینه‌ی نظریه بازی، برنده‌ی جایزه نوبل اقتصاد شدند. در سال‌های پس از آن نیز بسیاری از برندگان جایزه‌ی نوبل اقتصاد از میان متخصصین نظریه بازی انتخاب شدند. آخرین آن‌ها، ژان تیرول فرانسوی است که در سال ۲۰۱۴ این جایزه را کسب کرد [۲۶].

پژوهش‌ها در این زمینه اغلب بر مجموعه‌ای از راهبردهای شناخته شده به عنوان تعادل در بازی‌ها استوار است. این راهبردها به‌طور معمول از قواعد عقلانی به نتیجه می‌رسند. مشهورترین تعادل‌ها، تعادل نش است. تعادل نش در بازی‌هایی کاربرد دارد در آن فرض شده‌است که هر بازیکن به راهبرد تعادل دیگر بازیکنان آگاه است. بر اساس نظریه‌ی تعادل نش، در یک بازی که هر بازیکن امکان انتخاب‌های گوناگون دارد اگر بازیکنان به روش منطقی راهبردهای خود را انتخاب کنند و به دنبال حداکثر سود در بازی باشند، دست کم یک راهبرد برای به دست آوردن بهترین نتیجه برای هر بازیکن وجود دارد و چنانچه بازیکن راهکار دیگری را انتخاب کند، نتیجه‌ی بهتری به دست نخواهد آورد.

۲-۳ یادگیری تقویتی

۲-۴ یادگیری تقویتی چندعاملی

۲-۴-۱ ماموریت‌های بین‌مداری

تعریف ماموریت‌های بین‌مداری

اصول طراحی و بهینه‌سازی ماموریت‌ها و چالش‌های ارتباطی و کنترلی.

مدل‌سازی مسیرهای بین‌مداری

رویکردهای عددی برای بهینه‌سازی مسیر و استفاده از یادگیری ماشینی در برنامه‌ریزی مسیر.

ایمنی در ماموریت‌های بین‌مداری

تضمین پایداری مسیرها در حضور عدم قطعیت و مدیریت ریسک برخورد.

۲-۴-۲ بازی دیفرانسیلی

اصول بازی دیفرانسیلی

تعریف بازی دیفرانسیلی، مفاهیم پایه و کاربردهای آن در تعاملات چندعاملی.

بازی دیفرانسیلی با جمع صفر

مدل‌سازی مسائل با جمع صفر و کاربردهای آن در مسائل دفاعی و نظارتی.

تکنیک‌های حل بازی‌های دیفرانسیلی

روش‌های تحلیلی و استفاده از الگوریتم‌های یادگیری تقویتی.

۲-۴-۳ یادگیری تقویتی

مفاهیم یادگیری تقویتی

سیاست، پاداش، و به‌روزرسانی ارزش‌ها.

الگوریتم‌های پیشرفته یادگیری تقویتی

الگوریتم‌های Temporal-Difference Carlo، Monte و Q-Network. Deep

ایمنی در یادگیری تقویتی

یادگیری ایمن، محدودیت‌ها و تضمین‌ها در سیاست‌های یادگیری.

۲-۴-۴ یادگیری تقویتی چندعاملی

تعریف و اهمیت یادگیری تقویتی چندعاملی

مفهوم همکاری و رقابت در محیط‌های چندعاملی و اهمیت ایمنی.

بازی‌های چندعاملی در یادگیری تقویتی

بازی‌های جمع صفر، تعادل نش و کاربردهای آن‌ها.

چالش‌های یادگیری تقویتی چندعاملی

مدیریت تعارضات، تضمین پایداری و مشکلات همگرایی.

ایمنی در یادگیری تقویتی چندعاملی

تکنیک‌های تضمین ایمنی و کاربرد ایمنی در تعاملات حساس.

الگوریتم‌های یادگیری تقویتی چندعاملی

الگوریتم‌های همکاری مانند MADDPG و یادگیری توزیع‌شده با تضمین ایمنی.

کاربرد MARL در مسائل نظری

حل مسائل بازی‌های جمع صفر و مدل‌سازی مسائل رقابتی و همکاری.

کاربرد MARL در ماموریت‌های فضایی

هماهنگی میان ماهواره‌ها، تخصیص منابع و حل مسائل ترکیبی در کاوش سیارات.

۲-۵ یادگیری تقویتی چندعاملی

۲-۵-۱ تعریف و اهمیت یادگیری تقویتی چندعاملی

تعریف یادگیری تقویتی چندعاملی

اهمیت یادگیری تقویتی چندعاملی

۲-۵-۲ بازی‌های چندعاملی در یادگیری تقویتی

بازی‌های جمع صفر

تعادل نش و کاربردهای آن

۲-۵-۳ چالش‌های یادگیری تقویتی چندعاملی

مدیریت تعارضات

تضمین پایداری

مشکلات همگرایی

۲-۵-۴ ایمنی در یادگیری تقویتی چندعاملی

تکنیک‌های تضمین ایمنی

کاربرد ایمنی در تعاملات حساس

۲-۵-۵ الگوریتم‌های یادگیری تقویتی چندعاملی

الگوریتم‌های همکاری مانند MADDPG

یادگیری توزیع‌شده با تضمین ایمنی

۲-۵-۶ کاربرد MARL در مسائل نظری

حل مسائل بازی‌های جمع صفر

مدل‌سازی مسائل رقابتی و همکاری

۲-۷ کاربرد چند عاملی در بازی‌ها

در این بخش به بررسی کاربردهای یادگیری تقویتی چند عاملی در حوزه بازی‌ها پرداخته و نشان می‌دهیم که الگوریتم پیشنهادی ما نه تنها بهینه بلکه ایمن و قابل اعتماد است.

۲-۷-۱ تعریف بازی‌های چند عاملی

مفاهیم پایه در بازی‌های چند عاملی

بازی‌های چند عاملی شامل محیط‌هایی هستند که در آن چندین عامل به صورت همزمان و مستقل به تعامل می‌پردازند. این تعاملات می‌تواند شامل همکاری، رقابت یا ترکیبی از هر دو باشد. در چنین محیط‌هایی، هر عامل با هدف خود به حداکثر رساندن پاداش یا دستیابی به اهداف مشخص، استراتژی‌های خود را توسعه می‌دهد.

نمونه‌های بازی‌های چند عاملی

از جمله نمونه‌های معروف بازی‌های چند عاملی می‌توان به بازی‌های استراتژیک مانند StarCraft، بازی‌های ورزشی چند نفره و بازی‌های تخته‌ای مانند شطرنج چند عاملی اشاره کرد. این بازی‌ها به دلیل پیچیدگی‌های بالای تعاملات میان عوامل، محیط‌های مناسبی برای آزمایش و ارزیابی الگوریتم‌های یادگیری تقویتی چند عاملی فراهم می‌کنند.

۲-۷-۲ الگوریتم بهینه در بازی‌های چند عاملی

معرفی الگوریتم پیشنهادی

الگوریتم پیشنهادی ما بر اساس ترکیبی از یادگیری عمیق و تکنیک‌های بهینه‌سازی چند عاملی طراحی شده است. این الگوریتم با استفاده از شبکه‌های عصبی عمیق، استراتژی‌های بهینه برای هر عامل را در محیط‌های پیچیده بازی‌های چند عاملی یاد می‌گیرد. علاوه بر این، با استفاده از مکانیزم‌های تعاملی، هماهنگی و همکاری میان عوامل بهبود می‌یابد.

بهینه بودن الگوریتم

الگوریتم ما با هدف کاهش زمان همگرایی و افزایش کارایی در محیط‌های دینامیک بهینه شده است. با استفاده از تکنیک‌های پیشرفته مانند یادگیری انتقالی و تنظیم خودکار نرخ یادگیری، الگوریتم قادر است به سرعت به تعادل‌های مطلوب برسد و عملکرد بهینه‌ای در بازی‌ها ارائه دهد.

۲-۷-۳ ایمنی و قابلیت اطمینان الگوریتم در بازی‌ها

تضمین ایمنی در تعاملات میان عوامل

ایمنی در تعاملات چند عاملی به معنای جلوگیری از رفتارهای غیرمنتظره و تضمین هماهنگی میان عوامل است. در الگوریتم ما، از مکانیزم‌های نظارتی و محدودکننده استفاده شده است که اطمینان حاصل می‌کنند تعاملات میان عوامل منجر به نتیجه‌ای نامطلوب نمی‌شود. این شامل محدود کردن فضای عملیاتی و اعمال قیود بر سیاست‌های یادگیری هر عامل می‌باشد.

قابلیت اطمینان و مقاومت در برابر خطاها

الگوریتم پیشنهادی با هدف افزایش قابلیت اطمینان و مقاومت در برابر خطاها طراحی شده است. با استفاده از تکنیک‌های افزونگی و یادگیری توزیع‌شده، الگوریتم قادر است در صورت بروز خطا یا نقص در برخی از عوامل، به عملکرد مطلوب خود ادامه دهد. این ویژگی‌ها الگوریتم را برای استفاده در محیط‌های حساس و پویا مناسب می‌سازد.

۲-۷-۴ مطالعات موردی و نتایج تجربی

پیاده‌سازی در بازی‌های مشخص

الگوریتم ما در بازی‌های مختلفی مانند بازی استراتژیک StarCraft و بازی‌های ورزشی چند نفره پیاده‌سازی شده است. در هر یک از این بازی‌ها، عملکرد الگوریتم ما با توجه به معیارهای بهینه بودن و ایمنی ارزیابی شده است.

تحلیل عملکرد و مقایسه با الگوریتم‌های دیگر

نتایج تجربی نشان می‌دهد که الگوریتم پیشنهادی ما نسبت به الگوریتم‌های موجود در زمینه یادگیری تقویتی چند عاملی از نظر سرعت همگرایی و کارایی بهتری دارد. علاوه بر این، با استفاده از مکانیزم‌های تضمین ایمنی، رفتارهای ناخواسته و خطرناک در تعاملات میان عوامل به طور قابل توجهی کاهش یافته است.

۲-۷-۵ نتیجه‌گیری

در این بخش، به بررسی و تحلیل نتایج به دست آمده از پیاده‌سازی الگوریتم پیشنهادی در بازی‌های چند عاملی پرداخته شد. مشاهده شد که الگوریتم ما توانسته است با ارائه راه‌حل‌های بهینه و ایمن، عملکرد بهتری نسبت به الگوریتم‌های موجود داشته باشد. این امر نشان‌دهنده قابلیت‌های بالای الگوریتم در مدیریت تعاملات پیچیده و تضمین ایمنی در محیط‌های چند عاملی است.

۲-۷-۶ پیشنهادات برای تحقیقات آینده

با توجه به نتایج حاصل شده، پیشنهاد می‌شود که تحقیقات آینده بر روی بهبود بیشتر ایمنی و قابلیت اطمینان الگوریتم‌های چند عاملی متمرکز شود. همچنین، توسعه الگوریتم‌های سازگار با محیط‌های پویا و پیچیده‌تر و ارزیابی آن‌ها در بازی‌ها و محیط‌های واقعی‌تر از جمله مسیرهای پیشنهادی برای ادامه تحقیقات می‌باشد.

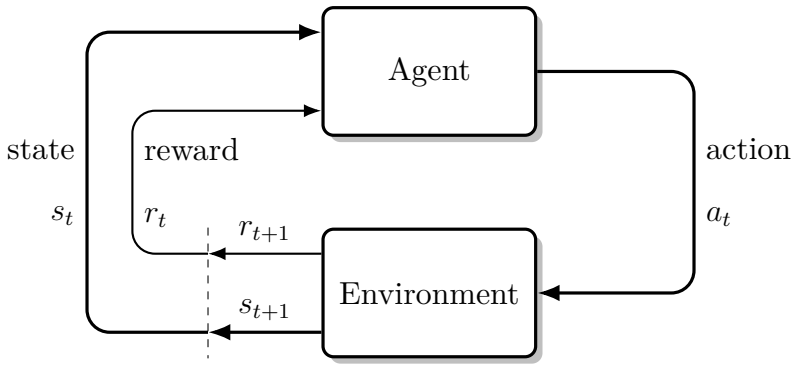
فصل ۳

یادگیری تقویتی

۳-۱ مفاهیم اولیه

دو بخش اصلی یادگیری تقویتی[۱] شامل عامل[۲] و محیط[۳] است. عامل در محیط قرار دارد و با آن تعامل دارد. 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 بر 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 می کند، سیگنالی که به آن می‌گوید وضعیت تعامل فعلی عامل محیط چقدر خوب یا بد است. هدف عامل به حداکثر رساندن پاداش انباشته خود است که بازگشت[۵] نام دارد. یادگیری تقویتی به روش‌هایی گفته می‌شود که در آن‌ها عامل رفتارهای مناسب برای رسیدن به هدف خود را می‌آموزد. در شکل ۳-۱ تعامل بین محیط و عامل نشان داده شده‌است.

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



شکل ۳-۱: حلقه تعامل عامل و محیط

۳-۱-۱ 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.

۳-۱-۲ 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.

۳-۱-۳ سیاست

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

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

زیر نشان داده می‌شود:

$$a_t = \mu(s_t) \tag{۱-۳}$$

$$a_t \sim \pi(\cdot|s_t) \tag{۲-۳}$$

که زیروند t بیانگر زمان است. در یادگیری تقویتی عمیق از سیاست‌های پارامتری‌شده استفاده می‌شود. خروجی این سیاست‌ها تابعی از مجموعه‌ای از پارامترها (برای مثال وزن‌ها و بایاس‌های یک شبکه عصبی) هستند که می‌توان از الگوریتم‌های بهینه‌سازی جهت تعیین پارامترها استفاده کرد. در این پژوهش پارامترهای سیاست با θ نشان داده شده‌است و سپس نماد آن به عنوان زیروند سیاست مانند معادله (۳-۳) نشان داده شده‌است.

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

۳-۱-۴ مسیر

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

$$\tau = (s_0, a_0, s_1, a_1, \cdots) \tag{۴-۳}$$

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 انجام شده توسط عامل (a_t) بستگی دارند. گذار حالت را می‌توان به‌صورت زیر تعریف کرد.

$$s_{t+1} = f(s_t, a_t) \tag{۵-۳}$$

۳-۱-۵ تابع پاداش و بازگشت

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

$$r_t = R(s_t, a_t, s_{t+1}) \tag{۶-۳}$$

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

در این پژوهش پاداش تنها تابعی از جفت حالت–عمل ($r_t = R(s_t, a_t)$) است. 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 بازگشت[۱۶] گفته می‌شود. یکی از انواع بازگشت، بازگشت 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 (۳-۷) نشان داده شده‌است.

$$R(\tau) = \sum_{t=0}^{T} r_t \tag{۳-۷}$$

نوع دیگری از بازگشت، بازگشت 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 بازگشت (۳-۸) شامل یک فاکتور تنزیل[۲۰] با نماد γ است که عددی بین صفر و یک است.

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t \tag{۳-۸}$$

۳-۱-۶ ارزش در یادگیری تقویتی

در یادگیری تقویتی، دانستن ارزش[۲۱] یک حالت یا جفت حالت–عمل ضروری است. منظور از ارزش، بازگشت 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، به طور میانگین چه مقدار پاداش دریافت خواهد کرد. توابع ارزش تقریبا 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 بازگشت مورد انتظار است در صورتی که از حالت s شروع شود و همیشه طبق سیاست π عمل شود و به‌صورت زیر بیان می‌شود:

$$V^\pi(s) = \mathop{\mathbb{E}}_{\tau\sim\pi}[R(\tau)|s_0 = s] \tag{۳-۹}$$

۲. تابع ارزش–عمل تحت سیاست[۲۴] ($Q^\pi(s,a)$): خروجی این تابع بازگشت 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:

$$Q^{\pi}(s,a) = \underset{\tau\sim\pi}{\mathbb{E}}[R(\tau)|s_0 = s, a_0 = a] \qquad (۱۰-۳)$$

۳. تابع ارزش بهینه[۲۵] $(V^*(s))$: خروجی این تابع بازگشت 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:

$$V^*(s) = \max_{\pi}(V^{\pi}(s)) \qquad (۱۱-۳)$$

۴. تابع ارزش–عمل بهینه[۲۶] $(Q^*(s,a))$: خروجی این تابع بازگشت 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:

$$Q^*(s,a) = \max_{\pi}(Q^{\pi}(s,a)) \qquad (۱۲-۳)$$

۳-۱-۷ معادلات بلمن

توابع ارزش اشاره شده از معادلات خاصی که به آن‌ها معادلات بلمن گفته می‌شود، پیروی می‌کنند. ایده اصلی پشت معادلات بلمن اسن 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:

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

$$Q^{\pi}(s,a) = \underset{s'\sim P}{\mathrm{E}}\Big[r(s,a) + \gamma \underset{a'\sim\pi}{\mathrm{E}}[Q^{\pi}(s',a')]\Big]$$

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 تخفیف 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

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

سیاست 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:

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

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.

۳-۱-۸ 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 به طور 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 کمتر بودن عملکرد آن نسبت به میانگین است.

[27] Advantage Function

۳-۲ 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 (۳-۱۳) به دست آورد.

$$a^*(s) = \arg\max_a Q^*(s,a) \tag{۳-۱۳}$$

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.

۳-۲-۱ یادگیری Q در DDPG

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

$$Q^*(s,a) = \mathop{\mathrm{E}}_{s'\sim P}\left[r(s,a) + \gamma \max_{a'} Q^*(s',a')\right] \tag{۳-۱۴}$$

عبارت $s' \sim P$ به این معنی است که وضعیت بعدی یعنی s' از توزیع احتمال $P(\cdot|s,a)$ نمونه‌گرفته 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

$$L(\phi, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}} \left[\left(Q_\phi(s,a) - \left(r + \gamma(1-d) \max_{a'} Q_\phi(s',a') \right) \right)^2 \right] \qquad (۱۵-۳)$$

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.

$$r + \gamma(1-d) \max_{a'} Q_\phi(s',a') \qquad (۱۶-۳)$$

در هنگام کمینه کردن تابع خطای میانگین مربعات بلمن، سعی شده‌است تا تابع Q شبیه‌تر به هدف یعنی رابطه (۱۶-۳) 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 با تاخیر پارامترهای شبکه اول را 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.

$$\phi_{\text{targ}} \leftarrow \rho\phi_{\text{targ}} + (1-\rho)\phi \qquad (۱۷-۳)$$

در رابطه بالا ρ یک ابرپارامتر[۳۴] است که بین صفر و یک انتخاب می‌شود. در این پژوهش این مقدار نزدیک به یک درنظرگرفته 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 (۳-۱۸) با استفاده از کاهش گرادیان تصادفی[۳۵] انجام می‌شود.

$$L(\phi, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\text{E}} \left[\left(Q_\phi(s,a) - \left(r + \gamma(1-d) Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))\right) \right)^2 \right] \qquad (۳-۱۸)$$

۳-۲-۲ 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.

$$\max_\theta \underset{s\sim\mathcal{D}}{\text{E}} \left[Q_\phi(s, \mu_\theta(s)) \right] \qquad (۳-۱۹)$$

۳-۲-۳ اکتشاف و بهره‌برداری در DDPG

برای بهبود اکتشاف[۳۷] در سیاست‌های DDPG، در زمان آموزش نویز به عمل‌ها اضافه می‌شود. نویسندگان مقاله DDPG [۲۷] 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.

۳-۲-۴ شبه‌کد DDPG

در این بخش، شبه‌کد الگوریتم DDPG پیاده‌سازی‌شده آورده شده‌است. در این پژوهش الگوریتم ۱ در محیط پایتون با استفاده از کتابخانه TensorFlow [۲۸] 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:

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

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

۳-۳ 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 شکست عامل DDPG در یادگیری این است که تابع Q یادگرفته شده شروع به بیش برآورد مقادیر Q می‌کند که منجر به واگرایی سیاست می‌شود. واگرایی به این دلیل رخ می‌دهد که در فرایند 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 اشاره شده استفاده می‌کند.

- یادگیری دوگانه‌ی محدود شده[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 اشاره شده در رابطه (۳-۲۰) آورده شده‌است.

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{i,\text{targ}}}(s', a'(s')) \tag{۳-۲۰}$$

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

$$L(\phi_1, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\text{E}} \left(Q_{\phi_1}(s, a) - y(r, s', d) \right)^2 \tag{۳-۲۱}$$

$$L(\phi_2, \mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\text{E}} \left(Q_{\phi_2}(s, a) - y(r, s', d) \right)^2 \tag{۳-۲۲}$$

- به‌روزرسانی‌های تاخیری سیاست[۴۴]: الگوریتم TD3 سیاست را با تاخیر بیشتری نسبت به تابع Q به‌روزرسانی می‌کند. در مرجع [۲۹] 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 به حداکثر رساندن Q_{ϕ_1} آموخته می‌شود:

$$\max_{\theta} \underset{s\sim\mathcal{D}}{\mathrm{E}} \left[Q_{\phi_1}(s, \mu_\theta(s))\right] \tag{۲۳-۳}$$

۳-۳-۱ اکتشاف و بهره‌برداری در TD3

الگوریتم TD3 یک سیاست قطعی را به‌صورت غیر سیاست محور آموزش را 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 اعمال شده‌است. شدت نویز جهت بهره‌برداری بهتر در طول زمان کاهش می‌یابد.

۳-۳-۲ شبه‌کد TD3

در این بخش الگوریتم TD3 پیاده‌سازی شده آورده شده‌است. در این پژوهش الگوریتم ۴ در محیط پایتون با استفاده از کتابخانه PyTorch [۳۰] 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:

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

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

۳-۴ عامل عملگر نقاد نرم

عملگرد نقاد نرم[۴۵] الگوریتمی است که یک سیاست تصادفی را به‌صورت سیاست محور بهینه می‌کند و پلی بین بهینه‌سازی سیاست تصادفی و رویکردهای مانند DDPG ایجاد می‌کند. این الگوریتم جانشین مستقیم TD3 نیست (زیرا تقریباً همزمان منتشر شده است)، اما ترفند یادگیری دوگانه محدود شده را در خود جای داده است و به دلیل سیاست تصادفی SAC، از چیزی روشی به نام صاف کردن سیاست هدف[۴۶] استفاده شده‌است. یکی از ویژگی های اصلی SAC، تنظیم آنتروپی است. آنتروپی معیاری از تصادفی بودن انتخاب عمل در سیاست است. سیاست به گونه ای آموزش داده می‌شود که حداکثر سازی تعادل بین بازده مورد انتظار و آنتروپی را بهینه کند. 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 بررسی یادگیری تقویتی تنظیم‌شده با آنتروپی[۴۷] را پرداخته شود. در RL تنظیم‌شده با آنتروپی، روابط تابع ارزش کمی متفاوت است.

۱-۴-۳ یادگیری تقویتی تنظیم‌شده با آنتروپی

آنتروپی کمیتی است که به طور کلی می‌گوید که یک متغیر تصادفی چقدر تصادفی است. اگر وزن یک سکه به گونه ای باشد که تقریباً همیشه نتیجه یک سمت آن باشد، آنتروپی پایینی دارد. اگر به طور مساوی وزن داشته باشد و شانس هر طرف سکه نصف باشد، آنتروپی بالایی دارد. فرض کنید x یک متغیر تصادفی با تابع چگالی احتمال P باشد. آنتروپی H متغیر x از توزیع آن P مطابق با رابطه زیر محاسبه می‌شود:

$$H(P) = \underset{x\sim P}{\mathrm{E}}\left[-\log P(x)\right]$$

۲-۴-۳ 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.

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

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

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

۳-۴-۳ 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.

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

۴-۴-۳ 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.

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

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

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

۵-۴-۳ معادله بلمن در SAC

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

$$Q^\pi(s,a) = \mathop{\mathrm{E}}_{\substack{s'\sim P\\ a'\sim\pi}}\left[R(s,a,s') + \gamma\left(Q^\pi(s',a') + \alpha H\left(\pi(\cdot|s')\right)\right)\right] \qquad (۲۴-۳)$$

$$= \mathop{\mathrm{E}}_{s'\sim P}\left[R(s,a,s') + \gamma V^\pi(s')\right] \qquad (۲۵-۳)$$

۶-۴-۳ یادگیری Q

با درنظرگرفتن موارد اشاره‌شده، یادگیری Q در SAC با کمینه‌کردن تابع خطای میانگین مربعات بلمن (MSBE) یعنی معادله (۶-۴-۳) با استفاده از کاهش گرادیان انجام می‌شود.

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

[48] Trade-Off

در معادله (۳-۴-۶) تابع هدف برای روش یادگیری تقویتی SAC به صورت زیر تعریف می‌شود.

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

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

۳-۴-۷ 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.

$$V^\pi(s) = \mathop{\mathrm{E}}_{a\sim\pi}\left[Q^\pi(s,a)\right] + \alpha H\left(\pi(\cdot|s)\right) \tag{۲۶-۳}$$

$$= \mathop{\mathrm{E}}_{a\sim\pi}\left[Q^\pi(s,a) - \alpha \log \pi(a|s)\right] \tag{۲۷-۳}$$

روش بهینه‌سازی سیاست از ترفند پارامترسازی مجدد[۴۹] استفاده می‌کند، 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 [۳۱]، از یک سیاست گاوسی[۵۰] فشرده استفاده شده است. بر اساس این روش نمونه‌ها مطابق با رابطه زیر بدست می‌آیند:

$$\tilde{a}_\theta(s, \xi) = \tanh\left(\mu_\theta(s) + \sigma_\theta(s) \odot \xi\right), \qquad \xi \sim \mathcal{N}(0, I)$$

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.

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.

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

[49] Reparameterization
[50] Squashed Gaussian Policy
[51] Likelihood Ratio Trick

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 بهینه‌سازی می‌شود:

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

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

۳-۴-۸ 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.

۳-۴-۹ شبه‌کد SAC

در این بخش الگوریتم SAC پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم ۳ در محیط پایتون با استفاده از کتابخانه PyTorch [۳۰] 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:

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

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:

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

۳-۵ 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 شده است 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 سیایت 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 شده است 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 شده است.

- 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 شده است. 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 شده است.

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

۳-۵-۱ سیاست در الگوریتم PPO

تابع سیاست در الگوریتم PPO به صورت یک شبکه عصبی پیچیده پیاده‌سازی شده است. 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 tanh استفاده شده است. تابع سیاست در الگوریتم PPO به صورت زیر به‌روزرسانی می‌شود:

$$\theta_{k+1} = \arg\max_{\theta} \underset{s,a\sim\pi_{\theta_k}}{\mathrm{E}} [L(s, a, \theta_k, \theta)] \tag{۳-۲۸}$$

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

$$L(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a),\ \ \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_k}}(s, a)\right) \tag{۳-۲۹}$$

که در آن ϵ یک فراپامتر است که مقدار آن معمولا کوچک است. این فراپامتر مشخص می‌کند که چقدر اندازه گام بهینه‌سازی باید محدود شود. در این پژوهش مقدار $\epsilon = 0.2$ انتخاب شده است.

در حالی که این نوع محدود کردن (PPO-Clip) تا حد زیادی به اطمینان از به‌روزرسانی‌های معقول سیاست کمک می‌کند، همچنان ممکن است با سیاست 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 شده است. اگر میانگین واگرایی کولباک–لیبلر (KL) خط‌مشی جدید از خط‌مشی قدیمی از یک آستانه فراتر رود، گام‌های گرادیان (بهینه‌سازی) را متوقف می‌شوند.

۳-۵-۲ اکتشاف و بهره‌برداری در PPO

الگوریتم PPO از یک سیاست تصادفی به‌صورت سیاست محور 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

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

می‌کند تا از پاداش‌هایی که قبلاً پیدا کرده است، بهره‌برداری کند. البته این موضوع می‌تواند منجر به گیر افتادن خط‌مشی در بهینه‌های محلی[۵۹] شود.

۳-۵-۳ شبه‌کد PPO

در این بخش الگوریتم PPO پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم ۴ در محیط پایتون با استفاده از کتابخانه PyTorch [۳۰] پیاده‌سازی شده است.

الگوریتم ۴ بهینه‌سازی سیاست مجاور (PPO-Clip)

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

۱: به ازای $k = 0, 1, 2, \ldots$:

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.

---

[59] Local Optima

فصل ۴

یادگیری تقویتی چند عاملی

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[2] معرفی شده تا رفتارهای توزیعی چندعاملی را تنظیم کنند. روابط همکاری و رقابت میان عامل‌ها را در سناریوهای مختلف تحلیل شده و آن‌ها با الگوریتم‌های معمول یادگیری تقویتی چندعاملی ترکیب شده است. 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 شد.

۴-۱ تعریف یادگیری تقویتی چندعاملی

یادگیری تقویتی چندعاملی (Multi-Agent Reinforcement Learning - MARL) شاخه‌ای از یادگیری ماشین است که به مطالعه و توسعه الگوریتم‌هایی می‌پردازد که در آن‌ها چندین عامل به طور همزمان در یک محیط تعامل می‌کنند. هر عامل با هدف به حداکثر رساندن پاداش خود، سیاست‌های خود را بهبود می‌بخشد. در MARL، تعاملات میان عامل‌ها می‌تواند شامل همکاری، رقابت یا هر دو باشد که این امر چالش‌ها و فرصت‌های جدیدی را به همراه دارد.

[1] Multi-Agent
[2] Equilibrium

۴-۱-۱ مفاهیم پایه در یادگیری تقویتی چندعاملی

یادگیری تقویتی چندعاملی ترکیبی از یادگیری تقویتی و سیستم‌های چندعاملی است. در اینجا به برخی از مفاهیم پایه پرداخته می‌شود:

عامل‌ها و محیط

در MARL، چندین عامل (Agents) در یک محیط مشترک فعالیت می‌کنند. هر عامل در هر لحظه زمانی، وضعیت محیط (State) را مشاهده کرده و بر اساس آن اقدام (Action)ی انجام می‌دهد. محیط سپس بازخوردی به صورت پاداش (Reward) و وضعیت جدید ارائه می‌دهد.

سیاست‌ها و اهداف

هر عامل دارای یک سیاست (Policy) است که نقشه‌ای از وضعیت‌ها به اقدامات را فراهم می‌کند. هدف هر عامل در MARL به حداکثر رساندن مجموع پاداش‌های دریافتی در طول زمان است. این اهداف می‌توانند همزمان شامل اهداف فردی و گروهی باشند.

۴-۱-۲ تعاملات میان عامل‌ها

در MARL، تعاملات میان عامل‌ها نقش مهمی در یادگیری و عملکرد سیستم دارند. این تعاملات می‌توانند به صورت‌های مختلفی ظاهر شوند:

همکاری

در تعاملات همکاری، عامل‌ها به منظور دستیابی به اهداف مشترک با یکدیگر همکاری می‌کنند. این نوع تعاملات معمولاً در مسائلی که نیاز به هماهنگی و اشتراک‌گذاری اطلاعات دارند، مشاهده می‌شود. به عنوان مثال، در یک تیم رباتیک که باید با هم کار کنند تا یک شیء را جابجا کنند.

رقابت

در تعاملات رقابتی، عامل‌ها با یکدیگر به رقابت می‌پردازند تا منابع محدود یا اهداف مشخص را به دست آورند. این نوع تعاملات معمولاً در بازی‌های استراتژیک یا سیستم‌های بازار مشاهده می‌شود.

ترکیبی از همکاری و رقابت

در بسیاری از موارد، تعاملات میان عامل‌ها ترکیبی از همکاری و رقابت هستند. به عنوان مثال، در یک بازی تیمی، اعضای هر تیم با یکدیگر همکاری می‌کنند در حالی که با تیم‌های دیگر رقابت می‌نمایند.

۴-۱-۳ تفاوت‌های MARL با یادگیری تقویتی تک عاملی

یادگیری تقویتی چندعاملی تفاوت‌های مهمی با یادگیری تقویتی تک عاملی دارد که در زیر به برخی از آن‌ها اشاره می‌شود:

دینامیک محیط

در MARL، محیط به طور همزمان توسط چندین عامل تحت تأثیر قرار می‌گیرد، که این امر دینامیک محیط را پیچیده‌تر می‌کند. در مقابل، در یادگیری تقویتی تک عاملی، تنها یک عامل وجود دارد که محیط را تحت تأثیر قرار می‌دهد.

تداخل و ناپایداری

وجود چندین عامل می‌تواند منجر به تداخل و ناپایداری در یادگیری شود، زیرا هر عامل ممکن است سیاست‌های خود را تغییر دهد که تأثیر مستقیم بر سایر عامل‌ها دارد. این امر به ایجاد یک محیط غیر ثابت و چالش‌برانگیز برای یادگیری منجر می‌شود.

هماهنگی و ارتباطات

در MARL، عامل‌ها ممکن است نیاز به هماهنگی و ارتباط با یکدیگر داشته باشند تا به اهداف مشترک دست یابند. این امر نیازمند مکانیزم‌های ارتباطی و هماهنگی موثر بین عامل‌ها است که در یادگیری تقویتی تک عاملی وجود ندارد.

۴-۱-۴ چالش‌های یادگیری تقویتی چندعاملی

یادگیری تقویتی چندعاملی با چالش‌های خاصی مواجه است که باید برای رسیدن به عملکرد مطلوب، آن‌ها را مدیریت کرد:

مسئله همگرایی

همگرایی الگوریتم‌های MARL پیچیده‌تر از الگوریتم‌های تک عاملی است زیرا تعاملات میان عامل‌ها می‌تواند منجر به رفتارهای غیرپیش‌بینی‌پذیر و ناپایدار شود. تضمین همگرایی به بهینه یا حداقل مطلوب بودن سیاست‌ها یکی از چالش‌های اصلی است.

مدیریت تعارضات

در محیط‌های رقابتی، تعارضات میان عامل‌ها می‌تواند به کاهش کارایی سیستم منجر شود. طراحی مکانیزم‌هایی برای مدیریت تعارضات و ایجاد تعادل میان اهداف فردی و گروهی ضروری است.

پایداری و مقیاس‌پذیری

با افزایش تعداد عامل‌ها، حفظ پایداری و مقیاس‌پذیری سیستم یک چالش مهم است. الگوریتم‌های MARL باید بتوانند به طور موثری با افزایش تعداد عامل‌ها سازگار شوند بدون اینکه عملکرد سیستم کاهش یابد.

اطلاعات ناقص و عدم قطعیت

در بسیاری از سناریوهای MARL، عامل‌ها ممکن است با اطلاعات ناقص یا عدم قطعیت در مورد سیاست‌های دیگر عامل‌ها مواجه شوند. طراحی الگوریتم‌هایی که بتوانند در شرایط عدم قطعیت به خوبی عمل کنند، بسیار مهم است.

۴-۱-۵ کاربردهای یادگیری تقویتی چندعاملی

یادگیری تقویتی چندعاملی در حوزه‌های متعددی کاربرد دارد که در زیر به برخی از آن‌ها اشاره می‌شود:

سیستم‌های رباتیک

در سیستم‌های رباتیک چندعاملی، ربات‌ها برای انجام وظایف پیچیده به صورت هماهنگ با یکدیگر کار می‌کنند. این وظایف می‌تواند شامل جست‌وجو و نجات، حمل و نقل مواد، یا عملیات هماهنگ در محیط‌های غیرقابل پیش‌بینی باشد.

مدیریت منابع در شبکه‌های ارتباطی

در شبکه‌های ارتباطی، تخصیص بهینه منابع مانند پهنای باند و انرژی به عامل‌های مختلف (مانند دستگاه‌های کاربر) می‌تواند با استفاده از MARL بهبود یابد. این الگوریتم‌ها می‌توانند به طور پویا و خودکار تخصیص منابع را بهینه کنند.

بازی‌ها و شبیه‌سازی‌های اقتصادی

در بازی‌های چندعاملی و شبیه‌سازی‌های اقتصادی، MARL می‌تواند به مدل‌سازی و تحلیل رفتارهای بازار و تصمیم‌گیری‌های اقتصادی کمک کند. این کاربردها به پیش‌بینی دقیق‌تر روندهای اقتصادی و بهبود سیاست‌گذاری‌های مالی منجر می‌شوند.

۴-۱-۶ مبنای نظریه بازی در MARL

یکی از مبانی نظری MARL، نظریه بازی‌ها است که به تحلیل تعاملات میان عامل‌ها در محیط‌های رقابتی و همکاری می‌پردازد. استفاده از مفاهیم تعادل نش (Nash Equilibrium) و دیگر تعادل‌های بازی‌ها، به طراحی الگوریتم‌های MARL کمک می‌کند تا رفتارهای پایدار و بهینه‌ای را در تعاملات میان عامل‌ها تضمین نمایند.

۴-۲ اهمیت یادگیری تقویتی چندعاملی

یادگیری تقویتی چندعاملی به دلیل قابلیت‌های بالقوه‌اش در مدل‌سازی و حل مسائل پیچیده و پویا، اهمیت زیادی در حوزه‌های مختلف علمی و صنعتی دارد. در این بخش، به بررسی اهمیت MARL در زمینه‌های مختلف پرداخته و نقش آن را در توسعه سیستم‌های هوشمند متعدد بررسی می‌کنیم.

مدل‌سازی سیستم‌های پیچیده و پویا

یکی از دلایل اصلی اهمیت MARL، توانایی آن در مدل‌سازی سیستم‌های پیچیده و پویا است. در بسیاری از کاربردهای واقعی، سیستم‌ها شامل چندین عامل هستند که به صورت همزمان و مستقل به تعامل می‌پردازند. به عنوان مثال، در شبکه‌های ترافیکی، هر خودرو می‌تواند به عنوان یک عامل مستقل عمل کند که نیاز به

هماهنگی و تعامل با سایر خودروها برای بهینه‌سازی جریان ترافیک دارد. MARL با فراهم کردن چارچوبی برای تعامل و یادگیری میان این عوامل، امکان بهبود کارایی و کاهش ترافیک را فراهم می‌کند.

کاربرد در رباتیک چندعاملی

در حوزه رباتیک، سیستم‌های چندعاملی می‌توانند برای انجام وظایف پیچیده‌ای مانند جست‌وجو و نجات، حمل و نقل مواد، و عملیات هماهنگ در محیط‌های غیرقابل پیش‌بینی مورد استفاده قرار گیرند. به عنوان مثال، گروهی از ربات‌های پرنده (دُرون‌ها) می‌توانند با همکاری و تبادل اطلاعات، منطقه‌ای وسیع را برای شناسایی اهداف نظارت کنند یا به سرعت به تغییرات محیطی واکنش نشان دهند. MARL در این زمینه بهبود هماهنگی میان ربات‌ها و افزایش کارایی عملیات‌های چندعاملی را ممکن می‌سازد.

مدیریت منابع در شبکه‌های ارتباطی

شبکه‌های ارتباطی مدرن نیازمند مدیریت بهینه منابع مانند پهنای باند، انرژی و ظرفیت ذخیره‌سازی هستند. در این راستا، MARL می‌تواند به عنوان یک ابزار قدرتمند برای تخصیص بهینه منابع به عوامل مختلف شبکه عمل کند. به عنوان مثال، در شبکه‌های بی‌سیم، هر دستگاه کاربر می‌تواند به عنوان یک عامل مستقل عمل کرده و با یادگیری و تعامل با سایر دستگاه‌ها، نحوه بهینه‌سازی مصرف انرژی و پهنای باند را پیدا کند. این امر منجر به افزایش کارایی شبکه و کاهش هزینه‌های عملیاتی می‌شود.

توسعه الگوریتم‌های پیشرفته‌تر و قابل اعتمادتر

یکی دیگر از جنبه‌های مهم MARL، فهم و تحلیل تعاملات میان عوامل مختلف است که می‌تواند به توسعه الگوریتم‌های پیشرفته‌تر و قابل اعتمادتر منجر شود. با مطالعه رفتارها و استراتژی‌های مختلف در محیط‌های چندعاملی، پژوهشگران قادر به طراحی الگوریتم‌هایی می‌شوند که نه تنها بهینه عمل می‌کنند بلکه مقاومت بالایی در برابر تغییرات محیطی و رفتارهای غیرمنتظره دارند. این الگوریتم‌ها می‌توانند در شرایط متنوع و پیچیده‌تر به خوبی عمل کنند و از خطاها و ناهنجاری‌های احتمالی جلوگیری نمایند.

کاربرد در بازی‌های چندعاملی و شبیه‌سازی‌های اقتصادی

بازی‌های چندعاملی و شبیه‌سازی‌های اقتصادی از دیگر حوزه‌هایی هستند که به شدت از MARL بهره‌مند می‌شوند. در بازی‌های استراتژیک چند نفره، MARL می‌تواند به بازیگران کمک کند تا استراتژی‌های بهینه‌ای

برای رقابت و همکاری با یکدیگر توسعه دهند. همچنین، در شبیه‌سازی‌های اقتصادی، MARL می‌تواند به مدل‌سازی و تحلیل رفتارهای بازار و تصمیم‌گیری‌های اقتصادی کمک کند، که این امر به پیش‌بینی دقیق‌تر روندهای اقتصادی و بهبود سیاست‌گذاری‌های مالی منجر می‌شود.

افزایش قابلیت انعطاف‌پذیری و مقیاس‌پذیری سیستم‌ها

سیستم‌های چندعاملی معمولاً نیازمند قابلیت انعطاف‌پذیری و مقیاس‌پذیری بالا هستند تا بتوانند با تغییرات محیطی و افزایش تعداد عوامل سازگار شوند. MARL با استفاده از الگوریتم‌های توزیع‌شده و یادگیری محلی، امکان توسعه سیستم‌هایی با مقیاس بزرگ و پیچیدگی بالا را فراهم می‌کند. این امر به ویژه در کاربردهایی مانند اینترنت اشیاء[۳]، هوش مصنوعی توزیع‌شده و سیستم‌های بزرگ‌مقیاس داده‌های بزرگ[۴] بسیار حائز اهمیت است.

۱-۲-۴ بازی‌های جمع صفر

بازی‌های جمع صفر[۵] یکی از انواع اصلی بازی‌های چندعاملی هستند که در آن سود یک بازیکن به طور مستقیم با ضرر بازیکنان دیگر مرتبط است. در این بازی‌ها، مجموع پاداش‌ها برای همه بازیکنان در هر حالت برابر با صفر است، به این معنی که هر افزایشی در پاداش یکی از بازیکنان منجر به کاهش معادل آن در بازیکنان دیگر می‌شود. این نوع بازی‌ها به خوبی می‌توانند رقابت‌های شدید و استراتژی‌های بهینه را مدل‌سازی کنند.

بازی‌های جمع صفر می‌توانند بر اساس سناریوهای مختلف به دسته‌های متنوعی تقسیم‌بندی شوند. دو دسته اصلی این بازی‌ها عبارتند از بازی‌های ثابت و بازی‌های تکراری.

- **بازی ثابت (Static Game):** بازی ثابت ساده‌ترین شکل برای مدل‌سازی تعاملات میان عوامل است. در بازی ثابت، هر عامل تنها یک تصمیم‌گیری واحد را انجام می‌دهد. از آنجایی که هر عامل تنها یک بار عمل می‌کند، تقلب و خیانت غیرمنتظره می‌تواند در این نوع بازی‌ها سودآور باشد. بنابراین، هر عامل نیاز دارد تا به دقت استراتژی‌های سایر عوامل را پیش‌بینی کند تا بتواند به طور هوشمندانه عمل کرده و بیشترین سود ممکن را کسب کند. بازی‌های ثابت معمولاً در سناریوهای رقابتی با تعاملات کوتاه مدت کاربرد دارند.
- **بازی تکراری (Repeated Game):** بازی تکراری به وضعیتی اشاره دارد که در آن تمام عوامل می‌توانند بر اساس همان وضعیت برای چندین تکرار اقداماتی انجام دهند. سود کلی هر عامل مجموع

[3] Internet of Things (IoT)
[4] Big Data
[5] Zero-Sum

سودهای تخفیف‌شده برای هر تکرار از بازی است. به دلیل اقدامات مکرر تمام عوامل، تقلب و خیانت در طول تعاملات می‌تواند منجر به مجازات یا انتقام از سوی سایر عوامل در تکرارهای آینده شود. بنابراین، بازی تکراری از رفتارهای مخرب عوامل جلوگیری می‌کند و به طور کلی سود کل برای تمام عوامل را افزایش می‌دهد. بازی‌های تکراری معمولاً در سناریوهای همکاری بلندمدت و تعاملات پویا کاربرد دارند.

این دسته‌بندی‌ها به محققان و توسعه‌دهندگان کمک می‌کنند تا بازی‌های چندعاملی را بر اساس ویژگی‌های مختلف آن‌ها شناسایی و تحلیل کنند. در بازی‌های ثابت، تمرکز بر پیش‌بینی دقیق استراتژی‌های دیگر عوامل و اتخاذ بهترین تصمیم در یک لحظه زمانی است. در مقابل، بازی‌های تکراری نیازمند توسعه استراتژی‌های پایدار و قابل اعتماد هستند که نه تنها در تکرار اول بلکه در تکرارهای بعدی نیز موثر باشند.

بازی‌های جمع صفر در یادگیری تقویتی چندعاملی به دلیل سادگی و قابلیت مدل‌سازی دقیق تعاملات رقابتی، به عنوان یک ابزار قدرتمند برای تحلیل و توسعه الگوریتم‌های MARL مورد استفاده قرار می‌گیرند. این بازی‌ها امکان بررسی رفتارهای استراتژیک، بهینه‌سازی سیاست‌ها و تحلیل تعادل‌های نش (Nash Equilibrium) را فراهم می‌کنند که در نهایت به بهبود عملکرد سیستم‌های چندعاملی منجر می‌شود.

مثال‌ها و کاربردها یکی از مثال‌های معروف بازی‌های جمع صفر، بازی شطرنج است که در آن هر حرکت یک بازیکن مستقیماً به نفع یا ضرر بازیکن دیگر است. سایر مثال‌ها شامل بازی‌های استراتژیک مانند Poker و Go می‌باشند که در آن‌ها تعاملات رقابتی میان بازیکنان به طور کامل با اصول بازی‌های جمع صفر مطابقت دارند.

در حوزه‌های عملی، بازی‌های جمع صفر می‌توانند برای مدل‌سازی رقابت‌های بازار، مذاکرات اقتصادی و حتی تعاملات میان ربات‌های خودران در محیط‌های رقابتی مورد استفاده قرار گیرند. این کاربردها به محققان امکان می‌دهند تا الگوریتم‌هایی طراحی کنند که قادر به بهینه‌سازی عملکرد در شرایط رقابتی و متغیر باشند.

چالش‌ها و فرصت‌ها یکی از چالش‌های اصلی در بازی‌های جمع صفر، پیش‌بینی دقیق رفتارهای رقبا و اتخاذ تصمیم‌های بهینه در مواجهه با استراتژی‌های متغیر آن‌ها است. همچنین، در بازی‌های تکراری، ایجاد تعادل‌های پایدار و جلوگیری از رفتارهای مخرب به عنوان یک چالش مهم مطرح است. با این حال، این چالش‌ها فرصت‌های قابل توجهی برای توسعه الگوریتم‌های پیشرفته و افزایش قابلیت‌های یادگیری تقویتی چندعاملی فراهم می‌کنند که می‌توانند در شرایط پیچیده‌تر و پویا نیز عملکرد مطلوبی داشته باشند.

۴-۲-۲ تعادل نش

تعادل نش (Nash Equilibrium) یکی از مفاهیم بنیادی در نظریه بازی‌ها است که به تحلیل تعاملات میان عوامل در محیط‌های رقابتی و همکاری می‌پردازد. این مفهوم به ما امکان می‌دهد تا نقاط تعادلی را شناسایی کنیم که در آن هیچ کدام از عوامل نمی‌توانند با تغییر یکجانبه استراتژی خود سود بیشتری کسب کنند، مشروط بر این که سایر عوامل استراتژی‌های خود را ثابت نگه دارند. در این بخش، به دو نوع اصلی تعادل نش، یعنی تعادل نش خالص و تعادل نش ترکیبی، پرداخته می‌شود.

تعادل نش خالص (Pure Strategy Nash Equilibrium)

تعادل نش خالص به حالتی از بازی گفته می‌شود که در آن هر عامل یک استراتژی مشخص و غیر تصادفی را انتخاب کرده است و هیچ کدام از عوامل نمی‌توانند با تغییر استراتژی خود بدون تغییر استراتژی‌های سایر عوامل سود بیشتری کسب کنند. به عبارت دیگر، در یک تعادل نش خالص، هیچ عاملی انگیزه‌ای برای انحراف از استراتژی انتخابی خود ندارد.

تعریف ریاضی فرض کنید یک بازی با n بازیکن داریم که هر بازیکن i دارای مجموعه‌ای از استراتژی‌ها S_i است. تابع سود هر بازیکن i به صورت $u_i : S_1 \times S_2 \times \cdots \times S_n \to \mathbb{R}$ تعریف می‌شود. یک پروفایل استراتژی $(s_1^*, s_2^*, \ldots, s_n^*)$ تعادل نش خالص است اگر برای هر بازیکن i و برای هر استراتژی $s_i \in S_i$، شرط زیر برقرار باشد:

$$u_i(s_i^*, s_{-i}^*) \geqslant u_i(s_i, s_{-i}^*)$$

که در آن s_{-i}^* نشان‌دهنده مجموعه استراتژی‌های سایر بازیکنان است.

مثال یک مثال کلاسیک از تعادل نش خالص، بازی Prisoner's Dilemma است. در این بازی، هر دو بازیکن انتخاب می‌کنند که همکاری کنند یا خیانت کنند. اگر هر دو بازیکن همکاری کنند، هر کدام دو سال زندانی می‌شوند. اگر یکی خیانت کند و دیگری همکاری کند، خیانتکار بی‌نیازی و همکار یک سال زندانی می‌شود. اگر هر دو خیانت کنند، هر کدام سه سال زندانی می‌شوند. تعادل نش خالص در این بازی زمانی رخ می‌دهد که هر دو بازیکن انتخاب به خیانت کردن هستند، زیرا هیچ یک از بازیکنان نمی‌توانند با تغییر استراتژی خود سود بیشتری کسب کنند.

تعادل نش ترکیبی (Mixed Strategy Nash Equilibrium)

تعادل نش ترکیبی به حالتی از بازی گفته می‌شود که در آن هر عامل می‌تواند با احتمال معینی بین استراتژی‌های مختلف خود انتخاب کند. در این نوع تعادل نش، هیچ کدام از عوامل نمی‌توانند با تغییر احتمالات انتخاب استراتژی خود بدون تغییر احتمالات انتخاب استراتژی‌های سایر عوامل سود بیشتری کسب کنند.

تعریف ریاضی در بازی‌هایی که تعادل نش خالص وجود ندارد، ممکن است تعادل نش ترکیبی وجود داشته باشد. فرض کنید مجموعه استراتژی‌های بازیکن i شامل $S_i = \{s_i^1, s_i^2, \ldots, s_i^m\}$ است. یک تعادل نش ترکیبی شامل توزیع‌های احتمالاتی σ_i^* برای هر بازیکن i است که در آن هیچ بازیکن نمی‌تواند با تغییر توزیع احتمالات انتخاب استراتژی خود سود بیشتری کسب کند. به طور ریاضی، برای هر بازیکن i و هر استراتژی $s_i^j \in S_i$، شرط زیر برقرار است:

$$\sum_{s_i \in S_i} \sigma_i^*(s_i) u_i(s_i, \sigma_{-i}^*) \geqslant \sum_{s_i \in S_i} \sigma_i(s_i) u_i(s_i, \sigma_{-i}^*)$$

که در آن σ_{-i}^* توزیع‌های احتمالاتی سایر بازیکنان است.

مثال یک مثال از تعادل نش ترکیبی، بازی Rock-Paper-Scissors است. در این بازی، هر بازیکن می‌تواند سنگ، کاغذ یا قیچی انتخاب کند. اگر هر دو بازیکن انتخاب یک گزینه را داشته باشند، بازی مساوی است. اگر یک بازیکن سنگ انتخاب کند و دیگری قیچی، سنگ برقی قیچی را شکست می‌دهد، و به همین ترتیب. در این بازی، تعادل نش ترکیبی زمانی رخ می‌دهد که هر بازیکن با احتمال برابر ۱/۳ سنگ، کاغذ و قیچی را انتخاب کند. در این حالت، هیچ کدام از بازیکنان نمی‌توانند با تغییر احتمالات انتخاب استراتژی خود سود بیشتری کسب کنند.

پیش‌بینی و استراتژی‌ها در تعادل نش ترکیبی، بازیکنان به گونه‌ای استراتژی‌های خود را انتخاب می‌کنند که بازیکنان دیگر نیز مجبور به انتخاب استراتژی‌های ترکیبی خود می‌شوند تا تعادل حفظ شود. این نوع تعادل نش به ویژه در بازی‌هایی که تعادل نش خالص وجود ندارد یا تعادل نش خالص ناامن است، اهمیت دارد.

چالش‌ها و فرصت‌ها یکی از چالش‌های اصلی در تعادل نش ترکیبی، محاسبه و پیش‌بینی توزیع‌های احتمالاتی مناسب برای هر بازیکن است. این امر به ویژه در بازی‌های با تعداد زیاد استراتژی‌ها و بازیکنان پیچیده‌تر است. با این حال، تعادل نش ترکیبی فرصت‌های زیادی را برای تحلیل و بهبود استراتژی‌های بازیکنان فراهم می‌کند، به ویژه در بازی‌هایی که نیاز به تعادل میان استراتژی‌های مختلف دارند.

کاربردها تعادل نش ترکیبی در بسیاری از حوزه‌ها کاربرد دارد، از جمله اقتصاد، سیاست، و هوش مصنوعی. در اقتصاد، این مفهوم می‌تواند برای تحلیل رقابت‌های بازار و تصمیم‌گیری‌های استراتژیک شرکت‌ها مورد استفاده قرار گیرد. در هوش مصنوعی، تعادل نش ترکیبی می‌تواند به طراحی الگوریتم‌های هوشمند برای بازی‌ها و سیستم‌های چندعاملی کمک کند که قادر به تعامل بهینه و تعادلی با سایر عوامل هستند.

نتیجه‌گیری تعادل نش، چه خالص و چه ترکیبی، ابزار قدرتمندی در تحلیل تعاملات میان عوامل در محیط‌های رقابتی و همکاری فراهم می‌کند. فهم و کاربرد این تعادل‌ها به محققان و توسعه‌دهندگان امکان می‌دهد تا الگوریتم‌های هوشمندتری طراحی کنند که قادر به اتخاذ تصمیمات بهینه در مواجهه با استراتژی‌های متغیر و پیچیده دیگر عوامل هستند. با ادامه تحقیقات در این زمینه، انتظار می‌رود که تعادل نش نقش مهم‌تری در بهبود عملکرد سیستم‌های چندعاملی ایفا کند.

۴-۲-۳ ایمنی و مقاومت در یادگیری تقویتی چندعاملی

در استفاده از یادگیری تقویتی چندعاملی (Multi-Agent Reinforcement Learning - MARL)، مسائل مربوط به ایمنی و مقاومت در برابر اختلالات یکی از چالش‌های اساسی مطرح می‌گردد. به منظور اطمینان از عملکرد قابل اعتماد و ایمن الگوریتم‌های یادگیری تقویتی چندعاملی، نیازمند توسعه روش‌هایی هستیم که بتوانند در مواجهه با رفتارهای غیرمنتظره یا مخرب سایر عوامل، پایداری و ایمنی سیستم را حفظ نمایند. در این بخش، به بررسی مفاهیم ایمنی و مقاومت در MARL پرداخته شده و چگونگی افزایش مقاومت الگوریتم‌ها از طریق در نظر گرفتن عوامل به عنوان اختلالات مورد بحث قرار گرفته است.

ایمنی در MARL به معنای تضمین این است که تعاملات میان عوامل منجر به نتایج نامطلوب یا خطرناک نشوند. برای دستیابی به ایمنی، روش‌هایی نظیر محدود کردن فضای عملیاتی، اعمال قیود بر سیاست‌های یادگیری و استفاده از الگوریتم‌های مقاوم در برابر خطا به کار گرفته شده‌اند. یکی از رویکردهای موثر در افزایش مقاومت سیستم، فرض کردن یکی از عوامل به عنوان اختلال (Disturbance) در محیط است. با این فرض، الگوریتم‌ها قادر خواهند بود تا به گونه‌ای طراحی شوند که در حضور اختلالات احتمالی، عملکرد سیستم همچنان قابل اعتماد باقی بماند.

فرض کردن اختلال به عنوان عامل

در محیط‌های چندعاملی، برخی از عوامل ممکن است رفتارهای مخرب یا غیرمنتظره‌ای را از خود نشان دهند که می‌تواند به عملکرد کلی سیستم آسیب برساند. برای مقابله با این مسئله، فرض می‌شود که یک یا چند عامل

به عنوان اختلالات در نظر گرفته شوند. این اختلالات می‌توانند به صورت عمدی یا غیرعمدی ایجاد شوند و هدف آن‌ها کاهش کارایی سیستم است. با فرض کردن این اختلالات، الگوریتم‌های MARL قادر خواهند بود تا سیاست‌هایی را یاد بگیرند که در مواجهه با این اختلالات نیز عملکرد بهینه و ایمنی را حفظ کنند.

تعریف مقاومت و ایمنی در MARL

مقاومت در MARL به معنای توانایی الگوریتم در حفظ عملکرد مطلوب در حضور اختلالات و تغییرات محیطی است. این مقاومت می‌تواند از طریق طراحی سیاست‌های بهینه که به گونه‌ای تنظیم شده‌اند که تأثیر اختلالات را به حداقل برسانند، به دست آید. به علاوه، ایمنی می‌تواند از طریق تضمین عدم وقوع رفتارهای خطرناک و حفظ تعادل سیستم در مواجهه با رفتارهای مخرب حاصل شود.

تعریف ریاضی مقاومت فرض کنید یک محیط چندعاملی با مجموعه‌ای از عوامل $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ وجود دارد که در آن یک عامل A_d به عنوان اختلال تعریف شده است. هدف این است که الگوریتم MARL به گونه‌ای طراحی شود که سیاست‌های یادگرفته شده $\pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ بتوانند عملکرد بهینه را حتی در حضور A_d حفظ کنند. به طور ریاضی، مقاومت به صورت زیر تعریف می‌شود:

$$\forall A_i \in \mathcal{A}, \quad \text{اگر } A_d \text{ رفتار مخرب نشان دهد، } \pi_i \text{ باید همچنان به حداکثر رساندن پاداش خود ادامه دهد.}$$

تعریف ریاضی ایمنی ایمنی در MARL به معنای اطمینان از این است که سیستم در هیچ حالت خطرناکی وارد نمی‌شود. به طور ریاضی، ایمنی می‌تواند به صورت مجموعه‌ای از قیود تعریف شود که سیاست‌های یادگرفته شده باید آن‌ها را رعایت کنند:

$$\forall A_i \in \mathcal{A}, \quad \text{ایمنی: } u_i(s_i, s_{-i}) \geqslant \theta_i \quad \text{برای همه } s_i \in S_i,\ s_{-i} \in S_{-i}$$

که در آن θ_i آستانه‌ای است که برای هر عامل A_i تعیین شده و نشان‌دهنده حداقل پاداش قابل قبول است.

روش‌های افزایش مقاومت و ایمنی

برای افزایش مقاومت و ایمنی در MARL، روش‌های متعددی مورد استفاده قرار گرفته‌اند که در زیر به برخی از آن‌ها پرداخته می‌شود:

- **الگوریتم‌های مقاوم در برابر اختلالات:** این الگوریتم‌ها به گونه‌ای طراحی شده‌اند که بتوانند به سرعت

با تغییرات محیطی و حضور اختلالات سازگار شوند. به عنوان مثال، الگوریتم‌های مبتنی بر یادگیری تطبیقی که قادر به تغییر سیاست‌های خود در پاسخ به تغییرات محیط هستند.

- **فریم‌ورک‌های ایمنی:** چارچوب‌هایی برای تضمین ایمنی در تعاملات چندعاملی طراحی شده‌اند که شامل محدود کردن فضای عملیاتی و اعمال قیود بر سیاست‌های یادگیری است. این فریم‌ورک‌ها معمولاً شامل روش‌هایی برای نظارت و تنظیم رفتار عامل‌ها به منظور جلوگیری از وقوع رفتارهای خطرناک هستند.
- **آموزش در حضور اختلالات:** با آموزش الگوریتم‌ها در محیط‌هایی که شامل اختلالات هستند، می‌توان مقاومت الگوریتم‌ها را افزایش داد. این روش به الگوریتم اجازه می‌دهد تا در مواجهه با اختلالات غیرمنتظره، سیاست‌های مقاومتی یاد بگیرد.
- **استفاده از اصول نظریه بازی‌ها:** با بهره‌گیری از تعادل‌های نظریه بازی‌ها مانند تعادل نش (Nash Equilibrium)، می‌توان سیاست‌هایی طراحی کرد که در مواجهه با استراتژی‌های متغیر سایر عوامل، پایداری و ایمنی سیستم حفظ شود.

نمونه‌های کاربردی

برای نشان دادن کاربردهای عملی MARL در افزایش ایمنی و مقاومت، به چند مثال اشاره می‌شود:

سامانه‌های خودران: در خودروهای خودران چندعاملی، ایمنی یکی از اولویت‌های اصلی است. با استفاده از MARL و فرض کردن سایر خودروها به عنوان عوامل یا اختلالات، می‌توان الگوریتم‌هایی توسعه داد که در مواجهه با رفتارهای غیرمنتظره سایر خودروها، ایمن باقی بمانند.

مدیریت انرژی در شبکه‌های هوشمند: در شبکه‌های انرژی هوشمند، MARL می‌تواند برای مدیریت بهینه انرژی در حضور اختلالات مانند خرابی‌ها یا حملات سایبری استفاده شود. الگوریتم‌های مقاوم می‌توانند با تغییرات ناگهانی در تقاضا یا عرضه انرژی سازگار شده و پایداری شبکه را حفظ کنند.

ربات‌های همکاری‌کننده: در سیستم‌های رباتیک همکاری‌کننده، اطمینان از ایمنی تعاملات میان ربات‌ها حیاتی است. MARL می‌تواند برای طراحی سیاست‌های ایمن که در مواجهه با رفتارهای مخرب یا اختلالات داخلی، سیستم را پایدار نگه دارند، به کار رود.

محیط‌های صنعتی: در محیط‌های صنعتی که شامل چندین عامل نظیر ربات‌ها و ماشین‌آلات است، ایمنی و مقاومت سیستم‌ها از اهمیت بالایی برخوردار است. با استفاده از MARL، می‌توان الگوریتم‌هایی توسعه داد که در مواجهه با اختلالات مانند خرابی تجهیزات یا خطاهای انسانی، عملکرد سیستم را حفظ کنند.

چالش‌ها و فرصت‌ها

با وجود مزایای متعدد MARL در افزایش ایمنی و مقاومت سیستم‌ها، چالش‌هایی نیز در این زمینه وجود دارد:

- **پیچیدگی مدل‌سازی اختلالات:** مدل‌سازی دقیق اختلالات و رفتارهای مخرب می‌تواند پیچیده و زمان‌بر باشد، به ویژه در محیط‌های پیچیده و پویا.
- **هماهنگی میان عوامل:** هماهنگی موثر میان عوامل در حضور اختلالات نیازمند الگوریتم‌های پیچیده و کارآمد است که بتوانند به سرعت و بهینه با تغییرات محیطی سازگار شوند.
- **تعادل میان کارایی و ایمنی:** حفظ تعادل میان بهینه‌سازی کارایی سیستم و تضمین ایمنی در مواجهه با اختلالات یک چالش بزرگ است که نیازمند طراحی دقیق الگوریتم‌ها است.
- **نیاز به داده‌های بزرگ:** برای آموزش الگوریتم‌های مقاوم و ایمن، نیاز به مجموعه‌های داده بزرگ و متنوعی است که شامل انواع اختلالات و رفتارهای مخرب باشند.

با این حال، فرصت‌های بزرگی نیز در این زمینه وجود دارد. توسعه الگوریتم‌های پیشرفته MARL که بتوانند به طور موثر با اختلالات مواجه شوند، می‌تواند منجر به سیستم‌های هوشمند و خودکار با کارایی و ایمنی بالاتر شود. همچنین، پژوهش‌های ادامه‌دار در زمینه نظریه بازی‌ها و روش‌های مقاوم‌سازی الگوریتم‌های یادگیری تقویتی، پتانسیل بالایی برای بهبود MARL و کاربردهای آن فراهم می‌آورد.

نتیجه‌گیری

در نهایت، ایمنی و مقاومت در یادگیری تقویتی چندعاملی به عنوان عوامل کلیدی در توسعه سیستم‌های هوشمند و خودکار مطرح می‌گردند. با فرض کردن اختلالات و توسعه الگوریتم‌های مقاوم، می‌توان عملکرد سیستم‌های MARL را در مواجهه با چالش‌های مختلف بهبود بخشید. با توجه به اهمیت و کاربردهای گسترده MARL در حوزه‌های علمی و صنعتی، تحقیقات در زمینه افزایش ایمنی و مقاومت این الگوریتم‌ها همچنان ادامه خواهد یافت تا به سیستم‌هایی با قابلیت‌های بیشتر و عملکردی پایدارتر دست یابند.

۴-۲-۴ الگوریتم‌های یادگیری تقویتی چندعاملی

در حوزه یادگیری تقویتی چندعاملی (Multi-Agent Reinforcement Learning - MARL)، توسعه الگوریتم‌های خاص برای تعامل و همکاری میان عوامل مختلف از اهمیت بالایی برخوردار است. الگوریتم‌های MARL باید بتوانند با پیچیدگی‌ها و چالش‌های ناشی از تعاملات متعدد میان عوامل، عملکرد بهینه‌ای را ارائه دهند. در این بخش، به بررسی چند الگوریتم پیشرفته چندعاملی پرداخته می‌شود که هر یک ویژگی‌ها و مزایای خاص خود را دارند.

الگوریتم Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

الگوریتم MADDPG (Multi-Agent Deep Deterministic Policy Gradient) به عنوان یکی از الگوریتم‌های پیشرفته در MARL معرفی شده است که برای تعاملات همکاری میان عوامل طراحی گردیده است. این الگوریتم بر پایه الگوریتم DDPG توسعه یافته و از شبکه‌های عصبی عمیق برای یادگیری سیاست‌های بهینه استفاده می‌نماید. ویژگی اصلی MADDPG استفاده از اطلاعات دیگر عوامل در فرآیند یادگیری است که بهبود هماهنگی و همکاری میان عوامل را ممکن می‌سازد.

تعریف ریاضی در MADDPG، برای هر عامل i سیاست (Policy) و تابع ارزش (Critic) جداگانه‌ای تعریف شده است. فرمول به‌روزرسانی سیاست برای هر عامل به صورت زیر است:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s,a\sim\mathcal{D}} \left[\nabla_{\theta_i} Q_i^{\mu}(s,a)\big|_{a=\mu_i(s|\theta_i)} \cdot \nabla_{\theta_i} \mu_i(s|\theta_i) \right]$$

که در آن $Q_i^{\mu}(s,a)$ تابع ارزش عملیات برای عامل i و $\mu_i(s|\theta_i)$ سیاست عامل i می‌باشد.

ویژگی‌ها و مزایا

- **هماهنگی و همکاری بهتر:** استفاده از اطلاعات دیگر عوامل به بهبود هماهنگی و همکاری میان عوامل کمک می‌نماید.
- **قابلیت مقیاس‌پذیری:** توانایی کارکرد در محیط‌های با تعداد زیادی عامل.
- **پشتیبانی از سیاست‌های پیوسته:** مناسب برای مسائل با فضای عمل پیوسته.

الگوریتم Actor-Critic Soft Multi-Agent (MASAC)

الگوریتم MASAC (Multi-Agent Soft Actor-Critic) یکی از الگوریتم‌های پیشرفته در MARL است که بر پایه SAC توسعه یافته و برای تعاملات همکاری و رقابتی میان عوامل طراحی گردیده است. این الگوریتم با بهینه‌سازی انتروپی به همراه تابع ارزش، تعادل بهتری بین اکتشاف و بهره‌برداری ایجاد می‌نماید.

تعریف ریاضی در MASAC، تابع هدف برای هر عامل به صورت زیر تعریف می‌گردد:

$$J(\pi_i) = \mathbb{E}_{(s,a)\sim\rho_\pi}\left[\alpha\mathcal{H}(\pi_i(\cdot|s)) - Q_i^\phi(s,a)\right]$$

که در آن $\mathcal{H}(\pi_i(\cdot|s))$ انتروپی سیاست و α ضریب تنظیم تعادل بین انتروپی و تابع ارزش است.

ویژگی‌ها و مزایا

- **تعادل بین اکتشاف و بهره‌برداری:** با بهینه‌سازی انتروپی، اکتشاف در فضای عمل افزایش می‌یابد.
- **پایداری بیشتر:** استفاده از تکنیک‌های مختلف برای افزایش پایداری فرآیند یادگیری.
- **پشتیبانی از محیط‌های پیچیده:** قابلیت یادگیری در محیط‌های با تعاملات پیچیده میان عوامل.

الگوریتم Multi-Agent Proximal Policy Optimization (MA-PPO)

الگوریتم MA-PPO (Multi-Agent Proximal Policy Optimization) به عنوان یک توسعه‌یافته از PPO برای محیط‌های چندعاملی معرفی شده است. این الگوریتم با استفاده از محدودیت‌های نزدیک به سیاست جدید، کارایی و پایداری بالایی در تعاملات میان عوامل فراهم می‌آورد.

تعریف ریاضی تابع هدف MA-PPO به صورت زیر تعریف می‌گردد:

$$L^{CLIP}(\theta_i) = \mathbb{E}_t\left[\min\left(r_t(\theta_i)\hat{A}_t, \text{clip}(r_t(\theta_i), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$$

که در آن $r_t(\theta_i)$ نسبت احتمالات سیاست جدید به سیاست قدیمی و $\hat{A}_t$ برآورد مزایای عامل در زمان t می‌باشد.

ویژگی‌ها و مزایا

- **سادگی در پیاده‌سازی:** نسبت به الگوریتم‌های پیچیده‌تر، پیاده‌سازی آسان‌تری دارد.

- **پایداری بالا:** با محدود کردن تغییرات سیاست، از نوسانات بزرگ در فرآیند یادگیری جلوگیری می‌شود.
- **کارایی بالا:** عملکرد قابل توجهی در مسائل مختلف MARL دارد.

الگوریتم Multi-Agent Twin Delayed Deep Deterministic Policy Gradient (MA-TD۳)

الگوریتم MA-TD۳ (Multi-Agent Twin Delayed Deep Deterministic Policy Gradient) به عنوان یک توسعه‌یافته از TD۳ برای محیط‌های چندعاملی معرفی شده است. این الگوریتم با استفاده از دو شبکه ارزش مجزا و به‌روزرسانی تأخیری سیاست، کارایی و پایداری بالایی را در تعاملات میان عوامل فراهم می‌آورد.

تعریف ریاضی تابع هدف MA-TD۳ به صورت زیر تعریف می‌گردد:

$$\mathcal{L}(\phi_i) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q_i(s,a|\theta_i^Q) - \left(r + \gamma \min_{j=1,2} Q_j(s', \mu_i(s'|\theta_i^\mu)|\theta_j^{Q'})\right)\right)^2\right]$$

که در آن $i = 1, 2$.

ویژگی‌ها و مزایا

- **کاهش نوسانات تخمین تابع ارزش:** استفاده از دو شبکه ارزش برای کاهش تخمین‌های بیش از حد.
- **به‌روزرسانی تأخیری شبکه سیاست:** افزایش پایداری و کاهش نوسانات در فرآیند یادگیری.
- **پشتیبانی از فضای عمل پیوسته:** حفظ قابلیت کارکرد در مسائل با فضای عمل پیوسته.

ویژگی‌های مشترک الگوریتم‌های MADDPG، MASAC MA-PPO و MA-TD۳ هر کدام با توجه به ویژگی‌های خاص خود، توانسته‌اند در محیط‌های پیچیده و پویا با تعاملات متنوع میان عوامل، عملکرد قابل توجهی را ارائه دهند. این الگوریتم‌ها با استفاده از شبکه‌های عصبی عمیق، تعادل بین اکتشاف و بهره‌برداری را بهبود داده و قابلیت‌های مقاوم‌سازی سیستم‌ها را افزایش داده‌اند.

کاربردها الگوریتم‌های یادگیری تقویتی چندعاملی در حوزه‌های مختلفی مانند سامانه‌های خودران، مدیریت انرژی در شبکه‌های هوشمند، ربات‌های همکاری‌کننده و محیط‌های صنعتی مورد استفاده قرار می‌گیرند. این الگوریتم‌ها با بهینه‌سازی سیاست‌ها و کاهش نوسانات در فرآیند یادگیری، امکان همکاری و تعامل موثر میان عوامل را فراهم می‌آورند.

چالش‌ها و فرصت‌ها هر کدام از این الگوریتم‌ها با چالش‌های خاص خود مواجه هستند، از جمله نیاز به تنظیم دقیق پارامترها، محاسبه توابع ارزش پیچیده‌تر و افزایش هزینه محاسباتی. با این حال، فرصت‌های زیادی برای بهبود و توسعه این الگوریتم‌ها وجود دارد که می‌تواند به افزایش کارایی و قابلیت‌های یادگیری تقویتی چندعاملی کمک کند.

نتیجه‌گیری

الگوریتم‌های یادگیری تقویتی چندعاملی نقش مهمی در توسعه سیستم‌های هوشمند و خودکار ایفا می‌نمایند. الگوریتم‌های مختلف مانند MADDPG، MASAC MA-PPO و MA-TD۳ با ویژگی‌ها و مزایای منحصر به فرد خود، توانسته‌اند در محیط‌های پیچیده و پویا عملکرد قابل توجهی داشته باشند. این الگوریتم‌ها با بهینه‌سازی سیاست‌ها و کاهش نوسانات در فرآیند یادگیری، امکان همکاری و تعامل موثر میان عوامل را فراهم می‌آورند. با ادامه تحقیقات و بهبود این الگوریتم‌ها، انتظار می‌رود که MARL نقش کلیدی‌تری در پیشرفت تکنولوژی‌های هوشمند و خودکار ایفا نماید.

فصل ۵

مدل‌سازی محیط یادگیری سه جسمی

فصل ۶

شبیه‌سازی عامل درمحیط سه جسمی

Bibliography

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

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

[21] 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.

[22] 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.

[23] 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.

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

[25] 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.

[26] nobelprize.org. Jean tirole, 2021. [Online; accessed October 17, 2024], Available at https://www.nobelprize.org/prizes/economic-sciences/2014/tirole/facts/.

[27] 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.

[28] 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.

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

[30] 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. 2017.

[31] 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.

Abstract

In this study, a quadcopter stand with three degrees of freedom was controlled using game theory-based control. The first player tracks a desired input, and the second player creates a disturbance in the tracking of the first player to cause an error in the tracking. The move is chosen using the Nash equilibrium, which presupposes that the other player made the worst move.. In addition to being resistant to input interruptions, this method may also be resilient to modeling system uncertainty. This method evaluated the performance through simulation in the Simulink environment and implementation on a three-degree-of-freedom stand.

Keywords: Quadcopter, Differential Game, Game Theory, Nash Equilibrium, Three Degree of Freedom Stand, Model Base Design, Linear Quadratic Regulator

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

December 2024