
بسم الله الرحمن الرحيم

دانشگاه صنعتی اصفهان
دانشکده مهندسی برق و کامپیوتر

یادگیری تقویتی عمیق

گزارش پروژه

پرنیان ملک‌زاده

استاد
دکتر محمد حسین منشئی

۱۴۰۳

فهرست مطالب

فهرست تصاویر

فهرست جداول

چکیده

در سال‌های اخیر، پیشرفت‌های قابل‌توجهی در حل مسائل چالش‌برانگیز در حوزه‌های مختلف با استفاده از یادگیری تقویتی عمیق[1] حاصل شده است. بازتولید[2] کارهای موجود و ارزیابی دقیق بهبودهای ارائه‌شده توسط روش‌های جدید برای حفظ این پیشرفت‌ها بسیار مهم است. متأسفانه، بازتولید نتایج برای روش‌های پیشرفته یادگیری عمیق تقویتی اغلب به‌سادگی امکان‌پذیر نیست. به‌ویژه، غیرقابل‌پیش‌بینی بودن محیط‌های استاندارد معیار، همراه با واریانس[3] ذاتی در روش‌ها، می‌تواند تفسیر نتایج گزارش‌شده را دشوار کند. بدون معیار[4]های معنادار و استانداردسازی دقیق گزارش‌های آزمایشی، دشوار است که مشخص کنیم آیا بهبودها نسبت به حالت پیشرفته قبلی معنادار هستند یا خیر. در این مقاله، چالش‌های مربوط به قابلیت بازتولید، تکنیک‌های آزمایشی صحیح و روش‌های گزارش‌دهی را بررسی می‌کنیم. ما به وضوح نشان می‌دهیم که چقدر متغیر بودن متریک[5]ها و نتایج گزارش‌شده هنگام مقایسه با مبناهای رایج می‌تواند متفاوت باشد و پیشنهاداتی برای راهنمایی جهت قابل‌بازتولیدتر شدن نتایج در یادگیری تقویتی عمیق ارائه می‌دهیم. هدف ما ایجاد بحث در مورد چگونگی تضمین پیشرفت مستمر در این حوزه با به حداقل رساندن تلاش‌های هدررفته ناشی از نتایج غیرقابل‌بازتولید و به‌راحتی قابل‌تفسیر است.

واژه‌های کلیدی: ۱- یادگیری تقویتی عمیق، ۲- قابلیت بازتولید، ۳- تکنیک‌های آزمایشی، ۴- محیط‌های معیار

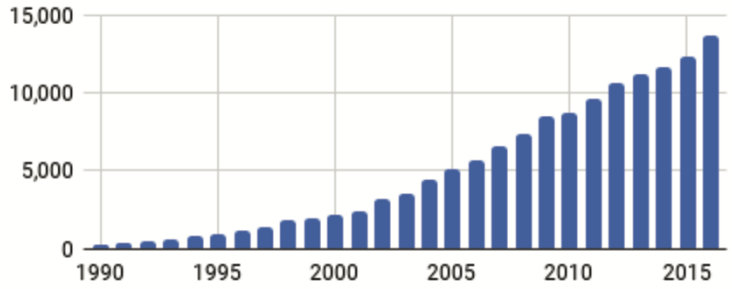
[1] Deep Reinforcement Learning
[2] Reproducing
[3] Variance
[4] Benchmark
[5] Metric

فصل اول
مقدمه

یادگیری تقویتی (RL) شاخه‌ای از مطالعه است که به بررسی چگونگی تعامل[1] یک عامل[2] با محیط[3] خود برای یادگیری یک سیاست[4] که هدف آن بیشینه کردن پاداش[5]های تجمعی[6] مورد انتظار برای یک وظیفه[7] است، می‌پردازد. در سال‌های اخیر، یادگیری تقویتی به دلیل نتایج امیدبخش در زمینه‌هایی مانند کنترل سیستم‌های پیوسته در رباتیک [۱] ، بازی شطرنج [۲]، بازی‌های آتاری [۳] و بازی‌های ویدئویی رقابتی [۴] و [۵] توجه و علاقه زیادی را به خود جلب کرده است. شکل ۱-۱ رشد این زمینه را از طریق تعداد مقالات منتشر شده در هر سال نشان می‌دهد. برای حفظ پیشرفت سریع در تحقیقات یادگیری تقویتی، مهم است که کارهای موجود به راحتی قابل بازتولید و مقایسه باشند تا بتوان به دقت بهبودهای ارائه شده توسط روش‌های نوین را ارزیابی کرد. با این حال، بازتولید نتایج یادگیری عمیق تقویتی اغلب ساده نیست و ادبیات این حوزه نتایج متنوعی برای همان الگوریتم‌های پایه گزارش می‌دهد [۶]. بازتولیدپذیری می‌تواند تحت تأثیر عوامل خارجی (مانند ابرپارامتر[8]ها یا کدهای پایه[9] استفاده‌شده) و عوامل داخلی (مانند اثرات دانه‌های تصادفی[10] یا خواص محیط) قرار گیرد. ما این منابع تفاوت در نتایج گزارش شده را از طریق یک مجموعه نماینده از آزمایش‌ها بررسی می‌کنیم. برای شفافیت، تمرکز خود را بر روی روش‌های گرادیان سیاست[11] (PG) در کنترل پیوسته[12] قرار می‌دهیم. روش‌های گرادیان سیاست با استفاده از تقریب‌زن[13]های تابع شبکه عصبی[14] به ویژه در کنترل پیوسته موفق بوده‌اند [۱]، [۷] و [۸] در محیط‌های گسسته با روش‌های مبتنی بر ارزش[15] رقابت می‌کنند. توجه داریم که تنوع معیارها و عدم وجود تست‌های معناداری در ادبیات یادگیری تقویتی پتانسیل گزارش نتایج گمراه‌کننده[16] را ایجاد می‌کند. ما فواید استفاده از تست‌های معنادار را با استفاده از تکنیک‌های رایج در یادگیری ماشین[17] و روش‌های آماری نشان می‌دهیم.

---

[1] Interaction
[2] Agent
[3] Environment
[4] Policy
[5] Reward
[6] Commulative
[7] Task
[8] Hyperparameter
[9] Codebases
[10] Random Seeds
[11] Policy Gradient
[12] Continuous Control
[13] Approximator
[14] Neural Network
[15] Value
[16] Misleading
[17] Machine Learning



شکل۱-۱: رشد مقالات منتشر شده در زمینه یادگیری تقویتی. در این شکل، تعداد مقالات مرتبط با یادگیری تقویتی (محور y) در هر سال (محور x) که از جستجوهای Google Scholar استخراج شده‌اند، نشان داده شده است.

۱-۱ پیشینه تحقیق

تاکنون چندین پژوهش به ارزیابی الگوریتم‌های یادگیری تقویتی پرداخته‌اند. در [۹] چندین الگوریتم یادگیری تقویتی مبنا قرار گرفته و پیاده‌سازی‌های پایه را برای جامعه علمی فراهم می‌کنند. معیارهای ارزیابی قابل تعمیم یاگیری تقویتی در [۱۰] پیشنهاد شده است. در [۱۱] محیط یادگیری Arcade را بازبینی کرده و روش‌های ارزیابی بهتری را در این مبناها پیشنهاد می‌دهند. با این حال، در حالی که موضوع بازتولیدپذیری و تمرین‌های تجربی در زمینه‌های مرتبط مورد بررسی قرار گرفته است؛ تا آنجا که می‌دانیم [۱۲]، [۱۳]، [۱۴]، [۱۵]، [۱۶] و [۱۷] اولین کاری است که این سوال مهم را در زمینه یادگیری تقویتی عمیق بررسی می‌کند.

۱-۲ اهداف و دستاورد های تحقیق

در هر بخش از تحلیل، سوالاتی را در مورد عوامل کلیدی تأثیرگذار بر خاصیت بازتولیدی مطرح می‌کنیم و می‌بینیم که منابع متعدد عدم قطعیت[1] در بازتولید و مقایسه الگوریتم‌های یادگیری تثو وجود دارد. به این منظور، نشان می‌دهیم که جزئیات دقیق روش تجربی می‌تواند بسیار مهم باشد. بر اساس آزمایش‌های انجام شده توسط ما به این نتیجه می‌رسیم که با توصیه‌ها، خطوط تحقیقات و نقاط بحثی برای کارهای آینده، یادگیری تقویتی عمیق قابل بازتولید بوده و همچنان اهمیت دارد.

۱-۳ ساختار گزارش

در فصل دوم، به طور کلی به بررسی مطالعه یادگیری تقویتی و الگوریتم‌های آن می‌پردازیم. در این بخش، به بررسی اصول و مبانی یادگیری و نیز الگوریتم‌های مهم در این حوزه می‌پردازیم.
در فصل سوم، به طور کلی به بررسی عوامل تأثیرگذار بر خاصیت بازتولیدی الگوریتم‌های یادگیری تقویتی می‌پردازیم. در این بخش، عوامل خارجی و داخلی که می‌توانند بر فرایند بازتولید نتایج تأثیرگذار باشند را مورد بررسی قرار می‌دهیم و روش‌های مختلف برای کاهش عدم قطعیت در بازتولید نتایج را بررسی می‌کنیم.

[1] Nondeterminism

فصل دوم
مفاهیم پایه در یادگیری تقویتی

۲-۱ مسئله یادگیری تقویتی

مسئله یادگیری تقویتی به عنوان یک چارچوب ساده برای یادگیری از تعامل به منظور دستیابی به هدف مطرح شده است. در این چارچوب، یادگیرنده و تصمیم‌گیرنده به عنوان عامل شناخته می‌شوند. هر چیزی که عامل با آن تعامل دارد و شامل تمام چیزهایی است که خارج از عامل قرار دارند، محیط نامیده می‌شود. این دو به طور مداوم با یکدیگر تعامل دارند، به این صورت که عامل اقداماتی[1] را انتخاب می‌کند و محیط به این اقدامات پاسخ داده و وضعیت[2]های جدیدی را به عامل ارائه می‌دهد. محیط همچنین پاداش هایی به شکل مقادیر عددی ارائه می‌دهد که عامل سعی در بیشینه‌سازی آن‌ها در طول زمان دارد. یک تعریف کامل از محیط، یک وظیفه را تعریف می‌کند که یک نمونه از مسئله یادگیری تقویتی است.

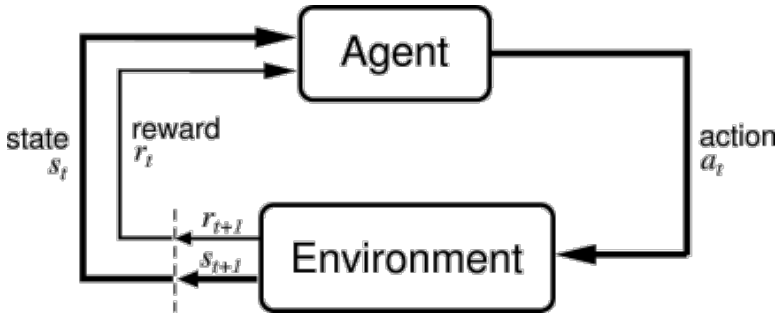
به طور خاص، عامل و محیط در هر دنباله از گام‌های زمانی[3] مجزا، با هم تعامل دارند $(t = 0, 1, 2, 3, \dots)$:

در هر گام زمانی t، عامل یک نمایش از وضعیت محیط دریافت می‌کند، $S_t \in S$ که در آن S مجموعه‌ای از وضعیت‌های ممکن است. سپس بر اساس آن، یک اقدام انتخاب می‌کند، $A_t \in A(S_t)$، که در آن $A(S_t)$ مجموعه‌ای از اقدامات موجود در وضعیت S_t است. درگام زمانی بعد، عامل یک پاداش عددی، به عنوان نتیجه‌ای از اقدام خود دریافت می‌کند، $R_{t+1} \in R \subset \mathbb{R}$ و خود را در وضعیت جدیدی می‌یابد، S_{t+1}.

شکل ۲-۱ تعامل عامل و محیط را نشان می‌دهد. در هر گام زمانی، عامل یک نگاشت[4] از وضعیت‌ها به احتمال انتخاب هر اقدام ممکن، پیاده‌سازی می‌کند. این نگاشت سیاست عامل نامیده می‌شود و به صورت (π_t) نمایش داده می‌شود، که در آن $\pi_t(a|s)$ احتمال این است که $A_t = a$ اگر $S_t = s$ باشد.

---

[1] Actions
[2] State
[3] Timesteps
[4] Mapping



شکل ۲-۱: تعامل عامل-محیط در یادگیری تقویتی

روش‌های یادگیری تقویتی مشخص می کنند که چگونه عامل سیاست خود را به عنوان نتیجه‌ای از تجربه[1]اش تغییر می‌دهد. هدف عامل، به طور خلاصه، بیشینه‌سازی مجموع کل پاداشی است که در طولانی مدت دریافت می کند.

۲-۲ طبقه‌بندی الگوریتم‌های یادگیری تقویتی

الگوریتم‌های یادگیری تقویتی به دو دسته کلی مدل-مبنا[2] و بدون مدل[3] تقسیم می‌شوند. منظور از مدل محیط، هر چیزی است که یک عامل می‌تواند از آن استفاده کند تا پیش‌بینی[4] کند محیط چگونه به اقداماتش پاسخ می‌دهد. مدل، با داشتن یک حالت و یک عمل، پیش‌بینی می کند که حالت بعدی و پاداش بعدی چه خواهد بود. برخی مدل‌ها توصیفی از تمامی حالت‌های ممکن و احتمالات آنها ارائه می‌دهند که به آنها مدل‌های توزیعی[5] می گوییم. مدل‌های دیگر فقط یکی از حالت‌ها را با توجه به احتمالات انتخاب می کنند که به آنها مدل‌های نمونه‌ای[6] می گوییم. مدل‌ها می‌توانند برای تقلید[7] یا شبیه‌سازی[8] تجربه استفاده شوند. با داشتن یک حالت شروع و یک عمل، یک مدل نمونه‌ای یک انتقال[9] ممکن را تولید می کند و یک مدل توزیعی تمامی انتقال‌های ممکن را با توجه به احتمالات آنها تولید می کند. با داشتن یک حالت شروع و یک سیاست، یک مدل نمونه‌ای می‌تواند یک روند[10] کامل تولید کند و یک مدل توزیعی تمامی روندهای ممکن و احتمالات آنها را تولید می کند. در هر دو حالت، می گوییم مدل برای شبیه‌سازی محیط و تولید تجربه شبیه‌سازی شده استفاده می‌شود [۱۸]. در ادامه به توضیح این دو دسته و الگوریتم‌های مرتبط با هر کدام می‌پردازیم.

۲-۲-۱ الگوریتم‌های مدل-مبنا

در روش‌های مدل-مبنا، عامل به یک مدل از محیط دسترسی دارد یا آن را یاد می گیرد. عامل‌ها می‌توانند نتایج پیش‌بینی‌شده را در سیاست یادگرفته‌شده اعمال کنند. در این روش به دلیل پیش‌بینی‌های انجام شده، عواقب[11] احتمالی سنجیده میشود و عامل می‌تواند تصمیمات[12] بهتری بگیرد، اما یادگیری مدل محیط چندان آسان نیست و پیچیدگی‌هایی را ایجاد می کند.

---

[1] Experience
[2] Model-Based
[3] Model-Free
[4] Predict
[5] Distribution Models
[6] Sample Model
[7] Imitation
[8] Simulation
[9] Transition
[10] Episode
[11] Consequences
[12] Decisions

۲-۲-۲ الگوریتم های بدون مدل

در روش‌های بدون مدل، عامل به مدل محیط دسترسی ندارد و به جای آن، مستقیماً از تعامل با محیط یادگیری می‌کند. در این روش‌ها، دو رویکرد اصلی برای آموزش[1] عامل وجود دارد:

بهینه‌سازی[2] سیاست: روش‌های بهینه‌سازی سیاست، سیاست را به صورت واضحی به شکل $\pi_\theta(a|s)$ نمایش می‌دهند. آن‌ها پارامترهای θ را به دو روش بهینه‌سازی می‌کنند: یا به طور مستقیم با استفاده از گرادیان افزایشی[3] بر روی هدف عملکرد[4] $J(\pi_\theta)$، یا به طور غیر مستقیم با بیشینه‌سازی[5] تقریب‌های محلی $J(\pi_\theta)$. در اینجا، $J(\pi_\theta)$ نشان‌دهنده تابع هدف عملکرد است که میزان عملکرد سیاست π_θ را ارزیابی می‌کند و نشان می‌دهد که این سیاست چقدر خوب عمل می‌کند. این بهینه‌سازی تقریباً همیشه به صورت بر-سیاست[6] انجام می‌شود، به این معنا که هر به‌روزرسانی[7] فقط از داده‌هایی استفاده می‌کند که در حین اجرای نسخه جدید سیاست جمع‌آوری شده‌اند. بهینه‌سازی سیاست معمولاً شامل یادگیری یک تقریب‌زن برای تابع ارزش[8] روی سیاست $V_\phi(s)$ نیز می‌شود، که در تعیین نحوه به‌روزرسانی سیاست مورد استفاده قرار می‌گیرد.

- **الگوریتم PPO[9] از طریق به‌روزرسانی‌هایی که به طور غیرمستقیم عملکرد را به حداکثر می‌رساند عمل می‌کند. این الگوریتم به جای اینکه مستقیماً هدف عملکرد $J(\pi_\theta)$ را بهینه‌سازی کند، یک تابع هدف جایگزین را به حداکثر می‌رساند که یک تخمین محافظه‌کارانه[10] از میزان تغییر $J(\pi_\theta)$ در نتیجه به‌روزرسانی‌ها ارائه می‌دهد.**
- **الگوریتم TRPO[11] سیاست را به گونه‌ای به‌روزرسانی می‌کند که تغییرات آن محدود به یک ناحیه اطمینان[12] باشد. این کار باعث می‌شود تغییرات پایدارتر و کنترل‌شده‌تر باشند و از نوسانات شدید جلوگیری شود.**
- **الگوریتم ACKTR[13] به جای اینکه مستقیماً هدف عملکرد $J(\pi_\theta)$ را بهینه‌سازی کند، از نواحی اطمینان برای به‌روزرسانی‌های پایدار و کارآمد سیاست بهره می‌برد. نواحی اطمینان تضمین می‌کنند که تغییرات سیاست به شکلی پایدار و کنترل‌شده انجام شود.**

یادگیری-Q[14]: در این رویکرد، یک تقریب‌زن $Q_\theta(s,a)$ برای تابع ارزش-عمل بهینه $Q^*(s,a)$ یاد گرفته می‌شود. این بهینه‌سازی معمولاً بر اساس معادله‌ی بلمن[15] و اکثراً به صورت بدون- سیاست[16] انجام می‌شود، به این معنا که بدون توجه به اینکه عامل چگونه محیط را جستجو[17] کرده است، هر به‌روزرسانی می‌تواند از داده‌های جمع‌آوری شده در هر زمان از

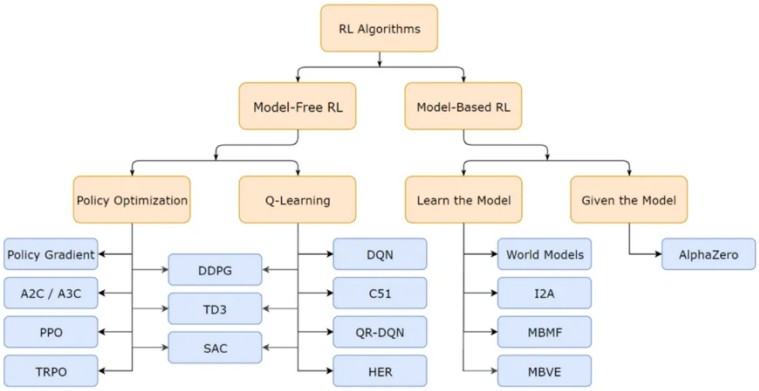
---

[1] Training
[2] Optimization
[3] Gradient Ascent
[4] Performance
[5] Maximizing
[6] On-Policy
[7] Update
[8] Value Function
[9] Proximal Policy Optimization
[10] Conservative
[11] Trust Region Policy Optimization
[12] Trust Region
[13] Actor-Critic using Kronecker-Factored Trust Region
[14] Q-Learning
[15] Bellman Equation
[16] Off-Policy
[17] Search

آموزش استفاده کند. سیاست مربوطه از طریق ارتباط بین Q^* و π^* به دست می‌آید و اقداماتی که توسط عامل یادگیری Q انجام می‌شود به صورت زیر است:

$$(s) = arg\,\max_a Q_\theta\,(s,a) \quad (۱-۲)$$

میان‌یابی[1] بین بهینه‌سازی سیاست و یادگیری-Q: به طرز عجیبی، بهینه‌سازی سیاست و یادگیری-Q با یکدیگر ناسازگار نیستند و تحت برخی شرایط حتی می‌توانند معادل باشند. الگوریتم‌هایی وجود دارند که در این طیف قرار می‌گیرند و می‌توانند به دقت بین نقاط قوت و ضعف هر دو روش تعادل برقرار کنند. این الگوریتم‌ها به بهره‌گیری از ویژگی‌های خوب هر دو روش کمک می‌کنند و می‌توانند در شرایط مختلف عملکرد بهتری داشته باشند. مثالی از این الگوریتم ها DDPG[2] است. DDPG یک الگوریتم است که به طور همزمان یک سیاست تعیین‌کننده[3] و یک تابع Q[4] را می‌آموزد. این الگوریتم از هر کدام برای بهبود دیگری استفاده می‌کند. به عبارت دیگر، سیاست تعیین‌کننده برای بهبود تابع Q و تابع Q برای بهبود سیاست تعیین‌کننده به کار می‌رود. این تعامل به الگوریتم اجازه می‌دهد تا به طور موثرتری بهینه‌سازی را انجام دهد و عملکرد بهتری در محیط‌های پیچیده داشته باشد. شکل ۲-۲ تصویر مناسبی از این دسته بندی‌ها را نشان می‌دهد [۱۹].



شکل ۲-۲: دسته بندی در حوزه یادگیری تقویتی

[1] Interpolating
[2] Deep Deterministic Policy Gradient
[3] Deterministic Policy
[4] Q-Function

فصل سوم
عوامل تأثیرگذار بر خاصیت بازتولیدی الگوریتم‌های یادگیری تقویتی

۳-۱ پیش‌زمینه فنی

این تحقیق به چندین الگوریتم یادگیری تقویتی بدون مدل مبتنی بر گرادیان سیاست پرداخته که پیاده‌سازی‌های آن‌ها به‌صورت عمومی در دسترس است و به‌طور گسترده در مقالات علمی به عنوان معیار مقایسه با روش‌های جدید به کار می‌روند. ما الگوریتم‌های بهینه‌سازی سیاست ناحیه اطمینان[1] [۸]،گرادیان‌های سیاست قطعی عمیق[2] [۱]، بهینه‌سازی سیاست پروگزیمال[3] [۷]، و بازیگر-منتقد با استفاده از ناحیه اطمینان فاکتوری کرونکر[4] [۲۰] را مورد بررسی قرار دادیم. این الگوریتم‌ها نتایج خوبی در وظایف کنترل پیوسته در محیط MuJoCo [۲۱] از OpenAI Gym [۲۲] نشان داده‌اند.

این الگوریتم‌ها عموماً به دنبال بهینه‌سازی پاداش تخفیفی مورد انتظار،

$$\rho(\theta, s_0) = \mathbb{E}_{\pi\theta}[\sum_{t=0}^{\infty} \gamma^t r(s_t) \,|s_0] \quad (۱-۳)$$

با استفاده از تئوری گرادیان سیاست هستند که بیان می‌کند:

$$\frac{\delta\rho(\theta, s_0)}{\delta\theta} = \sum_s \mu_{\pi\theta}(s|s_0) \sum_a \frac{\delta_{\pi\theta}(a|s)}{\delta\theta} Q_{\pi\theta}(s|a) \quad (۲-۳)$$

اینجا

$$\mu_{\pi\theta}(s|s_0) = \sum_{t=0}^{\infty} \gamma^t P(s_t = s|s_0) \quad (۳-۳)$$

[۲۳]

[۸] TRPOو [۷] PPO از محدودیت‌ها[5] و تخمین مزیت[6] برای انجام این به‌روزرسانی استفاده می‌کنند و مسئله بهینه‌سازی را به صورت زیر بازنویسی می‌کنند:

[1] TRPO
[2] DDPG
[3] PPO
[4] ACKTR
[5] Constraints
[6] Advantage

$$\max_{\theta} \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_t(s_t, a_t) \right]$$

(۴-۳)

اینجا (A_t) تابع مزیت تعمیم‌یافته [۲۴] است TRPO .از روش نزول گرادیان مزدوج با یک محدودیت KL استفاده می‌کند:

$$\mathbb{E}_t \left[\boldsymbol{KL}\left[\pi_{\theta_{old}}(.|s_t), \pi_\theta(., s_t)\right] \right] \leq \delta$$

(۵-۳)

PPO محدودیت را به عنوان یک جریمه (یا هدف برش) بازنویسی می‌کند.

الگوریتم‌های DDPG و ACKTR از روش بازیگر-منتقد[1] استفاده می‌کنند. در این روش، ابتدا تابع ارزش[2] که نشان‌دهنده مطلوبیت یک حرکت در یک وضعیت خاص است، تخمین زده می‌شود. سپس یک سیاست بهینه می‌شود که این تابع ارزش را بر اساس عرضه مونت‌کارلو[3] رول‌اوت بیشینه کند .

تفاوت اصلی DDPG و ACKTR در نوع سیاست‌هایی است که استفاده می‌کنند. DDPG از سیاست‌های قطعی[4] استفاده می‌کند، یعنی برای هر وضعیت، یک حرکت خاص را انتخاب می‌کند. اما ACKTR از سیاست‌های تصادفی[5] بهره می‌برد که در آن‌ها برای هر وضعیت، یک توزیع احتمال[6] بر روی حرکات وجود دارد. برای اطمینان از پایداری[7] در سیاست‌های تصادفی، ACKTR از روش مناطق اعتماد فاکتور کرونکر[8] استفاده می‌کند. این روش باعث می‌شود تغییرات در سیاست، محدود و کنترل‌شده باشند تا از ناپایداری‌ها جلوگیری شود.

۳-۲ تحلیل آزمایشی

در این بخش، به بررسی عواملی می‌پردازیم که بر قابلیت بازتولید روش‌های پیشرفته یادگیری تقویتی تأثیرگذار هستند. برای این منظور، مجموعه‌ای از آزمایش‌ها طراحی و اجرا شد تا اثرات این عوامل بررسی شوند. موارد زیر مورد توجه قرار گرفتند:

۱. **تنظیم ابرپارامترها: بررسی تأثیر تنظیمات نامناسب ابرپارامترها بر عملکرد الگوریتم‌ها.**
۲. **دانه‌های تصادفی و تعداد تکرارها: ارزیابی اثر دانه‌های تصادفی و تعداد دفعات تکرار آزمایش بر نتایج.**
۳. **ویژگی‌های محیط: بررسی چگونگی تأثیرگذاری ویژگی‌های خاص محیط بر عملکرد الگوریتم‌ها.**
۴. **معماری شبکه: ارزیابی تاثیر انتخاب معماری شبکه برای تقریب تابع ارزش و سیاست بر عملکرد.**
۵. **کدهای پایه: بررسی تفاوت‌های عملکرد الگوریتم‌ها به دلیل تفاوت در کدهای پایه، با ثابت نگه داشتن سایر عوامل.**

برای اکثر آزمایش‌ها، از پیاده‌سازی‌های موجود در OpenAI Baselines برای الگوریتم‌های ACKTR [۷]، PPO [۲۰]، DDPG [۲۵]، TRPO [۷] استفاده شد. محیط‌های مورد استفاده شامل Hopper-v۱ و HalfCheetah-v۱ از مجموعه MuJoCo [۲۱]

[1] Actor-Critic
[2] Q Function
[3] Monte Carlo Rollout
[4] Deterministic
[5] Stochastic
[6] Probability Distribution
[7] Stability
[8] Kronecker Factored Trust Regions

در OpenAI Gym [۲۲] بودند که دینامیک[1]های متفاوتی دارند و به بررسی اثرات مختلف کمک می‌کنند. برای اطمینان از نتایج منصفانه، هر آزمایش پنج بار با دانه‌های تصادفی متفاوت اجرا شده است (همه آزمایش‌ها از مجموعه یکسانی از دانه‌های تصادفی استفاده کردند). در این گزارش، جز در مواردی که به طور مشخص ذکر شده، از تنظیمات پیش‌فرض استفاده شده و تنها ابرپارامترهای مورد بررسی تغییر یافته‌اند. در همه آزمایش‌ها از تقریب‌زن‌های پرسپترون چندلایه[2] استفاده شد. اندازه لایه‌های مخفی[3] و توابع فعال‌سازی[4] به صورت (N، M، فعال‌سازی) نمایش داده شده‌اند. برای تنظیمات پیش‌فرض، ابرپارامترهای تحت بررسی به نوبت تغییر یافتند. برای DDPG از ساختار شبکه (۶۴، ۶۴، ReLU) برای هر دو بخش عامل و منتقد، برای TRPO و PPO از ساختار (۶۴، ۶۴، tanh) برای سیاست، و برای ACKTR از ساختار (۶۴، ۶۴، tanh) برای عامل و (۶۴، ۶۴، ELU) برای منتقد استفاده شده است.

۳-۲-۱ ابرپارامترها

انتخاب دقیق ابرپارامترها می‌تواند نقش بسیار مهمی در بهینه‌سازی نتایج الگوریتم‌ها داشته باشد. با این حال، در بسیاری از مقالات مرتبط، انتخاب تنظیمات بهینه ابرپارامترها یکسان نیست و محدوده مقادیر مورد بررسی به ندرت گزارش می‌شود. علاوه بر این، انتخاب نادرست ابرپارامترها می‌تواند به مقایسه ناعادلانه‌ای با الگوریتم‌های پایه منجر شود. در اینجا، به بررسی جنبه‌های مختلف انتخاب ابرپارامتر و تأثیر آن‌ها بر عملکرد الگوریتم‌ها می‌پردازیم.

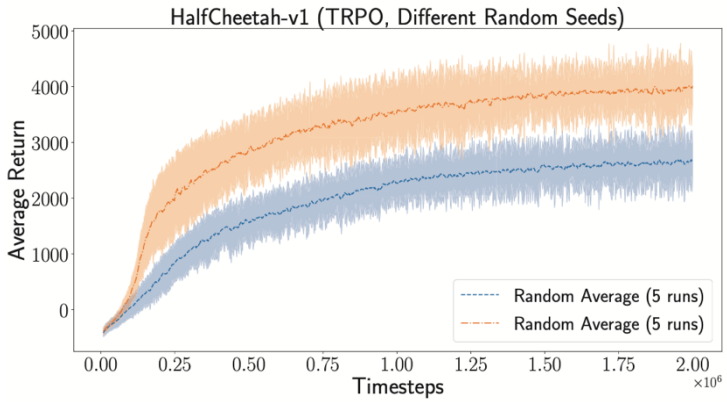
۳-۲-۲ دانه‌های تصادفی و تعداد آزمایش‌ها

یکی از دغدغه‌های اصلی در یادگیری تقویتی عمیق، تفاوت نتایج به خاطر تصادفی بودن محیط یا فرآیند یادگیری مثل مقداردهی اولیه تصادفی وزن‌هاست [۲۰] ؛ [۲۶] بنابراین، حتی اگر نتایج چند آزمایش با دانه‌های کاملاً متفاوت را میانگین بگیریم، ممکن است به نتایج گمراه‌کننده‌ای برسیم. ما این موضوع را از طریق یک آزمایش بررسی می‌کنیم.

نتایج: ما ۱۰ آزمایش را با همان تنظیمات ابرپارامتر انجام دادیم، فقط دانه‌های تصادفی را در هر ۱۰ آزمایش تغییر دادیم. سپس آزمایش‌ها را به دو گروه ۵ تایی تقسیم کرده و میانگین هر گروه را محاسبه کردیم. همانطور که در شکل ۳-۱ می‌بینید، عملکرد الگوریتم‌ها می‌تواند بسیار متفاوت باشد. نشان می‌دهیم که تفاوت بین اجراها به قدری زیاد است که فقط با تغییر دانه‌های تصادفی، توزیع‌های آماری کاملاً متفاوتی به دست می‌آید. متاسفانه در نتایج گزارش شده اخیر، گاهی فقط بهترین N آزمایش از میان چندین آزمایش انتخاب شده [۲۰]؛ [۲۶] یا نتایج روی تعداد کمی آزمایش (N<۵) میانگین گرفته شده است [۲۷]؛ [۲۰] آزمایش ما نشان می‌دهد که این کار می‌تواند گمراه‌کننده باشد. به خصوص برایHalfCheetah ، امکان دارد منحنی‌های یادگیری به دست آمده اصلاً در یک توزیع قرار نگیرند، فقط به این خاطر که میانگین اجراهای مختلف با همان ابرپارامترها اما دانه‌های متفاوت گرفته شده است. اگرچه نمی‌توان تعداد دقیقی از آزمایش‌ها را توصیه کرد، شاید بتوان از روش‌های تحلیل توان[5] برای پیشنهاد یک تعداد تقریبی استفاده کرد، اما نیاز به بررسی بیشتری برای حل این مشکل است.

---

[1] Dynamic
[2] Multi Layer Perceptron
[3] Hidden Layers
[4] Activation Functions
[5] Power Analysis



شکل ۳-۱: اجرای TRPO بر روی HalfCheetah-v۱ با استفاده از همان تنظیمات ابرپارامترها به طور میانگین بر روی دو مجموعه از ۵ بذر تصادفی مختلف. میانگین آزمون t دو نمونه‌ای در کل توزیع آموزشی به نتیجه t = -۹.۰۹۱۶ و p = ۰.۰۰۱۶ رسید.

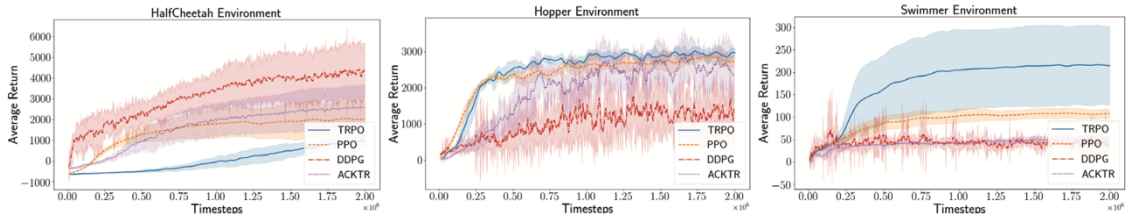
۳-۲-۳ ویژگی های محیط

برای ارزیابی اینکه انتخاب محیط چگونه می‌تواند بر نتایج تاثیر بگذارد، از مجموعه‌ای از ابرپارامترهای پیش‌فرض در الگوریتم‌های مختلف استفاده کردیم و بررسی کردیم که هر الگوریتم در مجموعه‌ای از وظایف کنترل پیوسته چقدر خوب عمل می‌کند. برای این آزمایش‌ها، از محیط‌های روبرو از OpenAI Gym استفاده کردیم : Hopper-v۱، HalfCheetah-v۱، Swimmer-v۱ و Walker۲d-v۱. **انتخاب محیط نقش مهمی در نشان دادن عملکرد یک الگوریتم جدید نسبت به الگوریتم‌های موجود دارد. در وظایف کنترل پیوسته، محیط‌ها معمولاً دارای رفتارهای تصادفی، مسیر[1]های کوتاه‌تر یا ویژگی‌های دینامیکی متفاوت هستند. ما نشان می‌دهیم که به دلیل این تفاوت‌ها، عملکرد الگوریتم‌ها می‌تواند در محیط‌های مختلف متفاوت باشد و بهترین الگوریتم برای همه محیط‌ها همیشه مشخص نیست. بنابراین، ارائه نتایج برای طیف گسترده‌ای از محیط‌ها اهمیت دارد و نباید فقط محیط‌هایی را انتخاب کنیم که کار جدید ما نسبت به محیط‌های دیگر بهتر عمل کند.**

نتایج: همانطور که در شکل ۴ نشان داده شده است، در محیط‌هایی با دینامیک پایدار مثل HalfCheetah-v۱، **الگوریتم** DDPG **عملکرد بهتری نسبت به بقیه دارد. اما با ناپایدارتر شدن دینامیک مثل در** Hopper-v۱ **عملکرد** DDPG **به شدت کاهش می‌یابد. چون** DDPG **یک روش بدون سیاست است، نویز ناشی از اکتشاف[2] می‌تواند باعث شکست[3]های ناگهانی در محیط‌های ناپایدار شود. بنابراین، یادگیری یک تخمین درست از مقدار** Q **برای بازده[4]های مورد انتظار دشوار است، خصوصاً وقتی که بسیاری از مسیرهای اکتشافی منجر به شکست می‌شوند. چون شکست‌ها در چنین وظایفی با مسیرهای کوتاه‌تر مشخص می‌شوند، یک بهینه محلی[5] در این حالت فقط زنده ماندن تا حداکثر طول مسیر (که معادل هزار گام زمانی و پاداش مشابه به خاطر پاداش بقا در حالت** Hopper-v۱ **است) می‌باشد. همانطور که در شکل ۳-۲ مشاهده می‌شود،** DDPG **در** Hopper **دقیقاً همین کار را انجام می‌دهد. این یک مثال واضح است که نشان می‌دهد تنها نشان دادن** HalfCheetah **پایدار و مناسب هنگام گزارش آزمایش‌های مبتنی بر** DDPG **ناعادلانه است. علاوه بر این، محیط** Swimmer-v۱ **را در نظر بگیریم که در شکل ۳-۲ نشان داده شده است. در اینجا،** TRPO **به طور قابل توجهی عملکرد بهتری نسبت به سایر الگوریتم‌ها دارد. به دلیل دینامیک**

[1] Trajectory
[2] Exploration
[3] Failure
[4] Return
[5] Local Optimal

محیط شبیه به آب، یک بهینه محلی برای سیستم این است که جمع شود و بدون شنا کردن صحیح تکان بخورد. اما این وضعیت منجر به بازده حدود ۱۳۰ می‌شود. با رسیدن به یک بهینه محلی، نمودارهای یادگیری می‌توانند بهینه‌سازی موفقیت‌آمیز سیاست را در طول زمان نشان دهند، در حالی که در واقع بازده‌های به دست آمده نمایانگر رفتار مطلوب نیستند، همانطور که در ویدئویی[1] از سیاست یادگرفته شده نشان داده شده است. بنابراین، مهم است که نه تنها بازده‌ها، بلکه نمایش‌هایی از سیاست یادگرفته شده در عمل نشان داده شود. بدون درک اینکه ارزیابی بازده‌ها چه چیزی را نشان می‌دهد، ممکن است نتایج گمراه‌کننده‌ای گزارش شود که در واقع تنها بهینه‌های محلی را بیشینه می‌کند، نه رسیدن به رفتار مطلوب.



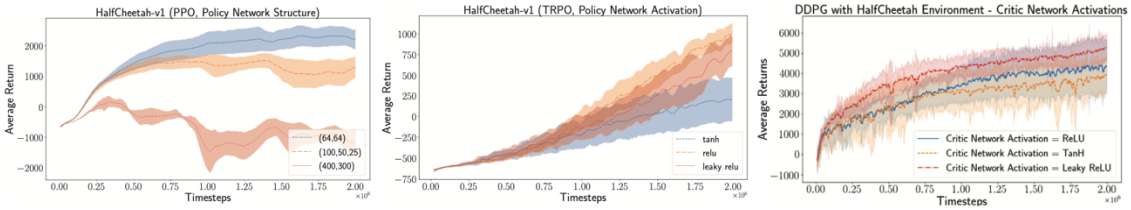
شکل ۳-۲: عملکرد چندین الگوریتم گرادیان سیاست در مجموعه محیط‌های معیار MuJoCo

۳-۲-۴ معماری شبکه

مطالعات قبلی نشان داده‌اند که معماری شبکه[2] سیاست می‌تواند نتایج را در الگوریتم‌های TRPO و DDPG به طور قابل توجهی تحت تأثیر قرار دهد [۶]. علاوه بر این، برخی تابع‌های فعال‌ساز مانند واحد خطی تصحیح شده[3] ممکن است به دلیل مشکل "رلو مرده[4]" باعث کاهش عملکرد یادگیری شوند. [۲۸] بنابراین، ما معماری شبکه و توابع فعال‌ساز را برای هر دو تقریب‌زن سیاست و ارزش بررسی کردیم. در ادبیات موضوع، بررسی‌های مشابهی نشان داده‌اند که عملکرد تقریب‌زن‌های خطی، RBF و شبکه‌های عصبی متفاوت است. [۲۹] جداول ۱ و ۲ خلاصه‌ای از عملکرد نهایی ارزیابی تمام تغییرات معماری پس از آموزش روی ۲ میلیون نمونه (یعنی ۲ میلیون گام زمانی در محیط) را نشان می‌دهند. ما سه معماری متداول پرسپترون چندلایه (MLP) را بررسی کردیم: (۶۴، ۶۴)، (۱۰۰، ۵۰، ۲۵) و (۴۰۰، ۳۰۰). علاوه بر این، توابع فعال‌ساز شبکه‌های ارزش و سیاست را در میان تانژانت هیپربولیک[5]، ReLU و Leaky ReLU تغییر دادیم. نتایج نشان می‌دهد که عملکرد می‌تواند با تغییرات ساده در توابع فعال‌ساز شبکه‌های سیاست یا ارزش به شدت تحت تأثیر قرار گیرد. معمولاً ReLU یا Leaky ReLU در میان محیط‌ها و الگوریتم‌ها بهترین عملکرد را دارند. این اثرات در الگوریتم‌ها یا محیط‌های مختلف یکسان نیستند. این ناهمگونی نشان می‌دهد که معماری شبکه چقدر به روش الگوریتم وابسته است. به عنوان مثال، استفاده از یک شبکه بزرگ با PPO ممکن است نیاز به تنظیم ابرپارامترهای دیگری مانند میزان محدودیت ناحیه اطمینان یا نرخ یادگیری[6] داشته باشد تا با تغییر معماری جبران شود. این بازی پیچیده ابرپارامترها یکی از دلایل دشواری تکرارپذیری روش‌های گرادیان سیاست فعلی است. انتخاب معماری مناسب برای نتایج پایه‌ای صحیح بسیار مهم است. این همچنین نیاز احتمالی به الگوریتم‌های بی‌توجه به ابرپارامتر را پیشنهاد

[1] Video Link
[2] Network Architecture
[3] ReLU
[4] Dying Relu
[5] Hypernolic Tangent (tanh)
[6] Learning Rate

می‌کند – یعنی الگوریتم‌هایی که انطباق ابرپارامتر را بخشی از طراحی در نظر می‌گیرند – تا بتوان مقایسه‌های منصفانه‌ای بدون نگرانی از تنظیمات نامناسب برای کار در دست انجام داد.



شکل ۳-۳: اهمیت معماری شبکه سیاست و توابع فعال‌سازی: PPO (چپ)، TRPO (وسط) و DDPG (راست).

Algorithm	**Environment**	**400,300**	**64,64**	**100,50,25**	**tanh**	**ReLU**	**LeakyReLU**
TRPO [8]	Hopper-v1 HalfCheetah-v1	2980 ± 35 1791 ± 224	2674 ± 227 1939 ± 140	3110 ± 78 2151 ± 27	2674 ± 227 1939 ± 140	2772 ± 211 3041 ± 161	- -
TRPO [9]	Hopper-v1 HalfCheetah-v 1	1243 ± 55 738 ± 240	1303 ± 89 834 ± 317	1243 ± 55 850 ± 378	1303 ± 89 834 ± 317	1131 ± 65 784 ± 352	1341± 127 1139 ± 364
TRPO [7]	Hopper-v1 HalfCheetah-v 1	2909 ± 87 -155 ± 188	2828 ± 70 205 ± 256	2812 ± 88 306 ± 261	2828 ± 70 205 ± 256	2941 ± 91 1045 ± 114	2865 ± 189 778 ± 177
PPO [7]	Hopper-v1 HalfCheetah-v1	61 ± 33 -1180 ± 444	2790 ± 62 2201 ± 323	2592 ± 196 1314 ± 340	2790 ± 62 2201 ± 323	2695 ± 86 2971 ± 364	2587 ± 53 2895 ± 365
DDPG [25]	Hopper-v1 HalfCheetah-v1	1419 ± 313 5579 ± 354	1632 ± 459 4198 ± 606	2142 ± 436 5600 ± 601	1491 ± 205 5325 ± 281	1632 ± 459 4198 ± 606	1384 ± 285 4094 ± 233
DDPG [27]	Hopper-v1 HalfCheetah-v1	600 ± 126 2845 ± 589	593 ± 155 2771 ± 535	501 ± 129 1638 ± 624	436 ± 48 1638 ± 624	593 ± 155 2771 ± 535	319 ± 127 1405 ± 511
DDPG [9]	Hopper-v1 HalfCheetah-v1	506 ± 208 850 ± 41	749 ± 271 1573 ± 385	629 ± 138 1224 ± 553	354 ± 91 1311 ± 271	749 ± 271 1573 ± 385	- -
ACKTR [20]	Hopper-v1 HalfCheetah-v1	2577 ± 529 2653 ± 408	1608 ± 66 2691 ± 231	2287 ± 946 2498 ± 112	1608 ± 66 2621 ± 381	2835 ± 503 2160 ± 151	2718 ± 434 2690 ± 231

جدول۳-۱: نتایج تغییرات معماری سیاست ما در میان پیاده سازی ها و الگوریتم های مختلف. میانگین نهایی ± خطای استاندارد در عرض ۵ آزمایش از بازده ها در طول ۱۰۰ مسیر آخر پس از ۲ میلیون نمونه آموزشی. برای ACKTR، از تابع فعال ساز ELU استفاده می کنیم.

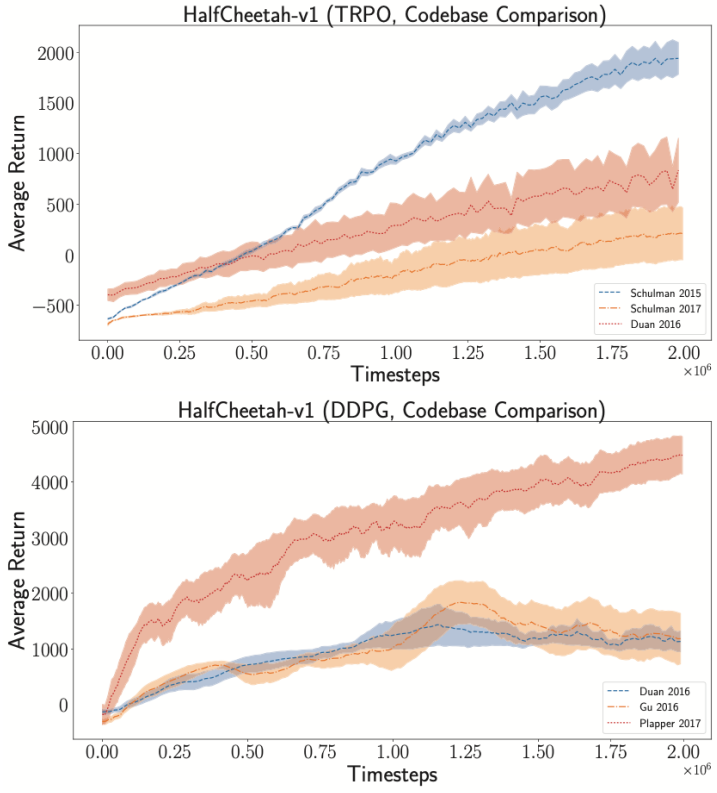
Algorithm	Environment	400,300	64,64	100,50,25	tanh	ReLU	LeakyReLU
TRPO [8]	Hopper-v 1 HalfCheetah-v1	3011 ± 171 2355 ± 48	2674 ± 227 1939 ± 140	2782 ±120 1673 ± 148	2674 ± 227 1939 ± 140	3104 ± 84 2281 ± 91	- -
TRPO [7]	Hopper-v1l HalfCheetah-v 1	2909 ± 87 178 ± 242	2828 ± 70 205 ± 256	2812 ± 88 172 ± 257	2828 ± 70 205 ± 256	2829 ± 76 235 ± 260	3047 ± 68 325 ± 208
PPO [7]	Hopper-v1l HalfCheetah-v1	2704 ± 37 1523 ± 297	2790 ± 62 2201 ± 323	2969 ± 111 1807 ± 309	2790 ± 62 2201 ±323	2687 ± 144 1288 ± 12	2748 ± 77 1227 ± 462
DDPG [25]	Hopper-v1 HalfCheetah-v 1	1419 ± 312 5600 ± 601	1632 ± 458 4197± 606	1569 ± 453 47134374	971 ± 137 3908± 293	852 ± 143 4197 ± 606	843 ± 160 5324 ± 280
DDPG [27]	Hopper-v1 HalfCheetah-v 1	523 ± 248 1373 ± 678	343 ± 34 1717 ± 508	345 ± 44 1868 ± 620	436 ± 48 1128 ±511	343 ± 34 1717 ± 508	- -
DDPG [9]	Hopper-v1 HalfCheetah-v 1	1208 ± 423 789 ± 91	394 ± 144 1095 ± 139	380 ± 65 988 ± 52	354± 91 1311 ± 271	394 ± 144 1095 ± 139	- -
ACKTR [20]	Hopper-v1 HalfCheetah-v1	152 ± 47 2653 ± 408	1930 ± 185 2691 ± 231	1589 ± 225 2498 ± 112	691 ±55 2621 ± 381	500 ± 379 2160 ± 151	1930 ± 185 2691 ± 231

جدول ۳-۲: نتایج تغییرات معماری تابع ارزش (Q یا V) در میان پیاده سازی ها و الگوریتم های مختلف. میانگین نهایی ± خطای استاندارد در عرض ۵ آزمایش از بازده ها در طول ۱۰۰ مسیر آخر پس از ۲ میلیون نمونه آموزشی. برای ACKTR، به جای Leaky ReLU، از توابع فعال ساز ELU استفاده می کنیم.

۳-۲-۵ کدهای پایه

در بسیاری از موارد، نویسندگان نسخه‌های خود را از الگوریتم‌های مبنا برای مقایسه پیاده‌سازی می‌کنند. ما پیاده‌سازی مبناهای OpenAI از الگوریتم TRPO که در (Schulman et al. ۲۰۱۷) استفاده شده، کد اصلی [۸] TRPO ، و پیاده‌سازی TRPOبا Tensorflow در [۹] را بررسی می‌کنیم. همچنین پیاده‌سازی‌های[۹] rllab Theano ، [۲۷] rllabplusplus و OpenAI baselines [۲۵]از DDPG را مقایسه می‌کنیم. هدف ما توجه به تفاوت‌های عملکرد به دلیل جزئیات پیاده‌سازی در الگوریتم‌ها است. ما زیرمجموعه‌ای از آزمایش‌های معماری خود را با استفاده از پیاده‌سازی‌های OpenAI baselines و همان ابرپارامترها انجام می‌دهیم.

نتایج: ما متوجه شدیم که تفاوت‌های پیاده‌سازی که اغلب در مقالات منعکس نمی‌شوند، می‌توانند تأثیرات چشمگیری بر عملکرد داشته باشند. این را می‌توان در عملکرد نهایی ارزیابی ما پس از آموزش روی ۲ میلیون نمونه در جداول ۱ و ۲، و همچنین یک مقایسه نمونه در شکل ۳-۴ مشاهده کرد. این نشان می‌دهد که جزئیات پیاده‌سازی باید شرح داده شوند، کدها همراه با مقالات منتشر شوند، و عملکرد آزمایش‌های مبنا در کارهای جدید با کدهای اصلی مقالات مبنا مطابقت داشته باشد.



شکل ۳-۴: مقایسه کد پایه TRPO با استفاده از مجموعه پیش‌فرض ابرپارامترهای ما (همانطور که در آزمایش‌های دیگر استفاده شده است).

۳-۳ گزارش معیارهای ارزیابی

در این بخش، برخی از معیارهای ارزیابی که معمولاً در مطالعات یادگیری تقویتی استفاده می‌شوند را بررسی می‌کنیم. در عمل، الگوریتم‌های یادگیری تقویتی معمولاً با نمایش نمودارها یا جداولی از پاداش تجمعی متوسط (بازده متوسط) و اخیراً، پاداش حداکثر به دست آمده در تعداد معینی از گام‌های زمانی ارزیابی می‌شوند. به دلیل ناپایدار بودن بسیاری از این الگوریتم‌ها، صرفاً گزارش پاداش‌های حداکثری برای مقایسه منصفانه کافی نیست؛ حتی گزارش بازده متوسط نیز می‌تواند

گمراه‌کننده باشد زیرا دامنه عملکرد در بین آزمایش‌ها و دانه‌های تصادفی متفاوت است. این گزارش‌ها ممکن است به تنهایی، تصویر واضحی از عملکرد یک الگوریتم ارائه ندهند. با این حال، هنگامی که با فواصل اطمینان[1] ترکیب شوند، ممکن است برای گرفتن تصمیمی آگاهانه[2] کافی باشند به شرطی که تعداد آزمایش‌ها کافی باشد. از این رو، ما استفاده از روش‌های بوت‌استرپ[3] و آزمون معناداری مشابه یادگیری ماشین (مانند مطالعات [۳۰] و [۳۱]) را برای ارزیابی عملکرد الگوریتم‌ها بررسی می‌کنیم.

۳-۳-۱ تفاوت بین مشاهده برخط[4] و بهینه‌سازی سیاست

یک تفاوت مهم در گزارش نتایج یادگیری تقویتی مشاهده برخط و بهینه‌سازی سیاست است. در مشاهده برخط، عامل در طول فرآیند یادگیری بهینه‌سازی بازده‌ها را هدف قرار می‌دهد و مسیر عامل پایانی ندارد. در این حالت، ارزیابی‌ها می‌توانند شامل پاداش‌های تجمعی متوسط در طول کل فرآیند یادگیری باشند. این روش به تعادل بین اکتشاف و بهره‌برداری[5] توجه دارد و معمولاً از پاداش‌های تجمعی متوسط در طول زمان برای ارزیابی استفاده می‌شود، همانطور که در مطالعات [۳۲] آمده است. همچنین، ممکن است از روش‌های ارزیابی برون‌خط[6] نیز استفاده شود، همانطور که در مطالعه[۳۳] دیده می‌شود.
در مقابل، بهینه‌سازی سیاست رویکرد دیگری است که در آن ارزیابی با استفاده از یک سیاست هدف در حالت برون‌خط انجام می‌شود. در این دیدگاه، ارزیابی‌ها باید در طول کل مسیر وظیفه با یک سیاست هدف واحد انجام شوند تا بازده‌های متوسطی که سیاست هدف می‌تواند به دست آورد، مشخص شود. این روش به ما اجازه می‌دهد تا عملکرد یک سیاست مشخص را در شرایطی ثابت و کنترل‌شده ارزیابی کنیم.

۳-۳-۲ فواصل اطمینان

روش بوت‌استرپ به عنوان یک روش محبوب برای به دست آوردن بینشی از توزیع یک جمعیت[7] با استفاده از یک نمونه[8] کوچکتر شناخته می‌شود. این تکنیک به خصوص در آزمایش‌های A/B بسیار پرکاربرد است و می‌توانیم از ایده‌های این حوزه بهره‌بگیریم. معمولاً یک تخمین‌گر[9] بوت‌استرپ با بازنمونه‌گیری[10] مکرر با جایگزینی ایجاد می‌شود تا میانگین آماری معتبر و حدود اطمینان به دست آید. با استفاده از این روش، می‌توانیم به بازه اطمینان[11] ٪۹۵ نتایج بخش محیط‌های خود دست پیدا کنیم.
جدول ۳ نشان‌دهنده میانگین بوت‌استرپ و حدود اطمینان ٪۹۵ در آزمایش‌های محیطی ما است. حدود اطمینان می‌تواند بسته به الگوریتم‌ها و محیط‌ها بسیار متغیر باشند. ما مشاهده کردیم که الگوریتم‌های TRPO و PPO از پایدارترین‌ها هستند و حدود اطمینان کوچکی از بوت‌استرپ دارند. در مواردی که حدود اطمینان بسیار بزرگ است، ممکن است نیاز باشد تعداد آزمایش‌ها را افزایش دهیم (یعنی اندازه نمونه را بیشتر کنیم).

---

[1] Confidence Intervals
[2] Informed
[3] Bootstrap
[4] Online
[5] Exploitation
[6] Offline
[7] Population
[8] Sample
[9] Estimator
[10] Resampling
[11] Confidence Interval

۳-۳-۳ تحلیل توان

یکی دیگر از روش‌های تعیین نیاز به بیشتر شدن حجم نمونه، تحلیل توان[1] بوت‌استرپ است [۳۴]. اگر از نمونه موجود استفاده کنیم و به آن یک افزایش[2] یکنواخت[3] اعمال کنیم (مثلاً با ضریب ۱.۲۵ به طور یکنواخت افزایش دهیم)، می‌توانیم شبیه‌سازی‌های بوت‌استرپ متعددی را اجرا کنیم و مشخص کنیم که چه درصدی از این شبیه‌سازی‌ها منجر به مقادیر آماری معنادار با این افزایش می‌شوند. اگر درصد کمی از مقادیر معنادار باشند، به نمونه بزرگتری نیاز داریم (باید آزمایشات بیشتری انجام شود). ما این کار را در تمامی آزمایش‌های محیطی انجام دادیم و به این نتیجه رسیدیم که در تنظیمات ناپایدارتر، درصد توان بوت‌استرپ به نتایج غیرمعنادار در آزمایش افزایش متمایل است. برعکس، در آزمایش‌های پایدار (مثلاً TRPO در Hopper-v۱) با نمونه‌های کوچک، آزمایش افزایش نشان داد که نیازی به آزمایش‌های بیشتر برای ایجاد مقایسه‌های معنادار نیست.

۳-۳-۴ اهمیت[4]

یک عامل مهم در انتخاب الگوریتم یادگیری تقویتی، اهمیت دستاورد[5]های گزارش شده بر اساس یک معیار مشخص است. چندین پژوهش به استفاده از معیارهای اهمیت برای ارزیابی قابلیت اطمینان معیارهای ارزیابی در یادگیری ماشین پرداخته‌اند. با این حال، تحقیقات کمی در یادگیری تقویتی به ارزیابی اهمیت معیارهای گزارش شده می‌پردازند.

بر اساس نتایج تجربی ما که نشان می‌دهد عملکرد الگوریتم‌ها می‌تواند به شدت با تغییرات تصادفی در دانه‌های تصادفی متفاوت باشد، واضح است که به یک معیار برای ارزیابی اهمیت دستاوردهای عملکرد الگوریتم و اطمینان از معیارهای گزارش شده نیاز است. در حالی که تحقیقات و بررسی‌های بیشتری برای تعیین بهترین معیارها برای ارزیابی الگوریتم‌های یادگیری تقویتی لازم است، ما یک مجموعه اولیه از معیارها را بر اساس نتایج یادگیری ماشین بررسی می‌کنیم.

در یادگیری نظارت شده[6]، معیارهای اهمیتی مانند k-fold t-test ،t-test باز نمونه گیری تصحیح شده[7] و دیگر معیارها برای مقایسه نتایج یادگیری ماشین مورد بحث قرار گرفته‌اند [۳۱] و [۳۵]؛ با این حال، فرضیاتی که مربوط به داده‌های اصلی با معیارهای اصلاح شده است، الزاماً در یادگیری تقویتی قابل اعمال نیستند. برای بررسی آزمون‌های اهمیت اصلاح شده مناسب برای یادگیری تقویتی به تحقیقات بیشتری نیاز است.

با این وجود، ما چندین معیار اهمیت را بررسی می‌کنیم که نشان می‌دهند آیا یک الگوریتم جدید واقعاً به اندازه الگوریتم‌های پیشرفته عمل می‌کند یا خیر. ما t-test ۲ نمونه‌ای ساده (مرتب‌سازی تمام بازده‌های نهایی ارزیابی در میان N آزمایش تصادفی با دانه‌های تصادفی مختلف)، آزمون کولموگروف-اسمیرنوف[8] [۳۶] و تفاوت درصدی بوت‌استرپ با فواصل اطمینان ۹۵٪ را در نظر می‌گیریم. تمامی معیارهای محاسبه شده در مواد تکمیلی قابل مشاهده هستند. به طور کلی، ما دریافتیم که مقادیر اهمیت با انتظارات ما مطابقت دارند. به عنوان مثال، مقایسه عملکرد Walker۲d-v۱ بین ACKTR و DDPG را در نظر بگیرید. عملکرد ACKTR کمی بهتر است، اما این عملکرد به دلیل همپوشانی فواصل اطمینان دو الگوریتم معنادار نیست: $t = ۱.۰۳$، $p = ۰.۳۳۴$، $KS = ۰.۴۰$، $p = ۰.۶۹۷$، تفاوت درصدی بوت‌استرپ ۴۴.۴۷٪ (-۸۰.۶۲٪, ۱۱۱.۷۲٪).

---

[1] Power
[2] Lift
[3] Uniform
[4] Significance
[5] Gain
[6] Supervised Learning
[7] Corrected
[8] Kolmogorov-Smirnov Test

Environment	DDPG	ACKTR	TRPO	PPO
HalfCheetah-v1	5037 (3664, 6574)	3888 (2288, 5131)	1254.5 (999, 1464)	3043 (1920, 4165)
Hopper-v1	1632 (607, 2370)	2546 (1875, 3217)	2965 (2854, 3076)	2715 (2589, 2847)
Walker2d-v1	1582 (901, 2174)	2285 (1246, 3235)	3072 (2957, 3183)	2926 (2514, 3361)
Swimmer-v1	31 (21, 46)	50 (42, 55)	214 (141, 287)	107 (101, 118)

جدول ۳-۳: میانگین بوت‌استرپ و حدود اطمینان ۹۵٪ برای زیرمجموعه‌ای از آزمایش‌های محیط. در این آزمایش‌ها از ۱۰ هزار تکرار بوت‌استرپ و روش محوری استفاده شده است.

۳-۴ نتیجه کلی:

از طریق آزمایش‌های متمرکز بر روش‌های گرادیان سیاست برای کنترل پیوسته، ما مشکلات قابلیت بازتولید در یادگیری تقویتی عمیق را بررسی کردیم. دریافتیم که هم منابع درونی (مانند دانه‌های تصادفی و ویژگی‌های محیط) و هم منابع بیرونی (نظیر ابرپارامترها و کدهای پایه) عدم قطعیت می‌توانند به دشواری‌های قابلیت بازتولید الگوریتم‌های پایه کمک کنند. بر اساس نتایج آزمایش‌ها، برخی توصیه‌های کلی را می‌توان ارائه داد: ابرپارامترها می‌توانند اثرات متفاوتی بر الگوریتم‌ها و محیط‌ها داشته باشند. بنابراین یافتن مجموعه‌ای که حداقل عملکرد گزارش شده اولیه الگوریتم‌های پایه را تکرار کند، مهم است. همچنین پیاده‌سازی‌های جدید برای مقایسه باید با نتایج کد پایه اصلی مطابقت داشته باشند. با توجه به واریانس بالا در میان آزمایش‌ها و دانه‌های تصادفی، باید آزمایش‌های متعددی با دانه‌های مختلف انجام شود. استفاده از آزمون‌های معنی‌دار مناسب برای تعیین اینکه آیا بازده‌های بالاتر نشان‌دهنده عملکرد بهتر هستند یا نه، ضروری است.گزارش همه جزئیات از جمله ابرپارامترها، پیاده‌سازی، تنظیمات آزمایشی و روش‌های ارزیابی برای هم روش‌های مقایسه پایه و هم کارهای جدید، مهمترین گام برای قابلیت بازتولید است. بدون انتشار پیاده‌سازی‌ها و جزئیات مربوطه، تلاش‌های بیهوده برای تکرار کارهای برتر، جامعه را آزار خواهد داد و پیشرفت را کند می‌کند.

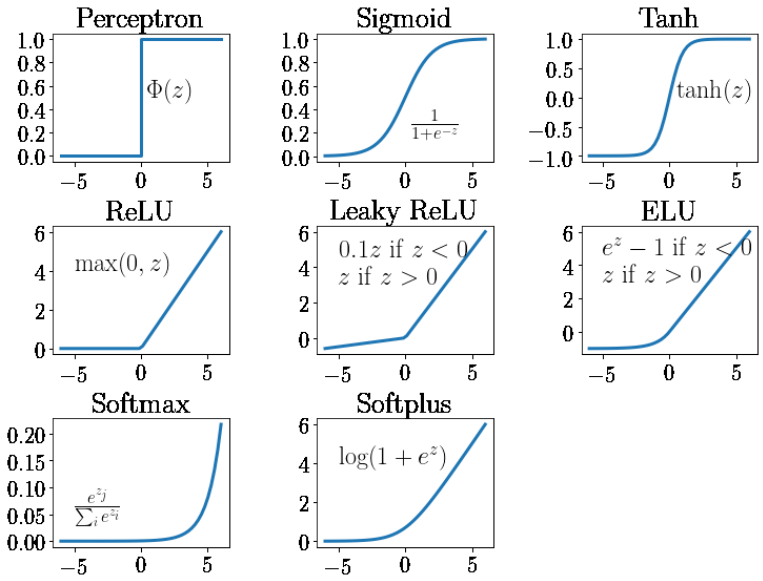
پیوست اول
شبکه های عصبی و توابع فعال‌سازی

توابع فعال‌سازی، برای تعیین خروجی شبکه‌های عصبی، مانند بله یا خیر استفاده می‌شوند. این توابع، یک نگاشت بین مقادیر حاصل (خروجی) و بازه ۰ تا ۱، یا ۱- تا ۱ (بسته به تابع) ایجاد می‌کنند. توابع فعال‌سازی، به دو دسته خطی[1] و غیر خطی[2] تقسیم بندی می‌شوند. توابع فعال‌سازی غیر خطی، پر استفاده ترین توابع فعال‌سازی هستند، زیرا این ویژگی، تعمیم یا انطباق مدل با انواع داده‌ها و تمایز بین خروجی ها را برای مدل آسان می‌کند. در اینجا، به معرفی برخی از توابع فعال‌سازی غیر خطی پرکاربرد می‌پردازیم:

- **تابع Sigmoid: منحنی این تابع، شبیه به شکل حرف S است. دلیل اصلی استفاده از این تابع این است که برد آن، بازه (۰،۱) است. از آنجایی که احتمال هرچیزی مقداری بین ۰ و ۱ است، بنابراین این تابع به ویژه برای مدل هایی که باید احتمال را به عنوان خروجی پیش‌بینی کنیم، انتخاب درستی است و استفاده می‌شود. این تابع همچنین می‌تواند باعث شود که شبکه عصبی در زمان آموزش، گیر کند.**
- **تابع Tanh: محنی این تابع نیز، شبیه به شکل حرف S است و محدوده آن، بازه (۱،۱-) است. مزیت این تابع این است که ورودی های منفی، به مقادیر به شدت منفی و ورودی های صفر، به مقادیر نزدیک به صفر، نگاشت می‌شوند. تابع Tanh، عمدتا برای طبقه‌بندی[3] استفاده می‌شود.**
- **تابع ReLU: از آنجایی که این تابع در تقریبا تمام شبکه‌های عصبی پیچشی[4] و یادگیری عمیق استفاده می‌شود، در حال حاضر پر استفاده ترین تابع فعال‌سازی است. همان‌طور که در شکل آ-۱ می‌بینید، این تابع برابر ۰ است وقتی ورودی آن کمتر از ۰ (منفی) باشد و برابر ورودی اش است وقتی ورودی آن بیشتر یا مساوی ۰ (نامنفی) باشد. مسئله اصلی این تابع این است که تمام مقادیر منفی، بلافاصله ۰ می‌شوند، بنابراین نگاشت خوبی از مقادیر منفی ایجاد نمی‌شود که این موضوع توانایی مدل را برای برازش[5] یا آموزش صحیح از داده‌ها کاهش می‌دهد. این مشکل، در واقع همان مشکل رلو مرده است که به آن اشاره کردیم.**
- **تابع Leaky Relu: این تابع، تلاشی برای حل مشکل رلو مرده است. همان‌طور که در شکل آ-۱ می‌بینید، وجود نشت[6] به افزایش محدوده تابع کمک می‌کند. معمولا، مقدار a (شیب خط در قسمت منفی) برابر ۰.۰۱ در نظر گرفته می‌شود.**

---

[1] Linear
[2] Non-Linear
[3] Classification
[4] Convolutional
[5] Fitting
[6] Leak



شکل آ-۱: برخی توابع فعال‌سازی پرکاربرد، به همراه رابطه و نمودار آن‌ها

مراجع

[1] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D., "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[2] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al., "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484-489, 2016.

[3] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M., "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[4] Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A. S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., et al., "Starcraft ii: A new challenge for reinforcement learning," *arXiv preprint arXiv:1708.04782*, 2017.

[5] Silva, V. d. N., and Chaimowicz, L., "Moba: a new arena for game ai," *arXiv preprint arXiv:1705.10443*, 2017.

[6] Islam, R., Henderson, P., Gomrokchi, M., and Precup, D., "Reproducibility of benchmarked deep reinforcement learning tasks for continuous control," in *ICML Reproducibility in Machine Learning Workshop*, 2017.

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

[8] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P., "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

[9] Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P., "Benchmarking deep reinforcement l earning for continuous control," in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016.

[10] Whiteson, S., Tanner, B., Taylor, M. E., and Stone, P., "Protecting against evaluation overfitting in empirical reinforcement learning," in *2011 IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning, ADPRL 2011*, Paris, France, Apr. 12-14, 2011, pp. 120-127.

[11] Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M., "Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents," *arXiv preprint arXiv:1709.06009*, 2017.

[12] Wagstaff, K., "Machine learning that matters," *arXiv preprint arXiv:1206.4656*, 2012.

[13] Boulesteix, A.-L., Lauer, S., and Eugster, M. J., "A plea for neutral comparison studies in computational sciences," *PloS One*, vol. 8, no. 4, p. e61562, 2013.

[14] Stodden, V., Leisch, F., and Peng, R. D., Implementing reproducible research, CRC Press, 2014.

[15] Bouckaert, R. R., and Frank, E., "Evaluating the replicability of significance tests for comparing learning algorithms," in *Proc. PAKDD*, 2004, pp. 3-12.

[16] Bouckaert, R. R., "Estimating replicability of classifier learning experiments," in *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004.

[17] Vaughan, R., and Wawerla, J., "Publishing identifiable experiment code and configuration is important, good and easy," *arXiv preprint arXiv:1204.2235*, 2012.

[18] Sutton, R. S., and Barto, A. G., "Reinforcement Learning: An Introduction", 2nd ed., in progress, 2014, 2015.

[19] Achiam, J., "Spinning Up Documentation", Release, Feb. 07, 2020.

[20] Wu, Y., Mansimov, E., Liao, S., Grosse, R., and Ba, J., "Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation," *arXiv preprint arXiv:1708.05144*, 2017.

[21] Todorov, E., Erez, T., and Tassa, Y., "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)*, Vilamoura, Algarve, Portugal, Oct. 7-12, 2012, pp. 5026-5033.

[22] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W., "OpenAI gym," *arXiv preprint arXiv:1606.01540*, 2016.

[23] Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y., "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems (NIPS)*, 2000.

[24] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P., "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[25] Plappert, M., Houthooft, R., Dhariwal, P., Sidor, S., Chen, R., Chen, X., Asfour, T., Abbeel, P., and Andrychowicz, M., "Parameter space noise for exploration," *arXiv preprint arXiv:1706.01905*, 2017.

[26] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K., "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016, pp. 1928-1937.

[27] Gu, S., Lillicrap, T., Ghahramani, Z., Turner, R. E., Schölkopf, B., and Levine, S., "Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning," arXiv preprint arXiv:1706.00387, 2017.

[28] Xu, B., Wang, N., Chen, T., and Li, M., "Empirical evaluation of rectified activations in convolutional network," arXiv preprint arXiv:1505.00853, 2015.

[29] Rajeswaran, A., Lowrey, K., Todorov, E., and Kakade, S., "Towards generalization and simplicity in continuous control," arXiv preprint arXiv:1703.02660, 2017.

[30] Kohavi, R., et al., "A study of cross-validation and bootstrap for accuracy estimation and model selection," in IJCAI, vol. 14, 1995.

[31] Nadeau, C., and Bengio, Y., "Inference for the generalization error," in Advances in Neural Information Processing Systems (NIPS), 2000.

[32] Hofer, L., and Gimbert, H., "Online reinforcement learning for real-time exploration in continuous state and action Markov decision processes," *arXiv preprint arXiv:1612.03780*, 2016.

[33] Mandel, T., Liu, Y.-E., Brunskill, E., and Popovic, Z., "Offline evaluation of online reinforcement learning algorithms," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

[34] Yuan, K.-H., and Hayashi, K., "Bootstrap approach to inference and power analysis based on three test statistics for covariance structure models," *British Journal of Mathematical and Statistical Psychology*, vol. 56, no. 1, pp. 93-110, 2003.

[35] Bouckaert, R. R., and Frank, E., "Evaluating the replicability of significance tests for comparing learning algorithms," in *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, 2004, pp. 3-12, Springer.

[36] Wilcox, R., "Kolmogorov–Smirnov test," in *Encyclopedia of Biostatistics*, 2005.

Deep Reinforcement Learning

Parniyan Malekzadeh

p.malekzadeh@ec.iut.ac.ir

June 15, 2024

Department of Electrical and Computer Engineering

Isfahan University of Technology, Isfahan 84156-83111, Iran

Degree: B.Sc. Language: Farsi

Suppervisor: Prof. Mohammad Hossein Manshaei (manshaei@cc.iut.ac.ir)

Abstract:
In recent years, significant progress has been made in solving challenging problems across various domains using deep reinforcement learning (RL). Reproducing existing work and accurately judging the improvements offered by novel methods is vital to sustaining this progress. Unfortunately, reproducing results for state-of-the-art deep RL methods is seldom straightforward. In particular, non-determinism in standard benchmark environments, combined with variance intrinsic to the methods, can make reported results tough to interpret. Without significance metrics and tighter standardization of experimental reporting, it is difficult to determine whether im provements over the prior state-of-the-art are meaningful. In this paper, we investigate challenges posed by reproducibility, proper experimental techniques, and reporting procedures. We illustrate the variability in reported metrics and results when comparing against common baselines and suggest guidelines to make future results in deep RL more reproducible. We aim to spur discussion about how to ensure continued progress in the field by minimizing wasted effort stemming from results that are non-reproducible and easily misinterpreted.

Key Words: Deep Reinforcement Learning, Reproducability, Experimental Techniques, Benchmark Environments.