یادگیری تقویتی (reinforcement learning):

یادگیری تقویتی، یکی از روش های اموزش یادگیری ماشین برپایه تشویق رفتار های مورد نظر و تنبیه در برابر رفتار های ناخواسته است. به زبان ساده تر، مدلی است که در آن عامل یا عامل هایی توانایی ادراک و تعامل با محیط، اتخاذ عمل و یادگیری از طریق ازمون و خطا را دارا می‌باشند.

یادگیری تقویتی چگونه کار می‌کند:

در یادگیری تقویتی، توسعه دهنده مدل و یا فرد یا افرادی متخصص، روشی برای تشویق رفتار های درست و تنبیه رفتار های نادرست پیشنهاد می‌دهد. این روش مقادیر برای هر عملی که عامل به آن تصمیم میگیرد، مقداری مثبت یا منفی با توجه به استراتژی به عنوان پاداش یا تنبیه اختصاص می‌دهد تا عامل از این طریق به انجام رفتار های بهتر تشویق شود و از انجام رفتار های نادرست دوری کند. این کار باعث برنامه ریزی مدل در مسیر به دست آوردن حداکثر امتاز ممکن در بلند مدت می‌شود تا در نهایت به پاسخ بهینه مسئله نزدیک شویم.

اهداف بلند مدت به مدل کمک می‌کند تا از تمرکز به روی اهداف کوچکتر بپرهیزد و با گذر زمان، عامل یاد می گیرد تا از رفتار های نادرست و دارای تنبیه دوری کند و به سمت رفتاری هایی با پاداش بیشتر حرکت کند.

یادگیری تقویتی در حوزه هوش مصنوعی به عنوان روشی برای جهت دهی به یادگیری بدون نظارت (unsupervised learning) از طریق تشویق و تنبه به کار گرفته می شود.

کاربرد های یادگیری تقویتی:

علاوه بر مورد توجه قرار گرفتن این روش در حوزه هوش مصنوعی، گستره استفاده آن در مسائل واقعی شامل موارد زیر است:

- بازی
- شخصی سازی پیشنهادات
- رباتیک
- کنترل منابع

یادگیری تقویتی به عنوان ابزاری کاربردی برای بهینه سازی مسائل کنترل پیچیده که حل آن ها از طریق روش های نظری و عددی ممکن نیست و یا بسیار سخت است، خود را ثابت کرده است.

چالش های پیاده سازی یادگیری تقویتی:

در کنار پتانسیل بالا و توانایی این روش در حل بسیاری از مسائل کاربردی، می تواند در پیاده سازی مشکل ساز باشد و در کاربرد محدود است. یکی از محدودیت های پیاده سازی این روش یادگیری ماشین، تکیه آن بر اکتشاف در محیط مسئله است.

برای مثال مسئله کنترل یک بازوی رباتیک را فرض کنید. عامل برای یادگیری جابجایی یک شی و هدایت بازو در یک محیط فیزیکی پیچیده به دنبال رسیدن به حالت های جدید و اتخاذ کردن تصمیم و عمل های متفاوت با جابجایی در محیط است. در یک محیط واقعی، انتخاب بهترین عمل و تصمیم ممکنه به صورت پیوسته مسئله سختی است چرا که یک محیط واقعی متاوبا و به شکل تصادفی در حال تغییر است.

همچنین مدت زمان صرف شده تا حاصل شدن نتیجه قابل قبول و تایید یادگیری مناسب مدل باعث می شود کاربردی بودن این روش محدود و حساس به منابع محاسباتی (توان پردازش کامپیوتری) باشد.

هرچند بزرگترین چالش موجود در مسائلی که توسط یادگیری تقویتی از آنها استفاده می شود، تعیین سیستم تنبیه و تشویق مناسبی است تا باعث شود مدل در مسیر مورد نظر حرکت کند و در جهت یافتن پاسخ مورد نظر انتخاب و تجربه کند.

اکتشاف و بهره برداری (exploration and exploitation):

تعامل اکتشاف و بهره برداری یکی از مسائل شناخته شده در سیستم هایی است که به صورت مکرر اقدادم به گرفتن تصمیم هایی با نتایج نامشخص می کنند را بیان می کند. در واقع دوراهی برای سیستم تصمیم گیری که دانشی نا کامل از محیط خود دارد که ایا با توجه به تجربیات عامل تا این لحظه، آیا بهترین تصمیم با توجه به تجربیات گذشته (تکرار تصمیمی که در گذشته بهترین نتیجه را داده است) را انتخاب کند (بهره برداری) و یا تصمیم جدید و آزموده نشده ای را انتخاب کند به امید اینکه شاید پاداشی بزرگتر از تصمیمات ازموده شده دریافت شود.

عناصر اصلی هر سیستم یادگیری تقویتی:

1. عامل : بخشی از سیستم که بر اساس تشویق و مجازات به انتخاب عمل های متقاوت دست می زند.
2. سیاست: استراتژی که به دنبال به دست آوردن هدف مورد نظر توسط عامل مورد استفاده قرار می گیرد.
3. سیگنال پاداش: مقدار عددی که پس از انجام عمل به عنوان پاداش به عامل داده می شود.
4. تابع مقدار: سیگنال پاداش نشان دهنده انتخاب خوب در زمان حال است در حالی که تابع مقدار مشخص می کند در بلند مدت کدام انتخاب پاداش بیشتری را فراهم می آورد.

برخی انواع یادگیری تقویتی:

Q-learning[1] :

در این روش، مدل هیچ اطلاعاتی از سیاست بهینه ندارد و اکتشاف محیط خود را بدون هدایت بیرونی انتخاب می کند.

SARSA (state-action-reward-state-action):

در این روش، الگوریتم با دادن سیاستی که شناخته شده است شروع به کار می کند. سیاست در واقع تابع احتمالی است که احتمال نتیجه دادن پاداش برای هر عملی را به ما می دهد.

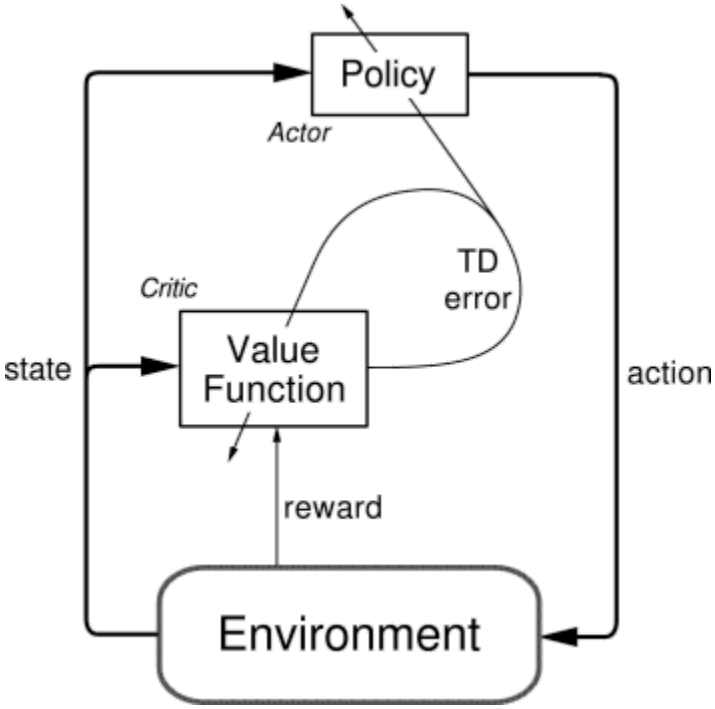
DQN (deep Q-networks):

در Q-learning سیاست به شکل یک ماتریس n بعدی که در ابتدا تمامی مقادیر آن 0 است در دسترس مدل می باشد. با توجه به گسترش استفاده از یادگیری تقویتی در مسائل پیچیده تر، ذخیره ماتریسی به ابعاد تمامی حالت های ممکن مسئله امکان پذیر نیست؛ از این رو برای پیشبینی پاداش احتمال هر تصمیم در هر وضعیتی، یادگیری تقویتی شبکه های عصبی عمیق را به کار می گیرد.
تصمیماتی که در آینده گرفته می شوند برپایه نمونه های تصادفی تصمیمات مفید قبلی یادگرفته شده توسط مدل می باشد.

Actor-Critic method :

این مدل از خانواده روش های تفاوت زمانی (temporal difference TD) است که دارای حافظه ای جدا برای نشان دادن صریح سیاست مستقل از تابع مقدار است. این ساختار سیاست به نام عامل شناخته می شود چراکه برای انتخاب عمل مورد نظر استفاده می شود و تابع مقدار به نام منتقد (critic) شناخته می شود چراکه این تابع به نقد عمل انتخابی توسط عامل می پردازد. این یادگیری به صورت مطابق سیاست (on policy) اتفاق می افتد. منتقد باید به یادگیری و نقد هر سیاستی که توسط عامل استفاده می شود بپردازد. منتقد شکل خطای تفاوت زمانی (temporal difference error) را به خود میگیرد. این مقدار تنها خروجی منتقد است و یادگیری عامل و منتقد را هدایت می کند.

روش های عامل منتقد بسط طبیعی از ایده ی روش های مقایسه تقویتی تا یادگیری تفاوت زمانی تا مسئله نهایی یعنی یادگیری تقویتی هستند. معمولا منتقد تابعی از حالت و مقدار است. پس از انتخاب هر عمل، منتقد به ارزیابی حالت جدید به دست آمده می پردازد تا تشخیص دهد آیا شرایط بهتری به دست آمده یا بدتر شده است. این ارزیابی همان خطای تفاوت زمانی است.



شکل 1-1: مدل عامل-منتقد (actor-critic)

خاج از سیاست و مطابق سیاست (off-policy and on-policy):

در مقایسه روش های یادگیری تقویتی برای هایپر پارامتر ها، بهینه سازی امری پر هزینه است و معمولا عملا غیر قابل اجرا می باشد. به همین دلیل عملکرد این الگوریتم ها به وسیله تاثیرات متقابل در راستای سیاست (مطابق سیاست) با محیط هدف ارزیابی می شود. این تاثیرات متقابل یادگیرنده های مطابق سیاست به ما کمک می کند تا بینش بهتری نسبت به سیاست استفاده شده توسط عامل به دست بیاوریم.

در حالی که در حالت خارج از سیاست، مستقل از سیاستی که عامل از آن استفاده می کند و انگیزه عامل، به دنبال پیدا کردن سیاست بهینه می رود.

به بیان دیگر، در روش های منطبق بر سیاست، فرض بر این است که عامل همواره از بهترین انتخاب بر اساس سیاست استفاده می کند و سیاست با این فرض بهینه می شود اما در روش های خارج از سیاست، سیاست استفاده شده توسط عامل مستقل از بهترین عمل ممکن استفاده می شود.

یادگیری تقویتی مطابق سیاست، زمان هایی مناسب هست که میخواهیم هدف عاملی که در حال اکتشاف است را بهینه کنیم در حالی که زمانی که عامل زیاد به اکتشاف نمی پردازد، روش های خارج از سیاست هزینه کمتری دارند و معمولا مناسب تر اند.

Policy Gradiant :

روش های گرادیان سیاست تکنیکی از یادگیری تقویتی هستند که روی بهینه سازی پارامتر های سیاست مورد استفاده با توجه به پاداش مورد توقع (نتیجه ای که در بلند مدت به دست می آید) توسط الگوریتم گرادیان کاهشی Gradient descent (یا گرادیان افزایشی Gradient Ascend) تکیه می کنند که منجر به پیدایش دسته بندی AC) Actor Critic) شده اند. در ادامه به دو نمونه از این روش های اشاره میکنیم.

[2] Trust Region Optimization (TRPO)]:

در سال 2015، TRPO استراتژی ناحیه اعتماد را به جای استراتژی جستجوی خط (line search) به یادگیری تقویتی معرفی کرد. TRPO محدودیت واگرایی KL) Kullback-Leibler) را برای فعال کردن ناحیه اعتماد برای بهینه سازی اضافه می کند.

این عمل باعث می شود که به روزرسانی جدید سیاست فاصله زیادی از سیاست قبلی نداشته باشد یا به بیانی دیگر، سیاست جدید در ناحیه اعتماد سیاست قبلی واقع می شود به این معنی که به روزرسانی سیاست زیاد باعث انحراف نمی شود. تابع هدف TRPO را می توانیم به شکل زیر بنویسیم:

$$E_{\tau\sim\pi_{\theta_{old}}}[\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}A_{\theta_{old}}(s,a)]$$

$$subject\ to\ E_{\tau\sim\pi_{\theta_{old}}}[\overline{D_{KL}}(\pi_{\theta_{old}}(.|s),\pi_\theta(.|s))]\leq\delta$$

تفسیر این معادله هم ارز بیشینه سازی تابع هدف به شرط برقراری محدودیت KL به روی طول قدم به روزرسانی سیاست می باشد.

TRPO الگوریتمی نسبتا پیچیده است. محدودیت KL سربار اضافی به شکل محدودیت های سختی برای پروسه بهینه سازی اضافه می کند. همچنین پیاده سازی الگوریتم TRPO کار آسانی نیست. این شرایط باعث می شود تا به PPO روی بیاوریم.

[3] Proximal Policy Optimization (PPO) :

PPO یک روش Policy Gradient برای یادگیری تقویتی است. هدف از این روش، داشتن الگوریتمی با کارایی داده و قابل اعتمادی TRPO تنها با استفاده از بهینه سازی درجه یک است.

PPO یک بهینه سازی درجه یک است که پیاده سازی آن را ساده تر می کند. مشابه تابع هدف TRPO، تابع هدف این الگورتیم نسبت احتمال بینی سیساست جدید و سیاست قدیمی را تعریف می کند:هس

$$r(\theta)=\frac{\pi_\theta(a|s)}{\pi_{\theta old}(a|s)}$$

حال می توانیم تابع هدف TRPO را به شکل زیر اصلاح کنیم:

$$J(\theta)^{TRPO} = E[r(\theta)\hat{A}_{\theta old}(s,a)]$$

بدون اضافه کردن محدودیت ها، تابع هدف میتواند منجر به ناپایداری و یا نرخ همگرایی کند شود. به جای اضافه کردن محدودیت های *PPO*، *KL* پیشنهاد میدهد که نسبت سیاست *r(θ)* در بازه کوچکی نزدیک به *1* محدود باشد. این بازه بین *ϵ-1* و *ϵ+1*است که *ϵ* به عنوان یک هایپر پارامتر به مدل داده می شود و در مقاله اصلی *PPO*، مقدار *0.2* به آن داده شده است. حال میتوانیم تابع هدف *PPO* را به شکل زیر بازنویسی کنیم:

$$J^{\text{CLIP}}(\theta) = \mathbb{E}[\min(r(\theta)\hat{A}_{\theta_{\text{old}}}(s,a), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_{\theta_{\text{old}}}(s,a))]$$

در معادله بالا، تابع *clip* نسبت سیاست را به بازه بین *ϵ-1* و *ϵ+1* محدود می کند. تابع هدف *PPO* مقدار حداقل بین مقدار اصلی و مقدار *clip* شده را به خود میگیرد.

شبه کد PPO:

- مقدار دهی اولیه پارامتر های سیاست، θ و تابع مقدار ϕ
- تعیین مقادیر هایپر پارامتر های: تعداد همه قدم ها (مثلا 4000 قدم) و تعداد به روزرسانی در هر قدم (مثلا 5)
- حلقه تا پایان تعداد همه قدم ها:
  - *o* به دست آوردن حالت، عمل، لگاریتم احتمال پاداش قبلی و پاداش در هر بچ از مجموعه مسیر ها به وسیله انتخاب توسط عامل با سیاست قبلی $\pi_{\theta old}$
  - o تخمین مقدار برتری A_k
  - o حلقه تا پایان تعداد بروزرسانی در هر قدم:
    - پیدا کردن لگاریتم احتمال پاداش بر مبنای π_θ (در اولین مرحله همواره داریم $\pi_\theta = \pi\theta old$) این عمل با اعمال عمل تصمیم گرفته شده به وسیله سیاست جدید π_θ روی حالت قبلی انجام می شود.
    - پیدا کردن خارج قسمت r(θ) = current_log_probability - prev_log_probs(در بالا این نسبت به صورت تقسیم نوشته شده است اما با اضافه کردن لگاریتم میتوان آن را به شکل تفریق بازنویسی کرد.
    - محاسبه $r(\theta)* A_k$ و خارج قسمت clip شده
    - پیدا کردن loss عامل با مقدار clip شده.
    - آموزش عامل
    - بروزرسانی پارامتر های سیاست عامل، θ اپدیت می شود، حال سیاست جدید π_θ را داریم

گاهی نسبت اهمیت با الگوریتم های خارج از سیاست اشتباه گرفته می شود. در الگوریتم های خارج از سیاست، نسبت اهمیت برابر نسبت سیاست عملی β و سیاست جدید π است. این ها دو سیاست جداگانه اند و نسبت اهمیت برای همگرایی الگوریتم های خارج از سیاست لازم است. این شباهت باعث می شود PPO به عنوان الگوریتم خارج از سیاست دیده شود اما الگوریتمی منطبق بر سیاست است. در اینجا این نسبت با πθ_old محاسبه می شود که درواقع مقداری قدیمی همان سیاست است. تلاش برای نگه داشتن سیاست جدید در ناحیه اعتماد سیاست قبلی با نگه داشتن نسبت اهمیت در بازه clip شده باعث همگرایی سریع تر روش می شود.

سیاست مورد استفاده:

با توجه به ذات مسئله ارائه شده، تصمیمات متوالی عامل میتواند در آینده تشویق و یا تنبیه هایی را باعث شود که سیاست های خطی و یاغیر خطی قادر به حل روابط بین تصمیم کنونی و پاداشی که شاید در 100 قدم اینده به دلیل این تصمیم اتفاق افتاده است نیستند.

PPO یکی از الگوریتم های یادگیری تقویتی عمیق (Deep Reinforcement Learning) است و میتوانیم با به کار گیری شبکه های عصبی بازگشتی (Recurrent Neural Networks) در این الگوریتم مقادیری از خطای ناشی از پاداش دارای تاخیر را کم کنیم.

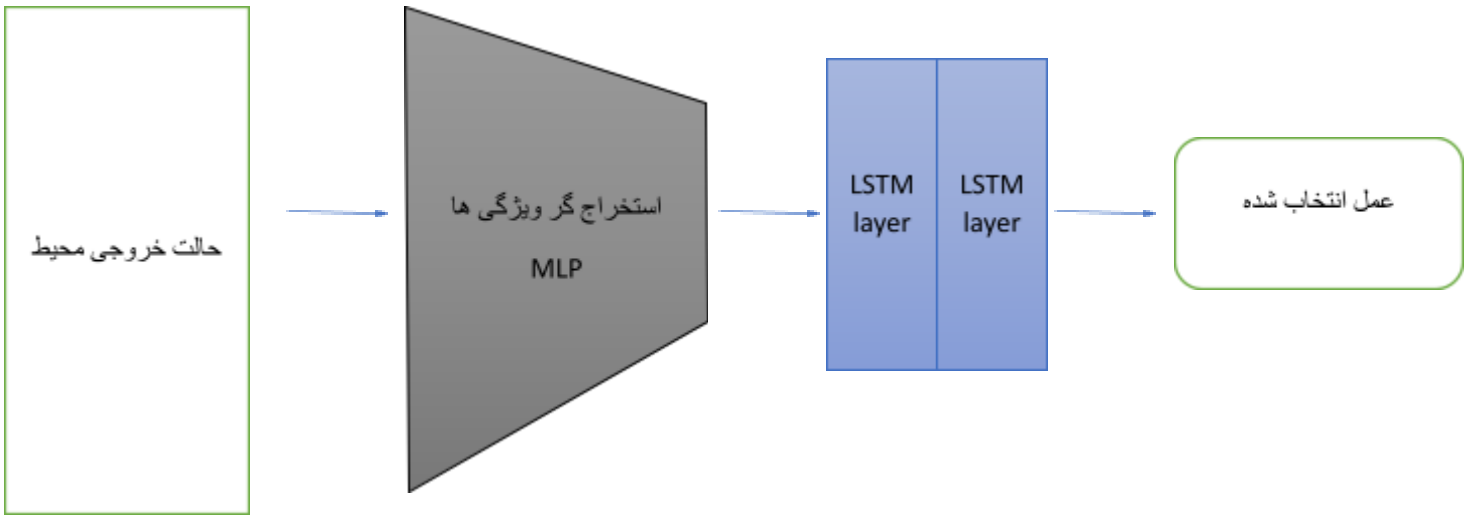
در حل این مسئله ما از LSTM Network استفاده می کنیم. لایه های LSTM دو حافظه کوتاه مدت و بلند مدت می باشد که قابلیت درک تاثیرات تصمیم این لحظه و تاثیر آن در آینده را به ما می دهد.

سیاست مورد استفاده به شکل زیر طراحی شده است:

سیاست استفاده شده شامل دو بخشی اصلی است:

- استخراج گر ویژگی: این بخش بین عامل و منتقد مشترک است و وظیفه آن استخراج ویژگی ها از حالت محیط که به صورت ورودی به شبکه داده می شود می باشد.
- شبکه کاملا متصل که ویژگی های استخراج شده را به خروجی عامل یعنی عمل/مقدار تبدیل میکند.

شبکه استفاده شده در این مسئله شامل دو لایه LSTM کاملا متصل است.



جزییات شبیه سازی محیط:

محیط شبیه سازی شده شامل عناصر زیر می‌باشد:

- مرکز تعمیر و نگهداری شامل 2 (و یا تعداد داخواه) تعمیرکار
- سایت نت
- بخش دپو برای AGV ها

- 3 (یا تعداد دلخواهی) AGV که فرآیندهای مربوط به آنها در محیط شبیه سازی شده اتفاق می‌افتد

هر AGV دارای 3 بخش اصلی 1- مکانیکال، 2- الکتریکال و 3- باتری است که هرکدام شامل سنسور های مختص خود هستند.

سلامت بخش های مختلف دستگاه در محیط شبیه سازی شده توسط درصد سلامت آن بخش نمایش داده می‌شود و در صورت بروز ناهنجاری (abnormality) در هر بخش، با توجه به میانگین زمان محاسبه شده تا خرابی دستگاه، میزان سلامت کلی بخش تغییر می‌کند.

تمامی مقادیر مربوط به این بخش‌ها با حرکت کردن دستگاه به روز می شود، در صورتی که دستگاه خراب شده باشد (یکی از سه بخش اصلی دستگاه دارای درصد سلامت 0 باشد و یا میزان باتری آن 0 درصد باشد) دستگاه از هر محلی که قرار گرفته است به سمت مرکز تعمیر و نگهداری انتقال داده می‌شود و تا زمانی که عامل تصمیم به تعمیر بخش آسیب دیده نگیرد، دستگاه به چرخه کار باز نمی‌گردد.

هر قدم شبیه سازی مدل با پایان رویداد تعریف شده برای دستگاه انتخاب می‌شود. به هر دستگاه یکی از رویداد های زیر نسبت داده می‌شود:

1. انجام کار مربوط به خط تولید
2. حضور در بخش تعمیر و نگهداری تا پایان تعمیرات
3. آماده به کار در بخش دپو
4. جابجایی از دپو به سمت سایت نت
5. جابجایی از دپو به سمت تعمیر و نگهداری
6. جابجایی از سایت نت به سمت تعمیر و نگهداری
7. جابجایی از سایت نت به سمت دپو
8. جابجایی از تعمیر و نگهداری به سمت دپو
9. جابجایی از تعمیر و نگهداری به سمت سایت نت
10. آماده به کار در سایت نت
11. آماده به کار در تعمیر و نگهداری

زمان جابجایی بین ایستگاه های مختلف بر اساس موقعیت مکانی انتخاب شده برای مدل محاسبه می شود، و زمان رویداد های آماده به کار بودن دستگاه در سایت نت و بخش تعمیر و نگهداری صفر است، اگر در همان لحظه، وظیفه ای به دستگاه داده شود، دستگاه شروع به حرکت به سمت محل مورد نظر می‌کند تا وظیفه دریافت شده را تکمیل کند، در غیر اینصورت، دستگاه به سمت محل دپو شروع به حرکت می کند تا زمانی که وظیفه ای به بافر اضافه شود.

ورودی مدل:

ورودی مدل برای هر مرحله باید وضعیت دستگاه باشد، با هدف کوچکتر تر کردن فضای مشاهده (observation space) مسئله، هر قدم از مراحل شبیه سازی محیط تنها مربوط به یکی از AGV ها می‌باشد و یا در صورت حضور تمامی AGV ها در بخش تعمیر و نگهداری، تا زمان تمام شدن تمامی تعمیر و نگهداری های تصمیم گرفته شده برای اولین AGV، تمامی قدم ها مربوط به بخش تعمیر و نگهداری خواهد بود.

ورودی مدل شامل 4 وضعیت آخر هر AGV است که عامل باید برای تصمیم گیری برای دستگاه اقدام کند. خروجی وضعیت دستگاه شامل برداری با درایه های زیر است:

1. رویداد کنونی
2. حجم وظایف ذخیره شده در بافر
3. درصد سلامت بخش مکانیکال
4. ناهنجاری در بخش مکانیکال

5. درصد سلامت بخش الکتریکال
6. ناهنجاری در بخش الکتریکال
7. درصد سلامت باتری
8. سطح شارژ کمتر از ۲۰ درصد
9. زمان باقی مانده تا تعمیر و نگهداری پیشگیرانه

معماری انتخاب شده برای مدل به شکلی است که در هر مرحله تنها تصمیم گیری برای یک دستگاه ممکن باشد و همینطور ورودی مدل وضعیت کنونی همان دستگاه است، این کار به معنی شبیه سازی و آموزش مدل برای هر دستگاه به طور جداگانه می باشد که باعث می شود مدل مستقل از تعداد دستگاه های موجود فرآیند یادگیری را طی کند.

خروجی مدل:

پس از دریافت تصمیم گرفته شده توسط مدل، عمل ماسک گذاری تصمیم انجام می شود. به عنوان مثال، اگر تنها بخش مکانیکال دستگاه آسیب دیده باشد عامل نباید برای بخش الکتریکال که درصد سلامت بالایی هم دارد تصمیم به انجام فرآیند تعمیر و نگهداری بگیرد. علاوه بر اصلاح تصمیم عامل و جلوگیری از انجام تصمیمات اشتباه، به ازای هر تصمیم نادرست پاداش منفی نسبتا بزرگی به عامل داده می شود تا از تکرار این انتخاب اشتباه جلوگیری شود.

اگر بخشی از دستگاه به صورت کامل خراب شده باشد(درصد سلامت آن بخش به ۰ رسیده باشد) عامل پاداش منفی بسیار بزرگی دریافت می کند تا زمانی که تصمیم به تعمیر و نگهداری بگیرد.

همچنین با توجه به ضرورت شارژ باتری، علاوه بر امکان تصمیم گیری عامل برای این منظور، در هر مرحله ای که دستگاه نیاز به شارژ داشته باشد و عامل این عمل را انتخاب نکند، سیستم به صورت خودکار دستگاه را برای شارژ به بخش تعمیر و نگهداری می فرستد تا دستگاه به صورت کامل شارژ شود و سپس به چرخه تولید بازگردد.

شبیه سازی محیط:

در این مرحله برای بررسی رفتار محیط، به دست آوردن معیاری برای مقایسه و مقایسه تاثیر فواصل مختلف پیشنهاد شده برای جایگاه تعمیر و نگهداری و محل دپو دستگاه ها، شبیه سازی را با تصمیمات مصنوعی اجرا کرده ایم.

سه حالت متفاوت پیشنهادی برای بخش های مختلف به شکل زیر می باشد:

1. محل دپو AGV ها در کنار سایت نت باشد و مکانی جدا برای تعمیر و نگهداری تعبیه شود
2. محل دپو ، مکان تعمیر و نگهداری و سایت نت از یکدیگر مجزا باشند
3. محل دپو، مکان تعمیر و نگهداری و سایت نت، هر سه در یک مکان باشند

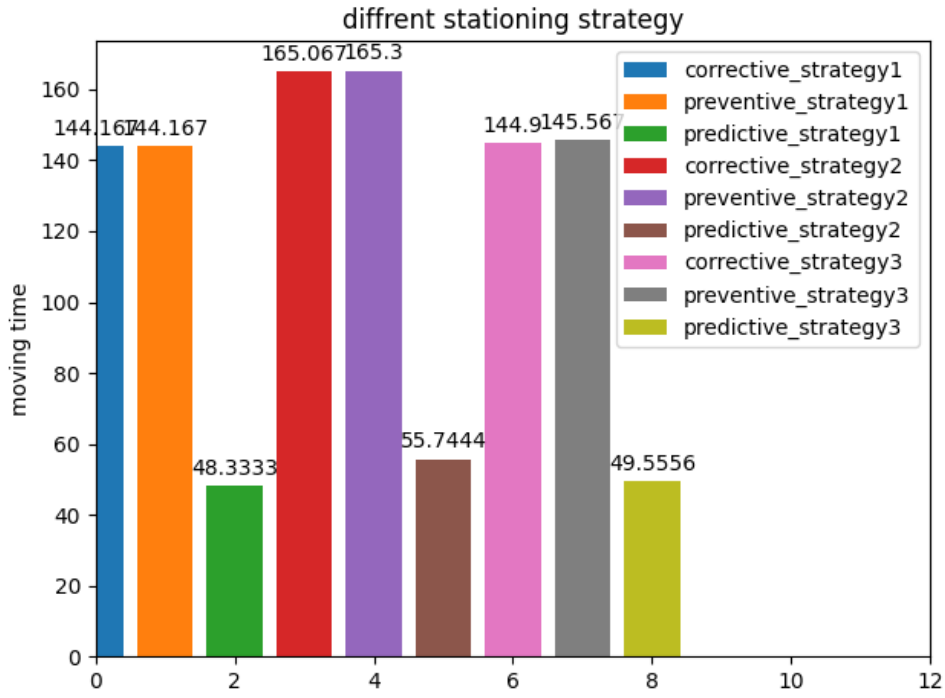
برای شبیه سازی این شرایط، زمان رسیدن AGV به ایستگاه بعدی، در صورتی که هردو ایستگاه در یک مکان تعبیه شده باشند ۱۰ دقیقه و در باقی حالات، ۲۰ دقیقه زمان مورد نیاز در نظر گرفته شده است.

برای بررسی تاثیر فاصله ی این ایستگاه ها برروی مدت زمان کارکرد دستگاه، به ابتدایی ترین حالت مدل را شبیه سازی می کنیم. در این حالت، در صورت وجود ناهنجاری در هرکدام از بخش ها به صورت تصادفی یکی از فرآیند های تعمیر و یا تعویض پیشگیرانه انتخاب می شود، همچنین به محض خرابی و یا اتمام شارژ باتری نیز تعمیرات اصلاحی و شارژ باتری انجام می شود.

به دلیل استفاده از توزیع های مختلف احتمالی امکان باز تولید شرایط به صورت کاملا یکسان وجود ندارد، به همین علت در ادامه هر کدام از حالت های مورد بحث را برای مدت ۶ ماه شبیه سازی کرده و این عمل را برای ۱۰ بار تکرار میکنیم. در اخر برای مقایسه شرایط مختلف، میانگین زمان های صرف شده در هرکدام از فعالیتد های AGV در جای خود مورد بحث قرار می گیرد.

مقایسه شیوه های متفاوت پیشنهادی برای قرار گیری ایستگاه ها:

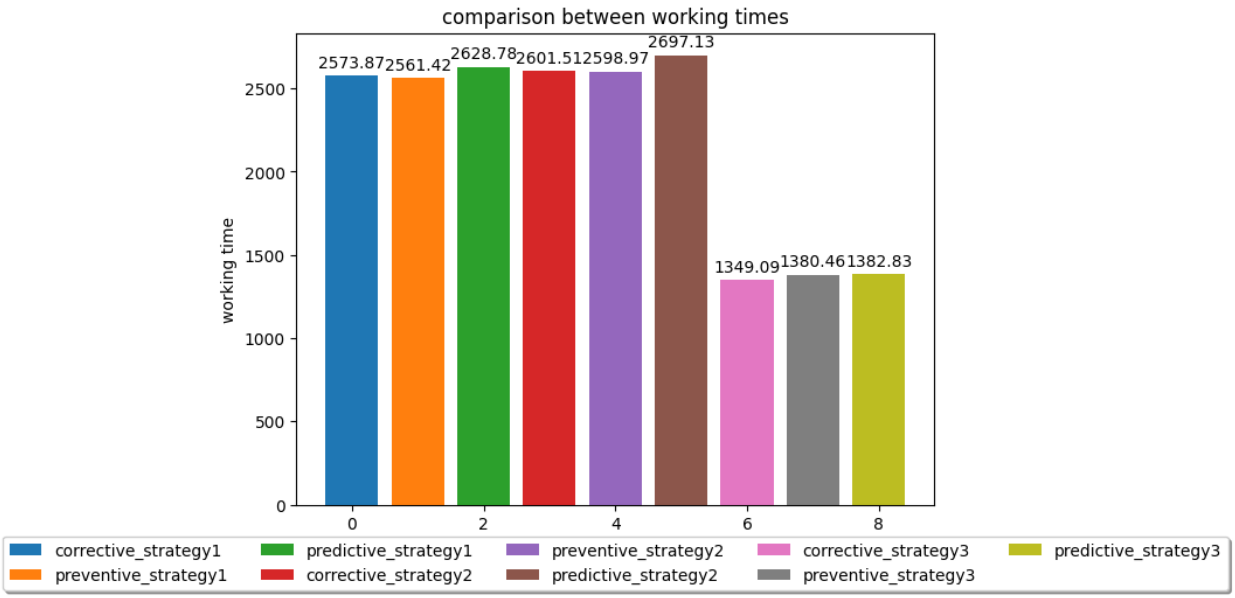
همانطور که در بالا توضیح دادیم، مدل را با حالت های مختلف پیشنهادی برای قرارگیری ایستگاه ها شبیه سازی می کنیم. هدف جهت بیشینه کردن زمان کار کرد هرکدام از AGV ها باید کمترین میزان حرکت بین ایستگاه های مختلف را پیدا کنیم.



همان طور که در شکل بالا مشاهده می کنید، شیوه سوم قرارگیری ایستگاه ها با اختلاف زیادی باعث کاهش زمان حرکت AGV بین ایستگاه های مختلف می شود. به همین دلیل مدل تنها با شبیه سازی حالت قرار گیری تعمیر و نگهداری، دپو و سایت نت در یک مکان آموزش می بیند. البته به دلیل نوع معماری سیاست استفاده شده، استفاده از آن برای حالات دیگر نیز قابل تعمیم است.

مقایسه میانگین زمان کار واقعی AGV:

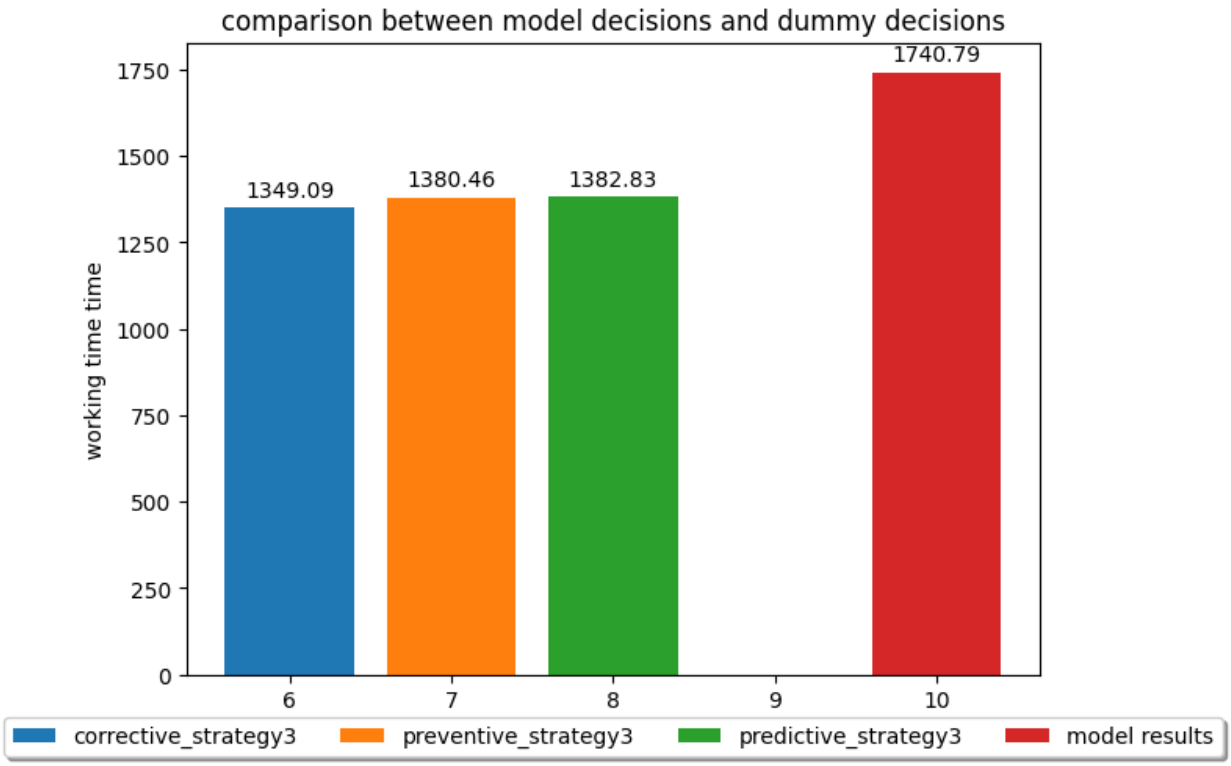
همانطور که در شکل زیر مشاهده می کنید، در ۶ حالت اول میانگین زمان کارکرد واقعی دستگاه ها بسیار بیشتر از ۳ حالت انتهایی است. اما این تنها به دلیل لحاظ نکردن خرابی قطعات کوچکتر دستگاه و استفاده نکردن از سنسور های موجود می باشد. پس از فعال کردن سنسور ها، ناهنجاری در هرکدام از بخش ها باعث کم شدن عمر ان بخش و همچنین افزایش خرابی باقی سنسور ها که هنوز در حالت ناهنجاری قرار نگرفته اند می شود و این اتفاق باعث کاهش ناگهانی زمان کارکرد سنسور ها و افزایش زمان صرف شده در تعمیر و نگهداری می شود.



آموزش مدل(training the model):

مدل برای یادگیری نیاز به تجربه محیط در حالت های مختلف دارد. با توجه به زمان در نظر گرفته شده برای میانگین زمان تا خرابی بخش های مختلف هر دستگاه، فاصله زمانی ۱ ماه برای مدت زمان هر اپیزود در نظر گرفته شده تا عامل در این مدت زمان بتواند به تجربه محیط بپردازد. به دلیل ماهیت احتمالی مدل، در هر بار شبیه سازی عامل در شرایط نسبتا جدیدی قرار می گیرد و همچنین، به دلیل نویز اضافه شده به تصمیم عامل و همینطور شرایط مختلف، در هر مرحله رشته داده هایی که برای backpropagation استفاده می شوند متفاوت خواهد بود. این عمل برای ۱۰۰۰ بار تکرار می شود که به معنی تجربه شدن محیط توسط عامل به مدل ۱۰۰۰ ماه می باشد با این تفاوت که پس از هر ماه محیط به حالت اولیه خود بازگشته و بازی از ابتدا شبیه سازی می شود.

پس از تکمیل آموزش مدل، برای مقایسه نتایج به دست آمده، دوباره محیط را برای ۱۰ بار و هربار به مدت ۶ ماه شبیه سازی می کنیم اما با این تفاوت که تصمیمات گرفته شده در محیط توسط عامل آموزش دیده اتفاق افتاده است. و سپس میانگین زمان های صرف شده را با داده های بالا مقایسه می کنیم.



همانطور که در شکل بالا مشاهده می شود، تصمیمات گرفته شده توسط عامل باعث افزایش حدود ۳۵۰ میانگین ساعتی کار هر AGV شده است که برابر ۲۶.۹٪ می باشد.

نتیجه گیری:

با رشد روز افزون صنعت و افزایش استفاده از ابزار آلات خودکار و به خصوص AGV ها، بهینه سازی مدیریت تعمیرات دستگاه ها امری ضروری است و با توجه به بزرگی این مدل مسایل بهینه سازی و غیر خطی و احتمالی بودن بسیاری از عوامل و متغیر های موجود، حل آن بسیار دشوار است اما نشان دادیم که روش های جدید یادگیری ماشین و یادگیری تقویتی، می تواند راه حلی موثر برای این مدل مسایل ارایه کرده و باعث افزایش سود آوری شرکت ها شود.

در این پژوهش، روشی نوین از یادگیری منطبق بر سیاست PPO ارایه کردیم که مستقل از تعداد AGV های استفاده شده است و میتواند به سرعت استراتژی بهینه ای برای مدیریت تعمیر و نگهداری دستگاه ها را مستقل از بزرگی سیستم یادبگیرد.