دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

گروه مستقل رباتیک

پایان‌نامه کارشناسی ارشد
گرایش هوش مصنوعی و رباتیک

یادگیری تقلیدی مبتنی بر مدل‌های احتمالاتی در کاربردهای رباتیک

نگارش
علی جوادی

استاد راهنما
دکتر شیری قیداری
دکتر نیک‌آبادی

تیر ۱۳۹۷

به نام خدا

تعهدنامه اصالت اثر

تاریخ:

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

اینجانب علی جوادی متعهد می‌شوم که مطالب مندرج در این پایان نامه 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 پایان نامه 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 پایان نامه 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 پایان نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است.
نقل مطالب با ذکر مآخذ بلامانع است.

علی جوادی

امضا

چکیده

امروزه استفاده از ربات‌ها در صنایع مختلف و در وظایف متنوع به سرعت رو به گسترش است. استفاده از ربات‌ها باعث کاهش هزینه‌ی نیروی انسانی و افزایش دقت در عملکرد می‌گردد. یکی از وظایف معمول که در صنایع مختلف بر عهده‌ی ربات‌ها قرار داده می‌شود، وظیفه‌ی انتقال جسم از یک موقعیت اولیه به موقعیت هدف است. برای آموزش انجام وظایف به ربات الگوریتم‌های متنوعی ارائه شده است که دو دسته‌ی مهم از این الگوریتم‌ها، الگوریتم‌های یادگیری تقلیدی و الگوریتم‌های یادگیری تقویتی هستند. در یادگیری تقلیدی یک عامل طوری آموزش داده می‌شود تا یک کار را از روی نمایش‌ها با یادگیری نگاشتی بین مشاهدات و اعمال انجام دهد. در الگوریتم‌های یادگیری تقویتی نحوه انجام وظیفه از طریق تعیین پاداش و جریمه به ازای هر یک از اعمال ربات، به ربات آموزش داده می‌شود.

در این پایان‌نامه، برای بهبود عملکرد الگوریتم یادگیری تقویتی Q-Learning، از مسیرهای راهنما استفاده شده است که این ایده برگرفته از روش‌های یادگیری تقلیدی می‌باشد. جهت استخراج مسیرهای راهنما می‌توان از ویدئوهایی استفاده کرد که در آن یک جسم توسط انسان به داخل حفره هدایت می‌شود.

اخیرا، برای حل مسائلی که طراحی مدل دینامیکی ربات با پیچیدگی بسیاری همراه است و یا مدل دینامیکی دارای عدم قطعیت می‌باشد، الگوریتم PILCO مورد استفاده قرار می‌گیرد. این الگوریتم که از استنتاج و مدل‌های احتمالاتی برای یادگیری کنترل استفاده می‌کند، پیش از این در مسائل مختلفی مورد استفاده قرار گرفته است، اما از آن در حل مسئله انتقال جسم به درون یک حفره توسط بازوی رباتیک شبیه‌سازی شده بهره برده نشده است. با توجه به وجود ذات عدم قطعیت در این مسئله، الگوریتم PILCO می‌تواند به عنوان راهکاری مناسب برای حل این مسئله در محیط‌های پیوسته مورد استفاده قرار بگیرد. از این‌رو در این پایان‌نامه روشی بر مبنای الگوریتم PILCO ارائه شده است. این الگوریتم به نحوی تغییر داده شده است که محدودیت‌های فضای حالت را نیز در روند محاسباتش در نظر می‌گیرد. جهت ارزیابی روش‌های پیشنهادی، شبیه‌سازی‌ها در محیط‌های گسسته و پیوسته انجام شده است و لازم به ذکر است که جهت انتقال جسم از نقطه اولیه به نقطه هدف از یک ربات شبیه‌سازی‌شده استفاده شده است. نتایج شبیه‌سازی در این دو محیط گسسته و پیوسته حاکی از عملکرد بهتر الگوریتم‌های پیشنهادی نسبت به الگوریتم‌های اصلی یعنی الگوریتم‌های Q-Learning و PILCO است.

واژه‌های کلیدی:

یادگیری تقلیدی، مسیرهای راهنما، یادگیری تقویتی،مدل‌های احتمالاتی، رباتیک

فهرست مطالب

فهرست اشکال

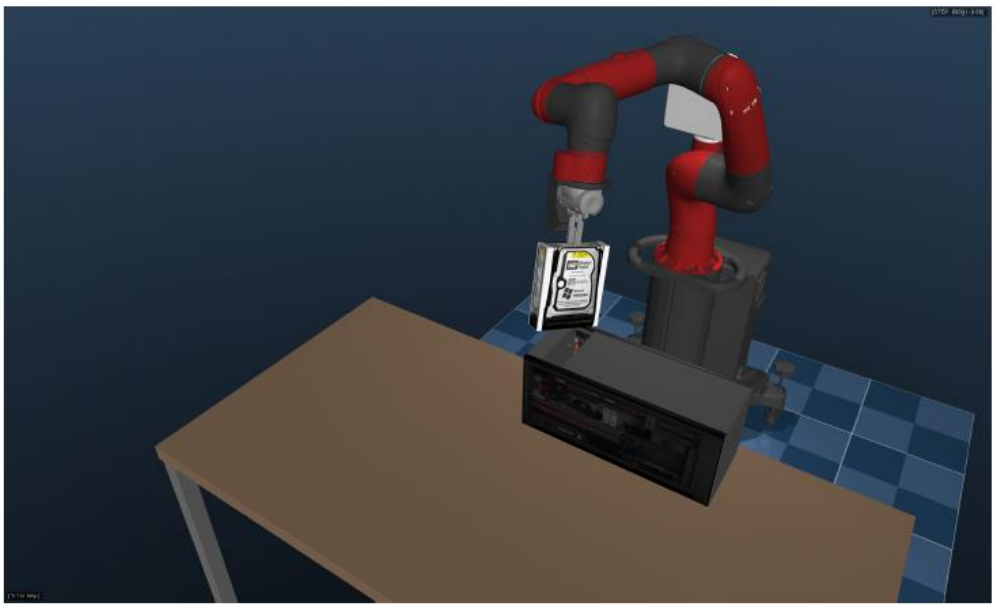
فهرست جداول

فصل اول

۱- مقدمه

مقدمه

در دهه‌های اخیر و با گسترش صنایع مختلف، به منظور کاهش هزینه نیروی انسانی و افزایش دقت عملکرد استفاده از ابزارهای خودکار برای انجام وظایف مختلف بسیار مورد توجه قرار گرفته است. ربات‌ها به عنوان یکی از مهم‌ترین ابزارهای انجام خودکار وظایف در صنایع مختلف مانند خودروسازی، ساخت تجهیزات پزشکی، ساخت تجهیزات کامپیوتر و ... مطرح هستند. ربات‌ها قادرند بسیاری از وظایف را با دقت بالاتری نسبت به انسان انجام دهند. نمونه‌ای از کوچک‌ترین وظایف که می‌تواند توسط یک ربات انجام شود در شکل ۱-۱ آمده است. در این شکل، وظیفه قرار دادن یک قطعه کامپیوتری درون یک محفظه در مراحل مونتاژ یک سیستم کامپیوتری بر عهده ربات قرار داده شده است. پیدا کردن محل مناسب قطعه و انتقال جسم به آن نقطه از اساسی‌ترین چالش‌هایی است که ربات با آن روبرو است.



شکل ۱-۱ نمایی از قرار دادن قطعه کامپیوتری درون محفظه توسط ربات در مراحل مونتاژ کامپیوتر.[1]

برای آموزش ربات به منظور انجام وظایف مختلف، الگوریتم‌های متنوعی ارائه شده است. دو دسته از مهم‌ترین الگوریتم‌ها برای این منظور، الگوریتم‌های یادگیری تقلیدی و یادگیری تقویتی هستند. روش کار در الگوریتم‌های یادگیری تقلیدی به این صورت است که نمونه‌هایی از انجام درست وظیفه به ربات نمایش داده می‌شود تا ربات بتواند نحوه انجام وظیفه را از روی نمونه‌ها بیاموزد. در الگوریتم‌های یادگیری تقویتی، نحوه

انجام وظیفه از طریق تعیین پاداش و جریمه به ازای هر یک از اعمال ربات به ربات آموزش داده می‌شود. در این پایان‌نامه، الگوریتم یادگیری تقویتی Q-Learning [2]، با استفاده از تزریق مسیرهای راهنما به سمت یک الگوریتم یادگیری تقلیدی سوق داده شده است. انجام این کار باعث بهبود عملکرد الگوریتم و کاهش زمان یادگیری می‌گردد. در بخش دوم از روش پیشنهادی ارائه شده در این پایان‌نامه، الگوریتم یادگیری تقویتی PILCO [3] به نحوی تغییر داده شده است که امکان در نظر گرفتن محدودیت‌های محیطی از جمله دیوارها در تابع هزینه الگوریتم وجود داشته باشد. افزودن این قابلیت به الگوریتم، عملکرد ربات را از جنبه درصد موفقیت رساندن جسم به هدف بهبود داده است. هر دو روش پیشنهادی، در وظیفه انتقال جسم از نقطه اولیه به درون حفره مورد بررسی و ارزیابی قرار گرفته‌اند، با این تفاوت که الگوریتم Q-Learnning در محیط‌های گسسته و الگوریتم PILCO در محیط‌های پیوسته مورد استفاده قرار گرفته است.

روند ادامه‌ی بخش‌های این پایان‌نامه به این صورت است که در فصل دوم مروری بر کارهای پیشین صورت گرفته در حوزه‌های یادگیری تقلیدی و یادگیری تقویتی خواهیم داشت و دسته‌بندی موجود در این حوزه‌ها را شرح خواهیم داد. سپس در فصل سوم پس از شرح الگوریتم‌های پایه‌ی Q-Learning و PILCO، روش پیشنهادی مربوط به استفاده از مسیرهای راهنما در الگوریتم Q-Learning و روش پیشنهادی مربوط به تزریق دانش محدودیت‌های محیط در الگوریتم PILCO توضیح داده می‌شود. در فصل چهارم محیط‌های شبیه‌سازی مورد استفاده معرفی شده و نتایج آزمایش‌ها در دو محیط پیوسته و گسسته با پارامترهای مختلف بیان می‌گردد. در فصل پنجم نیز نتیجه‌گیری و جمع‌بندی نهایی پایان‌نامه ارائه خواهد شد.

فصل دوم

۲- مروری بر ادبیات و کارهای پیشین

مروری بر ادبیات و کارهای پیشین

در این فصل مروری بر کارهای پیشین مرتبط با روش‌های یادگیری خواهیم داشت. ابتدا در بخش ۱-۲ مرور کلی بر روش‌های یادگیری تقلیدی و سپس روش‌های یادگیری تقویتی ارائه می‌شود. سپس با توجه مدل استفاده شده در این پایان‌نامه، در بخش ۲-۲ روش‌های یادگیری تقویتی با جستجوی سیاست و به خصوص روش‌های جستجوی سیاست مبتنی بر مدل با تفصیل بیش‌تری شرح داده می‌شوند.

۱-۲- مرور کلی روش‌های یادگیری تقلیدی و یادگیری تقویتی

۲-۱-۱- روش‌های یادگیری تقلیدی

تکنیک‌های یادگیری تقلیدی برای تقلید رفتار انسان در یک کار معین مناسب هستند. یک عامل (ماشین فراگیر) آموزش داده می‌شود تا یک کار را از روی نمایش‌ها با یادگیری نگاشتی بین مشاهدات و اعمال انجام دهد. ایده آموزش با استفاده از تقلید برای سالیان دراز وجود داشته است، اما این رشته اخیرا به دلیل پیشرفت‌های صورت گرفته در محاسبات و حسگرها و همچنین به دلیل نیاز فزاینده به برنامه‌های کاربردی هوشمند مورد توجه قرار گرفته است. الگوی یادگیری با استفاده از تقلید در حال محبوب شدن است چون آموزش کارهای پیچیده با در اختیار داشتن حداقل دانشی از دانش خبره در مورد آن کارها را آسان می‌کند. روش‌های یادگیری تقلیدی به طور بالقوه مسئله آموزش یک کار را به تهیه نمایش کاهش می‌دهند بدون اینکه نیاز به برنامه‌نویسی صریح یا طراحی توابع پاداش ویژه آن کار باشد. حسگرهای مدرن، به سرعت قادر به جمع‌آوری و انتقال حجم زیادی از داده‌ها هستند و پردازنده‌های با قدرت محاسباتی بالا امکان پردازش سریع را فراهم می‌کنند تا به موقع داده‌های دریافتی را از حسگرها به اعمال نگاشت کنند. این موضوع، بستری مناسب را برای خیلی از کاربردهای بالقوه هوش مصنوعی از قبیل ربات‌های انسان‌نما، خودروهای خودران، تعامل انسان کامپیوتر و بازی‌های کامپیوتری فراهم می‌کند که به مشاهده و عکس‌العمل بلادرنگ نیاز دارند. با این حال، از آنجایی که یادگیری با استفاده از تقلید مجموعه چالش‌های خودش را دارد الگوریتم‌های تخصصی برای یادگیری موثر و قوی مدل‌ها نیاز است. در این بخش، روش‌های یادگیری تقلیدی را مورد بررسی قرار می‌دهیم و گزینه‌های طراحی در مراحل مختلف فرآیند یادگیری را ارائه می‌کنیم. به طور دقیق‌تر، پس‌زمینه و انگیزه برای این رشته

را معرفی کرده و هم‌چنین چالش‌های ویژه مسئله تقلید را مورد تاکید قرار می‌دهیم. هم‌چنین، روش‌های طراحی و ارزیابی کارهای یادگیری تقلیدی دسته‌بندی و مرور می‌شوند. از آنجایی که در منابع علمی روش‌های یادگیری در رباتیک و بازی‌ها محبوب هستند و مجموعه گسترده‌ای از مسائل و متدلوژی‌ها را پیش رو قرار می‌دهند، توجه ویژه‌ای به این زمینه‌ها می‌شود. هم‌چنین در این بخش، به طور گسترده در مورد ترکیب رویکردهای یادگیری تقلیدی با استفاده از روش‌ها و منابع مختلف و نیز در مورد درآمیختن دیگر روش‌های یادگیری حرکت برای بهبود تقلید بحث می‌کنیم.

۱-۱-۱-۲- مقدمه

در سال‌های اخیر، اساسا نیاز به عامل‌های هوشمند که قابلیت تقلید رفتار انسان را دارند افزایش یافته است. پیشرفت‌ها در رباتیک و فناوری ارتباطات کاربردهای بالقوه زیادی به وجود آورده است که به هوش مصنوعی‌ای نیاز دارند که نه‌تنها تصمیمات هوشمند می‌گیرد بلکه هم‌چنین قادر است اعمال حرکتی را در موقعیت‌های گوناگون اجرا کند. خیلی از جهت‌گیری‌های آینده در فناوری متکی به این توانایی عامل‌های هوش مصنوعی است که هنگام مواجهه با موقعیت مشابه، همانند انسان رفتار کنند. خودروهای خودران، ربات‌های امدادرسان و تعامل انسان – کامپیوتر مثال‌هایی از این رشته‌ها هستند. خصوصا برای مورد آخر به خاطر گرایش اخیر به واقعیت مجازی مصرف‌کننده و سیستم‌های ضبط حرکت، فرصت‌ها برای کاربردهای جدید در حال افزایش هستند. در این کاربردها و خیلی از کارهای رباتیکی، ما با مسئله اجرای یک عمل با توجه به حالت فعلی عامل و محیط پیرامونش مواجه هستیم. تعداد سناریوهای ممکن در یک کاربرد پیچیده خیلی زیادتر از آن حدی است که با برنامه‌نویسی صریح پوشش داده شود و از همین رو یک عامل موفق باید قادر باشد سناریوهای دیده‌نشده را مدیریت کند. در حالی که چنین کاری می‌تواند به عنوان یک مسئله بهینه‌سازی فرموله‌سازی شود، به طور گسترده پذیرفته شده که داشتن دانش قبلی که توسط فرد خبره فراهم شده است موثرتر و مقرون به صرفه‌تر از جستجو بدون دانش قبلی برای یک راه حل است[4]. علاوه بر این، بهینه‌سازی از طریق آزمون و خطا به توابع پاداشی نیاز دارد که مخصوص هر کار طراحی شده‌اند. می‌توان تصور کرد که حتی برای کارهای ساده، تعداد اعمال ممکن که عامل می‌تواند انجام دهد به طور نمایی زیاد می‌شود. تعریف پاداش برای چنین مسئله‌ای مشکل و در خیلی از موارد نامشخص است.

یکی از راه‌های طبیعی‌تر و شهودی‌تر ابلاغ دانش توسط فرد خبره، فراهم کردن نمایش‌هایی برای رفتار مطلوب است که نیاز است یادگیرنده از آنها تقلید کند. برای (انسان) معلم انتقال دانشش از طریق نمایش آسان‌تر از این است که آن را طوری که یادگیرنده خواهد فهمید بیان کند [5]. در این بخش روش‌هایی مرور می‌شوند که با استفاده از آنها به عامل‌های مصنوعی آموزش داده می‌شود تا سلسله اعمال پیچیده را از طریق تقلید انجام دهند.

یادگیری تقلیدی یک رشته تحقیقاتی میان‌رشته‌ای است. به طور ویژه‌تر، در این بخش روش‌های هوش مصنوعی را مرور می‌کنیم که برای یادگیری سیاست‌هایی استفاده می‌شوند که مسائل را مطابق با نمایش‌های انسان حل می‌کنند. این بخش با تمرکز بر روی روش‌های یادگیری، یادگیری برای هر عامل هوشمند اعم از اینکه یک ربات فیزیکی باشد یا یک عامل نرم‌افزاری (از قبیل بازی‌ها، شبیه‌سازی‌ها و غیره) را مورد بحث قرار می‌دهد. منابع علمی مرور شده کاربردهای متنوع را مورد بررسی قرار می‌دهند، اما خیلی از روش‌هایی که استفاده می‌شوند کلی هستند و می‌توانند به کارهای عمومی (کلی) یادگیری حرکت اعمال شوند. فرآیند یادگیری به ایجاد بازنمایی ویژگی، تقلید مستقیم و یادگیری غیرمستقیم دسته‌بندی می‌شود. روش‌ها و منابع یادگیری برای هر فرآیند و همچنین معیارهای ارزیابی و کاربردهای مناسب این روش‌ها مرور می‌شوند.

یادگیری تقلیدی به فراگیری مهارت‌ها و رفتارهای یک عامل از طریق مشاهده یک مربی که کار مورد نظر را به نمایش می‌گذارد گفته می‌شود. با الهام از عصب‌شناسی، یادگیری تقلیدی بخش مهمی از هوش ماشین و تعامل انسان-کامپیوتر است و از ابتدا به آن به عنوان بخش اساسی در آینده رباتیک نگاه شده است[6]. یک الگوی محبوب دیگر، یادگیری از طریق آزمون و خطاست. اما فراهم کردن مثال‌های خوب برای یادگیری از آنها، فرآیند پیدا کردن یک مدل عمل مناسب را تسریع کرده و از گیر افتادن عامل در کمینه محلی جلوگیری می‌کند. علاوه بر این، یادگیرنده می‌تواند خودش به خوبی به یک راه‌حل مناسب برسد یعنی راه‌حلی که به هدف معین قابل سنجشی می‌رسد، اما با روشی که یک انسان با همین کار مواجه می‌شود به طور قابل‌توجهی فرق می‌کند. گاهی اوقات مهم است که اعمال یادگیرنده قابل باور بوده و طبیعی به نظر برسند. در خیلی از زمینه‌های رباتیک و همچنین تعامل انسان-کامپیوتر که کارایی یادگیرنده فقط بستگی به فهم و درک انسان مشاهده کننده از آن دارد این امری ضروری است. بنابراین مطلوب است که به یادگیرنده رفتار مطلوب را از مجموعه‌ای از نمونه‌های جمع‌آوری شده آموزش داد. اما اغلب اوقات این مورد پیش می‌آید که به خاطر تغییرات در کار از قبیل محل

اشیا یا نمایش‌های ناکافی و نامساعد، تقلید مستقیم حرکت فرد خبره کفایت نمی‌کند. بنابراین، تکنیک‌های یادگیری تقلیدی باید قادر باشند که به‌گونه‌ای یک سیاست را از نمایش‌های (در اختیار قرار) داده شده یاد گرفته که بتوانند آن را به سناریوهای دیده نشده تعمیم دهند. بدین ترتیب، عامل به جای اینکه به طور کامل از معلم کپی کند، یاد می‌گیرد که کار را انجام دهد.

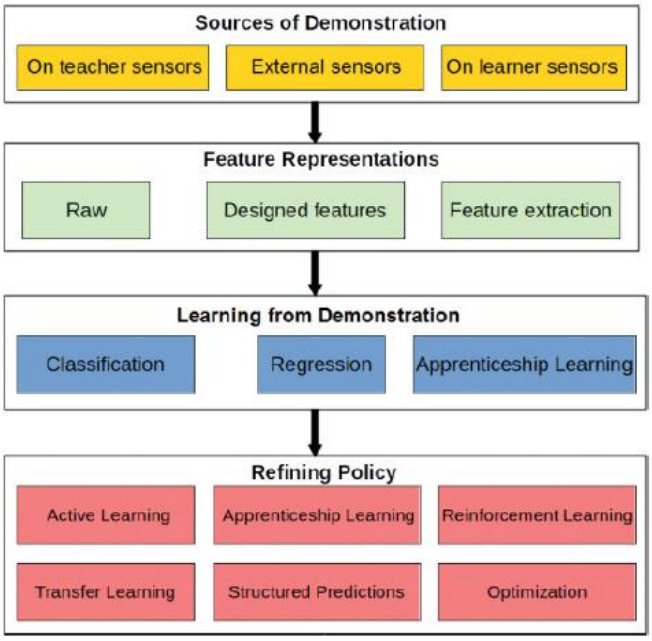
اهمیت رشته یادگیری تقلیدی از ارتباطش با انواع کاربردها از قبیل تعامل انسان-کامپیوتر و ربات‌های امدادگر نشأت می‌گیرد. از این روش برای آموزش آرایه‌ای از کارهای مختلف به ربات‌های با اسکلت‌ها و درجه آزادی‌های مختلف استفاده شده است. بعضی از مثال‌ها، مسائل ناوبری هستند که معمولا از ربات‌های شبیه وسایل نقلیه (با درجه آزادی نسبتا کم‌تر) استفاده می‌کنند که شامل وسایل نقلیه پروازی[7] و وسایل نقلیه زمینی[8] می‌شوند. کاربردهای دیگر، روی ربات‌های با درجات آزادی بالاتر از قبیل ربات‌های انسان‌نما[9] و بازوهای رباتیکی [10] تمرکز دارند. ربات‌های انسان‌نمای با درجه آزادی بالا می‌توانند کارهای گسسته از قبیل ایستادن و کارهای چرخه‌ای از قبیل قدم‌زنی را یاد بگیرند[11]. اگرچه اکثریت کاربردها رباتیک را هدف قرار می‌دهند، یادگیری تقلیدی برای کارهای شبیه‌سازی نیز به کار رفته[11] و حتی در بازی‌های کامپیوتری استفاده می‌شود[12].

یادگیری تقلیدی با استخراج اطلاعات درباره رفتار مربی و محیط پیرامون شامل هر شیء دستکاری شده و هم‌چنین، یادگیری یک نگاشت بین وضعیت و رفتار مشاهده شده کار می‌کند. الگوریتم‌های یادگیری ماشین مرسوم، به عامل‌های با ابعاد بالا و با درجه آزادی زیاد مقیاس نمی‌شوند[13]. بنابراین الگوریتم‌های به خصوصی نیاز است تا بازنمایی‌ها و پیش‌بینی‌های مناسب تولید کنند که قادر به تقلید کارکردهای حرکتی در انسان باشند.

مشابه با یادگیری با ناظر مرسوم که نمونه‌ها جفت ویژگی‌ها و برچسب‌ها را نشان می‌دهند، در یادگیری تقلیدی نیز نمونه‌ها جفت حالات و اعمال را نشان می‌دهند که حالت، وضع فعلی عامل شامل موقعیت و سرعت مفاصل مربوطه و در صورت وجود هدف، وضعیت هدف (از قبیل موقعیت، سرعت، اطلاعات هندسی و غیره) را بازنمایی می‌کند. از همین رو، فرآیندهای تصمیم مارکفی (MDP) به طور طبیعی خودشان را معطوف مسائل یادگیری تقلیدی کرده و به طور رایج برای بازنمایی نمایش‌های خبره استفاده می‌شوند. ویژگی مارکفی تعیین می‌کند که حالت بعدی فقط به حالت و عمل قبلی وابسته است، موضوعی که نیاز به درنظرگرفتن حالات قبل‌تر را در

بازنمایی حالت کم می‌کند[14]. یک چارچوب کاری یادگیری تقلیدی معمول، با به دست آوردن نمایش‌های نمونه از یک خبره شروع می‌شود، نمایش‌هایی که بعدا به صورت جفت حالت – عمل کدگذاری می‌شوند. سپس، این نمونه‌ها برای آموزش یک سیاست استفاده می‌شوند. اما اغلب اوقات برای رسیدن به رفتار مورد نیاز، یادگیری یک نگاشت مستقیم بین حالت و عمل کافی نیست. این موضوع می‌تواند بر اثر برخی از مسائل رخ دهد که خطا در به دست آوردن نمایش‌ها، مغایرت در اسکلت مربی و یادگیرنده (مسئله تناظر) یا نمایش‌های ناکافی از این قبیل هستند. علاوه بر این، کاری که توسط یادگیرنده اجرا می‌شود ممکن است به دلیل تغییرات در محیط، موانع یا اهداف کمی از کار نمایش داده شده متفاوت باشد. بنابراین، غالبا یادگیری تقلیدی گام دیگری را دربرمی‌گیرد که نیاز دارد که یادگیرنده، عمل آموخته شده را اجرا کرده و مطابق با عملکرد خودش در آن کار، سیاست آموخته شده را دوباره بهینه کند. این خودبهبودی می‌تواند با توجه به یک پاداش قابل سنجش انجام شده یا از نمونه‌ها آموخته شود. خیلی از این رویکردها زیر چتر گسترده یادگیری تقویتی قرار می‌گیرند.

شکل ۲-۱، چارچوب کاری یک فرآیند یادگیری تقلیدی را نشان می‌دهد. این فرآیند، با ضبط اعمالی که قرار است از آنها آموخته شود شروع می‌شود که می‌توان از طریق روش‌های حسی مختلف به آن دست یافت. سپس، داده حسگرها پردازش شده تا ویژگی‌هایی استخراج شوند که حالت و محیط پیرامون انجام دهنده کار را توصیف می‌کنند. از ویژگی‌ها برای یادگیری یک سیاست استفاده شده تا با استفاده از آن، رفتار نمایش داده شده را تقلید کرد. سرانجام، با اجازه دادن به عامل برای عمل کردن به سیاست آموخته شده و بهبود آن براساس عملکردش، این سیاست می‌تواند بهتر شود. این گام می‌تواند به ورودی مربی نیاز داشته و یا نداشته باشد. در نظر گرفتن بهبود سیاست به عنوان یک گام بعد از یادگیری می‌تواند قابل درک باشد، اما در بسیاری از موارد همراه با یادگیری از نمایش‌ها اتفاق می‌افتد.



شکل ۲-۱ فلوچارت یادگیری تقلیدی.[15]

کاربردهای یادگیری تقلیدی به خاطر ماهیت میان‌رشته‌ایشان با چالش‌های متنوعی روبرو هستند که عبارتند از:

-شروع این چالش‌ها با فرآیند به دست آوردن نمایش‌هاست که چه داده را از حسگرهای روی یادگیرنده یا مربی ضبط کرده و چه از اطلاعات بصری استفاده شود، سیگنال‌های ضبط شده در معرض نویز و خطاهای حسگر هستند. مشکلات مشابه در طول اجرا، زمانی که عامل محیط را حس می‌کند پیش می‌آیند. حس کردن نویزی یا غیرقابل اعتماد منجر به رفتار اشتباه می‌شود، اگرچه مدل به اندازه کافی آموزش داده شده باشد. مرجع [16] روش‌های مختلف جمع‌آوری نمایش‌ها و چالش‌های هر رویکرد را مورد بررسی قرار داده است.

-مسئله دیگر که مربوط به نمایش است، مسئله یا مشکل تناسب می‌باشد[17]. تناسب، تطبیق قابلیت‌ها، اسکلت و درجه آزادی یادگیرنده و مربی با هم است. نیاز است هر تفاوتی در اندازه یا ساختار بین مربی و یادگیرنده در طول آموزش جبران شود. غالبا در این مورد، یادگیرنده می‌تواند شکل حرکت را از نمایش‌ها یاد گرفته، سپس مدل را با آزمون و خطا بهتر کرده تا به هدفش برسد.

-یک چالش مرتبط، مسئله مشاهده‌پذیری است که در آن، سینماتیک مربی برای یادگیرنده شناخته شده نیست[18]. اگر نمایش‌ها، توسط یک مربی برگزیده تهیه نشده باشند، یادگیرنده ممکن است از قابلیت‌ها و اعمال احتمالی مربی آگاه نباشد. یادگیرنده فقط می‌تواند آثار اعمال مربی را مشاهده کرده و با استفاده از سینماتیک خودش سعی در تکرار آنها داشته باشد.

-از آنجایی که تکنیک‌های سنتی یادگیری ماشین در درجات آزادی بالا به خوبی عمل نمی‌کنند، فرآیند یادگیری با مشکلات عملی نیز مواجه است[13]. به خاطر ماهیت بلادرنگ بسیاری از کاربردهای یادگیری تقلیدی، الگوریتم‌های یادگیری از لحاظ قدرت محاسباتی و محدودیت‌های حافظه محدود هستند، به خصوص، در کاربردهای رباتیکی که به کامپیوترهای جاسازی‌شده[1] برای اجرای پردازش بلادرنگ نیاز است.

-علاوه بر این، غالبا رفتارهای پیچیده می‌توانند به صورت یک تراژکتوری از اعمال کوچک وابسته دیده شوند که فرض i.i.d بودن که در بیش‌تر کارهای یادگیری ماشین اتخاذ شده را نقض می‌کند. سیاست آموخته شده باید قادر باشد که رفتارش را مبتنی بر اعمال قبلی وفق داده و در صورت لزوم ویرایش‌هایی انجام دهد.

-سیاست باید قادر باشد تا خودش را به تغییرات در کار مورد نظر و محیط پیرامون وفق دهد. ماهیت پیچیده کاربردهای یادگیری تقلیدی این امر را تحمیل می‌کند که عامل‌ها باید قادر باشند تا کار مورد نظر را حتی تحت شرایطی که با نمایش آموزشی فرق می‌کند به طور قابل اعتماد اجرا کنند.

-کارهایی که تعامل انسان-کامپیوتر را دربرمی‌گیرند مجموعه جدیدی از چالش‌ها را پیش رو قرار می‌دهند. طبیعتا ایمنی، نگرانی اصلی در چنین کاربردهایی است[19] و اندازه‌گیری‌هایی باید انجام شود تا مانع از صدمه زدن به همکاران شده و ایمنی آنها تضمین شود. علاوه بر این، چالش‌های دیگر به مکانیک ربات مربوط می‌شوند که توانایی آن برای عکس‌العمل نشان دادن به نیروی انسان‌ها و وفق دادن خود به اعمالشان از این موارد هستند.

۲-۱-۱-۲- بازنمایی ویژگی

قبل از یادگیری یک سیاست، مهم است که حالت قابل مشاهده به فرمی بازنمایی شود که برای آموزش، مناسب و کارا باشد. این بازنمایی، بردار ویژگی نامیده می‌شود. یک ویژگی ممکن است شامل اطلاعاتی درباره یادگیرنده،

[1] on-board

محیطش، اشیای قابل دستکاری و عامل‌های دیگر در آزمایش باشد. ویژگی‌های آموزش باید مناسب باشند، به این معنی که اطلاعات کافی را منتقل کنند تا یک سیاست یکپارچه برای حل مسئله مورد نظر را تشکیل دهند. هم‌چنین مهم است که ویژگی‌ها بتوانند از لحاظ زمانی و محدودیت محاسباتی کاربردهای یادگیری تقلیدی به طور کارا استخراج شده و پردازش شوند.

هنگام ضبط داده، سوال مهم این است: چه چیزی را باید تقلید کرد؟ در بیش‌تر کاربردهای واقعی، غالبا محیط برای بازنمایی کلیت آن خیلی پیچیده است، چون معمولا مقدار زیادی اطلاعات نامرتبط یا زاید دارد. بنابراین، ضروری است در نظر بگیریم که چه جنبه‌هایی از نمایش‌ها را می‌خواهیم در اختیار یادگیرنده قرار دهیم.

۳-۱-۱-۲- یادگیری مستقیم

حال، روش‌های مختلف یادگیری یک سیاست از نمایش‌ها را مورد بررسی قرار می‌دهیم. بعد از در نظر گرفتن این‌که چه چیزی را باید گرفت، این فرآیند به این سوال که چطور باید یاد گرفت مربوط می‌شود.

سرراست‌ترین راه برای یادگیری یک سیاست از نمایش‌ها تقلید مستقیم است، یعنی این که یک مدل باناظر از نمایش یاد می‌گیرد که در آن عملی که توسط خبره در اختیار قرار گرفته، به صورت برچسب برای نمونه داده شده عمل می‌کند. از همین رو، مدل قابلیت این را دارد که عمل مناسب را به هنگام مواجهه با یک وضعیت پیش‌بینی کند. روش‌های یادگیری باناظر به دو دسته دسته‌بندی و رگرسیون تفکیک می‌شوند.

- **دسته‌بندی**

دسته‌بندی، یک کار محبوب در یادگیری ماشین است که در آن به طور خودکار مشاهدات به مجموعه متناهی از کلاس‌ها دسته‌بندی می‌شوند. یک دسته‌بند $h(x)$ برای پیش‌بینی کلاس y ای که یک مشاهده مستقل x به آن مربوط است استفاده می‌شود که $y \in Y$ و $Y = \{y_1, y_2 \cdots y_p\}$ مجموعه متناهی از کلاس‌ها و $x =$ $\{x_1, x_2 \cdots x_m\}$ برداری از m ویژگی است. در دسته‌بندی باناظر، $h(x)$ با استفاده از یک مجموعه داده با n نمونه برچسب‌زده شده آموزش داده می‌شود که $i = 1, 2 \cdots n$ و $x^i \in X, y^i \in Y$ هستند.

رویکردهای دسته‌بندی، زمانی استفاده می‌شوند که اعمال یادگیرنده می‌توانند به کلاس‌های گسسته دسته‌بندی شوند[16]. این رویکرد برای کاربردهایی از قبیل ناوبری[20] و شبیه‌سازهای پرواز[7] مناسب است که در آن

به عمل می‌توان به عنوان یک تصمیم نگاه کرد. در مرجع [20]، یک مدل ترکیبی گوسی (GMM) برای پیش‌بینی تصمیمات ناوبری آموزش داده شده است. در مرجع [21]، متادسته‌بند[1]ها برای یادگیری انجام بازی‌های کامپیوتری استفاده شده‌اند. دسته‌بند پایه‌ای که در این مرجع استفاده شده یک شبکه عصبی است. در بازی اتومبیل‌رانی کارت[2]، دستورات دسته بازی[3] آنالوگ به ۱۵ دسته گسسته‌سازی شده و مسئله به یک مسئله دسته‌بندی ۱۵ کلاسه کاهش می‌یابد. بنابراین شبکه عصبی‌ای که استفاده شده ۱۵ نود خروجی دارد. بازی ماریو بروز[4] از یک کنترل‌کننده گسسته استفاده می‌کند. اعمال با فشردن یک یا تعداد بیش‌تری از چهار دکمه انتخاب می‌شوند. بنابراین در شبکه عصبی مربوطه، عمل مربوط به یک فریم با چهار نود خروجی بازنمایی می‌شود. این امر، دسته‌بند را قادر می‌سازد تا برای یک نمونه چندین کلاس را انتخاب کند. اگرچه نتایج امیدوارکننده هستند، استدلال می‌شود که استفاده از تکنیک کنترل بهینه معکوس[22] به عنوان دسته‌بند پایه ممکن است مفید باشد. در مرجع [23]، این آزمایش‌ها با استفاده از رگرسیون تکرار شده تا ورودی آنالوگ در سوپر تاکس کارت[5] یاد گرفته شود. برای بازی ماریو بروز، چهار دسته‌بند SVM جایگزین شبکه عصبی شده تا مقدار هر یک از چهار کلاس دودویی پیش‌بینی شود. همچنین، دسته‌بندی می‌تواند برای اتخاذ تصمیماتی که اعمال سطح پایین‌تری را شامل می‌شوند استفاده شود. در مرجع [5]، در یک شبیه‌سازی چندعامله فوتبال، تصمیمات سطح بالا به وسیله یک دسته‌بند پیش‌بینی می‌شوند. تصمیماتی از قبیل "نزدیک شدن به توپ" و "دریبل کردن به سمت گل" می‌توانند به طور قطعی و با استفاده از اعمال سطح پایین‌تر اجرا شوند. یک مطالعه تجربی انجام شده تا ارزیابی کند که کدام دسته‌بندها برای کار یادگیری تقلیدی مناسب‌تر هستند. چهار دسته‌بند مختلف بر حسب دقت و زمان یادگیری مقایسه می‌شوند. نتایج نشان می‌دهند که تعدادی از دسته‌بندها می‌توانند پیش‌بینی‌هایی با دقت قابل مقایسه انجام دهند. اما زمان یادگیری نسبت به تعداد نمایش‌ها می‌تواند تا حد زیادی تغییر کند[5]. در مرجع [24]، شبکه‌های عصبی بازگشتی (RNN) به منظور یادگیری تراژکتوری‌هایی از نمایش‌ها برای دستکاری شی استفاده شده‌اند. RNN ها هنگام در نظر

[1] Meta-classifier
[2] kart
[3] Joystick
[4] Mario Bros
[5] Super Tux Kart

گرفتن عمل بعدی، حافظه اعمال قبلی را با هم ادغام می‌کنند. ذخیره‌سازی حافظه، شبکه را قادر می‌کند تا رفتار اصلاحی مانند بازیابی از شکست را یاد بگیرد به این شرط که مربی چنین سناریویی را نمایش داده باشد.

- **رگرسیون**

روش‌های رگرسیون، برای یادگیری اعمال در یک فضای پیوسته استفاده می‌شوند. برخلاف دسته‌بندی، روش‌های رگرسیون ورودی حالت را به یک خروجی عددی که یک عمل را نشان می‌دهد نگاشت می‌کنند. بنابراین، این روش‌ها برای اعمال حرکتی سطح پایین مناسب هستند تا تصمیمات سطح بالاتر، به خصوص زمانی که اعمال به صورت مقادیر پیوسته بازنمایی می‌شوند مانند ورودی از یک دسته بازی[23]. رگرسور $I(x)$ ، یک نمونه مستقل x را به جای مجموعه‌ای از کلاس‌ها به یک مقدار پیوسته y نگاشت می‌کند که $y \in \mathbb{R}$ مجموعه اعداد حقیقی است. به طور مشابه، رگرسور با استفاده از مجموعه‌ای از نمونه‌های برچسب خورده (x^i, y^i) آموزش داده می‌شود که $y^i \in \mathbb{R}$ و $i = 1, 2 \cdots n$ است.

تکنیکی که به طور رایج استفاده می‌شود رگرسیون وزن‌دهی‌شده محلی[1] (LWR) است. LWR برای یادگیری تراژکتوری‌ها مناسب است چون این حرکات از دنباله‌ای از مقادیر پیوسته تشکیل شده‌اند. مثال‌هایی از این قبیل حرکات کارهای ضربه‌زدن[2][10] و گام برداشتن[25] هستند که در مورد اول نیاز است که عامل یک تراژکتوری حرکت را اجرا کرده تا از یک نقطه عبور کرده و یک هدف را بزند و در مورد دوم عامل باید یک تراژکتوری تولید کند که منجر به حرکت پایدار و آرام می‌شود. یک کاربرد فراگیرتر، تنیس روی میز است. در مرجع [26]، از رگرسیون بیزی خطی استفاده شده تا به یک بازوی ربات آموزش داده شود که بازی پیوسته تنیس روی میز را انجام دهد. نیاز است که این عامل، با دقت در یک فضای سه بعدی پیوسته و در وضعیت‌های مختلف از قبیل هنگام زدن توپ، بازیابی موقعیت بعد از زدن توپ و آماده شدن برای حرکت بعدی رقیب حرکت کند. نمونه دیگری که به طور رایج برای رگرسیون استفاده می‌شود شبکه‌های عصبی مصنوعی (ANNs) است. شبکه‌های عصبی از این لحاظ با دیگر تکنیک‌های رگرسیون متفاوت هستند که به زمان آموزش زیاد و تعداد نمونه‌های آموزش زیاد نیاز دارند. رویکردهای شبکه عصبی غالبا از مطالعات زیست‌شناسی و عصب‌شناسی الهام گرفته و قصد تقلید فرآیند تقلید و یادگیری انسان‌ها و حیوانات را دارند[4]. استفاده از رگرسیون همراه با یک

[1] Locally weighted regression
[2] Batting

سیستم دینامیکی از مقدمات حرکتی، کاربردهایی برای یادگیری حرکات گسسته و ریتمیک تولید کرده است[27]، اما بیش‌تر رویکردها برروی تقلید مستقیم بدون بهینه‌سازی تمرکز بیش‌تری کرده‌اند[28]. در چنین کاربردهایی یک سیستم پویا[1] یک درجه آزادی را بازنمایی می‌کند، به طوری که هر درجه آزادی هدف و محدودیت‌های متفاوت دارد[29].

سیستم‌های پویا می‌توانند با روش‌های یادگیری ماشین احتمالاتی ترکیب شده تا فواید هر دو رویکرد به دست آید. این امر استخراج الگوهایی که برای کار مورد نظر مهم هستند و تعمیم به سناریوهای مختلف را امکان‌پذیر می‌کند در حالی که توانایی تطبیق و تصحیح تراژکتوری‌های حرکت در زمان واقعی را حفظ می‌کند[30]. در مرجع [30]، تخمین پارامترهای سیستم‌های دینامیکی به صورت یک مسئله رگرسیون ترکیب گوسی (GMR) بازنمایی شده است. این رویکرد یک مزیت بر رویکردهای مبتنی بر LWR دارد چنانچه امکان یادگیری توابع فعال‌سازی به همراه اعمال حرکتی را فراهم می‌کند. روش پیشنهاد شده برای یادگیری حرکت مبتنی بر زمان و حرکت نامتغیر با زمان استفاده شده است. در مرجع [31]، از یک روش مبتنی بر GMM مشابه در یک چارچوب‌کاری وظیفه-پارامتربندی‌شده[2] استفاده شده است که اجازه شکل‌دهی حرکت ربات به صورت تابعی از پارامترهای کار را می‌دهد. نمایش‌های انسان کدگذاری شده تا پارامترهایی که به کار دم دست[3] مرتبط هستند را بازتاب کرده و موقعیت، سرعت و محدودیت‌های نیروی آن کار را مشخص کنند. این کدگذاری به چارچوب‌کاری امکان داده تا حالتی که ربات باید در آن باشد را استنتاج کرده و مطابق آن حرکت ربات را بهینه کند. این رویکرد در زمینه همکاری ربات و انسان (HRC) استفاده شده و قصد بهینه‌سازی مداخله انسان و همچنین تلاش ربات را دارد. در مرجع [32]، یادگیری عمیق با سیستم‌های دینامیکی ترکیب شده است. مقدمات حرکت پویا (DMPs) در اتوانکدرهایی که بازنمایی‌های حرکت از داده نمایش را یاد گرفته جاسازی شده‌اند. اتوانکدرها در لایه مخفی ویژگی‌ها را به صورت غیرخطی به فضای نهفته[4] با بعد کم‌تر نگاشت می‌کنند. در این رویکرد، واحدهای مخفی به DMP ها محدود شده تا لایه مخفی به بازنمایی پویای سیستم محدود شود.

[1] Dynamic system
[2] task-parameterized
[3] At hand
[4] Latent

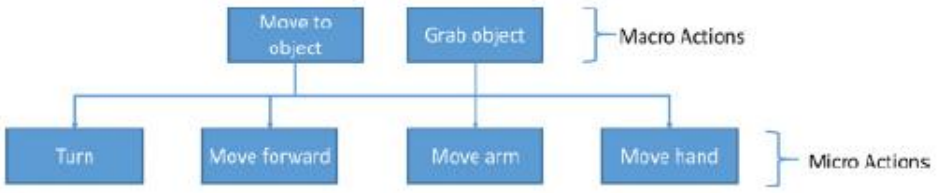
در هر دو روش دسته‌بندی و رگرسیون با توجه به منابع مدل‌های یادگیری، یک تصمیم طراحی می‌تواند اتخاذ شود. یادگیرنده‌های تنبل از قبیل kNN و LWR نیازی به آموزش ندارند اما نیاز دارند که به هنگام پیش‌بینی کردن تمام نمونه‌های آموزشی حفظ شوند. از طرف دیگر، مدل‌های آموزش داده شده از قبیل ANN و SVM به زمان آموزش نیاز دارند اما به محض ایجاد یک مدل، دیگر به نمونه‌های آموزشی احتیاجی نیست و فقط مدل ذخیره می‌شود که این امر مانع از هدر رفتن حافظه می‌شود. هم‌چنین این مدل‌ها می‌توانند منجر به زمان‌های پیش‌بینی خیلی کوتاه شوند.

- **مدل‌های سلسله مراتبی**

به جای استفاده از یک مدل برای بازتولید رفتار انسان، از یک مدل سلسله‌مراتبی می‌تواند استفاده شود که اعمال آموخته شده را به تعدادی از فازها تجزیه می‌کند. یک مدل دسته‌بندی می‌تواند استفاده شود تا تصمیم گرفته شود که کدام عمل یا زیرعمل از مجموعه‌ای از اعمال ممکن باید در زمان مورد نظر انجام شود. سپس از یک مدل متفاوت برای تعیین جزئیات عمل انتخاب‌شده استفاده می‌شود که در آن، هر عمل سطح پایین ممکن یک مدل مشخص و معین دارد. شکل ۲-۲ مثالی از اعمال سلسله‌مراتبی را نشان می‌دهد. در مرجع [33]، از یک رویکرد سلسله‌مراتبی برای یادگیری تقلیدی در دو مسئله مختلف استفاده شده است. مسئله اول، ایرهاکی[1] است که در مقابل یک رقیب بازی می‌شود و هدف، شوت کردن یک گوی درون دروازه رقیب و در عین حال حفاظت از دروازه خودی است. بازی دوم، ماربل میز[2] است. هزارتو[3] می‌تواند حول محورهای مختلف کج شده تا توپ به سمت انتهای هزارتو حرکت کند. هر کدام از دو کار ذکر شده مجموعه‌ای از اعمال سطح پایین به نام مقدمات حرکتی دارند که امکان‌های بازی کردن برای عامل را تشکیل می‌دهند (مثل شوت مستقیم، دفاع از دروازه و غلت دادن توپ به سمت گوشه). در مرحله اول، از یک دسته‌بند نزدیک‌ترین همسایه برای انتخاب عملی که باید انجام شود استفاده شده است. با مشاهده حالت بازی، این دسته‌بند در نمایش‌های تهیه‌شده توسط انسان خبره دنبال مشابه‌ترین نمونه‌ها می‌گردد و عمل مقدماتی که در آن نقطه توسط انسان انتخاب شده است را بازیابی می‌کند. مرحله بعدی تعیین هدف عمل انتخاب شده است برای مثال سرعت توپ یا موقعیت گوی هنگامی که عمل مقدماتی کامل می‌شود. سپس، هدف در یک مدل رگرسیون استفاده شده تا

[1] air hockey
[2] marble maze
[3] Maze

پارامترهای عملی که هدف مورد نظر را بهینه می‌کند پیدا شود. هدف، از k نزدیک‌ترین نمایش همسایه که در مرحله قبل پیدا شدند استنتاج می‌شود. اهداف در آن نمایش‌ها ورودی یک مدل رگرسیون وزن‌دار برای انجام عمل مقدماتی هستند. با سبکی مشابه در مرجع [34]، از یک دسته‌بند برای تصمیم‌گیری در یک کار مرتب‌سازی استفاده شده که شامل اعمال بزرگ[1] (wait, sort left, sort right and pass) است. هر عمل بزرگ، اعمال حرکتی گذرا[2] از قبیل برداشتن یک توپ و حرکت دادن و قرار دادن آن توپ را شامل می‌شود.



شکل ۲-۲ مثالی از یادگیری سلسله‌مراتبی عمل‌ها. [15]

۴-۱-۱-۲- یادگیری غیرمستقیم

در این بخش، در مورد روش‌های غیرمستقیم یادگیری سیاست‌ها بحث می‌شود که می‌تواند یا یادگیری مستقیم را کامل کرده یا جایگزین آن شود. سیاست می‌تواند با استفاده از نمایش‌ها، تجربه یا مشاهده بهتر شود تا یا دقیق‌تر باشد یا در شرایط دیده نشده عمومی‌تر و قوی‌تر باشد. غالبا یادگیری مستقیم به تنهایی برای بازتولید رفتار قوی و شبیه انسان در عامل‌های هوشمند کافی نیست. این محدودیت می‌تواند به دو عامل اصلی نسبت داده شود: ۱) خطاهای در نمایش و ۲) تعمیم ضعیف. به خاطر محدودیت‌های تکنیک‌های به دست آوردن داده که عبارتند از مسئله تناسب، خطای حسگر و تاثیرات فیزیکی در نمایش‌های حرکتی[16]، تقلید مستقیم می‌تواند منجر به کارایی ناپایدار یا نادقیق شود (این مشکل، به ویژه در کارهایی از قبیل زدن گوی یا رسیدن به جایی یا چیزی نمود پیدا می‌کند که نیاز به حرکت دقیق در فضای پیوسته دارند). برای مثال، در مرجع [11]، یک ربات که قصد راه رفتن با تقلید مستقیم نمایش‌ها را دارد می‌افتد چون نمایش‌ها به طور دقیق

[1] Macro
[2] Temporal

ویژگی‌های فیزیکی از قبیل وزن و مرکز جرم ربات که در این کار درگیر هستند را در نظر نمی‌گیرند. اما بهبود سیاست از طریق آزمون و خطا این عوامل را در نظر گرفته و یک حرکت پایدار تولید می‌کند.

در حالی که تعمیم‌پذیری مقوله‌ای مهم در کارهای یادگیری ماشین است یک مورد خاص تعمیم‌پذیری در کاربردهای یادگیری تقلیدی مورد تاکید قرار گرفته است. رایج است که نمایش‌های انسان به صورت دنباله‌ای از اعمال در اختیار قرار می‌گیرند. وابستگی هر عمل به بخش قبلی دنباله، فرض i.i.d بودن نمونه‌های آموزشی که در یادگیری باناظر برای تعمیم‌پذیری مهم است را نقض می‌کند[21]. علاوه بر این، چون انسان‌های خبره فقط نمونه‌های درست را در اختیار قرار می‌دهند یادگیرنده آماده مدیریت خطاها در تراژکتوری نیست. اگر یادگیرنده در هر نقطه‌ای، از عملکرد بهینه در تراژکتوری فاصله بگیرد (چیزی که در هر کار یادگیری ماشین انتظار می‌رود)، در معرض یک موقعیت دیده‌نشده قرار می‌گیرد که مدل برای وفق یافتن با آن آموزش داده نشده است. یک مثال واضح در مرجع [35] آورده شده است که در آن از یادگیری باناظر استفاده شده تا یک سیاست برای راندن یک ماشین یاد گرفته شود. با توجه به این‌که نمایش‌های انسان فقط شامل رانندگی خوب بدون تصادف یا حتی نزدیک خطر است، زمانی که خطایی رخ می‌دهد و ماشین از تراژکتوری‌های نمایش داده شده منحرف می‌شود یادگیرنده نمی‌داند که چگونه جبران کند.

- **یادگیری تقویتی**

یادگیری تقویتی یک سیاست را یاد گرفته تا از طریق آزمون و خطا یک مسئله را حل کند.

در یادگیری تقویتی، یک عامل به صورت یک MDP مدل شده که حرکت در یک فضای حالت را یاد می‌گیرد. یک MDP متناهی شامل چهارتایی (S,A,T,R) است که S مجموعه متناهی حالت‌ها، A مجموعه اعمال ممکن، T مجموعه احتمالات انتقال حالت و R یک تابع پاداش است. TP_{sa} شامل مجموعه‌ای از احتمالات است که P_{sa} احتمال رسیدن در حالت s به شرط عمل a است به طوری که $a \in A$ و $s \in S$ است. تابع پاداش $R(s_k, a_k, s_{k+1})$ یک پاداش آنی برای انجام یک عمل در یک حالت معلوم و قرار گرفتن در یک حالت جدید برمی‌گرداند $a_k \rightarrow s_{k+1}$ که k گام زمانی است. این پاداش با گذشت زمان به اندازه ضریب کاهش $\gamma \in [0,1)$ کاهش می‌یابد و هدف عامل، بیشینه کردن امید ریاضی پاداش کاهش‌یافته در هر گام زمانی است.

یادگیری تقویتی با یک سیاست تصادفی شروع شده و پارامترهایش را براساس پاداش‌هایی که از اجرای این سیاست به دست می‌آورد تغییر می‌دهد. یادگیری تقویتی می‌تواند به تنهایی برای یادگیری یک سیاست در انواع کاربردهای رباتیک استفاده شود. اما اگر یک سیاست از نمایش یاد گرفته شود، یادگیری تقویتی می‌تواند برای تنظیم پارامترها استفاده شود. تهیه کردن نمونه‌های مثبت و منفی برای آموزش یک سیاست، با کاهش فضای جستجوی دردسترس به یادگیری تقویتی کمک می‌کند[4]. بهبود سیاست با استفاده از یادگیری تقویتی گاهی اوقات ضروری است برای نمونه در صورتی که بین مربی و یادگیرنده مغایرت‌های فیزیکی وجود داشته باشد یا برای کاهش خطاهای موجود در به دست آوردن نمایش‌ها. یادگیری تقویتی همچنین می‌تواند برای آموزش سیاست برای وضعیت‌های دیده نشده که در نمایش‌ها پوشش داده نشده‌اند مفید باشد. اعمال یادگیری تقویتی به سیاست آموخته شده به جای یک سیاست تصادفی می‌تواند به طور قابل توجهی سرعت فرآیند یادگیری تقویتی را زیاد کرده و از خطر همگرایی سیاست به کمینه محلی جلوگیری کند. علاوه بر این، یادگیری تقویتی می‌تواند سیاستی را برای اجرای یک کار پیدا کند که به نظر (انسان) شاهد طبیعی نیست. در کاربردهایی که یادگیرنده با یک انسان تعامل می‌کند، برای کاربر مهم است که اعمال عامل را به طور شهودی تشخیص دهد. این امر در مواردی که ربات‌ها در محیط‌های تثبیت شده (از قبیل خانه‌ها و ادارات) قرار داده می‌شوند تا با کاربران (انسان) آموزش دیده نشده تعامل کنند رایج است[36]. با اعمال یادگیری تقویتی به سیاستی که از نمایش‌های انسان آموخته شده است می‌توان از مشکل رفتار ناشناخته پرهیز کرد. در روش‌های یادگیری تقلیدی، زمانی‌که شایستگی کار اجرا شده قابل ارزیابی است غالبا برای بهبود سیاست آموخته شده یادگیری تقویتی با یادگیری از نمایش‌ها ترکیب می‌شود.

در پژوهش‌های اولیه، آموزش با استفاده از نمایش‌های اعمال موفق برای بهبود و افزایش سرعت یادگیری تقویتی استفاده شده است. در مرجع [37]، از یادگیری تقویتی استفاده شده است تا یک سیاست برای انجام یک بازی در محیط پویای دو بعدی آموزش داده شود. روش‌های مختلف برای بهبود سیاست یادگیری تقویتی مورد بررسی قرار گرفته‌اند. نتایج نشان می‌دهند که آموزش یادگیرنده با نمایش‌ها امتیازش را بهتر کرده و به جلوگیری از افتادن یادگیرنده در کمینه محلی کمک می‌کند. همچنین مشاهده شده است که بهبود از آموزش با دشواری کار افزایش می‌یابد. راه‌حل‌های مسائل ساده به سادگی و بدون نیاز به نمایش‌هایی از خبره می‌توانند

استنتاج شوند. اما چنان‌چه پیچیدگی کار افزایش می‌یابد فایده یادگیری از نمایش‌ها مهم‌تر می‌شود و حتی برای یادگیری موفق در کارهای دشوارتر ضروری می‌شود[38] .

در مرجع [39]، GMR برای آموزش یک ربات در کار در دست گرفتن شی استفاده شده است. چون سناریوهای دیده‌نشده از قبیل موانع و موقعیت متغیر شی در این کاربرد مورد انتظار هستند یادگیری تقویتی استفاده شده تا راه‌های جدید برای انجام این کار مورد بررسی قرار گیرد. سیستم آموزش داده شده یک سیستم پویاست که روی تراژکتوری‌های تقلید شده میرایی[1] را اجرا می‌کند. این امر به ربات اجازه داده تا به آرامی به هدفش رسیده و از اینکه یادگیری تقویتی منجر به نوسانات شود جلوگیری می‌کند. استفاده از میرایی در سیستم‌های پویا به هنگام ترکیب یادگیری تقلیدی و یادگیری تقویتی یک رویکرد رایج است[13].

یک کاربرد شگفت‌انگیز یادگیری تقلیدی و تقویتی، آموزش یک عامل برای انجام بازی تخته‌ای "GO" است که با انسان‌های خبره رقابت می‌کند[40]. یک شبکه عصبی کانولوشنی عمیق با استفاده از بازی‌های قبلی آموزش داده شده است. سپس یادگیری تقویتی برای اصلاح وزن‌های شبکه و بهبود سیاست استفاده شده است.

یک رویکرد متفاوت برای ترکیب یادگیری از نمایش‌ها با یادگیری تقویتی در مرجع [41] استفاده شده است. به جای استفاده از نمایش‌ها برای آموزش یک سیاست اولیه، از آنها برای به دست آوردن دانش قبلی برای شکل‌دهی پاداش استفاده شده است[42]. از یک تابع پاداش برای تشویق دست‌یابی به اهداف کوچک‌تر در کار (از قبیل نقاط عطفی[2] که در نمایش‌ها به آنها دست یافته شده است) استفاده شده است. این تابع پاداش با تابع پاداش اولیه ترکیب شده تا هزینه اعمال عامل را در اختیارش قرار دهد. این الگوی استفاده از نمایش‌های خبره برای به دست آوردن یک تابع پاداش مشابه رویکردهای یادگیری تقویتی معکوس است[43].

روش‌های جستجوی سیاست زیرمجموعه‌ای از یادگیری تقویتی هستند که به طور طبیعی مناسب کاربردهای رباتیکی هستند چون به MDP های با ابعاد بالا مقیاس می‌شوند[14]. بنابراین، روش‌های جستجوی سیاست برای ادغام شدن با روش‌های یادگیری تقلیدی مناسب هستند. یک روش گرادیان سیاست در مرجع [44] استفاده شده تا یک سیاست موجود که می‌تواند از طریق یادگیری با ناظر یا برنامه‌نویسی صریح[3] تولید شود را

[1] Damping
[2] Milestone
[3] Explicit programming

بهبود دهد. یک رویکرد مشابه در مرجع [45] در یک سیستم پویا استفاده شده است که قبلا برای یادگیری باناظر از نمایش‌ها استفاده شده بود[46]. این امر منجر به یک سری از کارها شده است که از سیستم پویای معرفی شده در مرجع [46] بهره می‌برند تا از نمایش‌ها یاد گرفته و متعاقبا از یادگیری تقویتی برای خود بهسازی استفاده می‌کنند[13]. از این چارچوب کاری برای آموزش تعدادی از کاربردها به بازوهای رباتیکی از قبیل قرار دادن توپ در فنجان، زدن توپ با پارو[1][10] و انجام بازی تنیس[26] استفاده شده است. به جای استفاده از یادگیری تقویتی برای اصلاح سیاستی که با استفاده از نمایش‌ها آموزش داده شده است، نمایش‌ها می‌توانند برای راهنمایی جستجوی سیاست استفاده شوند. در مرجع [47]، از برنامه‌نویسی پویای تفاضلی[2] استفاده شده تا از نمایش‌های انسان نمونه‌های هدایت‌کننده تولید شود. این نمونه‌های هدایت‌کننده به جستجوی سیاست کمک کرده تا مناطق با پاداش بالا از فضای جستجو را مورد بررسی قرار دهند.

در مرجع [48]، شبکه‌های عصبی بازگشتی در جستجوی هدایت‌شده سیاست گنجانده شده تا پرداختن به مسائل مشاهده‌پذیر جزئی را آسان کند. حافظه‌های پیشین به فضای حالت افزوده شده و هنگام پیش‌بینی عمل بعدی در نظر گرفته می‌شوند. یک رویکرد باناظر از تراژکتوری‌های نمایش‌داده شده استفاده کرده تا در مورد اینکه کدام حافظه‌ها باید ذخیره شوند تصمیم‌گیری کرده، در حالی که از یادگیری تقویتی استفاده شده تا سیاستی را که شامل مقادیر حالت حافظه می‌شود بهینه کند.

یک روش متفاوت برای استفاده از یادگیری تقویتی در یادگیری تقلیدی، استفاده از یادگیری تقویتی برای تهیه نمایش‌های تقلید مستقیم است. این رویکرد به (انسان) مربی احتیاج ندارد چون سیاست از آغاز و با استفاده از آزمون و خطا آموخته شده و سپس برای تولید نمایش‌های آموزش استفاده می‌شود. یک دلیل برای تولید نمایش‌ها و آموزش یک مدل باناظر به جای استفاده مستقیم از سیاست یادگیری تقویتی این است که روش یادگیری تقویتی در زمان واقعی (به طور بلادرنگ) کار نمی‌کند[49]. وضعیت دیگر هنگامی است که سیاست یادگیری تقویتی در یک محیط کنترل شده آموخته می‌شود. در مرجع [50]، یادگیری تقویتی برای یادگیری انواع کارهای رباتیکی در یک محیط کنترل شده استفاده شده است. اطلاعاتی از قبیل موقعیت اشیای هدف در طول این فاز در دسترس است. سپس یک شبکه عصبی کانولوشنی عمیق، با استفاده از نمایش‌های برگرفته از

[1] ball paddling
[2] Differential

سیاست یادگیری تقویتی آموزش داده می‌شود. شبکه عصبی، نگاشت ورودی بصری به اعمال را یاد گرفته و در نتیجه، اجرای کارها را بدون اطلاعاتی که در فاز یادگیری تقویتی به آنها نیاز است یاد می‌گیرد. این روش، از نمایش‌های انسان تقلید می‌کند همان‌طور که انسان‌ها برای تهیه نمایش‌ها از دانش خبره‌ای که در فرآیند آموزش درگیر نشده است بهره می‌برند.

- **بهینه‌سازی**

رویکردهای بهینه‌سازی نیز می‌توانند برای پیدا کردن راه حلی برای یک مسئله معلوم استفاده شوند.

با فرض یک تابع هزینه $f : A \rightarrow \mathbb{R}$ که منعکس‌کننده عملکرد یک عامل است، به طوری که A مجموعه‌ای از پارامترهای ورودی و $\mathbb{R}$ مجموعه اعداد حقیقی است، روش‌های بهینه‌سازی سعی در پیدا کردن پارامترهای ورودی x_0 دارند که تابع هزینه را کمینه می‌کند که $\forall x \in A \quad f(x_0) \leq f(x)$ است.

مشابه یادگیری تقویتی، تکنیک‌های بهینه‌سازی می‌توانند با شروع از یک راه‌حل تصادفی و بهبود پیاپی آن به منظور بهینه‌سازی تابع برازش، برای پیدا کردن راه‌حل‌های مسائل استفاده شوند. الگوریتم‌های تکاملی (EAs) روش‌های محبوب بهینه‌سازی هستند که به طور گسترده برای یافتن تراژکتوری‌های حرکتی در کارهای رباتیکی استفاده شده‌اند[51]. الگوریتم‌های تکاملی برای تولید تراژکتوری‌های حرکتی برای ربات‌های با درجه آزادی بالا و پایین استفاده شده‌اند[52]. روش‌های محبوب هوش گروهی[1] از قبیل بهینه‌سازی ازدحام ذرات[2](PSO) [53] و بهینه‌سازی کلونی مورچگان[3] (ACO) [54] برای تولید تراژکتوری‌هایی برای ناوبری وسیله بدون سرنشین استفاده شده‌اند. این تکنیک‌ها رفتار موجودات زنده را شبیه‌سازی کرده تا یک راه‌حل سراسری بهینه را در فضای جستجو بیابند. مشابه یادگیری تقویتی، الگوریتم‌های تکاملی می‌توانند با یادگیری تقلیدی آمیخته شده تا تراژکتوری‌هایی که از نمایش‌ها آموخته شده را بهبود داده یا سرعت فرآیند بهینه‌سازی را افزایش دهند.

در مرجع [11]، یک الگوریتم ژنتیک (GA) برای بهینه‌سازی تراژکتوری‌های حرکتی نمایش داده شده استفاده شده است. تراژکتوری‌ها به عنوان یک جمعیت آغازی برای الگوریتم ژنتیک استفاده شده‌اند. تراژکتوری‌های ضبط شده به صورت کروموزوم‌هایی کدگذاری شده‌اند که از ژن‌هایی که مقدمات حرکتی را بازنمایی می‌کنند

[1] Swarm intelligence
[2] Particle Swarm Optimization
[3] Ant Colony Optimization

تشکیل شده‌اند. الگوریتم ژنتیک دنبال کروموزومی می‌گردد که یک تابع برازش که موفقیت کار را ارزیابی کرده بهینه می‌کند. تصویر کردن کردن تراژکتوری‌های حرکتی به بعد پایین‌تر، تغییر قابل توجه بین حرکت بهینه‌شده و حرکتی که مستقیما از دست‌کاری جنبشی[1] آموخته شده را نشان می‌دهد[11].

به طور مشابه در مرجع [55]، الگوریتم‌های تکاملی بعد از آموزش عامل‌ها در یک شبیه‌سازی فوتبال استفاده شده‌اند. یک راه‌حل ممکن (کروموزوم)، به صورت مجموعه‌ای از قوانین اگر- آنگاه بازنمایی می‌شود. به دلیل جایگشت‌های متناهی مشاهدات و اعمال، قوانین متناهی هستند. برای ارزیابی مناسب بودن یک راه‌حل از یک تابع وزن‌دار از تعداد اهداف و مقیاس‌های عملکرد دیگر استفاده می‌شود. اگرچه الگوریتم تکاملی اندازه جمعیت کمی داشته و تقاطع[2] را به کار نگرفته، اما نتایج امیدوارکننده‌ای روی قوانین آموخته شده از نمایش‌ها نشان داده است.

در مرجع [35] نیز، از الگوریتم‌های تکاملی برای بهینه‌سازی چندین هدف در یک بازی مسابقه سرعت استفاده شده است. این الگوریتم‌ها یک راه‌حل بهینه‌شده (کنترل‌کننده) را از یک جمعیت اولیه از تراژکتوری‌های رانندگی استنتاج کرده‌اند. ارزیابی کنترل‌کننده‌های استنتاج شده مشخص کرده است که آنها به سبک رانندگی بازیکنانی که از روی آنها مدل‌سازی شده‌اند وفادار می‌مانند. این امر در مورد مقیاس‌های کمی از قبیل سرعت و پیشرفت و در مورد مشاهدات ذهنی[3] از قبیل راندن در وسط جاده صدق می‌کند.

در مرجع [56]، وزن‌های شبکه عصبی به صورت ژنومی که باید بهینه‌سازی شود در نظر گرفته شده است. جمعیت اولیه از طریق آموزش شبکه با نمونه‌های نمایش داده شده فراهم شده تا وزن‌ها را مقداردهی اولیه کنند. از نمایش‌ها نیز برای ایجاد یک مقدار برازش که مربوط به فاصله میانگین مربع خطا از خروجی‌های مورد نظر (اعمال انسان) است استفاده شده است.

در مرجع [57]، از PSO استفاده شده تا مسیر بهینه برای یک وسیله هوایی بدون سرنشین (UAV) از طریق یافتن بهترین نقاط کنترلی روی یک منحنی B-spline پیدا شود. نقاط اولیه‌ای که حکم ذرات اولیه PSO را دارند از اسکلت‌سازی[4] تهیه می‌شوند. یک گونه اجتماعی PSO در مرجع [58] معرفی شده است که از یادگیری

[1] Kinesthetic
[2] Crossover
[3] Subjective observations
[4] Skeletonization

حیوانات در طبیعت که از طریق مشاهده همتایانشان صورت می‌گیرد الهام گرفته است. هر ذره با یک راه‌حل تصادفی شروع کرده و یک تابع برازش برای ارزیابی هر راه‌حل استفاده می‌شود. سپس ذرات تقلیدگر (همه به جز ذره‌ای با بهترین برازش) رفتارشان را از طریق مشاهده ذرات نمایش‌دهنده (ذرات با عملکرد بهتر) تغییر می‌دهند. همانند طبیعت هر تقلیدگر می‌تواند از چندین نمایش‌دهنده یاد گرفته و یک نمایش‌دهنده می‌تواند برای آموزش بیش‌تر از یک تقلیدگر استفاده شود. الگوریتم‌های تکاملی تعاملی (IEAs) [59] الگوی متفاوتی را به کار می‌گیرند. به جای استفاده از ورودی انسان برای شروع یک جمعیت اولیه از راه‌حل‌ها و بهینه‌سازی آنها، الگوریتم تکاملی تعاملی از ورودی انسان برای برآورد برازش (مناسب بودن) راه‌حل‌ها استفاده می‌کند. برای پرهیز از ارزیابی تعداد خیلی زیادی از راه‌حل‌های بالقوه، یک مدل روی نمونه‌های باناظر آموزش داده شده تا ارزیابی کاربر (انسان) را تخمین بزند. در مرجع [60]، برای یادگیری ناوبری ربات جستجوی مبتنی بر برازش با یادگیری سیاست مبتنی بر ارجحیت (PPL) ترکیب شده است. ارزیابی‌های کاربر از PPL جستجو را به مناطقی دور از کمینه محلی هدایت کرده در حالی که جستجوی مبتنی بر برازش دنبال یک راه‌حل می‌گردد. با یک ماهیت مشابه، در مرجع [61] یک ربات آموزش شده تا حرکت بازوی انسان را تقلید کند. با استفاده از نمایش‌ها به عنوان جمعیت اولیه، از نمود اختلاف در درجه آزادی بین (انسان) نمایش‌دهنده و ربات جلوگیری می‌شود. اما به جای استفاده از ورودی انسان برای ارزیابی ذهنی یک راه‌حل، شباهت حرکت ربات به نمایش‌های انسان به صورت کیفی ارزیابی می‌شود. از بازنمایی مفصل مستقل از توالی نمایش‌دهنده و یادگیرنده برای تشکیل یک تابع برازش استفاده شده است. از PSO برای یافتن زوایای مفصل استفاده شده تا این مقیاس شباهت بهینه شود. یک روش متفاوت برای ادغام نمایش‌ها در مرجع [62] پیشنهاد شده است. با الهام از یادگیری تقویتی معکوس، از یک چارچوب کاری تنظیم‌کننده مربعی خطی معکوس (ILQR) استفاده شده تا تابع هزینه‌ای که توسط (انسان) نمایش‌دهنده بهینه شده یاد گرفته شود. سپس به جای روش‌های گرادیان از PSO به منظور یافتن راه‌حلی برای تابع آموخته شده استفاده شده است.

- **یادگیری انتقالی**[1]

یادگیری انتقالی یک الگوی یادگیری ماشین است که از دانش یک کار یا یک حیطه[2] برای بهبود یا تقویت یادگیری کار دیگر استفاده می‌کند.

با فرض حیطه مبدا D_s و کار مبدا T_s ، یادگیری انتقالی به این صورت تعریف می‌شود که یادگیری یک کار هدف T_t در حیطه D_t را با استفاده از دانش D_s و T_s بهبود می‌دهد به طوری که $D_s \neq D_t$ یا $T_s \neq T_t$. یک حیطه $D = \{\chi , P(X)\}$ به صورت یک فضای ویژگی χ و توزیع احتمال حاشیه‌ای $P(X)$ تعریف می‌شود به طوری‌که $X = \{x_1 , \cdots x_n\} \in \chi$ است. شرط $D_s \neq D_t$ در صورتی برقرار است که $\chi_s \neq \chi_t$ یا $P_s(X) \neq P_t(X)$ [63].

یک یادگیرنده می‌تواند اشکال متنوع دانش در مورد یک کار از قبیل بازنمایی‌های ویژگی یا پارامترهای مفید برای یادگیری مدل را از عامل دیگر فراگیرد. یادگیری انتقالی به یادگیری تقلیدی و کاربردهای رباتیکی مرتبط است چون به دست آوردن نمونه‌ها دشوار و پرهزینه است. بهره بردن از دانش کاری که قبل از این در آن سرمایه‌گذاری کرده تا آن را یاد بگیریم می‌تواند کارآمد و نتیجه‌بخش باشد.

سیاستی که در یک کار یاد گرفته شده، می‌تواند برای نصیحت (آموزش) یک یادگیرنده در کار دیگری که شباهت‌هایی با کار اول دارد استفاده شود. در مرجع [64]، این رویکرد بر روی دو کار شبیه‌ساز فوتبال روبوکاپ پیاده‌سازی شده است. کار اول دور نگه داشتن توپ از تیم دیگر است و کار دوم گل زدن است. واضح است که مهارت‌هایی که برای اجرای کار اول یاد گرفته شده‌اند می‌توانند در کار دوم نیز استفاده شوند. در این مورد، نصیحت به صورت یک قانون که در مورد حالت و یک یا تعداد بیش‌تری عمل است فرموله شده است. برای ایجاد نصیحت، سیاست مربوط به کار اول با استفاده از یادگیری تقویتی یاد گرفته شده است. سپس سیاست یاد گرفته شده، توسط یک کاربر (برای پرهیز از اختلافات در فضاهای حالت و عمل) به فرم نصیحتی نگاشت شده که برای آغاز سیاست مربوط به کار دوم استفاده می‌شود. بعد از دریافت نصیحت، یادگیرنده به اصلاح

[1] Transfer learning
[2] Domain

سیاست از طریق یادگیری تقویتی ادامه داده و در صورتی که از طریق تجربه اثبات کند که نصیحت داده شده نادقیق یا نامربوط است می‌تواند آن را تغییر داده یا نادیده بگیرد.

غالبا در یادگیری انتقالی، ورودی انسان برای نگاشت دانش از یک حیطه به حیطه دیگر نیاز است. اما در برخی موارد، رویه نگاشت می‌تواند خودکار شود[65]. برای مثال در مرجع [66]، یک تابع نگاشت برای انجام بازی ارائه شده است. این تابع به صورت خودکار بین حیطه‌های مختلف نگاشت کرده تا از تجربه قبلی آموخته شود. عامل قادر بوده که بازی‌هایی که قبلا انجام شده و به کار فعلی مرتبط هستند را شناسایی کند. ممکن است عامل قبلا همان بازی یا مشابهش را انجام داده باشد و قادر است که یک کار مبدا مناسب را انتخاب کرده تا از آن بیاموزد بدون اینکه آن کار به طور واضح معین شده باشد. آزمایشات نشان می‌دهند که رویکرد یادگیری انتقالی سرعت یادگیری بازی از طریق یادگیری تقویتی را (در مقایسه با یادگیری بدون دانش قبلی) افزایش داده و بعد از تکمیل تکرارهای یادگیری به عملکرد بهتری دست می‌یابد. همچنین نتایج حاکی از آنند که فایده استفاده از یادگیری انتقالی وابسته به تعداد نمونه‌های آموزشی است که از کارهای مبدا منتقل می‌شوند. حتی اگر عامل با انتقال منفی مواجه شود[63] برای مثال بر اثر بیش برازش به کار مبدا، می‌تواند از طریق یادگیری از تجربه و تصحیح مدلش در کار فعلی جبران (بازیابی) کرده تا در یک زمان مناسب همگرا شود[66].

در مرجع [67]، شکل‌دهی پاداش و یادگیری انتقالی ترکیب شده تا انواع کارهای محک (معیار)[1] یاد گرفته شوند. از آنجایی که شکل‌دهی پاداش متکی به دانش قبلی است تا بر روی تابع پاداش تاثیر بگذارد، یادگیری انتقالی می‌تواند از یک سیاست آموخته شده برای یک کار سود برده و شکل‌دهی پاداش را برای یک کار مشابه انجام دهد. در مرجع [67]، یادگیری انتقالی از یک نسخه ساده مسئله به نسخه پیچیده‌تر آن اعمال شده است (برای مثال از ماشین کوهنوردی دوبعدی به سه‌بعدی و از بازی ماریو بدون دشمن به یک بازی با دشمن).

- **یادگیری کارآموزی[2]**

در خیلی از کاربردهای هوش مصنوعی از قبیل بازی‌ها یا کارهای پیچیده رباتیک، اندازه‌گیری (سنجیدن) موفقیت یک عمل دشوار است. در این‌گونه موارد، نمونه‌های نمایش داده شده می‌توانند به عنوان یک قالب برای

[1] Benchmark
[2] Apprenticeship

عملکرد مورد نظر استفاده شوند. در مرجع [43]، یادگیری کارآموزی (یا یادگیری تقویتی معکوس) پیشنهاد شده تا زمانی که تابع پاداش صریحی در دسترس نباشد یک سیاست آموخته شده را بهبود دهد.

در چنین کاربردهایی، هدف تقلید رفتار مربی‌ها (انسان) به این شرط است که مربی در حال بهینه‌سازی یک تابع ناشناخته باشد.

یادگیری تقویتی معکوس (IRL) از نمونه‌های آموزشی استفاده کرده تا تابع پاداشی که توسط خبره بهینه می‌شود را یاد گرفته و از آن برای بهبود مدل آموزش داده شده استفاده کند.

از همین رو، یادگیری تقویتی معکوس عملکردی مشابه خبره به دست می‌آورد. بدون هیچ تابع پاداشی، عامل به صورت یک MDP/R (S,A,T) مدل می‌شود. در عوض، بعد از اینکه امید ریاضی‌های ویژگی μ_E از نمایش‌های خبره‌ها استنتاج شدند سیاست مدل می‌شود. با فرض n تراژکتوری $\{s_0^{(i)}, s_1^{(i)}, \cdots\}_{i=1}^{m}$ ، تخمین تجربی برای امید ریاضی ویژگی سیاست خبره $\mu_E = \mu(\prod_E)$ به صورت معادله(۲-۱) مشخص می‌شود

$$\hat{\mu}_E = \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{\infty}\gamma^t\phi(s_t^{(i)}) \qquad \textbf{معادله(۲-۱)}$$

که در آن γ یک ضریب کاهش و $\phi(s_t^{(i)})$ بردار ویژگی در زمان t از نمایش i است. هدف الگوریتم یادگیری تقویتی، پیدا کردن یک سیاست $\bar{\pi}$ است به طوری که $\|\mu(\bar{\pi}) - \mu_E\|_2 \leq \epsilon$ که در آن $\mu(\bar{\pi})$ امید ریاضی سیاست است[43].

در مرجع [48]، یک رویکرد بیشینه آنتروپی برای یادگیری تقویتی معکوس به کار گرفته شده تا ابهام را کاهش دهد. ابهام در کارهای یادگیری تقویتی معکوس به وجود می‌آید چون خیلی از توابع پاداش می‌توانند با سیاست یکسان بهینه شوند. این امر هنگام یادگیری تابع پاداش، یک مشکل را نمایان می‌کند مخصوصا زمانی که نمایش‌ها ناقص و معیوب باشند. روش پیشنهادی بر روی کار یادگیری انتخاب‌های مسیر راننده نشان داده شده است که در آن ممکن است نمایش‌ها زیربهینه[1] و غیرقطعی باشند. این رویکرد در مرجع [68] به یک چارچوب

[1] Suboptimal

کاری یادگیری عمیق بسط داده شده است. توابع هدف بیشینه آنتروپی، یادگیری مستقیم وزن‌های شبکه و در نتیجه استفاده از شبکه‌های عمیق که با گرادیان تصادفی نزولی آموزش داده شده‌اند را امکان‌پذیر می‌کنند[68].

معماری عمیق بیش‌تر بسط داده شده تا به جای استفاده از ویژگی‌های از پیش استخراج شده، ویژگی‌ها از طریق لایه‌های کانولوشنی یاد گرفته شوند. این یک گام مهم در مسیر خودکار کردن فرآیند یادگیری است. یکی از چالش‌های اصلی در یادگیری تقویتی از راه آزمون و خطا نیاز به دانش انسان در طراحی بازنمایی‌های ویژگی و توابع هدف است[14]. با استفاده از یادگیری عمیق برای یادگیری خودکار بازنمایی‌های ویژگی و با استفاده از یادگیری تقویتی معکوس برای استنتاج توابع پاداش از نمایش‌ها نیاز به ورودی و طراحی انسان کمینه می‌شود. الگوی یادگیری تقویتی معکوس مزیتی که بر اشکال دیگر یادگیری از نمایش‌ها دارد این است که تابع هزینه کار از محیط جدا شده است. چون هدف نمایش‌ها یاد گرفته شده‌اند نه خود نمایش‌ها، نیاز نیست که نمایش‌دهنده و یادگیرنده، اسکلت و محیط پیرامون یکسان داشته باشند، بنابراین چالش‌هایی از قبیل مسئله تناسب را کاهش می‌دهد. بنابراین، تهیه نمایش‌هایی که کلی بوده و برای یک ربات یا محیط خاص ساخته نشده‌اند آسان‌تر است.

علاوه بر این، یادگیری تقویتی معکوس می‌تواند به جای یادگیری تقویتی سنتی به کار گرفته شود اگرچه تابع پاداشی وجود داشته باشد (به این شرط که نمایش‌ها در درسترس باشند). برای مثال در مرجع [69]، از یادگیری کارآموزی استفاده شده تا یک تابع پاداش از نمایش‌های خبره در بازی ماریو استنتاج شود. در حالی که اهداف در یک بازی مثل ماریو می‌توانند از قبل تعریف شوند (از قبیل امتیاز کشتن دشمنان و جمع‌آوری سکه‌ها یا زمان کامل کردن سطح)، معلوم نیست که چگونه یک کاربر خبره این اهداف را اولویت‌بندی می‌کند. بنابراین در تلاش برای تقلید رفتار انسان، تابع پاداشی که از نمایش‌ها استخراج شده به تابع پاداشی که به‌طور دستی طراحی شده ترجیح داده می‌شود.

- **یادگیری فعال**

یادگیری فعال الگویی است که در آن، مدل قادر است از یک خبره برای جواب بهینه به یک حالت یا وضعیت معلوم سوال کرده و از این نمونه‌های فعال برای بهبود سیاستش استفاده کند.

یک دسته‌بند $h(x)$ روی یک مجموعه داده $D_K(x^{(i)}, y^{(i)})$ آموزش داده شده و برای پیش‌بینی برچسب‌های یک مجموعه داده بدون برچسب $D_U(x^{(i)})$ استفاده شده است. یک زیرمجموعه $D_C(x^{(i)}) \subset D_U$ توسط یادگیرنده انتخاب شده تا در مورد برچسب‌های صحیح آنها $y^{*(i)}$ از خبره سوال کند. از نمونه‌های فعال $D_C(x^{(i)}, y^{*(i)})$ برای آموزش $h(x)$ با هدف کمینه کردن n استفاده شده است: n تعداد نمونه‌های در D_C است.

یادگیری فعال روشی مفید برای تطبیق مدل به وضعیت‌هایی است که در نمونه‌های آموزشی اصلی پوشش داده نشده‌اند. از آنجایی که یادگیری تقلیدی دربرگیرنده تقلید تراژکتوری کامل یک حرکت است ممکن است خطا در هر مرحله‌ای از اجرا رخ دهد. ساخت مجموعه‌های آموزشی منفعل[1] که می‌توانند از این مشکل پرهیز کنند خیلی دشوار است.

یک رویکرد برای تصمیم‌گیری در مورد زمان سوال کردن از خبره، استفاده از تخمین‌های اطمینان است تا بخش‌هایی از مدل آموخته شده که نیاز به بهبود دارند شناسایی شوند. هنگام اجرای اعمال آموخته شده، اطمینان در این پیش‌بینی تخمین زده شده و یادگیرنده می‌تواند تصمیم بگیرد که درخواست نمایش‌های جدید برای بهبود این قسمت از کاربرد را داده یا در صورتی که اطمینان کافی باشد از سیاست فعلی استفاده کند. با تناوب کردن بین اجرای سیاست و به‌روز رسانی آن با نمونه‌های جدید، کم‌کم یادگیرنده اطمینان پیدا کرده و یک سیاست کلی به دست آورده که بعد از مدتی، دیگر نیازی به درخواست برای به‌روز رسانی‌های بیش‌تر ندارد. در مرجع [20]، از بهبود سیاست مبتنی بر اطمینان برای یادگیری ناوبری استفاده شده و در مرجع [34]، از همین روش برای یک کار مرتب‌سازی کلان[2] استفاده شده است.

در مرجع [70]، یادگیری فعال معرفی شده تا عامل را قادر ساخته که در هر گامی در تراژکتوری، با در نظر گرفتن همه گام‌های قبلی، از خبره‌ها سوال کند. این مسئله به یادگیری فعال i.i.d کاهش یافته (تبدیل شده) و استدلال شده که به طور قابل‌توجهی تعداد نمایش‌های مورد نیاز را کاهش می‌دهد.

در مرجع [71]، یادگیری فعال در کارهای مشارکتی انسان- ربات مطرح شده است. انسان و ربات به صورت فیزیکی تعامل کرده تا در یک کار نامتقارن (به عبارت دیگر، انسان و ربات نقش‌های متفاوتی دارند) به یک

[1] Passive
[2] Macro sorting

هدف مشترک برسند. یادگیری فعال بین دوره[1]های تعامل رخ داده و انسان از طریق یک واسط کاربری گرافیکی (GUI) به ربات بازخورد می‌دهد. بازخورد ضبط شده و به پایگاه داده نمونه‌های آموزشی اضافه شده که برای آموزش مدل ترکیبی گوسی که اعمال ربات را کنترل می‌کند استفاده می‌شود. تعامل فیزیکی بین انسان و ربات منجر به رفتار وابسته متقابل می‌شود. بنابراین با هر تکرار تعامل، اعمال به‌هم پیوسته دو طرف به تراژکتوری حرکت هموارتری همگرا می‌شود. تجزیه و تحلیل کیفی آزمایشات نشان می‌دهد که در صورتی که انسان به اعمال ربات وفق یابد تعامل بین آنها می‌تواند بهتر شود و در صورتی که ربات هم به نوبه خود با هر نوبت تعامل به عمل انسان وفق یابد تعامل به طور قابل توجهی بهتر می‌شود.

در مرجع [72]، به‌جای اینکه یادگیرنده یک سوال بفرستد مربی اصلاحات را آغاز می‌کند. در حین اجرای کار توسط ربات، مربی رفتار یادگیرنده را مشاهده کرده و به طور جنبشی[2] موقعیت مفاصل ربات را تصحیح می‌کند. یادگیرنده حرکتش که به کمک مربی انجام داده را از طریق سنسورهایش دنبال کرده و از این تراژکتوری‌ها برای اصلاح مدل استفاده می‌کند، مدلی که به تدریج آموخته شده تا امکان اضافه شدن نمایش‌ها در هر نقطه‌ای را فراهم کند.

- **پیش‌بینی‌های ساختارمند**

به سبکی مشابه، DAGGER [23] اجتماع نمونه را به کار گرفته تا برای وضعیت‌های دیده نشده تعمیم داده شود. اما اساسا این رویکرد متفاوت است. DAGGER مسئله یادگیری تقلیدی را به صورت یک مسئله پیش‌بینی ساختارمند که از مرجع [73] الهام گرفته شده فرموله می‌کند. یک عمل به صورت دنباله‌ای از پیش‌بینی‌های وابسته تلقی می‌شود. چون یک عمل وابسته به حالت قبلی است یک خطا منجر به یک حالت دیده نشده می‌شود که یادگیرنده نمی‌تواند از آن بازیابی کند که این امر منجر به خطاهای مرکب می‌شود. DAGGER نشان می‌دهد که تجمیع نمونه‌هایی که خطاهای یادگیری اولیه را پوشش می‌دهند لازم و کافی است. بنابراین، یک رویکرد تکراری پیشنهاد شده که از یک سیاست بهینه استفاده می‌کند تا هر مرحله از اعمالی را که با استفاده از سیاست فعلی پیش‌بینی شده تصحیح کند، بنابراین نمونه‌هایی اصلاح شده ایجاد می‌شوند

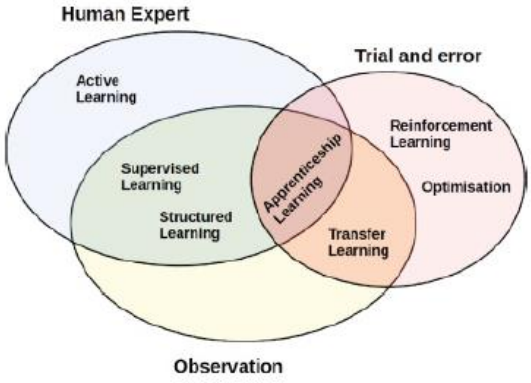
[1] round
[2] Kinesthetically

که برای به‌روز رسانی سیاست به کار می‌روند. همان‌طور که الگوریتم تکرار می‌شود استفاده از سیاست بهینه کم می‌شود تا زمانی که سیاست یاد گرفته شده به عنوان مدل نهایی استفاده می‌شود.

در مرجع [74]، یک الگوریتم به نام SIMILE پیشنهاد شده که محدودیت‌های روش‌های معرفی شده در مراجع [23] و [73] را از طریق تولید یک سیاست پایا که به تجمیع داده نیاز ندارد کم کرده است. SIMILE، نیاز به خبره برای تامین عمل در هر مرحله از تراژکتوری را از طریق فراهم کردن "بازخورد خبره مجازی" کم کرده که همواری تراژکتوری تصحیح شده را کنترل کرده و به اعمال خبره همگرا می‌شود.

در نظر گرفتن اعمال قبلی در فرآیند یادگیری یک نکته مهم در یادگیری تقلیدی است چون تعداد زیادی از کاربردها به اجرای تراژکتوری‌هایی از مقدمات حرکتی وابسته متکی هستند. یک روش کلی برای گنجاندن حافظه در یادگیری استفاده از RNN هاست[75]. RNN ها میان لایه‌های مخفی یک حلقه بازخورد ایجاد کرده تا خروجی‌های قبلی شبکه را در نظر بگیرند و از همین رو برای کارهای با تراژکتوری‌های ساختارمند مناسب هستند[76].

در نهایت، شکل ۲-۳ یک نمودار ون را نشان می‌دهد که منابع داده‌ای که توسط روش‌های یادگیری مختلف به کار گرفته شده‌اند را به طور خلاصه شرح داده است. یک عامل می‌تواند از نمایش‌های اختصاصی مربی و با مشاهده اعمال عامل‌های دیگر یا از طریق آزمون وخطا یاد بگیرد. یادگیری فعال به یک اوراکل اختصاصی نیاز دارد که بتواند از آن درخواست نمایش کند، در حالی که روش‌های دیگر که از نمایش‌ها بهره می‌برند می‌توانند آنها را از یک خبره اختصاصی یا با مشاهده رفتار مورد نیاز از عامل‌های دیگر به دست بیاورند. روش‌های یادگیری تقویتی و بهینه‌سازی با آزمون و خطا یاد گرفته و از نمایش‌ها استفاده نمی‌کنند. یادگیری انتقالی از تجربه کارهای قدیمی یا از دانش عامل‌های دیگر استفاده کرده تا یک سیاست جدید یاد بگیرد. یادگیری کارآموزی از نمایش‌های یک خبره یا مشاهده استفاده کرده تا یک تابع پاداش یاد بگیرد. سپس یک سیاست که تابع پاداش را بهینه می‌کند می‌تواند از طریق تجربه آموخته شود.



شکل ۲-۳ روش‌های یادگیری از منابع مختلف.[15]

۲-۱-۲- روش‌های یادگیری تقویتی

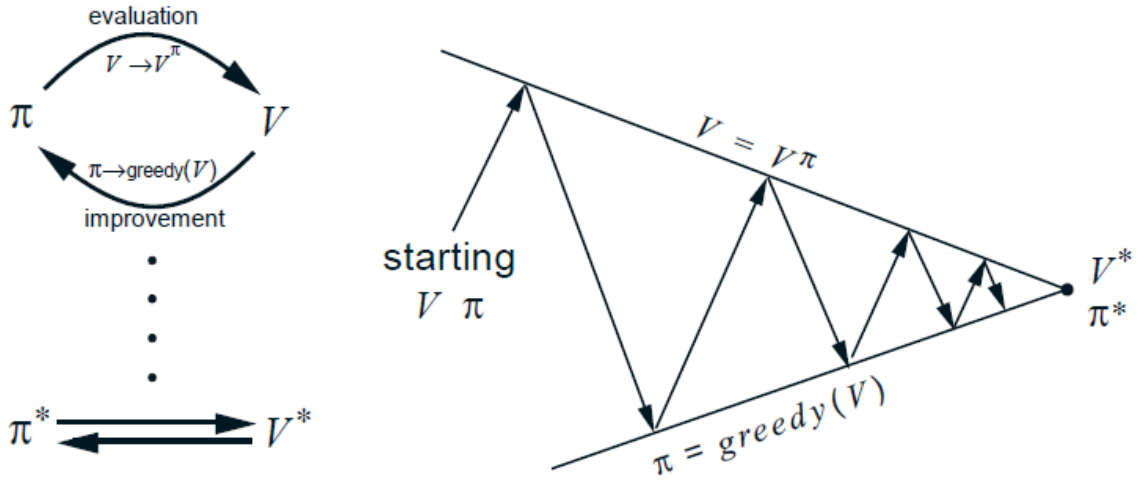
عمل کردن در شرایطی که عدم قطعیت وجود دارد نقش مهمی در مطالعه مسائل تصمیم‌گیری دنباله‌ای[1] دارد و در رشته‌های مختلفی از قبیل برنامه‌ریزی نظری تصمیم‌گیری، تحقیق در عملیات، یادگیری تقویتی و اقتصاد مورد بررسی قرار گرفته است. این هدف که با روشی معقول و به صورت بهینه عمل کرد، مقوله اصلی در بسیاری از رشته‌های کمابیش مرتبط به هوش مصنوعی است. بخش زیادی از مسائلی که تصمیم‌گیری دنباله‌ای بهینه را دربرمی‌گیرند می‌توانند به صورت فرآیندهای تصمیم مارکف (MDP)[77] بیان شوند. در این مدل، یک محیط به صورت مجموعه‌ای از حالت‌ها و عمل‌هایی که برای کنترل حالت سیستم انجام می‌شوند مدل می‌شود. هدف، کنترل سیستم به نحوی است که بهترین عملکرد حاصل شود. خیلی از مسائل از قبیل مسائل برنامه‌ریزی (تصادفی)، یادگیری کنترل ربات و مسائل انجام بازی در قالب یک MDP مدل شوند.

۱-۲-۱-۲- حل فرآیندهای تصادفی مارکف

حل یک MDP معلوم به معنای محاسبه سیاست بهینه π^* است که به هر حالتی در فضای حالت بهترین عمل را انتساب می‌کند. برای دست‌یابی به این هدف، تعداد زیادی روش‌های برنامه‌نویسی پویا و یادگیری تقویتی وجود دارد. فرق اصلی بین این روش‌ها این است که آیا دانش (کامل) در مورد تابع انتقال T و تابع

[1] Sequential decision making

پاداش R را فرض می‌کنند یا خیر. در دو بخش بعدی به طور مختصر برخی از الگوریتم‌های پایه را شرح می‌دهیم. فرم کلی همه این الگوریتم‌ها شامل دو فرآیند است (به شکل ۲-۴ رجوع شود)، یکی تخمین مقادیر برای حالت‌ها و عمل‌هاست و دیگری بهبود سیاست‌هاست.



(ب) تکرار سیاست تعمیم‌یافته (الف) دو فرآیند

شکل ۲-۴ فرم کلی الگوریتم‌های حل فرآیندهای تصمیم مارکف. (الف) همگرایی تدریجی تابع مقدار و سیاست به نسخه‌های بهینه. (ب) الگوریتم‌های معرفی شده در بخش ۱-۲-۱-۲- می‌توانند به صورت مقداردهی اولیه حلقه تکرار سیاست تعمیم‌یافته (GPI) دیده شوند. گام ارزیابی سیاست، عملکرد سیاست V^{π} را تخمین می‌زند. گام بهبود سیاست، سیاست π را براساس تخمین‌های در V^{π} بهبود می‌دهد.[78]

- **برنامه‌نویسی پویا: تکنیک‌های حل مبتنی بر مدل**

تعداد زیادی تکنیک حل مبتنی بر مدل که به برنامه‌نویسی پویا (DP) معروف هستند وجود دارد. ایده اصلی، محاسبه توابع مقدار و استنتاج سیاست‌های بهینه (که معمولا با یک روش تکراری صورت می‌پذیرد) از آنهاست. دو روش DP کلاسیک، تکرار سیاست و تکرار مقدار هستند. آنها را به ترتیب به طور مختصر شرح خواهیم داد.

روش تکرار سیاست (PI) [79] با یک سیاست اولیه دلخواه π_0 آغاز می‌شود. سپس دنباله‌ای از تکرارها در آن می‌آیند که در آنها سیاست فعلی بعد از اینکه بهبود داده شد ارزیابی می‌شود. گام اول، گام ارزیابی سیاست، V^{π_k} را با استفاده مکرر از معادله(۲-۲) محاسبه می‌کند.

$$V_{n+1}(s) := R(s) + \gamma \max_{a \in A} [\sum_{s' \in S} T(s,a,s') V_n(s')] \qquad \textbf{معادله(۲-۲)}$$

گام دوم، گام بهبود سیاست، با استفاده از V^{π_k}، π_{k+1} را از π_k محاسبه می‌کند. برای هر حالت، با استفاده از معادله(۲-۳)، عمل بهینه معین می‌شود.

$$\pi^*(s) := \underset{a}{argmax} [R(s) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s')] \qquad \textbf{معادله(۲-۳)}$$

در صورتی که برای همه حالت‌ها $\pi_{k+1}(s) = \pi_k(s)$ باشد، سیاست پایدار است و الگوریتم تکرار سیاست می‌تواند متوقف شود. شکل کامل الگوریتم در الگوریتم ۲-۱ آورده شده است. هنگامی که MDP متناهی است یا به عبارت دیگر مجموعه‌های حالت و عمل متناهی هستند، تکرار سیاست بعد از تعداد متناهی تکرار همگرا می‌شود. هر سیاست π_{k+1} اکیدا از سیاست π_k بهتر است مگر اینکه در موردی که $\pi_k = \pi^*$ است که در این‌صورت الگوریتم متوقف می‌شود. و چون برای یک MDP متناهی تعداد سیاست‌های مختلف متناهی است، تکرار سیاست در زمان متناهی همگرا می‌شود. در عمل، معمولا بعد از تعداد کمی تکرار همگرا می‌شود. اگرچه تکرار سیاست، سیاست بهینه برای یک MDP معلوم را در زمان متناهی محاسبه می‌کند اما نسبتا ناکاراست. به‌ویژه، گام ارزیابی سیاست از لحاظ محاسباتی پرهزینه است. توابع مقدار برای همه سیاست‌های میانی $\pi_0, \cdots, \pi_k, \cdots, \pi^*$ محاسبه می‌شوند که این به‌ازای هر تکرار شامل چندین جابجایی در فضای حالت کامل است.

Require: $V(s) \in \mathbb{R}$ and $\pi(s) \in A(s)$ arbitrarily for all $s \in S$
  {POLICY EVALUATION}
  repeat
    $\Delta := 0$
    for each $s \in S$ do
      $v := V(s)$
      $V(s) := R(s) + \gamma . \sum_{s'} T(s, \pi(s), s') V(s')$
      $\Delta := \max(\Delta, |v - V(s)|)$
  until $\Delta < \sigma$
  {POLICY IMPROVEMENT}
  policy-stable := true
  for each $s \in S$ do
    $b := \pi(s)$
    $\pi(s) := R(s) + \gamma \underset{a}{\operatorname{argmax}} [\sum_{s'} T(s, a, s') V(s')]$
    if $b \neq \pi(s)$ then policy-stable := false
  if policy-stable then stop; else go to POLICY EVALUATION

الگوریتم ۲-۱ تکرار سیاست[79].

در مقابل، تکرار مقدار (VI) تا همگرایی، هر تابع مقدار را محاسبه نمی‌کند، بلکه به‌روز رسانی‌های مورد نیاز را در حین کار انجام می‌دهد. در اصل، نسخه مختصر شده گام ارزیابی سیاست را با گام بهبود سیاست ترکیب می‌کند که اساسا معادله(۲-۴) است که به یک قانون به‌روز رسانی تبدیل شده است:

$$V^{\pi}(s) := R(s) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^{\pi}(s') \qquad \textbf{معادله(۲-۴)}$$

$$V_{n+1}(s) := R(s) + \underset{a}{max} [\sum_{s'} T(s, a, s') V_t(s')] \qquad \textbf{معادله(۲-۵)}$$

$$:= \underset{a}{max}\, Q_{t+1}(s, a). \qquad \textbf{معادله(۲-۶)}$$

براساس معادلات معادله(**۲-۵**) و معادله(**۲-۶**) ، الگوریتم VI (به الگوریتم ۲-۲ رجوع شود) می‌تواند به صورتی که در ادامه گفته می‌شود بیان شود: با شروع از یک تابع مقدار V_0 برای همه حالت‌ها، به طور مکرر مقدار هر حالت مطابق با معادله(۲-۷) به‌روز رسانی شده تا توابع مقدار بعدی $V_t(t = 1,2,3, \cdots)$ به دست آیند.

تضمین می‌شود که VI در حد به V^* همگرا می‌شود، به عبارت دیگر، معادله بهینگی Bellman برای هر حالت صدق می‌کند. برای همه حالت‌ها $s \in S$، یک سیاست قطعی π می‌تواند محاسبه شود.

$$\pi^*(s) := \max_a [R(s) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s')] \quad \textbf{معادله(۲-۷)}$$

Require: initialize V arbitrarily (e.g. $V(s) := 0, \forall s \in S$)
  repeat
    $\Delta := 0$
    for each $s \in S$ do
      $v := V(s)$
      for each $a \in A(s)$ do
        $Q(s,a) := R(s) + \gamma . \sum_{s'} T(s,a,s') V(s')$
      $V(s) := \max_a Q(s,a)$
    $\Delta := \max(\Delta, |v - V(s)|)$
  until $\Delta < \sigma$

الگوریتم ۲-۲ تکرار مقدار[80].

تکرار سیاست تغییریافته (MPI) [81] یک روش بینابینی بین تکرار سیاست و تکرار مقدار را اتخاذ کرده است. تکرار مقدار اساسا هر دو گام تخمین سیاست و بهبود سیاست را در یک به‌روز رسانی ترکیب می‌کند. از طرفی دیگر، تکرار سیاست دو گام را جدا کرده و هر دو آنها را در حد محاسبه می‌کند. MPI دو گام جدا را نگه می‌دارد، اما هر دو گام لزوما در حد محاسبه نمی‌شوند. در اینجا دیدگاه کلیدی این است که برای بهبود سیاست، به یک سیاست به طور دقیق ارزیابی شده نیاز نیست تا آن را بهبود داد. برای مثال، گام تخمین سیاست می‌تواند تقریبی باشد که بعد از آن یک گام بهبود سیاست می‌تواند بیاید. به طور کلی، هر دو گام می‌توانند کاملا مستقل و با روش‌های مختلف انجام شوند. برای مثال، به جای اعمال مکرر قانون به‌روز رسانی Bellman از معادله(۲-۶) ، گام تخمین سیاست می‌تواند با استفاده از یک رویه نمونه‌برداری از قبیل تخمین Monte Carlo [78] انجام شود. تکرار سیاست و تکرار مقدار، هر دو از موارد افراطی تکرار سیاست تغییر یافته هستند.

تعداد زیادی از نسخه‌های دیگر تکرار سیاست و مقدار پیشنهاد شده‌اند. یک دسته بزرگ تحت عنوان روش‌های ناهمگام از لحاظ ترتیبی - و همچنین بخش‌هایی خاص در آن ترتیب - که به‌روز رسانی‌ها انجام

می‌شوند با VI و PI استاندارد تفاوت دارند. روش‌های ناهمگام مبتنی بر تخمین مقدار، مقادیر حالت‌ها یا جفت حالت-عمل را نه در یک جابجایی در سراسر فضای حالت بلکه در عوض، مقادیر را در مناطق مخصوصا هدف قرار داده شده به‌روز رسانی می‌کنند برای مثال، در امتداد تراژکتوری‌های شبیه‌سازی شده که در رویکردهای یادگیری تقویتی رایج است. روش‌های تکرار مقدار ناهمگام به‌روز رسانی‌ها را با یک ترتیب غیرثابتی انجام می‌دهند. تکنیک برنامه‌نویسی پویای بلادرنگ که توسط بارتو و همکاران[82] ارائه شده، جستجوی پیشرو را با برنامه‌نویسی پویا ترکیب می‌کند. برای بسیاری از این روش‌های ناهمگام، همگرایی می‌تواند تحت شرایط کلی از قبیل اینکه اغلب همه حالت‌ها باید به طور نامتناهی به‌روز رسانی شوند تضمین شود.

- **یادگیری تقویتی: تکنیک‌های حل فارغ از مدل**

در مقایسه با الگوریتم‌هایی که در بخش قبل در موردشان بحث شد، روش‌های فارغ از مدل به در دسترس بودن مدل‌های انتقال و پاداش از پیش معلوم، به طور مختصر مدل MDP ، متکی نیستند. نبود یک مدل این نیاز را ایجاد می‌کند که باید از MDP نمونه‌برداری کرد تا دانش آماری در مورد مدل نامعلوم جمع‌آوری شود. خیلی از تکنیک‌های یادگیری تقویتی فارغ از مدل وجود دارند که محیط را با انجام عمل‌ها مورد بررسی قرار می‌دهند و از این راه، مشابه تکنیک‌های مبتنی بر مدل همان نوع توابع مقدار حالت و حالت-عمل را تخمین می‌زنند. یکی از پایه‌ای‌ترین و محبوب‌ترین روش‌ها برای تخمین توابع Q - مقدار به شیوه فارغ از مدل، الگوریتم Q-learning است که توسط واتکینز و دایان در سال ۱۹۹۲ ارائه شده است (الگوریتم ۲-۳ طرح کلی آن را نشان می‌دهد).

Require: discount factor γ, learning parameter α
  initialize Q arbitrarily (e.g. $Q(s,a) := 0, \forall s \in S, \forall a \in A$)
  for each episode do
    s is initialized as the *starting* state
    repeat
      choose an action $a \in A(s)$ based on an exploration strategy
      perform action a
      observe the new state s' and received reward r

$$Q(s,a) := Q(s,a) + \alpha \left[R(s) + \gamma . \max_{a' \in A(s')} Q(s',a') - Q(s,a) \right]$$

      $s := s'$

الگوریتم ۲-۳ Q-Learning [2].

یک جنبه مهم الگوریتم‌های فارغ از مدل این است که نیاز به اکتشاف وجود دارد. چون مدل ناشناخته (نامعلوم) است، یادگیرنده باید عمل‌های مختلفی را امتحان کرده تا نتایجشان را ببیند. یک الگوریتم یادگیری باید بین اکتشاف و بهره‌برداری توازن برقرار کند، به عبارت دیگر برای اینکه پاداش زیادی به دست آورد یادگیرنده باید دانش فعلیش در مورد عمل‌های خوب را به کار گیرد، اگرچه گاهی اوقات باید عمل‌های مختلف را امتحان کرده

تا محیط را برای عمل‌های احتمالا بهتر مورد بررسی قرار دهد. پایه‌ای‌ترین استراتژی اکتشاف سیاست ϵ-greedy است، به عبارت دیگر، یادگیرنده بهترین عمل فعلیش را با احتمال $(1-\epsilon)$ و عمل دیگر (که به طور تصادفی انتخاب شده است) را با احتمال ϵ برمی‌گزیند.

الگوریتم‌هایی از قبیل Q-learning وگونه‌هایی از قبیل SARSA [83] نیز، در صورتی که همه مقادیر مجزا در یک جدول پشتیبان ذخیره شده باشند همگراییشان به تابع مقدار (و سیاست) بهینه تضمین شده است. به دلیل تجرید[1]، موقعیت‌هایی پیش می‌آیند که این نتایج همگرایی برقرار نیستند.

الگوریتم‌هایی از قبیل Q-learning از خود راه‌اندازی[2] برای محاسبه مقادیر استفاده می‌کنند: تخمین یک مقدار با مقدار تخمین‌زده شده حالت بعدی به‌روز رسانی می‌شود. الگوریتم‌های دیگری که از تخمین‌های بدون تبعیض‌تر استفاده می‌کنند تکنیک‌های Monte Carlo هستند. این روش‌ها، تعداد تکرار انتقال‌ها و پاداش‌ها را نگه داشته و مقادیرشان را بر مبنای این تخمین‌ها قرار می‌دهند.

روش‌های غیرمستقیم توازنی بین یادگیری مبتنی بر مدل و یادگیری فارغ از مدل برقرار می‌کنند. این روش‌ها اساسا فارغ از مدل هستند، اما به طور موازی با یادگیری تقویتی فارغ از مدل، یک مدل انتقال و پاداش یاد می‌گیرند و از این مدل برای یادگیری کاراتر تابع مقدار استفاده می‌کنند. نمونه‌ای از این روش مدل Dyna [84] است که توسط ساتن در سال ۱۹۹۱ ارائه شده است. روش دیگری که در آن سودمند بودن یادگیری مدل اثبات شده است پاک‌سازی اولویت‌بندی شده[3][85] است.

۲-۲- روش‌های یادگیری تقویتی با جستجوی سیاست

جستجوی سیاست زیررشته‌ای در یادگیری تقویتی است که روی پیدا کردن پارامترهای خوب برای پارامتربندی یک سیاست معلوم تمرکز می‌کند. این روش مناسب رباتیک است چون می‌تواند از عهده فضاهای عمل و حالت با ابعاد بالا که یکی از چالش‌های اصلی در یادگیری ربات است برآید.

[1] Abstraction
[2] bootstrapping
[3] prioritized sweeping

جستجوی سیاست فارغ از مدل، یک رویکرد کلی برای یادگیری سیاست‌های مبتنی بر تراژکتوری‌های نمونه‌برداری شده است. روش‌های فارغ از مدل را براساس استراتژی‌های ارزیابی سیاست، به‌روز رسانی سیاست واکتشاف آنها دسته‌بندی کرده و یک دید یکپارچه به الگوریتم‌های موجود ارائه می‌دهیم. غالبا یادگیری یک سیاست آسان‌تر از یادگیری یک مدل پیشرو دقیق است و از همین رو، روش‌های فارغ از مدل به کرات در عمل استفاده می‌شوند. اما برای هر تراژکتوری نمونه‌برداری شده تعامل با ربات ضروری است، امری که در عمل می‌تواند زمان‌بر و چالش‌برانگیز باشد.

جستجوی سیاست مبتنی بر مدل، این مشکل را ابتدا با یادگیری یک شبیه‌ساز دینامیک ربات از داده‌ها حل می‌کند. سپس، شبیه‌ساز تراژکتوری‌هایی تولید می‌کند که برای یادگیری سیاست استفاده می‌شوند.

۲-۲-۱- مقدمه

از ربات‌های نظافت‌کار خانه گرفته تا صندلی چرخ‌دارهای رباتیکی و ربات‌های عمومی حمل و نقل، تعداد و انواع ربات‌هایی که در زندگی همه‌روزه ما استفاده می‌شوند به سرعت در حال افزایش هستند. اکثرا کنترل‌کننده‌های این ربات‌ها توسط یک مهندس طراحی و تنظیم می‌شوند. برنامه‌نویسی ربات‌ها کاری خسته‌کننده است که به سال‌ها تجربه و درجه‌ای بالا از تخصص نیاز دارد. کنترل‌کننده‌های برنامه‌نویسی شده حاصله مبتنی بر فرض مدل‌های دقیق از رفتار ربات و محیط پیرامونش هستند. در نتیجه، زمانی که یک ربات باید با وضعیت‌های جدید تطبیق پیدا کرده یا زمانی که ربات/محیط نمی‌توانند به اندازه کافی دقیق مدل شوند هاردکد کردن[1] کنترل‌کننده‌ها محدودیت‌های خودش را دارد. از همین رو، بین ربات‌هایی که در حال حاضر از آنها استفاده می‌شود و ربات‌های کاملا خودمختار فاصله وجود دارد. در یادگیری ربات، برای حل یک کار رباتیکی از روش‌های یادگیری ماشین به منظور استخراج خودکار اطلاعات مرتبط از داده‌ها استفاده می‌شود. با استفاده از قدرت و انعطاف‌پذیری تکنیک‌های مدرن یادگیری ماشین، حوزه کنترل ربات بیش‌تر می‌تواند خودکار شود و فاصله تا ربات‌های خودمختار (برای مثال برای همکاری کلی در کارهای خانه، مراقبت از سالمندان و خدمات عمومی) می‌تواند به طور چشم‌گیری کاهش یابد.

[1] Hardcode

روش‌های جستجوی سیاست فارغ از مدل، از تعامل‌ها با ربات واقعی به منظور تولید تراژکتوری‌های نمونه استفاده می‌کنند. در حالی که نمونه‌برداری تراژکتوری‌ها در شبیه‌سازی کامپیوتر نسبتا سرراست و مستقیم است، معمولا هنگام کار با ربات‌ها تولید هر نمونه به سطحی از نظارت انسانی نیاز دارد. در نتیجه، تولید تراژکتوری با سیستم واقعی به طور قابل ملاحظه‌ای وقت‌گیرتر از کار با سیستم‌های شبیه‌سازی شده است. علاوه بر این، تعاملات ربات واقعی موجب فرسودگی ربات‌های غیرصنعتی می‌شود. اما با وجود تعداد نسبتا زیاد تعامل‌ها با ربات که برای جستجوی سیاست فارغ از مدل مورد نیاز است، غالبا یادگیری یک سیاست آسان‌تر از یادگیری مدل‌های پیشرو دقیق است و از همین رو، جستجوی سیاست فارغ از مدل در مقایسه با روش‌های مبتنی بر مدل به طور گسترده‌تر استفاده می‌شود.

روش‌های جستجوی سیاست مبتنی بر مدل سعی دارند که مشکل کارا نبودن از لحاظ نمونه را با استفاده از تراژکتوری‌های مشاهده شده به منظور یادگیری یک مدل پیشرو از دینامیک ربات و محیط پیرامونش حل کنند. سپس، این مدل پیشرو برای شبیه‌سازی‌های درونی محیط و دینامیک ربات استفاده می‌شود که مبتنی بر آن سیاست آموخته می‌شود. روش‌های جستجوی سیاست مبتنی بر مدل این پتانسیل را دارند که به تعامل‌های کم‌تری با ربات نیاز داشته باشند و به طور کارا به وضعیت‌های دیده نشده تعمیم یابند[86]. در حالی که ایده استفاده از مدل‌ها در زمینه یادگیری ربات از دهه ۱۹۸۰ شناخته شده است[87]، به خاطر وابستگی زیادش به کیفیت مدل‌های آموخته شده محدود شده است. در عمل مدل آموخته شده دقیق نیست، بلکه فقط یک تقریب کمابیش دقیق از دینامیک واقعی است. از آنجایی که سیاست آموخته شده ذاتا مبتنی بر شبیه‌سازی‌های درونی با مدل آموخته شده است بنابراین مدل‌های نادقیق می‌توانند منجر به استراتژی‌های کنترلی شوند که در برابر خطاهای مدل مقاوم نیستند. در بعضی از موارد، مدل‌های آموخته شده ممکن است از لحاظ فیزیکی غیرمحتمل بوده و شامل ضرایب اصطکاک منفی یا جرم‌های منفی باشند. اغلب اوقات، این آثار غیرمحتمل توسط الگوریتم جستجوی سیاست به کار گرفته شده که منجر به کیفیت پایین سیاست آموخته شده می‌شود. این اثر می‌تواند با استفاده از مدل‌هایی که خطاهای مدل را به طور صریح در نظر می‌گیرند کاهش یابد[3].

۲-۲-۲- جستجوی سیاست فارغ از مدل

روش‌های جستجوی سیاست فارغ از مدل (PS)، سیاست را مستقیما و مبتنی بر تراژکتوری‌های نمونه‌برداری شده $\tau^{[i]}$ به روز رسانی می‌کنند که i اندیس تراژکتوری را نشان می‌دهد و پاداش‌های آنی به دست آمده برای تراژکتوری‌ها $r_0^{[i]}, r_1^{[i]}, \cdots, r_T^{[i]}$ هستند. روش‌های جستجوی سیاست فارغ از مدل سعی می‌کنند که پارامترها (θ) را طوری به‌روز رسانی کنند که هنگام دنبال کردن سیاست جدید، تراژکتوری‌های با پاداش‌های بیش‌تر محتمل‌تر شوند و از همین‌رو، میانگین حاصل[1] افزایش پیدا می‌کند.

$$J_\theta = \mathbb{E}[R(\tau)|\theta] = \int R(\tau) p_\theta(\tau) d\tau \qquad \textbf{معادله(۲-۸)}$$

یادگیری سیاست غالبا آسان‌تر از یادگیری مدل ربات و محیط پیرامونش است و از همین رو، غالبا روش‌های جستجوی سیاست فارغ از مدل بیش‌تر از روش‌های جستجوی سیاست مبتنی بر مدل استفاده می‌شوند. رویکردهای جستجوی سیاست فارغ از مدل را مبتنی بر استراتژی‌های ارزیابی سیاستشان، به‌روز رسانیشان[88] و اکتشافشان[89] دسته‌بندی می‌کنیم.

repeat
  Explore: Generate trajectories $\tau^{[i]}$ using policy π_k
  Evaluate: Assess quality of trajectories or actions
  Update: Compute π_{k+1} given trajectories $\tau^{[i]}$ and evaluations
until Policy converges $\pi_{k+1} \approx \pi_k$

الگوریتم ۲-۴ جستجوی سیاست فارغ از مدل[90].

استراتژی اکتشاف معین می‌کند که چگونه تراژکتوری‌های جدید برای گام ارزیابی سیاست بعدی ایجاد شوند. استراتژی اکتشاف برای جستجوی سیاست فارغ از مدل کارا ضروری است چون به تنوع در تراژکتوری‌های تولید شده نیاز داریم تا به‌روز رسانی سیاست را معین کنیم اما اکتشاف بیش از حد نیز احتمال دارد به ربات صدمه بزند. بنابراین، بیش‌تر روش‌های فارغ از مدل از یک سیاست تصادفی برای اکتشاف استفاده می‌کنند که فقط به صورت محلی اکتشاف می‌کند. استراتژی‌های اکتشاف می‌توانند به استراتژی‌های اکتشاف مبتنی بر گام و استراتژی‌های اکتشاف مبتنی بر اپیزود دسته‌بندی شوند. در حالی که اکتشاف مبتنی بر گام از عمل

[1] Average return

اکتشافی در هر گام زمانی استفاده می‌کند، اکتشاف مبتنی بر اپیزود مستقیما بردار پارامتر θ سیاست را فقط در آغاز اپیزود تغییر می‌دهد.

استراتژی ارزیابی سیاست در مورد چگونگی ارزیابی کیفیت تراژکتوری‌های اجرا شده تصمیم‌گیری می‌کند. در اینجا نیز می‌توانیم بین ارزیابی‌های مبتنی بر اپیزود و ارزیابی‌های مبتنی بر گام تفاوت قائل شویم. استراتژی‌های ارزیابی مبتنی بر گام تراژکتوری τ را به تک‌تک گام‌هایش $(x_0, u_0, x_1, u_1, \cdots)$ تجزیه کرده و سعی می‌کنند کیفیت تک‌تک اعمال را ارزیابی کنند. در مقابل، ارزیابی مبتنی بر اپیزود مستقیما از حاصل‌های کل تراژکتوری‌ها استفاده کرده تا کیفیت پارامترهای سیاست استفاده شده θ را ارزیابی کند.

در آخر، استراتژی به‌روز رسانی سیاست از کیفیت برآورد استراتژی ارزیابی استفاده کرده تا به‌روز رسانی سیاست را تعیین کند. استراتژی‌های به‌روز رسانی می‌توانند براساس روش بهینه‌سازی‌ای که توسط الگوریتم جستجوی سیاست به کار گرفته شده است دسته‌بندی شوند. در حالی که بیش‌تر استراتژی‌های رایج به‌روز رسانی مبتنی بر گرادیان صعودی هستند که منجر به روش‌های گرادیان سیاست می‌شوند[91]، رویکردهای مبتنی بر استنتاج از بیشینه‌سازی امید ریاضی استفاده کرده[89] و رویکردهای نظریه اطلاعات[92] از دیدگاه‌های نظریه اطلاعات برای به‌روز رسانی سیاست استفاده می‌کنند. روش‌های مهم دیگری از قبیل رویکردهای انتگرال مسیر و بهینه‌سازی تصادفی نیز وجود دارند.

جستجوی سیاست فارغ از مدل می‌تواند به سیاست‌هایی با تعداد متوسط پارامتر یعنی تا چند صد پارامتر اعمال شود. بیش‌تر کاربردها از بازنمایی‌های سیاست خطی از قبیل کنترل‌کننده‌های خطی یا مقدمات حرکت دینامیکی استفاده می‌کنند.

۲-۲-۳- جستجوی سیاست مبتنی بر مدل

روش‌های جستجوی سیاست فارغ از مدل ذاتا مبتنی بر نمونه‌برداری تراژکتوری‌ها با استفاده از ربات هستند تا سیاست‌های خوب پیدا را کنند. نمونه‌برداری کردن تراژکتوری‌ها در شبیه‌سازی کامپیوتر نسبتا سرراست و مستقیم است. اما هنگام کار با سیستم‌های مکانیکی از قبیل ربات‌ها، هر نمونه مربوط به تعامل مستقیم با ربات می‌شود، چیزی که نیاز به زمان آزمایش قابل توجه داشته و موجب فرسودگی در ربات‌های غیرصنعتی می‌شود. بسته به کار می‌تواند یادگیری مدل آسان‌تر بوده یا یادگیری مستقیم سیاست آسان‌تر باشد. روش‌های جستجوی

سیاست مبتنی بر مدل سعی دارند مشکل کارا نبودن از لحاظ داده را با استفاده از داده مشاهده شده به منظور یادگیری مدل پیشرو دینامیک ربات حل کنند. سپس، این مدل پیشرو برای شبیه‌سازی‌های درونی دینامیک ربات استفاده می‌شود که مبتنی بر آن سیاست یاد گرفته می‌شود.

الگوریتم‌های جستجوی سیاست مبتنی بر مدل معمولا تنظیم[1] زیر را در نظر می‌گیرند: حالت X مطابق دینامیک مارکفی زیر تکامل می‌یابد

$$x_{t+1} = f(x_t, u_t) + \omega,\ x_0 \sim p(x_0) \qquad \textbf{معادله(۲-۹)}$$

که در این رابطه، f یک تابع غیرخطی، u یک سیگنال کنترلی (عمل) و ω نویز جمع‌پذیر (افزودنی)[2] است که غالبا گوسی i.i.d در نظر گرفته می‌شود. علاوه بر این، یک تنظیم اپیزودیک (چند بخشی)[3] در نظر گرفته شده است که در آن ربات بعد از اجرای یک سیاست به حالت اولیه x_0 بازگردانده می‌شود. توزیع حالت اولیه $p(x_0)$ غالبا یک توزیع گوسی $\mathcal{N}(x_0|\mu_0^x, \Sigma_0^x)$ در نظر گرفته می‌شود. علاوه بر این، مسائل با افق متناهی را در نظر می‌گیریم، به این معنی که هدف از جستجوی سیاست پیدا کردن یک سیاست پارامتربندی شده π_θ^* است که امید ریاضی پاداش بلندمدت را بیشینه می‌کند

$$\pi_\theta^* \in \underset{\pi_\theta}{argmax}\, J_\theta \quad \underset{\pi}{argmax} \sum_{t=1}^{T} \gamma^t \mathbb{E}[r(x_t, u_t)|\pi_\theta], \qquad \gamma \in [0,1] \qquad \textbf{معادله(۲-۱۰)}$$

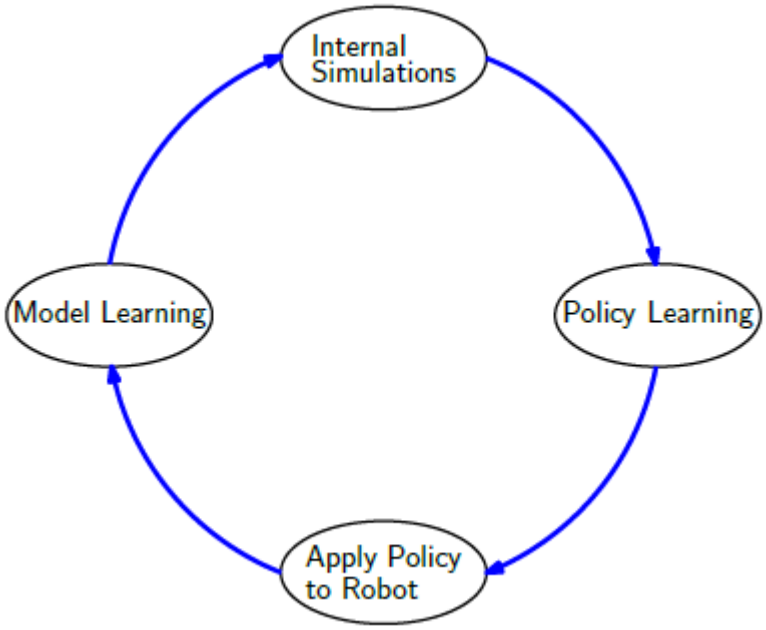
که در این رابطه r سیگنال پاداش آنی، γ ضریب کاهش و سیاست π_θ با پارامترهای θ پارامتربندی شده است. بنابراین، پیدا کردن π_θ^* در معادله(۲-۱۰) معادل با پیدا کردن پارامترهای سیاست بهینه مربوطه θ^* است.

برای برخی مسائل، روش‌های یادگیری تقویتی مبتنی بر مدل نوید این را می‌دهند که با یادگیری مدلی از نگاشت انتقال در معادله(۲-۹)، در مقایسه با یادگیری تقویتی فارغ از مدل به تعامل‌های کم‌تری با ربات نیاز دارند، در عین حال با استفاده از مدلی که از داده‌های مشاهده شده آموخته شده به صورت کارا به وضعیت‌های دیده نشده تعمیم می‌یابند[86].

[1] Set-up
[2] Additive
[3] Episodic

ایده کلی یادگیری تقویتی مبتنی بر مدل در شکل ۲-۵ نشان داده شده است. مدل آموخته شده برای شبیه‌سازی‌های درونی استفاده می‌شود، به عبارت دیگر، پیش‌بینی‌ها در مورد چگونگی رفتار ربات واقعی و محیط پیرامونش است در صورتی که ربات سیاست فعلی را دنبال کند. براساس این شبیه‌سازی‌های درونی، کیفیت سیاست با استفاده از معادله(۲-۱۰) ارزیابی شده و مطابق آن بهبود می‌یابد. سپس، سیاست به‌روز رسانی شده دوباره با استفاده از معادله(۲-۱۰) ارزیابی شده و بهبود می‌یابد. این چرخه ارزیابی/ بهبود سیاست هنگامی که سیاست یاد گرفته می‌شود پایان می‌یابد، به این معنی که دیگر تغییر نمی‌کند و به بهینه (محلی) رسیده است. به محض یادگیری یک سیاست، آن سیاست به ربات اعمال شده و داده جدید ضبط می‌شود. این مجموعه داده بعد از ترکیب با داده از قبل جمع‌آوری شده، برای به‌روز رسانی و اصلاح مدل دینامیکی آموخته شده استفاده می‌شود. از لحاظ نظری، این چرخه برای همیشه ادامه می‌یابد. توجه داشته باشید که فقط اعمال سیاست به تعامل با ربات نیاز دارد. شبیه‌سازی‌های درونی و یادگیری سیاست فقط از مدل کامپیوتری یادگرفته شده دینامیک ربات استفاده می‌کنند.

در حالی که ایده استفاده از مدل‌ها در زمینه یادگیری ربات، از کار Aboaf [87] در دهه ۱۹۸۰ شناخته شده است، به خاطر وابستگی شدیدش به کیفیت مدل آموخته شده محدود شده است، موضوعی که در شکل ۲-۵ مشهود است: سیاست آموخته شده ذاتا مبتنی بر شبیه‌سازی‌های درونی است که با استفاده از مدل آموخته شده انجام شده است. زمانی که مدل دقیقا به دینامیک درست ربات مربوط می‌شود، نمونه‌برداری از مدل آموخته شده معادل با نمونه‌برداری از ربات واقعی است.

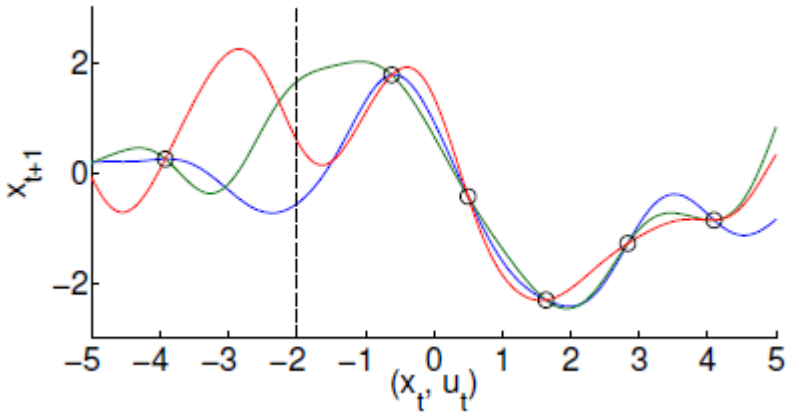


شکل ۲-۵ چرخه کلی در یادگیری تقویتی مبتنی بر مدل. مدل یاد گرفته شده برای شبیه‌سازی‌های درونی استفاده می‌شود (تمرین ذهنی) که براساس آن سیاست بهبود می‌یابد. سپس سیاست آموخته شده به ربات اعمال می‌شود. داده حاصل از این تعامل برای اصلاح مدل استفاده می‌شود و این چرخه ادامه می‌یابد تا زمانی که مدل و سیاست آموخته شده همگرا شوند.[90]

اما، در عمل مدل آموخته شده دقیق نیست بلکه فقط یک تقریب کمابیش دقیق از دینامیک واقعی است. برای مثال، در مناطقی که داده آموزشی پراکنده است، همان‌طور که در شکل ۲-۶ نشان داده شده است کیفیت مدل یاد گرفته شده می‌تواند نامناسب باشد. چندین تابع محتمل وجود دارند که می‌توانند مقادیر تابع مشاهده شده را تولید کرده باشند (دایره‌های سیاه). در مناطق با داده آموزشی پراکنده، مدل‌ها و پیش‌بینی‌هایشان به طور قابل توجهی فرق می‌کنند. هر مدل منفرد منجر به پیش‌بینی‌های خود رای[1] می‌شود که این پیش‌بینی‌ها به نوبه خود می‌توانند منجر به استراتژی‌های کنترلی شوند که در برابر خطاهای مدل مقاوم نیستند. این رفتار می‌تواند اثری شدید در رباتیک داشته باشد، برای مثال می‌تواند منجر به تخمین جرم‌های منفی یا ضرایب اصطکاک منفی شود. این آثار غیرمحتمل غالبا توسط سیستم یادگیری به کار گرفته می‌شوند چون انرژی را در سیستم قرار می‌دهند که باعث می‌شود سیستم به "حرکت دائمی"[2] اعتقاد پیدا کند. بنابراین، به جای انتخاب

[1] Overconfident
[2] Perpetuum mobile

یک مدل منفرد (برای مثال مدل بیشینه درست‌نمایی)، باید عدم اطمینانمان در مورد تابع پنهان f را با یک توزیع احتمالاتی $p(f)$ نشان دهیم تا در مقابل این‌چنین خطاهای مدل مقاوم باشد[93]. با در نظر گرفتن عدم قطعیت مدل، احتمال اثر حرکت دائمی به طور چشم‌گیر کم‌تر است.



شکل ۲-۶ خطاهای مدل در جستجوی سیاست مبتنی بر مدل. در این مثال، شش مقدار تابع مشاهده شده‌اند (دایره‌های سیاه). سه تابع نشان داده شده‌اند که می‌توانند این مشاهدات را تولید کرده باشند. هر تابع منفرد (برای مثال تابع بیشینه درست‌نمایی) پیش‌بینی‌های کمابیش دلخواهی در مناطق با داده آموزشی پراکنده تولید می‌کند که یک مکان نمونه از آن با خط‌چین نشان داده شده است. به جای انتخاب یک تقریب‌زننده تابع منفرد، عدم اطمینانمان در مورد تابع زیرین را با یک توزیع احتمالاتی نشان می‌دهیم تا در برابر این‌چنین خطاهای مدل مقاوم باشد.[90]

از آنجایی که سیاست آموخته شده ذاتا به کیفیت مدل پیشرو آموخته شده (که اساسا به عنوان شبیه‌ساز ربات به کار می‌رود) متکی است، خطاهای مدل فقط باعث سیاست‌های ضعیف‌تر نمی‌شوند، بلکه غالبا فرآیند یادگیری را به شدت به یک طرف متمایل می‌کنند[1]. از همین رو، منابع علمی جستجوی سیاست مبتنی بر مدل اکثرا روی ساخت مدل تمرکز می‌کند یا به عبارت دیگر توضیح می‌دهد که چه نوع مدلی برای دینامیک پیشرو استفاده شده و چگونه آموزش داده شده است.

[1] Bias

رویکردهای سر و کله زدن[1] با مدل‌های غیرقطعی. سر و کله زدن با مدل‌های دینامیکی نادقیق یکی از بزرگ‌ترین چالش‌ها در یادگیری تقویتی مبتنی بر مدل است چون خطاهای کوچک در مدل می‌تواند منجر به خطاهای بزرگ در سیاست شود[86]. بی‌دقتی‌ها ممکن است از یک دسته مدل بیش‌ازحد محدودکننده یا از نبود مجموعه دادگان به اندازه کافی قوی برای آموزش مدل‌ها نشأت بگیرند، چیزی که می‌تواند منجر به مدلسازی بد دینامیک پیشرو درست شود.

علاوه بر این، معمولا نویز سیستم منشأ دیگری از عدم قطعیت را اضافه می‌کند.

به طور معمول، یک فرض معادل قطعیت[2] انجام می‌شود و مدل بیشینه درست‌نمایی برای برنامه‌ریزی انتخاب می‌شود[93]. اما این فرض معادل قطعیت در بیش‌تر موارد جالب توجه نقض شده و می‌تواند منجر به خطاهای بزرگ در سیاست شود. علاوه بر این، همان‌طور که قبلا در مرجع [87] اشاره شده است، خیلی از رویکردها مشتقات بازگشتی مورد نظر (امید ریاضی بازگشت) را با پس- انتشار مشتقات از طریق مدل‌های پیشرو آموخته شده از سیستم به دست می‌آورند. به ویژه، این گام در معرض خطاهای مدل است چون بهینه‌سازهای مبتنی بر گرادیان، پارامترهای سیاست را در امتداد گرادیان‌هایشان بهبود می‌بخشند. سیاست آموخته شده باید قوی باشد تا خطاهای مدل را جبران کند به‌طوری که زمانی که به سیستم واقعی اعمال می‌شود منجر به عملکرد خوب شود. یادگیری مدل‌های دینامیکی درست برای ساختن سیاست‌های قوی حیاتی است و یکی از بزرگ‌ترین چالش‌ها در جستجوی سیاست مبتنی بر مدل باقی مانده است.

در مرجع [94]، نویسندگان خطای ناشناخته (نامعلوم) دینامیکی را با استفاده از رگرسیون وزن‌دار میدان دریافتی[3][95] مدلسازی کرده‌اند. مدلسازی صریح اختلالات ناشناخته منجر به نیرومندی فزاینده کنترل‌کننده‌های یاد گرفته شده می‌شود. ایده طراحی کنترل‌کننده‌ها در مواجهه با مدل‌های پیشرو نادقیق (تلقی شده به عنوان ایده‌آل)، به طور تنگاتنگ مرتبط با کنترل مقاوم در رباتیک کلاسیک است.

[1] Deal

[2] فرض می‌شود که سیاست بهینه برای مدل آموخته شده به سیاست بهینه برای دینامیک درست مربوط می‌شود. عدم قطعیت در مورد مدل آموخته شده نادیده گرفته می‌شود.

[3] Receptive-field weighted regression

کنترل مقاوم سعی دارد با وجود خطاهای مدلسازی (به طور معمول محدود) به عملکرد تضمین‌شده یا پایداری برسد. برای مثال، شکل‌دهی چرخه $\mathcal{H}_\infty$ [96] تضمین می‌کند که سیستم نزدیک به رفتار مورد انتظارش می‌ماند اگرچه اختلالات (محدود) به سیستم وارد شوند.

در کنترل تطبیقی، عدم قطعیت‌های پارامتر معمولا به وسیله توزیع‌های احتمالاتی نامحدود (بی‌کران) توصیف می‌شود[97]. عدم قطعیت پارامتر مدل به طور معمول در طراحی الگوریتم‌های کنترل تطبیقی استفاده نمی‌شود. در عوض، تخمین‌های پارامترها به عنوان پارامترهای درست تلقی می‌شوند[98]. یک رویکرد به طراحی کنترل‌کننده‌های تطبیقی که عدم قطعیت در مورد پارامترهای مدل را در نظر می‌گیرد کنترل تطبیقی تصادفی[1] است[97]. هنگامی که عدم قطعیت پارامتر از طریق کاوش کاهش می‌یابد، کنترل تطبیقی تصادفی به اصل کنترل دوگانه منجر می‌شود[99]. کنترل تطبیقی دوگانه عمدتا برای سیستم‌های خطی مورد بررسی قرار گرفته است[98]. در مرجع [100]، بسطی از کنترل تطبیقی دوگانه به مورد سیستم‌های غیرخطی با کنترل‌های همگر[2] پیشنهاد شده است. یک قانون کنترلی واریانس کمینه به دست می‌آید و عدم قطعیت در مورد پارامترهای مدل جریمه شده تا تخمینشان را بهبود دهد، امری که نیاز به شناسایی قبلی سیستم را حذف می‌کند.

رویکردهای یادگیری تقویتی که به طور صریح مشکل دقیق نبودن مدل‌ها در رباتیک را حل می‌کنند اخیرا معرفی شده‌اند[93]. برای مثال، در مرجع [101]، ایده کلیدی استفاده از آزمایش واقعی[3] برای ارزیابی یک سیاست است، اما سپس از یک مدل ابتدایی[4] سیستم برای تخمین مشتق ارزیابی نسبت به پارامترهای سیاست استفاده می‌کند (و بهبودهای محلی را پیشنهاد می‌کند). به ویژه، الگوریتم پیشنهاد شده به صورت مکرر مدل f را مطابق با $f^{[i+1]}(x_t,u_t)=f^{[i]}(x_t,u_t)+x_{t+1}^{[i]}-f^{[i]}\left(x_t^{[i]},u_t^{[i]}\right)$ به‌روز رسانی می‌کند که در این رابطه t اندیس‌های زمان هستند. در اینجا، $x_t^{[i]}$ و $u_t^{[i]}$ از یک تراژکتوری مشاهده شده برگرفته شده‌اند. نتیجه می‌شود که مدل به‌روز رسانی شده $f^{[i+1]}$ دقیقا تراژکتوری مشاهده شده را پیش‌بینی می‌کند به عبارت دیگر $f^{[i+1]}\left(x_t^{[i]},u_t^{[i]}\right)=x_{t+1}^{[i]}$. این الگوریتم، گرادیان‌های سیاست را در امتداد تراژکتوری‌ای از حالت‌ها و

[1] Stochastic
[2] Affine
[3] Real-life
[4] Crude

کنترل‌ها در سیستم واقعی ارزیابی می‌کند. در مقابل، یک رویکرد مبتنی بر مدل معمول[1]، مشتقات را در امتداد تراژکتوری‌ای که به وسیله مدل پیش‌بینی شده است ارزیابی می‌کند، چیزی که به هنگام دقیق نبودن مدل به تراژکتوری سیستم واقعی مربوط نمی‌شود. توجه داشته باشید که رویکرد مرجع [101] مستقیما به دینامیک انتقال تصادفی یا سیستم‌های با حالت‌های پنهان تعمیم نمی‌یابد. علاوه بر این، یک مدل پارامتری تقریبی از دینامیک اصلی باید از پیش معلوم باشد.

چالش‌های بزرگ در جستجوی سیاست مبتنی بر مدل. سه چالش کلی وجود دارد که در روش‌های جستجوی سیاست مبتنی بر مدل باید مورد بررسی قرار گیرد: چه مدلی را باید یاد بگیرد، چگونه از این مدل برای پیش‌بینی‌های بلندمدت استفاده کند و بر اساس این پیش‌بینی‌های بلندمدت چگونه این سیاست را به‌روزرسانی کند. این سه چالش به سه عنصر در شکل ۲-۵ مربوط می‌شوند که نیازی به تعامل فیزیکی با ربات ندارند: یادگیری مدل، شبیه‌سازی‌های درونی و یادگیری سیاست.

در زیربخش اول، مرور کلی مختصری درباره دو مدلی که غالبا در جستجوی سیاست مبتنی بر مدل استفاده می‌شوند[3] ارائه می‌دهیم که عبارتند از رگرسیون (بیزی) وزن‌دار محلی (LWBR) و فرآیندهای گوسی (GPs) [102]. در زیربخش دوم، در مورد دو روش کلی طرز استفاده از این مدل‌ها برای پیش‌بینی‌های بلندمدت بحث می‌کنیم: استنتاج تصادفی یا به عبارت دیگر نمونه‌برداری و استنتاج تقریبی قطعی. در زیربخش سوم، مختصرا در مورد چند گزینه به‌روز رسانی سیاست بحث می‌کنیم.

بعد از معرفی این مفاهیم کلی، در زیربخش چهارم، در مورد الگوریتم‌های جستجوی سیاست مبتنی بر مدلی بحث می‌کنیم که مدل‌های آموخته شده و الگوریتم‌های استنتاجی که در جدول ۲-۱ نشان داده شده‌اند را ترکیب می‌کنند. ما روی مورد اپیزودیک تمرکز می‌کنیم که در آن توزیع حالت آغازین $p(x_0)$ معلوم است. به ویژه، جزئیات چهار روش جستجوی سیاست مبتنی بر مدل را بیان می‌کنیم. ابتدا با PEGASUS شروع می‌کنیم که یک مفهوم کلی برای نمونه‌برداری کارای تراژکتوری در MDP های تصادفی برای مدلی معلوم است[103]. سپس، دو روش ترکیب PEGASUS با LWBR را به منظور یادگیری کنترل‌کننده‌های مقاوم برای پرواز هلی‌کوپترها[104] ارائه می‌دهیم. سپس، رویکردی برای استفاده از الگوریتم PEGASUS برای

[1] Typical

نمونه‌برداری تراژکتوری‌ها از مدل‌های پیشرو GP [105] در زمینه یادگیری یک کنترل‌کننده بالون را ارائه می‌دهیم. سرانجام به عنوان رویکرد چهارم، ایده‌های چارچوب کاری جستجوی سیاست PILCO [3] را به طور کلی شرح می‌دهیم که استنتاج تقریبی قطعی کارا برای پیش‌بینی‌های بلندمدت را با مدل‌های دینامیکی GP برای یادگیری کنترل سیستم‌های مکانیکی و بازوهای ربات ترکیب می‌کند.

جدول ۲-۱ رویکردهای جستجوی سیاست مبتنی بر مدل بر اساس مدل آموخته شده و روش تولید تراژکتوری‌ها گروه‌بندی می‌شوند. به خاطر سادگی نمونه‌برداری تراژکتوری‌ها، بیش‌تر روش‌های جستجوی سیاست این رویکرد را دنبال می‌کنند. پیش‌بینی‌های قطعی تراژکتوری فقط می‌توانند در موارد خاصی انجام شوند که استنتاج تقریبی به فرم بسته ممکن است. گرچه آنها از لحاظ ریاضیاتی بیش‌تر از نمونه‌برداری تراژکتوری درگیر هستند، اما از واریانس‌های بزرگ نمونه‌ها رنج نمی‌برند. علاوه بر این، می‌توانند امکان محاسبات تحلیلی گرادیان را فراهم کنند که در هنگام وجود صدها پارامتر سیاست حیاتی است.[90]

مدل پیشرو آموخته شده	پیش‌بینی تراژکتوری	
	تصادفی	قطعی
مدلهای خطی (محلی)	[104][106][107]	–
فرآیندهای گوسی	[105]	[3][108]

۱-۳-۲-۲- مدل‌های پیشرو احتمالاتی

برای کاهش اثر خطاهای مدل، مدل‌های احتمالاتی که عدم قطعیت در مورد دینامیک انتقال اصلی را بیان می‌کنند به مدل‌های قطعی (برای مثال مدل‌های بیشینه درست‌نمایی از دینامیک انتقال) که یک فرض معادل قطعیت را ایجاب می‌کنند ترجیح داده می‌شوند.

دو مدل احتمالاتی غیرپارامتری را در زیر به طور مختصر معرفی می‌کنیم که غالبا برای یادگیری دینامیک پیشرو در زمینه یادگیری تقویتی و رباتیک استفاده می‌شوند: رگرسیون بیزی وزن‌دار محلی (LWBR) [104] و فرآیندهای گوسی[3].

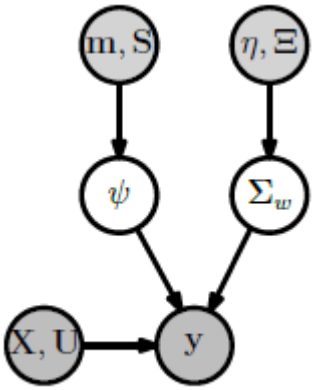
- **رگرسیون بیزی وزن‌دار محلی**

ابتدا با مدل رگرسیون خطی شروع می‌کنیم که در آن دینامیک انتقال به صورت زیر بیان می‌شود:

$$x_{t+1} = [x_t, u_t]^T \psi + \omega, \qquad \omega \sim \mathcal{N}(0, \Sigma_\omega) \qquad \textbf{معادله(۲-۱۱)}$$

در اینجا، ψ پارامترهای مدل رگرسیون خطی بیزی هستند و $\omega \sim \mathcal{N}(0, \Sigma_\omega)$ نویز گوسی i.i.d سیستم است. این مدل نسبت به پارامترهای نامعلوم ψ که ورودی (x_t, u_t) را وزن‌دار می‌کند خطی است.

در رگرسیون خطی بیزی، توزیع‌های مقدم روی پارامترها ψ و روی واریانس نویز Σ_ω در نظر می‌گیریم. به طور معمول، توزیع مقدم روی ψ، گوسی با میانگین m و کوواریانس S و توزیع مقدم روی عناصر قطری $1/\sigma_i^2$ از Σ_ω^{-1}، یک توزیع گاما با پارامترهای مقیاس و شکل η و ξ است، به طوری که مدل کلی مزدوج است، به شکل ۲-۷ نگاه کنید که در آن ورودی‌های آموزشی با [X,U] و اهداف آموزشی با y نشان داده شده‌اند. در معادله(۲-۱۱) مدل رگرسیون خطی (بیزی)، پیدا کردن تخمین‌های بیشینه درست‌نمایی یا توزیع‌های پسین پارامترها ψ نسبتا سرراست و مستقیم است. اما خود مدل خیلی گویا نیست چون یک رابطه خطی کلی بین ورودی‌ها (x_t, u_t) و حالت بعدی x_{t+1} فرض می‌کند.



شکل ۲-۷ مدل گرافی برای رگرسیون خطی بیزی. یک مقدم گوسی با پارامترهای m و S که روی پارامترهای مدل ψ در نظر گرفته شده است و مقدم‌های گاما با پارامترهای η و ξ که روی عناصر قطری ماتریس دقت Σ_ω^{-1} در نظر گرفته شده‌اند. ورودی‌ها و اهداف آموزشی به ترتیب با [X,U] و y نشان داده شده‌اند.[90]

ایده رگرسیون خطی وزن‌دار محلی (LWR)، استفاده از ویژگی‌های خوب مدل رگرسیون خطی و اما در نظر گرفتن دسته کلی‌تر از توابع است: توابع خطی محلی. LWR یک تقریب خطی محلی از تابع اصلی پیدا می‌کند[109]. به این منظور، هر ورودی آزمون (x_t, u_t) با یک ضریب وزن‌دهی b_i تجهیز می‌شوند که معین می‌کند نقطه آموزشی (x_i, u_i) چقدر به (x_t, u_t) نزدیک است. یک مثال برای چنین وزنی، وزن‌دهی با شکل گوسی است $b_i = \exp(-\|(x_i, u_i) - (x_t, u_t)\|^2/\kappa^2)$. اگر فاصله بین (x_i, u_i) و (x_*, u_*) خیلی بزرگ‌تر از κ باشد، وزن مربوطه b_i به ۰ کاهش می‌یابد. چون این وزن‌ها باید برای هر نقطه مورد سوال محاسبه شوند، فقط ذخیره پارامترها ψ کافی نیست بلکه کل مجموعه داده آموزشی [X,U] نیاز است، چیزی که منجر به رویکرد غیرپارامتری می‌شود.

مشابه رگرسیون خطی بیزی، می‌توانیم مقدم‌هایی روی پارامترها و کوواریانس نویز در نظر بگیریم. برای سادگی، یک ماتریس کوواریانس نویز Σ_ω و یک توزیع مقدم گوسی با میانگین صفر روی پارامترها ψ فرض می‌کنیم $\mathcal{N}(\psi|0, S)$.

برای هر نقطه مورد سوال (x_t, u_t)، مطابق قضیه بیز به صورت زیر یک توزیع پسین روی پارامترها ψ محاسبه می‌شود:

$$p(\psi|X, U, y) = \frac{p(\psi)p(y|X, U, \psi)}{p(y|X, U)} \propto p(\psi)p(y|X, U, \psi) \quad \textbf{معادله(۲-۱۲)}$$

برای راحتی در نمادگذاری $\tilde{X} := [X, U]$ را تعریف می‌کنیم. روابط میانگین و کوواریانس پسین ψ در (x_t, u_t) به ترتیب به صورت زیر است:

$$\mathbb{E}[\psi|\tilde{X}, y] = S\tilde{X}B(B\tilde{X}^T S\tilde{X}B + \Sigma_\omega)^{-1}y = S\tilde{X}B\Omega^{-1}y \quad \textbf{معادله(۲-۱۳)}$$

$$\Omega^{-1} = (B\tilde{X}^T S\tilde{X}B + \Sigma_\omega)^{-1}$$

$$cov[\psi|\tilde{X}, y] = S - S^T\tilde{X}B\Omega^{-1}B\tilde{X}^T S \quad \textbf{معادله(۲-۱۴)}$$

$$b_i = exp(-\|(x_i, u_i) - (x_t, u_t)\|^2/\kappa^2) \quad \textbf{معادله(۲-۱۵)}$$

در روابط بالا، $B = diag(b_1, \cdots, b_n)$ است و y اهداف آموزشی هستند.

توزیع پیش‌بینی‌کننده. میانگین و کوواریانس توزیع پیش‌بینی‌کننده $p(x_{t+1})$ برای یک جفت حالت-کنترل معلوم (x_t, u_t) به ترتیب به صورت زیر هستند:

$$\mu_{t+1}^{x} = [x_t, u_t]^T \mathbb{E}[\psi | \tilde{X}, y] = [x_t, u_t]^T S \tilde{X} B \Omega^{-1} y \quad \textbf{معادله(۲-۱۶)}$$

$$\Sigma_{t+1}^{x} = [x_t, u_t]^T cov[\psi | \tilde{X}, y][x_t, u_t] + \Sigma_{\omega} \quad \textbf{معادله(۲-۱۷)}$$

در عمل، میانگین و کوواریانس پسین روی پارامترها ψ می‌تواند از طریق اعمال لم‌های معکوس‌سازی ماتریس[109] و استفاده از پراکندگی خیلی کاراتر محاسبه شود.

حال نگاهی به کوواریانس پیش‌بینی‌کننده می‌کنیم در زمانی که $[x_t, u_t]$ از مجموعه آموزشی $[X, U]$ خیلی دور است: ماتریس وزن تقریبا صفر است، چیزی که منجر به واریانس پسین روی پارامترهای مدل ψ می‌شود که معادل با عدم قطعیت مقدم S است (به معادله(۲-۱۴) رجوع شود). از همین رو، برخلاف مورد رگرسیون وزن‌دار محلی غیربیزی، واریانس پیش‌بینی‌کننده در $[x_t, u_t]$ غیرصفر است.

- **رگرسیون فرآیند گوسی**

یک فرآیند گوسی یک توزیع $p(f)$ روی توابع f است. مطابق تعریف، یک GP مجموعه‌ای از متغیرهای تصادفی $f_1, f_2, \cdots$ است که هر تعداد متناهی از آنها دارای توزیع گوسی هستند[102]. در بستر این بخش، یک GP بر روی توابع انتقال در نظر گرفته می‌شود. از آنجایی که GP یک مدل غیرپارامتری است، کافی است که فرضیات سطح بالا از قبیل مشتق‌پذیری یا تناوبی بودن روی تابع اصلی مشخص شوند. به طور معمول، این ویژگی‌های سطح بالا آسان‌تر از مدل صریحا پارامتری مشخص می‌شوند.

یک GP، تماما با یک تابع میانگین $m(.)$ و یک تابع/کرنل کوواریانس نیمه‌معین مثبت $k(.,.)$ مشخص می‌شود. فرضیات استاندارد در مدل‌های GP، یک تابع مقدم میانگین $m \equiv 0$ و تابع کوواریانس به صورت معادله(۲-۱۸) هستند که $\tilde{x} := [x^T u^T]^T$ است.

$$k(\tilde{x}_p, \tilde{x}_q) = \sigma_f^2 \, exp\left(-\frac{1}{2}(\tilde{x}_p - \tilde{x}_q)^T \Lambda^{-1} (\tilde{x}_p - \tilde{x}_q)\right) + \delta_{pq}\sigma_{\omega}^2 \quad \textbf{معادله(۲-۱۸)}$$

در معادله(۲-۱۸)، $\Lambda := diag([\ell_1^2, \cdots, \ell_D^2])$ را تعریف کردیم که به مقیاس‌های طول مشخصه ℓ_i وابسته است و σ_f^2 واریانس مقدم تابع پنهان f است. با فرض n ورودی آموزشی $\tilde{X} = [\tilde{x}_1, \cdots, \tilde{x}_n]$ و اهداف آموزشی

مربوطه $y = [y_1, \cdots, y_n]^T$ ، ابر پارامترهای GP پسین (مقیاس‌های طول ℓ_i ، واریانس سیگنال σ_f^2 و واریانس نویز σ_ω^2) با استفاده از بیشینه‌سازی شواهد یاد گرفته می‌شوند[102].

توزیع پیش‌بینی‌کننده. GP پسین یک مدل پیش‌بینی یک گامه است و حالت بعدی پیش‌بینی شده x_{t+1} دارای توزیع گوسی است

$$p(x_{t+1}|x_t, u_t) = \mathcal{N}(x_{t+1}|\mu_{t+1}^x, \Sigma_{t+1}^x) \quad \textbf{معادله(۲-۱۹)}$$

$$\mu_{t+1}^x = \mathbb{E}_f[f(x_t, u_t)]\,, \quad \Sigma_{t+1}^x = var_f[f(x_t, u_t)] \quad \textbf{معادله(۲-۲۰)}$$

که میانگین و واریانس پیش‌بینی GP به ترتیب به صورت زیر هستند

$$\mu_{t+1}^x = k_*^T K^{-1} y = k_*^T \beta \quad \textbf{معادله(۲-۲۱)}$$

$$\Sigma_{t+1}^x = k_{**} - k_*^T K^{-1} k_* \quad \textbf{معادله(۲-۲۲)}$$

که $k_* := k(\tilde{X}, \tilde{x}_t)$ ، $k_{**} := k(\tilde{x}_t, \tilde{x}_t)$ ، $\beta := K^{-1}y$ هستند و K ماتریس کرنل با عناصر $K_{ij} = k(\tilde{x}_i, \tilde{x}_j)$ است.

توجه داشته باشید که در مناطق دور از داده آموزشی، عدم قطعیت پیش‌بینی‌کننده در معادله(۲-۲۲) به عدم قطعیت مقدم در مورد تابع مربوط می‌شود، به عبارت دیگر برای سیستم‌های قطعی که $\Sigma_\omega = 0$ است $k_{**} > 0$ به دست می‌آوریم. بنابراین، GP یک مدل غیرپارامتری غیرمنحط است.

۲-۳-۲-۲- پیش‌بینی‌های بلندمدت با یک مدل معلوم

در بخش‌های زیر فرض می‌کنیم که یک مدل برای دینامیک انتقال معلوم است. مشروط به این مدل، بین دو رویکرد تولید پیش‌بینی‌های بلندمدت تمایز قائل می‌شویم: رویکردهای مبتنی بر نمونه‌برداری Monte-Carlo یا استنتاج تقریبی قطعی

- **پیش‌بینی مبتنی بر نمونه‌برداری تراژکتوری: PEGASUS**

PEGASUS (جستجو و ارزیابی خوب بودن سیاست با استفاده از سناریوها[1]) یک چارچوب کاری مفهومی برای نمونه‌برداری تراژکتوری در MDP های تصادفی است[103]. ایده کلیدی، انتقال MDP تصادفی به یک MDP قطعی تقویت شده است. به این منظور، PEGASUS دسترسی به یک شبیه‌ساز بدون مولد عدد تصادفی داخلی را فرض می‌کند. زمانی که از این مدل نمونه‌برداری می‌کنیم، PEGASUS پیشاپیش از بیرون اعداد تصادفی را فراهم می‌کند. از این طریق، PEGASUS واریانس نمونه‌برداری را به شدت کاهش می‌دهد. از همین رو، نمونه‌برداری با پیروی از رویکرد PEGASUS، در جستجوی سیاست فارغ از مدل نیز به صورت رایج استفاده می‌شود.

- **نمونه‌برداری تراژکتوری و ارزیابی سیاست**

فرض کنید که یک مدل پیشرو از سیستم پیش‌رو داده شده است که می‌تواند برای نمونه‌برداری تراژکتوری‌ها استفاده شود. اگر انتقالات حالت تصادفی باشند، محاسبه پاداش بلندمدت مورد انتظار که رابطه آن به صورت معادله(۲-۲۳) است

$$J_\theta = \sum_{t=0}^{T} \gamma^t \mathbb{E}[r(x_t)|\pi_\theta], \quad x_0 \sim p(x_0), \quad \gamma \in [0,1] \qquad \textbf{معادله(۲-۲۳)}$$

به تعداد زیادی نمونه تراژکتوری برای محاسبه تقریب $\tilde{J}$ به J نیاز خواهد داشت که معادله آن به صورت زیر است و و در آن نمونه‌های $x_0^{[i]}$ از $p(x_0)$ هستند.

$$\tilde{J}_\theta = \frac{1}{m}\sum_{i=1}^{m} J_\theta\left(x_0^{[i]}\right) \qquad \textbf{معادله(۲-۲۴)}$$

حتی محاسبه گرادیان‌های سیاست قابل اعتماد به نمونه‌های بیش‌تری برای مشتقات قوی نیاز خواهد داشت. اما، به عنوان حد تعداد نامتناهی نمونه، $\lim_{m\to\infty} \tilde{J}_\theta = J_\theta$ را به دست می‌آوریم.

[1] Policy Evaluation-of-Goodness And Search Using Scenarios

برای محاسبات کاراتر، PEGASUS حالت x را با دنباله‌ای از مقادیر تصادفی $w_0, w_1, \cdots$ که از بیرون داده شده‌اند تقویت می‌کند. برای استخراج یک نمونه از $p(x_{t+1}|x_t, u_t)$، PEGASUS از مقادیر نویز از پیش داده شده w_t استفاده می‌کند، به طوری که $x_{t+1} = g(x_t, u_t, w_t)$ است.

از آنجایی که دنباله اعداد تصادفی ثابت است، تکرار آزمایش یکسان منجر به نمونه تراژکتوری یکسان می‌شود.

PEGASUS می‌تواند به صورت تولید m تراژکتوری Monte Carlo و به دست آوردن میانگین پاداششان توصیف شود، اما تولید عدد تصادفی از پیش تعیین شده است. می‌توان نشان داد که حل MDP قطعی تقویت شده معادل با حل MDP تصادفی اصلی است[103]. الگوریتم PEGASUS در الگوریتم ۲-۵ خلاصه شده است.

Init: $g(x, u, w)$, reward r, random numbers w_0 , w_1 , … , initial state distribution $p(x_0)$, policy π_θ
for $i = 1, \dots, m$ do
  $x_0^{[i]} \sim p(x_0)$ ▷ Sample "scenario" from initial state distribution
  for $t = 0, \dots, T-1$ do
    $x_{t+1}^{[i]} = g(x_t^{[i]}, \pi_\theta(x_t), w_t)$ ▷ Succ. state in augmented MDP
  end for
end for

الگوریتم ۲-۵ الگوریتم PEGASUS برای نمونه‌برداری تراژکتوری‌ها[90].

- **پیش‌بینی‌های بلندمدت قطعی**

به جای انجام نمونه‌برداری تصادفی، یک توزیع احتمالاتی $p(\tau)$ روی تراژکتوری‌ها $\tau = (x_0, \cdots, x_T)$ می‌تواند با استفاده از تقریب‌های قطعی از قبیل خطی‌سازی[110]، روش‌های نقطه سیگما[1](برای مثال انتقال بی‌رنگ و

[1] Sigm-point

بو[1] ([111]) و یا تطبیق ممان[2] نیز محاسبه شود. این روش‌های استنتاج رایج، توزیع‌های پیش‌بینی‌کننده سنگین[3] را با گوسی‌ها تقریب می‌زنند.

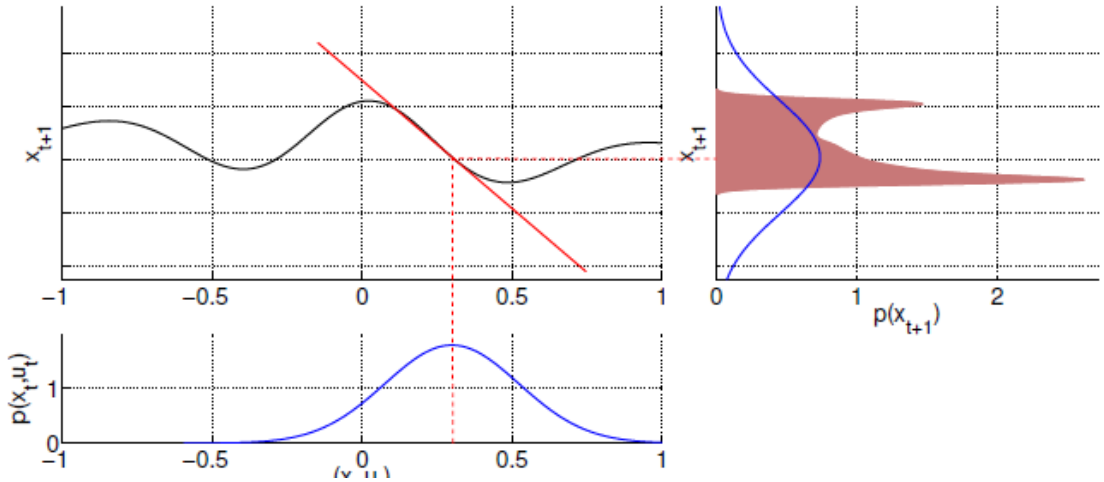
با فرض یک توزیع احتمالاتی گوسی توام $p(x_t, u_t) = \mathcal{N}([x_t, u_t]|\mu_t^{xu}, \Sigma_t^{xu}$، مسئله محاسبه توزیع حالت بعدی $p(x_{t+1})$ به حل انتگرال زیر مربوط می‌شود که در آن $x_{t+1} = f(x_t, u_t) + w$ است.

$$p(x_{t+1}) = \iiint p(x_{t+1}|x_t, u_t)\, p(x_t, u_t) dx_t du_t dw \qquad \textbf{معادله(۲-۲۵)}$$

اگر تابع انتقال f غیرخطی باشد، $p(x_{t+1})$ غیرگوسی است و باید به تکنیک‌های استنتاج تقریبی متوسل شویم. یک تقریب آسان از توزیع پیش‌بینی‌کننده سنگین $p(x_{t+1})$، گوسی $\mathcal{N}(x_{t+1}|\mu_t^x, \Sigma_t^x)$ است. میانگین μ_{t+1}^x و کوواریانس Σ_{t+1}^x این توزیع پیش‌بینی‌کننده می‌توانند به روش‌های مختلفی محاسبه شوند. در ادامه، سه رویکردی که به طور رایج استفاده می‌شوند را به صورت خلاصه شرح می‌دهیم: خطی‌سازی، انتقال بی‌رنگ و بو و تطبیق ممان.

خطی‌سازی. یک روش محاسبه μ_{t+1}^x و Σ_{t+1}^x، این است که تابع انتقال $f \approx F$ را به صورت محلی حول (μ_t^x, Σ_t^x) خطی کرده و سپس، کوواریانس پیش‌بینی‌کننده را با نگاشت توزیع ورودی گوسی به واسطه سیستم خطی شده تخمین زد. با خطی‌سازی، کوواریانس و میانگین پیش‌بینی‌کننده را به ترتیب با رابطه‌های $\mu_{t+1}^x = f(\mu_t^{xu})$ و $\Sigma_{t+1}^x = F\Sigma_t^{xu}F^T + \Sigma_w$ به دست می‌آوریم. شکل ۲-۸ ایده خطی‌سازی را نشان می‌دهد.

[1] Unscented transformation
[2] Moment matching
[3] Unwieldy



شکل ۲-۸ محاسبه توزیع پیش‌بینی‌کننده تقریبی با استفاده از خطی‌سازی. یک توزیع گوسی $p(x_t, u_t)$ (پنل سمت چپ پایین) باید به واسطه یک تابع غیرخطی (نمودار مشکی در پنل سمت چپ بالا) نگاشت شود. توزیع پیش‌بینی‌کننده درست با ناحیه سایه خورده در پنل راست نشان داده شده است. برای به دست آوردن یک تقریب گوسی از توزیع سایه خورده سنگین، تابع غیرخطی در میانگین توزیع ورودی خطی‌سازی شده است (پنل سمت چپ بالا، خط قرمز). سپس، این گوسی به واسطه این تقریب خطی نگاشت شده و توزیع پیش‌بینی‌کننده تقریبی گوسی $p(x_{t+1})$ حاصل شده که در پنل سمت راست به رنگ آبی نشان داده شده است. [90]

خطی‌سازی از لحاظ مفهومی سرراست و از لحاظ محاسباتی کاراست. توجه داشته باشید که این رویکرد، توزیع ورودی گوسی $p(x_t, u_t)$ را دست‌نخورده می‌گذارد اما تابع انتقال f را تقریب می‌زند. یک اشکال بالقوه، این است که برای اجرای خطی‌سازی، تابع انتقال f باید مشتق‌پذیر باشد.[1] علاوه بر این، خطی‌سازی به راحتی می‌تواند واریانس‌های پیش‌بینی‌کننده را کم تخمین بزند[2]، چیزی که می‌تواند باعث شود که سیاست‌ها بیش از حد مهاجم[3] باشند که باعث صدمه به سیستم‌های ربات واقعی می‌شود.

انتقال بی‌رنگ و بو. ایده کلیدی پشت انتقال بی‌رنگ و بو[111]، بازنمایی توزیع $p(x_t, u_t)$ با مجموعه‌ای از نقاط سیگما $(\mathcal{X}_t^{[i]}, \mathcal{U}_t^{[i]})$ است که به طور قطعی انتخاب شده‌اند. برای این نقاط سیگما، مقادیر دقیق تابع

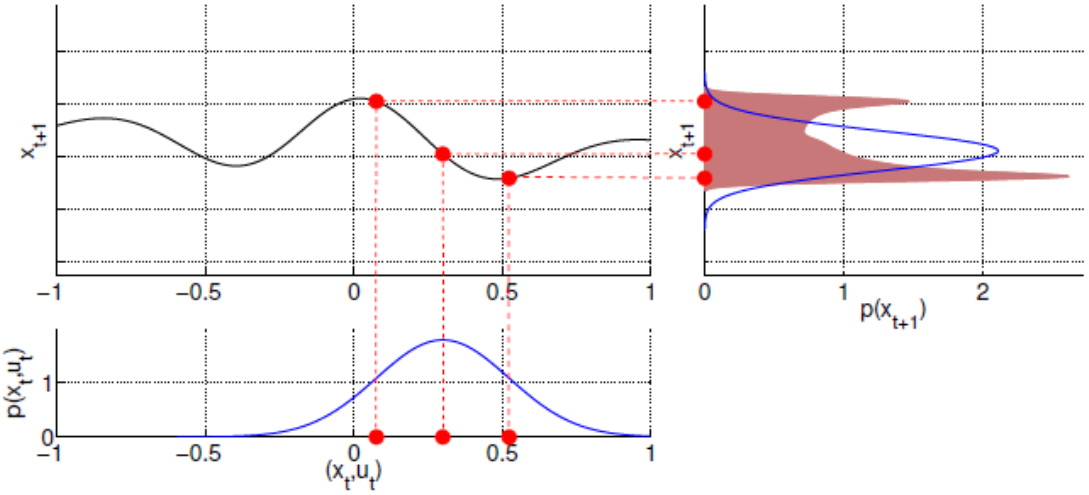
[1] فرضیات مشتق‌پذیری می‌تواند در رباتیک مشکل‌ساز باشد. برای مثال، اتصالات در جابجایی و دستکاری می‌توانند این فرض را نادرست جلوه دهند.
[2] Underestimate
[3] Aggressive

مربوطه محاسبه می‌شوند. میانگین μ_{t+1}^{x} و کوواریانس Σ_{t+1}^{x} توزیع پیش‌بینی‌کننده $p(x_{t+1})$، از نقاط سیگمای نگاشت‌شده وزن‌دار محاسبه می‌شوند. میانگین و واریانس به ترتیب از رابطه‌های زیر به دست می‌آیند

$$\mu_{t+1}^{x} = \sum_{i=0}^{2d} w_m^{[i]} f(\mathcal{X}_t^{[i]}, \mathcal{U}_t^{[i]}) \qquad \textbf{معادله(۲-۲۶)}$$

$$\Sigma_{t+1}^{x} = \sum_{i=0}^{2d} w_c^{[i]} (f(\mathcal{X}_t^{[i]}, \mathcal{U}_t^{[i]}) - \mu_{t+1}^{x})(f(\mathcal{X}_t^{[i]}, \mathcal{U}_t^{[i]}) - \mu_{t+1}^{x})^T \qquad \textbf{معادله(۲-۲۷)}$$

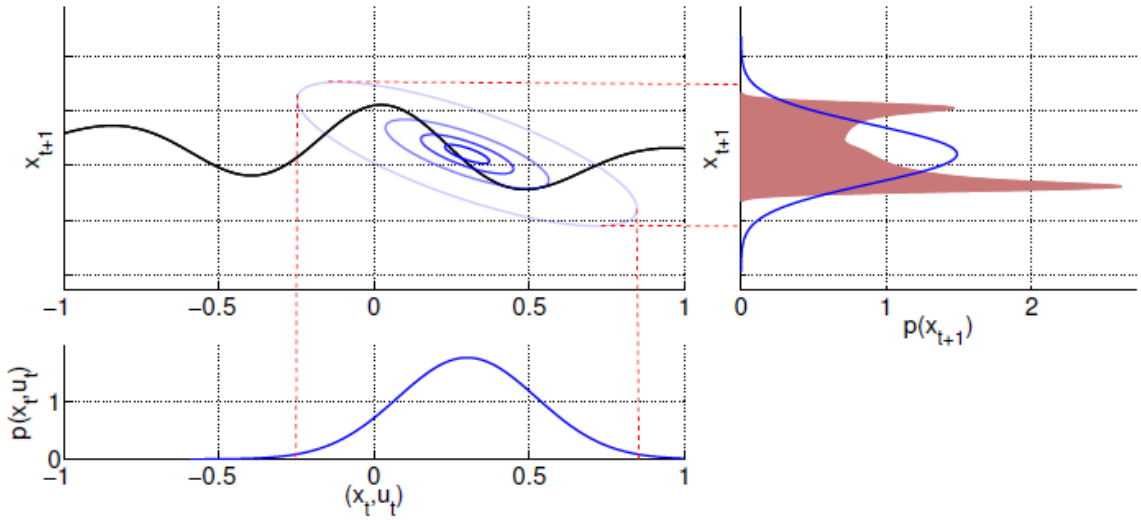
که در این روابط، d بعد (x, u) است، $\left(\mathcal{X}_t^{[i]}, \mathcal{U}_t^{[i]}\right)$ نقاط سیگما هستند به عبارت دیگر، نمونه‌هایی از توزیع توام $p(x_t, u_t)$ هستند که به طور قطعی انتخاب شده‌اند و $w_m^{[i]}$ و $w_c^{[i]}$ وزن‌ها هستند. برای جزئیات بیش‌تر در مورد انتقال بی‌رنگ و بو، به مرجع [111] مراجعه شود. شکل ۲-۹ ایده انتقال بی‌رنگ و بو را نشان می‌دهد.



شکل ۲-۹ محاسبه توزیع پیش‌بینی‌شده تقریبی با استفاده از انتقال بی‌رنگ و بو. یک توزیع گوسی $p(x_t, u_t)$ (پنل سمت چپ پایین) باید به واسطه یک تابع غیرخطی نگاشت شود (نمودار مشکلی در پنل سمت چپ بالا). توزیع پیش‌بینی‌کننده درست در ناحیه سایه خورده در پنل سمت راست نشان داده شده است. برای به دست آوردن یک تقریب گوسی از توزیع سایه خورده سنگین، توزیع ورودی با سه نقطه سیگما نشان داده شده است (نقاط قرمز در پنل سمت چپ پایین). سپس، نقاط سیگما به واسطه تابع غیرخطی نگاشت می‌شوند (پنل سمت چپ بالا) و میانگین و کوواریانس نمونه‌اشان، توزیع پیش‌بینی‌کننده تقریبی $p(x_{t+1})$ را نتیجه می‌دهد که در پنل سمت راست نشان داده شده است.[90]

باید توجه داشت که انتقال بی‌رنگ و بو توزیع گوسی $p(x_t, u_t)$ را با نقاط سیگما تقریب می‌زند، نقاطی که سپس به واسطه تابع انتقال اصلی f نگاشت می‌شوند. انتقال بی‌رنگ و بو به مشتق‌پذیری نیاز ندارد و انتظار می‌رود که نسبت به خطی‌سازی صریح، تقریب‌های دقیق‌تری از توزیع پیش‌بینی‌کننده $p(x_{t+1})$ نتیجه دهد[112].

تطبیق ممان. ایده تطبیق ممان، محاسبه دقیق میانگین و کوواریانس پیش‌بینی‌کننده $p(x_{t+1})$ و تقریب $p(x_{t+1})$ با یک گوسی است که آن میانگین و کوواریانس دقیق به آن تعلق دارد. در اینجا، نه توزیع توام $p(x_t, u_t)$ و نه توزیع انتقال f تقریب زده می‌شوند. تقریب تطبیق ممان، از این لحاظ بهترین تقریب تک‌مدی از توزیع پیش‌بینی‌کننده است که انحراف Kullback-Leibler بین توزیع پیش‌بینی‌کننده درست و تقریب تک‌مدی را کمینه می‌کند[113]. شکل ۲-۱۰ ایده تطبیق ممان را نشان می‌دهد.



***شکل ۲-۱۰* محاسبه توزیع پیش‌بینی‌شده تقریبی با استفاده از تطبیق ممان. یک توزیع گوسی $p(x_t, u_t)$ (پنل سمت چپ پایین) باید به واسطه یک تابع غیرخطی نگاشت شود (نمودار مشکی در پنل سمت چپ بالا). توزیع پیش‌بینی‌کننده درست با ناحیه سایه خورده در پنل راست نشان داده شده است. برای به دست آوردن یک تقریب گوسی از توزیع سایه خورده سنگین، میانگین و کوواریانس توزیع سایه خورده به صورت تحلیلی محاسبه می‌شوند. این ممان‌های مرتبه اول و دوم، به طور کامل توزیع پیش‌بینی‌کننده تقریبی گوسی $p(x_{t+1})$ را معین می‌کنند که نمودار آن به رنگ آبی در پنل راست نشان**

داده شده است. خطوط ترازی که در پنل سمت چپ بالا نشان داده شده‌اند توزیع توام بین ورودی‌ها و پیش‌بینی را نشان می‌دهند.[90]

- **ملاحظات عملی**

ممان‌های دقیق فقط در موارد خاص می‌توانند محاسبه شوند چون انتگرال‌های مورد نیاز برای محاسبه میانگین و کوواریانس پیش‌بینی‌کننده ممکن است لاینحل باشند. علاوه بر این، به طور معمول تقریب دقیق تطبیق ممان از لحاظ محاسباتی پرهزینه‌تر از تقریب‌ها به وسیله خطی‌سازی یا نقاط سیگماست.

برخلاف رویکردهای مبتنی بر نمونه‌برداری از قبیل PEGASUS، روش‌های استنتاج تقریبی قطعی برای برنامه‌ریزی بلندمدت می‌توانند به منظور یادگیری چندین هزار پارامتر سیاست استفاده شوند[3]. دلیلی که پیش‌بینی‌های بلندمدت قطعی می‌توانند سیاست‌هایی با تعداد زیادی پارامتر را یاد بگیرند این است که گرادیان‌ها می‌توانند به صورت تحلیلی محاسبه شوند. بنابراین، این تخمین‌های گرادیان از واریانس‌های بزرگ رنج نمی‌برند که یک مشکل معمول در تخمین مبتنی بر نمونه‌برداری است. با این وجود، غالبا استنتاج قطعی نسبت به رویکردهای نمونه‌برداری به تلاش بیش‌تری برای پیاده‌سازی نیاز دارد.

۳-۳-۲-۲- به‌روز رسانی‌های سیاست

با معرفی دو دسته مدل بزرگ و دو روش کلی انجام پیش‌بینی‌های بلندمدت با این مدل‌ها، در ادامه در مورد روش‌های به‌روز رسانی سیاست بحث می‌کنیم و وجوه اختلاف بین به‌روز رسانی‌های سیاست فارغ از گرادیان و مبتنی بر گرادیان را برمی‌شماریم.

- **به‌روز رسانی‌های سیاست مبتنی بر مدل بدون اطلاعات گرادیان**

روش‌های فارغ از گرادیان احتمالا آسان‌ترین راه به‌روز رسانی سیاست هستند چون به محاسبه و تخمین گرادیان‌های سیاست نیازی ندارند. اصولا این روش‌ها محدودیت‌های مشتق‌پذیری روی سیاست یا مدل انتقال ندارند. روش بهینه‌سازی فارغ از گرادیان استاندارد عبارتند از روش Nelder-Mead [114] که یک روش هیوریستیک ساده است و یا تپه‌نوردی[1] که یک روش جستجوی محلی است که به طور تنگاتنگ به تابکاری شبیه‌سازی شده[2][115]

[1] Hill-climbing
[2] Simulated annealing

مرتبط است. به خاطر سادگیشان و تلاش محاسباتی کمی که نیاز دارند، به طور رایج در زمینه جستجوی سیاست مبتنی بر مدل[107] و به‌خصوص در ترکیب با تولید تراژکتوری مبتنی بر نمونه‌برداری استفاده می‌شوند.

اشکال واضح بهینه‌سازی فارغ از گرادیان، نرخ همگرایی نسبتا کند آن است. برای همگرایی سریع‌تر، می‌توانیم از به‌روز رسانی‌های سیاست مبتنی بر گرادیان استفاده کنیم که در بخش‌های بعدی معرفی می‌شوند.

- **به‌روز رسانی‌های سیاست مبتنی بر مدل با اطلاعات گرادیان**

انتظار می‌رود که به‌روز رسانی‌های سیاست مبتنی بر گرادیان همگرایی سریع‌تری از به‌روز رسانی‌های فارغ از گرادیان داشته باشند. بین دو مورد تمایز قائل می‌شویم: تخمین مبتنی بر نمونه گرادیان‌های سیاست و محاسبه تحلیلی گرادیان‌های سیاست $dJ_\theta(\theta)/d\theta$.

- **گرادیان‌های سیاست مبتنی بر نمونه‌برداری**

زمانی که از نمونه تراژکتوری‌های $\tau^{[i]}$ از مدل آموخته شده استفاده می‌کنیم تا پاداش بلندمدت مورد انتظار J_θ در معادله(۲-۱۰) را تخمین بزنیم، می‌توانیم گرادیان $dJ_\theta/d\theta$ را به صورت عددی تقریب بزنیم.

آسان‌ترین روش برای تخمین گرادیان‌ها، استفاده از روش‌های تفاضل متناهی (محدود) است. اما، روش‌های تفاضل متناهی به تعداد $O(F)$ ارزیابی پاداش بلندمدت مورد انتظار نیاز دارند که در آن F تعداد پارامترهای سیاست θ است. از آنجایی که هر یک از این ارزیابی‌ها مبتنی بر میانگین m نمونه roll-out است، تعداد نمونه تراژکتوری‌های مورد نیاز بیش از حد می‌شود. در تنظیم مبتنی بر مدل، این فقط یک مشکل محاسباتی است اما مشکلی برای فرسوده کردن ربات نیست چون نمونه‌ها از مدل تولید می‌شوند نه از خود ربات.

چندین روش برای کاراتر کردن تخمین گرادیان مبتنی بر مدل وجود دارد که عبارتند از: اولا، برای تخمین قوی‌تر از J_θ یا به عبارت دیگر تخمینی با واریانس کوچک‌تر، رویکرد PEGASUS [103] می‌تواند استفاده شود. ثانیا، برای تخمین کاراتر گرادیان، هر کدام از روش‌های فارغ از مدل برای تخمین گرادیان می‌توانند در زمینه مبتنی بر مدل استفاده شوند. تنها تفاوت این است که به جای ربات، از مدل آموخته‌شده برای تولید تراژکتوری‌ها استفاده می‌شود.

- **گرادیان‌های تحلیلی سیاست**

محاسبه تحلیلی گرادیان‌های $dJ_\theta/d\theta$ نیاز دارد که سیاست، (امید ریاضی) تابع پاداش و مدل انتقال آموخته‌شده مشتق‌پذیر باشند. با وجود این محدودیت، محاسبات تحلیلی گرادیان به دو دلیل، جایگزینی پایا برای گرادیان‌های مبتنی بر نمونه‌برداری هستند: اولا، از واریانس نمونه‌برداری رنج نمی‌برند، چیزی که به خصوص هنگام محاسبه گرادیان‌ها نمود پیدا می‌کند. ثانیا، به طور مطلوب تلاش محاسباتی با تعداد پارامترهای سیاست مقیاس می‌شود که این موضوع امکان یادگیری سیاست‌ها با هزاران پارامتر را می‌دهد. اما به خاطر اعمال مکرر قاعده زنجیره‌ای، غالبا محاسبه خود گرادیان از لحاظ ریاضیاتی پیچیده‌تر از یک تخمین مبتنی بر نمونه‌برداری است.

یک مثال را در نظر بگیرید که پاداش آنی r فقط به حالت بستگی دارد (تعمیم به کنترل‌های وابسته به پاداش سرراست است) و دینامیک سیستم قطعی است، به طوری که $x_{t+1} = f(x_t, u_t) = f(x_t, \pi_\theta(x_t, \theta))$ است که در آن f یک تابع انتقال (غیرخطی)، π_θ سیاست (قطعی) و θ پارامترهای سیاست هستند. گرادیان پاداش بلندمدت $J_\theta = \sum_t \gamma^t r(x_t)$ نسبت به پارامترهای سیاست با اعمال مکرر قاعده زنجیره‌ای به دست می‌آید:

$$\frac{dJ_\theta}{d\theta} = \sum_t \gamma^t \frac{dr(x_t)}{d\theta} = \sum_t \gamma^t \frac{dr(x_t)}{dx_t}\frac{dx_t}{d\theta} \qquad \textbf{معادله(۲-۲۸)}$$

$$= \sum_t \gamma^t \frac{dr(x_t)}{dx_t}\left(\frac{\partial x_t}{\partial x_{t-1}}\frac{\partial x_{t-1}}{d\theta} + \frac{\partial x_t}{\partial u_{t-1}}\frac{\partial u_{t-1}}{d\theta}\right) \qquad \textbf{معادله(۲-۲۹)}$$

با توجه به این معادلات مشاهده می‌کنیم که مشتق کلی $dx_t/d\theta$، به مشتق کلی $dx_{t-1}/d\theta$ در گام زمانی قبلی بستگی دارد. بنابراین، مشتق $dJ_\theta/d\theta$ می‌تواند با تکرار محاسبه شود.

تعمیم به مدل‌های احتمالاتی و MDP های تصادفی. برای بسط مشتق‌ها به MDP های تصادفی و/یا مدل‌های احتمالاتی، باید تغییرات کمی در گرادیان‌های معادلات معادله(**۲-۲۸**)-معادله(**۲-۲۹**) ایجاد کنیم: زمانی که حالت x_t با یک توزیع احتمالاتی $p(x_t)$ نشان داده می‌شود باید امید ریاضی پاداش را محاسبه کنیم $\mathbb{E}[r(x_t)] = \int r(x_t)p(x_t)dx_t$. علاوه بر این، نیاز است که مشتق‌ها را نسبت به پارامترهای توزیع حالت محاسبه کنیم، با این فرض که $p(x_t)$ بازنمایی پارامتری دارد.

برای مثال، اگر $p(x_t) = \mathcal{N}(x_t|\mu_t^x, \Sigma_t^x)$ باشد، مشتقات $\mathbb{E}[r(x_t)]$ را نسبت به میانگین μ_t^x و کوواریانس Σ_t^x توزیع حالت محاسبه کرده و اعمال قاعده زنجیره‌ای را مشابه معادلات معادله(**۲-۲۸**)-معادله(**۲-۲۹**) ادامه می‌دهیم:
با تعریف $\mathcal{E}_t := \mathbb{E}_{x_t}[r(x_t)]$، گرادیان را به صورت زیر به دست می‌آوریم

$$\frac{dJ_\theta}{d\theta} = \sum_t \gamma^t \frac{d\mathcal{E}_t}{d\theta} \ ,$$

$$\frac{d\mathcal{E}_t}{d\theta} = \frac{d\mathcal{E}_t}{dp(x_t)}\frac{dp(x_t)}{d\theta} := \frac{d\mathcal{E}_t}{d\mu_t^x}\frac{d\mu_t^x}{d\theta} + \frac{d\mathcal{E}_t}{d\Sigma_t^x}\frac{d\Sigma_t^x}{d\theta} \qquad \textbf{معادله(۲-۳۰)}$$

که در آن، از نمادگذاری مختصر $d\mathcal{E}_t/dp(x_t) = \{d\mathcal{E}_t/d\mu_t^x \ , d\mathcal{E}_t/d\Sigma_t^x\}$ برای گرفتن مشتق (کلی) $\mathcal{E}_t$ نسبت به پارامترهای $p(x_t) = \mathcal{N}(x_t|\mu_t^x, \Sigma_t^x)$ (و یا به‌عبارت دیگر میانگین و کوواریانس) استفاده کرده‌ایم. میانگین μ_t^x و کوواریانس Σ_t^x از لحاظ کاربردی به ممان‌های μ_{t-1}^x و Σ_{t-1}^x از $p(x_{t-1})$ و پارامترهای θ کنترل‌کننده وابسته هستند. با اعمال قاعده زنجیره‌ای به معادله(۲-۳۰) به دست می‌آوریم:

$$\frac{d\mu_t^x}{d\theta} = \frac{d\mu_t^x}{d\mu_{t-1}^x}\frac{d\mu_{t-1}^x}{d\theta} + \frac{d\mu_t^x}{d\Sigma_{t-1}^x}\frac{d\Sigma_{t-1}^x}{d\theta} + \frac{d\mu_t^x}{d\theta} \qquad \textbf{معادله(۲-۳۱)}$$

$$\frac{d\Sigma_t^x}{d\theta} = \frac{d\Sigma_t^x}{d\mu_{t-1}^x}\frac{d\mu_{t-1}^x}{d\theta} + \frac{d\Sigma_t^x}{d\Sigma_{t-1}^x}\frac{d\Sigma_{t-1}^x}{d\theta} + \frac{d\Sigma_t^x}{d\theta} \qquad \textbf{معادله(۲-۳۲)}$$

توجه داشته باشید که مشتقات کلی $d\mu_{t-1}^x/d\theta$ و $d\Sigma_{t-1}^x/d\theta$ از گام زمانی $t-1$ معلوم هستند.

اگر همه این محاسبات بتوانند به فرم بسته انجام شوند، گرادیان‌های سیاست $d\tilde{J}_\theta/d\theta$ می‌توانند با اعمال مکرر قاعده زنجیره‌ای به صورت تحلیلی و بدون نیاز به نمونه‌برداری محاسبه شوند. بنابراین، تکنیک‌های استاندارد بهینه‌سازی (برای مثال BFGS یا CG) می‌توانند برای یادگیری سیاست‌ها با هزاران پارامتر استفاده شوند[3].

- **مباحثه**

استفاده از گرادیان‌های امید ریاضی پاداش بلندمدت J_θ نسبت به پارامترهای سیاست θ غالبا منجر به یادگیری سریع‌تر از به‌روز رسانی‌های سیاست فارغ از گرادیان می‌شود. علاوه بر این، روش‌های فارغ از گرادیان به طور معمول محدود به ده‌ها پارامتر سیاست هستند. محاسبه گرادیان‌ها می‌تواند سنگین باشد و به منابع محاسباتی اضافی نیاز دارد. زمانی که گرادیان‌ها را محاسبه می‌کنیم، گرادیان‌های تحلیلی دقیق به گرادیان‌های مبتنی بر نمونه‌برداری ترجیح داده می‌شوند چون مورد دوم غالبا از واریانس بزرگ رنج می‌برد. این واریانس‌ها می‌توانند حتی منجر به

همگرایی کندتر از به‌روز رسانی‌های سیاست فارغ از مدل[107] شوند. برای گرادیان‌های تحلیلی، فرضیاتی برای مشتق‌پذیری تابع پاداش r و تابع انتقال f در نظر می‌گیریم. علاوه بر این، برای گرادیان‌های تحلیلی، فقط به روش‌های استنتاج تقریبی قطعی از قبیل تطبیق ممان یا خطی‌سازی متکی هستیم به طوری که فقط تقریب $\tilde{J}_\theta$ به J_θ می‌تواند محاسبه شود؛ اما فقط با گرادیان‌های دقیق $d\tilde{J}_\theta/d\theta$.

برای به‌روز رسانی سیاست، استفاده از اطلاعات گرادیان را توصیه می‌کنیم تا ویژگی‌های بهتر همگرایی استفاده شوند. به طور ایده‌آل، گرادیان‌ها به صورت تحلیلی و بدون هیچ تقریبی معین می‌شوند. از آنجایی که فقط برای سیستم‌های خطی می‌توان به این هدف رسید باید به تقریب‌ها (با استفاده از رویکردهای مبتنی بر نمونه‌برداری یا گرادیان‌های تقریبی تحلیلی) متوسل شویم. رویکردهای مبتنی بر نمونه‌برداری عملا محدود به پارامترهای سیاست نسبتا با ابعاد پایین ($\theta \in \mathbb{R}^k, k \leq 50$) هستند. برای پارامترهای سیاست با ابعاد بالا (با $k > 50$)، استفاده از گرادیان‌های سیاست تحلیلی را (در صورتی که در دسترس باشند) توصیه می‌کنیم.

۴-۳-۲-۲- الگوریتم‌های جستجوی سیاست مبتنی بر مدل با کاربردهای رباتیکی

در این بخش، به صورت خلاصه روش‌های جستجوی سیاستی را شرح می‌دهیم که با موفقیت به یادگیری سیاست‌ها برای ربات‌ها اعمال شده‌اند. تمایز بین رویکردهایی که امید ریاضی پاداش بلندمدت J_θ را با استفاده از روش‌های نمونه‌برداری ارزیابی می‌کنند و رویکردهایی که آن را با استفاده از روش‌های استنتاج تقریبی قطعی ارزیابی می‌کنند شرح می‌دهیم.

- **پیش‌بینی مبتنی بر نمونه‌برداری تراژکتوری**

نمونه‌برداری مستقیم از شبیه‌ساز آموخته‌شده، توسط یک سری از محققان برای حرکت[1] هلی‌کوپترها[107] و برای کنترل بالون‌ها[105] مورد بررسی قرار گرفته است. همه رویکردها از الگوریتم PEGASUS [103] استفاده کرده‌اند تا تراژکتوری‌ها را از مدل‌های تصادفی آموخته‌شده تولید کنند.

[1] Maneuvering

ان‌جی و همکاران[106]، مدل‌هایی برای حرکت در جای هلی‌کوپتر یاد گرفته‌اند که مبتنی بر رگرسیون خطی وزن‌دار محلی است. برای در نظر گرفتن نویز و بی‌دقتی‌های مدل، با اضافه کردن نویز (سیستمی) گوسی i.i.d به دینامیک انتقال، این مدل اصالتا قطعی، تصادفی شده است.

برخلاف مقاله [106]، باگنل و اشنایدر[107] عدم قطعیت در مورد مدل آموخته‌شده را به وسیله یک توزیع پسین روی مجموعه‌ای متناهی از مدل‌های affine محلی به‌طور صریح شرح داده‌اند. برای یادگیری سیاست، تراژکتوری‌ها از این ترکیب مدل‌ها نمونه‌برداری شده‌اند.

کو و همکاران[105]، مدل‌های پارامتری ایده‌آل‌شده را با فرآیندهای گوسی غیرپارامتری ترکیب کرده‌اند تا دینامیک بالون خودمختار را مدلسازی کنند. GP ها برای مدلسازی اختلاف بین مدل پارامتری غیرخطی بالون و داده‌ها استفاده شده‌اند. هنگام یادگیری سیاست، تراژکتوری‌ها از این مدل آمیخته[1] نمونه‌برداری می‌شوند. در ادامه، در مورد این الگوریتم‌های جستجوی سیاست بحث می‌کنیم.

- **مدل‌های پیشرو رگرسیون وزن‌دار محلی و پیش‌بینی مبتنی بر نمونه‌برداری تراژکتوری**

در مرجع [107]، رگرسیون بیزی وزن‌دار محلی به منظور یادگیری مدل‌های پیشرو برای حرکت در جای یک هلی‌کوپتر خودمختار استفاده شده است. برای در نظر گرفتن عدم قطعیت مدل، به جای تخمین نقطه‌ای از پارامترهای مدل، یک توزیع پسین احتمالاتی روی پارامترهای ψ مدل و از این رو، روی خود مدل در نظر گرفته شده است. برای یادگیری سیاست ، تراژکتوری‌ها از این ترکیب مدل‌ها نمونه‌برداری شده‌اند.

تراژکتوری‌ها $\tau^{[i]}$ با استفاده از رویکرد PEGASUS [103] تولید شده‌اند. در هر گام زمانی، یک مجموعه پارامتر ψ_i مدل از توزیع پسین $p(\psi|X, U, y, x_t, u_t)$ نمونه‌برداری شده است. بعد از هر انتقال، مدل دینامیکی با انتقال (شبیه‌سازی شده) مشاهده شده به‌روز رسانی می‌شود. بعد از هر تراژکتوری تولید شده $\tau^{[i]}$، با حذف تراژکتوری شبیه‌سازی شده $\tau^{[i]}$ از مدل، مدل بازنشانی (راه‌اندازی مجدد) می‌شود[107]. برای تخمین‌گرهای مدل بهینه بیز[2]، این رویه معادل با نمونه‌برداری مدل و نمونه‌برداری یک تراژکتوری کامل از آن است. الگوریتم ۲-۶ به طور مختصر

[1] Hybrid
[2] Bayes-optimal

بیان کرده است که باید چگونه تراژکتوری‌ها را از مدل آموخته‌شده نمونه‌برداری کرد در حالی که عدم قطعیت پسین در مورد خود مدل تلفیق شده است. با میانگین گرفتن روی امید ریاضی پاداش‌های بلندمدت برای همه تراژکتوری‌های تولید شده $\tau^{[i]}$، عدم قطعیت مدل به صورت ضمنی ادغام می‌شود.

Input: transition model f, posterior distribution over model parameters $p(\psi|X,U,y)$, policy parameters θ
for $i = 1, \dots, m$ do
  for $t = 0, \dots, T-1$ do ▷ Sample trajectory $\tau^{[i]}$
    Sample local model parameters $\psi_i \sim p(\psi|X,U,y,x_t,u_t)$
    Compute control $u_t = \pi_\theta(x_t)$
    Generate a sample state transition $x_{t+1} \sim p(x_{t+1}|x_t,u_t,\psi_i)$
    Update X , U with simulated transition (x_t,u_t,x_{t+1})
  end for
  Compute $J_{\theta,i}$
  Reset the learned model to the original model f
end for

$$\tilde{J}_\theta = \frac{1}{m}\sum_{i=1}^{m} J_{\theta,i}$$

الگوریتم ۲-۶ الگوریتم ارزیابی سیاست و پیش‌بینی‌های T-گامی[107].

در مرجع [107]، روش Nelder-Mead که یک بهینه‌سازی فارغ از گرادیان است، برای به‌روز رسانی پارامترهای θ سیاست استفاده شده است که از بهینه‌سازی مبتنی بر گرادیان ساده عملکرد بهتری داشته است. رویکرد حاصله از یک کنترل‌کننده شبکه عصبی با ده پارامتر استفاده کرده تا یک هلی‌کوپتر را حول یک نقطه ثابت حرکت دهد[107]، شکل ۲-۱۱(الف). با جریمه‌های زیاد روی حالت‌های مربوطه، از برون‌یابی خارج از محدوده داده جمع‌آوری شده جلوگیری شده است. این کنترل‌کننده آموخته شده که مبتنی بر توزیع احتمالاتی روی مدل‌هاست، به طور چشم‌گیر کم‌نوسان‌تر از کنترل‌کننده‌ای است که با استفاده از مدل بیشینه درست‌نمایی و یا به عبارت دیگر تخمین نقطه‌ای از پارامترهای مدل، آموخته شده است.

(الف) حرکت درجای هلی‌کوپتر[106]

(ب) حرکت در جای هلی‌کوپتر وارونه[107]

شکل ۲-۱۱ روش‌های جستجوی سیاست مبتنی بر مدل با استنتاج تصادفی که برای یادگیری حرکت در جای هلی‌کوپتر استفاده شده‌اند.[90]

ان‌جی و همکاران[106]، مدل‌هایی مبتنی بر رگرسیون خطی وزن‌دار محلی برای حرکت درجای هلی‌کوپتر یاد گرفته‌اند، شکل ۲-۱۱(ب). برخلاف مرجع [107]، تخمین نقطه‌ای از پارامترها ψ (برای نمونه با بیشینه درست‌نمایی یا تخمین بیشینه پسین) در معادله(۲-۱۱) تعیین شده است. برای در نظر گرفتن نویز و بی‌دقتی‌های مدل، با اضافه کردن نویز (سیستمی) گوسی i.i.d به دینامیک انتقال، این مدل اصالتا قطعی، تصادفی شده است. سرعت‌های زاویه‌ای که در مختصات هلی‌کوپتر بیان شده‌اند انتگرال گرفته شده و سپس به زوایایی در مختصات جهانی منتقل شده‌اند، کاری که مدل را غیرخطی می‌کند. با این رویکرد، مدل‌های مربوط به حرکت درجای هلی‌کوپتر در وضعیت استاندارد[104] یا وارونه[106]، با استفاده از داده جمع‌آوری شده از تراژکتوری‌های (انسان) خلبان‌ها تعیین شده‌اند.

برای یادگیری یک کنترل‌کننده با این دینامیک انتقال غیرخطی تصادفی، به‌منظور نمونه‌برداری تراژکتوری‌ها از مدل، از روش نمونه‌برداری PEGASUS [103] استفاده شده است. با این تراژکتوری‌های نمونه‌برداری شده، یک تخمین Monte Carlo از امید ریاضی پاداش بلندمدت محاسبه شده است. یک روش تپه‌نوردی حریصانه برای یادگیری پارامترهای θ سیاست π_θ استفاده شده است که با یک شبکه عصبی ساده‌شده نشان داده شده است[106].

در مورد حرکت در جای هلی‌کوپتر وارونه، یک (انسان) خلبان هلی‌کوپتر را وارونه کرده است. سپس، کنترل‌کننده آموخته‌شده کنترل را در دست گرفته و هلی‌کوپتر را در موقعیت وارونه پایدار کرده است[106] که مثالی از آن در شکل ۲-۱۱ (ب) نشان داده شده است.

- **مدل‌های پیشرو فرآیند گوسی و پیش‌بینی مبتنی بر نمونه‌برداری تراژکتوری**

در مرجع [105]، مدل‌های پیشرو GP برای مدلسازی دینامیک انحراف[1] بالون خودمختار یاد گرفته شده‌اند، شکل ۲-۱۲. مدل‌های GP با مدل پارامتری ایده‌آل‌شده دینامیک بالون ترکیب شده‌اند، به عبارت دیگر GP اختلاف بین مدل غیرخطی پارامتری و داده مشاهده‌شده را مدلسازی می‌کند. این مدل روی داده تراژکتوری بالون آموزش داده شده است که به وسیله انسانی تولید شده که بالون را با استفاده از کنترل از راه دور به پرواز در آورده است.

رویکرد PEGASUS [103] برای نمونه‌برداری تراژکتوری‌های بلندمدت استفاده شده است. هر نمونه جدید از طریق به‌روز رسانی ماتریس کرنل در مدل آمیخته شده است. کنترل‌کننده با استفاده از روش بهینه‌سازی فارغ از مدل Nelder-Mead [114] آموخته شده است. چهار پارامتر θ کنترل‌کننده عبارتند از ضریب کشش[2]، بهره‌های موتور راست/چپ و شیب تابع هموارکننده سیاست. کنترل‌کننده آموخته شده یک کنترل‌کننده حلقه-باز است، به عبارت دیگر کنترل‌ها برون‌خط[3] و از قبل محاسبه شده و سپس به بالون اعمال شده‌اند. کنترل‌کننده مبتنی بر دینامیک GP آموخته شده، از کنترل‌کننده بهینه‌ای که تنها مبتنی بر مدل دینامیکی پارامتری ایده‌آل‌شده بالون است[105] عملکرد بهتری دارد.

[1] Yaw-dynamics
[2] Drag coefficient
[3] Offline

شکل ۲-۱۲ ترکیب مقدم پارامتری و فرآیندهای گوسی برای مدلسازی و یادگیری کنترل یک بالون خودمختار.[105]

- **پیش‌بینی‌های قطعی تراژکتوری**

در ادامه، به طور مختصر روش‌های جستجوی سیاستی را شرح می‌دهیم که پیش‌بینی‌های قطعی تراژکتوری را برای ارزیابی سیاست انجام می‌دهند.

- **مدل‌های پیشرو فرآیند گوسی و پیش‌بینی قطعی تراژکتوری**

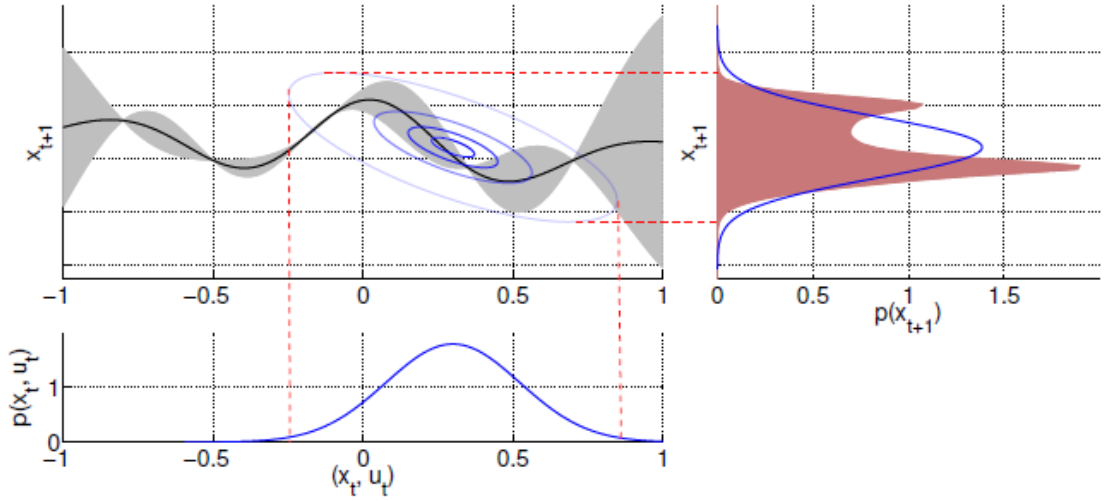
چارچوب کاری جستجوی سیاست PILCO (استنتاج احتمالاتی برای یادگیری کنترل)[3]، از یک مدل پیشرو GP مربوط به دینامیک ربات استفاده کرده تا به طور مداوم خطاهای مدل را در نظر بگیرد. در ترکیب با استنتاج قطعی به وسیله تطبیق ممان برای پیش‌بینی تراژکتوری‌های حالت $p(\tau) = (p(x_1), \cdots, p(x_T))$، PILCO یک تقریب تحلیلی $\tilde{J}$ به امید ریاضی پاداش بلندمدت J_θ در معادله(۲-۱۰) را محاسبه می‌کند. علاوه بر این، گرادیان‌های $d\tilde{J}_\theta/d\theta$ امید ریاضی پاداش بلندمدت نسبت به پارامترهای سیاست به صورت تحلیلی محاسبه می‌شوند. برای یادگیری سیاست، بهینه‌سازهای استاندارد می‌توانند استفاده شوند. الگوریتم PILCO در الگوریتم ۲-۷ به طور مختصر بیان شده است. این الگوریتم به طور معمول بدون اطلاعات مفید مقداردهی اولیه می‌شود،

به عبارت دیگر پارامترهای سیاست به طور تصادفی نمونه‌برداری می‌شوند و داده از یک تراژکتوری حالت کوتاه که با اعمال عمل‌های تصادفی تولید شده است ضبط می‌شود. در ادامه، به طور مختصر جزئیات محاسبه پیش‌بینی‌های بلندمدت، گرادیان‌های سیاست و اعمال چارچوب کاری PILCO به سیستم‌های کنترلی و رباتیکی را بیان می‌کنیم.

Init: Sample controller parameters $\theta \sim \mathcal{N}(0, I)$. Apply random control signals and record data.
repeat
  Learn probabilistic (GP) dynamics model using all data.
  repeat
    Compute $p(x_1), \ldots, p(x_T)$ using moment matching and $\tilde{J}_\theta$
    Analytically compute policy gradients $\mathrm{d}\tilde{J}_\theta / d\theta$
    Update parameters θ (line-search in BFGS or CG).
  until convergence; return θ^*
  Apply π_θ^* to system (single trial/episode) and record data.
until task learned

الگوریتم ۲-۷ چارچوب کاری جستجوی سیاست PILCO [3].

پیش‌بینی‌های بلندمدت. برای پیش‌بینی یک توزیع $p(\tau|\pi_\theta)$ روی تراژکتوری‌ها برای یک سیاست معلوم، PILCO مکررا توزیع‌های حالت $p(x_1), \cdots, p(x_T)$ را محاسبه می‌کند. برای این پیش‌بینی‌ها، عدم قطعیت پسین در مورد مدل پیشرو GP آموخته‌شده ادغام می‌شود. شکل ۲-۱۳ این سناریو را نشان می‌دهد: فرض می‌کنیم که یک توزیع توام گوسی $p(x_t, u_t)$ داده شده است. برای پیش‌بینی توزیع $p(x_{t+1})$ حالت بعدی، توزیع توام $p(x_t, u_t)$ در پنل سمت چپ-پایین باید به واسطه توزیع GP پسین روی تابع انتقال پنهان نگاشت شود که در پنل سمت راست-بالا نشان داده شده است. به خاطر تابع کوواریانس غیرخطی، استنتاج دقیق لاینحل است. نتیجه نمونه‌برداری Monte Carlo گسترده، یک تقریب نزدیک به توزیع پیش‌بینی‌کننده است که با توزیع دو مدی سایه‌خورده در پنل سمت راست نشان داده شده است. PILCO، میانگین و واریانس این توزیع سایه خورده را دقیقا محاسبه کرده و همان‌طور که با توزیع آبی در پنل سمت راست-بالا نشان داده شده است، توزیع سایه خورده را با یک گوسی با میانگین و واریانس صحیح تقریب می‌زند [3].

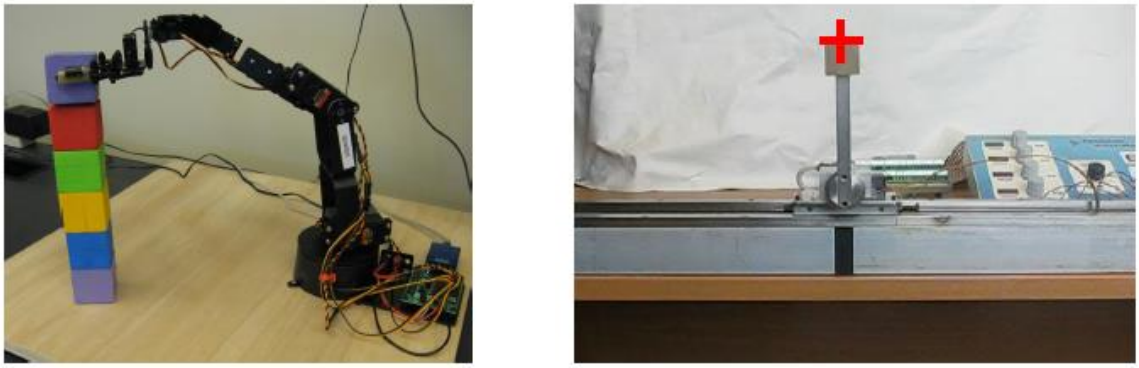


شکل ۲-۱۳ پیش‌بینی‌های تقریبی با فرآیندهای گوسی در ورودی‌های نامطمئن. برای تعیین توزیع پیش‌بینی‌کننده $p(x_{t+1})$، نیاز است که توزیع ورودی $p(x_t, u_t)$ (پنل سمت چپ پایین) را به واسطه توزیع GP پسین (پنل سمت چپ بالا) نگاشت کرد در حالی که صریحا عدم قطعیت مدل را میانگین گرفت (ناحیه سایه خورده). نمونه‌برداری جامع Monte-Carlo توزیع دقیق را نتیجه می‌دهد که با توزیع به رنگ قرمز سایه خورده نشان داده شده است (پنل سمت راست). تقریب تطبیق ممان قطعی، میانگین و واریانس توزیع پیش‌بینی‌کننده دقیق را محاسبه کرده و یک گوسی (منحنی آبی در پنل سمت راست) با دو ممان اول دقیق به آن برازش می‌کند. منحنی‌های تراز در پنل سمت چپ بالا توزیع توام بین ورودی‌ها و پیش‌بینی را نشان می‌دهند.

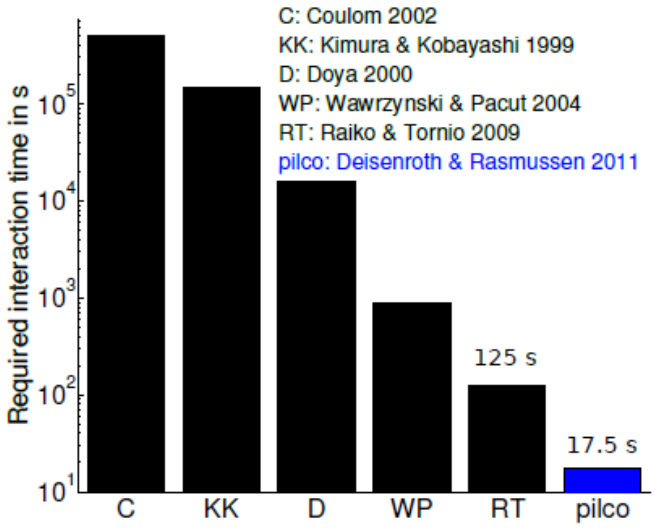
گرادیان‌های سیاست تحلیلی. حالت‌های پیش‌بینی شده تخمین‌های نقطه‌ای هستند اما با توزیع‌های احتمالاتی گوسی $p(x_t),\ t = 1, \cdots, T$ نشان داده می‌شوند. هنگام محاسبه گرادیان‌های سیاست $dJ_\theta/d\theta$، PILCO صریحا از طریق محاسبه تحلیلی گرادیان‌های سیاست برای مدل‌های احتمالاتی، فرمول‌بندی احتمالاتی را در نظر می‌گیرد.

به خاطر گنجاندن صریح عدم قطعیت مدل در پیش‌بینی‌های بلندمدت و محاسبه گرادیان، به طور معمول PILCO خیلی زیاد از خطاهای مدل رنج نمی‌برد.

کاربردهای ربات. همان‌طور که در شکل ۲-۱۵ نشان داده شده است، الگوریتم PILCO به سرعتی بی‌سابقه در یادگیری روی یک کار محک استاندارد (کار under-actuated نوسان و متعادل‌سازی cart-pole که در شکل ۲-۱۴ نشان داده شده‌اند) رسیده است.



شکل ۲-۱۴ موفقیت‌های یادگیری PILCO . سمت چپ: یادگیری خودمختار برای روی هم قرار دادن دسته‌ای از بلوک‌ها با استفاده از بازوی ارزان موجود در بازار[108]. سمت راست: یادگیری خودمختار نوسان و متعادل کردن یک آونگ آزادانه معلق که به یک ارابه متصل است[3].



شکل ۲-۱۵ مقایسه عملکرد PILCO با روش‌های دیگر. PILCO به سرعتی بی‌سابقه در یادگیری کار نوسان cart-pole می‌رسد. محور افقی به رویکردهای یادگیری تقویتی مراجعه می‌کند که کار مشابه را حل کرده‌اند، محور عمودی زمان تعامل مورد نیاز در واحد ثانیه با مقیاس لگاریتمی را نشان می‌دهد.[90]

به طور ویژه، نوسان cart-pole توسط PILCO یاد گرفته شده است که از هر روش یادگیری تقویتی دیگری که از هیچ (یا به عبارت دیگر بدون مقداردهی اولیه با اطلاعات مفید مثلا با نمایش‌ها) یاد می‌گیرد به زمان تعامل خیلی کم‌تری با ربات نیاز دارد. برای مسئله نوسان cart-pole، سیاست غیرخطی آموخته شده یک شبکه‌ی تابع پایه شعاعی با ۵۰ تابع پایه گوسی axis-aligned است. پارامترهای θ سیاست شامل وزن‌ها، موقعیت‌ها و عرض‌های توابع پایه هستند که در مجموع ۳۰۵ پارامتر سیاست می‌شوند.

همچنین، PILCO در یک کار روی هم قرار دادن بلوک توسط یک بازوی ربات ارزان با ۵ درجه آزادی، برای یادگیری کارا و موثر کنترل‌کننده‌ها از هیچ با موفقیت اعمال شده است[108]، شکل ۲-۱۴. حالت x سیستم به صورت مختصات سه بعدی بلوک در محرک (مجری) پایانی[1] بازو تعریف شده است. برای دنبال کردن این مختصات، از یک دوربین RGB-D استفاده شده است. سیاست آموخته شده π_θ یک تابع affine از حالت است، یعنی $u = Ax + b$. با پیروی دقیق از الگوریتم ۲-۷ ، PILCO روی هم قرار دادن برجی از شش بلوک را در کم‌تر از ۲۰ آزمایش یاد گرفته است. محدودیت‌های فضای حالت برای دوری از مانع نیز مستقیما در فرآیند یادگیری گنجانده شده است[108].

- **مرور الگوریتم‌های جستجوی سیاست مبتنی بر مدل**

جدول ۲-۲، رویکردهای جستجوی سیاست مبتنی بر مدلی را خلاصه کرده است که در این بخش ارائه شده‌اند. هر الگوریتم مطابق با روش پیش‌بینی(نمونه‌برداری/ قطعی)، مدل پیشرو آموخته شده(LWR/LWBR/GP)، به‌روز رسانی‌های سیاست(فارغ از گرادیان/ مبتنی بر گرادیان) و کاربردهای رباتیکی مربوطه‌اش فهرست شده است.

باید توجه داشت که در مقاله [106]، خطاهای مدل و کمینه محلی با وارد کردن نویز اضافی به سیستم مدیریت شده‌اند تا خطر بیش‌برازش کاهش یابد. به طور کلی، نویز در سیستم (چه با تزریق مصنوعی آن[106] و چه با استفاده از مدل‌های احتمالاتی[3])، تابع هدف و از همین‌رو کمینه محلی را هموار می‌کند.

[1] end-effector

جدول ۲-۲ مرور الگوریتم‌های جستجوی سیاست مبتنی بر مدل با کاربردهای رباتیکی[90].

الگوریتم	پیش‌بینی‌ها	مدل پیشرو	به‌روز رسانی سیاست	کاربرد
[107]	نمونه‌برداری (PEGASUS)	LWBR	فارغ از گرادیان	حرکت در جای هلی‌کوپتر
[106][104]	نمونه‌برداری (PEGASUS)	LWR + نویز	فارغ از گرادیان	حرکت در جای هلی‌کوپتر
[103]	نمونه‌برداری (PEGASUS)	GP	فارغ از گرادیان	کنترل بالون
[3][108]	تطبیق ممان	GP	مبتنی بر گرادیان	بازوی ربات، cart-pole

۵-۳-۲-۲- ویژگی‌های مهم روش‌های مبتنی بر مدل

در ادامه، در مورد سه موضوع مهم که به جستجوی سیاست مبتنی بر مدل مرتبط هستند بحث می‌کنیم. به ویژه، ابتدا در مورد فواید و معایب استنتاج تصادفی در برابر استنتاج قطعی بحث می‌کنیم. سپس، در مورد این بحث می‌کنیم که چگونه در منابع علمی به عدم قطعیت در مورد مدل آموخته شده پرداخته می‌شود. سرانجام، نیازمندی‌ها را مشخص می‌کنیم در هنگامی که یک سیاست از هیچ یاد گرفته می‌شود یا به عبارت دیگر یادگیری باید بدون مقداردهی اولیه مناسب رخ دهد. نکته آخر درصورتی مهم است که نه دانش مفید در مورد دینامیک در دسترس باشد و نه مجموعه داده مناسب از نمایش‌ها. در عوض، ربات باید یاد بگیرد که از داده بالقوه پراکنده و دانش پیشین (مقدم) بی‌فایده شروع کند.

- **پیش‌بینی‌های بلندمدت تصادفی و قطعی**

در مورد دو رویکرد کلی مبتنی بر مدل برای محاسبه توزیع‌ها روی تراژکتوری‌ها و پاداش بلندمدت مربوطه بحث کردیم: نمونه‌برداری Monte-Carlo از ترفند PEGASUS استفاده کرده و پیش‌بینی‌های قطعی از

خطی‌سازی، انتقال بی‌رنگ و بو یا تطبیق ممان استفاده می‌کنند. فایده نمونه‌برداری تصادفی این است که نمونه‌بردار، تخمینی درست از امید ریاضی پاداش بلندمدت J_θ در حد[1] تعداد نامتناهی از تراژکتوری‌های نمونه‌برداری شده بازخواهد گرداند. نمونه‌برداری جامع می‌تواند از لحاظ محاسباتی کارا نباشد، اما به صورت سر راست می‌تواند موازی شود. یک معضل مهم‌تر در رابطه با نمونه‌برداری، حتی هنگام استفاده از رویکرد PEGASUS [103]، این است که این روش فقط برای چند ده پارامتر سیاست عملی و کاربردی است.

به عنوان یک جایگزین برای نمونه‌برداری تصادفی، پیش‌بینی‌های قطعی فقط یک توزیع تراژکتوری دقیق برای سیستم‌های خطی- گوسی را محاسبه می‌کنند. بنابراین، در سیستم‌های غیرخطی، فقط تقریبی از امید ریاضی پاداش بلندمدت بازگردانده می‌شود. محاسباتی که برای محاسبه توزیع‌های پیش‌بینی‌کننده نیاز است غیربدیهی هستند و می‌توانند از لحاظ محاسباتی پرهزینه باشند. برخلاف نمونه‌برداری تصادفی، پیش‌بینی‌های قطعی به صورت سرراست قابل موازی کردن نیستند. از طرفی دیگر، پیش‌بینی‌های قطعی فواید متعددی دارند که می‌تواند بر معایبش بچربد: اولا، با وجود این حقیقت که پیش‌بینی‌های قطعی از لحاظ محاسباتی پرهزینه‌تر از تولید یک انتقال ساده هستند نیاز به تعداد زیادی نمونه از لحاظ محاسباتی حتی پرهزینه‌تر می‌شود. فایده جالب توجه پیش‌بینی‌های قطعی این است که گرادیان‌های نسبت به پارامترهای سیاست می‌توانند به صورت تحلیلی محاسبه شوند. بنابراین، جستجوی سیاست با روش‌های پیش‌بینی قطعی می‌تواند سیاست‌های با هزاران پارامتر را یاد بگیرد[3].

جدول ۲-۳ ویژگی‌های پیش‌بینی‌های قطعی و تصادفی تراژکتوری را خلاصه کرده است. این جدول فهرست کرده که آیا امید ریاضی پاداش بلندمدت J_θ و گرادیان‌های مربوطه $dJ_\theta/d\theta$ می‌توانند دقیقا ارزیابی شده یا فقط می‌توانند تقریبا ارزیابی شوند. برای پیش‌بینی‌های تصادفی تراژکتوری یا به عبارت دیگر نمونه‌برداری، محاسبات مورد نیاز نسبتا ساده هستند در حالی که محاسبات مربوط به پیش‌بینی‌های قطعی از لحاظ ریاضیاتی پیچیده‌تر هستند. در آخر، حدود ممکن روی تعداد پارامترهای سیاست که با استفاده از هر یک از روش‌های پیش‌بینی می‌توانند یاد گرفته شوند را ارائه می‌دهیم. برای تولید تراژکتوری تصادفی، J_θ می‌تواند دقیقا در حد

[1] In the limit

تعداد نامتناهی نمونه تراژکتوری ارزیابی شود. گرادیان‌های سیاست مربوطه حتی کندتر همگرا می‌شوند. در عمل که تعداد متناهی نمونه در دسترس است هر دو J_θ و $dJ_\theta/d\theta$ نمی‌توانند دقیقا ارزیابی شوند.

جدول ۲-۳ ویژگی‌های پیش‌بینی‌های قطعی و تصادفی تراژکتوری در جستجوی سیاست مبتنی بر مدل[90].

	تصادفی	**قطعی**
J_θ	دقیق در حد	تقریبی
$dJ_\theta/d\theta$	دقیق در حد	دقیق
محاسبات	ساده	پیچیده
تعداد پارامترهای سیاست	$1 \leq \lvert\theta\rvert \leq 50$	$1 \leq \lvert\theta\rvert \leq ?$

- **پرداختن به عدم قطعیت مدل**

بیان عدم قطعیت در مورد مدل آموخته‌شده برای جستجوی سیاست مبتنی بر مدل از این نظر مهم است که در مقابل خطاهای مدل مقاوم باشد. هنگام پیش‌بینی یا تولید تراژکتوری‌ها، دو روش کلی برای پرداختن به عدم قطعیت مدل وجود دارد.

در مقاله [3]، به عدم قطعیت مدل به عنوان نویز ناهمبسته از لحاظ زمانی پرداخته شده است، به عبارت دیگر خطاهای مدل در هر گام زمانی مستقل در نظر گرفته شده‌اند. این رویکرد از لحاظ محاسباتی نسبتا ارزان بوده و امکان در نظر گرفتن تعداد نامتناهی مدل در طول میانگین‌گیری مدل را می‌دهد[3]. در عوض، نمونه‌برداری از پارامترهای مدل در ابتدا و تثبیت کردن آنها برای تولید تراژکتوری این فایده را دارد که زمانی که تراژکتوری‌های به طور نسبی نمونه‌برداری شده به عنوان داده آموزشی در نظر گرفته شده تا زمانی که پارامترهای مدل دوباره نمونه‌برداری شوند به طور خودکار همبستگی زمانی در نظر گرفته می‌شود[107]. در اینجا منظور از همبستگی زمانی این است که حالت یا وضعیت در یک گام زمانی در طول یک تراژکتوری، با حالت در گام زمانی قبلی همبسته است. از طرفی دیگر، فقط تعداد متناهی (محدود) مدل می‌توانند نمونه‌برداری شوند.

- **ویژگی‌های برون‌یابی مدل‌ها**

در جستجوی سیاست مبتنی بر مدل فرض می‌شود که مدل‌ها معلوم هستند یا در گام پیش‌پردازش آموزش داده شده‌اند[104]. در اینجا، از انسان‌ها خواسته شده که ربات (برای مثال یک هلی‌کوپتر یا یک بالون) را حرکت داده تا برای ساخت مدل داده جمع‌آوری شود. یک جنبه بسیار مهم داده جمع‌آوری شده این است که مناطقی از فضای حالت را پوشش می‌دهد که برای یادگیری موفق کار پیش‌رو مناسب هستند. با این وجود ممکن است که بررسی مناطق خارج از داده آموزشی مدل فعلی بتواند (براساس تابع پاداش) بهینه باشد. اما در این مورد، مدل آموخته شده باید قادر باشد اطمینانش در مورد مناطق دور از داده آموزشی را صادقانه پیش‌بینی کند. مدل‌های قطعی (برای مثال LWR یا شبکه‌های عصبی) نمی‌توانند اطمینانشان در مورد مناطق دور از داده آموزشی را صادقانه نشان دهند که همین موضوع دلیل این است که غالبا با استفاده از عبارات جریمه بزرگ در تابع پاداش، از بررسی (اکتشاف)[1] جلوگیری می‌شود[107]. دو مدل که میزان خطای قابل باوری در خارج از مجموعه آموزشی دارند عبارتند از رگرسیون بیزی وزن‌دار محلی و فرآیندهای گوسی. بنابراین، در صورتی که ربات در برابر اکتشاف دلخواه آغازی مقاوم باشد این مدل‌ها حتی می‌توانند به منظور یادگیری از هیچ در زمینه رباتیک استفاده شوند، به عبارت دیگر بدون نیاز به اینکه از یک انسان خبره بخواهند تا داده مناسب برای یادگیری مدل یا سیاست آغازی فطری معقول[2] را تولید کند[3].

- **مجموعه داده‌های بزرگ**

در رباتیک، غیرمتداول نیست که مجموعه داده‌های بزرگ با میلیون‌ها داده[3] در دسترس باشد. برای نمونه، ضبط ۱۰۰ ثانیه داده با فرکانس یک کیلوهرتز منجر به مجموعه داده‌ای با یک میلیون داده می‌شود. برای مدل‌های سراسری[4] از قبیل GP استاندارد، این اندازه‌های مجموعه داده‌ها منجر به زمان محاسبه غیرعملی می‌شود. برای نمونه، GP نیاز خواهد داشت تا در طول آموزش به صورت مکرر ماتریس کرنل $10^6 \times 10^6$ را ذخیره و معکوس کند. یک روش رایج برای کاهش اندازه مجموعه داده زیرنمونه‌گیری (نمونه‌گیری از نمونه)[5] است، برای مثال

[1] Exploration
[2] Reasonably innate starting policy
[3] Data point
[4] Gloabal
[5] Subsampling

برداشتن فقط هر دهمین یا صدمین داده. این امر غالبا شدنی است چون دینامیک به اندازه کافی هموار است و حالت ربات در 1/1000 ثانیه خیلی تغییر نمی‌کند. علاوه بر این، تقریب‌های پراکنده به GP سراسری وجود دارند[116] که به طور مطلوب‌تر مقیاس می‌شوند. اما، حتی برای این روش‌ها میلیون‌ها داده غیرعملی هستند. بنابراین، در صورتی‌که مجموعه‌های داده بزرگ باشند مدل‌های محلی از قبیل LWR یا مدل‌های GP محلی [117] باید به کار گرفته شوند. ایده مدل‌های محلی، آموزش تعداد زیادی مدل با اطلاعات محلی و ترکیب پیش‌بینی‌های این مدل‌هاست.

۳-۲- جمع‌بندی

در این بخش ابتدا به شرح کلی روش‌های یادگیری تقلیدی و یادگیری تقویتی پرداختیم. در توضیح روش‌های یادگیری تقلیدی «بازنمایی ویژگی»، «یادگیری مستقیم» و «یادگیری غیرمستقیم» بیان شدند. در قسمت روش‌های یادگیری تقویتی نیز «روش‌های یادگیری تقویتی مبتنی بر مدل» و «روش‌های یادگیری تقویتی فارغ از مدل» توضیح داده شدند. با توجه به اینکه مدل استفاده شده در این پایان نامه در دسته‌ی «روش‌های یادگیری تقویتی جستجوی سیاست مبتنی بر مدل» قرار می‌گیرد در بخش بعدی این فصل این مدل‌ها به تفصیل شرح داده شدند.

فصل سوم

۳- روش‌های پیشنهادی

روش‌های پیشنهادی

به طور کلی، در این پژوهش هدف انتقال جسم به درون یک حفره توسط بازوی رباتیک شبیه‌سازی شده است که از روش‌های یادگیری تقلیدی و یادگیری تقویتی برای حل مسئله استفاده شده است. برای شبیه‌سازی در محیط‌های گسسته از الگوریتم Q-Learning استفاده می‌شود که در این پایان‌نامه با افزودن مسیرهای راهنما که از درون ویدئوهای ضبط شده استخراج شده‌اند عملکرد الگوریتم بهبود پیدا کرده است. در این فصل در بخش ۱-۳ به شرح الگوریتم Q-Learning به عنوان مدل پایه محیط‌های گسسته برای حل مسئله مورد نظر می‌پردازیم. در بخش ۲-۳ نیز افزودن مسیرهای راهنما به الگوریتم Q-Learning به عنوان بخشی از روش پیشنهادی ارائه شده در این پایان‌نامه شرح داده خواهد شد.

الگوریتم PILCO [3] به عنوان یک چارچوب برای حل مسئله ارائه شده است که در مواردی که طراحی مدل دینامیکی ربات با پیچیدگی بسیاری همراه است و یا مدل دینامیکی دارای عدم قطعیت می‌باشد مورد استفاده قرار می‌گیرد. این الگوریتم، پیش از این در مسائل مختلفی مورد استفاده قرار گرفته است، اما از آن در حل مسئله انتقال جسم به درون یک حفره توسط بازوی رباتیک شبیه‌سازی شده بهره برده نشده است. با توجه به وجود ذات عدم قطعیت در این مسئله، الگوریتم PILCO می‌تواند به عنوان راهکاری مناسب برای حل این مسئله در محیط‌های پیوسته مورد استفاده قرار بگیرد. در بخش ۳-۳ استفاده از الگوریتم PILCO در حل مسئله انتقال جسم به درون حفره در محیط‌های پیوسته به عنوان بخشی دیگر از روش پیشنهادی ارائه شده در این پایان‌نامه تشریح می‌گردد. همچنین، نحوه ایجاد تغییر در تابع هزینه به منظور افزودن قابلیت در نظر گرفتن محدودیت فضای حالت (مانند دیوار یا موانع) به مدل PILCO نیز بیان می‌گردد.

۱-۳- الگوریتم Q-Learning

Q-learning یک روش یادگیری ساده برای عامل‌هاست که با تجربه عواقب و نتایج اعمال خود و بدون نیاز به ساخت نقشه‌های حوزه‌ها، در حوزه‌های مارکفی کنترل‌شده به صورت بهینه عمل کنند. این روش معادل روش

افزایشی برای برنامه‌نویسی پویا است که نیازهای محاسباتی محدودی را اعمال می‌کند. این روش با بهره‌گیری از بهبود پی‌درپی ارزیابی‌هایش از کیفیت عمل‌های خاص در حالات خاص عمل می‌کند.

یادگیری الگوریتم Q-Learning مشابه روش تفاوت زمانی[1] ساتن عمل می‌کند. به این صورت که عامل یک عمل را در یک حالت خاص امتحان کرده و عواقب و نتایج آن عمل از لحاظ پاداش یا جریمه آنی که دریافت می‌کند و تخمین عامل از مقدار حالتی که به آن می‌رود را ارزیابی می‌کند. با امتحان کردن همه اعمال در همه حالات به طور مکرر، عامل یاد می‌گیرد که به طور کلی کدام موارد از لحاظ پاداش بلندمدت کاهش‌یافته بهتر هستند. Q-learning یک شکل ابتدایی یادگیری است اما به تنهایی می‌تواند به عنوان پایه و اساس شیوه‌های خیلی پیچیده‌تر عمل کند.

یک عامل محاسباتی که در یک جهان گسسته متناهی حرکت می‌کند و در هر گام زمانی یک عمل را از مجموعه متناهی از اعمال انتخاب می‌کند را در نظر بگیرید. جهان از یک فرآیند مارکف کنترل‌شده به همراه یک عامل به عنوان کنترل‌کننده تشکیل شده است. در گام n، عامل می‌تواند حالت $x_n(\epsilon X)$ را ثبت کند و مطابق با آن عمل $a_n(\epsilon A)$ را انتخاب کند. عامل پاداش احتمالاتی r_n را دریافت می‌کند که مقدار میانگین آن $R_{x_n}(a_n)$ فقط به حالت و عمل بستگی دارد و حالت جهان به‌طور احتمالاتی مطابق معادله (۳-۱) به y_n تغییر می‌کند:

$$Prob[y_n = y|x_n, a_n] = P_{x_n y}[a_n]. \qquad \textbf{معادله (۳-۱)}$$

کاری که عامل با آن مواجه است تعیین یک سیاست بهینه است که پاداش کاهش‌یافته مورد انتظار کلی را حداکثر می‌کند. منظور از پاداش کاهش‌یافته، پاداش‌هایی است که در s گام دریافت می‌شود بنابراین از پاداش‌هایی که در حال حاضر دریافت می‌شوند با ضریب γ^s $(0 < \gamma < 1)$ ارزش کم‌تری دارند. تحت سیاست π، ارزش حالت x به صورت معادله (۳-۲) است:

$$V^{\pi}(x) = R_x\big(\pi(x)\big) + \gamma \sum_y P_{xy}[\pi(x)]V^{\pi}(y) \qquad \textbf{معادله (۳-۲)}$$

با توجه به این معادله، عامل انتظار دارد برای اجرای عمل π که پیشنهاد کرده پاداش $R_x\big(\pi(x)\big)$ را فورا دریافت کند و سپس با احتمال $P_{xy}[\pi(x)]$ به یک حالت که برایش ارزش $V^{\pi}(y)$ دارد حرکت می‌کند.

[1] Temporal difference

اصول نظری برنامه‌نویسی پویا به ما اطمینان می‌دهد که حداقل یک سیاست مانا و بهینه π^* وجود دارد که به صورت معادله(۳-۳) است.

$$V^*(x) = V^{\pi^*}(x) = \max_a \{R_x(\pi(a)) + \gamma \sum_y P_{xy}[\pi(a)]V^{\pi^*}(y)\} \quad \textbf{معادله(۳-۳)}$$

اگرچه ممکن است این رابطه بازگشتی به نظر برسد، در واقع خوش تعریف است و با فرض دانستن $R_x(a)$ و $P_{xy}[a]$، برنامه‌نویسی پویا تعدادی روش برای محاسبه V^* و یک π^* در اختیار قرار می‌دهد. تکلیفی که یک Q-learner با آن مواجه است تعیین π^* بدون دانستن این مقادیر در ابتداست. روش‌های سنتی برای یادگیری $R_x(a)$ و $P_{xy}[a]$ به طور همزمان با اجرای برنامه‌نویسی پویا وجود دارند، اما هر گونه فرض هم‌ارزی قطعیت، یعنی محاسبه اعمال به گونه‌ای که گویا مدل فعلی دقیق است، در مراحل اولیه یادگیری گران تمام می‌شود. واتکینز[1]، Q-learning را به دلیل اینکه با روش گام به گام، سیاست بهینه را تعیین می‌کند به عنوان برنامه‌نویسی پویای افزایشی دسته‌بندی کرده است.

برای یک سیاست π، مقادیر Q به صورت معادله(۳-۴) تعریف می‌شود.

$$Q^\pi(x,a) = R_x(a) + \gamma \sum_y P_{xy}[\pi(x)]V^\pi(y) \quad \textbf{معادله(۳-۴)}$$

به عبارت دیگر، مقدار Q امید ریاضی پاداش کاهش‌یافته برای اجرای عمل a در حالت x و دنبال کردن سیاست π بعد از آن است. در Q-learning، هدف تخمین مقادیر Q برای یک سیاست بهینه است. برای راحتی، به ازای هر x و a، $Q^*(x,a) \equiv Q^{\pi^*}(x,a)$ تعریف می‌شود. به راحتی می‌توان نشان داد که $V^*(x) = \max_a Q^*(x,a)$ و همچنین اگر a^* عملی باشد که در آن به بیشینه دست یافته می‌شود، یک سیاست بهینه می‌تواند به صورت $\pi^*(x) \equiv a^*$ شکل بگیرد. سودمندی مقادیر Q در اینجاست-اگر یک عامل بتواند آنها را یاد بگیرد، به راحتی می‌تواند تصمیم بگیرد که انجام چه کاری بهینه است. اگرچه ممکن است بیش‌تر از یک سیاست بهینه یا a^* وجود داشته باشد مقادیر Q^* منحصر به فرد هستند.

در Q-learning، تجربه عامل شامل دنباله‌ای از مراحل یا قسمت‌های مجزاست. در nامین قسمت برای عامل فرآیند زیر اتفاق می‌افتد:

[1] Watkins

- حالت فعلی خودش x_n را مشاهده می‌کند،
- عمل a_n را انتخاب کرده و اجرا می‌کند،
- حالت بعدی y_n را مشاهده می‌کند،
- بازده آنی r_n را دریافت می‌کند،
- مقادیر Q_{n-1} را با استفاده از ضریب یادگیری α_n براساس معادله(۳-۵) تنظیم می‌کند:

معادله(۳-۵)

$$Q_n(x,a) = \begin{cases} (1-\alpha_n)Q_{n-1}(x,a) + \alpha_n[r_n + \gamma V_{n-1}(y_n)] & if\ x = x_n\ and\ a = a_n, \\ Q_{n-1}(x,a) & otherwise, \end{cases}$$

بهترین کاری که عامل فکر می‌کند می‌تواند از حالت y انجام دهد به صورت معادله(۳-۶) قابل بیان است.

معادله(۳-۶)

$$V_{n-1}(y_n) \equiv \max_b\{Q_{n-1}(y,b)\}$$

البته، در مراحل اولیه یادگیری، مقادیر Q ممکن است سیاستی را که به طور ضمنی تعریف می‌کنند به طور دقیق منعکس نکنند. فرض شده است که مقادیر اولیه Q یا $Q_0(x,a)$، برای همه حالات و اعمال داده شده‌اند.

۳-۲- روش پیشنهادی: استفاده از مسیرهای راهنما در Q-Learning

یکی از اصلی‌ترین اجزای الگوریتم Q-Learning ماتریس Q است. این ماتریس، میزان ارزش هر عمل را در هر یک از حالت‌های ممکن مشخص می‌کند. در حالت معمول الگوریتم Q-Learning، ماتریس Q با صفر مقداردهی اولیه می‌شود. این در حالی است که تعیین مقدار اولیه مناسب برای ماتریس Q می‌تواند تاثیر بسیار قابل توجهی در عملکرد ربات داشته باشد. برای مثال، در حالتی که ماتریس Q با صفر مقداردهی شده باشد، در فضای حالت با ابعاد بزرگ‌تر از 20×20، ربات در زمان معقول نمی‌تواند جسم را با شروع از موقعیت اولیه‌ای که بیرون از حفره قرار دارد به موقعیت هدف درون حفره برساند. روش پیشنهادی ارائه شده در این پایان‌نامه، تعیین چارچوبی است که با استفاده از آن بتوان با مقداردهی اولیه مناسب ماتریس Q، الگوریتم Q-Learning را در جهت بهبود عملکرد و کاهش زمان اجرا هدایت نمود. پایه چارچوب پیشنهادی ارائه شده برای حل این مسئله، استفاده از مسیرهای راهنما می‌باشد. این مسیرهای راهنما در قالب مقدار اولیه ماتریس

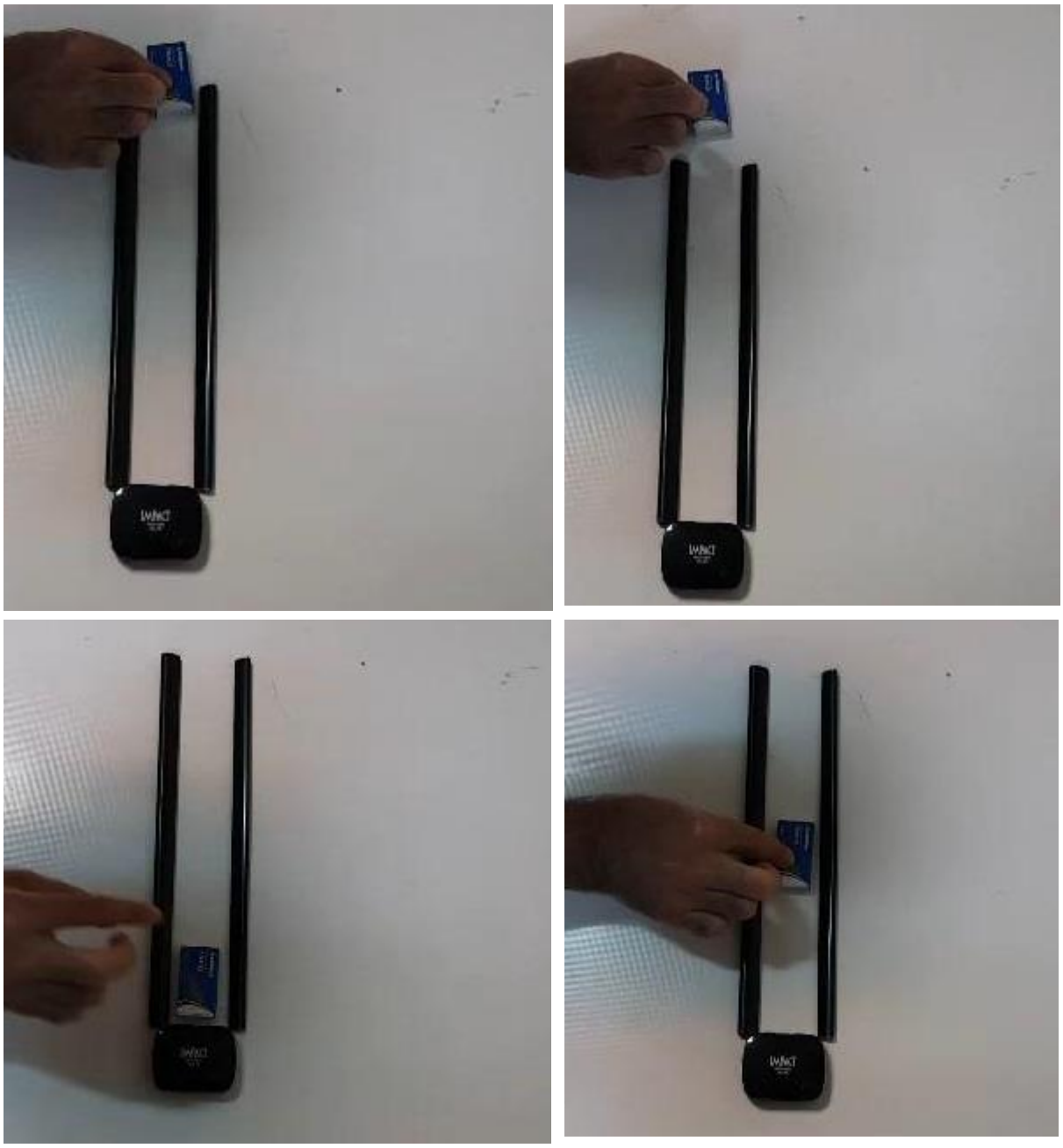
Q در اختیار الگوریتم Q-Learning قرار می‌گیرند. مسیرهای راهنما می‌توانند از دو طریق در اختیار الگوریتم قرار بگیرند. در حالت اول، مسیرهای رساندن جسم به هدف به صورت دستی توسط عامل انسانی طراحی شده و در اختیار الگوریتم قرار گرفته‌اند. استفاده از این روش، در فضاهای حالت با ابعاد بزرگ دارای پیچیدگی زیادی است. در حالت دوم، این مسیرهای راهنما از درون ویدئوهای ضبط شده و با استفاده از الگوریتم‌های ردیابی شی استخراج می‌شوند. در ادامه این فصل، در ابتدا نحوه استفاده از ویدئوهای ضبط شده برای ساخت مسیر راهنما در بخش ۱-۲-۳ بیان خواهد شد و سپس در بخش ۲-۲-۳ نحوه نفوذ دادن مسیرهای راهنما به عنوان مقدار اولیه ماتریس Q الگوریتم Q-Learning به تفصیل توضیح داده خواهد شد.

۱-۲-۳- ایجاد مسیرهای راهنما از روی ویدئوهای ضبط شده

مسیرهای راهنما می‌توانند از دو طریق مختلف ایجاد شوند. در روش اول، مسیر انتقال جسم از نقطه اولیه به هدف توسط عامل انسانی مشخص می‌شود. این روش در محیط‌های با ابعاد بزرگ چندان کارایی ندارد و پیچیدگی‌های زیادی را برای عامل انسانی به همراه می‌آورد. با توجه به این مسئله روش دوم ارائه می‌گردد که در آن مسیر انتقال جسم از نقطه اولیه به هدف به صورت خودکار و توسط ویدئوهای ضبط شده (دمو) تعیین می‌گردد. در این بخش، هدف تشریح نحوه استفاده از ویدئوهای ضبط شده برای تعیین مسیر راهنما است.

برای اینکه بتوان ویدئوهایی به منظور استخراج مسیر راهنما ایجاد کرد، لازم است که ویدئوها در فضایی مشابه فضای حالت ربات ضبط شوند. با توجه به این موضوع، لازم است محیطی طراحی کنیم که شامل جسم، حفره و فضای حرکت جسم باشد. جسم باید در یک مسیر معقول به درون حفره منتقل شود و ویدئویی از این فرآیند ضبط گردد. در شکل ۳-۱ نمونه‌ای از چهار نمای ویدئوی ضبط شده مشاهده می‌شود که جسم و حفره در آن تعبیه شده‌اند.

پس از آماده‌سازی ویدئوها لازم است تا مسیر حرکت جسم از موقعیت اولیه به هدف به طور خودکار از ویدئوها استخراج شود. برای این منظور، از الگوریتم‌های ردیابی شی در محیط دو بعدی استفاده شده است. در این گونه از الگوریتم‌ها محل جسم درون محیط در هر فریم تشخیص داده شده و به صورت مختصات در فضای دو بعدی در اختیار قرار می‌گیرد.



شکل ۳-۱ چهار نمای مختلف از یکی از ویدئوهای ضبط شده مسیر راهنما. شکل سمت راست بالا زمان شروع انتقال جسم، شکل بالا سمت چپ هنگام عبور دادن جسم از دریچه حفره، شکل سمت راست پایین انتقال جسم به سمت هدف در میانه حفره، شکل سمت چپ پایین قرار گرفتن جسم در نقطه هدف را نشان می‌دهند.

۳-۲-۲- مقداردهی اولیه ماتریس Q با استفاده از مسیرهای راهنما

پس از استخراج مختصات جسم از فریم‌های مختلف ویدئو لازم است که ماتریس Q با توجه به مختصات جسم در ویدئوهای ضبط شده مقداردهی اولیه شود. برای این منظور، در ابتدا مقادیر تمام درایه‌های ماتریس Q صفر در نظر گرفته می‌شود و سپس درایه‌هایی از ماتریس Q که جسم در مختصات مربوط به آنها در فریم‌های ویدئو مشاهده شده‌اند ارزش بالاتری متناسب با محل خود دریافت می‌کنند.

در گام بعدی، با در نظر گرفتن یک میدان پتانسیل پیرامون هر یک از نقاط دارای ارزش (نقاطی که جسم در مختصات مربوط به آنها در ویدئوها مشاهده شده است)، ارزش نقاط دیگر با توجه به میزان قرارگیری آنها در میدان پتانسیل تعیین می‌گردد. به این صورت که نقاط مجاور مراکز پتانسیل ارزش بیش‌تری دریافت می‌کنند و سایر نقاط متناسب با فاصله خود با مراکز پتانسیل ارزش کم‌تری را به خود اختصاص می‌دهند.

۳-۳- استفاده از مدل PILCO در حل مسئله انتقال جسم به درون حفره در محیط‌های پیوسته

در این بخش، توضیح می‌دهیم که یک بازوی رباتیک شبیه‌سازی شده چگونه می‌تواند یاد بگیرد که یک بلوک را به طور کاملا خودمختار داخل یک حفره قرار دهد. از فرضیاتی که در ادامه گفته خواهد شد استفاده می‌کنیم: ابتدا، از آنجایی که در دست گرفتن شی، تمرکز این کار نیست، فرض می‌کنیم که بلوک در گیره ربات قرار داده می‌شود. ثانیا، زوایا و سرعت‌های مفاصل بازو اندازه‌گیری نمی‌شوند و فقط موقعیت و سرعت مرکز بلوک در گیره ربات اندازه‌گیری می‌شود. ثالثا، هیچ مسیر یا مسیر مطلوبی از پیش معلوم نیست. رابعا، فرض می‌کنیم که موقعیت اولیه و موقعیت هدف بلوک در گیره ربات ثابت است.

به طور ساده دنبال کردن یک مسیر مستقیم بین موقعیت اولیه و هدف ممکن است به دلیل موانع موفقیت‌آمیز نباشد. برای برنامه‌ریزی بلندمدت و یادگیری کنترل‌کننده، این عدم قطعیت باید در نظر گرفته شود.

در هر گام زمانی، ربات از مرکز بلوک در گیره‌اش برای محاسبه سیگنال کنترلی با مقدار گسسته $u \in \mathbb{R}^2$ استفاده می‌کند. چرخش مچ و باز/بسته شدن گیره یاد گرفته نمی‌شوند. الگوریتم یادگیری تقویتی، از مرکز ۲ بعدی شی و سرعتش به عنوان حالت $x \in \mathbb{R}^4$ استفاده می‌کند.

۱-۳-۳- یادگیری سیاست با محدودیت‌های فضای حالت

در ادامه، چارچوب کاری PILCO [3] را برای یادگیری یک سیاست حلقه-بسته مناسب (کنترل‌کننده بازخورد حالت) $\pi : \mathbb{R}^4 \to \mathbb{R}^2, x \mapsto u$ به طور مختصر شرح می‌دهیم. در اینجا، x حالت نامیده می‌شود که به صورت مختصات و سرعت مرکز بلوک (x_c, y_c, x'_c, y'_c) در گیره تعریف شده است. ما قصد داریم که این سیاست را از هیچ یا به عبارت دیگر فقط با دانش قبلی خیلی کلی در مورد کار و خود راه‌حل یاد بگیریم. علاوه بر این، می‌خواهیم π را در تعداد کمی آزمایش پیدا کنیم یا به عبارت دیگر، به یک روش یادگیری داده-کارا نیاز داریم.

به عنوان معیاری برای قضاوت در مورد عملکرد کنترل‌کننده π، می‌توانیم از امید ریاضی حاصل بلندمدت یک تراژکتوری $(x_0, \cdots, x_T)$ به هنگام اعمال π استفاده کنیم که به صورت معادله(۳-۷) است:

$$J^\pi = \sum_{t=0}^{T} \mathbb{E}_{x_t}[c(x_t)] \qquad \textbf{معادله(۳-۷)}$$

در معادله(۳-۷)، T افق پیش‌بینی و $c(x_t)$ هزینه لحظه‌ای بودن در حالت x در زمان t است. در صورتی‌که به گونه‌ای دیگر بیان نشود، در کل این پایان‌نامه از یک تابع هزینه اشباع شده $c = 1 - \exp(-d^2/\sigma_c^2)$ استفاده می‌کنیم که فواصل اقلیدسی d بلوک در محرک (مجری) پایانی[1] از موقعیت هدف x_{target} را جریمه می‌کند. فرض می‌کنیم که سیاست π با ψ پارامتربندی می‌شود. PILCO سیاست پارامتربندی‌شده مناسب را با استفاده از الگوریتم ۲-۷ یاد می‌گیرد.

[1] End effector

۱-۱-۳-۳- مدل دینامیکی احتمالاتی

برای پرهیز از فرضیات هم‌ارزی اطمینان (قطعیت)[1] روی مدل آموخته شده، PILCO عدم قطعیت‌های مدل را در طول برنامه‌ریزی در نظر می‌گیرد. از همین رو، یک توزیع (پسین) روی مدل‌های دینامیکی ممکن نیاز است. ما از GP ها[102] برای استنتاج این توزیع پسین از تجربه در حال حاضر در دسترس استفاده می‌کنیم.

با پیروی از [102]، به طور مختصر نمادگذاری و مدل‌های پیش‌بینی استاندارد برای GP ها را معرفی می‌کنیم که به منظور استنتاج توزیع روی یک تابع پنهان f از مشاهدات نویزی $y_i = f(\mathrm{x}_i) + \varepsilon$ استفاده می‌شوند، در این پایان‌نامه، $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ نویز i.i.d سیستم در نظر گرفته شده است. یک GP به طور کامل با یک تابع میانگین و $m(.)$ و یک تابع کوواریانس نیمه‌معین مثبت $k(. \ , \ .)$ که کرنل نامیده می‌شود مشخص می‌شود. در سرتاسر این پایان‌نامه، یک تابع میانگین مقدم $m \equiv 0$ و کرنل نمایی مجذور (SE) با تعیین ارتباط خودکار در نظر می‌گیریم که به صورت زیر تعریف شده است:

$$k(x, x') = \alpha^2 \, exp\left(-\frac{1}{2}(x - x')^T \Lambda^{-1}(x - x')\right) \qquad \textbf{معادله(۳-۸)}$$

در اینجا، α^2 را به عنوان واریانس تابع پنهان f و $\Lambda := diag([\ell_1^2, \cdots, \ell_D^2])$ را تعریف کرده‌ایم که به مقیاس‌های طول مشخصه ℓ_i بستگی دارد. با در اختیار داشتن n ورودی آموزشی $\mathrm{X} = [\mathrm{x}_1, \cdots, \mathrm{x}_n]$ و اهداف آموزشی مربوطه $\mathrm{y} = [y_1, \cdots, y_n]^{\mathrm{T}}$، فراپارامترهای GP پسین (مقیاس‌های طول ℓ_i، واریانس سیگنال α^2 و واریانس‌های نویز σ_ε^2) با بیشینه‌سازی شواهد[2] یاد گرفته می‌شوند[102].

توزیع پیش‌بینی‌کننده پسین $p(f_*|\mathrm{x}_*)$ تابع مقدار $f_* = f(\mathrm{x}_*)$ برای یک ورودی آزمایشی دلخواه اما معلوم x_*، گوسی با میانگین و واریانسی است که به ترتیب به صورت معادلات معادله(۳-۹) و معادله(۳-۱۰) هستند

$$m_f(x_*) = \mathbb{E}_f[f_*] = k_*^T(K + \sigma_\varepsilon^2)^{-1} y = k_*^T \beta \qquad \textbf{معادله(۳-۹)}$$

$$\sigma_f^2(x_*) = var_f[f_*] = k_{**} - k_*^T(K + \sigma_\varepsilon^2 I)^{-1} k_* + \sigma_\varepsilon^2 \qquad \textbf{معادله(۳-۱۰)}$$

[1] Certainty equivalence
[2] Evidence maximization

که در آن $k_* := k(\mathrm{X}, \mathrm{x}_*)$، $k_{**} := k(\mathrm{x}_*, \mathrm{x}_*)$، $\beta := (K + \sigma_\varepsilon^2)^{-1}\mathrm{y}$ و K ماتریس کرنل با عناصر $K_{ij} = k(\mathrm{x}_i, \mathrm{x}_j)$ است.

در سیستم رباتیکی ما، GP تابع $f: \mathbb{R}^6 \to \mathbb{R}^4, (\mathrm{x}_{t-1}, \mathrm{u}_{t-1}) \mapsto \Delta_t := \mathrm{x}_t - \mathrm{x}_{t-1} + \varepsilon_t$ را مدل می‌کند که $\varepsilon_t \in \mathbb{R}^4$ نویز گوسی i.i.d سیستم است. ورودی‌ها و اهداف آموزشی به مدل GP به ترتیب، چندتایی‌های $(\mathrm{x}_{t-1}, \mathrm{u}_{t-1})$ و تفاضل‌های مربوطه Δ_t هستند.

۲-۱-۳-۳- برنامه‌ریزی بلندمدت از طریق استنتاج تقریبی

کمینه کردن و ارزیابی J^π در معادله(۳-۷)، به پیش‌بینی‌های بلندمدت از تغییر و تحول حالت نیاز دارد. برای به دست آوردن توزیع‌های حالت $p(\mathrm{x}_1), \cdots, p(\mathrm{x}_T)$، پیش‌بینی‌های یک گامه را پشت سر هم می‌چینیم. انجام صحیح این کار به نگاشت ورودی‌های آزمایشی نامطمئن به واسطه مدل دینامیکیGP نیاز دارد. در ادامه، فرض می‌کنیم که این ورودی‌های آزمایشی دارای توزیع گوسی هستند و نتایج مرجع [3] را به برنامه‌ریزی بلندمدت در سیستم‌های تصادفی با ورودی‌های کنترلی تعمیم می‌دهیم.

برای پیش‌بینی x_t از $p(\mathrm{x}_{t-1})$، به توزیع توام $p(\mathrm{x}_{t-1}, \mathrm{u}_{t-1})$ نیاز داریم. برای محاسبه این توزیع، از $\mathrm{u}_{t-1} = \pi(\mathrm{x}_{t-1})$ استفاده می‌کنیم که این یعنی کنترل تابعی از حالت است: ابتدا، توزیع پیش‌بینی‌کننده کنترل $p(\mathrm{u}_{t-1})$ و سپس، کوواریانس متقابل $cov[\mathrm{x}_{t-1}, \mathrm{u}_{t-1}]$ را محاسبه می‌کنیم. سرانجام، $p(\mathrm{x}_{t-1}, \mathrm{u}_{t-1})$ را با یک توزیع گوسی با میانگین و کواریانس درست تقریب می‌زنیم. این محاسبات به پارامتربندی ψ سیاست π بستگی دارند. در این پایان‌نامه، فرض می‌کنیم $\mathrm{u}_{t-1} = \pi(\mathrm{x}_{t-1}) = \mathrm{A}\mathrm{x}_{t-1} + \mathrm{b}$ با $\psi = \{\mathrm{A}, \mathrm{b}\}$. با $p(\mathrm{x}_{t-1}) = \mathcal{N}(\mathrm{x}_{t-1}|\mu_{t-1}, \Sigma_{t-1})$، با اعمال نتایج استاندارد از مدل‌های گوسی خطی[1] دو رابطه زیر حاصل می‌شوند:

$$p(\mathrm{u}_{t-1}) = \mathcal{N}(\mathrm{u}_{t-1}|\mu_u, \Sigma_u)$$
$$\mu_u = \mathrm{A}\mu_{t-1} + \mathrm{b}\,, \quad \Sigma_u = \mathrm{A}\Sigma_{t-1}\mathrm{A}^T$$

[1] Linear-Gaussian

در این مثال، π یک تابع خطی از x_{t-1} است و بنابراین توزیع توام مطلوب $p(\mathrm{x}_{t-1}, \mathrm{u}_{t-1})$ دقیقا گوسی است و به صورت معادله(۳-۱۱) مشخص می‌شود

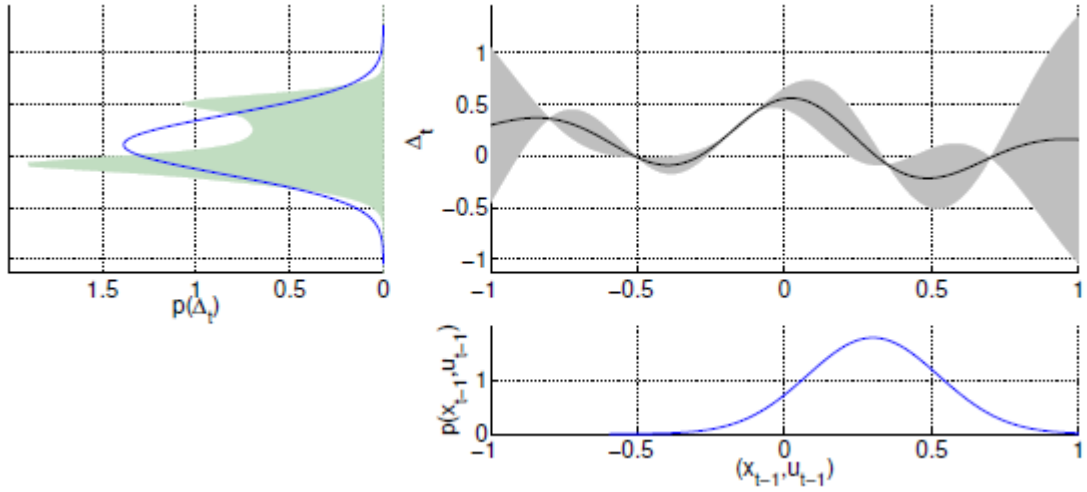
$$\mathcal{N}\left(\begin{bmatrix}\mu_{t-1}\\ A\mu_{t-1}+b\end{bmatrix}, \begin{bmatrix}\Sigma_{t-1} & \Sigma_{t-1}A^T\\ A\Sigma_{t-1} & A\Sigma_{t-1}A^T\end{bmatrix}\right) \qquad \textbf{معادله(۳-۱۱)}$$

که در این معادله، کوواریانس متقابل $cov[\mathrm{x}_{t-1}, \mathrm{u}_{t-1}] = \Sigma_{t-1}\mathrm{A}^T$ است. برای خیلی از پارامتربندی‌های کنترل‌کننده جالب دیگر، میانگین و کوواریانس می‌توانند به صورت تحلیلی انجام شوند[118]، گرچه ممکن است $p(\mathrm{x}_{t-1}, \mathrm{u}_{t-1})$ دیگر دقیقا گوسی نباشد.

از الان به بعد، یک توزیع توام گوسی $p(\tilde{\mathrm{x}}_{t-1}) = \mathcal{N}\left(\tilde{x}_{t-1}|\tilde{\mu}_{t-1}, \tilde{\Sigma}_{t-1}\right)$ در زمان $t-1$ فرض می‌کنیم که $\tilde{\mathrm{x}} := [\mathrm{x}^T\ \mathrm{u}^T]^T$ را تعریف می‌کنیم و $\tilde{\mu}$ و $\tilde{\Sigma}$ میانگین و کوواریانس این متغیر تقویت‌شده هستند. هنگامی که توزیع نشان داده شده در معادله(۳-۱۲) را پیش‌بینی می‌کنیم، روی متغیر تصادفی $\tilde{\mathrm{x}}_{t-1}$ انتگرال‌گیری می‌کنیم.

$$p(\Delta_t) = \int p(f(\tilde{x}_{t-1})|\tilde{x}_{t-1})p(\tilde{x}_{t-1})d\tilde{x}_{t-1} \qquad \textbf{معادله(۳-۱۲)}$$

احتمال انتقال $p(f(\tilde{\mathrm{x}}_{t-1})|\tilde{\mathrm{x}}_{t-1})$ از توزیع پسین GP به دست می‌آید. محاسبه توزیع پیش‌بینی‌کننده دقیق در معادله(۳-۱۲) به صورت تحلیلی لاینحل است. بنابراین، $p(\Delta_t)$ را با یک گوسی با میانگین و واریانس دقیق (تطبیق ممان) تقریب می‌زنیم. شکل ۳-۲ این سناریو را نشان می‌دهد. توجه داشته باشید که برای محاسبه میانگین μ_Δ و واریانس σ_Δ^2 توزیع پیش‌بینی‌کننده، توزیع پیش‌بینی‌کننده GP استاندارد (به معادلات معادله(**۳-۹**) و معادله(**۳-۱۰**) مراجعه شود) کفایت نمی‌کند چون $\tilde{\mathrm{x}}_{t-1}$ به طور قطعی داده نشده است.



شکل ۳-۲ پیش‌بینی GP در یک ورودی نامطمئن. توزیع ورودی $p(x_{t-1}, u_{t-1})$ گوسی فرض شده است (پنل سمت راست پایین). انتشار آن از طریق مدل GP (پنل سمت راست بالا)، توزیع سایه خورده $p(\Delta_t)$ در پنل سمت چپ بالا را نتیجه می‌دهد. $p(\Delta_t)$ با یک گوسی با میانگین و واریانس دقیق تقریب زده می‌شود (پنل سمت چپ پایین).[108]

فرض می‌کنیم که میانگین μ_Δ و کوواریانس Σ_Δ توزیع پیش‌بینی‌کننده $p(\Delta_t)$ معلوم هستند. آنگاه، یک تقریب گوسی به توزیع حالت مورد نظر $p(\mathrm{x}_t)$، به‌ترتیب میانگین و کوواریانس به صورت معادلات معادله(۳-۱۳) ومعادله(۳-۱۴) دارد.

$$\mu_t = \mu_{t-1} + \mu_\Delta \qquad \textbf{معادله(۳-۱۳)}$$

$$\Sigma_t = \Sigma_{t-1} + \Sigma_\Delta + cov[x_{t-1}, \Delta_t] + cov[\Delta_t, x_{t-1}] \qquad \textbf{معادله(۳-۱۴)}$$

$$cov[x_{t-1}, \Delta_t] = cov[x_{t-1}, u_{t-1}]\Sigma_u^{-1} cov[u_{t-1}, \Delta_t] \qquad \textbf{معادله(۳-۱۵)}$$

محاسبه کوواریانس‌های متقابل مورد نیاز در معادله(۳-۱۵) به پارامتربندی سیاست بستگی دارد، اما غالبا می‌تواند به صورت تحلیلی محاسبه شود.

در ادامه، میانگین μ_Δ و واریانس σ_Δ^2 توزیع پیش‌بینی‌کننده $p(\Delta_t)$ را محاسبه می‌کنیم (به معادله(۳-۱۲) رجوع شود).

۱) میانگین: با پیروی از قانون امید ریاضی‌های مکرر، به دست می‌آوریم:

$$\mu_\Delta = \mathbb{E}_{\tilde{\mathrm{x}}_{t-1}}\left[\mathbb{E}_f[f(\tilde{\mathrm{x}}_{t-1})|\tilde{\mathrm{x}}_{t-1}]\right] = \mathbb{E}_{\mathrm{x}_*}\left[m_f(\tilde{\mathrm{x}}_{t-1})\right]$$
$$= \int m_f(\tilde{x}_{t-1})\mathcal{N}\left(\tilde{x}_{t-1}\middle|\tilde{\mu}_{t-1}, \tilde{\Sigma}_{t-1}\right)d\tilde{x}_{t-1} = \beta^T q \qquad \textbf{معادله(۳-۱۶)}$$

که $\beta = (K + \sigma_\varepsilon^2 \mathrm{I})^{-1}y$ و $\mathrm{q} = [q_1, \cdots, q_n]^{\mathrm{T}}$. عناصر $\mathrm{q} \in \mathbb{R}^n$ به صورت زیر هستند:

$$q_i = \int k(\mathrm{x}_i, \mathrm{x}_*)\mathcal{N}\left(\tilde{x}_{t-1}\middle|\tilde{\mu}_{t-1}, \tilde{\Sigma}_{t-1}\right)d\tilde{\mathrm{x}}_{t-1}$$
$$= \frac{\alpha_a^2}{\sqrt{\left|\tilde{\Sigma}_{t-1}\Lambda^{-1} + \mathrm{I}\right|}} \exp\left(-\frac{1}{2}(\mathrm{x}_i - \tilde{\mu}_{t-1})^T\left(\tilde{\Sigma}_{t-1} + \Lambda\right)^{-1}(\mathrm{x}_i - \tilde{\mu}_{t-1})\right) .$$

۲) واریانس: با استفاده از قانون واریانس کل، به دست می‌آوریم:

$$\sigma_\Delta^2 = \mathbb{E}_{\tilde{\mathrm{x}}_{t-1}}\left[m_f(\tilde{\mathrm{x}}_{t-1})^2\right] - \mathbb{E}_{\tilde{\mathrm{x}}_{t-1}}\left[m_f(\tilde{\mathrm{x}}_{t-1})\right]^2 + \mathbb{E}_{\tilde{\mathrm{x}}_{t-1}}\left[\sigma_f^2(\tilde{\mathrm{x}}_{t-1})\right]$$
$$= \beta^T Q\beta + \alpha^2 - tr((K + \sigma_\varepsilon^2 I)^{-1}Q) - \mu_\Delta^2 + \sigma_\varepsilon^2 \qquad \textbf{معادله(۳-۱۷)}$$

که منظور از $tr(.)$، مجموع عناصر روی قطر اصلی ماتریس[1] است. عناصر $Q \in \mathbb{R}^{n\times n}$ به صورت زیر هستند:

$$Q_{ij} = k(\mathrm{x}_i, \tilde{\mu}_{t-1})k\left(\mathrm{x}_j, \tilde{\mu}_{t-1}\right)\left|2\tilde{\Sigma}_{t-1}\Lambda^{-1} + \mathrm{I}\right|^{-\frac{1}{2}} \times \exp\left(\frac{1}{2}z_{ij}^T\left(2\tilde{\Sigma}_{t-1}\Lambda^{-1} + \mathrm{I}\right)\tilde{\Sigma}_{t-1}z_{ij}\right)$$

که $\varsigma_i := (\mathrm{x}_i - \tilde{\mu}_{t-1})$ و $z_{ij} := \Lambda^{-1}(\varsigma_i + \varsigma_j)$ است.

توجه داشته باشید که هر دو μ_Δ و σ_Δ^2 از لحاظ کاربردی به‌ترتیب به واسطه $\tilde{\mu}_{t-1}$ و $\tilde{\Sigma}_{t-1}$ به میانگین μ_u و کوواریانس Σ_u سیگنال کنترلی وابسته هستند (به معادلات معادله(***۳-۱۶***) و معادله(***۳-۱۷***) رجوع شود). می‌توانیم از معادلات معادله(***۳-۱۶***) و معادله(***۳-۱۷***) بفهمیم که عدم قطعیت در مورد تابع پنهان f انتگرال‌گیری می‌شود، چیزی که صریحا عدم قطعیت مدل را در نظر می‌گیرد.

[1] Trace

۳-۱-۳-۳- یادگیری کنترل‌کننده از طریق جستجوی غیرمستقیم سیاست

از بخش ۲-۱-۳-۳ می‌دانیم که چگونه پیش‌بینی‌های یک گامه را پشت سر هم انجام دهیم تا تقریبهای گوسی از توزیع‌های پیش‌بینی‌کننده $p(\mathrm{x}_1), \cdots, p(\mathrm{x}_T)$ را به دست آوریم. برای ارزیابی امیدریاضی حاصل J^{π} در معادله(۳-۷)، محاسبه امید ریاضی مقادیر هزینه c نسبت به توزیع‌های پیش‌بینی‌کننده حالت باقی مانده است.

$$\mathbb{E}[c(x_t)] = \int c(x_t)\mathcal{N}(x_t|\mu_t, \Sigma_t)dx_t \quad , t = 0, \cdots, T \qquad \textbf{معادله(۳-۱۸)}$$

فرض می‌کنیم که هزینه c طوری انتخاب شده است (برای مثال چندجمله‌ای) که معادله(۳-۱۸) می‌تواند به صورت تحلیلی حل شود.

برای اعمال جستجوی مبتنی بر گرادیان سیاست به منظور پیدا کردن پارامترهای ψ که J^{π} را کمینه می‌کنند (به معادله(۳-۷) رجوع شود)، ابتدا ترتیب مشتق‌گیری و جمع کردن در معادله(۳-۷) را عوض می‌کنیم. با در نظر گرفتن $\mathcal{E}_t := \mathbb{E}_{\mathrm{x}_t}[c(\mathrm{x}_t)]$ به دست می‌آوریم:

$$\frac{d\mathcal{E}_t}{d\theta} = \frac{d\mathcal{E}_t}{d\mu_t}\frac{d\mu_t}{d\psi} + \frac{d\mathcal{E}_t}{d\Sigma_t}\frac{d\Sigma_t}{d\psi} \qquad \textbf{معادله(۳-۱۹)}$$

مشتقات کلی میانگین μ_t و کوواریانس Σ_t مربوط به $p(\mathrm{x}_t)$ نسبت به پارامترهای سیاست ψ می‌توانند به صورت تحلیلی و با اعمال مکرر قاعده زنجیره‌ای به معادلات معادله(۳-۱۳)، معادله(۳-۱۴)، معادله(۳-۱۵)، معادله(۳-۱۶) و معادله(۳-۱۷) محاسبه شوند. این کار، محاسبه مشتقات جزئی $d\mu_t/d\psi$ و $d\Sigma_t/d\psi$ را نیز دربرمی‌گیرد. در اینجا جزئیات بیش‌تر را حذف کرده‌ایم، اما این مشتقات در مقاله [118] به صورت تحلیلی محاسبه شده‌اند. این کار امکان استفاده از روش‌های بهینه‌سازی غیرمحدب مبتنی بر گرادیان استاندارد از قبیل CG یا L-BFGS را فراهم می‌کند که بردار پارامتر بهینه‌شده ψ^* را برمی‌گردانند.

۴-۱-۳-۳- برنامه‌ریزی با محدودیت‌های فضای حالت

در ساز و کار یادگیری تقویتی کلاسیک، فرض بر این است که یادگیرنده از هیچ‌کدام از محدودیت‌های در فضای حالت باخبر نیست، بلکه باید دیوارها و غیره را با برخورد به آنها و متحمل جریمه سنگین شدن کشف کند. در ساز و کار رباتیک، این فرض کلی اما نه لازم، خیلی مطلوب نیست چون ربات می‌تواند صدمه ببیند.

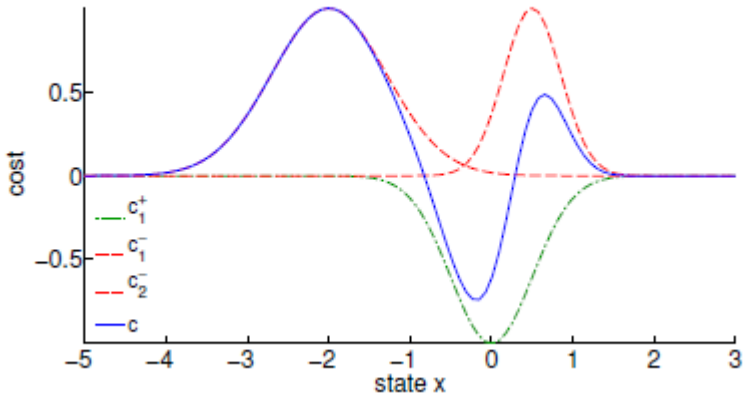
اگر محدودیت‌ها (برای مثال موانع) در فضای حالت از قبل معلوم باشند، ترجیح بر این است که این دانش قبلی را مستقیما در برنامه‌ریزی و یادگیری سیاست بگنجانیم. پیشنهاد می‌کنیم که موانع را به صورت مناطق "نامطلوب" تعریف کنیم، به عبارت دیگر مناطقی که ربات قرار است از آنها پرهیز کند. "نامطلوب بودن" را به صورت جریمه در تابع هزینه لحظه‌ای c تعریف می‌کنیم. از همین رو، تابع هزینه c را به صورت زیر تعریف می‌کنیم

$$c(x) = \sum_{k=1}^{K} c_k^+(x) - \sum_{j=1}^{J} \iota_j c_j^-(x) \qquad \textbf{معادله(۳-۲۰)}$$

که c_k^+ حالت‌های مطلوب (برای مثال حالت هدف) هستند و c_j^- حالت‌های نامطلوب (برای مثال موانع) هستند که با $\iota_j \geq 0$ وزن‌دهی شده‌اند. مقادیر بزرگ‌تر برای ι_j باعث می‌شوند سیاست بیش‌تر از یک حالت نامطلوب خاص پرهیز کند. در این پایان‌نامه، همیشه $\iota_j = 1$ را قرار می‌دهیم. برای c_k^+ و c_j^-، نمایی‌های مجذور منفی‌شده را انتخاب می‌کنیم که در هنگام میانگین‌گیری مطابق با توزیع حالت، بین اکتشاف و بهره‌برداری توازن ایجاد می‌کند[118]. نمایی‌های مجذور با عرض‌های بالقوه مختلف Σ_k^+ نرمال‌نشده هستند. عرض‌های تک‌تک محدودیت‌ها c_j^- معین می‌کنند که محدودیت‌ها چقدر "نرم" هستند. محدودیت‌های سفت و سخت با نمایی‌های مجذور c_j^- خیلی نوک‌تیز[1] و $\iota_j \to \infty$ توصیف می‌شوند. این ایده مربوط به مرجع [119] است که در آن برنامه‌ریزی با دینامیک کاملا معلوم و کنترل‌کننده تکه‌ای خطی انجام شده است.

شکل ۳-۳، معادله(۳-۲۰) را با دو جریمه c_j^- و یک پاداش c_k^+ نشان می‌دهد. این شکل نشان می‌دهد که اگر یک حالت نامطلوب و یک حالت مطلوب به هم نزدیک باشند، هزینه کلی c تا حدی بین دو هدف توازن ایجاد می‌کند. علاوه بر این، دیگر مثل گذشته حالت بهینه $x_* \in \operatorname{argmin}_x c(x)$ مربوط به $x_*^+ \in \operatorname{argmin}_x c^+(x)$ نیست: کمی دور شدن از حالت هدف (دور از حالت نامطلوب) بهینه است.

[1] Peaked



شکل ۳-۳ تابع هزینه‌ای که محدودیت‌ها (برای مثال موانع) را با "نامطلوب" کردن آنها در نظر می‌گیرد. منحنی‌های با خط‌چین نشان داده شده اجزای منفرد c_k^+ و c_j^- هستند، به معادله(۳-۲۰) رجوع شود، منحنی با خط یکنواخت نشان داده شده جمع آنها c است.[108]

امید ریاضی‌های هزینه در معادله(۳-۲۰) و مشتقات نسبت به میانگین μ_t و کوواریانس Σ_t توزیع حالت $p(\mathrm{x}_t)$ می‌توانند برای هر c_k^+ و c_j^- محاسبه شده و در آخر با هم جمع شوند. سپس، براساس معادله(۳-۱۹) قاعده زنجیره‌ای را برای جستجوی سیاست مبتنی بر گرادیان اعمال می‌کنیم.

بیان محدودیت‌ها در قالب نامطلوب بودن در تابع هزینه در معادله(۳-۲۰)، هنوز اجازه برنامه‌ریزی بلندمدت کاملا احتمالاتی را می‌دهد و امکان هدایت ربات در فضای حالت را فراهم می‌کند بدون اینکه با برخورد به موانع، آنها را "تجربه" کند.

برخوردهای درون چارچوب کاری استنتاج بیزی می‌توانند منع شوند، اما اکیدا از امید ریاضی حذف نمی‌شوند. این موضوع به این معنا نیست که میانگین‌گیری عدم قطعیت‌ها اشتباه است – در عوض به ما می‌گوید که انتظار نمی‌رود که با یک اطمینان مشخص محدودیت‌ها نقض شوند. توصیف صادقانه عدم قطعیت قابل پیش‌بینی غالبا باارزش‌تر از ادعا کردن با اطمینان کامل و گاه و بیگاه نقض غیرمنتظره محدودیت‌هاست.

۳-۴- جمع‌بندی

در این فصل، روش‌های پایه و پیشنهادی برای حل مسئله انتقال جسم به درون یک حفره توسط بازوی رباتیک شبیه‌سازی شده بیان شدند. برای شبیه‌سازی در محیط‌های گسسته از الگوریتم Q-Learning استفاده شد و به عنوان روش پیشنهادی این پایان‌نامه، افزودن مسیرهای راهنما که از درون ویدئوهای ضبط شده استخراج شده‌اند شرح داده شد. برای شبیه‌سازی در محیط‌های پیوسته از الگوریتم PILCO استفاده شد و قابلیت در نظر گرفتن محدودیت‌های فضای حالت (مانند دیوار و مانع) به عنوان روش پیشنهادی به مدل PILCO افزوده شد.

فصل چهارم

۴- تحلیل و ارزیابی نتایج

تحلیل و ارزیابی نتایج

هدف از این فصل بررسی آزمایش‌های صورت گرفته در زمینه‌ی این پایان‌نامه و ارزیابی عملکرد مدل پیشنهادی است. ابتدا در بخش ۴-۱ به معرفی مجموعه داده‌های مورد استفاده در آزمایش‌ها خواهیم پرداخت و ویژگی‌های مجموعه داده‌های مورد استفاده را بیان می‌کنیم. سپس در بخش ۴-۲ پیش‌پردازش‌های صورت گرفته بر روی مجموعه داده شرح داده می‌شود. در بخش ۴-۳ معیارهای ارزیابی مورد استفاده توضیح داده می‌شوند و همچنین پارامترهای مختلف مدل به طور خلاصه بیان می‌شوند و در مورد نحوه‌ی تنظیم آن‌ها بحث خواهد شد. در بخش ۴-۴ نیز آزمایش‌های مختلف صورت گرفته برای بررسی پارامترهای مدل و همچنین مقایسه‌ی روش پیشنهادی با روش‌های پیشین ارائه خواهند شد.

۴-۱- معرفی محیط‌های شبیه‌سازی

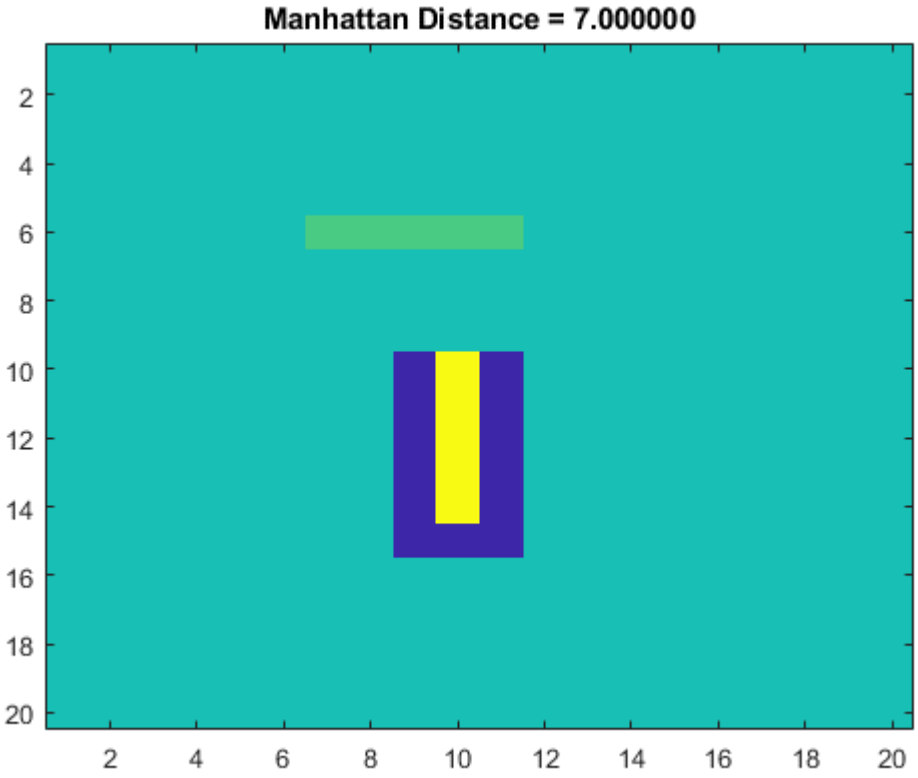
در این پایان‌نامه، محیط شبیه‌سازی به دو صورت گسسته و پیوسته در نظر گرفته شده است. در محیط گسسته از الگوریتم Q-Learning و Q-Learning با راهنما برای یادگیری مسیر از موقعیت اولیه به موقعیت هدف استفاده شده است. همچنین، در محیط پیوسته از الگوریتم PILCO برای یادگیری مسیر از موقعیت اولیه به موقعیت هدف استفاده شده است. ویژگی‌های محیط گسسته و محیط پیوسته به ترتیب در بخش‌های ۴-۱-۱ و ۴-۱-۲ شرح داده شده‌اند.

۴-۱-۱- ویژگی‌های محیط گسسته

محیط گسسته در نظر گرفته شده در این پایان‌نامه، یک محیط دوبعدی شبکه‌بندی شده است. این محیط دارای ابعاد $m \times n$ می‌باشد که m و n به ترتیب طول (جهت x) و عرض (جهت y) را مشخص می‌کنند. هر نقطه i در محیط شبیه‌سازی با زوج مرتب (x_i, y_i) مشخص می‌شود که در آن x_i یک عدد طبیعی کوچک‌تر یا مساوی m و y_i یک عدد طبیعی کوچک‌تر یا مساوی n است.

در این محیط شبیه‌سازی یک جسم وجود دارد که می‌تواند ابعاد مختلفی داشته باشد و با توجه به ابعاد آن و موقعیتش در محیط، سلول‌هایی از محیط را اشغال کند. ربات، جسم را از یک نقطه اولیه مشخص به یک نقطه هدف مشخص می‌رساند.

در این محیط شبیه‌سازی بعضی از نقاط به عنوان دیوار یا مانع در نظر گرفته شده‌اند که ربات امکان انتقال جسم به این خانه‌ها را ندارد. سه دیوار در محیط در نظر گرفته شده است که این سه دیوار در کنار یکدیگر تشکیل یک حفره را می‌دهند که موقعیت هدف می‌تواند در این حفره قرار گیرد. شکل ۴-۱ یک نمونه از محیط شبیه‌سازی گسسته با ابعاد ۲۰ در ۲۰ و جسم با ابعاد ۱ در ۵ را نشان می‌دهد.



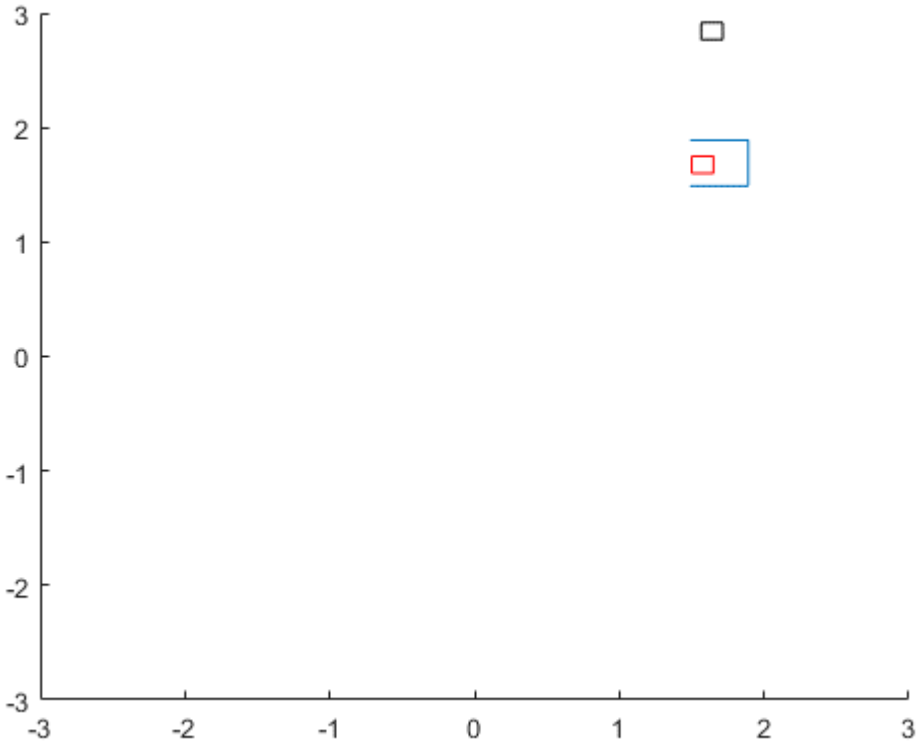
شکل ۴-۱ نمایی از یک محیط شبیه‌سازی گسسته. مستطیل سبز رنگ جسم گرفته شده توسط ربات را نشان می‌دهد که باید به موقعیت هدف که با مستطیل زرد رنگ مشخص شده است منتقل شود. موقعیت هدف درون حفره‌ای که توسط سه دیوار سرمه‌ای رنگ مشخص شده قرار گرفته است.

۴-۱-۲- ویژگی‌های محیط پیوسته

محیط پیوسته در نظر گرفته شده در این پایان‌نامه، یک محیط دوبعدی شده است. این محیط دارای ابعاد $m \times n$ می‌باشد که m و n به ترتیب طول (جهت x) و عرض (جهت y) را مشخص می‌کنند. هر نقطه i در محیط شبیه‌سازی با زوج مرتب (x_i, y_i) مشخص می‌شود که در آن x_i یک عدد حقیقی بین $-\frac{m}{2}$ و $\frac{m}{2}$ و y_i یک عدد حقیقی بین $-\frac{n}{2}$ و $\frac{n}{2}$ است.

در این محیط شبیه‌سازی یک جسم وجود دارد که می‌تواند ابعاد مختلفی داشته باشد و با توجه به ابعاد آن و موقعیتش در محیط، بخشی از محیط را اشغال می‌کند. ربات، جسم را از یک نقطه اولیه مشخص به یک نقطه هدف مشخص می‌رساند.

در این محیط شبیه‌سازی بعضی از نقاط به عنوان دیوار یا مانع در نظر گرفته شده‌اند که امکان عبور ربات از این خانه‌ها وجود ندارد. سه دیوار در محیط در نظر گرفته شده است که این سه دیوار در کنار یکدیگر تشکیل یک حفره را می‌دهند که موقعیت هدف می‌تواند در این حفره قرار گیرد. این سه دیوار با سه پاره‌خط که دارای طول و معادله خط مشخص هستند معین می‌شود. معادله این خطوط به صورت $x = a$ یا $y = b$ است که a و b دو عدد ثابت را مشخص می‌کنند. شکل ۴-۲ یک نمونه از محیط شبیه‌سازی با ابعاد ۶ در ۶ و جسم با ابعاد ۰/۱۵ در ۰/۱۵ را نشان می‌دهد.



شکل ۴-۲ نمایی از یک محیط شبیه‌سازی پیوسته. مربع به رنگ مشکی، جسم گرفته شده توسط ربات را نشان می‌دهد که باید به موقعیت هدف که با مربع قرمز رنگ نشان داده شده است منتقل شود. موقعیت هدف درون حفره‌ای که توسط سه دیوار آبی رنگ مشخص شده قرار گرفته است.

۴-۲- نحوه آماده‌سازی ویدئوهای مسیر راهنما

یکی از الگوریتم‌های استفاده شده در این پایان‌نامه، Q-Learning با مسیر راهنما است. در این الگوریتم لازم است که مسیر[1]های راهنما به عنوان ورودی به الگوریتم داده شوند. ما در این پایان‌نامه، مسیرهای راهنما را از ویدئوهایی که توسط خودمان ضبط شده است استخراج کرده‌ایم. در این پروژه از چهار ویدئوی تهیه شده استفاده گردید که در هر ویدئو جسم از نقطه اولیه مشخصی توسط یک فرد به یک نقطه هدف که داخل یک حفره قرار گرفته است منتقل می‌شود. برای اینکه بتوانیم از مسیرهای نشان داده شده در ویدئوها در الگوریتم مورد نظر استفاده کنیم لازم است تا مختصات مربوط به جسم در ویدئو تشخیص داده شود. برای این منظور از الگوریتم ردیابی شی[2] استفاده شده است. با اعمال این الگوریتم[3] بر روی ویدئوهای ضبط شده، مختصات مربوط به جسم تشخیص داده شده و مسیر راهنما مشخص می‌گردد. این مسیر، برای راهنمایی مدل به عنوان ورودی به الگوریتم داده می‌شود. در شکل ۴-۳ نمایی از یکی از این ویدئوها نشان داده شده است.

شکل ۴-۳ نمایی از ویدئوهای مسیر راهنما.

[1] Trajectory
[2] Object tracking
[3] https://github.com/bikz05/object-tracker

۴-۳- تنظیم پارامترها و معیار ارزیابی

در این بخش قصد داریم به معرفی پارامترهای مربوط به الگوریتم‌های مورد استفاده در این پایان‌نامه بپردازیم. در بخش ۴-۳-۱ به معرفی پارامترها و معیار ارزیابی الگوریتم Q-Learning می‌پردازیم و در ادامه در بخش ۲-۳-۴ به معرفی پارامترها و معیار ارزیابی الگوریتم PILCO می‌پردازیم.

۱-۳-۴- پارامترها و معیار ارزیابی الگوریتم Q-Learning

پارامترهای مختلفی در الگوریتم Q-Learning وجود دارد که در این بخش به معرفی آنها و شرایط تنظیم آنها می‌پردازیم. یکی از ابتدایی‌ترین پارامترهای موجود در این الگوریتم، ابعاد فضای حالت است که به طور دلخواه تعیین می‌شود. در اکثر آزمایش‌ها، فضای حالت به صورت مربعی در اندازه‌های 9×9 ، 18×18 و 36×36 در نظر گرفته شده است.

در ابتدای همه آزمایش‌ها، موقعیت اولیه جسم نیز به طور دلخواه مشخص می‌شود. در تمامی آزمایش‌های صورت گرفته در این پایان‌نامه، موقعیت اولیه جسم در ناحیه بالا سمت راست فضای حالت در نظر گرفته شده است. هم‌چنین، موقعیت نهایی برای جسم به عنوان یکی از پارامترهای آزمایش مطرح است که در ابتدای آزمایش باید مشخص شود. در اکثر آزمایش‌ها، این پارامتر وسط فضای حالت و درون حفره در نظر گرفته شده است.

مشابه بسیاری از الگوریتم‌های یادگیری، در الگوریتم Q-Learning نیز پارامتری تحت عنوان ضریب یادگیری الگوریتم که با آلفا (α) نشان داده می‌شود مطرح است. این پارامتر، نسبت حرکت الگوریتم در جهت اکتشاف[1] موقعیت‌های جدید و بهره‌برداری[2] از دانش پیشین را مشخص می‌کند. مقدار این پارامتر عددی بین صفر و یک است که هر چه به یک نزدیک‌تر باشد، الگوریتم بیش‌تر اکتشاف می‌کند و هر چه به صفر نزدیک‌تر باشد، بیش‌تر به بهره‌برداری از دانش پیشین روی می‌آورد. با توجه به آزمایش‌های صورت‌گرفته، در اکثر آزمایشات مقدار ۰/۸ برای این پارامتر در نظر گرفته شده است.

یکی دیگر از پارامترهای مطرح در این الگوریتم، پارامتر گاما (γ) می‌باشد که ضریب کاهش[3] نام دارد. بدیهی است پاداش دریافت‌شده در تکرار فعلی اهمیت بیش‌تری از پاداش دریافتی در دوره‌های قبلی دارد.

[1] Exploration
[2] Exploitation
[3] Discount factor

پارامتر ضریب کاهش وظیفه تنظیم تاثیر تدریجی پاداش‌ها در تکرارهای مختلف را دارد، به این صورت که هر چه این عدد بزرگ‌تر باشد اثر پاداش‌های قبلی دیرتر از بین می‌رود. مقدار این پارامتر عددی بین صفر و یک است که با توجه به بررسی‌های انجام شده در آزمایش‌های مختلف، در اکثر آزمایشات مقدار ۰/۵ به عنوان مقدار مناسب برای این پارامتر در نظر گرفته شده است.

با توجه به وجود روند تکراری در الگوریتم Q-Learning یک شرط خاتمه برای آن باید در نظر گرفته شود که از پارامتر تعداد دور تکرار به این منظور استفاده شده است. الگوریتم بعد از رسیدن به تعداد تکرار لازم متوقف می‌شود و این نقطه از اجرا به عنوان نقطه همگرایی در نظر گرفته می‌شود. در تمام آزمایشات این پارامتر ۱۰۰ در نظر گرفته شده است چرا که با بررسی آزمایشات مختلف به این نتیجه رسیده‌ایم که بعد از این تکرار تغییر قابل توجهی در نتایج الگوریتم دیده نمی‌شود.

در الگوریتم Q-Learning یک ماتریس Q وجود دارد که تخمین مناسب بودن هر عمل در هر حالت را نشان می‌دهد. این ماتریس در ابتدا مقداردهی اولیه شده و در طی اجرای الگوریتم به تدریج به مقادیر مناسب هر عمل در هر حالت همگرا می‌شود. تعیین مقدار اولیه مناسب برای این ماتریس می‌تواند باعث همگرایی سریع‌تر و بهتر الگوریتم شود. لذا، این مقدار اولیه می‌تواند به عنوان یکی از پارامترهای الگوریتم در نظر گرفته شود. این ماتریس در الگوریتم Q-Learning با صفر مقداردهی اولیه می‌شود در حالی‌که در الگوریتم Q-Learning با مسیر راهنما، مقدار اولیه این ماتریس با توجه به دانش استخراج شده از ویدئوهای راهنما تعیین می‌گردد.

برای مقایسه آزمایشات مختلف که با پارامترهای مختلف و در شرایط مختلف صورت می‌گیرند، نیاز به یک معیار ارزیابی مقایسه‌ای وجود دارد. معیارهای ارزیابی مختلفی در کارهای پیشین در نظر گرفته شده است که ما در این پایان‌نامه از معیار میانگین تعداد عمل‌های لازم برای رسیدن به هدف استفاده کرده‌ایم. در هر تکرار الگوریتم جسم بعد از انجام تعدادی عمل به هدف می‌رسد. اگر تعداد عمل‌های انجام شده توسط ربات در تکرار i ام را با n_i نشان دهیم در الگوریتمی که دارای k تکرار می‌باشد، معیار میانگین تعداد عمل‌های لازم برای رسیدن به هدف با توجه به معادله(۴-۱) محاسبه می‌شود:

$$avgActCount = \frac{1}{k} \times \sum_{i=1}^{k} n_i$$ **معادله(۴-۱)**

۴-۳-۲- پارامترها و معیار ارزیابی الگوریتم PILCO

در الگوریتم PILCO، پارامترهای بسیار زیادی وجود دارد. این پارامترها، در دسته‌های «تنظیمات کلی»، «تنظیمات Plant»، «تنظیمات تابع سیاست»، «تنظیمات تابع هزینه»، «تنظیمات مدل دینامیکی GP» و «تنظیمات بهینه‌سازی (پارامترهای سیاست)» قرار می‌گیرند. در این بخش، به تفکیک به معرفی اجمالی پارامترهای هر یک از این دسته‌ها می‌پردازیم.

- **تنظیمات کلی**

پارامتر dt، بیانگر زمان نمونه‌برداری است. یا به عبارتی دیگر $1/dt$، فرکانس نمونه‌برداری را مشخص می‌کند. یکی دیگر از پارامترهای کلی، پارامتر T است که طول افق پیش‌بینی در واحد ثانیه را نشان می‌دهد. پارامتر μ_0، میانگین توزیع حالت آغازی مرکز جسم، یعنی $p(x_0)$، را مشخص می‌کند. پارامتر Σ_0، نشان‌دهنده ماتریس کوواریانس توزیع $p(x_0)$ است. N، تعداد دفعاتی است که حلقه اصلی الگوریتم PILCO اجرا می‌شود. J، تعداد rollout های اولیه با یک سیاست تصادفی است. این rollout ها برای جمع‌آوری یک مجموعه داده اولیه برای آموزش اولین مدل دینامیکی فرآیند گوسی استفاده می‌شوند. معمولا این پارامتر ۱ در نظر گرفته می‌شود. پارامتر K، تعداد حالات اولیه‌ای است که برای آنها سیاستی آموخته می‌شود. در PILCO، توزیع‌های حالت آغازی مرکز جسم به صورت معادله (۴-۲) بیان می‌شود که به توزیع‌های با میانگین‌های $\mu_0^{(i)}$ متفاوت و اما با ماتریس‌های کوواریانس مشترک Σ_0 مربوط است.

$$p(x_0) = \sum_{i=1}^{K} \mathcal{N}(\mu_0^{(i)}, \Sigma_0)$$ **معادله (۴-۲)**

- **تنظیمات Plant**

dynamics، پارامتری است که تابع ODE[1] شبیه‌سازی سیستم را مشخص می‌کند. در حالتی که از ربات واقعی استفاده شود، نیازی به مشخص کردن معادله مدل دینامیکی نیست. اما در حالت شبیه‌سازی لازم است معادله مدل دینامیکی مسئله شبیه‌سازی شده را مشخص کنیم که در معادله (۴-۳) آمده است:

[1] Ordinary Differential Equation

$$\frac{dz}{dt} = \begin{bmatrix} z_1 \\ z_2 \\ \frac{f_1(t) - b \times z_3}{m} \\ \frac{f_2(t) - b \times z_4}{m} \end{bmatrix}$$

معادله (۴-۳)

در معادله (۴-۳)، $z = [x, y, v_x, v_y]$ حالت[1] سیستم شبیه‌سازی شده، $f_1(t)$ نیروی وارد بر جسم در جهت x، $f_2(t)$ نیروی وارد بر جسم در جهت y، m جرم جسم و b اصطکاک در مقابل حرکت جسم است.

پارامتر noise، ماتریس کوواریانس نویز اندازه‌گیری در نظر گرفته شده در فرآیند شبیه‌سازی است. در آزمایش‌های صورت گرفته در این پایان‌نامه، فرض بر این است که از یک نویز گوسی با میانگین صفر استفاده شده است. این نویز، در طول rollout تراژکتوری به حالت (پنهان) اضافه می‌شود.

پارامتر ctrl، بیانگر کنترل‌کننده‌ای است که باید اعمال شود. در آزمایش‌های صورت گرفته در این پایان‌نامه، از یک کنترل‌کننده zero-order-hold استفاده شده است. گزینه‌های دیگر first-order-hold و first-order-lag هستند.

prop، تابعی را مشخص می‌کند که توزیع $p(x_{t+1})$ را با استفاده از توزیع $p(x_t)$ (پیش‌بینی احتمالاتی یک‌گامه) محاسبه می‌کند. در آزمایش‌های صورت گرفته، از تابعی به نام propagated استفاده شده که توزیع حالت پیش‌بینی‌کننده $p(x_{t+1})$ و مشتقات جزئی که برای جستجوی سیاست مبتنی بر گرادیان نیاز هستند را محاسبه می‌کند.

- **تنظیمات تابع سیاست**

تابع سیاستی که در آزمایش‌های این پایان‌نامه استفاده شده، تابع خطی است که پارامترهای آن w و b هستند. این دو پارامتر به‌ترتیب، وزن‌های خطی (ماتریس 4×2) و بایاس (ماتریس 1×2) تابع خطی $wx + b$ را مشخص می‌کنند. در آزمایش‌های صورت گرفته، مقدار اولیه آنها صفر در نظر گرفته شده است. توضیحات بیش‌تر در مورد این سیاست، در بخش روش پیشنهادی آورده شده است.

[1] State

maxU، حداکثر مقدار نیرویی که به جسم وارد می‌شود را معین می‌کند. به این صورت که الگوریتم u را در بازه‌ی $[-u_{max}, u_{max}]$ در نظر می‌گیرد.

- **تنظیمات تابع هزینه**

در این بخش، پارامترهای تابع هزینه آنی را معرفی می‌کنیم. پارامتر c، بیانگر تابع هزینه مورد استفاده است. در آزمایش‌های صورت گرفته، از تابع هزینه اشباع شده (که با تفریق یک گوسی نرمال‌نشده از عدد ۱ حاصل می‌شود) با پراکندگی σ_c استفاده شده است که در معادله (۴-۴) بیان شده است. در این معادله، x_{target} حالت هدف را مشخص می‌کند که یکی دیگر از پارامترهای مورد استفاده در الگوریتم PILCO است که باید از قبل معین شود.

$$c(x) = 1 - exp\left(-\frac{1}{2\sigma_c^2}\left\|x - x_{target}\right\|^2\right) \quad \in [0,1]$$ **معادله (۴-۴)**

gamma، پارامتر ضریب کاهش است و از آنجایی که مسئله مورد نظر ما، یک مسئله با افق متناهی است، مقدار این پارامتر را ۱ در نظر گرفته‌ایم.

پارامتر σ_c، بیانگر پراکندگی یا عرض تابع هزینه است. بر حسب تجربه، بهتر است این پارامتر $\frac{\|\mu_0 - x_{target}\|}{10}$ در نظر گرفته شود.

expl، پارامتر اکتشاف از نوع UCB[1] است. مقادیر منفی، باعث اکتشاف بیش‌تر می‌شوند و مقادیر مثبت، باعث ماندن سیاست در مناطق با عملکرد پیش‌بینی خوب می‌شوند. در آزمایش‌های صورت گرفته، مقدار این پارامتر را صفر در نظر گرفته‌ایم تا از هر نوع اکتشاف یا بهره‌برداری اضافی جلوگیری کنیم.

- **تنظیمات مدل دینامیکی GP**

در این بخش، پارامترهای مدل دینامیکی GP را معرفی می‌کنیم.

induce یک پارامتر اختیاری است و زمان تعویض از GP های کامل به GP های پراکنده را مشخص می‌کند. این پارامتر، یک ماتریس به فرم $a \times b \times c$ است که در آن a تعداد ورودی‌های القایی[2] است. b

[1] UCB-type

[2] Inducing iputs . اگر اندازه مجموعه آموزشی از a بزرگ‌تر شود، GP کامل به طور خودکار به تقریب پراکنده‌اش تغییر می‌کند.

برابر با ۰ مشخص می‌کند که هیچ ورودی القایی وجود ندارد. c یا برابر با ۱ است یا برابر با تعداد ابعاد قابل پیش‌بینی است که به تعداد خروجی‌های مدل دینامیکی مربوط می‌شود. برای $c = 1$، ورودی‌های القایی بین همه GP ها به اشتراک گذاشته می‌شوند. در غیر اینصورت، مجموعه‌های ورودی‌های القایی، برای هر بعد قابل پیش‌بینی به طور جداگانه یاد گرفته می‌شوند.

پارامتر trainOpt، تعداد جستجوهای خطی برای آموزش فراپارامتر GP را مشخص می‌کند که (عدد اول) توسط GP کامل و (عدد دوم) توسط GP پراکنده استفاده می‌شود.

- **تنظیمات بهینه‌سازی (یادگیری سیاست)**

در این بخش، پارامترهای اختیاری برای یادگیری سیاست را تعریف می‌کنیم. به طور کلی، از یک بهینه‌ساز مبتنی بر گرادیان غیرمحدب استفاده شده است.

پارامتر length، حداکثر تعداد جستجوهای خطی را مشخص می‌کند که بهینه‌ساز بعد از آن، بهترین مجموعه پارامتر تاکنون را برمی‌گرداند.

پارامتر MFEPLS، حداکثر تعداد ارزیابی‌های تابع در هر جستجوی خطی است. یا جستجوی خطی، موفق به پیدا کردن یک مجموعه پارامتر با گرادیان نزدیک به صفر می‌شود یا موفق نمی‌شود و بعد از MFEPLS بار ارزیابی تابع (و گرادیان) متوقف می‌شود.

برای مقایسه آزمایشات مختلف که با پارامترهای مختلف و در شرایط مختلف صورت می‌گیرند، نیاز به یک معیار ارزیابی مقایسه‌ای وجود دارد. معیارهای ارزیابی مختلفی در کارهای پیشین در نظر گرفته شده است که ما در این پایان‌نامه از معیار درصد موفقیت ربات در رساندن جسم به هدف استفاده کرده‌ایم. هر آزمایش شامل n بار تلاش ربات برای رساندن جسم از موقعیت اولیه به موقعیت هدف است، اگر در این n بار تکرار ربات موفق شود که جسم را در k تکرار به هدف برساند، معیار درصد موفقیت ربات در رساندن جسم به هدف با استفاده از معادله(**۴-۵**) محاسبه می‌شود:

$$successRate = \frac{k}{n} \times 100$$ **معادله(۴-۵)**

۴-۴- نتایج آزمایش

در این بخش آزمایش‌های مختلف صورت گرفته در زمینه‌ی بررسی پارامترهای مختلف بر نتایج انتقال جسم به درون حفره در محیط‌های گسسته و پیوسته گزارش شده‌اند. در ادامه، به ترتیب بخش‌های «بررسی تاثیر اندازه جسم گرفته شده توسط ربات»، «بررسی تاثیر قابلیت چرخاندن جسم گرفته شده توسط ربات»، «بررسی تاثیر تعداد مسیرهای راهنما»، «بررسی تاثیر اندازه فضای حالت» و «بررسی تاثیر در نظر گرفتن محدودیت‌های فضای حالت در محیط پیوسته» خواهند آمد.

۴-۴-۱- بررسی تاثیر اندازه جسم گرفته شده توسط ربات

اندازه جسم گرفته شده توسط ربات، یکی از عوامل موثر در عملکرد الگوریتم به کار گرفته شده توسط ربات است. با توجه به این موضوع در این آزمایش قصد داریم تا به بررسی عملکرد ربات در حالات مختلف اندازه جسم گرفته شده توسط ربات بپردازیم. آزمایش بر اساس دو اندازه یک سلولی و دو سلولی انجام شده است و از معیار میانگین تعداد عمل‌های انجام شده توسط ربات برای رسیدن به مقصد، به منظور مقایسه حالت‌ها استفاده شده است. لیست پارامترهای مورد استفاده در این آزمایش، در جدول ۴-۱ آمده است.

جدول ۴-۱ تنظیمات پارامترهای مورد استفاده در آزمایش بررسی تاثیر اندازه جسم گرفته شده توسط ربات

پارامتر	مقدار	توضیح
فضای حالت	۹،۱۸،۳۶	فضای حالت مربعی در نظر گرفته شده است
مدل مورد بررسی	Q-Learning	----
موقعیت اولیه جسم	راست بالا	جسم در نقطه راست بالای فضای حالت قرار می‌گیرد
موقعیت نهایی جسم	درون حفره	جسم باید درون حفره قرار گیرد
آلفا (α)	۰/۸	ضریب یادگیری
گاما (γ)	۰/۵	ضریب عامل کاهش
تعداد دور تکرار الگوریتم	۱۰۰	----
قابلیت چرخاندن جسم	ندارد	
اندازه جسم	۱،۲	پارامتر مورد آزمایش

جدول ۴-۲ نتایج حاصل از آزمایش بررسی تاثیر اندازه جسم گرفته شده توسط ربات

فضای حالت	اندازه جسم برحسب تعداد سلول	میانگین تعداد عمل‌ها
9×9	۱	۷۶/۶۴
	۲	۸۹/۸۶
18×18	۱	۳۲۰/۳۹
	۲	۴۴۰/۶۴
36×36	۱	۲۱۸۹/۲
	۲	۳۳۰۶/۸

همان‌طور که از قبل قابل پیش‌بینی بود، انتقال جسم بزرگ‌تر به هدف پیچیدگی بیش‌تری دارد لذا به تعداد عمل‌های بیش‌تری نیاز دارد. همان‌طور که از نتایج گزارش شده در جدول ۴-۲ نیز مشخص است، در تمامی فضاهای حالت آزمایش شده تعداد عمل‌های لازم برای انتقال جسم با اندازه ۲ سلول به هدف بیش‌تر از تعداد عمل‌های لازم برای انتقال جسم با اندازه ۱ سلول است.

۲-۴-۴- بررسی تاثیر قابلیت چرخاندن جسم گرفته شده توسط ربات

جهت جسمی که در اختیار ربات قرار گرفته است، می‌تواند در نحوه انتقال جسم به هدف موثر باشد. در بسیاری از موارد، چرخاندن جسم می‌تواند باعث سهولت در انتقال آن به هدف شود. از این نقطه‌نظر، وجود توانایی چرخاندن جسم توسط ربات می‌تواند عامل موثری در عملکرد ربات باشد. با توجه به این موضوع در این آزمایش قصد داریم تا به بررسی عملکرد ربات در دو حالت مختلف بپردازیم. در حالت اول، ربات توانایی چرخاندن جسم را ندارد ولی در حالت دوم، این توانایی برای ربات در نظر گرفته شده است. آزمایش بر اساس این دو حالت انجام شده است و از معیار میانگین تعداد عمل‌های انجام شده توسط ربات برای رسیدن به مقصد، به منظور مقایسه حالت‌ها استفاده شده است. لیست پارامترهای مورد استفاده در این آزمایش، در جدول ۴-۳ به طور خلاصه آمده است.

جدول ۴-۳ تنظیمات پارامترهای مورد استفاده در آزمایش بررسی تاثیر قابلیت چرخاندن جسم گرفته شده توسط ربات

پارامتر	مقدار	توضیح
فضای حالت	۹،۱۸،۳۶	فضای حالت مربعی در نظر گرفته شده است
مدل مورد بررسی	Q-Learning	----
موقعیت اولیه جسم	راست بالا	جسم در نقطه راست بالای فضای حالت قرار می‌گیرد
موقعیت نهایی جسم	درون حفره	جسم باید درون حفره قرار گیرد
آلفا (α)	۰/۸	ضریب یادگیری
گاما (γ)	۰/۵	ضریب عامل کاهش
تعداد دور تکرار الگوریتم	۱۰۰	----
قابلیت چرخاندن جسم	دارد، ندارد	پارامتر مورد آزمایش
اندازه جسم	۲	----

جدول ۴-۴ نتایج حاصل از بررسی تاثیر قابلیت چرخاندن جسم گرفته شده توسط ربات

فضای حالت	قابلیت چرخاندن جسم	میانگین تعداد عمل‌ها
9×9	دارد	۱۶۴/۰۳
	ندارد	۸۹/۸۶
18×18	دارد	۹۰۰/۷۵
	ندارد	۴۴۰/۶۴
36×36	دارد	----
	ندارد	۳۳۰۶/۸

در حالتی که ربات قابلیت چرخاندن جسم را نداشته باشد، در صورتی که در ابتدای کار جهت جسم معادل با جهت هدف جسم نباشد امکان رسیدن جسم به هدف وجود ندارد. با توجه به این موضوع، در تمام آزمایش‌های صورت گرفته در حالتی که ربات قابلیت چرخاندن جسم را ندارد، جهت اولیه جسم معادل با جهت نهایی جسم در نظر گرفته شده است. بنابراین، پیچیدگی مسئله در حالت وجود قابلیت چرخاندن جسم بیش‌تر از حالت دیگر است چرا که علاوه بر ۴ عمل حرکت در چهار جهت اصلی، باید عمل چرخاندن جسم نیز توسط ربات صورت گیرد. با توجه به پیچیدگی بیش‌تر موجود در این حالت، تاخیر در رساندن جسم به هدف قابل‌پیش‌بینی است. نتایج گزارش شده در جدول ۴-۴ نیز موید این موضوع در تمامی فضاهای حالت می‌باشد.

۳-۴-۴- بررسی تاثیر تعداد مسیرهای راهنما

در حالت معمول الگوریتم Q-Learning که ماتریس Q با صفر مقداردهی اولیه می‌شود، در فضای حالت با ابعاد بزرگ‌تر از 20×20 ربات در زمان معقول نمی‌تواند جسم را با شروع از موقعیت اولیه‌ای که بیرون از حفره قرار دارد به موقعیت هدف درون حفره برساند. با توجه به این موضوع، یک راه حل برای کاهش زمان اجرا می‌تواند استفاده از مسیرهای راهنما باشد. این مسیرهای راهنما در قالب مقدار اولیه ماتریس Q در اختیار الگوریتم Q-Learning قرار می‌گیرند. مسیرهای راهنما می‌توانند از دو طریق در اختیار الگوریتم قرار بگیرند. در حالت اول، مسیرهای رساندن جسم به هدف به صورت دستی توسط عامل انسانی طراحی شده و در اختیار الگوریتم قرار گرفته‌اند. استفاده از این روش، در فضاهای حالت با ابعاد بزرگ دارای پیچیدگی زیادی است. در حالت دوم، این مسیرهای راهنما از درون ویدئوهای ضبط شده و با استفاده از الگوریتم‌های ردیابی شی استخراج می‌شوند. در هر دو حالت، تعداد مسیرهای راهنما که در اختیار الگوریتم قرار می‌گیرند می‌توانند در عملکرد الگوریتم تاثیرگذار باشند. هدف از این آزمایش بررسی تاثیر تعداد مسیرهای راهنما است. لیست پارامترهای مورد استفاده در این آزمایش، در جدول ۴-۵ به طور خلاصه آمده است. نتایج آزمایش‌های صورت گرفته در سه حالت جسم ۱ سلولی، ۲ سلولی، ۲ سلولی با چرخش به ترتیب در جدول ۴-۶ ، جدول ۴-۷ و جدول ۴-۸ آمده است.

جدول ۴-۵ تنظیمات پارامترهای مورد استفاده در آزمایش بررسی تاثیر تعداد مسیرهای راهنما

پارامتر	مقدار	توضیح
فضای حالت	۹،۱۸،۳۶،۱۴۴	فضای حالت مربعی در نظر گرفته شده است
مدل مورد بررسی	Q-Learning with guidance	مسیرهای راهنما به صورت دستی در اختیار الگوریتم قرار گرفته است
موقعیت اولیه جسم	راست بالا	جسم در نقطه راست بالای فضای حالت قرار می‌گیرد
موقعیت نهایی جسم	درون	جسم باید درون حفره قرار گیرد
آلفا (α)	۰/۸	ضریب یادگیری
گاما (γ)	۰/۵	ضریب عامل کاهش
تعداد دور تکرار الگوریتم	۱۰۰	----
قابلیت چرخاندن جسم	دارد، ندارد	----
اندازه جسم	۱،۲	----

جدول ۴-۶ نتایج حاصل از بررسی تاثیر تعداد مسیرهای راهنما در انتقال جسم به اندازه ۱ سلول

فضای حالت	تعداد مسیرهای راهنما	میانگین تعداد عمل‌ها
9×9	۱	۳۳/۱۵
	۲	۲۱/۶۹
	۳	۲۷/۵۱
	۴	**۲۱/۶۲**
18×18	۱	۱۹۴/۷۷
	۲	۱۴۲
	۳	۱۱۴/۰۶
	۴	**۲۹/۹۵**
36×36	۱	۲۲۰۶
	۲	۱۰۲۶/۴
	۳	۱۰۶۷/۷
	۴	**۵۵۳/۲۲**

جدول ۴-۷ نتایج حاصل از بررسی تاثیر تعداد مسیرهای راهنما در انتقال جسم به اندازه ۲ سلول بدون قابلیت چرخاندن آن

فضای حالت	تعداد مسیرهای راهنما	میانگین تعداد عمل‌ها
9×9	۱	۳۹/۸۵
	۲	۲۰/۲۱
	۳	**۱۷/۸۴**
	۴	۱۸/۱۹
18×18	۱	۲۴۴/۸۶
	۲	۲۴۴/۷۱
	۳	۸۹/۷۴
	۴	**۵۲/۳**
36×36	۱	۱۵۷۶/۴
	۲	۱۰۲۹/۸
	۳	۱۲۷۳/۴
	۴	**۶۵۵/۶۵**

جدول ۴-۸ نتایج حاصل از بررسی تاثیر تعداد مسیرهای راهنما در انتقال جسم به اندازه ۲ سلول با قابلیت چرخاندن آن

فضای حالت	تعداد مسیرهای راهنما	میانگین تعداد عمل‌ها
144×144	۳	۳۴۵۳۲/۹۴
	۴	**۱۸۳۷۶/۹۴**

همان‌طور که از نتایج ارائه شده در جداول بالا نیز مشخص است، افزایش تعداد مسیرهای راهنما توانسته است میانگین تعداد عمل‌های لازم برای رسیدن به هدف را کاهش دهد. البته توجه به این موضوع حائز اهمیت است که افزودن مسیر راهنما باید دارای ارزش افزوده‌ای برای الگوریتم باشد به این معنی که اگر چندین مسیر همگی در یک بخش از فضای حالت طراحی شده باشند و پراکندگی لازم را در فضای حالت نداشته باشند، لزوما با کاهش معیار ارزیابی روبرو نیستیم. یعنی تنها افزودن مسیر راهنما در بخش‌هایی از فضای حالت که در آنها مسیری طراحی نشده است، می‌تواند باعث بهبود عملکرد ربات گردد.

۴-۴-۴- بررسی تاثیر اندازه فضای حالت

در فضاهای حالت بزرگ، اجرای الگوریتم Q-Learning بدون استفاده از مسیر راهنما عملا امکان‌پذیر نیست و حتما باید از مسیر راهنما استفاده شود. طراحی دستی مسیر راهنما در فضاهای بزرگ با پیچیدگی زیادی همراه است و در نتیجه در چنین حالتی از مسیرهای استخراج شده از ویدئو استفاده می‌شود. در این آزمایش، هدف استفاده از مسیر استخراج شده از ویدئو در ابعاد مختلف فضای حالت می‌باشد. لیست پارامترهای مورد استفاده در این آزمایش، در جدول ۴-۹ به طور خلاصه آمده است. در همه آزمایش‌های این بخش از چهار مسیر راهنما استفاده شده است.

جدول ۴-۹ تنظیمات پارامترهای مورد استفاده در آزمایش بررسی تاثیر اندازه فضای حالت

پارامتر	مقدار	توضیح
فضای حالت	----	**پارامتر مورد بررسی (در جدول نتایج آورده شده است)**
مدل مورد بررسی	Q-Learning with guidance	----
موقعیت اولیه جسم	راست بالا	جسم در نقطه راست بالای فضای حالت قرار می‌گیرد
موقعیت نهایی جسم	درون حفره	جسم باید درون حفره قرار گیرد
آلفا (α)	۰/۸	ضریب یادگیری
گاما (γ)	۰/۵	ضریب عامل کاهش
تعداد دور تکرار الگوریتم	۱۰۰	----
قابلیت چرخاندن جسم	دارد	----
اندازه جسم	----	در جدول نتایج آورده شده است

جدول ۴-۱۰ نتایج حاصل از بررسی تاثیر اندازه فضای حالت

فضای حالت	اندازه جسم	میانگین تعداد عمل‌ها
69×39	4×2	۷۳۱۳/۹
86×49	5×3	۲۵۹۸۳
138×80	9×5	۶۰۸۷۴/۸۸
172×99	11×6	۱۳۵۸۲۴/۱۴
231×132	15×9	۷۴۸۶۲/۶۱

۵-۴-۴- بررسی تاثیر در نظر گرفتن محدودیت‌های فضای حالت در محیط پیوسته

در محیط‌های پیوسته، از الگوریتم PILCO برای انتقال جسم به هدف استفاده کرده‌ایم. این الگوریتم، از یک تابع هزینه برای ارزیابی سیاست استفاده می‌کند. در تابع هزینه‌های معمول هیچ‌گونه اطلاعاتی در مورد محدودیت‌های فضای حالت (مانند دیوارها و موانع موجود در فضای حالت) در نظر گرفته نمی‌شود. تغییر تابع هزینه به نحوی که بتوان محدودیت‌های فضای حالت را به آن تزریق کرد می‌تواند نقش بسیار موثری در عملکرد ربات داشته باشد. در تابع هزینه‌های معمول از فاصله اقلیدسی جسم تا هدف به عنوان یک پارامتر جهت‌دهنده استفاده می‌شود. این در حالی است که در فضای حالتی که در آن مانع یا دیوار وجود داشته باشد، نزدیکی اقلیدسی به هدف لزوما به معنی مناسب بودن سیاست یاد گرفته شده نیست چرا که ممکن است جسم در فاصله‌ای بسیار نزدیک به هدف قرار داشته باشد اما دیوار مانع رسیدن به هدف شود. برای رویارویی با این چالش لازم است که میزان نزدیکی جسم به دیوار نیز به عنوان پارامتری جهت‌دهنده مورد توجه قرار بگیرد. در این پایان‌نامه، تابع هزینه به گونه‌ای ساخته شده است که در آن هم از فاصله اقلیدسی جسم تا هدف و هم از فاصله جسم تا دیوار بهره برده شده است. در این آزمایش، هدف مقایسه تابع هزینه پیشین که هیچ توجهی به محدودیت‌های فضای حالت ندارد با تابع هزینه ارائه شده که محدودیت‌های فضای حالت را نیز در نظر می‌گیرد است. مجموعه پارامترهای مورد استفاده در این آزمایش در جدول ۴-۱۱به طور کامل آورده شده است. نتایج آزمایش نیز برای موقعیت‌های اولیه مختلف جسم نیز در جدول ۴-۱۲ گزارش شده است.

جدول ۴-۱۱ تنظیمات پارامترهای مورد استفاده در آزمایش بررسی تاثیر در نظر گرفتن محدودیت‌های فضای حالت

پارامتر	مقدار	توضیح
مدل مورد بررسی	PILCO	-----
dt	۰/۱	زمان نمونه‌برداری
T	۱۰	طول افق پیش‌بینی

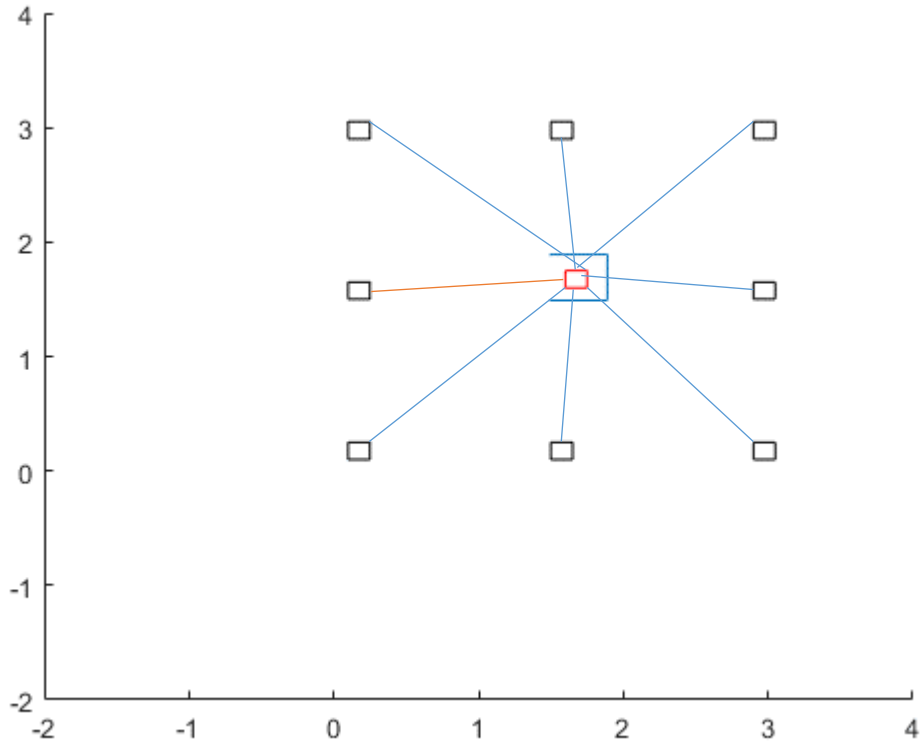
μ_0	پارامتر مورد بررسی	بردار به شکل $[x, y, v_x, v_y]$
Σ_0	ماتریس با درایه‌های قطری ۰/۰۱ و بقیه درایه‌های صفر	----
حالت نهایی جسم	(۱/۵ ، ۱/۶، ۰،۰)	جسم باید در نقطه‌ای درون حفره و به طور ساکن قرار گیرد
N	۵۰	تعداد تکرارها برای بهینه‌سازی
J	۱	----
K	۸	۸ نقطه شروع اولیه مختلف در نظر گرفته شده است
dynamics	dynamics_mini	از مدل دینامیکی در نظر گرفته شده برای این شبیه‌سازی که در بخش روش‌های پیشنهادی توضیح داده شده استفاده شده است
noise	ماتریس با درایه‌های قطری ۰/۰۰۱ و بقیه درایه‌های صفر	نویز اندازه‌گیری هر بعد از حالت ۰/۰۰۱ در نظر گرفته شده است
ctrt	zero-order-hold	**----**
prop	propagated	----
maxU	(۵۰،۵۰)	حداکثر نیروی وارد بر جسم در هر جهت ۵۰ نیوتن در نظر گرفته شده است

c	تابع هزینه اشباع شده	به معادله (۴-۴) رجوع شود
گاما (γ)	۱	ضریب کاهش
σ_c	----	به فاصله بین موقعیت اولیه و نهایی جسم بستگی دارد
expl	۰	عرض تابع هزینه که از نوع UCB است
طول و عرض جسم	(۰/۱۵ ، ۰/۱۵)	در آزمایشات این بخش جسم مربعی در نظر گرفته شده است
induce	ماتریس با درایه‌های صفر و با ابعاد (۱،۰،۳۰۰)	ورودی‌های القایی مشترک
trainOpt	(۵۰۰،۳۰۰)	حداکثر تعداد جستجوهای خطی به هنگام آموزش مدل دینامیکی GP
length	۱۵۰	حداکثر تعداد جستجوهای خطی
MFEPLS	۳۰	حداکثر تعداد ارزیابی‌ها به ازای هر جستجوی خطی

جدول ۴-۱۲ نتایج حاصل از بررسی تاثیر در نظر گرفتن محدودیت‌های فضای حالت

μ_0	**دانش در مورد محدودیت‌های فضای حالت**	**درصد موفقیت**
[0.1,0.1,0,0]	دارد	۵۸
	ندارد	۴۶
[1.5,0.1,0,0]	دارد	۵۶
	ندارد	۴۲
[2.9,0.1,0,0]	دارد	۵۰
	ندارد	۳۰
[2.9,1.5,0,0]	دارد	۴۴
	ندارد	۲۰
[2.9,2.9,0,0]	دارد	۵۴
	ندارد	۳۴
[1.5,2.9,0,0]	دارد	۵۴
	ندارد	۴۰
[0.1,2.9,0,0]	دارد	۵۴
	ندارد	۴۶
[0.1,1.5,0,0]	دارد	۶۶
	ندارد	۸۰

همان‌طور که نتایج گزارش‌شده در جدول ۴-۱۲ نشان می‌دهد، در اکثر موارد (به جز حالت $\mu_0 = [0.1,1.5,0,0]$ که موقعیت ابتدایی جسم دقیقا مقابل حفره قرار دارد) تابع هزینه‌ای که محدودیت‌های فضای حالت را نیز در نظر گرفته توانسته است براساس معیار ارزیابی درصد موفقیت رساندن جسم به هدف، نسبت به تابع هزینه‌ای که توجهی به محدودیت‌های فضای حالت ندارد عملکرد بهتری داشته باشد. برخلاف تمامی موقعیت‌های اولیه، در حالت $\mu_0 = [0.1,1.5,0,0]$ با کاهش عملکرد براساس معیار ارزیابی درصد موفقیت رساندن جسم به هدف روبرو هستیم. تفاوت اصلی که بین این نقطه شروع و سایر نقاط اولیه وجود دارد این است که در نزدیک‌ترین مسیر از نظر فاصله اقلیدسی از این نقطه به هدف مانعی مشاهده نمی‌شود، این در حالی است که کوتاه‌ترین مسیر از نظر فاصله اقلیدسی از سایر نقاط اولیه به نقطه هدف توسط مانع بسته می‌شود. این موضوع در شکل ۴-۴ مشاهده می‌گردد.



شکل ۴-۴ کوتاه‌ترین مسیر از نظر فاصله اقلیدسی از موقعیت‌های اولیه مختلف به هدف.

به طور کلی، در مواقعی که در کوتاه‌ترین مسیر از نظر فاصله اقلیدسی از نقطه اولیه جسم به هدف با مانع روبرو نیستیم، در نظر گرفتن پارامتر جهت‌دهی مرتبط با موانع در تابع هزینه مورد استفاده می‌تواند باعث ایجاد پیچیدگی برای الگوریتم گردد. برای مثال، در مواقعی که جسم بدون برخورد با دیوار، اما با فاصله‌ای

کم از آن وارد حفره می‌شود پارامتر جهت‌دهی مرتبط با موانع باعث انحراف جسم از مسیر معمول شده و باعث ایجاد اختلال در عملکرد الگوریتم می‌گردد. نتایج گزارش شده در جدول ۴-۱۲ برای نقاط اولیه دیگر نیز این‌طور نشان می‌دهد که هر چه مسیرهای کوتاه رساندن جسم از نقطه اولیه به هدف بیش‌تر با موانع همراه باشد، افزودن پارامتر جهت‌دهی مرتبط با موانع کمک به بیش‌تری به راهنمایی ربات می‌نماید.

۵-۴- جمع‌بندی

در این فصل ابتدا به معرفی مجموعه محیط‌های شبیه‌سازی گسسته و پیوسته مورد استفاده در آزمایش‌های انجام گرفته پرداختیم. سپس پارامترهای مختلف الگوریتم Q-Learning و الگوریتم PILCO و روش پیشنهادی بیان شدند و در مورد نحوه‌ی تنظیم هر کدام از آن‌ها بحث شد. معیارهای ارزیابی مورد استفاده برای مقایسه‌ی مدل‌ها نیز معرفی شدند. در بخش انتهایی این فصل، آزمایش‌های مختلفی که در طول انجام این پروژه برای بررسی مدل پایه و همچنین مقایسه‌ی روش پیشنهادی با روش‌های پیشین مورد استفاده قرار گرفته‌اند آورده شده است. آزمایش‌ها نشان می‌دهند که روش پیشنهادی ارائه شده در این پایان‌نامه که از تابع هزینه برای تزریق دانش در مورد فضای حالت در الگوریتم‌های یادگیری تقویتی که به مساله‌ی انتقال جسم به درون حفره می‌پردازند استفاده می‌کند، توانسته است نتایج قابل قبولی را در مقابل روش‌های پیشین که از دانش در مورد فضای حالت استفاده نمی‌کردند کسب کند.

فصل پنجم

۵- نتیجه‌گیری و پیشنهادها

نتیجه‌گیری و پیشنهادها

۵-۱- نتیجه‌گیری

در این پژوهش به مساله‌ی انتقال جسم به درون یک حفره که یکی از مسائل معمول در آموزش ربات‌ها به حساب می‌آید پرداخته شد. در این مساله، هدف انتقال جسم از هر نقطه اولیه‌ای به موقعیت هدفی است که درون یک حفره قرار گرفته است. این وظیفه جزء مطرح‌ترین مسائلی است که در صنایع مختلف بر عهده‌ی ربات‌ها قرار داده می‌شود. الگوریتم‌های مختلفی برای آموزش ربات‌ها برای انجام وظایف ارائه شده است که در این پژوهش به بررسی دو دسته‌ی مهم از الگوریتم‌ها یعنی الگوریتم‌های یادگیری تقویتی و الگوریتم‌های یادگیری تقلیدی پرداخته شد.

آزمایش‌ها بر روی الگوریتم‌های یادگیری تقویتی از قبیل Q-Learning و PILCO نشان داده است که این‌گونه از الگوریتم‌ها بدون در نظر داشتن دانش پیشین قادر به یادگیری مسئله مورد نظر در زمان معقول نخواهند بود. به همین دلیل به عنوان روش پیشنهادی در این پایان‌نامه، دو چارچوب برای تزریق دانش پیشین به دو الگوریتم Q-Learning و PILCO معرفی شد. در روش پیشنهادی ارائه شده برای الگوریتم تقویتی Q-Learning، از مسیرهای راهنما برای بهبود عملکرد الگوریتم و کاهش زمان یادگیری استفاده شد. این مسیرهای راهنما از درون ویدئوهای ضبط شده و با استفاده از مدل‌های ردیابی جسم استخراج شدند. با افزودن مسیرهای راهنما به الگوریتم یادگیری تقویتی Q-Learning، این الگوریتم به سمت یک الگوریتم یادگیری تقلیدی سوق داده شد. همچنین در روش پیشنهادی دیگری که برای تزریق دانش پیشین به الگوریتم PILCO ارائه شد، تابع هزینه الگوریتم PILCO به نحوی تغییر داده شد که بتواند محدودیت‌های فضای حالت از جمله دیوارها را نیز در نظر بگیرد. آزمایش‌های مختلفی برای بررسی عملکرد روش‌های پیشنهادی انجام شد که همگی نشان از عملکرد بهتر این روش‌ها به نسبت روش‌های پایه‌ی الگوریتم‌های Q-Learning و PILCO دارند.

۵-۲- کارهای پیشنهادی

روش پیشنهادی ارائه شده در این پایان‌نامه مشابه خیلی از کارهای یادگیری تقویتی در مسائل کنترل پیوسته، برای اینکه اطلاعات هدف مورد نظر را در اختیار عامل قرار داده و همچنین برای ساده کردن مسئله اکتشاف به ساختن تابع پاداش مناسب متکی است. در حالی که در کار رباتیکی که در این پایان‌نامه

مورد بررسی قرار گرفت و در کارهای مشابه آن که در آنها به آسانی می‌توان موقعیت هدف نهایی را معین کرد، تنظیم یک تابع پاداش مناسب که منجر به راه‌حل‌های خوب می‌شود دشوار است. می‌توان این کار سخت تنظیم توابع پاداش را با نمایش‌های انسان از کار رباتیکی مورد نظر جایگزین کرد. این کار مسئله اکتشاف را آسان می‌کند بدون اینکه نیازی به تنظیم دقیق تابع پاداش باشد. همچنین، از آنجایی که در روش PILCO از فرآیندهای گوسی برای آموزش مدل دینامیکی سیستم مورد بررسی استفاده می‌کند و با توجه به اینکه پیچیدگی محاسباتی استنتاج تقریبی استفاده شده در فرآیندهای گوسی $\mathcal{O}(n^2D^2E)$ است و در مسئله بررسی شده در این پایان‌نامه، هرچه تعداد داده‌های آموزشی فرآیند گوسی یا n بیش‌تر شود مدل بهتری از سیستم به دست می‌آید، می‌توان از روش‌های جدیدتر از قبیل [120] استفاده کرد که هم از لحاظ داده-کارایی قابل مقایسه با روش PILCO هستند و هم پیچیدگی محاسباتیشان نسبت به تعداد داده‌های آموزش مدل خطی است. در مقاله [120]، یک رویکرد یادگیری تقویتی مبتنی بر مدل ارائه شده است که از شبکه‌های عصبی بیزی برای تخمین مدل دینامیکی و تابع سیاست استفاده می‌کند.

مراجع

[1] M. Večerík, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, “Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards,” 2017.

[2] C. J. C. H. W. and P. Dayan, “Q-learning,” *Mach. Learn. J.*, vol. 8, 1992.

[3] M. P. D. and C. E. Rasmussen, “PILCO: A Model-Based and Data-Efficient Approach to Policy Search,” *Proc. Int. Conf. Mach. Learn.*, pp. 465–472, 2011.

[4] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, “Robot Programming by Demonstration,” *Springer Handb. Robot.*, no. February, pp. 1371–1394, 2008.

[5] S. Raza, S. Haider, and M. A. Williams, “Teaching coordinated strategies to soccer robots via imitation,” *2012 IEEE Int. Conf. Robot. Biomimetics, ROBIO 2012 - Conf. Dig.*, pp. 1434–1439, 2012.

[6] S. Schaal, “Is imitation learning the route to humanoid robots?,” *Trends Cogn. Sci.*, vol. 3, no. 6, pp. 233–242, 1999.

[7] and others Claude Sammut, Scott Hurst, Dana Kedzier, Donald Michie, “Learning to fly,” *Proc. 9th Int. Work. Mach. Learn.*, pp. 385–393, 2014.

[8] D. Silver, J. A. Bagnell, and A. Stentz, “High Performance Outdoor Navigation from Overhead Data using Imitation Learning,” *Robot. Sci. Syst.*, 2008.

[9] M. J. Matarić, “Getting humanoids to move and imitate,” *IEEE Intell. Syst. Their Appl.*, vol. 15, no. 4, pp. 18–24, 2000.

[10] J. K. and J. R. Peters, “Policy search for motor primitives in robotics,” *Adv. Neural Inf. Process. Syst.*, pp. 849–856.

[11] E. Berger, H. Ben Amor, D. Vogt, and B. Jung, “Towards a Simulator for Imitation Learning with Kinesthetic Bootstrapping,” *Work. “The Universe Rob. Simulators”, Int. Conf. Simulation, Model. Program. Auton. Robot.*, pp. 167–173, 2008.

[12] and G. S. Christian Thurau, Christian Bauckhage, “Imitation learning at all levels of Game-AI,” *Proc. Int. Conf. Comput. Games, Artif. Intell. Des. Educ.*, vol. 5, pp. 402–408, 2004.

[13] J. Kober and J. Peters, “Imitation and reinforcement learning,” *IEEE Robot. Autom. Mag.*, vol. 17, no. 2, pp. 55–62, 2010.

[14] and J. P. Jens Kober, J. Andrew Bagnell, “Reinforcement Learning in Robotics: A Survey,” *Int. J. Rob. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.

[15] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne, “Imitation Learning: A Survey of Learning Methods,” *ACM Comput. Surv.*, vol. April, pp. 0–35, 2017.

[16] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, “A survey of robot learning from demonstration,” *Rob. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[17] C. Nehaniv and K. Dautenhahn, “The Correspondence Problem,” *Imitation Anim. Artifacts*, pp. 1–40, 2002.

[18] A. Schaal, Stefan, Ijspeert, Auke, Billard, "Computational approaches to motor learning by imitation," *Philos. Trans. R. Soc. London B Biol. Sci.*, vol. 358, no. 1431, 2003.

[19] A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi, "An atlas of physical human--robot interaction," *Mech. Mach. Theory*, vol. 43, no. 3, pp. 253–270, 2008.

[20] S. Chernova and M. Veloso, "Confidence-based policy learning from demonstration using gaussian mixture models," *Proc. 6th Int. Jt. Conf. Auton. agents multiagent Syst.*, p. 233, 2007.

[21] J. A. Bagnell and S. Ross, "Efficient Reductions for Imitation Learning," *Proc. 13th Int. Conf. Artif. Intell. Stat. 2010*, vol. 9, pp. 661–668, 2010.

[22] N. D. Ratliff, D. Bradley, J. A. Bagnell, and J. Chestnutt, "Boosting Structured Prediction for Imitation Learning," *Adv. Neural Inf. Process. Syst.*, pp. 1153–1160, 2006.

[23] S. Ross, G. Gordon, and J. A. (Drew) Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," *Proc. 14th Int. Conf. Artifical Intell. Stat.*, 2011.

[24] R. Rahmatizadeh, P. Abolghasemi, and L. Bölöni, "Learning Manipulation Trajectories Using Recurrent Neural Networks," *arXiv*, 2016.

[25] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, "Learning from demonstration and adaptation of biped locomotion," *Rob. Auton. Syst.*, vol. 47, no. 2–3, pp. 79–91, 2004.

[26] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *Int. J. Rob. Res.*, vol. 32, no. 3, pp. 263–279, 2013.

[27] a J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning Attractor Landscapes for Learning Motor Primitives," *Adv. Neural Inf. Process. Syst. 15*, pp. 1547–1554, 2002.

[28] J. Kober and J. Peters, "Learning motor primitives for robotics," *2009 IEEE Int. Conf. Robot. Autom.*, pp. 2112–2118, 2009.

[29] S. Schaal, P. Mohajerian, and A. Ijspeert, "Dynamics systems vs. optimal control - a unifying view," *Prog. Brain Res.*, vol. 165, pp. 425–445, 2007.

[30] S. Calinon, Z. Li, T. Alizadeh, N. G. Tsagarakis, and D. G. Caldwell, "Statistical dynamical systems for skills acquisition in humanoids," *IEEE-RAS Int. Conf. Humanoid Robot.*, pp. 323–329, 2012.

[31] L. Rozo, D. Bruno, S. Calinon, and D. G. Caldwell, "Learning optimal controllers in human-robot cooperative transportation tasks with position and force constraints," *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 2015–Decem, pp. 1024–1030, 2015.

[32] N. Chen, J. Bayer, S. Urban, and P. Van Der Smagt, "Efficient movement representation by embedding Dynamic Movement Primitives in deep autoencoders," *IEEE-RAS Int. Conf. Humanoid Robot.*, vol. 2015–Decem, pp. 434–440, 2015.

[33] and G. C. Darrin C. Bentivegna, Christopher G. Atkeson, "Learning tasks from observation and practice," *Rob. Auton. Syst.*, vol. 47, no. 2, pp. 163–169, 2004.

[34] S. Chernova and M. Veloso, "Teaching collaborative multi-robot tasks through demonstration," *2008 8th IEEE-RAS Int. Conf. Humanoid Robot. Humanoids 2008*, pp. 385–390, 2008.

[35] J. Togelius, R. De Nardi, and S. M. Lucas, "Towards automatic personalised content creation for racing games," *Proc. 2007 IEEE Symp. Comput. Intell. Games, CIG 2007*, pp. 252–259, 2007.

[36] S. Calinon and A. Billard, "A framework integrating statistical and social cues to teach a humanoid robot new skills," *Proc IEEE Int. Conf. Robot. Autom. ICRA Work. Soc. Interact. with Intell. Indoor Robot.*, 2008.

[37] L.-J. Lin, "Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching," *Mach. Learn.*, vol. 8, pp. 293–321, 1992.

[38] L. Lin, "Programming Robots Using Reinforcement Learning and Teaching.," *Aaai*, vol. 2, pp. 781–786, 1991.

[39] F. Guenter, M. Hersch, S. Calinon, and A. Billard, "Reinforcement learning for imitating constrained reaching movements," *Adv. Robot.*, vol. 21, no. 13, pp. 1521–1544, 2007.

[40] K. S. et al. D Silver J Schrittwieser, "Mastering the game of Go with deep neural networks and tree search.," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[41] T. Brys, A. Harutyunyan, V. U. Brussel, and M. E. Taylor, "Reinforcement Learning from Demonstration through Shaping," *Proc. Twenty-Fourth Int. Jt. Conf. Artif. Intell.*, no. Ijcai, pp. 3352–3358, 2015.

[42] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations : Theory and application to reward shaping," *Sixt. Int. Conf. Mach. Learn.*, vol. 99, pp. 278–287, 1999.

[43] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," *Twenty-first Int. Conf. Mach. Learn. - ICML '04*, p. 1, 2004.

[44] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," *IEEE Int. Conf. Robot. Autom. 2004. Proceedings. ICRA '04. 2004*, p. 2619–2624 Vol.3, 2004.

[45] J. Peters and S. Schaal, "Reinforcement learning of motor skills with policy gradients," *Neural Networks*, vol. 21, no. 4, pp. 682–697, 2008.

[46] A. J. I. A. J. Ijspeort, J. N. J. Nakanishi, and S. Schaal, "Learning rhythmic movements by demonstration using nonlinear oscillators," *IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, vol. 1, no. October, pp. 958–963, 2002.

[47] S. Levine and V. Koltun, "Guided Policy Search," *Proc. 30th Int. Conf. Mach. Learn.*, vol. 28, pp. 1–9, 2013.

[48] M. Zhang, Z. McCarthy, C. Finn, S. Levine, and P. Abbeel, "Learning deep neural

network policies with continuous memory states," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2016–June, pp. 520–527, 2016.

[49] X. Guo, S. Singh, H. Lee, R. Lewis, and X. Wang, "Deep Learning for Real-Time Atari Game Play Using Offline Monte-Carlo Tree Search Planning," *Adv. Neural Inf. Process. Syst. 27*, vol. 2600, pp. 3338–3346, 2014.

[50] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End Training of Deep Visuomotor Policies," 2015.

[51] S. N. and D. Floreano, "Evolutionary Robotics: The Biology, Intelligence, and Technology," 2000.

[52] N. Rokbani, A. Zaidi, and A. M. Alimi, "Prototyping a biped robot using an educational robotics kit," *2012 Int. Conf. Educ. e-Learning Innov. ICEELI 2012*, 2012.

[53] Y. Zhang, S. Wang, and G. Ji, "A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications," *Math. Probl. Eng.*, vol. 2015, 2015.

[54] C. Zhang, Z. Zhen, D. Wang, and M. Li, "UAV path planning method based on ant colony optimization," *Control Decis. Conf. (CCDC), 2010 Chinese*, pp. 3790–3792, 2010.

[55] R. Aler, O. Garcia, and J. M. Valls, "Correcting and improving imitation models of humans for Robosoccer agents," *Proc. 2005 IEEE Congr. Evol. Comput.*, vol. 3, p. 2402–2409 Vol. 3, 2005.

[56] J. Ortega, N. Shaker, J. Togelius, and G. N. Yannakakis, "Imitating human playing styles in Super Mario Bros," *Entertain. Comput.*, vol. 4, no. 2, pp. 93–104, 2013.

[57] T. Sun, C. Huo, S. Tsai, and C. Liu, "Optimal UAV Flight Path Planning Using Skeletonization and Particle Swarm Optimizer," *Proc. 2008 IEEE Congr. Evol. Comput. (IEEE World Congr. Comput. Intell.*, pp. 1183–1188, 2008.

[58] R. Cheng and Y. Jin, "A Social Learning Particle Swarm Optimization Algorithm for Scalable Optimization," *Inf. Sci. (Ny).*, vol. 291, no. 2015, pp. 43–60, 2015.

[59] F. Gruau and K. Quatramaran, "Cellular Encoding for Interactive Evolutionary Robotics," *Proc. Eur. Conf. Artif. Life*, pp. 368–377, 1997.

[60] J. C. Bongard and G. S. Hornby, "Combining fitness-based search and user modeling in evolutionary robotics," *Proceeding fifteenth Annu. Conf. Genet. Evol. Comput. Conf. - GECCO '13*, pp. 159–166, 2013.

[61] H.-I. Institute of Electrical and Electronics Engineers., Y.-C. IEEE Computer Society., C.-L. Chinese Automatic Control Society., IEEE Control Systems Society., Guo li jiao tong da xue (Taiwan), and Asian Control Association., "Evaluation of Human-Robot Arm Movement Imitation," *Control Conf. (ASCC), 2011 8th Asian*, pp. 287–292, 2011.

[62] H. El-Hussieny, S. F. M. Assal, A. A. Abouelsoud, S. M. Megahed, and T. Ogasawara, "Incremental learning of reach-to-grasp behavior: A PSO-based Inverse optimal control approach," *Proc. 2015 7th Int. Conf. Soft Comput. Pattern*

Recognition, SoCPaR 2015, pp. 129–135, 2016.

[63] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2010.

[64] L. Torrey, T. Walker, J. Shavlik, and R. Maclin, "Using advice to transfer knowledge acquired in one reinforcement learning task to another," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 3720 LNAI, pp. 412–424, 2005.

[65] L. Torrey and J. Shavlik, "Transfer Learning," *Handb. Res. Mach. Learn. Appl.*, pp. 2–4, 2009.

[66] G. Kuhlmann and P. Stone, "Graph-based domain mapping for transfer learning in general games," *Mach. Learn. ECML 2007*, no. September, pp. 188–200, 2007.

[67] T. Brys, A. Harutyunyan, M. E. Taylor, and A. Nowé, "Policy Transfer using Reward Shaping," *Auton. Agents Multiagent Syst.*, pp. 181–188, 2015.

[68] M. Wulfmeier, P. Ondruska, and I. Posner, "Maximum Entropy Deep Inverse Reinforcement Learning," *AAAI Conf. Artif. Intell.*, pp. 1433–1438, 2015.

[69] G. Lee, M. Luo, F. Zambetta, and X. Li, "Learning a Super Mario controller from examples of human play," *Proc. 2014 IEEE Congr. Evol. Comput. CEC 2014*, pp. 1–8, 2014.

[70] K. Judah, A. P. Fern, and T. G. Dietterich, "Active Imitation Learning via Reduction to I.I.D. Active Learning," *Uai*, pp. 428–437, 2012.

[71] S. Ikemoto, H. Amor, T. Minato, B. Jung, and H. Ishiguro, "Physical Human-Robot Interaction: Mutual Learning and Adaptation," *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 24–35, 2012.

[72] S. Calinon and A. G. Billard, "What is the Teacher's Role in Robot Programming by Demonstration? Toward Benchmarks for Improved Learning," *Interact. Stud.*, vol. 8, no. 3, pp. 441–464, 2007.

[73] H. Daumé, J. Langford, and D. Marcu, "Search-based structured prediction," *Mach. Learn.*, vol. 75, no. 3, pp. 297–325, 2009.

[74] H. M. Le, A. Kang, Y. Yue, and P. Carr, "Smooth Imitation Learning for Online Sequence Prediction," 2016.

[75] A. Droniou, S. Ivaldi, and O. Sigaud, "Learning a repertoire of actions with deep neural networks," *IEEE ICDL-EPIROB 2014 - 4th Jt. IEEE Int. Conf. Dev. Learn. Epigenetic Robot.*, pp. 229–234, 2014.

[76] H. Mayer, F. Gomez, D. Wierstra, I. Nagy, A. Knoll, and J. Schmidhuber, "A system for robotic heart surgery that learns to tie knots using recurrent neural networks," *Adv. Robot.*, vol. 22, no. 13–14, pp. 1521–1537, 2008.

[77] M. L. Puterman, "Markov Decision Processes--Discrete Stochastic Dynamic Programming," 1994.

[78] R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction,” 1988.

[79] R. Howard, “Dynamic Programming and Markov Processes,” p. 42, 1960.

[80] R. E. Bellman, *Dynamic Programming*. 1957.

[81] M. L. Puterman and M. C. Shin, “Modified Policy Iteration Algorithms for Discounted Markov Decision Problems,” *Manage. Sci.*, vol. 24, no. 11, pp. 1127–1137, 1978.

[82] A. G. Barto, S. J. Bradtke, and S. P. Singh, “Learning to act using real-time dynamic programming,” *Artif. Intell.*, vol. 72, no. 1–2, pp. 81–138, 1995.

[83] R. R. S. Sutton, “Generalization in Reinforcement Learning : Successful Examples Using Sparse Coarse Coding,” *Adv. Neural Inf. Process. Syst.*, vol. 8, pp. 1038–1044, 1996.

[84] R. S. Sutton, “Dyna, an integrated architecture for learning, planning, and reacting,” *ACM SIGART Bull.*, vol. 2, no. 4, pp. 160–163, 1991.

[85] A. Moore and C. Atkeson, “Prioritized sweeping: Reinforcement learning with less data and less time,” *Mach. Learn.*, vol. 13, pp. 103–130, 1993.

[86] C. G. Atkeson and J. C. Santamaria, “A comparison of direct and model-based reinforcement learning,” *Proc. Int. Conf. Robot. Autom.*, vol. 4, no. April, pp. 3557–3564, 1997.

[87] E. W. Aboaf, S. Drucker, and C. G. Atkeson, “Task-level robot learning: juggling a tennis ball more accurately,” *Robot. Autom. 1989. Proceedings., 1989 IEEE Int. Conf.*, pp. 1290–1295, 1989.

[88] J. Peters, K. Mülling, and Y. Altun, “Relative Entropy Policy Search.,” *Proc. 24th Natl. Conf. Arti - cial Intell.*, 2010.

[89] J. Kober and J. Peter, “Policy search for motor primitives in robotics,” *Mach. Learn.*, pp. 1–33, 2010.

[90] M. P. Deisenroth, “A Survey on Policy Search for Robotics,” *Found. Trends Robot.*, vol. 2, no. 1–2, pp. 1–142, 2011.

[91] R. J. Willia, “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning,” *Mach. Learn.*, vol. 8, no. 3, pp. 229–256, 1992.

[92] C. Daniel, G. Neumann, and J. Peters, “Hierarchical Relative Entropy Policy Search,” *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS 2012)*, vol. XX, pp. 273–281, 2012.

[93] J. Schneider, “Exploiting Model Uncertainty Estimates for Safe Dynamic Control Learning,” *Adv. Neural Inf. Process. Syst.*, 1997.

[94] J. M. and C. G. Atkeson, “Minimax Differential Dynamic Programming : An Application to Robust Biped Walking,” *Adv. Neural Inf. Process. Syst.*

[95] S. Schaal and C. G. Atkeson, “Constructive incremental learning from only local information,” *Neural Comput.*, vol. 10, no. 8, pp. 2047–2084, 1998.

[96] D. C. M. and K. Glover, “Robust Controller Design Using Normalized Coprime Factor Plant Descriptions,” *Control Inf. Sci.*, vol. 138, 1990.

[97] K. J. Åström and B. Wittænmark, “Adaptive Control,” *Dover Publ.*, 2008.

[98] B. Wittenmark, “Adaptive dual control methods: An overview,” *IFAC Symp. Adapt. Syst. Control Signal Process.*, pp. 67–72, 1995.

[99] A. A. Fel’dbaum, “Dual-Control Theory, Parts I and II,” *Autom. Remote Control*, vol. 21, no. 11, pp. 874–880, 1961.

[100] S. G. Fabri and V. Kadirkamanathan, “Dual adaptive control of nonlinear stochastic systems using neural networks,” *Automatica*, vol. 34, no. 2, pp. 245–253, 1998.

[101] P. Abbeel, M. Quigley, and A. Y. Ng, “Using inaccurate models in reinforcement learning,” *Proc. 23rd Int. Conf. Mach. Learn. - ICML ’06*, pp. 1–8, 2006.

[102] C. E. Rasmussen and C. K. I. Williams, “Gaussian Processes for Machine Learning,” *Massachusetts Inst. Technol.*, p. 267, 2006.

[103] A. Y. Ng and M. Jordan, “PEGASUS: A Policy Search Method for Large MDPs and POMDPs,” *Conf. Uncertain. Artif. Intell.*, vol. 94720, pp. 406–415, 2000.

[104] and E. L. A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, “Autonomous helicopter flight via Reinforcement Learning,” *Adv. Neural Inf. Process. Syst. 16*, vol. 16, pp. 363–372, 2004.

[105] J. Ko, D. J. Klein, D. Fox, and D. Haehnel, “Gaussian Processes and Reinforcement Learning for Identi cation and Control of an Autonomous Blimp,” *Proc. Int. Conf. Robot. Autom.*, pp. 742–747, 2007.

[106] and E. Andrew Y. Ng, Adam Coates, Mark Diel, Varun Ganapathi, Jamie Schulte, Ben Tse, Eric Berger and Liang, “Autonomous Inverted helicopter flight via Reinforcement Learning,” *Exp. Robot. IX. Springer*, pp. 363–372, 2004.

[107] J. Bagnell and J. Schneider, “Autonomous helicopter control using reinforcement learning policy search methods,” *Int. Conf. Robot. Autom.*, pp. 1615–1620, 2001.

[108] M. Deisenroth, C. Rasmussen, and D. Fox, “Learning to control a low-cost manipulator using data-efficient reinforcement learning,” *Robot. Sci. Syst. VII*, pp. 57–64, 2011.

[109] W. S. Cleveland and S. J. Devlin, “Locally-Weighted Regression: An Approach to Regression Analysis by Local Fitting,” *J. Am. Stat. Assoc.*, vol. 83, pp. 596–610, 1988.

[110] B. D. O. A. and J. B. Moore, “Optimal filtering,” *Dover Publ.*, 2005.

[111] S. J. Julier and J. K. Uhlmann, “Unscented Filtering and Nonlinear Estimation,” *Comput. Eng.*, vol. 92, no. 3, pp. 401–422, 2004.

[112] K. Xiong, H. Y. Zhang, and C. W. Chan, “Performance evaluation of UKF-based nonlinear filtering,” *Automatica*, vol. 42, no. 2, pp. 261–270, 2006.

[113] C. M. Bishop, “Pattern Recognition and Machine Learning,” *Inf. Sci. Stat.*, p. 407,

2006.

[114] J. a. A. Nelder, R. Mead, B. J. a Nelder, and R. Mead, “A Simplex Method for Function Minimization,” *Comput. J.*, vol. 7, no. 4, pp. 308–313, 1964.

[115] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial Intelligence A Modern Approach*, vol. 27. Pearson Education, 2003.

[116] J. Quiñonero-candela, C. E. Rasmussen, and R. Herbrich, “A unifying view of sparse approximate Gaussian process regression,” *J. Mach. Learn. Res.*, vol. 6, pp. 1935–1959, 2005.

[117] D. Nguyen-Tuong, M. Seeger, and J. Peters, “Model learning with local gaussian process regression,” *Adv. Robot.*, vol. 23, no. 15, pp. 2015–2034, 2009.

[118] M. Deisenroth, *Efficient Reinforcement Learning Using Gaussian Processes*. KIT Scientific Publishing, 2010.

[119] M. Toussaint and C. Goerick, “A bayesian view on motor control and planning,” *Stud. Comput. Intell.*, vol. 264, pp. 227–252, 2010.

[120] Higuera, Juan Camilo Gamboa, David Meger, and Gregory Dudek. "Synthesizing Neural Network Controllers with Probabilistic Model-Based Reinforcement Learning." In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2538-2544. IEEE, 2018.

Abstract

Today, the use of robots in various industries and different tasks is rapidly increasing. Using robots decreases human resources' costs and increases accuracy of performance. One of the common tasks in various industries in which robots are used is moving an object from an initial position to a target position. Various algorithms are provided for training different tasks to robots. Two important categories of these algorithms are reinforcement learning and imitation learning algorithms. In imitation learning, an agent is taught to accomplish a task by learning a mapping between observations and actions from demonstrations. In reinforcement learning, the robot is trained to perform a task by considering rewards and penalties for each action.

In this thesis, based on imitation learning methods guiding paths have been used to improve the performance of the Q-Learning algorithm. To extract guiding paths, we have used demonstrations in which an object is moved into a hole by a human.

Recently, the PILCO algorithm is used to solve problems where the design of the dynamic model of the robot is very complicated or the dynamic model is uncertain. This algorithm, which uses probabilistic inference and models for learning control, has been used in various problems in the past, but has not been used to solve the problem of moving an object into a hole by simulated manipulator. Given the uncertainty inherent in this problem, PILCO algorithm can be used as an appropriate solution to solve this problem in continuous environments. Hence, in this thesis, an approach based on the PILCO algorithm is presented. This algorithm is modified in such a way that it takes into account state space constraints in its computational process. In order to evaluate the proposed methods, simulations are performed in discrete and continuous environments. It is worth mentioning that a simulated robot is used to move an object from an initial position to a target position. Simulation results in these two discrete and continuous environments show that the proposed algorithms perform better than the original algorithms, Q-learning algorithms and PILCO algorithms.

Abstract

Key Words: Imitation Learning, Guiding Paths, Reinforcement Learning, Probabilistic Models, Robotics

Amirkabir University of Technology
(Tehran Polytechnic)

Department of Robotics Engineering

M.Sc. Thesis

Probabilistic Model based Imitation Learning in Robotic Applications

By
Ali Javadi

Supervisors
Dr. Saeed Shiry Ghidary
Dr. Ahmad Nickabadi

July 2018