



دانشگاه صنعتی امیرکبیر  
(پلی تکنیک تهران)  
دانشکده مهندسی کامپیوتر

پایان نامه کارشناسی

# استفاده از یادگیری تقویتی برای گل زدن در موقعیت تک به تک در فوتبال

نگارش

آراد فیروزکوهی

استاد راهنما

دکتر احسان ناظر فرد

فروردین ۱۴۰۳

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



دانشگاه صنعتی امیرکبیر  
(پلی تکنیک تهران)  
دانشکده مهندسی کامپیوتر

پایان نامه کارشناسی

# استفاده از یادگیری تقویتی برای گل زدن در موقعیت تک به تک در فوتبال

نگارش

آراد فیروزکوهی

استاد راهنما

دکتر احسان ناظر فرد

فروردین ۱۴۰۳

# صفحه فرم ارزیابی و تصویب پایان نامه - فرم تأیید اعضاء کمیته دفاع

در این صفحه فرم دفاع یا تأیید و تصویب پایان نامه موسوم به فرم کمیته دفاع - موجود در پرونده آموزشی - را قرار دهید.

## نکات مهم:

- نگارش پایان نامه/رساله باید به **زبان فارسی** و بر اساس آخرین نسخه دستورالعمل و راهنمای تدوین پایان نامه های دانشگاه صنعتی امیرکبیر باشد.(دستورالعمل و راهنمای حاضر)
- رنگ جلد پایان نامه/رساله چاپی کارشناسی، کارشناسی ارشد و دکترا باید به ترتیب مشکی، طوسی و سفید رنگ باشد.
- چاپ و صحافی پایان نامه/رساله بصورت **پشت و رو(دورو)** بلامانع است و انجام آن توصیه می شود.



دانشگاه صنعتی امیرکبیر  
(پلی تکنیک تهران)

به نام خدا

## تعهدنامه اصالت اثر

تاریخ: فروردین ۱۴۰۳

اینجانب **آراد فیروزکوهی** متعهد می‌شوم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب تحت نظارت و راهنمایی اساتید دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آنها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مآخذ ذکر گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است. در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان‌نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان‌نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مآخذ بلامانع است.

**آراد فیروزکوهی**

امضا

تقدیم به پدر بزرگوار و مادر عزیز و مهربانم

که در سختی ها و دشواری های زندگی، همواره یاری دلسوز و فداکار و پشتیبانی  
محکم و مطمئن برایم بوده اند.

# سپاس‌گزاری

از استاد دلسوز و محترم؛ جناب دکتر ناظر فرد که با صبر و حوصله، از هیچ کمکی در مسیر انجام این پروژه و نوشتن این پایان‌نامه از من دریغ ننمودند؛ کمال تشکر و قدرانی را دارم.

آراد فیروز کوهی  
فروردین ۱۴۰۳

## چکیده

این پایان نامه به اکتشاف عمیق یادگیری تقویتی و نحوه عملکرد آن در گل زدن در موقعیت های تک به تک فوتبال می پردازد. ابتدا، ما به بررسی مفاهیم یادگیری تقویتی و اصول اساسی آن می پردازیم تا درکی جامع از چگونگی تصمیم گیری و یادگیری ماشین در محیط های پویا ارائه شود. سپس پلتفرم شبیه ساز دو بعدی فوتبال ربوکاپ که از آن قرار است استفاده کنیم، معرفی می شود. در این پژوهش، الگوریتم های مختلف یادگیری تقویتی از جمله شبکه کیو عمیق، یادگیری تقویتی مبتنی بر سیاست و دیگر رویکردهای پیشرفته بررسی شده اند تا تأثیر آنها بر بهبود عملکرد بازیکنان در گل ها ارزیابی گردد. بخش مهمی از این تحقیق به ایجاد یک روش استاندارد برای انجام یادگیری تقویتی در شبیه سازی فوتبال ربوکاپ با استفاده از چارچوب جیم اختصاص یافته است؛ که نوید بخش یکپارچه سازی و استانداردسازی روش های یادگیری تقویتی در این حوزه است. در نهایت خواهیم دید که چگونه استفاده از این الگوریتم ها می تواند به طور موثری به بازیکنان کمک کنند تا در موقعیت های تک به تک بهترین تصمیم ها را بگیرند و گل های بیشتری بزنند. علاوه بر این، پایان نامه به بررسی تأثیر پارامترها و تنظیمات مختلف الگوریتمی بر عملکرد یادگیری تقویتی می پردازد و راهکارهایی برای بهبود عملکرد ارائه می دهد.

## واژه های کلیدی:

یادگیری تقویتی، یادگیری عمیق، شبیه ساز دوبعدی فوتبال، جیم، تک به تک با دروازه بان



# فهرست مطالب

صفحه

عنوان

۱	.....	۱ مقدمه
۲	.....	۱-۱ مقدمه
۲	.....	۲-۱ تعریف مساله
۳	.....	۳-۱ کارهای مشابه
۴	.....	۴-۱ ابزارها و نرم افزارهای مورد استفاده
۷	.....	۲ یادگیری تقویتی
۸	.....	۱-۲ مقدمه
۹	.....	۲-۲ اصول یادگیری تقویتی
۹	.....	۱-۲-۲ عامل، محیط، حالت، عمل و پاداش
۹	.....	۲-۲-۲ فرایند تصمیم گیری مارکوف
۱۰	.....	۳-۲-۲ سیاست، تابع ارزش، و تابع ارزش عمل
۱۱	.....	۳-۲ الگوریتم های پایه یادگیری تقویتی
۱۱	.....	۱-۳-۲ برنامه نویسی پویا
۱۲	.....	۲-۳-۲ یادگیری به کمک نمونه برداری مونته کارلو
۱۳	.....	۳-۳-۲ یادگیری به کمک تفاوت زمانی
۱۶	.....	۴-۲ استفاده از تقریب گر تابع
۱۷	.....	۵-۲ الگوریتم شبکه کیو عمیق
۱۸	.....	۱-۵-۲ بازیابی تجربه
۱۸	.....	۲-۵-۲ استفاده از شبکه هدف
۱۸	.....	۳-۵-۲ فرآیند یادگیری
۱۹	.....	۴-۵-۲ اثرات برخی از ابرپارامترها
۲۰	.....	۶-۲ الگوریتم بهبود گرادین سیاست معین عمیق
۲۳	.....	۷-۲ جمع بندی
۲۴	.....	۳ محیط فوتبال روبوکاپ

۲۵	۱-۳ معرفی لیگ
۲۵	۱-۱-۳ اهداف لیگ
۲۵	۲-۱-۳ ویژگی‌های محیط و قوانین بازی
۲۶	۲-۳ کد پایه ایجنت
۲۷	۳-۳ معرفی رفتارهای ممکن
۲۷	۱-۳-۳ رفتارهای سطح پایین
۲۸	۲-۳-۳ رفتارهای سطح متوسط
۲۸	۳-۳-۳ رفتارهای سطح بالا
۲۸	۴-۳ حالت دید کامل
۲۹	۵-۳ معرفی حالت پناستی
۳۰	۶-۳ کار با مربی تمرینی برای تولید محیط قابل تکرار
۳۰	۷-۳ جمع‌بندی
۳۲	۴ پیاده‌سازی و آماده‌سازی محیط استاندارد
۳۳	۱-۴ زیرساخت پایتونی تهاجم نصف زمین
۳۳	۲-۴ کد پایه پایرس
۳۴	۳-۴ کد پایه جی‌آرپی‌سی
۳۷	۴-۴ محیط استاندارد جیم
۳۷	۱-۴-۴ توصیف رابط و توابع موجود
۳۹	۲-۴-۴ نحوه اداام با فضای ربوکاپ
۴۱	۳-۴-۴ فضای حالت و فضای عمل عامل
۴۲	۴-۴-۴ طراحی پاداش
۴۳	۵-۴ پیاده‌سازی یادگیری تقویتی
۴۳	۱-۵-۴ پیاده‌سازی شبکه کیو عمیق
۴۴	۲-۵-۴ پیاده‌سازی سایر الگوریتم‌ها به کمک کتابخانه
۴۵	۶-۴ جمع‌بندی
۴۶	۵ مقایسه، آزمایش‌ها، و نتایج
۴۷	۱-۵ فرآیند آزمایش

۴۷	۲-۵ ارزیابی الگوریتم‌ها
۵۰	۳-۵ بهبود تابع پاداش
۵۰	۴-۵ جداسازی عمل شوت
۵۰	۱-۴-۵ شوت با رفتار سطح بالای کد پایه
۵۱	۲-۴-۵ عمل شوت به نقاط ثابت دروازه
۵۲	۵-۵ تغییر گسسته‌سازی
۵۴	۶-۵ بررسی تعمیم پذیری
۵۶	۷-۵ جمع‌بندی
۵۷	۶ جمع‌بندی، نتیجه‌گیری و پیشنهادات
۵۸	۱-۶ جمع‌بندی و نتیجه‌گیری
۵۸	۲-۶ کارهای آتی
۶۰	منابع و مراجع

# فهرست تصاویر

صفحه

شکل

۱-۲	تعامل عامل و محیط . . . . .	۹
۲-۲	تاثیر پارامتر $\lambda$ بر اهمیت پاداش‌های آینده. محور افقی نماینده تعداد گام‌ها، و محور عمودی نماینده وزن این بازگشت متناظر با آن است. . . . .	۱۴
۳-۲	روش‌های مختلف استفاده از تقریب‌گر تابع . . . . .	۱۶
۴-۲	شبکه‌های عصبی بازیگر و نقاد در الگوریتم DDPG و مقایسه با DQN . . . . .	۲۱
۱-۳	نمونه دید یک بازیکن. در این مثال، بازیکن ۳، بازیکن ۵ و ۸ و توپ را می‌بیند و از موقعیت سایر بازیکنان اطلاعی ندارد. . . . .	۲۶
۲-۳	ارتباط بین محیط و عامل، و روش محاسبه پاداش . . . . .	۲۹
۳-۳	نمونه‌ای از بازی در حالت پنالیتی . . . . .	۳۰
۱-۴	پرچم‌های کنار زمین، که برای موقعیت‌یابی استفاده می‌شوند. . . . .	۳۴
۲-۴	نحوه تعریف اشیا در جی‌آرپی‌سی . . . . .	۳۵
۳-۴	نحوه تعریف توابع در جی‌آرپی‌سی . . . . .	۳۵
۴-۴	پروتکل‌های استفاده‌شده برای ارتباط بین اجزای مسابقه . . . . .	۳۶
۵-۴	نحوه کارکرد و اتصال کد پایه جی‌آرپی‌سی به سرور مسابقات و نمایشگر بازی . . . . .	۳۶
۶-۴	نمونه‌ای از محیط‌های آماده شده در پلتفرم جیم . . . . .	۳۷
۷-۴	نحوه کلی ارتباط کد پایه، تصمیم‌گیرنده جی‌آرپی‌سی و جیم . . . . .	۴۰
۸-۴	ترتیب فراخوانی توابع برای اجرای جیم با سرور ربوکاپ . . . . .	۴۰
۹-۴	تقسیم‌بندی فضای عمل به ۶ زاویه و ۲ سرعت . . . . .	۴۲
۱۰-۴	کاهش نرخ کاوش تصادفی در طی آموزش . . . . .	۴۴
۱۱-۴	استفاده از کتاب‌خانه استیبل بیس‌لاینز . . . . .	۴۵
۱-۵	نمودار سه پارامتر متفاوت برای یک اجرای الگوریتم DQN . . . . .	۴۸
۲-۵	نمودار سه پارامتر متفاوت برای یک اجرای الگوریتم DDPG . . . . .	۴۹
۳-۵	نتیجه ۱۰۰ ضربه پنالیتی با شوت سطح بالا مقابل کد پایه ایجنت برای عامل یادگیری‌شده . . . . .	۵۱
۴-۵	نتیجه تست ۱۰۰ پنالیتی مقابل کد پایه ایجنت با ۵ نقطه شوت . . . . .	۵۲

۵-۵ نمودار میانگین پاداش و میانگین نرخ گل زنی DQN با تغییرات گسسته سازی . . . . . ۵۳

۶-۵ نتیجه تست ۱۰۰ پناستی مقابل کد پایه ایجنت با بهبود روش گسسته سازی . . . . . ۵۳

## فهرست جداول

صفحه

جدول

۴۹	۱-۵	بهترین نتایج الگوریتم‌ها مقابل کد پایه ایجنت برای صد پناالتی.
۵۴	۲-۵	نتایج تست الگوریتم DDPG علیه تیم‌های مختلف
۵۴	۳-۵	نتایج تست الگوریتم DQN بهبود یافته مقابل تیم‌های مختلف
۵۵	۴-۵	نتایج تست الگوریتم DDPG علیه تیم‌های مختلف با آموزش مقابل تیم هلیوس
۵۵	۵-۵	نتایج تست الگوریتم DDPG علیه تیم‌های مختلف با آموزش مقابل تیم یوشان

# فصل اول

## مقدمه

## ۱-۱ مقدمه

روبوکاپ<sup>۱</sup> مجموعه‌ای از مسابقات است که به هدف ترویج و پیشرفت علم رباتیک و هوش مصنوعی به صورت سالانه برگزار می‌شود. در این مسابقات تیم‌های مختلفی از سراسر دنیا شرکت می‌کنند و با استفاده از ربات‌های خود در مسابقات مختلفی همچون ربات‌های انسان‌نما، ربات‌های چرخ‌دار، و لیگ‌های شبیه‌سازی به رقابت می‌پردازند. از طریق این مسابقات، تیم‌ها می‌توانند تکنیک‌های جدیدی را برای حل مسائل مختلف در علم رباتیک و هوش مصنوعی ارائه کنند و از تجربیات دیگر تیم‌ها بهره‌مند شوند.

یکی از لیگ‌های معروف روبوکاپ، لیگ شبیه‌ساز دو بعدی فوتبال<sup>۲</sup> است که محیطی چندعامله با دید ناقص<sup>۳</sup> را برای تیم‌ها فراهم می‌کند. تمرکز این لیگ بیشتر در راستای تصمیم‌گیری در محیط‌های چندعامله و هماهنگی بین عوامل است. در این لیگ بازیکنان با کمک حسگرهای مجازی، سایر بازیکنان و توپ را مشاهده می‌کنند و بر اساس این اطلاعات باید تصمیم‌گیری‌های مناسبی انجام دهند. تصمیم‌های عامل می‌توانند از نوع حرکت، ضربه زدن به توپ، تکل زدن، چرخاندن بدن و گردن باشد. در فصل‌های آتی به توضیحات بیشتری در مورد این لیگ و نحوه عملکرد آن خواهیم پرداخت.

این محیط با توجه به چالش‌هایی که دارد و از سوی دیگر به علت تکرار پذیری و ابزارهایی که در اختیار می‌گذارد، یک محیط مناسب برای تست و ارزیابی الگوریتم‌های هوش مصنوعی و یادگیری تقویتی است. از این رو انجام پژوهش‌های مختلف در این محیط می‌تواند به پیشرفت علمی و فناوری کمک کند.

## ۲-۱ تعریف مساله

در این پروژه قصد داریم از یکی از محیط‌های موجود در لیگ شبیه‌سازی دو بعدی فوتبال استفاده کنیم و با چند الگوریتم متفاوت مسئله تک به تک با دروازه‌بان را حل کنیم. سپس راه حل‌های ارائه شده را با یکدیگر مقایسه کرده و به نتیجه‌گیری‌هایی در مورد عملکرد هر الگوریتم و تاثیر پارامترهای مختلف آنها بر عملکرد می‌رسیم.

در این پروژه قصد داریم عامل مهاجم را به صورت یک الگوریتم یادگیری تقویتی آموزش دهیم تا بتواند بهترین حرکت را برای گل‌زنی انجام دهد. از آنجا که در صورتی که در زمان عدم تصاحب توپ بهترین حرکت به طور قطعی تعقیب توپ است، با کمک یادگیری تقویتی نحوه ضربه زدن به توپ برای

<sup>1</sup>RoboCup

<sup>2</sup>2D Soccer Simulation

<sup>3</sup>Partial Observation



فریب دروازه‌بان را بررسی می‌کنیم. برای دروازه‌بان رقیب به طور پیش‌فرض از کد پایه‌ای که منشا اکثر تیم‌های امروزی است استفاده می‌کنیم؛ اما در نهایت اثر تغییر عامل دروازه‌بان را نیز بررسی می‌کنیم. برای رسیدن به این هدف، قصد داریم محیط شبیه‌ساز دو بعدی فوتبال را به شکل یکی از محیط‌های معروف یادگیری تقویتی، یعنی محیط جیم<sup>۴</sup> که توسط شرکت اوپن‌ای‌آی<sup>۵</sup> ارائه شده پیاده‌سازی کنیم.

## ۳-۱ کارهای مشابه

بسیاری از تیم‌ها سالانه تلاش می‌کنند با کمک یادگیری ماشین<sup>۶</sup> و هوش مصنوعی، عامل‌های خود را بهبود بخشند. در این بخش به برخی از این تلاش‌ها اشاره می‌کنیم.

تیم برین‌استرمرز<sup>۷</sup> که در سال ۱۹۹۸ تشکیل شد، یکی از قدیمی‌ترین تیم‌های لیگ شبیه‌ساز دو بعدی است که قصد داشته تمام بخش‌های کد خود را به کمک یادگیری تقویتی پیاده‌سازی کند. این تیم در مراحل ابتدایی بخش‌های سطح پایین تیم خود را با کمک یادگیری تقویتی بهبود دادند، و به مرور زمان مسائل سطح بالا و تاکتیکی خود را نیز به سمت یادگیری تقویتی گسترش دادند [۱، ۲].

تیم اف‌آرای-یونایتد<sup>۸</sup> که باقی‌مانده تیم برین‌استرمرز و وارث کدهای آن است، سالیانه در گزارش فنی خود یک عمل را برای بهبود به کمک یادگیری تقویتی انتخاب می‌کند [۳]. این تیم یک زیرساخت خاص منظوره دارد که به کمک آن جمع‌آوری داده و تکرار آزمایش‌ها را به سادگی انجام می‌دهد. متأسفانه این زیرساخت‌های خاص منظوره خصوصی اند و در دسترس عموم نیستند.

پروژه متن‌باز<sup>۹</sup> تهاجم نیمه میدانی<sup>۱۰</sup> یک پروژه بر پایه محیط شبیه‌ساز دو بعدی است که با کمک آن می‌توان یک تیم مهاجم و یک تیم مدافع را اجرا کرد [۴، ۵]. این پروژه به منظور توسعه الگوریتم‌های یادگیری تقویتی برای تیم‌های مهاجم و مدافع ایجاد شده است، و امکان نوشتن عامل به کمک زبان پایتون<sup>۱۱</sup> و سی‌پلاس‌پلاس<sup>۱۲</sup> را فراهم می‌کند. برای انجام این پروژه این می‌شد تیم‌ها را با یک مهاجم و مدافع اجرا کرد، اما در فصل‌های آینده با معایب و مشکل‌های این پروژه بیشتر آشنا می‌شویم.

<sup>۴</sup>Gym

<sup>۵</sup>OpenAI

<sup>۶</sup>Machine Learning

<sup>۷</sup>Brain Stormers

<sup>۸</sup>FRA-United

<sup>۹</sup>Open Source

<sup>۱۰</sup>Half Field Offense

<sup>۱۱</sup>Python

<sup>۱۲</sup>C++

## ۴-۱ ابزارها و نرم‌افزارهای مورد استفاده

در این بخش به معرفی ابزارها و نرم‌افزارهایی که در این پروژه استفاده شده‌اند، و نحوه استفاده از این ابزارها می‌پردازیم.

- **سرور مسابقات:** سرور مسابقات یک نرم‌افزار است که برای اجرای مسابقات روبوکاپ استفاده می‌شود. این نرم‌افزار وظیفه شبیه‌سازی عمل‌های بازیکنان، تشکیل حالت بازی و شبیه‌سازی فیزیک بازی را بر عهده دارد. در کنار این وظایف، این نرم‌افزار ارتباط با عامل‌ها را برقرار می‌کند و اطلاعات مربوط به حالت بازی را به عامل‌ها ارسال می‌کند.
- **نمایش‌گر مسابقات:** این نرم‌افزار وظیفه نمایش بازی‌های مسابقات را بر عهده دارد. این نرم‌افزار از اطلاعاتی که از سرور مسابقات دریافت می‌کند برای نمایش حالت بازی استفاده می‌کند.
- **کد پایه ایجنت دو بعدی<sup>۱۳</sup>:** این کد پایه یک کد پایه برای عامل‌های مسابقات دو بعدی است که در سال ۲۰۱۲ توسط هیدهیسا آکیاما از تیم هلیوس ارائه شد [۶]. این کد پایه شامل توابعی برای اتصال به سرور، تصمیم‌گیری برای حرکت، ضربه زدن به توپ، تکل زدن، و ... است. این کد پایه به صورت متن‌باز در دسترس تیم‌های مختلف است و می‌تواند به عنوان یک نقطه شروع برای توسعه عامل‌های جدید استفاده شود.
- **سی پلاس پلاس:** از آنجا که کد پایه ایجنت به زبان سی پلاس پلاس نوشته شده است، برای توسعه و تغییرات در این کد پایه نیاز به استفاده از این زبان را داریم.
- **پایتون:** با توجه به محبوبیت زبان برنامه نویسی پایتون و امکانات بالای آن برای پیاده‌سازی الگوریتم‌های یادگیری ماشین، از این زبان برنامه نویسی برای پیاده‌سازی محیط آموزش عامل مهاجم استفاده می‌کنیم. در این پروژه از پایتون نسخه ۳.۱۱ استفاده شده است.
- **جی آرپی سی<sup>۱۴</sup>:** این ابزار یک روش اجرای کد از راه دور است. از این رو که جی آرپی سی مستقل از زبان است، می‌توان از آن برای ارسال پیام و فراخوانی توابع بین دو برنامه به زبان‌های مختلف استفاده کرد. در ادامه توضیح خواهیم داد که چگونه به این کمک این ابزار محیط پایتونی‌ای که برای آموزش عامل مهاجم استفاده می‌کنیم، را به محیط سی پلاس پلاسی که برای اجرای مسابقات استفاده می‌کنیم، متصل کرد.

<sup>13</sup> Agent2D

<sup>14</sup> gRPC

• **کتابخانه چند رشته‌ای<sup>۱۵</sup>:** از این کتابخانه برای اجرای چندین رشته همزمان در پایتون استفاده می‌کنیم. از این رو به این کتابخانه نیاز داریم که حداقل یک رشته برای پاسخ درخواست‌های سرور و یک رشته برای اجرای الگوریتم‌های یادگیری تقویتی داشته باشیم. در فصل‌های بعدی به طور دقیق با زیرساخت پیاده‌سازی و اتصال این دو بخش به یکدیگر خواهیم پرداخت.

• **محیط جیم<sup>۱۶</sup>:** این محیط یک رابط استاندارد برای آموزش عامل‌های یادگیری تقویتی است [۷]. از ابزارهای مختلفی برای پیاده‌سازی محیط‌های آموزشی و ارتباط با عامل‌ها پشتیبانی می‌کند. به کمک این ابزار می‌توان محیط‌های آموزشی متنوعی را پیاده‌سازی کرد و از الگوریتم‌های یادگیری تقویتی مختلف برای آموزش عامل‌ها استفاده کرد، به گونه‌ای که محیط جیم تقریباً در یادگیری تقویتی به صورت همگانی به عنوان یک محیط استاندارد شناخته شده است، و معروف‌ترین محیط‌های تحقیقاتی مانند ماشین تپهنورد<sup>۱۷</sup> [۸]، بازی‌های آتاری<sup>۱۸</sup>، و آونگ معکوس شده<sup>۱۹</sup>، در این محیط پیاده‌سازی شده‌اند. از این محیط برای پیاده‌سازی محیط آموزش عامل مهاجم استفاده می‌کنیم.

• **کتابخانه پای تورچ<sup>۲۰</sup>:** این کتابخانه یکی از کتابخانه‌های معروف برای پیاده‌سازی شبکه‌های عصبی و الگوریتم‌های یادگیری عمیق است. از این کتابخانه برای پیاده‌سازی مدل‌های یادگیری تقویتی و شبکه‌های عصبی در این پروژه استفاده می‌کنیم. پیاده‌سازی از پیش آماده شده گرادیان و بهینه‌سازهای مختلف، و امکان استفاده از پردازنده‌های گرافیکی برای محاسبات سریع، از ویژگی‌های این کتابخانه است [۹].

• **کتابخانه استیبل بیس لاینز<sup>۲۱</sup>:** این کتابخانه یک کتابخانه معروف برای پیاده‌سازی الگوریتم‌های یادگیری تقویتی است [۱۰]. این کتابخانه از الگوریتم‌های معروفی همچون DQN، A2C، PPO، و ... پشتیبانی می‌کند. مدل‌ها به صورت مستقل پیاده‌سازی شده‌اند، اما برای نمایش قدرت پیاده‌سازی یک محیط استاندارد یادگیری تقویتی، از این کتابخانه نیز استفاده می‌کنیم.

<sup>15</sup>Threading

<sup>16</sup>Gym

<sup>17</sup>Mountain Car

<sup>18</sup>Atari Games

<sup>19</sup>Inverted Pendulum

<sup>20</sup>PyTorch

<sup>21</sup>Stable Baselines

- **کتابخانه تنسوربرد<sup>۲۲</sup>:** این کتابخانه یک ابزار بسیار رایج برای ذخیره‌سازی اطلاعات و ویژگی‌های مدل‌های یادگیری عمیق در حین آموزش است. رایج است این کتابخانه را با کتابخانه تنسورفلو<sup>۲۳</sup> استفاده کرد؛ اما در این پروژه از آن به صورت مستقل استفاده می‌کنیم.
- **کتابخانه مت‌پلات‌لیب<sup>۲۴</sup>:** از این کتابخانه برای رسم نمودارها و نمایش داده‌ها استفاده می‌کنیم.

---

<sup>22</sup>TensorBoard

<sup>23</sup>textbfTensorFlow

<sup>24</sup>Matplotlib

## فصل دوم

### یادگیری تقویتی

## ۱-۲ مقدمه

به طور عمومی، یادگیری ماشین به دسته‌ای از الگوریتم‌ها و روش‌های محاسباتی گفته می‌شود که به ماشین‌ها امکان یادگیری از داده‌ها و تجربه‌های خود را می‌دهند. یادگیری ماشین به دو دسته اصلی تقسیم می‌شود: یادگیری نظارت‌شده<sup>۱</sup> و یادگیری بدون نظارت<sup>۲</sup>. در یادگیری نظارت‌شده، مدل به کمک داده‌های برچسب‌خورده آموزش داده می‌شود و سپس برای پیش‌بینی خروجی‌های جدید از این مدل استفاده می‌شود. در یادگیری بدون نظارت، مدل بدون داده‌های برچسب‌خورده آموزش داده می‌شود و باید خودش مفاهیم و الگوهای موجود در داده‌ها را کشف کند. یادگیری تقویتی در این دو دسته قرار نمی‌گیرد و به عنوان یک دسته جداگانه در نظر گرفته می‌شود.

یادگیری تقویتی یک روش یادگیری ماشین است که به عامل اجازه می‌دهد تا رفتار بهینه را از طریق تعامل و آزمون و خطا با محیط یاد بگیرد. در این روش، عامل مشاهدات خود را از محیط (به کمک حسگر) دریافت کرده، و بر اساس آن تصمیم خود را اخذ می‌کند. پس از انجام هر عمل، محیط به عامل پاداشی می‌دهد که نشان‌دهنده عملکرد عامل در آن حالت است. هدف این است که عامل مجموع پاداش‌های دریافتی خود را بیشینه کند؛ که در صورت صحیح بودن تعریف سیگنال پاداش، معمولاً به معنای رسیدن به هدف مطلوب است.

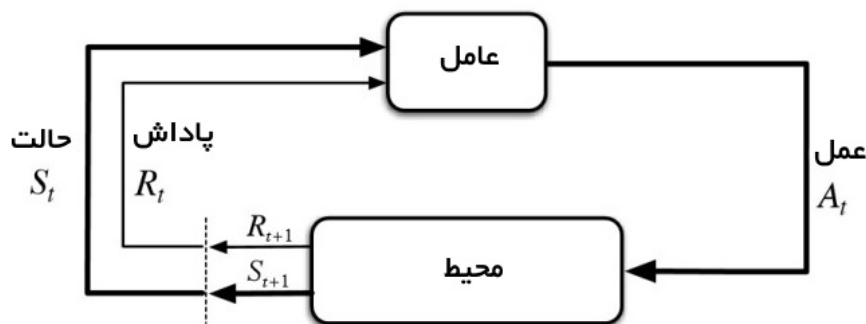
یادگیری تقویتی با سایر رویکردهای یادگیری ماشین مانند یادگیری نظارت‌شده و یادگیری بدون نظارت تفاوت‌های زیادی دارد، و معمولاً بر مسائلی به کار می‌رود که تمرکز بر تصمیم‌گیری یا پیدا کردن سیاست بهینه بدون نیاز به داده‌های برچسب‌خورده است. در عوض، عامل با اکتشاف و بهره‌برداری ابتدا انتقالات بین حالت‌ها و پاداش‌های مرتبط با آن‌ها را یاد می‌گیرد، و سپس سیاستی را یاد می‌گیرد که مجموع پاداش‌ها را بیشینه کند.

<sup>۱</sup>Supervised Learning<sup>۲</sup>Unsupervised Learning

## ۲-۲ اصول یادگیری تقویتی

### ۱-۲-۲ عامل، محیط، حالت، عمل و پاداش

در یادگیری تقویتی، عامل<sup>۳</sup> تصمیم‌گیرنده برای رسیدن به هدف خود با محیط<sup>۴</sup> تعامل دارد. محیط معمولاً به صورت مجموعه‌ای از حالت‌ها<sup>۵</sup> و عمل‌هایی<sup>۶</sup> که عامل می‌تواند انجام دهد مدل می‌شود. در هر گام، عامل مشاهده‌ای از حالت محیط را دریافت می‌کند و بر اساس آن تصمیمی اتخاذ می‌کند. پس از انجام عمل، محیط به عامل پاداش<sup>۷</sup> می‌دهد که نشان‌دهنده عملکرد عامل در آن حالت است.



شکل ۱-۲: تعامل عامل و محیط

در اکثر مواقع، یادگیری تقویتی در مسائلی استفاده می‌شود که بتوان آن را به دنباله‌ای از گام‌ها تقسیم کرد که قطعاً به یک حالت پایانی می‌رسد. هر یک از این دنباله‌ها را قسمت<sup>۸</sup> می‌نامند.

### ۲-۲-۲ فرایند تصمیم‌گیری مارکوف

فرایند تصمیم‌گیری مارکوف<sup>۹</sup> مدلی است برای تصمیم‌گیری در محیط‌هایی که به صورت مارکوف هستند. محیط‌های دارای خاصیت مارکوف، محیط‌هایی هستند که حالت بعدی به صورت کامل به حالت فعلی و عمل انجام شده وابسته است.

یک فرایند تصمیم‌گیری مارکوف، چهارتایی  $(S, A, P, R)$  است که در آن:

<sup>۳</sup>Agent

<sup>۴</sup>Environment

<sup>۵</sup>State

<sup>۶</sup>Action

<sup>۷</sup>Reward

<sup>۸</sup>Episode

<sup>۹</sup>Markov Decision Process (MDP)

- $S$  مجموعه‌ی تمام حالت‌های ممکن محیط است.
  - $A$  مجموعه‌ی تمام عمل‌های ممکن است.
  - $P$  تابع انتقال<sup>۱۰</sup> است که به ازای هر حالت و عمل، توزیع احتمال حالت بعدی را مشخص می‌کند.
  - $R$  تابع پاداش<sup>۱۱</sup> است که به ازای هر حالت و عمل، پاداش مورد انتظار را مشخص می‌کند.
  - نرخ تخفیف  $\gamma$  نیز معمولاً به عنوان یک پارامتر دیگر در نظر گرفته می‌شود که نشان‌دهنده‌ی اهمیت پاداش‌های آینده نسبت به پاداش‌های فعلی است.
- همانطور که گفته شد، در هر گام عامل با اخذ تصمیم خود، محیط را به حالت جدیدی می‌برد و پاداشی دریافت می‌کند. به مجموع کل پاداش‌هایی که عامل در یک قسمت دریافت می‌کند، خروجی<sup>۱۲</sup> گفته می‌شود.

$$G_t = R_{t+1} + \gamma \times R_{t+2} + \gamma^2 \times R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k \times R_{t+k+1} \quad (1-2)$$

## ۳-۲-۲ سیاست، تابع ارزش، و تابع ارزش عمل

سیاست<sup>۱۳</sup> یک تابع از حالت‌ها به عمل‌ها است که نشان‌دهنده‌ی رفتار عامل در هر حالت محیط است. در واقع مسئله یادگیری تقویتی را می‌توان به «یافتن سیاستی که مجموع پاداش‌ها را بیشینه می‌کند» تعبیر کرد.

$$\pi(a|s) = \mathbb{P}\{A_t = a | S_t = s\} \quad (2-2)$$

تابع ارزش<sup>۱۴</sup>  $V(s)$  یک تابع از حالت‌ها است که نشان‌دهنده‌ی میزان پاداش مورد انتظار از یک حالت

<sup>10</sup>Transition Function

<sup>11</sup>Reward Function

<sup>12</sup>Return

<sup>13</sup>Policy

<sup>14</sup>Value Function



تا به انتهای قسمت است.

$$v_{\pi}(s) = \mathbb{E}_{\pi}\{G_t | S_t = s\} = \sum_{a \in A} \pi(a|s) \times \{R_s^a + \gamma \times \sum_{s' \in S} p(s'|s, a) \times v_{\pi}(s')\} \quad (3-2)$$

تابع ارزش عمل<sup>۱۵</sup>  $Q(s, a)$  نیز مشابه تابع ارزش است؛ با این تفاوت که به جای حالت، از یک حالت و یک عمل مشخص محاسبه می‌شود. رایج است که به این تابع، تابع کیو گفته شود.

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}\{G_t | S_t = s, A_t = a\} = R_s^a + \gamma \times \sum_{s' \in S} p(s'|s, a) \times v_{\pi}(s') \quad (4-2)$$

به این معادلات، که از کلیدی‌ترین روابط در یادگیری تقویتی هستند، معادله بلمن<sup>۱۶</sup> گفته می‌شود.

## ۳-۲ الگوریتم‌های پایه یادگیری تقویتی

### ۱-۳-۲ برنامه‌نویسی پویا

همان‌طور که در معادله ۳-۲ مشخص است، می‌توان تابع ارزش را به صورت بازگشتی محاسبه کرد. این روش برنامه‌نویسی پویا<sup>۱۷</sup> نام دارد.

#### یادگیری به کمک تکرار ارزش

در روش تکرار ارزش<sup>۱۸</sup>، ابتدا تابع ارزش را به صورت تصادفی مقداردهی می‌کنیم و سپس آن را به صورت بازگشتی به‌روزرسانی می‌کنیم. قابل اثبات است که این روش به تابع ارزش بهینه همگرا می‌شود [۱۱]. به صورت شهودی هم می‌توان دید که ارزش از سمت حالت‌های پایانی به سمت حالت‌های ابتدایی به‌روزرسانی می‌شود.

<sup>15</sup>Action Value Function

<sup>16</sup>Bellman Equation

<sup>17</sup>Dynamic Programming

<sup>18</sup>Value Iteration

## یادگیری به کمک تکرار سیاست

در روش تکرار سیاست<sup>۱۹</sup>، ابتدا سیاست را به صورت تصادفی مقداردهی می‌کنیم و سپس تابع ارزش را برای آن محاسبه می‌کنیم. سپس سیاست را به صورتی تغییر می‌دهیم که در هر گام عامل به حالت بعدی ممکن با بیشترین ارزش برود. این فرآیند را تا زمانی که سیاست تغییر نکند ادامه می‌دهیم. قابل اثبات است که این روش به سیاست بهینه همگرا می‌شود.

در عمل، با توجه به اینکه معمولاً به تابع انتقال دسترسی نداریم، نمی‌توانیم از روش‌های برنامه‌نویسی پویا به صورت مستقیم استفاده کنیم. در واقع نیاز به راه‌حل‌های مستقل از مدل داریم که به کمک نمونه‌برداری<sup>۲۰</sup> و کاوش<sup>۲۱</sup> محیط، سیاست بهینه را یاد بگیرند.

## ۲-۳-۲ یادگیری به کمک نمونه‌برداری مونته کارلو

در روش‌های یادگیری به کمک نمونه‌برداری مونته کارلو<sup>۲۲</sup>، به جای استفاده از مدل، از نمونه‌برداری برای تخمین ارزش استفاده می‌شود [۱۲]. کافیست ابتدا یک قسمت را به طور کامل اجرا کنیم، و سپس ارزش هر حالت را، به سمت خروجی قسمت، به‌روزرسانی کنیم:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \times (G_t - Q(s_t, a_t)) \quad (5-2)$$

که در این فرمول،  $\alpha$  نرخ یادگیری<sup>۲۳</sup> است.

لازم به ذکر است که در حین اجرای قسمت، از سیاست اپسیلون-حریصانه<sup>۲۴</sup> استفاده می‌شود. در این سیاست، با احتمال  $\epsilon$  عملی تصادفی انجام می‌شود و با احتمال  $1 - \epsilon$  عملی که بیشترین ارزش را دارد انجام می‌شود. دلیل استفاده از این سیاست، نیاز به کاوش محیط و جلوگیری از گیر کردن در حالت‌های محلی است.

از معایب این روش، می‌توان به نیاز به اجرای کامل قسمت‌ها و نیاز به زمان برای یادگیری اشاره کرد. به همین دلیل، این روش برای مسائلی که قسمت‌های طولانی دارند، مناسب نیست. از مشکلات دیگر

<sup>19</sup>Policy Iteration

<sup>20</sup>Sampling

<sup>21</sup>Exploration

<sup>22</sup>Monte Carlo Sampling

<sup>23</sup>Learning Rate

<sup>24</sup> $\epsilon$ -greedy

این روش، عدم استفاده از ویژگی مارکوف محیط است.

### ۳-۳-۲ یادگیری به کمک تفاوت زمانی

همان طور که گفته شد، استفاده از یادگیری مونته کارلو باعث می شود که عامل در حین انجام قسمت، از تجربه ی قبلی خود استفاده نکند و به روزرسانی ارزش ها فقط پس از اتمام قسمت انجام شود. به این روش، یادگیری برون خط<sup>۲۵</sup> گفته می شود. در روش یادگیری به کمک تفاوت زمانی<sup>۲۶</sup> [۱۳]، عامل در حین انجام قسمت، از تجربه ی خود استفاده می کند و ارزش ها را به صورت برخط<sup>۲۷</sup> به روزرسانی می کند. در واقع به کمک معادله بلمن، مقادیر کیو به سمت مقدار کیو بعدی به روزرسانی می شوند. پیاده سازی این دو الگوریتم، به دو روش دید رو به جلو و دید رو به عقب انجام می شود. در حالت دید رو به جلو، مشابه با یادگیری مونته کارلو، پس از رسیدن به پایان قسمت، ارزش ها به روزرسانی می شوند. در حالت دید رو به عقب، ارزش ها به صورت برخط به روزرسانی می شوند. به این صورت که پس از دریافت یک پاداش، عامل مقادیر کیو  $n$  حالت قبلی خود را به روزرسانی می کند.

#### تی دی صفر

در ساده ترین حالت، عامل در حین انجام عمل، با دیدن یک گام در آینده یا گذشته، ارزش عمل فعلی را به روزرسانی می کند. به این روش تی دی صفر<sup>۲۸</sup> گفته می شود.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \times (R_{t+1} + \gamma \times Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \quad (۶-۲)$$

در واقع، به کمک رابطه بلمن (که فرم ارزش عمل آن در رابطه ۲-۴ دیده می شود) میزان صحیح بودن ارزش عمل فعلی، با ارزش عمل بعدی و پاداش فعلی مقایسه می شود و ارزش عمل فعلی به روزرسانی می شود.

<sup>25</sup>Offline Learning

<sup>26</sup>Temporal Difference Learning

<sup>27</sup>Online

<sup>28</sup>TD(0)

## تی‌دی لامبدا رو به جلو

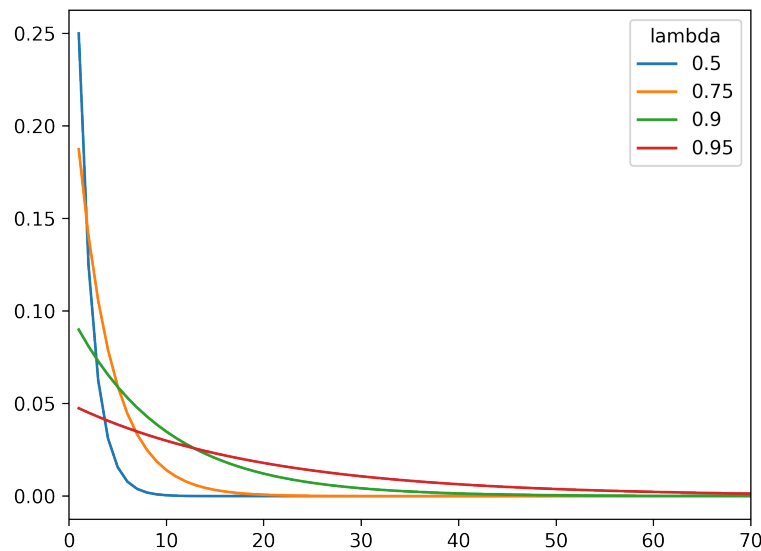
در روش تی‌دی  $\lambda$ <sup>۲۹</sup>، با دید رو به جلو به جای استفاده از یک گام در آینده، از یک ترکیب خطی از پاداش‌ها و بازگشت چندین گام استفاده می‌شود. به این ترکیب خطی،  $\lambda$ -بازگشت<sup>۳۰</sup> گفته می‌شود.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha \times (G_t^\lambda - Q(s_t, a_t)) \quad (۷-۲)$$

که در این رابطه،  $G_t^\lambda$  به صورت زیر محاسبه می‌شود:

$$G_t^\lambda = (1 - \lambda) \times \sum_{n=1}^{\infty} \lambda^{n-1} \times G_t^{(n)} \quad (۸-۲)$$

که در آن  $G_t^{(n)}$ ، مقدار بازگشتی بعد از  $n$  گام است، و  $\lambda$  یک پارامتر بین صفر و یک است که نشان‌دهنده اهمیت پاداش‌های آینده نسبت به پاداش‌های فعلی است. می‌توان میزان اهمیت پاداش‌های را به ازای مقادیر مختلف  $\lambda$  مشاهده کرد.



شکل ۲-۲: تاثیر پارامتر  $\lambda$  بر اهمیت پاداش‌های آینده. محور افقی نماینده تعداد گام‌ها، و محور عمودی نماینده وزن این بازگشت متناظر با آن است.

<sup>۲۹</sup>TD( $\lambda$ )

<sup>۳۰</sup> $\lambda$ -return

## تی‌دی لامبدا رو به عقب

در روش تی‌دی  $\lambda$  با دید رو به عقب، از مفهومی به نام آثار شایستگی<sup>۳۱</sup> استفاده می‌شود. این مفهوم، نشان‌دهنده‌ی این است که هر عملی که در گذشته انجام شده و موجب تغییر ارزش عملی شده است، به چه میزان روی این تغییر سهمیم بوده‌است [۱۴].

ابتدا به ازای هر حالت و عمل، یک متغیر آثار شایستگی  $E(s, a)$  صفر تعیین می‌شود و سپس در هر گام، این متغیر به صورت زیر به‌روزرسانی می‌شود:

$$E(s, a) = \gamma \times \lambda \times E(s, a) + 1(s = s_t, a = a_t) \quad (9-2)$$

در واقع کل آثار شایستگی در لامبدا ضرب شده، و یکی به آثار شایستگی عمل و حالت فعلی اضافه می‌شود. در این حالت، آثار شایستگی معادل با میزان نزدیکی زمانی هر حالت و عمل، به حالت و عمل فعلی است.

سپس ارزش عمل به صورت زیر به‌روزرسانی می‌شود:

$$Q(s, a) = Q(s, a) + \alpha \times \delta \times E(s, a) \quad (10-2)$$

که در این رابطه،  $\delta$  خطای تخمین<sup>۳۲</sup> است که به صورت زیر محاسبه می‌شود:

$$\delta = R_{t+1} + \gamma \times (Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \quad (11-2)$$

می‌توان نشان داد که پس از انجام یک قسمت، به‌روزرسانی‌های انجام شده توسط این روش، معادل با به‌روزرسانی‌های انجام شده توسط تی‌دی  $\lambda$  با دید رو به جلو است.

با ترکیب این روش و سیاست اپسیلون-حریصانه، می‌توان به یک روش یادگیری تقویتی کارا دست یافت که سارسا-لامبدا<sup>۳۳</sup> نام دارد.

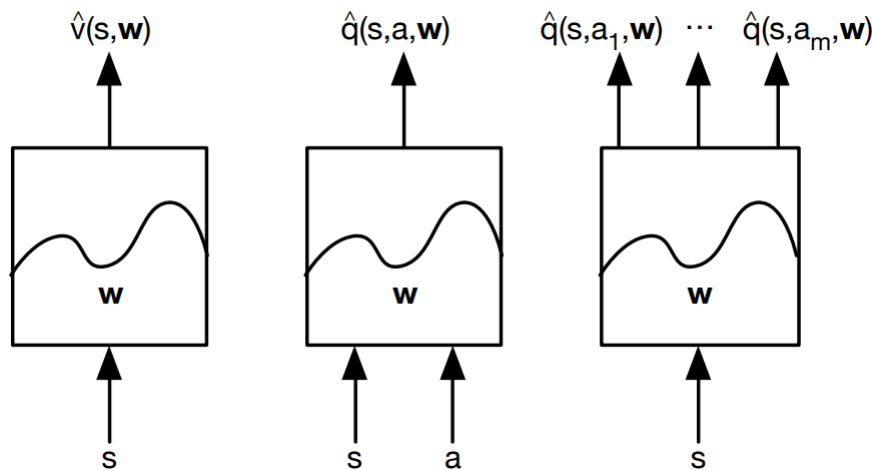
<sup>31</sup>Eligibility Traces

<sup>32</sup>Estimate Error

<sup>33</sup>SARSA( $\lambda$ )

## ۴-۲ استفاده از تقریب گر تابع

با توجه به زیاد بودن تعداد حالت‌های ممکن در اکثر مسائل، یادگیری با روش‌های توضیح‌داده‌شده بسیار کند و حتی غیرمعقول می‌باشد [۱۵]. از این رو، به جای نگهداری مقادیر  $Q$  در یک جدول به ازای هر زوج حالت-عمل از تقریب‌گرهای تابع<sup>۳۴</sup> برای تخمین این مقادیر استفاده می‌کنیم. از تقریب‌گرهای تابع مختلفی می‌توان برای این کار استفاده کرد. از جمله‌ی این تقریب‌گرها می‌توان به شبکه‌های عصبی، درخت‌های تصمیم، و توابع پایه‌ای مثل چندجمله‌ای‌ها اشاره کرد. در واقع ابتدا از حالت، برخی ویژگی‌های مهم را استخراج کرده، سپس از این ویژگی‌ها برای تخمین ارزش‌ها به کمک پارامترهایی که قرار است یاد گرفته شوند استفاده می‌کنیم. رایج است که در روابط، به این پارامترها با  $\theta$  یا  $w$  اشاره شود.



شکل ۲-۳: روش‌های مختلف استفاده از تقریب گر تابع

حال کافیست که معادلات ۲-۶، ۲-۷، و ۲-۱۰ را به صورت تقریبی برای تقریب گر تابع بنویسیم و به جای به‌روزرسانی مقادیر  $Q$ ، پارامترهای تقریب گر تابع را به‌روز کنیم. با توجه به نیاز به به‌روزرسانی، لازم است از تقریب‌گرهای مشتق‌پذیر استفاده کنیم. همانطور که در عکس ۲-۳ می‌توان دید، در حالتی که از تقریب‌گرهای تابع برای پیش‌بینی مقادیر  $Q$  استفاده می‌شود، رایج است از عبارت  $Q(s, a; \theta)$  یا  $Q(s, a; w)$  برای نشان دادن این تقریب‌گرها استفاده کرد.

<sup>34</sup>Function Approximators

## ۵-۲ الگوریتم شبکه کیو عمیق

الگوریتم شبکه کیو عمیق<sup>۳۵</sup> یکی از مستقیم‌ترین راه‌های استفاده از تقریب‌گرهای توابع برای ترکیب یادگیری عمیق و تقویتی است. در این الگوریتم، هدف این است که مشابه با حالت سوم شکل ۲-۳، پارامترهای یک شبکه عصبی را به‌گونه‌ای تنظیم کنیم که مقادیر ارزش عمل را بتواند تخمین بزند. از دستاوردهای این الگوریتم، عملکرد در حد انسان در چندین بازی آتاری<sup>۳۶</sup> است که توسط تیم دیپ‌ماینند<sup>۳۷</sup> در سال ۲۰۱۵ به‌وقوع پیوست [۱۶].

شبکه‌های عصبی الهام گرفته از ساختار و عملکرد مغز انسان هستند که برای یادگیری از داده‌ها و تصمیم‌گیری‌های پیچیده استفاده می‌شوند. این شبکه‌ها از واحدهای پردازشی به نام پرسپترون‌ها<sup>۳۸</sup> تشکیل شده‌اند که در لایه‌های مختلف قرار گرفته‌اند و از طریق وزن‌هایی به هم متصل می‌شوند. یادگیری در شبکه‌های عصبی اغلب از طریق فرایندی به نام نزول گرادیان<sup>۳۹</sup> انجام می‌گیرد که در آن وزن‌های شبکه به صورت مکرر تنظیم می‌شوند تا خطا بین پیش‌بینی‌های شبکه و داده‌های واقعی به حداقل برسد. این فرایند شامل محاسبه گرادیان یا شیب تابع خطا نسبت به وزن‌ها و به‌روزرسانی وزن‌ها در جهت مخالف گرادیان برای کاهش خطا است.

$$\theta_{t+1} = \theta_t - \alpha \nabla J(\theta_t) \quad (۱۲-۲)$$

در این فرمول،  $\theta$  پارامترهای شبکه،  $\alpha$  نرخ یادگیری،  $J$  تابع خطا، و  $\nabla J$  گرادیان تابع خطا نسبت به پارامترها هستند.

برای اینکه یادگیری شبکه عصبی با ثبات بالاتری رخ دهد، این الگوریتم از دو تکنیک بازیابی تجربه<sup>۴۰</sup> [۱۷] و استفاده از شبکه هدف<sup>۴۱</sup> بهره می‌برد.

<sup>۳۵</sup>Deep Q Network (DQN)

<sup>۳۶</sup>Atari

<sup>۳۷</sup>DeepMind

<sup>۳۸</sup>Perceptrons

<sup>۳۹</sup>Gradient Descent

<sup>۴۰</sup>Experience Replay

<sup>۴۱</sup>Target Network

## ۲-۵-۱ بازیابی تجربه

برای رفع مشکلات داده‌های همبسته و توزیع‌های غیر ایستا در یادگیری برخط، DQN از یک مکانیزم بازیابی تجربه استفاده می‌کند. این شامل ذخیره تجربیات عامل در هر گام زمانی در یک بافر بازیابی و سپس نمونه‌برداری تصادفی ریزدسته‌ها<sup>۴۲</sup> از این بافر برای آموزش شبکه است. این رویکرد به شکستن همبستگی بین نمونه‌های پیایی کمک می‌کند و فرآیند یادگیری را پایدار می‌سازد.

## ۲-۵-۲ استفاده از شبکه هدف

DQN یک شبکه دوم به نام شبکه هدف را برای استقرار بیشتر آموزش معرفی می‌کند. شبکه هدف یک کپی از شبکه Q است؛ اما وزن‌های آن با بسامد کمتری به‌روز می‌شوند. این جداسازی نوسان ارزش‌های هدف در به‌روزرسانی یادگیری Q را کاهش می‌دهد و خطر حلقه‌های بازخورد خودتقویتی<sup>۴۳</sup> را کاهش می‌دهد.

## ۲-۵-۳ فرآیند یادگیری

در ابتدا وزن‌های شبکه ارزش عمل را به صورت تصادفی تنظیم می‌کنیم و آن را  $\theta$  می‌نامیم. شبکه هدف را با وزن‌های یکسان مقداردهی می‌کنیم و آن را  $\theta^-$  می‌نامیم. و بافر بازیابی را به طول  $N$  که یک ابرپارامتر<sup>۴۴</sup> است، مقداردهی می‌کنیم.

سپس در هر حالت، پاداش‌های خروجی شبکه اصلی را می‌گیریم و به صورت اپسیلون-حریصانه عمل را انتخاب می‌کنیم؛ یعنی به احتمال  $\epsilon$  رفتار تصادفی داریم، و با احتمال  $1 - \epsilon$  رفتار با بالاترین ارزش

عمل پیش‌بینی شده را برمی‌داریم، به عبارتی:  $a_t = \operatorname{argmax}_a Q(s_t, a; \theta)$

ترکیب چهارتایی حالت قبل عمل، عمل انتخاب شده، پاداش دریافتی، و حالت بعد عمل  $(s_t, a_t, r_t, s_{t+1})$  را به بافر اضافه می‌کنیم. یک گروه با اندازه مشخص را با نمونه‌برداری تصادفی از بافر انتخاب می‌کنیم، و به صورت زیر یادگیری را انجام می‌دهیم:

$$y_j = \begin{cases} r_j & \text{اگر } s_{j+1} \text{ حالت پایانی قسمت باشد} \\ r_j + \gamma \times \max_{a'} Q(s_{j+1}, a'; \theta^-) & \text{در غیر این صورت} \end{cases} \quad (۱۳-۲)$$

<sup>۴۲</sup>Mini-batches

<sup>۴۳</sup>Self-Reinforced Feedback Loop

<sup>۴۴</sup>Hyperparameter



در این معادله  $y_j$  تخمینی از میزان پاداش دریافتی واقعی است، که مشابه با روش‌های تفاوت زمانی، از ترکیب پاداش دریافتی و ارزش عمل بعدی به دست می‌آید. بنابراین می‌توان از این مقدار، مشابه با برچسب در یادگیری نظارت‌شده، برای آموزش شبکه استفاده کرد. میزان خطای شبکه به صورت زیر تعریف می‌شود:

$$L(\theta) = \mathbb{E}[(y_j - Q(s_j, a_j; \theta))^2] \quad (14-2)$$

که با استفاده از این خطا، می‌توان گرادیان تابع خطا نسبت به وزن‌های شبکه را به دست آورد و با استفاده از این گرادیان، می‌توان وزن‌های شبکه را به‌روز کرد:

$$\theta_{t+1} = \theta_t - \alpha \nabla_{\theta} L(\theta) \quad (15-2)$$

در نهایت کافیست به صورت تناوبی و هر چند گام یک بار وزن‌های شبکه هدف را برابر با کپی وزن‌های شبکه اصلی قرار دهیم. با تکرار این مراحل، به شبکه عصبی‌ای دست می‌یابیم که قدرت پیش‌بینی ارزش‌های عمل را دارد.

حال می‌توان اهمیت دو تکنیک گفته شده را بهتر فهمید: در صورت عدم استفاده از شبکه هدف، باید همزمان از شبکه اصلی برای پیش‌بینی آینده (در فرمول ۲-۱۳) و انتخاب عمل استفاده کنیم، که می‌تواند به مشکلاتی مانند نوسانات در آموزش و حلقه‌های بازخورد خودتقویتی منجر شود. همچنین در صورت عدم استفاده از بافر تجربه، باید صرفاً از تجارب اخیر خود استفاده کنیم، که در این صورت همبستگی بین نمونه‌ها و توزیع‌های غیر ایستا می‌تواند مشکل‌ساز شود.

در بسیاری از پیاده‌سازی‌ها، مقدار اپسیلون که به آن نرخ کاوش<sup>۴۵</sup> هم گفته می‌شود، متغیر است و ابتدا زیاد بوده و در طول آموزش به تدریج کاهش می‌یابد و به مقدار ثابتی می‌رسد.

## ۴-۵-۲ اثرات برخی از ابرپارامترها

در این بخش به بررسی تاثیر تغییر «اندازه ریزدسته» و «اندازه بافر تجربه» بر عملکرد الگوریتم DQN می‌پردازیم.

<sup>۴۵</sup>Exploration Rate

## اندازه ریزدسته

اندازه ریزدسته تعداد نمونه‌هایی است که از بافر تجربه برای آموزش شبکه استفاده می‌شود. با افزایش این مقدار، تنوع نمونه‌ها افزایش می‌یابد و از طرف دیگر، باعث افزایش پیچیدگی محاسباتی می‌شود. بنابراین باید یک تعادل بین زمان لازم برای هر به‌روزرسانی شبکه، و کیفیت آموزش شبکه داشته باشیم.

## اندازه بافر تجربه

اندازه بافر تجربه تعداد نمونه‌هایی است که برای آموزش شبکه در طول یادگیری ذخیره می‌شود. با افزایش این تعداد، تنوع این تجربه‌ها افزایش می‌یابد و از طرف دیگر، باعث افزایش حافظه مصرفی می‌شود. در صورت زیادی کوچک بودن این پارامتر، فقط تجارب اخیر در بافر ذخیره می‌شوند و از تجربیات گذشته استفاده نمی‌شود. بنابراین عامل ممکن است فراموشی تجربیات گذشته را تجربه کند و یادگیری برخط آن را مختل کند [۱۸]. از سوی دیگر، در صورت کوچک بودن تجربیات، احتمال وجود تجارب اخیر درون ریزدسته کاهش می‌یابد که باعث کند شدن فرآیند آموزش و عدم دریافت واکنش نسبت به امتحان کردن حالت‌های جدید می‌شود [۱۹]. بنابراین باید اندازه بافر نسبت به اندازه ریزدسته و همچنین طول کل آموزش به درستی انتخاب شود.

## ۶-۲ الگوریتم بهبود گرادیان سیاست معین عمیق

الگوریتم بهبود گرادیان سیاست معین عمیق<sup>۴۶</sup> یا به اختصار DDPG، یک الگوریتم یادگیری تقویتی است که برای حل مسائل پیوسته و فضای عمل پیوسته طراحی شده است [۲۰]. همانطور که در بخش‌های قبل مشاهده کردیم، برای انتخاب عمل معمولاً از سیاست اپسیلون-حریصانه استفاده می‌شود؛ اما این روش برای فضای عمل پیوسته مناسب نیست، چرا که به‌دست‌آوردن بیشینه در فضای پیوسته به مراتب دشوارتر از حالت گسسته است. این الگوریتم از نوع یادگیری خارج از سیاست است؛ به این معنا که عامل با دنبال کردن سیاستی که سیاست انتخابی خود نیست به بهبود عملکرد خود می‌پردازد.

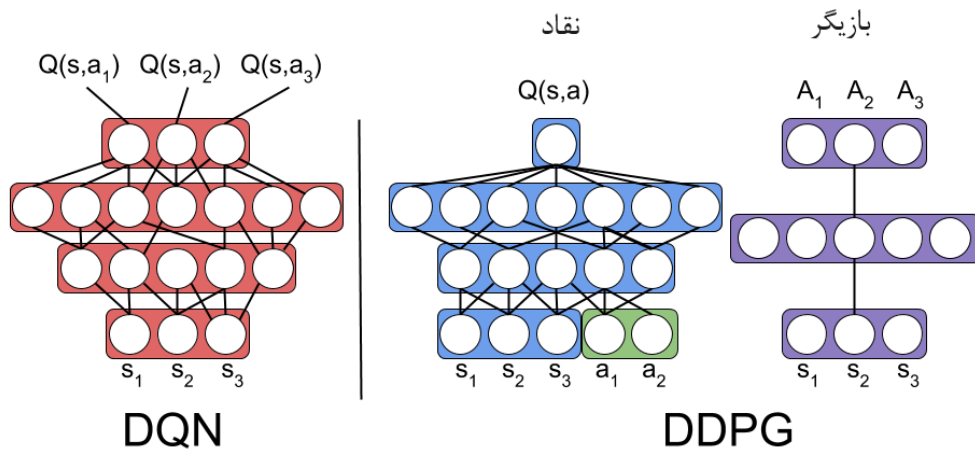
DDPG از دو شبکه عصبی استفاده می‌کند [۲۱]: یک شبکه برای تخمین ارزش عمل که به آن نقاد<sup>۴۷</sup> می‌گوییم، و یک شبکه برای تخمین سیاست و انتخاب عمل که به آن بازیگر<sup>۴۸</sup> می‌گوییم. شبکه نقاد مشابه با شبکه Q در DQN است؛ با این تفاوت که حالت و عمل را ورودی گرفته و ارزش عمل را

<sup>۴۶</sup>Deep Deterministic Policy Gradient (DDPG)

<sup>۴۷</sup>Critic

<sup>۴۸</sup>Actor

خروجی می‌دهد. شبکه سیاست یک شبکه عصبی است که خروجی آن عمل است. این الگوریتم مشابه با DQN از بافر تجربه برای ذخیره تجربیات عامل، و از شبکه‌های هدف برای هر دو شبکه نقاد و بازیگر استفاده می‌کند. البته در این الگوریتم، به جای کپی وزن‌های شبکه به شبکه هدف، از میانگین وزن‌دار استفاده می‌شود تا به‌روزرسانی آرام‌تر انجام شود و ثبات بیشتری داشته باشد.



شکل ۲-۴: شبکه‌های عصبی بازیگر و نقاد در الگوریتم DDPG و مقایسه با DQN

رایج است که شبکه ارزش عمل را با  $Q(s, a; \phi)$  و شبکه سیاست را با  $\mu(s; \theta)$  نشان دهیم. فرآیند یادگیری این الگوریتم به صورت زیر است:

۱. ابتدا وزن‌های شبکه‌های نقاد و بازیگر را به صورت تصادفی مقداردهی می‌کنیم و آن‌ها را به ترتیب  $\phi$  و  $\theta$  می‌نامیم.
۲. بافر تجربه را به طول  $N$  مقداردهی می‌کنیم.
۳. در هر گام زمانی، حالت را به شبکه بازیگر می‌دهیم و عملی که این شبکه پیشنهاد می‌دهد را انجام می‌دهیم.
۴. پاداش دریافتی و حالت بعد عمل را به بافر تجربه اضافه می‌کنیم.
۵. یک ریزدسته از بافر تجربه را انتخاب کرده و با استفاده از فرمول زیر، مقدار واقعی ارزش حالت را (طبق الگوریتم تفاوت زمانی) محاسبه می‌کنیم:

$$y_j = \begin{cases} r_j & \text{اگر } s_{j+1} \text{ حالت پایانی قسمت باشد} \\ r_j + \gamma \times Q(s_{j+1}, \mu(s_{j+1}; \theta^-); \phi^-) & \text{در غیر این صورت} \end{cases} \quad (۱۶-۲)$$

۶. خطای شبکه نقاد را به صورت زیر محاسبه می‌کنیم:

$$L(\phi) = \mathbb{E}[(y_j - Q(s_j, a_j; \phi))^2] \quad (۱۷-۲)$$

و وزن‌های شبکه نقاد را به‌روز می‌کنیم:

$$\phi_{t+1} = \phi_t - \alpha \nabla_{\phi} L(\phi) \quad (۱۸-۲)$$

۷. خطای شبکه بازیگر را به صورت زیر محاسبه می‌کنیم:

$$L(\theta) = -\mathbb{E}[Q(s, \mu(s; \theta); \phi)] \quad (۱۹-۲)$$

و وزن‌های شبکه بازیگر را به‌روز می‌کنیم:

$$\theta_{t+1} = \theta_t + \alpha \nabla_{\theta} L(\theta) \quad (۲۰-۲)$$

۸. در نهایت هر چند گام یک بار، وزن‌های شبکه‌های هدف را به‌روز می‌کنیم:

$$\theta_{t+1}^- = \tau \theta_t + (1 - \tau) \theta_t^- \quad (۲۱-۲)$$

$$\phi_{t+1}^- = \tau \phi_t + (1 - \tau) \phi_t^- \quad (۲۲-۲)$$

همانطور که مشاهده‌شد، این الگوریتم تعداد زیادی ابرپارامتر دارد که باید به درستی تنظیم شوند تا به عملکرد بهینه برسد. از جمله این پارامترها می‌توان به نرخ یادگیری ( $\alpha$ ) که می‌تواند برای شبکه‌های نقاد و بازیگر متفاوت باشد، نرخ تخفیف ( $\gamma$ )، فرکانس به‌روزرسانی شبکه‌ها (هر چند گام یک بار ریزدسته انتخاب می‌کنیم و مراحل ۵ تا ۸ را طی می‌کنیم)، فرکانس به‌روزرسانی وزن‌های هدف (هر چند گام یک بار وزن‌های شبکه‌های هدف را به‌روزرسانی می‌کنیم)، اندازه ریزدسته، طول بافر تجربه، و ضریب

میانگین گیری برای به روزرسانی شبکه های هدف ( $\tau$ ) اشاره کرد.

## ۷-۲ جمع بندی

در این فصل، ابتدا به معنا و اهمیت یادگیری تقویتی پرداختیم و سپس به معرفی مفاهیم اصلی این حوزه پرداختیم. سپس به معرفی الگوریتم های یادگیری تقویتی کلاسیک بر مبنای مدل همچون برنامه نویسی پویا و روش های مبتنی بر ارزش و سیاست پرداختیم. در ادامه به معرفی الگوریتم های یادگیری تقویتی بدون مدل پرداختیم که از تقریب گرهای تابع برای تخمین ارزش ها استفاده می کنند. در این بخش، به معرفی دو الگوریتم معروف یادگیری تقویتی بدون مدل، یعنی DQN و DDPG پرداختیم. در ادامه، در فصل بعد به معرفی یک مسئله و کاربرد این الگوریتم ها در حل آن خواهیم پرداخت.

# فصل سوم

## محیط فوتبال ريوکاپ

در این بخش به معرفی مفاهیم پایه محیط فوتبال روبوکاپ و توصیف چگونگی عملکرد آن می‌پردازیم. این مفاهیم شامل قوانین بازی، رفتارهای ممکن، کد پایه ایجنت، حالت پناستی، و روش اجرای بازی می‌باشند.

## ۳-۱ معرفی لیگ

### ۳-۱-۱ اهداف لیگ

لیگ روبوکاپ مجموعه مسابقاتی سالانه است که قصد دارد با کمک فوتبال، به پیشرفت زمینه‌های رباتیک و هوش مصنوعی کمک کند. علت انتخاب فوتبال به عنوان محیط مسابقه، این است که فوتبال یکی از محیط‌هایی است که می‌تواند مسائل مختلفی از جمله تصمیم‌گیری، هماهنگی، بینایی ربات، و ارتباط را در بر داشته باشد. یکی از اهداف بلندمدت لیگ، ساختن ربات‌هایی است که بتوانند تا سال ۲۰۵۰، تیمی از انسان‌ها را به چالش بکشند.

یکی از لیگ‌های این مسابقات، لیگ شبیه‌ساز فوتبال دو بعدی است. همانطور که از اسم لیگ پیداست، مسابقه حالت دو بعدی دارد، به این منظور که فضا حالت دید از بالا دارد، و همه حرکات بازیکنان و توپ روی سطح زمین انجام می‌شود. تمرکز اصلی این لیگ، تصمیم‌گیری و استراتژی، و ساختن الگوریتم‌های مناسب با محیط‌های چندعامله با دید ناقص است.

### ۳-۱-۲ ویژگی‌های محیط و قوانین بازی

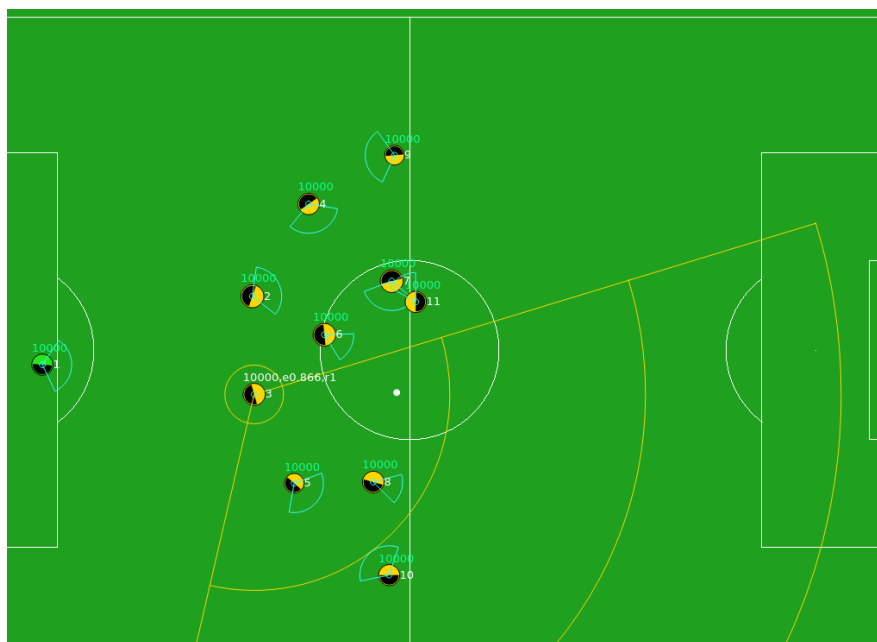
مشابه با فوتبال واقعی، هر بازی متشکل از دو تیم ۱۱ نفره است که هر کدام از این نفرات یک عامل مستقل می‌باشد. بازی در یک مستطیل به ابعاد ۱۱۵ در ۶۸ انجام می‌شود، و هر تیم یک دروازه‌بان دارد که در هر طرف زمین قرار دارد. بازی در ۶۰۰۰ گام<sup>۱</sup> انجام می‌شود که به دو نیمه تقسیم شده، و هر گام ۱۰۰ میلی‌ثانیه طول می‌کشد. سرور مسابقات در هر گام، اطلاعات مربوط به وضعیت بازی را به تمامی عامل‌ها می‌فرستد، و عامل‌ها باید تصمیم‌گیری خود را بر اساس این اطلاعات انجام دهند.

بازیکنان دید محدودی دارند. با توجه به زاویه گردنی که تنظیم کرده‌اند، یک قطاع از زمین را مشاهده می‌کنند و فقط اطلاعات مربوط به بازیکنان و اشیاء داخل این قطاع را دریافت می‌کنند در شکل ۳-۱ نمونه دید یک بازیکن نشان داده شده است. اطلاعات بازیکنانی که داخل دید نیستند، باید از حافظه

<sup>1</sup>Cycle

بازیکن و یا از ارتباط با سایر بازیکنان به دست آید. شایان به ذکر است که مشاهدات هر بازیکن، دارای کمی خطای اندازه‌گیری است و به طور کامل دقیق نیست. به طور مشابه، ضربات بازیکن به توپ نیز کمی خطا دارند و کامل دقیق نیستند.

هر عامل باید به صورت مستقل تصمیم‌گیری کند و ارتباط بسیار ناچیزی با سایر عاملین دارد. در صورتی که دو عامل بخواهند ارتباط برقرار کنند، گیرنده باید از قبل توجه خود را به فرستنده تنظیم کرده‌باشد، و فرستنده حداکثر ۸ بایت می‌تواند ارسال کند.



شکل ۳-۱: نمونه دید یک بازیکن. در این مثال، بازیکن ۳، بازیکن ۵ و ۸ و ۱۰ و توپ را می‌بیند و از موقعیت سایر بازیکنان اطلاعی ندارد.

هر عامل یک رزرو انرژی و یک منبع انرژی دارد. در صورت پر نبودن انرژی بازیکن، انرژی به نرخ ثابتی از رزرو به منبع منتقل می‌شود. در صورت اتمام انرژی منبع، رفتارهای بازیکن از جمله حرکت کردن کندتر می‌شوند. رفتارهای بیشتری (مانند تکل زدن، خطا و ضربه آزاد، کارت زرد و قرمز و ...) نیز در محیط به کمک مدل‌های ریاضیاتی پیاده‌سازی شده‌است که خارج از دامنه این پروژه می‌باشد.

## ۲-۳ کد پایه ایجنت

در سال‌های اول مسابقات، هر تیمی کد عامل خود را از صفر می‌نوشت، که میزان در دسترس بودن لیگ را بسیار پایین آورده‌بود. با توجه به یکسان بودن بخش عمده‌ای از نیازمندی‌های تیم‌ها، همچون نیاز به



اتصال به سرور، نیاز به تفکیک وظایف بازیکنان به دروازه‌بان و مدافع و مهاجم، نیاز به موقعیت‌یابی اشیا در زمین، توابع هندسی، و ...، هیده‌یسا آکیام از تیم هلیوس در سال ۲۰۱۲ تصمیم به ساختن یک کد پایه به صورت متن‌باز محض استفاده سایر تیم‌ها گرفت. این کد پایه، که کد پایه ایجنت نام دارد، زیربنای ۱۳ تیم از ۱۵ شرکت‌کننده سال اخیر ربوکاپ بوده، و نقطه شروع اکثر کسانی است که قصد فعالیت در این فضا را دارند. این کد پایه همچنان در حال به‌روزرسانی و تقویت‌شدن است<sup>۲</sup>، و خود منشا سایر کد پایه‌های به اشتراک گذاشته‌شده همچون کد پایه گلایدرز<sup>۳</sup> و کد پایه سایرس<sup>۴</sup> است.

با استفاده از این کد پایه، توسعه‌دهندگان می‌توانند سطح کدزدن خود را از سطوح پایین، به سطح استراتژی و تاکتیک و تصمیم‌گیری منتقل کنند. به طور مثال بدون استفاده از یک کد پایه، برای حرکت عامل به مرکز زمین باید کد اتصال به سرور و موقعیت‌یابی را پیاده‌سازی کنیم. سپس درگیر مسائلی همچون محاسبه شتاب بازیکن، نیرویی که باید اعمال شود، مسیر حرکت بهینه (چرخیدن و دویدن یا دویدن مورب) و ... شوند. در حالی که این عمل، به صورت یک دستور سطح بالا در کد پایه ایجنت قابل اجراست. در بخش بعدی به تمام رفتارهای ممکن در این کد پایه و سطح‌بندی رفتارها می‌پردازیم.

### ۳-۳ معرفی رفتارهای ممکن

هر عامل در هر لحظه، باید تصمیم‌گیری خود را به سرور بفرستد. این تصمیم‌گیری می‌تواند شامل انجام یکی از رفتارهای ممکن باشد. در هر لحظه، عامل می‌تواند گردن خود را بچرخاند تا اطراف خود را ببیند، و همزمان یکی از پنج رفتار ضربه به توپ، حرکت بدن، چرخش بدن، تکل، یا گرفتن توپ را انجام دهد. رایج است که رفتارهای ممکن و پیاده‌سازی شده را به سه طبقه تقسیم‌بندی کنیم که در پایین‌ترین سطح، رفتارهای سطح سرور، و در بالاترین سطح، رفتارهای سطح استراتژیک و فکری قرار می‌گیرند.

#### ۱-۳-۳ رفتارهای سطح پایین

در پایین‌ترین سطح، رفتارها مستقیماً معادل با رفتارهای مورد پذیرش سرور می‌باشند. این رفتارها شامل اعمال نیرو روی توپ در یک راستا (نسبت به بدن بازیکن) و نیروی خاص، چرخاندن بازیکن به یک راستا، اعمال نیروی حرکت بازیکن در راستا و نیرو، تکل زدن در یک راستا، و گرفتن توپ (در صورتی که بازیکن دروازه‌بان باشد) می‌باشند.

<sup>۲</sup><https://github.com/helios-base/helios-base>

<sup>۳</sup>Gliders

<sup>۴</sup>Cyrus

### ۲-۳-۳ رفتارهای سطح متوسط

در این سطح، رفتارها ساده‌سازی شده‌اند تا استفاده از آن‌ها برای تصمیم‌گیری راحت‌تر باشد. به طور مثال رفتار حرکت به سمت یک نقطه خاص از زمین، رفتار ضربه زدن به توپ با سرعت دلخواه در راستای غیرنسبی، رفتار ضربه زدن توپ به صورت چندضرب به یک نقطه خاص، و... را می‌توان از رفتارهای این سطح معرفی کرد.

### ۳-۳-۳ رفتارهای سطح بالا

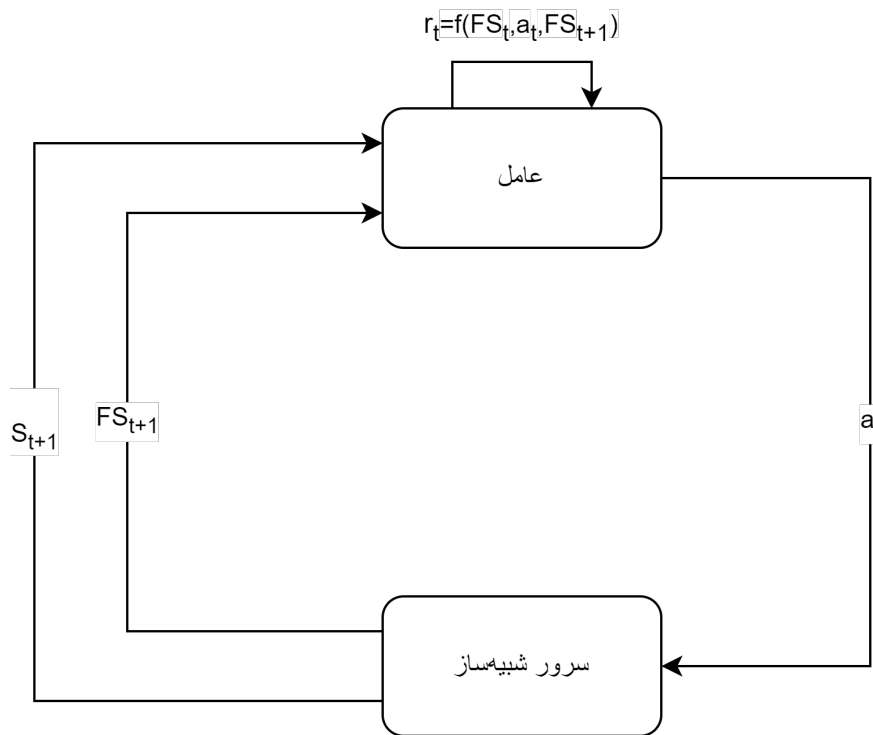
در این سطح، رفتارها حالت استراتژی و فکر کردن دارند، و به صورت انتزاعی و مشابه با فوتبال واقعی می‌باشند. به طور مثال، حرکت به سمت نقطه‌ی آرایش<sup>۵</sup>، حرکت برای قطع توپ، زدن شوت (در صورت ممکن بودن شوت موفق)، و... از رفتارهای سطح بالا می‌باشند.

### ۴-۳ حالت دید کامل

همانطور که گفته شد، محیط اطلاعات کامل را در اختیار عامل قرار نمی‌دهد. با توجه به اینکه قصد تغییر کد سرور مسابقات و افزودن پاداش به سرور را نداریم، این محاسبات درون خود عامل باید صورت بگیرند. خوش‌بختانه سرور مسابقات قابلیت ارسال محیط در حالت دید کامل<sup>۶</sup> را دارد که در این حالت، همه اطلاعات بدون نویز به عامل ارسال می‌شوند. داخل کد ایجنت می‌توان انتخاب کرد که در صورت دریافت حالت دید کامل آن‌را جایگزین دید ناقص کند، یا اینکه هر دو را کنار هم نگه‌دارد. ما در فرآیند یادگیری تقویتی از حالت دوم استفاده می‌کنیم تا با کمک دید کامل محاسبات پاداش را انجام دهیم، و با حالت دید ناقص تصمیم‌گیری کنیم. در عکس ۲-۳ می‌توان مشاهده کرد که پاداش داخل عامل محاسبه شده، و تابعی از حالت دید کامل (Fullstate یا به اختصار FS) است، و محیط هر دو حالت دید را برای عامل ارسال می‌کند. این نمودار در واقع حالت اصلاح شده ۱-۲ است که خاص‌منظوره این پروژه می‌باشد.

<sup>5</sup>Formation

<sup>6</sup>Fullstate WorldModel



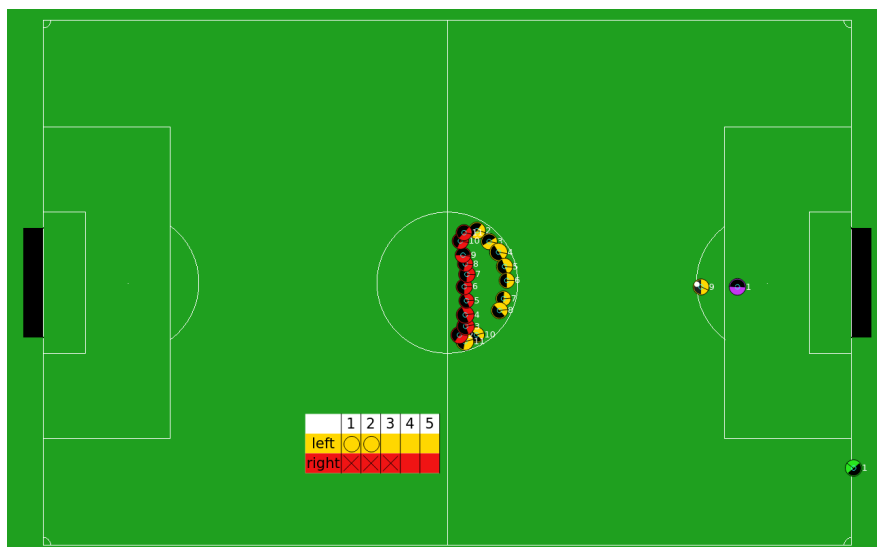
شکل ۳-۲: ارتباط بین محیط و عامل، و روش محاسبه پاداش

### ۵-۳ معرفی حالت پنالتی

در صورت تساوی بازی در ده دقیقه زمان عادی، و تداوم تساوی پس از وقت اضافی، بازی به حالت پنالتی می‌رود. با توجه به دو بعدی بودن محیط، در صورتی که مشابه با فوتبال واقعی، پنالتی به صورت تک ضرب باشد، می‌توان برای آن استراتژی قطعی ارائه داد. به همین منظور، پنالتی در فضای لیگ دوبعدی، به صورت تک به تک با دروازه‌بان است.

بازیکن مهاجم، ۱۵ ثانیه فرصت دارد تا توپ را به گل تبدیل کند. دروازه‌بان باید در این مدت تلاش کند جوری موقعیت‌گیری کند که حریف نتواند شوت منجر به گل داشته‌باشد. در صورتی که زمان مهاجم تمام شود، توپ به بیرون محیط بازی برود، یا توپ توسط دروازه‌بان گرفته‌شود، دروازه‌بان برنده می‌شود. در صورتی که توپ وارد گل شود، مهاجم برنده می‌شود.

با توجه به تک‌عامله بودن، و محدود بودن زمان محیط، می‌توان از روش‌های یادگیری تقویتی برای یادگیری استراتژی‌های بهینه برای پنالتی استفاده کرد. در شکل ۳-۳ یک نمونه از حالت پنالتی را مشاهده می‌کنید.



شکل ۳-۳: نمونه‌ای از بازی در حالت پنالتی

### ۳-۶ کار با مربی تمرینی برای تولید محیط قابل تکرار

با توجه به اینکه برای یادگیری تقویتی، نیاز به بازگرداندن محیط به حالت اولیه را داریم، می‌توانیم از مربی تمرینی<sup>۷</sup> استفاده کنیم. در صورتی که در تنظیمات سرور، این حالت فعال شده باشد، می‌توان کدی نوشت که در شرایط دلخواه، توپ و بازیکنان را جابه‌جا کنیم، حالت بازی را تنظیم کنیم، انرژی بازیکنان را پر کنیم و ... . در فصل‌های آینده از این امکان، برای ساختن یک رابط استاندارد یادگیری تقویتی برای محیط فوتبال استفاده خواهیم کرد.

### ۳-۷ جمع‌بندی

در این فصل، محیط فوتبال ربوکاپ دوبعدی به عنوان یک ابزار مطالعاتی برای تقویت پیشرفت‌ها در زمینه هوش مصنوعی و رباتیک معرفی شد. این محیط، با ارائه یک پلتفرم مسابقه‌ای مبتنی بر قوانین فوتبال واقعی، فرصت‌هایی برای توسعه و آزمایش الگوریتم‌های تصمیم‌گیری، هماهنگی، و استراتژی در محیط‌های چندعامله فراهم می‌کند.

معرفی این محیط شامل توضیح قوانین بازی، نحوه ارتباط و تصمیم‌گیری عامل‌ها، و همچنین ساختار کد پایه ایجنت بود که توسعه‌دهندگان را قادر می‌سازد تا تمرکز خود را بر روی تاکتیک و تصمیم‌گیری‌های

<sup>7</sup>Trainer

سطح بالای مشابه با فوتبال معطوف دارند.

یکی از نوآوری‌های کلیدی در این فصل، استفاده از حالت دید کامل در کنار دید ناقص است که به عامل‌ها امکان می‌دهد تا پاداش‌ها را بر اساس اطلاعات کامل محیط محاسبه کنند، در حالی که تصمیم‌گیری بر اساس دید ناقص انجام می‌شود. این رویکرد یک قدم مهم در راستای افزایش قابلیت‌های یادگیری تقویتی در محیط‌هایی با دید ناقص است.

همچنین، با معرفی حالت پنالتی و استفاده از مربی تمرینی برای تولید محیط‌های قابل تکرار، زمینه‌های لازم برای پیاده‌سازی الگوریتم‌های یادگیری تقویتی فراهم شده است. این امکانات به پژوهشگران اجازه می‌دهد تا در یک محیط مسابقه‌ای استاندارد، استراتژی‌های بهینه‌سازی شده را آزمایش و توسعه دهند.

## فصل چهارم

### پیاده‌سازی و آماده‌سازی محیط استاندارد

## ۱-۴ زیرساخت پایتونی تهاجم نصف زمین

در سال ۲۰۱۵، پروژه تهاجم نصف زمین<sup>۱</sup> یا همان اچ‌اف‌او تلاش کرد محیط پایتونی‌ای برای یادگیری تقویتی در فوتبال دو بعدی ایجاد کند [۴، ۵]. کد سرور به گونه‌ای تغییر یافته بود، که بازیکن بتواند به سرور دستور اعمال رفتارها و رفتن به گام بعدی را بدهد.

این روش متکی بر قابلیت‌های لیب‌سی<sup>۲</sup> بود، که از آن برای ارتباط با کد سی‌پلاس‌پلاس استفاده می‌کرد. لیب‌سی اجازه می‌دهد که اگر کلاس معادل پایتونی و سی‌پلاس‌پلاس وجود داشته باشد، از سمت پایتون می‌توان به آن دسترسی داشت. در این پروژه، عامل پایتونی به عنوان ایجنت دسترسی به محیط دارد، که با آن از کد سی‌پلاس‌پلاس محیط را درخواست می‌کند. محیط در دو سطح بالا و پایین قابل درخواست است؛ حالت سطح پایین ۶۰ ویژگی محیط، و حالت سطح بالا ۹ ویژگی را به صورت یک آرایه صفر تا یک به عامل پس می‌دهد. عامل سپس تصمیم خود را اخذ کرده، و به سرور دستور اجرای آن را می‌دهد. معایب استفاده از این محیط عبارتند از:

- دشواری در تغییر محیط عامل به صورت دلخواه
- تفاوت زیاد بین محیط آماده‌شده در ایجنت، که مورد استفاده اکثر تیم‌هاست، و محیط اچ‌اف‌او
- عقب‌ماندگی محیط به علت تغییر کدهای سرور و به روز نبودن با تغییرات شبیه‌ساز
- دشواری نصب، به علت پیش‌نیازهای قدیمی همچون Qt4
- عدم امکان استفاده از تیم‌های جدید برای حریف یادگیری به علت نسخه قدیمی سرور

## ۲-۴ کد پایه پایرس

با توجه به دشواری استفاده از یادگیری ماشین در سی‌پلاس‌پلاس، ما در تیم سایرس تصمیم به پیاده‌سازی یک کد پایه معادل با کد پایه ایجنت در پایتون گرفتیم، که نام آن پایرس<sup>۳</sup> است [۲۲]. این تلاش‌ها از سال ۲۰۱۹ آغاز شد و در نهایت بعد از سه سال و حدود ۲۵ هزار خط کد، پروژه به حالت پایدار و قابل استفاده رسیده است.<sup>۴</sup>

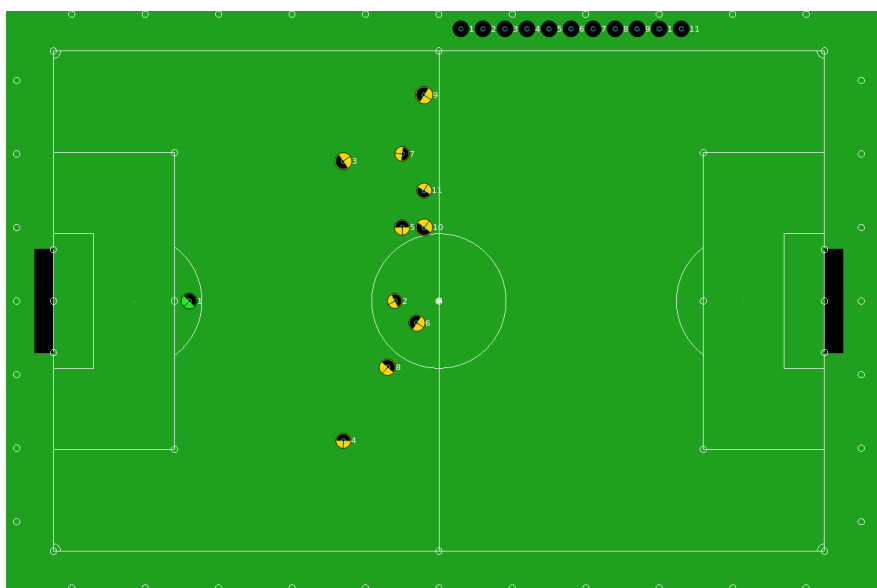
<sup>۱</sup>Half Field Offense (HFO)

<sup>۲</sup>LibC

<sup>۳</sup>Pyrus

<sup>۴</sup><https://github.com/Cyrus2D/Pyrus2D>

امید بر آن بود که با توجه به اتکا بر الگوریتم‌های یادگیری ماشین به‌جای الگوریتم‌های درخت و گراف، بتوان کندی زبان پایتون را جایگزین کرد. در کد ایجنت، قبل از رسیدن به گام تصمیم‌گیری، محاسبات زیادی رخ می‌دهد تا بازیکن موقعیت خود و سایر بازیکنان را بیابد، و سریع‌ترین بازیکنان به توپ، موقعیت آفساید و... را به دست آورد. با توجه به کند بودن پایتون در مقایسه با سی‌پلاس‌پلاس، همین محاسبات مقدماتی نیز بسیار زمان‌بر بودند و بخش بزرگی از ۱۰۰ میلی‌ثانیه موجود برای هر گام بازی را اشغال می‌کردند. به همین منظور، پایرس جایگزین مناسبی برای پایتونی کردن کد نبود.



شکل ۴-۱: پرچم‌های کنار زمین، که برای موقعیت‌یابی استفاده می‌شوند.

## ۳-۴ کد پایه جی‌آرپی‌سی

با توجه به سریع بودن سی‌پلاس‌پلاس برای پیش‌پردازش، و کاربردی و راحت بودن پایتون برای یادگیری تقویتی، تصمیم گرفتیم به کمک پروتکل فراخوانی تابع از راه دور<sup>۵</sup>، این دو زبان را به هم متصل کنیم. در این پروژه از چارچوب فراخوانی تابع از راه دور گوگل، یا همان جی‌آرپی‌سی<sup>۶</sup> استفاده می‌کنیم. در این چارچوب، ابتدا یک فایل پروتو<sup>۷</sup> باید تعریف کرد، که حاوی مشخصات اشیاء و امضای توابع است. سپس با استفاده از کامپایلر پروتو می‌توان کدهای سرور و کلاینت را به زبان‌های دلخواه تبدیل کرد. سپس با اضافه کردن کد تولیدشده توسط کامپایلر پروتو می‌توان درون کلاینت از توابع به‌گونه‌ای

<sup>۵</sup>Remote Procedure Call (RPC)

<sup>۶</sup>gRPC (google Remote Procedure Call)

<sup>۷</sup>Proto



استفاده کرد که گویا داخل خود کد قرار دارند. از مهم‌ترین مزایای این پروتکل ارتباطی، مستقل از زبان بودن آن، و نوشتن بسته‌های پیام به صورت صفر و یکی (در مقابل متنی) است که سربرار زمانی نوشتن و خواندن پیام را ناچیز می‌کند. همچنین به علت استفاده از پروتکل اچ‌تی‌تی‌پی<sup>۸</sup> برای ارسال بسته‌ها روی شبکه، می‌توان از ضمانت‌های دریافت پیام نیز استفاده کرد.

```
message Vector2D {
    float x = 1;
    float y = 2;
    float dist = 3;
    float angle = 4;
}
```

شکل ۴-۲: نحوه تعریف اشیا در جی‌آرپی‌سی

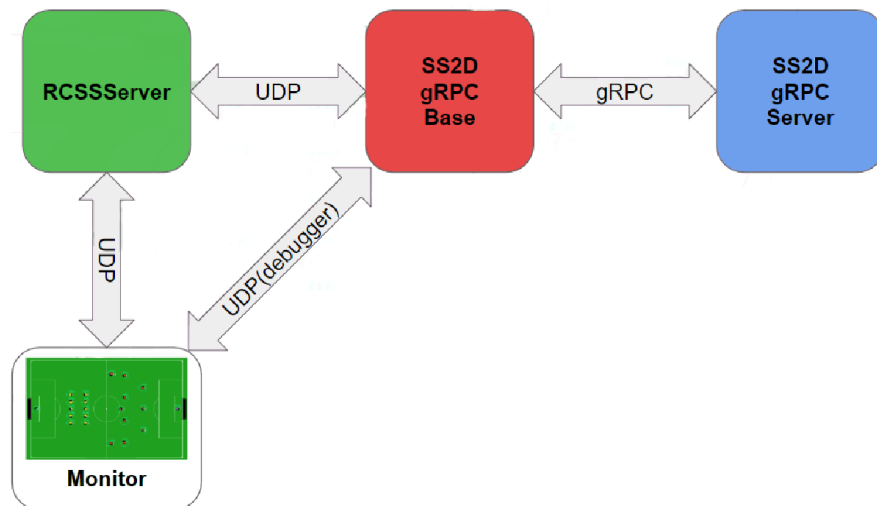
```
service Game {
    rpc GetPlayerActions(State) returns (PlayerActions) {}
    rpc GetCoachActions(State) returns (CoachActions) {}
    rpc GetTrainerActions(State) returns (TrainerActions) {}
    rpc SendInitMessage(InitMessage) returns (Empty) {}
    rpc SendServerParams(ServerParam) returns (Empty) {}
    rpc SendPlayerParams(PlayerParam) returns (Empty) {}
    rpc SendPlayerType(PlayerType) returns (Empty) {}
}
```

شکل ۴-۳: نحوه تعریف توابع در جی‌آرپی‌سی

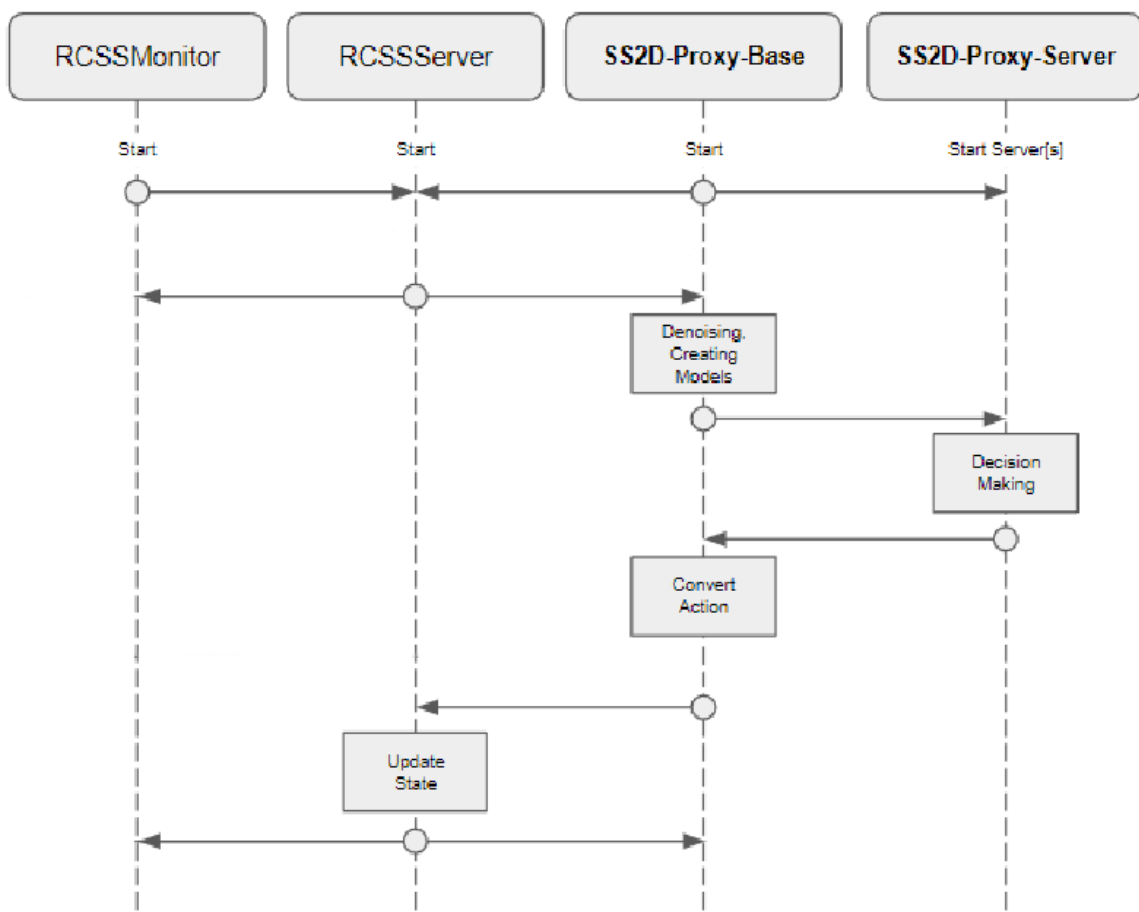
برای تشکیل اتصال بین این دو زبان کد پایه ایجنت را به گونه‌ای تغییر دادیم که پس از اتمام مراحل پیش‌پردازش و آماده شدن مدل دنیای بازیکن<sup>۹</sup> در سی‌پلاس‌پلاس، اطلاعات آن را با یک درخواست جی‌آرپی‌سی به یک سرور تصمیم‌گیری پایتونی بفرستد. در شکل ۴-۴ و ۴-۵ نحوه‌ی اتصال اجزای مسابقه، و روند عمومی تصمیم‌گیری نمایش داده شده‌است.

<sup>۸</sup>HTTP

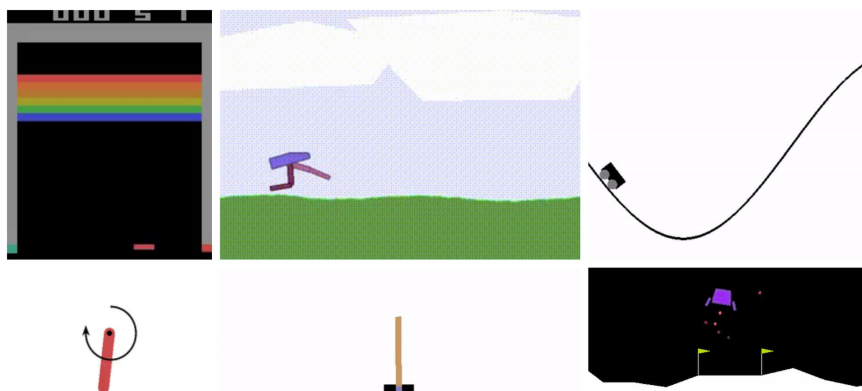
<sup>۹</sup>WorldModel



شکل ۴-۴: پروتکل‌های استفاده‌شده برای ارتباط بین اجزای مسابقه



شکل ۴-۵: نحوه کارکرد و اتصال کد پایه جی‌آرپی‌سی به سرور مسابقات و نمایشگر بازی



شکل ۴-۶: نمونه‌ای از محیط‌های آماده شده در پلتفرم جیم

## ۴-۴ محیط استاندارد جیم

پلتفرم جیم که توسط گروه اپن‌ای‌ای توسعه داده شده است، یک محیط استاندارد برای یادگیری تقویتی است. این پلتفرم شامل محیط‌های گوناگون از پیش آماده شده است که می‌توانند برای یادگیری تقویتی استفاده شود. این محیط همچنین به ما قابلیت تعریف محیط‌های جدید را می‌دهد. در سال ۲۰۲۳، مدیریت و نگهداری این پلتفرم به شرکت فاراما<sup>۱۰</sup> واگذار شد و از آن زمان، این پلتفرم به نام پلتفرم جیمنازیوم<sup>۱۱</sup> شناخته می‌شود. تبدیل محیط یادگیری به این نوع محیط، نه تنها به ما قابلیت استفاده از کتابخانه‌های پیشین و کمک گرفتن از ابزارهای از پیش تعبیه شده را می‌دهد، بلکه امکان به اشتراک گذاری راحت این محیط به سایر محققان را نیز فراهم می‌کند.

### ۴-۴-۱ توصیف رابط و توابع موجود

برای تعریف یک محیط جدید در جیم، باید توابع استاندارد آن را پیاده‌سازی کنیم. به این منظور، لازم است با روش کارکرد هر جزء محیط آشنا شویم. لازم به ذکر است که تفاوت‌های جزئی‌ای بین نسخه اپن‌ای‌ای و فاراما وجود دارد؛ که در اینجا به توصیف نسخه فاراما می‌پردازیم.

#### تابع گام برداشتن

این تابع، یک عمل را به عنوان ورودی می‌گیرد، و وضعیت جدید، پاداش این عمل، و اطلاعاتی همچون اتمام بازی، یا تمام شدن وقت عامل را به عنوان خروجی برمی‌گرداند.

<sup>10</sup>Farama

<sup>11</sup>Gymnasium

## تابع شروع مجدد

این تابع، محیط را به حالت اولیه باز می‌گرداند، و حالت جدید را به عامل برمی‌گرداند. در صورتی که شروع بازی شامل المان‌های تصادفی باشد، ورودی این تابع باید کاوش تصادفی<sup>۱۲</sup> را نیز به عنوان ورودی بگیرد.

## انواع فضاهای حالت و عمل

در حین تعریف محیط، باید فضاهای حالت و عمل را نیز تعریف کنیم. هر فضا می‌تواند از انواع زیر باشد:

- گسسته<sup>۱۳</sup>: به ازای ورودی  $n$ ، در این فضا می‌توانیم از اعداد صحیح از ۰ تا  $n-1$  استفاده کنیم. این حالت بیشتر برای فضای خروجی استفاده می‌شود.
- گسسته چندتایی<sup>۱۴</sup>: این فضا مشابه فضای گسسته است، با این تفاوت که می‌توانیم چند عدد از این فضا را به عنوان خروجی بگیریم. مشابه با گسسته است، ولی جای ورودی یک عدد، یک آرایه از اعداد است؛ هر عدد نشان‌دهنده تعداد حالت‌های ممکن در هر بعد است.
- فضای بسته<sup>۱۵</sup>: نشان‌دهنده یک فضای چندبعدی پیوسته است، که باید حد پایین و بالای هر بعد را مشخص کنیم.
- دوتایی چندبعدی<sup>۱۶</sup>: حالت چندبعدی است، که هر بعد مقدار صفر یا یک می‌تواند به خود بگیرد. به کمک فضاهای فوق، می‌توان فضاهایی از ترکیب حالت‌های فوق تعریف کرد، که به انواع زیر است:
- دیکشنری<sup>۱۷</sup>: یک دیکشنری پایتونی از فضاهای مختلف است.
- چندتایی (توپل)<sup>۱۸</sup>: یک چندتایی مرتب از فضاهای مختلف است.
- دنباله<sup>۱۹</sup>: یک دنباله از فضاهای مختلف است.

<sup>12</sup>Random Seed

<sup>13</sup>Discrete

<sup>14</sup>MultiDiscrete

<sup>15</sup>Box

<sup>16</sup>MultiBinary

<sup>17</sup>Dict

<sup>18</sup>Tuple

<sup>19</sup>Sequence

## تابع پایان

در اتمام کار با محیط، این تابع فراخوانی می‌شود تا محیط بسته شود، و منابع آن آزاد شوند.

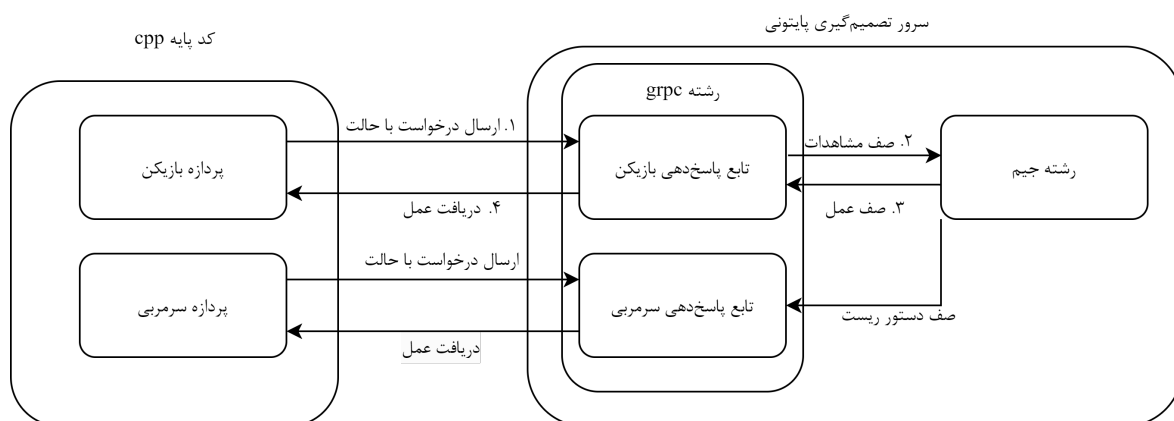
## تابع ترسیم

با توجه به اینکه اکثر محیط‌های جیم، به کمک کتابخانه‌هایی مانند پای‌گیم<sup>۲۰</sup> ترسیم می‌شوند، این تابع برای ترسیم محیط در هر گام استفاده می‌شود. همچنین به عنوان ورودی این تابع می‌توان معین کرد که ترسیم به چه گونه‌ای انجام شود، چرا که ممکن است در حین آموزش برای تسریع عملیات، نیازی به ترسیم نباشد. از آن‌جا که در محیط ما وظیفه ترسیم با نمایش‌گر خارجی انجام می‌شود، این تابع برای ما کاربردی ندارد.

## ۴-۴-۲ نحوه اقدام با فضای ربوکاپ

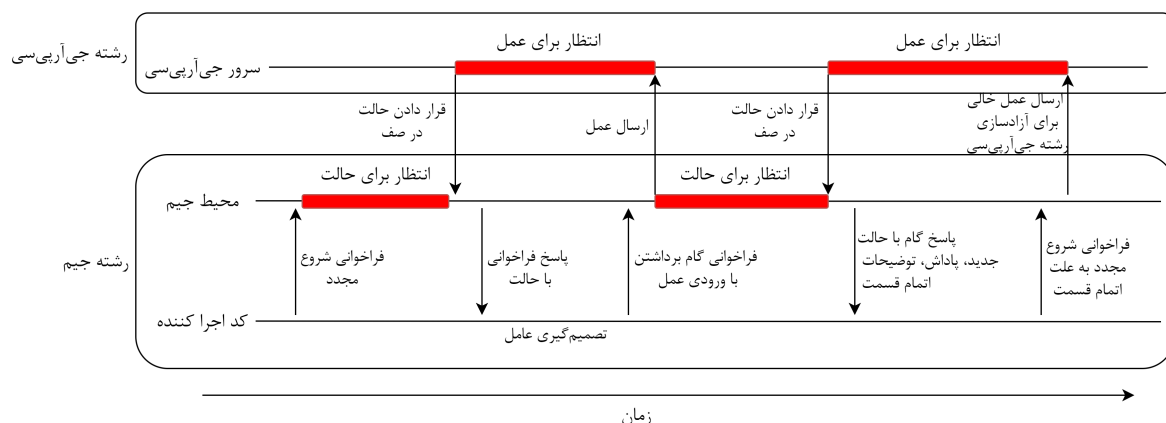
برای پیاده‌سازی محیط جیم در فضای ربوکاپ، از کد پایه gRPC استفاده می‌کنیم. فرآیند سرو کردن درخواست‌های این پروتکل را با کمک کتابخانه‌های استاندارد پایتون به یک رشته جدا منتقل می‌کنیم، که با کمک دو صف با رشته محیط جیم ارتباط دارد. در هنگام دریافت حالت از بخش پایه سی‌پلاس‌پلاس، رشته سرویس‌دهنده حالت را داخل صف مشاهدات می‌گذارد، و منتظر می‌شود که رشته جیم، تصمیم عامل را بر اساس مشاهده ثبت‌شده در صف عمل‌ها قرار دهد. صف سومی نیز برای سرمربی وجود دارد، که در صورت نیاز به شروع مجدد محیط، از سمت جیم پر می‌شود. کد سرمربی تمرینی در هر لحظه از اجرایش چک می‌کند و در صورت خالی نبودن صف، دستورات لازم برای از نو کردن محیط را می‌فرستد. در صورت خالی بودن صف نیز عمل خالی به کد پایه پس می‌فرستد.

<sup>20</sup>PyGame



شکل ۴-۷: نحوه کلی ارتباط کد پایه، تصمیم‌گیرنده جی‌آرپی‌سی و جیم

در شکل ۴-۸ به طور دقیق می‌توان ترتیب فراخوانی توابع، و نحوه استفاده از محیط جیم را مشاهده کرد. کد اجرا کننده<sup>۲۱</sup> دقیقاً فرم مشابه سایر محیط‌های جیم را دارد و از این رابط به طور کامل طبیعت می‌کند. بخش‌های قرمز رنگ به معنی قفل بودن رشته‌ی اجرایی و در انتظار ماندن برای محتویات صف است.



شکل ۴-۸: ترتیب فراخوانی توابع برای اجرای جیم با سرور ربوکاپ

داخل شکل، یک قسمت که فقط یک گام تصمیم‌گیری دارد را می‌توان دید. در واقعیت مراحل «گام برداشتن» تا قبل از «فراخوانی شروع مجدد» به تعداد گام‌های قسمت، تکرار می‌شوند. برای محیط پنالتی، در صورتی که توپ قابل ضربه‌زدن نباشد، حرکت صحیح همیشه قطع توپ با حداکثر سرعت برای ضربه مجدد است. بنابراین فقط حالت را در شرایطی به جیم ارسال می‌کنیم که توپ قابل ضربه‌زدن باشد.

<sup>21</sup>Driver Code

## ۳-۴-۴ فضای حالت و فضای عمل عامل

### فضای حالت

فضای حالت انتخاب‌شده یک بسته ۹ بعدی است، که شامل ویژگی‌های زیر می‌باشد: موقعیت قطبی بازیکن نسبت به مرکز دروازه (اندازه و زاویه)، موقعیت دکارتی توپ (ایکس و ایگرگ)، موقعیت دکارتی دروازه‌بان (ایکس و ایگرگ)، زاویه نسبی حریف نسبت به توپ، موقعیت قطبی دروازه‌بان نسبت به بازیکن (اندازه و زاویه).

علت اهمیت عاملی مثل زاویه بدن دروازه‌بان، قابلیت حرکت سریع‌تر بازیکنان در راستای مستقیم است. می‌توان عواملی هم‌چون سرعت بازیکنان را نیز اثر داد، اما برای همگرایی بهتر و سریع‌تر از این عوامل صرف نظر شده.

### فضای عمل

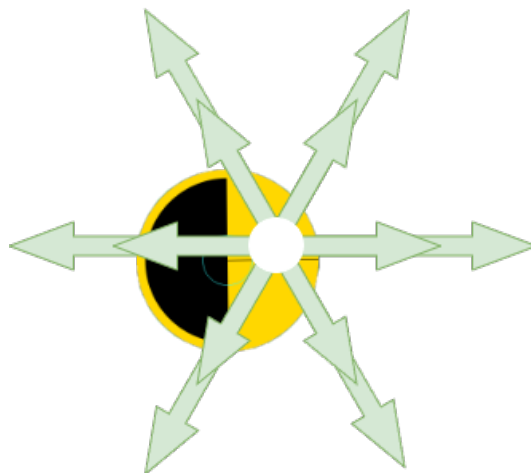
همان‌طور که گفته‌شد، عامل به صورت خودکار به دنبال توپ می‌رود و ما با کمک یادگیری تقویتی قرار است این عامل را به یادگیری ضربه‌زدن به دروازه برسانیم. برای این منظور، از رفتار سطح متوسط ضربه با سرعت دلخواه در راستای دلخواه (KickOneStep) استفاده می‌کنیم تا محاسبات مستقل از سرعت توپ قبلی و شتاب توپ شوند.

به این منظور، فضای عمل ما نیاز است سرعت و زاویه ضربه را تعیین کند. با توجه به اینکه برخی الگوریتم‌ها مانند DQN بهترین عمل را از بین اعمال ممکن انتخاب می‌کنند و نیاز به فضای عمل گسسته دارند، و سایر الگوریتم‌ها مانند DDPG نیاز به فضای عمل پیوسته دارند، محیط را در دو حالت گسسته و پیوسته پیاده‌سازی کردیم.

در فضای پیوسته، دو مقدار سرعت توپ و زاویه ضربه به عامل داده می‌شود. سرعت توپ بین صفر تا یک می‌باشد که نشان‌دهنده سرعت توپ نسبت به حداکثر سرعت ممکن است. زاویه ضربه بین منفی یک تا یک می‌باشد که نرمال‌شده زاویه مطلق ضربه بین  $-180$  تا  $180$  درجه است.

در حالت گسسته نیز، خروجی ما سرعت توپ و زاویه است، اما به جای مقادیر پیوسته، زوایای ضربه ممکن را به ۱۲ قسمت مساوی تقسیم کرده‌ایم و سرعت توپ را به ۱۰ قسمت مساوی تقسیم کرده‌ایم. می‌توان این تقسیم‌بندی را به راحتی تغییر داد و تاثیرات آن روی سرعت یادگیری و نتیجه نهایی آن را بررسی کرد. در شکل ۴-۹ می‌توان یک نمونه از تقسیم‌بندی فضای عمل را مشاهده کرد. از آنجا که سرعت به ۲ درجه تقسیم شده‌است، حالت‌های ممکن آن معادل با نصف حداکثر سرعت توپ و حداکثر

سرعت توپ می‌باشد.



شکل ۴-۹: تقسیم‌بندی فضای عمل به ۶ زاویه و ۲ سرعت

#### ۴-۴-۴ طراحی پاداش

همان‌طور که گفته شد، محیط جیم در حالت‌هایی که توپ قابل ضربه‌زدن است، یا توپ به بیرون رفته و یا توسط دروازه‌بان گرفته شده اجرا می‌شود. در این حالت‌ها باید بر اساس شرایط قبلی، عمل و شرایط جدید پاداشی محاسبه کنیم که به عامل کمک می‌کند تا راحت‌تر به هدف خود نزدیک شود.

#### پاداش‌های پایانی

همان‌طور که گفته شد، چهار حالت پایانی داریم:

۱. حالت گل زدن: با توجه به رسیدن به هدف، پاداش بزرگی به عامل می‌دهیم که این مقدار، ۱۵۰۰ فرض شده است.
۲. حالت بیرون رفتن توپ: از آن‌جا که تنها با ضربات بد این سناریو رخ می‌دهد، پاداش بسیار بزرگ منفی ۵۰۰ دارد.
۳. گرفتن توپ توسط دروازه‌بان: این حالت از آنجا که بهتر از بیرون رفتن توپ است، پاداش منفی کوچک‌تری دارد و مقدار ۲۰۰- دارد.
۴. حالت اتمام زمان: به این حالت، پاداش ۱۵۰- تخصیص داده شده، تا عامل در حالتی که هنوز به گل‌زدن نرسیده است، مستقیماً توپ را به دروازه‌بان ندهد. در واقع با کوچک‌تر در نظر گرفتن این



پاداش، عامل را به کاوش بیشتر وامی‌داریم.

### پاداش در حالت عادی

در حالت بین دو ضربه، پاداش را بر اساس سه فاکتور حساب می‌کنیم:

- می‌خواهیم عامل را به حرکت به سوی دروازه واداریم، در نتیجه از تفاضل فاصله توپ با دروازه در حالت جدید و حالت قبلی به عنوان معیار اصلی استفاده می‌کنیم. در واقع از عکس این فاکتور استفاده می‌کنیم، چرا که کاهش فاصله ویژگی مثبتی است.
- نزدیک شدن بیش از حد به دروازه‌بان ویژگی خوبی نیست، پس از ضربی از تفاضل فاصله توپ با دروازه‌بان در بین دو حالت استفاده می‌کنیم.
- پاداش ثابت ۱۰- که برای تشویق عامل برای سریع‌تر رسیدن به حالت گل می‌باشد. از این رو که عامل این پاداش را فقط در لحظات ضربه زدن می‌بیند، این فاکتور عامل را به زدن ضربات با قدرت بیشتر تشویق می‌کند.

## ۴-۵ پیاده‌سازی یادگیری تقویتی

### ۴-۵-۱ پیاده‌سازی شبکه کیو عمیق

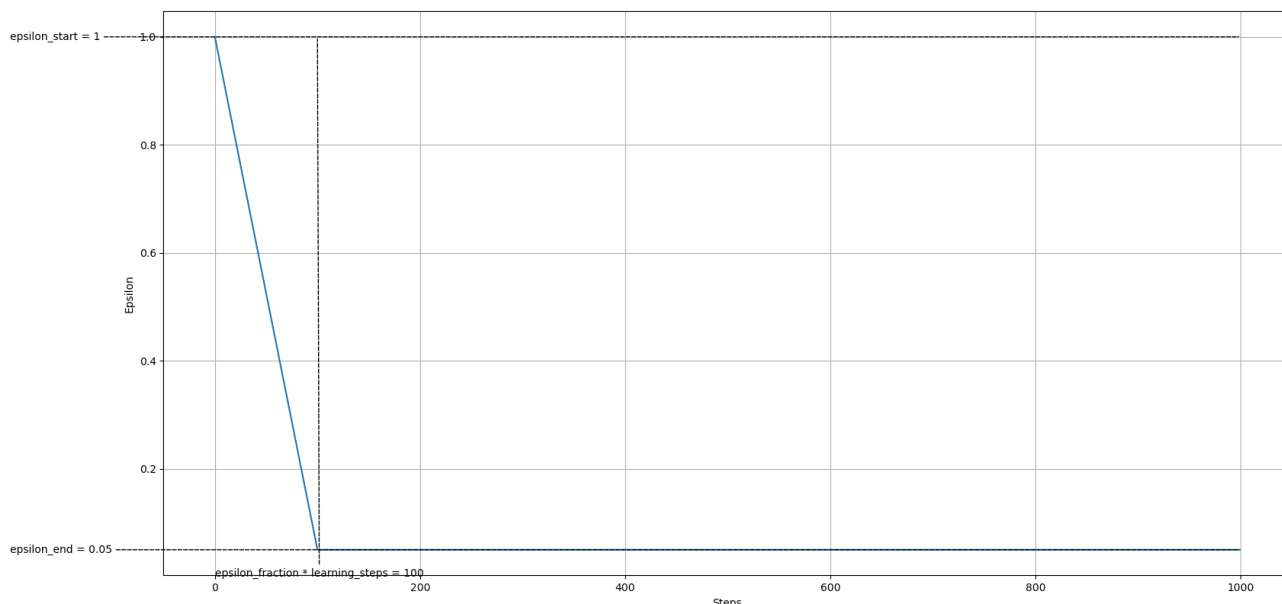
با کمک کتاب‌خانه پای‌تورچ، یک شبکه عصبی با یک لایه پنهان و تابع فعال‌سازی یک‌سوساز<sup>۲۲</sup> پیاده‌سازی شد. این شبکه، ورودی به ابعاد فضای حالت محیط دارد و خروجی به اندازه فضای عمل است. تمامی ابرپارامترهای ما به عنوان ورودی تابع سازنده مدل داده می‌شوند.

سپس تابع یادگیری فراخوانی می‌شود، که ورودی آن مشخص می‌کند که چند گام از محیط برای آموزش استفاده شود. این تابع، با استفاده از الگوریتم DQN، به تعداد مراتب در محیط گام بر می‌دارد و تجارب را به بافر اضافه می‌کند. در گام‌هایی که شماره آن ضربی از ابرپارامتر `update_freq` است، شبکه را به کمک گزینش ریزدسته‌ای از تجربیات و الگوریتم گرادیان، به‌روز می‌کند. همچنین در گام‌هایی که شماره آن ضربی از ابرپارامتر `target_update_freq` است، شبکه هدف را به‌روز می‌کند.

همانطور که در فصل‌های قبل اشاره شد، نرخ کاوش تصادفی به مرور زمان کاهش می‌یابد. در ابرپارامترها، نرخ اولیه، نرخ نهایی، و درصد گام‌هایی که این کاهش نرخ در آن رخ می‌دهد با پارامترهای

<sup>22</sup>ReLU

$\epsilon_{start}$ ،  $\epsilon_{end}$  و  $\epsilon_{fraction}$  مشخص می‌شود. پارامتر  $\epsilon_{fraction}$  در واقع نشان می‌دهد که در چند درصد اول گام‌های آموزشی، نرخ کاوش تصادفی به نرخ نهایی می‌رسد. در شکل ۴-۱۰ می‌توان نمودار کاهش نرخ کاوش تصادفی را مشاهده کرد. در این مثال آموزش برای ۱۰۰۰ گام رخ داده، نرخ اولیه ۱ و نرخ نهایی ۰.۰۵ انتخاب شده‌است، و کسر اِپسیلون  $10^{-23}$  است.



شکل ۴-۱۰: کاهش نرخ کاوش تصادفی در طی آموزش

## ۴-۵-۲ پیاده‌سازی سایر الگوریتم‌ها به کمک کتاب‌خانه

یکی از مهم‌ترین مزایای پیاده‌سازی رابط استاندارد جیم، ایجاد امکان استفاده از پیاده‌سازی‌ها و کتاب‌خانه‌های از پیش موجودی‌اند که وجود دارند. با توجه به اینکه اکثر الگوریتم‌های یادگیری تقویتی در کتاب‌خانه‌هایی مانند استیبل بیس‌لاینز<sup>۲۴</sup> و تورچ‌آرال<sup>۲۵</sup> پیاده‌سازی شده‌اند، می‌توان از این کتاب‌خانه‌ها برای آموزش عامل‌ها استفاده کرد. در این پروژه، از کتاب‌خانه استیبل بیس‌لاینز برای آموزش عامل‌ها استفاده شده‌است. کافی‌ست ابتدا یک محیط جدید از جیم تعریف کنیم، مدلی از الگوریتم دلخواه بسازیم، و سپس با استفاده از تابع `learn` مدل را به تعداد گام‌های دلخواه آموزش دهیم.

<sup>23</sup>Epsilon fraction

<sup>24</sup>Stable Baselines

<sup>25</sup>TorchRL

```
model = DDPG('MlpPolicy', gym_env)
model = model.learn(1_000_000, progress_bar=True)
model.save("DDPG_model_1M")
```

شکل ۴-۱۱: استفاده از کتابخانه استیبل بیس لاینز

طبق مستندات این کتابخانه، می‌توان توابعی را به مدل به عنوان ورودی داد تا در حین آموزش به صورت مکرر و پس از تعداد گام دلخواه به صورت تناوبی صدا زده شوند. از این توابع می‌توان برای ارزیابی مدل به صورت تناوبی، و ذخیره‌سازی در حین آموزش استفاده کرد که برای مقایسات فصل بعدی بسیار کاربردی است.

## ۴-۶ جمع‌بندی

در این فصل، مروری بر راه‌های مختلفی که می‌توان از آن‌ها برای پیاده‌سازی یادگیری تقویتی استفاده کرد، داشتیم. از جمله این روش‌ها، استفاده از تهاجم نیمه میدانی و پاپرس بودند. سپس با معرفی پلتفرم جیم، نحوه پیاده‌سازی محیط‌های جدید و تعریف فضای حالت و عمل را مورد بررسی قرار دادیم. در ادامه، نحوه ادغام این محیط‌ها با فضای ربوکاپ و پیاده‌سازی یک محیط جدید به نام پنالتی را مورد بررسی قرار دادیم. سپس نحوه پیاده‌سازی یک عامل یادگیری تقویتی با استفاده از شبکه کیو عمیق و کتابخانه استیبل بیس لاینز را بررسی کردیم. در فصل بعدی، نتایج این پیاده‌سازی‌ها را مورد بررسی قرار می‌دهیم.

## فصل پنجم

مقایسه، آزمایش‌ها، و نتایج

در این فصل به بررسی نتایج حاصل از پیاده‌سازی الگوریتم‌ها و مقایسه آن‌ها می‌پردازیم.

## ۵-۱ فرآیند آزمایش

برای آزمایش، ابتدا الگوریتم‌های معرفی شده در فصل‌های قبل را با یک میلیون گام علیه دروازه‌بان کد پایه ایجنت آموزش می‌دهیم، و نمودارهای فرآیند یادگیری را بررسی می‌کنیم. در هر ۵۰۰۰ گام از آموزش مدل به صورت تناوبی ذخیره می‌شود، و در نهایت بهترین مدل‌ها را انتخاب می‌کنیم. لازم به ذکر است که منظور از گام، گام‌های یادگیری تقویتی است، که هر کدام متناظر با یک ضربه به توپ اند و منظور گام‌های شبیه‌ساز فوتبال نیست.

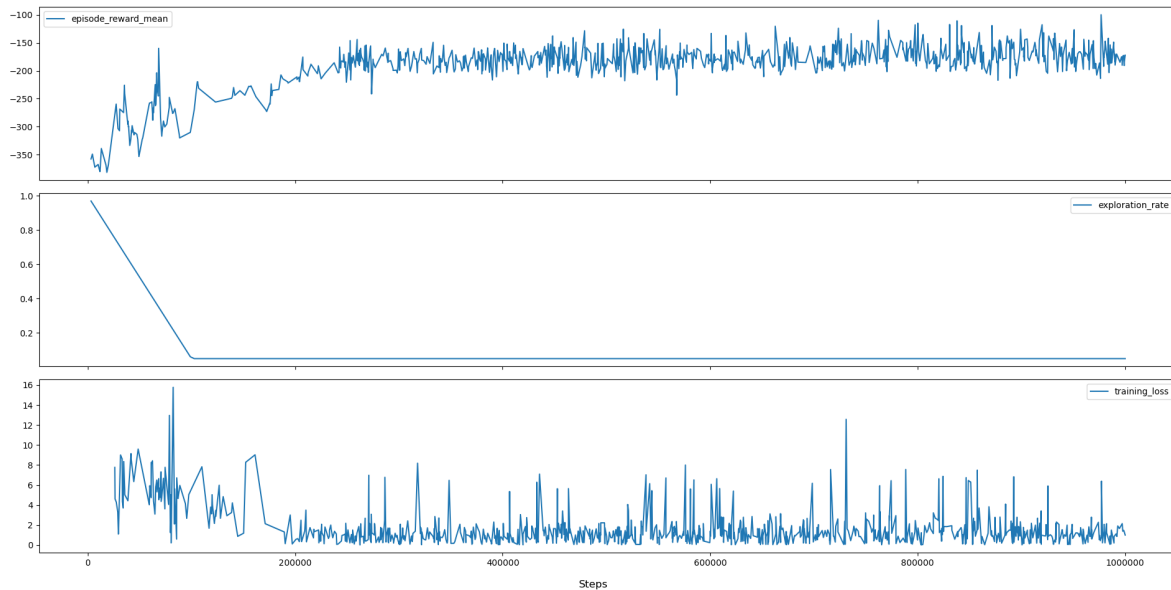
در نهایت مدل انتخاب شده را برای ۱۰۰ ضربه پنالتی علیه کد پایه قرار می‌دهیم تا با سیاست حریصانه و بدون تصمیم‌گیری تصادفی سنجیده شود، و درصد وقوع هر یک از حالت‌های پایانی بازی را بررسی می‌کنیم. در نهایت راهکارهای متفاوتی را برای بهبود نتایج یادگیری امتحان می‌کنیم. تمامی آزمایش‌ها روی یک سیستم با پردازنده‌ی AMD Ryzen 7 6800H، حافظه ۳۲ گیگابایت، و کارت گرافیک NVIDIA RTX 3070Ti انجام شده است، همراه با کتابخانه‌های کودا<sup>۱</sup> CUDA 12.4، و cudnn 8.9.2 و cublas 12.1.3 انجام شده است.

## ۵-۲ ارزیابی الگوریتم‌ها

در ابتدا الگوریتم DQN را برای یادگیری از صفر امتحان می‌کنیم. بعد از یک میلیون گام آموزش که حدود ۱۲ ساعت طول کشید، عامل موفق به کشف تکنیک گل زدن نشد. فرآیند آموزش ۵ بار و با ابرپارامترهای متفاوت آزمایش شد؛ اما در هیچ یک از اجراها عامل موفق به یادگیری نشد. یکی از دلایل ممکن این موضوع می‌تواند تفاوت زیاد بین پاداش‌های موفق و ناموفق باشد؛ چرا که این موضوع ابعاد گرادیان را بزرگ کرده و باعث عدم ثبات می‌شود. به همین منظور یک ضریب مقیاس‌گر برای پاداش‌ها اضافه شد، که باز هم موثر در موفقیت نبود.

---

<sup>۱</sup>CUDA

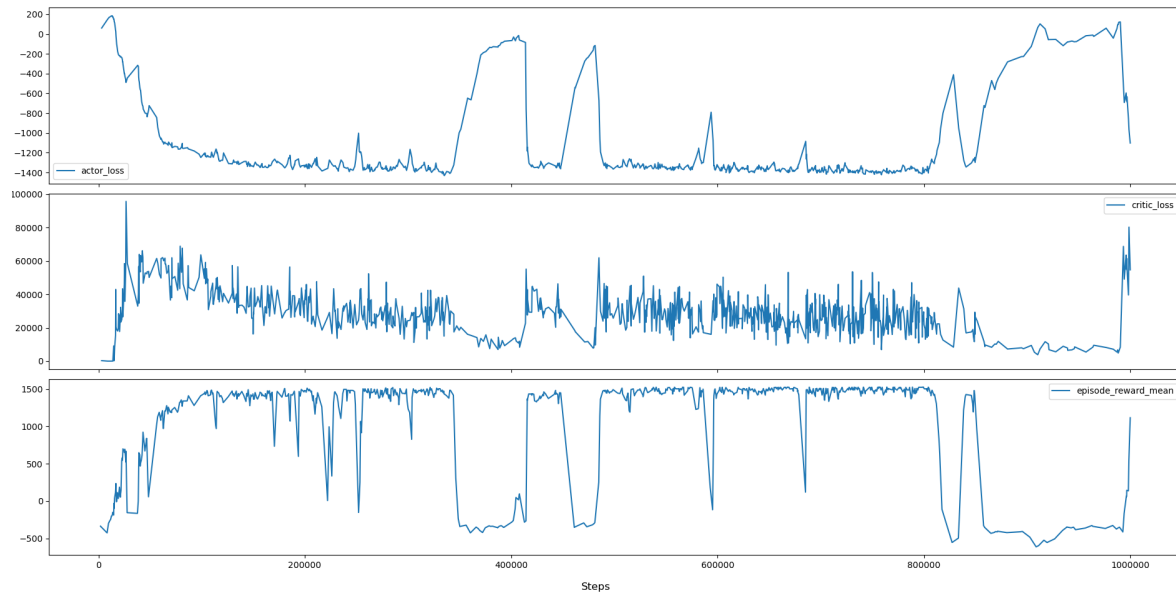


شکل ۵-۱: نمودار سه پارامتر متفاوت برای اجرای الگوریتم DQN

در شکل ۵-۱ می‌توان به ترتیب میانگین پاداش عامل، نرخ کاوش، و خطای شبکه عصبی را در طی زمان آموزش دید. محور افقی این نمودار گام‌های آموزش است، که در اینجا ۱ میلیون گام است. مقیاس پاداش‌ها برای این اجرا، ۲۰۰ بوده‌است و برای رسم نمودار، به مقیاس ۱ تبدیل شده‌است. همانطور که می‌توان مشاهده کرد، نمودار خطای شبکه از گام ۵۰۰۰۰ آغاز شده. این به این علت است که آموزش شبکه از این گام آغاز می‌شود؛ تا پیش از آن بافر با حالت‌های متفاوت پر شود.

همانطور که قابل مشاهده است، پاداش‌ها نسبت به حالت آغازین افزایش می‌یابند اما عامل موفق به کشف حالت گل زدن نشده‌است. میزان خطای شبکه عصبی نیز در طول زمان کاهش یافته و نزدیک به ۱ شده، اما این هم نشان‌دهنده‌ی یادگیری موفق نیست، بلکه شبکه همواره (به درستی) میزان ارزش حالت شکست را خروجی می‌دهد.

همین فرآیند را با الگوریتم DDPG تکرار می‌کنیم. می‌توان مشاهده کرد که عامل در زیر ۱۰۰ هزار گام موفق به کشف تکنیک گل زدن شده‌است. همچنین همانطور که در فصل‌های قبلی ذکر شد، به علت محدود بودن اندازه بافر، عامل ممکن است یادگیری خود را فراموش کند.



شکل ۵-۲: نمودار سه پارامتر متفاوت برای یک اجرای الگوریتم DDPG

در شکل ۵-۲ می‌توان به ترتیب خطای شبکه بازیگر، خطای شبکه نقاد، و میانگین پاداش عامل را در طی زمان آموزش دید. همانطور که مشاهده می‌شود، در حین موفقیت عامل، خطای شبکه‌ها و به ویژه خطای بازیگر کاهش یافته است.

جدول ۵-۱: بهترین نتایج الگوریتم‌ها مقابل کد پایه ایجنت برای صد پنالتی.

خروجی	DDPG	DQN
گل زدن	۹۹	۲
بیرون رفتن توپ	۰	۱۸
گرفتن توسط دروازه‌بان	۱	۷۴
اتمام زمان	۰	۶

با توجه به اینکه الگوریتم DDPG موفق بود، می‌توان حدس‌هایی راجع به عدم موفقیت DQN زد. یکی از دلایل، این است که به علت گسسته بودن اعمال، تعداد گام‌های لازم برای رسیدن به گل بیشتر است، و از آنجا که عامل باید به صورت تصادفی گل زدن را کشف کند، عامل به تعداد دفعات کافی به این حالت نمی‌رسد. همچنین همانطور که در شکل ۲-۴ می‌توان دید، شبکه در الگوریتم DQN به ازای هر عمل، یک خروجی دارد و به همین علت، فضای عمل بزرگ یادگیری را به مراتب دشوار می‌کند. از این رو می‌توان سه راه را برای بهبود یادگیری این الگوریتم پیشنهاد داد:

۱. بهبود گسسته‌سازی عمل‌ها.

۲. استفاده از پاداش‌های بهتر برای حالت‌های غیرنهایی که عامل را به سمت حالت‌های مفیدتر هدایت کند.

۳. تغییر فضای عمل و جداسازی اعمال به دریل زدن و شوت زدن.

## ۳-۵ بهبود تابع پاداش

برای بهبود تابع پاداش، به روش گل‌زنی الگوریتم DDPG نگاه می‌کنیم. به نظر می‌آید که نزدیک شدن دروازه‌بان به توپ عامل منفی‌ای نیست؛ چرا که این حالت ممکن است به عامل کمک کند تا با یک ضربه به توپ، زاویه دروازه را باز کند. همچنین در صورت نزدیکی بیش از حد دروازه‌بان به توپ در حین ضربه، عامل می‌تواند از قابلیت گسسته بودن زمان شبیه‌ساز استفاده کند و با انجام قوی‌ترین ضربه ممکن، توپ را از روی دروازه‌بان بگذرانند. بنابراین برای گام‌های بعدی، عامل فاصله با دروازه‌بان را از محاسبات پاداش حذف می‌کنیم.

## ۴-۵ جداسازی عمل شوت

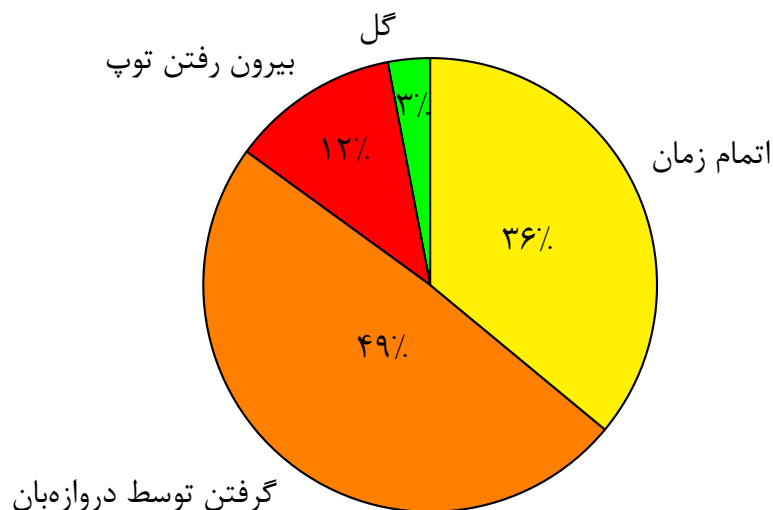
عامل در ابتدای یادگیری، هنوز به درک اینکه در چه حالتی می‌تواند شوت منجر به گل داشته باشد نرسیده است. از آنجا که این عمل با عمل رسیدن به موقعیت شوت‌زنی متفاوت است، عامل ممکن است دیر به دیر به حالت‌هایی که شوت زدن در آن ممکن است برسد. بنابراین در حین یادگیری نسبت به ضربه با سرعت‌های بالا بدبین می‌شود.

برای حل این مشکل، فضای عمل را به گونه‌ای اصلاح می‌کنیم که شوت زدن جدا باشد.

## ۴-۵-۱ شوت با رفتار سطح بالای کد پایه

یک عمل خاص مدنظر می‌گیریم که در صورت انتخاب شدن توسط عامل، شوت کد پایه ایجنت انجام می‌شود. این کد در صورت وجود شوت ممکن، آن را انجام می‌دهد و در غیر این صورت، هیچ عملی انجام نمی‌شود. در صورتی که عامل این عمل را انتخاب کند ولی به گل نرسد، پاداش منفی‌ای به آن داده می‌شود، تا عامل استفاده صحیح از این ابزار را یاد بگیرد.



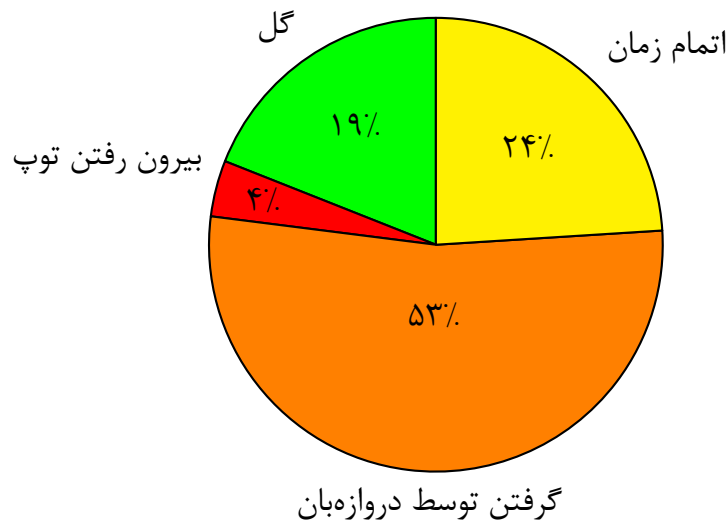


شکل ۵-۳: نتیجه ۱۰۰ ضربه پنالتی با شوت سطح بالا مقابل کد پایه ایجنت برای عامل یادگیری شده

طبق تجربه پیش از این پروژه، این رفتار سطح بالا بسیار محافظ کارانه تصمیم می‌گیرد. از این رو شاید کمک چندانی به عامل برای یادگیری نکند. همچنین از آنجا که استفاده از این قابلیت بسیار وابسته به شبیه‌سازی ضربه به توپ و بررسی امکان قطع توپ توسط دروازه بان است، خلاف ماهیت انجام این پروژه است.

#### ۵-۴-۲ عمل شوت به نقاط ثابت دروازه

فرض کنید تعدادی نقطه ثابت دروازه را انتخاب کنیم. هر یک از این نقاط متناظر با یک عمل شوت است. در این عمل، به توپ با حداکثر سرعت ممکن در راستای این عمل ضربه می‌زنیم. برای انتخاب نقاط، دو تیرک دروازه را به عنوان نقاط ثابت در نظر گرفته، و فضای بین این نقطه را به  $n - 1$  نقطه تقسیم می‌کنیم.



شکل ۵-۴: نتیجه تست ۱۰۰ پناالتی مقابل کد پایه ایجنت با ۵ نقطه شوت

## ۵-۵ تغییر گسسته‌سازی

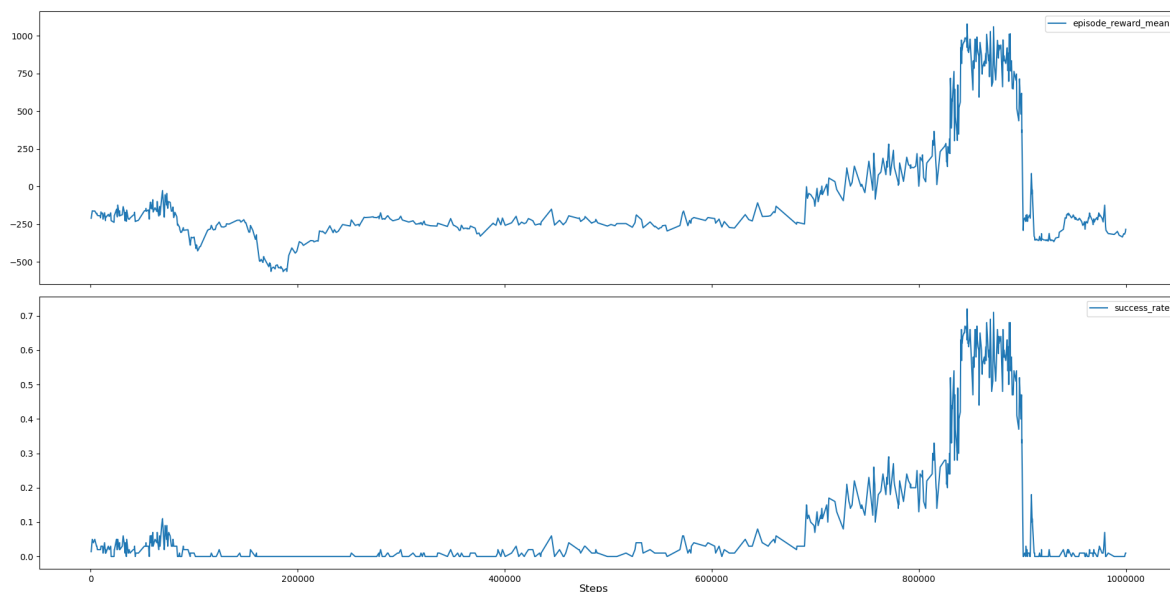
حال که عامل دارای عمل مستقیم شوت است، می‌توانیم گسسته‌سازی قدرت را به گونه‌ای تغییر دهیم که سرعت‌های بالا را نداشته باشد. برای این منظور ضربه به توپ را به سه سرعت تقسیم می‌کنیم: ۱۵ درصد حداکثر سرعت توپ، ۲۵ درصد حداکثر سرعت توپ، و ۵۰ درصد حداکثر سرعت توپ. در واقع اعمال ممکن را به شوت به نقطه یا دریبل با سرعت و زاویه معین تقسیم می‌کنیم.

همچنین به صورت شهودی می‌توان برداشت کرد که تقسیم زوایای جلوی بازیکن به شدت مهم‌تر از زوایای پشت بازیکن است، چرا که به ندرت نیاز می‌شود بازیکن با ضربه به عقب پیشرفت کند. زوایای جلوی بازیکن را به ۹ قسمت تقسیم می‌کنیم، و دو راستا برای دریبل به عقب (بالا عقب و پایین عقب) اضافه می‌کنیم.

با ترکیب این دو تکنیک، فضای حالت از ۱۲۰ عمل، به ۳۸ عمل کاهش می‌یابد. این کاهش از دو راستا به یادگیری ما کمک می‌کند:

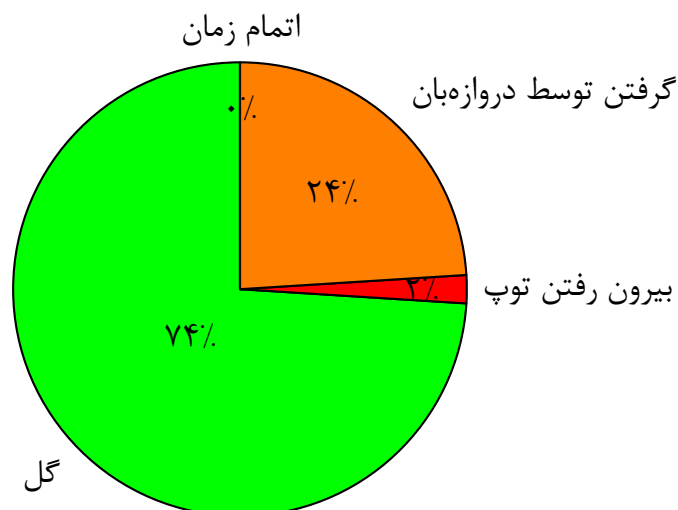
- کاهش ابعاد گرادیان و افزایش سرعت یادگیری
- افزایش احتمال انجام عمل مفید به کمک کاوش، به علت حذف رفتارهایی که به احتمال زیاد مفید نیستند

پس از یک میلیون گام آموزش، نمودار پاداش‌ها و درصد موفقیت در حین آموزش را مشاهده می‌کنیم.



شکل ۵-۵: نمودار میانگین پاداش و میانگین نرخ گل‌زنی DQN با تغییرات گسسته‌سازی

همانطور که در تصویر مشاهده می‌کنید، با این تغییرات عامل موفق به کشف روش گل‌زنی و موفقیت ۷۲ درصد در حین آموزش شده‌است. همچنین می‌توان دید که نمودار پاداش و نرخ گل‌زنی هم‌رفتار و تقریباً هم‌شکل هستند.



شکل ۵-۶: نتیجه تست ۱۰۰ پنالتی مقابل کد پایه ایجنت با بهبود روش گسسته‌سازی

## ۵-۶ بررسی تعمیم‌پذیری

در این بخش، با تست علیه دروازه‌بان تیم‌هایی که مقابل آن‌ها یادگیری رخ نداده، می‌بینیم که آیا این یادگیری تعمیم‌پذیر بوده یا خیر. با توجه به اینکه پس از مسابقات، فایل‌های اجرایی تیم‌ها در اختیار سایر شرکت‌کنندگان قرار می‌گیرد، می‌توانیم این تست را انجام دهیم. از ۳ تیم برای این منظور استفاده می‌کنیم: نسخه ۲۰۲۳ تیم هلیوس<sup>۲</sup> که قهرمان پیاپی چندین دوره مسابقات است، تیم کد پایه ایجنت که مبنای بسیاری از سایر تیم‌ها است، و نسخه ۲۰۲۳ تیم یوشان<sup>۳</sup> که از تیم‌های موفق دیگر است و استراتژی متفاوتی نسبت به دو تیم قبلی دارد.

جدول ۵-۲: نتایج تست الگوریتم DDPG علیه تیم‌های مختلف

خروجی	ایجنت	هلیوس ۲۰۲۳	یوشان ۲۰۲۳
گل زدن	۹۹	۴۰	۲۷
بیرون رفتن توپ	۰	۴	۱۱
گرفتن توسط دروازه‌بان	۱	۴۰	۵۷
اتمام زمان	۰	۱۶	۴

به صورت شهودی و با دیدن نتایج تست، می‌توان پی برد که نتایج تا حدودی قابل تعمیم است، اما شوت زدن عامل بسیار وابسته به مختصات دروازه‌بان است و از این رو، شوت‌های به بیرون و یا گرفته شده توسط دروازه‌بان افزایش یافته‌اند. همچنین دروازه‌بان سایر تیم‌ها مانند هلیوس، در شرایطی که نتوانند به توپ برسند از دستور تکل استفاده می‌کنند، که برد بیشتری نسبت به گرفتن توپ دارد ولی به صورت احتمالاتی است. در صورت تکل موفق حالت گرفتن توسط دروازه‌بان ثبت می‌شود.

جدول ۵-۳: نتایج تست الگوریتم DQN بهبود یافته مقابل تیم‌های مختلف

خروجی	ایجنت	هلیوس ۲۰۲۳	یوشان ۲۰۲۳
گل زدن	۷۱	۰	۱۰
بیرون رفتن توپ	۰	۰	۱۵
گرفتن توسط دروازه‌بان	۲۹	۱۰۰	۷۰
اتمام زمان	۰	۰	۵

با توجه به گسسته‌سازی، هر گونه تغییر در موقعیت دروازه‌بان عمل بعدی عامل را به شدت تحت تاثیر قرار می‌دهد و عامل به شدت با حالت‌های از پیش دیده‌شده فاصله می‌گیرد. از این رو همانطور که پیش‌بینی می‌شد، تعمیم‌پذیری این الگوریتم به تیم‌های دیگر بسیار پایین است.

<sup>۲</sup>Helios2023

<sup>۳</sup>YuShan2023

در نهایت یادگیری الگوریتم DDPG را تکرار می‌کنیم، با این تفاوت که آموزش را از ابتدا و مقابل تیم‌های مختلف انجام می‌دهیم. این آزمایش از این رو است که ببینیم در صورتی که آموزش روی تیم‌های قوی‌تر انجام شود، آیا عامل موفق به یادگیری تکنیک‌های مفیدتری می‌شود یا خیر.

فرآیند آموزش تا زمانی ادامه می‌یابد که عامل به تکنیک گل‌زنی مقابل تیم آموزشی برسد، و تا یک میلیون گام ادامه نخواهد یافت. برای یادگیری مقابل تیم هلیوس فقط ۱۵۰ هزار گام کافی بود تا نرخ موفقیت به بالای ۹۰ درصد برسد که حدود یک ساعت زمان برد. یادگیری مقابل تیم یوشان حدود ۷۰ هزار گام طول کشید که ۲۰ دقیقه زمان برد.

جدول ۴-۵: نتایج تست الگوریتم DDPG علیه تیم‌های مختلف با آموزش مقابل تیم هلیوس

خروجی	ایجنت	هلیوس ۲۰۲۳	یوشان ۲۰۲۳
گل زدن	۴۷	۹۰	۲۳
بیرون رفتن توپ	۱۶	۰	۵
گرفتن توسط دروازه‌بان	۳۷	۷	۶۵
اتمام زمان	۰	۳	۷

جدول ۵-۵: نتایج تست الگوریتم DDPG علیه تیم‌های مختلف با آموزش مقابل تیم یوشان

خروجی	ایجنت	هلیوس ۲۰۲۳	یوشان ۲۰۲۳
گل زدن	۰	۰	۹۹
بیرون رفتن توپ	۰	۰	۰
گرفتن توسط دروازه‌بان	۸۳	۱۰۰	۱
اتمام زمان	۱۷	۰	۰

به نظر می‌رسد که استراتژی متفاوت یوشان باعث می‌شود عامل‌هایی که روی سایر تیم‌ها آموزش دیده‌اند گیج شوند و نتوانند به موفقیت برسند. اما آموزش روی یوشان که به دور زدن دروازه‌بان در آن آسان‌تر است، باعث می‌شود که عامل به موفقیت جلوی سایر تیم‌ها نرسد، چرا که تاکتیک موفق جلوی یوشان منجر به گرفتن توسط دروازه‌بان می‌شود.

از سوی دیگر، با توجه به این که استراتژی هلیوس مشابه ولی قوی‌تر نسبت به کد ایجنت است، آموزش مقابل آن باعث می‌شود که عامل به موفقیت نسبی نزدیک شود.

لازم به ذکر است که حتی اگر عامل در دو دور آموزش مقابل تیم آموزشی به نتایج یکسان برسد، نمی‌توان انتظار داشت که نتایج تست مقابل سایر تیم‌ها یکسان باشد. به این منظور که اگر آموزش را دو بار از صفر آغاز کنیم و تا صد درصد موفقیت پیش ببریم، ممکن است نتایج تست مقابل تیم‌های دیگر بین دو حالت نزدیک نباشد. از این رو این آزمایش معیار خیلی دقیقی برای بررسی کمی تعمیم‌پذیری نیست.

در آینده می‌توان راه‌های تولید مدل‌های تعمیم‌پذیرتر را بیشتر بررسی کرد. یکی از این راه‌ها استفاده از حریف‌های متفاوت در حین یادگیری، و روش‌های یادگیری تقویتی مداوم<sup>۴</sup> است.

## ۷-۵ جمع‌بندی

در این فصل، ما الگوریتم‌های یادگیری تقویتی DQN و DDPG را برای یادگیری تکنیک گل‌زنی در فوتبال تک به تک مورد بررسی قرار دادیم. همچنین چند راه حل برای بهبود یادگیری این الگوریتم‌ها ارائه دادیم. در نهایت، تعمیم‌پذیری این الگوریتم‌ها را بررسی کردیم و دیدیم که مطابق انتظار، عامل در مقابل حریف‌هایی که از پیش ندیده، قدرت یادگیری کمتری دارد.

---

<sup>۴</sup>Continual Reinforcement Learning

## فصل ششم

جمع‌بندی، نتیجه‌گیری و پیشنهادات

در این فصل مروری بر سیر مطالب عنوان شده خواهیم داشت. سپس مروری بر نتایج و دستاوردهای این پروژه خواهیم داشت و در نهایت پیشنهاداتی برای ادامه‌ی کار در این موضوع ارائه خواهیم کرد.

## ۱-۶ جمع‌بندی و نتیجه‌گیری

در این پروژه قصد داشتیم تا با استفاده از روش‌های یادگیری تقویتی یک محیط مجازی برای آموزش و تقویت بازیکنان فوتبال ایجاد کنیم. به این منظور ابتدا با مفاهیم اساسی یادگیری تقویتی آشنا شدیم، با مهم‌ترین روش‌های پیاده‌سازی کلاسیک آن آشنا شدیم و در نهایت با روش‌های پیشرفته‌تری همچون یادگیری تقویتی عمیق و برخی الگوریتم‌های آن آشنا شدیم. در دو الگوریتم یادگیری کیو عمیق و یادگیری گرادیان سیاست قطعی عمیق به طور دقیق با فرآیند یادگیری و عوامل موثر در آن آشنا شدیم. در ادامه محیط شبیه‌ساز دو بعدی فوتبال را معرفی کردیم و برخی از ویژگی‌های آن را توضیح دادیم. رفتارهای مجاز بازیکنان را بررسی کردیم، و کمی با تاریخچه ربوکاپ آشنا شدیم. سپس حالت پناستی را به عنوان یک محیط مناسب برای یادگیری تقویتی معرفی کردیم. در بخش بعدی تلاش‌های متفاوتی که برای پیاده‌سازی یادگیری تقویتی در شبیه‌سازی فوتبال انجام شده بود را بررسی کردیم. در نهایت رابط استاندارد جیم را معرفی کردیم و یک پیاده‌سازی برای اتصال این رابط با کد پایه شبیه‌ساز به کمک جی‌آرپی‌سی انجام دادیم.

در نهایت با استفاده از الگوریتم‌های یادگیری تقویتی عمیق، یک مدل برای آموزش سمت مهاجم در موقعیت‌های تک به تک با دروازه‌بان ایجاد کردیم و الگوریتم‌های متفاوت را مقایسه کرده، و پیشنهاداتی برای بهبود عملکرد آن‌ها ارائه کردیم.

## ۲-۶ کارهای آتی

در آینده می‌توان از زیرساخت ایجاد شده در سه راستا استفاده کرد: حل سایر مسائل در لیگ شبیه‌ساز دو بعدی، استفاده از محیط برای ارزیابی الگوریتم‌ها و روش‌های یادگیری، و بررسی تاثیر ویژگی‌های مختلف بر عملکرد مدل‌های یادگیری تقویتی. برخی از پیشنهاداتی که در این زمینه‌ها می‌توان ارائه کرد عبارتند از:

۱. بررسی یادگیری تقویتی در حالت‌های چند عامله: یکی از بزرگترین چالش‌های یادگیری تقویتی، یادگیری در حالت‌های چند عامله‌ای است که نیاز به هماهنگی بین عامل‌ها دارد. این زیرساخت



و محیط فوتبال می‌تواند محل مناسبی برای توسعه ایده‌های جدید در این راستا باشد. یک نمونه از این مسائل انتخاب حریف و یارگیری به صورت مجزا برای هر بازیکن است. از چالش‌های این مسئله می‌توان به مشکل همگام بودن عوامل در تصمیم‌گیری، بزرگی فضای حالت، و تفاوت میان اطلاعات عامل‌ها به دلیل دید ناقص اشاره کرد.

۲. بررسی تغییر روش نمایش ویژگی‌ها و تاثیر آن بر عملکرد مدل‌های یادگیری تقویتی: یکی از مهم‌ترین مسائل در یادگیری تقویتی، انتخاب نمایش مناسب برای ویژگی‌ها است. انتخاب نمایش مناسب می‌تواند تاثیر بزرگی بر عملکرد مدل‌ها داشته باشد. به طور مثال، موقعیت بازیکن را می‌توان به صورت دکارتی یا قطبی از مبدا مرکز زمین، مرکز دروازه، دروازه‌بان یا توپ نسبت به بازیکن دیگر نمایش داد. بررسی تاثیر و اهمیت نحوه نمایش ویژگی‌ها بر عملکرد مدل‌ها می‌تواند موضوعی جالب برای تحقیقات آینده باشد.

همچنین افزودن برخی ویژگی‌ها مانند سرعت بازیکنان یا انرژی بازیکن می‌تواند تاثیر مهمی روی عملکرد تیم داشته‌باشد، اگرچه ممکن است به نفرین ابعاد<sup>۱</sup> بر بخوریم.

۳. استفاده از روش‌هایی همچون یادگیری انتقالی<sup>۲</sup> برای افزایش سرعت آموزش: همانطور که در نمودارهای یادگیری مانند شکل ۵-۵ می‌توان دید، عامل زمان زیادی را در ابتدای آموزش صرف جستجوی بخش‌های بیهوده فضای حالت می‌کند. استفاده از یادگیری انتقالی می‌تواند کمک کند از عامل‌های از پیش نوشته‌شده استفاده کنیم تا روند آموزش را سریع‌تر کنیم.

۴. استفاده از روش‌های یادگیری متقابل برای آموزش دروازه‌بان و مهاجم به صورت همزمان: می‌توان از روش‌های یادگیری متقابل برای آموزش دو بازیکن به صورت همزمان استفاده کرد. این روش‌ها می‌توانند بهبود عملکرد هر دو بازیکن را به صورت همزمان بهینه کنند. به کمک این روش‌ها می‌توان همزمان دروازه‌بان و مهاجمی ساخت که هر دو از یادگیری دیگری بهره‌مند شوند و از بهترین تیم‌ها هم قوی‌ترند.

<sup>1</sup>Curse of Dimensionality

<sup>2</sup>Transfer Learning

## منابع و مراجع

- [1] Riedmiller, Martin, Merke, Artur, Meier, David, Hoffmann, Andreas, Sinner, Alex, Thate, Ortwin, and Ehrmann, Ralf. Karlsruhe brainstormers-a reinforcement learning approach to robotic soccer. in *RoboCup 2000: Robot Soccer World Cup IV 4*, pp. 367–372. Springer, 2001.
- [2] Riedmiller, Martin, Gabel, Thomas, Knabe, Johannes, and Strasdat, Hauke. Brainstormers 2d-team description 2005. *RoboCup*, pp. 219–229, 2005.
- [3] Gabel, Thomas and Roser, Constantin. Fra-united—team description 2018. in *RoboCup 2019 Symposium and Competitions: Team Description Papers*. Sydney, Australia, 2019.
- [4] Kalyanakrishnan, Shivaram, Liu, Yaxin, and Stone, Peter. Half field offense in robocup soccer: A multiagent reinforcement learning case study. in *RoboCup 2006: Robot Soccer World Cup X 10*, pp. 72–85. Springer, 2007.
- [5] Hausknecht, Matthew, Mupparaju, Prannoy, Subramanian, Sandeep, Kalyanakrishnan, Shivaram, and Stone, Peter. Half field offense: An environment for multiagent learning and ad hoc teamwork. in *AAMAS Adaptive Learning Agents (ALA) Workshop*, vol. 3. sn, 2016.
- [6] Akiyama, Hidehisa and Nakashima, Tomoharu. Helios base: An open source package for the robocup soccer 2d simulation. in *RoboCup 2013: Robot World Cup XVII 17*, pp. 528–535. Springer, 2014.

- [7] Brockman, Greg, Cheung, Vicki, Pettersson, Ludwig, Schneider, Jonas, Schulman, John, Tang, Jie, and Zaremba, Wojciech. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.
- [8] Moore, Andrew William. Efficient memory-based learning for robot control. tech. rep., University of Cambridge, 1990.
- [9] Paszke, Adam, Gross, Sam, Massa, Francisco, Lerer, Adam, Bradbury, James, Chanan, Gregory, Killeen, Trevor, Lin, Zeming, Gimelshein, Natalia, Antiga, Luca, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
- [10] Raffin, Antonin, Hill, Ashley, Gleave, Adam, Kanervisto, Anssi, Ernestus, Maximilian, and Dormann, Noah. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.
- [11] Howard, Ronald A. Dynamic programming and markov processes. 1960.
- [12] Barto, Andrew G. Reinforcement learning: An introduction by richards' sutton. *SIAM Rev*, 6(2):423, 2021.
- [13] Sutton, Richard S. Learning to predict by the methods of temporal differences. *Machine learning*, 3:9–44, 1988.
- [14] Geist, Matthieu, Scherrer, Bruno, et al. Off-policy learning with eligibility traces: a survey. *J. Mach. Learn. Res.*, 15(1):289–333, 2014.
- [15] Kimura, Hajime, Miyazaki, Kazuteru, and Kobayashi, Shigenobu. Reinforcement learning in pomdps with function approximation. in *ICML*, vol. 97, pp. 152–160, 1997.
- [16] Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Graves, Alex, Antonoglou, Ioannis, Wierstra, Daan, and Riedmiller, Martin. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

- [17] Fedus, William, Ramachandran, Prajit, Agarwal, Rishabh, Bengio, Yoshua, Larochelle, Hugo, Rowland, Mark, and Dabney, Will. Revisiting fundamentals of experience replay. in *International Conference on Machine Learning*, pp. 3061–3071. PMLR, 2020.
- [18] Zhang, Shangdong and Sutton, Richard S. A deeper look at experience replay. *arXiv preprint arXiv:1712.01275*, 2017.
- [19] Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.
- [20] Lillicrap, Timothy P, Hunt, Jonathan J, Pritzel, Alexander, Heess, Nicolas, Erez, Tom, Tassa, Yuval, Silver, David, and Wierstra, Daan. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
- [21] Silver, David, Lever, Guy, Heess, Nicolas, Degris, Thomas, Wierstra, Daan, and Riedmiller, Martin. Deterministic policy gradient algorithms. in *International conference on machine learning*, pp. 387–395. Pmlr, 2014.
- [22] Zare, Nader, Sayareh, Aref, Amini, Omid, Sarvmaili, Mahtab, Firouzkouhi, Arad, Matwin, Stan, and Soares, Amilcar. Pyrus base: An open source python framework for the robocup 2d soccer simulation. *arXiv preprint arXiv:2307.16875*, 2023.