
بسم الله الرحمن الرحيم

دانشکده مهندسی صنایع

رساله به‌عنوان تحقق بخشی از شرایط دریافت درجه دکتری

گرایش

کاربردهای یادگیری تقویتی عمیق در برنامه ریزی پروژه با محدودیت منابع

نگارش

امیرحسین رحیمی مقدم

استاد راهنما

دکتر محسن ورمزیار

بهمن 1403

تصویب‌نامه

به نام خدا

دانشگاه صنعتی شریف

دانشکده.....

رساله دکتری

عنوان:.....

نگارش:.....

کمیته ممتحنین:

استاد راهنما:..... **امضاء.....**

استاد راهنمای همکار:..... **امضاء.....**

استاد مشاور:..... **امضاء.....**

استاد مدعو:..... **امضاء.....**

تاریخ:.....

اظهارنامه

(اصالت متن و محتوای رساله دکتری)

عنوان رساله: .....

.....

نام استاد راهنما: ..... نام استاد راهنمای همکار: ..... نام استاد مشاور: .....

این‌جانب ..... اظهار می‌دارم:

۱- متن و نتایج علمی ارائه شده در این رساله اصیل بوده و منحصراً توسط این‌جانب و زیرنظر استادان (راهنما، همکار و مشاور) نام‌برده شده در بالا تهیه شده است.

۲- متن رساله به این صورت در هیچ جای دیگری منتشر نشده است.

۳- متن و نتایج مندرج در این رساله، حاصل تحقیقات این‌جانب به عنوان دانشجوی دکتری دانشگاه صنعتی شریف است.

۴- کلیه مطالبی که از منابع دیگر در این رساله مورد استفاده قرار گرفته، با ذکر مرجع مشخص شده است.

نام دانشجو: .....

تاریخ

امضا

نتایج تحقیقات مندرج در این رساله و دستاوردهای مادی و معنوی ناشی از آن (شامل فرمول‌ها، نرم‌افزارها، سخت‌افزارها و مواردی که قابلیت ثبت اختراع دارد) متعلق به دانشگاه صنعتی شریف است. هیچ شخصیت حقیقی یا حقوقی بدون کسب اجازه از دانشگاه صنعتی شریف حق فروش و ادعای مالکیت مادی یا معنوی بر آن یا ثبت اختراع از آن را ندارد. همچنین کلیه حقوق مربوط به چاپ، تکثیر، نسخه‌برداری، ترجمه، اقتباس و نظائر آن در محیط‌های مختلف اعم از الکترونیکی، مجازی یا فیزیکی برای دانشگاه صنعتی شریف محفوظ است. نقل مطالب با ذکر ماخذ بلامانع است.

نام استادان راهنما: .....

تاریخ

امضا

نام دانشجو: .....

تاریخ

امضا

تقدیم به (اختیاری):

آنانی که برای آینده خود جنگی

تشکر و قدردانی (اختیاری):

از آقا / خانم... به‌خاطر.... سپاس‌گزاری می‌شود.

چکیده

هدف مقاله بررسی روش‌های جدید در زمینه یادگیری تقویتی با اعمال یادگیری عمیق به‌عنوان یک ابزار برای رسیدن به جواب بهتر در زمان کمتر می‌باشد. در این مقاله سعی شده‌است که تمامی روش‌های یادگیری تقویتی و روش‌های یادگیری عمیق استفاده شده و یک راهکار جامع و نو در این زمینه ارائه دهد.در انتها نیز سعی می شود که با توجه به نتایج به دست آمده روش های مختلف یادگیری عمیق تحلیل و مقایسه شوند و برای پروژه های مختلف با توجه به شرایطشان بهترین رویکرد توصیه شود.

کلمات کلیدی (5 تا 7 کلید واژه): شبکه‌های نورونی، برنامه ریزی پروژه، روش تخمین سیاست بهینه، شبکه توجه گرافی، روش یادگیریq

فهرست مطالب

عنوان **صفحه**

فهرست شکل‌ها

.

عنوان **صفحه**

فهرست علائم

فصل1 معرفی پژوهش

1-1 مقدمه

یادگیری تقویتی عمیق (DRL[1]) به‌عنوان یک رویکرد قدرتمند برای حل مسئله زمان‌بندی پروژه با محدودیت منابع (RCPSP[2])، که یک مسئله بهینه‌سازی سخت شناخته شده‌است، ظهور کرده است. روش‌های سنتی مانند الگوریتم‌های دقیق و اکتشافی اغلب با نمونه‌های مقیاس بزرگ به‌دلیل پیچیدگی محاسباتی دست و پنجه نرم می‌کنند. DRL، با استفاده از شبکه‌های عصبی و الگوهای یادگیری تقویتی، یک جایگزین امیدوارکننده با یادگیری سیاست‌های زمان بندی کارآمد از طریق تعامل با محیط ارائه می‌دهد.

در راه‌حل‌های RCPSP مبتنی بر DRL، یک عامل آموزش داده‌می‌شود تا منابع را به‌طور متوالی تخصیص دهد و وظایف را برنامه‌ریزی کند و در عین حال یک هدف را به حداکثر برساند، مانند به حداقل رساندن طول پروژه. فضای حالت معمولاً وضعیت زمان‌بندی فعلی را نشان‌می‌دهد، فضای عمل شامل انتخاب کار و تخصیص منابع است، و تابع پاداش، عامل را به‌سمت زمان‌بندی‌های بهینه هدایت می‌کند. تکنیک‌هایی مانند بهینه‌سازی سیاست پروگزیمال (PPO)، شبکه‌های Q–عمیق (DQN) و روش‌های Actor-Critic برای بهبود کیفیت راه‌حل و سازگاری استفاده‌شده‌اند.

در مقایسه با روش‌های مرسوم، DRL می‌تواند بهتر در بین نمونه‌های مشکل تعمیم دهد، با محدودیت‌های دینامیکی سازگار شود و راه‌حل‌های امکان‌پذیر را بدون شمارش کامل به‌طور کارآمد بررسی کند. با این حال، چالش‌هایی مانند فضاهای حالت بزرگ، پاداش‌های تأخیری، و کارایی آموزش، جهت‌های تحقیقاتی کلیدی در استفاده از DRL در RCPSP باقی‌مانده‌اند.

---

[1] Deep Reinforcement Learning
[2] Resource Constraint Project Scheduling Problem

1-2 برنامه ریزی پروژه

زمان‌بندی پروژه یک جنبه حیاتی از مدیریت پروژه است که شامل برنامه‌ریزی، سازماندهی و کنترل توالی فعالیت‌های مورد نیاز برای تکمیل یک پروژه در یک بازه زمانی مشخص است. زمان و نحوه انجام وظایف را با در نظر گرفتن وابستگی‌های کار، در دسترس بودن منابع و مهلت‌های پروژه مشخص می‌کند. تکنیک‌های رایج شامل روش مسیر بحرانی (CPM[1]) و تکنیک بررسی ارزیابی برنامه (PERT[2]) است که به شناسایی حساس‌ترین وظایف، تخصیص منابع و ارزیابی تأثیر تأخیرهای احتمالی بر روی تاریخ اتمام پروژه می‌پردازد.

با گذشت زمان، زمان‌بندی پروژه از روش‌های سنتی برای ترکیب رویکردهای پیشرفته‌تر، به‌ویژه در پاسخ به پروژه‌های پیچیده و چند وجهی تکامل یافته‌است. اخیراً، مدیریت پروژه ترکیبی، که روش‌های Agile و Waterfall را با هم ترکیب می‌کند، رایج شده‌است و به تیم‌های پروژه اجازه می‌دهد تا ضمن حفظ ساختار، با نیازهای متغیر سازگار شوند.

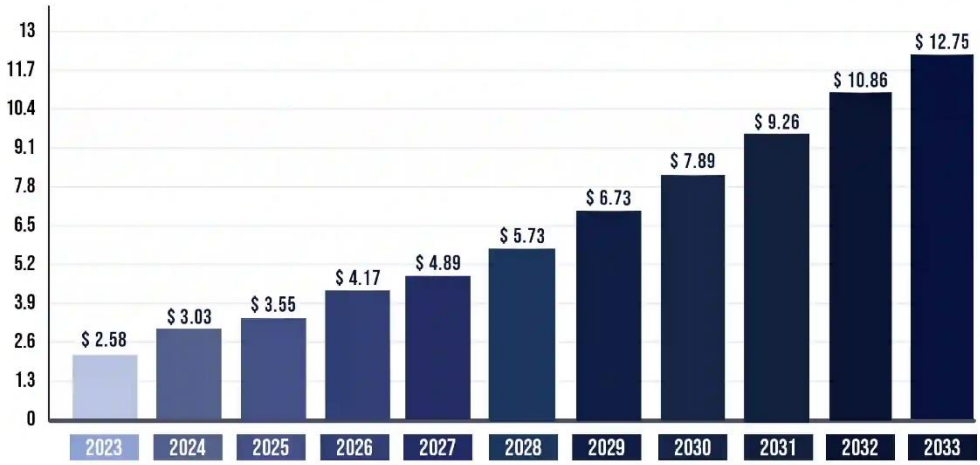
پیشرفت‌های اخیر در فناوری، هوش‌مصنوعی (AI[3]) و یادگیری ماشینی (ML[4]) را وارد برنامه ریزی پروژه کرده است. ابزارهای مبتنی بر هوش‌مصنوعی می‌توانند تأخیرهای احتمالی را پیش‌بینی کنند، تخصیص منابع را بهینه کنند و زمان‌بندی‌ها را در زمان واقعی بر اساس عملکرد پروژه تطبیق دهند. این ادغام فناوری به سیستم‌های زمان‌بندی پویا و انعطاف‌پذیرتر اجازه می‌دهد، مدیران را قادر می‌سازد تا ریسک‌ها را پیش‌بینی کنند، تصمیم‌گیری را بهبود بخشند، و به‌طور مؤثرتر با عدم قطعیت‌ها برخورد کنند.

در 50 سال گذشته شاهد پیشرفت‌های قابل توجهی در زمان‌بندی پروژه بوده‌ایم که با توسعه روش‌های جدید، نوآوری‌های تکنولوژیکی و الگوریتم‌های پیچیده‌تر هدایت می‌شود. روش مسیر بحرانی (CPM) و تکنیک ارزیابی و بررسی برنامه (PERT) در دهه‌های 1970 و 1980 غالب بودند. این روش‌ها که در دهه 1950 توسعه یافتند، به‌دلیل سادگی و اثربخشی در مدیریت پروژه‌های حساس به زمان، به‌ویژه در صنایع ساختمانی و دفاعی، به‌طور گسترده مورد استفاده قرار گرفتند. تکنیک‌های سطح منابع و خرابی برای رفع محدودیت‌های منابع و کاهش مدت زمان پروژه معرفی شدند. با پیچیده‌تر شدن پروژه‌ها، **مشکل زمان‌بندی پروژه با محدودیت منابع (RCPSP) برای رسیدگی به تخصیص منابع محدود

---

[1] Critical Path Method
[2] Project Management Planning Tool
[3] Artificial Intelligence
[4] Machine Learning

پدیدار شـــد. این روش کاســـتی‌های CPM و PERT را که محدودیت‌های منابع را در نظر نمی‌گرفت، برطرف کرد. RCPSP به یک حوزه تحقیقاتی اصـــلی برای حل چالش‌های زمان بندی در ســـاخت و ســـاز، تولید و توســـعه نرم‌افزار تبدیل شـــد. لگوریتم‌های ابتکاری مانند الگوریتم‌های ژنتیک (GAs[1])، بازپخت شبیه سازی شده، و جستجوی تابو برای رسیدگی به پیچیدگی مشکلات زمان بندی پروژه، به‌ویژه در حوزه RCPSP، معرفی شدند. این روش‌ها برای پروژه‌های بزرگ و پیچیده که روش‌های سنتی ناکارآمد بودند، مؤثر بودند. - بهینه‌سازی چند هدفه نیز با هدف بهینه‌ســـازی نه‌تنها زمان، بلکه هزینه، کیفیت و اســـتفاده از منابع نیز مورد توجه قرار گرفت. - اوایل دهه 2000 با ظهور روش‌های مدیریت پروژه چابک، به‌ویژه در صـنعت نرم‌افزار، شـاهد یک تغییر اسـاسـی بودیم. مدیریت پروژه چابک امکان زمان‌بندی تکراری را فراهم می‌آورد که با نیازهای متغیر پروژه سازگار می‌شود. تمرکز را از جدول زمانی ثابت به انعطاف‌پذیری، همکاری مشتری و پاسخ‌گویی تغییر داد



شکل 1روند ارزش بازار برنامه‌ریزی و مدیریت پروژه مبتنی بر هوش مصنوعی در سال های 2023 الی 2033

.

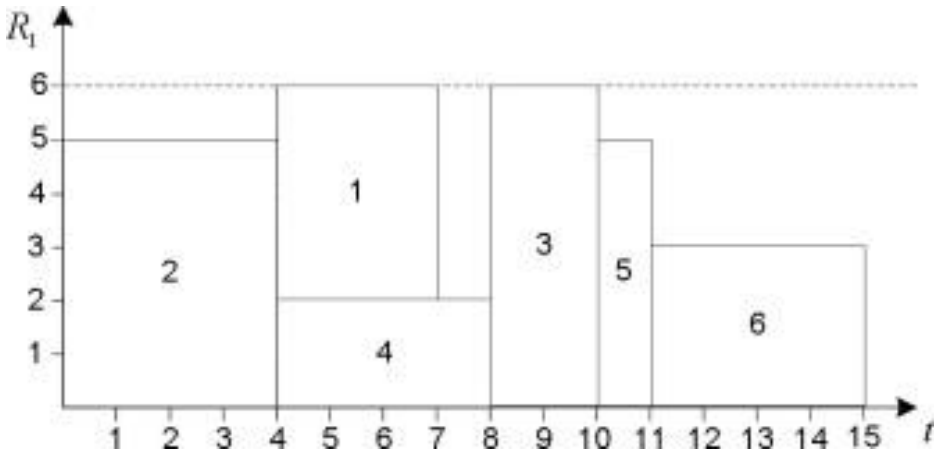
1-2-1 برنامه ریزی پروژه با محدودیت منابع

مســـئله زمان‌بندی پروژه با محدودیت منابع (RCPSP) یک نوع مســـئله بهینه‌ســـازی در مدیریت پروژه است، که در آن هدف، برنامه‌ریزی مجموعه‌ای از فعالیت‌های پروژه در یک بازه

[1] Genetic Algorithms

زمانی معین با در نظر گرفتن محدودیت‌های منابع است. هر فعالیت مدت زمان خاصی دارد و فعالیت‌های خاصی ممکن است به فعالیت‌های دیگر وابسته باشند، به این معنی که برخی از کارها نمی‌توانند شروع شوند تا زمانی که برخی دیگر تکمیل شوند. منابع موجود مانند نیروی انسانی، ماشین آلات یا بودجه محدود است و هر فعالیتی بخشی از این منابع را مصرف می‌کند. هدف RCPSP معمولاً به حداقل رساندن زمان تکمیل پروژه (makespan) در عین رعایت محدودیت‌های منابع و اولویت است.

مشکل زمان‌بندی پروژه با محدودیت منابع (RCPSP) یک مسئله به‌خوبی مطالعه‌شده در تحقیق در عملیات و مدیریت پروژه است. تاریخچه آن را می‌توان به اواسط قرن بیستم بازگرداند، زمانی که محققان شروع به فرموله‌کردن و حل مسائل پیچیده زمان‌بندی کردند.



شکل 2نمایی از مصرف منابع در مسائل RCPSP

پایه‌های RCPSP در دهه‌های 1950 و 1960، در روزهای اولیه تحقیق در عملیات و مدیریت پروژه، گذاشته شد. در این دوره، محققان شروع به بررسی مسائل زمان‌بندی در زمینه‌های صنعتی و ساخت‌وساز کردند. روش مسیر بحرانی (CPM) و تکنیک ارزیابی و بازبینی برنامه (PERT) در اواخر دهه 1950 برای رسیدگی به زمان‌بندی پروژه بدون محدودیت منابع توسعه یافتند. با این حال، این روش‌ها محدودیت منابع را در نظر نمی‌گرفتند، که منجر به نیاز به مدل‌های پیشرفته‌تر شد.

در دهه 1960، محققان شروع به فرموله‌کردن RCPSP به‌عنوان یک مسئله مجزا کردند . RCPSPبه‌عنوان مسئله‌ای تعریف شد که در آن فعالیت‌ها باید با توجه به محدودیت‌های پیش‌نیاز (وابستگی‌های بین وظایف) و محدودیت‌های منابع (دسترسی محدود به منابع مانند

نیروی کار، تجهیزات یا مواد) زمان‌بندی شوند. این دوره شاهد توسعه فرمول‌های ریاضی و الگوریتم‌های اولیه برای حل نمونه‌های کوچک مسئله بود.

دهه 1970 پیشرفت قابل توجهی در درک پیچیدگی RCPSP داشت. محققان ثابت کردند که این مسئله NP-Hard است، به این معنی که یافتن یک راه‌حل بهینه برای نمونه‌های بزرگ از نظر محاسباتی غیرممکن است. این درک منجر به توسعه روش‌های ابتکاری و فرا ابتکاری برای یافتن راه‌حل‌های نزدیک به بهینه شد. در این زمان، الگوریتم‌های شاخه و کران نیز برای حل نمونه‌های کوچک به‌طور بهینه بررسی شدند.

دهه 1980 شاهد افزایش توسعه روش‌های ابتکاری برای RCPSP بود. روش‌های مبتنی بر قوانین اولویت، مانند قانون حداقل زمان تأخیر (MST[1]) و قانون زمان پایان دیرتر (LFT[2])، برای تولید برنامه‌های امکان‌پذیر به‌سرعت محبوب شدند. محققان همچنین شروع به بررسی تکنیک‌های پیچیده‌تر، مانند الگوریتم‌های ژنتیک و تبرید شبیه‌سازی‌شده، برای بهبود کیفیت راه‌حل‌ها کردند.

دهه 1990 شاهد ظهور روش‌های فرا ابتکاری برای حل RCPSP بود. تکنیک‌هایی مانند جستجوی ممنوع، الگوریتم‌های ژنتیک و بهینه‌سازی کلونی مورچه‌ها برای رسیدگی به نمونه‌های بزرگ‌تر و پیچیده‌تر مسئله به کار گرفته شدند. این روش‌ها به محققان اجازه دادند فضای راه‌حل را به‌طور مؤثرتری بررسی کنند و راه‌حل‌های با کیفیت بالا را در زمان‌های محاسباتی معقول پیدا کنند.

در دهه 2000، محققان شروع به بررسی گسترش‌ها و انواع مختلف RCPSP برای رسیدگی به پیچیدگی‌های دنیای واقعی کردند. این موارد شامل RCPSP چند حالته (جایی که فعالیت‌ها می‌توانند در حالت‌های مختلف با نیازهای منابع متفاوت انجام شوند)، RCPSP تصادفی (جایی که مدت زمان فعالیت‌ها نامشخص است) و RCPSP با پنجره‌های زمانی بود. این گسترش‌ها مسئله را کاربردی‌تر کردند اما پیچیدگی آن را نیز افزایش دادند.

در طول دهه‌های 2000 و 2010، RCPSPدر کاربردهای صنعتی، به‌ویژه در ساخت‌وساز، تولید و مدیریت پروژه‌های نرم‌افزاری، مورد توجه قرار گرفت. پیشرفت‌ها در قدرت محاسباتی و توسعه ابزارهای نرم‌افزاری تخصصی، اجرای راه‌حل‌های RCPSP را در پروژه‌های واقعی ممکن ساخت. این دوره همچنین شاهد ادغام RCPSP با سایر تکنیک‌های بهینه‌سازی و سیستم‌های پشتیبانی تصمیم َشد.

---

[1] Minimum Spanning Tree

[2] Latest Finish Time

در سال‌های اخیر، تحقیقات روی RCPSP بر ترکیب روش‌های بهینه‌سازی سنتی با تکنیک‌های یادگیری ماشین و هوش‌مصنوعی متمرکز شده‌است. روش‌های ترکیبی که از نقاط قوت الگوریتم‌های دقیق و فراابتکارها استفاده می‌کنند، محبوبیت یافته‌اند. علاوه بر این، ظهور داده‌های بزرگ و رایانش ابری امکان‌های جدیدی برای حل نمونه‌های بزرگ‌مقیاس RCPSPباز کرده است. محققان همچنین در حال بررسی زمان‌بندی پویا و بلادرنگ برای تطبیق با شرایط متغیر پروژه هستند.

RCPSPتأثیر عمیقی بر روش‌های مدیریت پروژه داشته است. با ارائه چارچوبی برای بهینه‌سازی تخصیص منابع و زمان‌بندی، به سازمان‌ها کمک کرده است تا هزینه‌ها را کاهش دهند، تأخیرها را به حداقل برسانند و کارایی را بهبود بخشند. این مسئله همچنان کانون تحقیقات است و تلاش‌های مداوم برای رسیدگی به چالش‌های آن و گسترش کاربرد آن انجام‌می‌شود.

تاریخچه RCPSP بازتابی از تکامل تحقیق در عملیات و مدیریت پروژه در طول چند دهه گذشته‌است. از ریشه‌های آن در دهه 1950 تا وضعیت فعلی آن به‌عنوان یک مسئله پیچیده و به‌طور گسترده مطالعه‌شده، RCPSPباعث نوآوری در تکنیک‌های بهینه‌سازی و کاربردهای عملی شده‌است. با پیچیده‌تر شدن پروژه‌ها و سخت‌تر شدن محدودیت‌های منابع، RCPSPهمچنان به‌عنوان یک حوزه تحقیقاتی و توسعه‌ای حیاتی در سال‌های آینده باقی خواهد ماند.

RCPSP به‌دلیل نیاز به تعادل چندین محدودیت به‌طور همزمان پیچیده‌است. به‌عنوان مثال، برخی از فعالیت‌ها ممکن است برای منابع مشابه رقابت کنند، که منجر به درگیری‌های احتمالی می‌شود که باید حل شود. علاوه بر این، فعالیت‌ها وابستگی‌هایی دارند، به این معنی که ترتیب برنامه ریزی وظایف مهم است. حل RCPSP اغلب مستلزم یافتن مبادله بهینه بین ترتیب فعالیت‌ها و تخصیص کارآمد منابع محدود است. رویکردهای سنتی برای حل این مشکل شامل الگوریتم‌های دقیق مانند برنامه‌نویسی شاخه و کران و پویا و همچنین روش‌های ابتکاری یا فرا ابتکاری مانند الگوریتم‌های ژنتیک و بازپخت شبیه‌سازی شده‌است که برای نمونه‌های بزرگ‌تر یا پیچیده‌تر از مسئله استفاده می‌شود.

RCPSP کاربردهای مهم دنیای واقعی در زمینه‌هایی مانند ساخت و ساز، توسعه نرم‌افزار و ساخت دارد، جایی که مدیران پروژه باید محدودیت‌های منابع و محدودیت‌های زمانی را متعادل کنند. حل کارآمد RCPSP منجر به استفاده بهتر از منابع، صرفه‌جویی در هزینه و

تحویل سریع‌تر پروژه می‌شود. با این حال یک مسئله NP-hard است، به این معنی که با افزایش اندازه مسئله (با کارها و منابع بیشتر)، حل دقیق آن از نظر محاسباتی گران می‌شود. در نتیجه، روش‌های اکتشافی که راه‌حل‌های نزدیک به بهینه را در یک زمان معقول ارائه می‌کنند، اغلب در عمل مورد علاقه هستند.

2-2-1 روش‌های سنتی حل مسائل برنامه ریزی پروژه با محدودیت منابع

مشکل زمان‌بندی پروژه با محدودیت منابع (RCPSP) شامل زمان‌بندی مجموعه‌ای از فعالیت‌های مرتبط با یکدیگر در عین رعایت محدودیت‌های اولویت (ترتیب) و ظرفیت‌های منابع محدود است. از آنجایی که RCPSP یک NP-hard است، روش‌های حل سنتی به طور کلی به دو دسته تقسیم می‌شوند: روش‌های دقیق (که راه‌حل بهینه را تضمین می‌کنند اما می‌توانند محاسباتی فشرده باشند) و روش‌های اکتشافی یا سازنده (که سریع‌تر هستند اما ممکن است همیشه جواب بهینه را ارائه ندهند). الگوریتم‌های قدیمی حل مسئله RCPSP عبارتند از:

1. شعبه و محدود: این الگوریتم با اتخاذ تصمیمات متوالی (مانند اختصاص زمان شروع به فعالیت ها) و "شاخه بندی" فضای راه حل، به طور سیستماتیک تمام زمان بندی های ممکن را بررسی می کند. در هر گره درخت جستجو، یک کران پایینی روی هدف (به عنوان مثال، طول پروژه) محاسبه می شود. اگر محدوده یک شاخه از بهترین راه حل بیشتر شود، آن شاخه هرس می شود. در حالی که این روش یک راه حل بهینه را تضمین می کند، با افزایش لندازه مسئله از نظر محاسباتی سنگین و پیچیده می شود.
1. برنامه نویسی عدد صحیح (IP[1]):RCPSP را می توان به عنوان یک مدل برنامه ریزی خطی عدد صحیح فرموله کرد که در آن متغیرهای تصمیم نشان دهنده زمان شروع (یا ترتیب) فعالیت ها هستند و محدودیت ها تضمین می کنند که هم محدودیت های منابع و هم روابط اولویت برآورده می شوند. حل کننده های پیشرفته با استفاده از تکنیک هایی مانند شاخه و برش می توانند این مدل ها را برای نمونه های کوچک تا متوسط به طور بهینه حل کنند. با این حال، با افزایش تعداد فعالیت‌ها، پیچیدگی فرمول IP می‌تواند این رویکرد را غیرعملی کند.
2. برنامه نویسی پویا: در تئوری، برنامه‌نویسی پویا می‌تواند RCPSP را به مشکلات فرعی تجزیه کند و آنها را به صورت بازگشتی حل کند. با این حال، به دلیل "نفرین ابعاد" (به عنوان مثال،

[1] Integer Programming

رشـد تصـاعدی فضـای حالت)، این روش به طور کلی به نمونه های بسـیار کوچک RCPSP محدود می شود.

3-2-1 روش‌های ابتکاری حل مسائل برنامه ریزی پروژه با محدودیت منابع

این الگوریتم ها بر اساس قوانینی مانند:
کوتاه‌ترین زمان پردازش (SPT[1]): اولویت‌بندی فعالیت‌هایی با مدت زمان کوتاه‌تر.
زودترین زمان شروع (EST[2]): اولویت بندی فعالیت هایی که می توانند زودتر شروع شوند.
حداکثر تقاضای منابع: اولویت بندی فعالیت هایی که از منابع مهم تری استفاده می کنند.
پس از تعیین اولویت ها، فعالیت ها به صـورت متوالی برنامه ریزی می شـوند. ایده این اسـت که بدون جستجوی جامع در فضای راه حل، یک برنامه زمان بندی عملی بسازند.
این الگوریتم ها عبارتند از:

- طرح تولید برنامه سریال (SSGS[3]):در این رویکرد، فعالیت ها یک به یک به دنبال یک ترتیب از پیش تعیین شـده (اغلب برگرفته از قوانین اولویت) در برنامه قرار می گیرند. برای هر فعالیت، الگوریتم اولین شـکاف زمانی را پیدا می کند که در آن هر دو محدودیت منبع و اولویت برآورده می شوند.
- طرح تولید برنامه موازی (PSGS[4]):بر خلاف طرح سریال، PSGS در طول زمان پیشرفت می‌کند و در هر واحد زمانی، تمام فعالیت‌های موجود را به طور همزمان برنامه‌ریزی می‌کند – تا سـقف تحمیل شـده توسـط در دسترس بودن منابع. این روش گاهی اوقات می تواند به زمان بندی های متفاوتی منجر شـود و ممکن اسـت از منابع موجود بهتر بهره برداری کند. یک الگوریتم اکتشـافی سـاده که در آن فعالیت‌ها در فهرسـتی که بر اسـاس قوانین اولویت‌بندی مرتب شـده‌اند نگهداری می‌شـوند. الگوریتم به طور مکرر فعالیت بعدی را از لیسـت انتخاب می کند و آن را در اولین زمان ممکن برنامه ریزی می کند که هیچ محدودیتی را نقض نکند.
- جسـتجوی محلی و روش های بهبود:با شـروع از یک زمان‌بندی اولیه امکان‌پذیر (اغلب توسـط یکی از روش‌های اکتشـافی بالا ایجاد می‌شـود)، روش‌های جسـتجوی محلی با ایجاد تغییرات کوچک، برنامه‌های "همسـایه" را بررسـی می‌کنند – مانند تعویض ترتیب دو فعالیت یا تنظیم

[1] Shortest Processing Time
[2] Earliest Start Time
[3] Serial Scheduling Generation
[4] Parallel Scheduling Generation

زمان شـروع. این فرآیند به طور مکرر ادامه می یابد تا برنامه ای با عملکرد بهبود یافته پیدا شـود (به عنوان مثال، مدت زمان کوتاه تر). این روش ها به طور مکرر برنامه فعلی را تغییر می دهند (مثلاً با مرتب کردن مجدد فعالیت ها یا تغییر زمان شـــروع آنها) تا کیفیت کلی برنامه را بهبود بخشند. اگرچه آنها یافتن بهینه جهانی را تضمین نمی کنند، اما می توانند راه حل اولیه را در یک زمان نسبتا کوتاه به طور قابل توجهی بهبود بخشند. روش‌های دقیق (مانند برنامه‌نویسی شاخه و کران و عدد صـحیح) راه‌حل‌های بهینه را ارائه می‌دهند، اما عموماً به دلیل هزینه محاسـباتی، تنها برای مسـائل کوچک یا متوسـط مناسـب هسـتند. روش‌های اکتشـافی و سـازنده (مانند زمان‌بندی مبتنی بر قوانین اولویت و طرح‌های تولید زمان‌بندی) راه‌حل‌های سـریع‌تر و امکان‌پذیری را ارائه می‌کنند که برای مســـائل بزرگ‌تر به خوبی کار می‌کنند، هرچند بدون تضـــمین بهینه‌ســـازی تکنیک‌های جستجوی محلی می‌توانند این راه‌حل‌های اکتشافی را با کاوش در همسایگی راه‌حل برای بهبودهای بالقوه اصلاح کنند.

4-2-1 الگوریتم ژنتیک

مشـــکل زمان‌بندی پروژه با محدودیت منابع (RCPSP) شـــامل زمان‌بندی فعالیت‌های پروژه با رعایت روابط تقدم و دسترسی محدود به منابع، با هدف به حداقل رساندن طول پروژه است. با توجه به ماهیت NP-Hard آن، راه‌حل‌های دقیق برای نمونه‌های بزرگ از نظر محاســـباتی غیرممکن می‌شـــوند و رویکردهای اکتشـــافی و فراابتکاری، مانند الگوریتم‌های ژنتیک (GAs[1])، ابزار ارزشـــمندی برای یافتن راه‌حل‌های تقریباً بهینه هستند.

الگوریتم ژنتیک از اصـول انتخاب طبیعی و ژنتیک الهام گرفته شـده اسـت. در زمینه RCPSP، یک پیاده سازی GA معمولی شامل اجزای زیر است:

1. نمایش کروموزوم: هر کروموزوم یک راه حل بالقوه را نشان می دهد که اغلب به عنوان یک توالی فعالیت یا لیست اولویت کدگذاری می شود که ترتیب اجرای کار را دیکته می کند.

3. جمعیت اولیه: مجموعه ای از راه حل های اولیه ایجاد می شـود که می تواند به صـورت تصـادفی یا با اسـتفاده از روش های اکتشـافی برای اطمینان از امکان سـنجی در مورد اولویت و محدودیت های منابع ایجاد شود.

[1] Genetic Algorithms

4. تابع Fitness: این تابع هر کروموزوم را بر اســـاس طول پروژه ارزیابی می کند، با مدت زمان کوتاه تر که نشان دهنده تناسب اندام بهتر است.

5. انتخاب: تکنیک هایی مانند انتخاب چرخ رولت یا انتخاب مســـابقات برای انتخاب کروموزوم های والد برای تولید مثل استفاده می شود و به نفع افرادی است که نمرات تناسب اندام بالاتری دارند.

6. متقاطع (بازترکیب): کروموزوم های والدین برای تولید فرزندان با اســـتفاده از عملگرهایی مانند متقاطع یک نقطه ای یا دو نقطه ای ترکیب می شـــوند و در عین حال امکان پذیری را حفظ می کنند.

7. جهش: این تنوع را با ایجاد تغییرات تصـــادفی در کروموزوم های فرزندان، مانند تعویض دو فعالیت، برای کشـف مناطق جدید فضـای محلول و جلوگیری از همگرایی زودرس، معرفی می کند.

8. جایگزینی: نسـل جدید کروموزوم ها جایگزین کروموزوم های قدیمی می شـوند که اغلب برای حفظ بهترین راه حل های یافت شده، نخبگی را در خود جای می دهند.

از طریق کاربرد تکراری این مراحل، GAها راه‌حل‌ها را به ســـمت زمان‌بندی بهینه یا نزدیک به بهینه تکامل می‌دهند. مطالعات اثربخشی GAs را در حل RCPSP با پیمایش موثر فضاهای جستجوی بزرگ و پیچیده برای شـــناســـایی برنامه‌های با کیفیت بالا که هم اولویت و هم محدودیت‌های منابع را رعایت می‌کنند، نشان داده‌اند.

به طور خلاصـــه، الگوریتم‌های ژنتیک یک رویکرد قوی و انعطاف‌پذیر برای مقابله با RCPSP ارائه می‌دهند، که تعادلی بین کیفیت راه‌حل و کارایی محاســـباتی، به ویژه در ســـناریوهای زمان‌بندی پروژه در مقیاس بزرگ و پیچیده ارائه می‌دهد.

5-2-1 روش GPPH

برنامه نویسی ژنتیکی Hyper-Heuristics (GPHH) یک رویکرد پویا برای این چالش ارائه می دهد. در GPHH، برنامه نویسی ژنتیکی برای تکامل اکتشافی های جدید یا قوانین اولویت با ترکیب اکتشافی های مختلف سطح پایین استفاده می شود. این فرآیند تکاملی، کشف استراتژی‌های زمان‌بندی مؤثر متناسب

با نمونه‌های مشکل خاص را امکان‌پذیر می‌سازد. اکتشافی تکامل یافته می تواند با محیط های مختلف پروژه سازگار شود و اغلب از قوانین طراحی شده به صورت دستی بهتر عمل می کند.

یکی از کاربردهای قابل توجه GPHH در RCPSP طراحی خودکار قوانین اولویت است. محققان از GPHH برای تکامل قوانینی استفاده کرده‌اند که چندین ویژگی پروژه را در نظر می‌گیرند و منجر به تصمیم‌گیری‌های زمان‌بندی کارآمدتر می‌شوند. به عنوان مثال، یک مطالعه نشان داد که GPHH می تواند به طور موثر قوانین اولویتی را ایجاد کند که در سناریوهای مختلف RCPSP از اکتشافی سنتی بهتر عمل می کند.

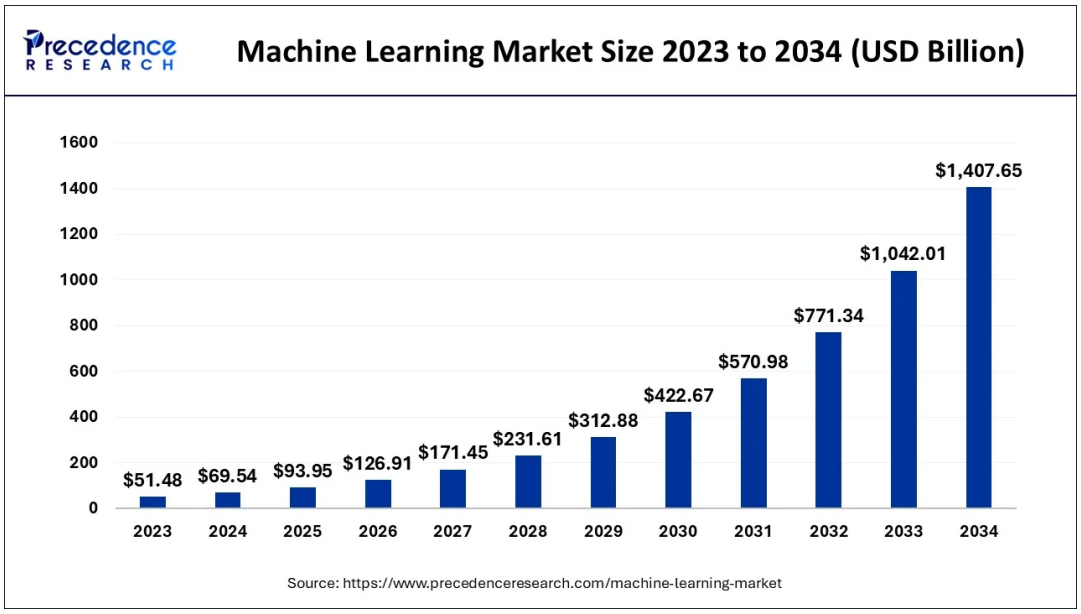
به طور خلاصه، GPHH یک چارچوب انعطاف‌پذیر و خودکار برای توسعه اکتشافی در RCPSP فراهم می‌کند، که توانایی رسیدگی به مشکلات پیچیده زمان‌بندی را با تکامل قوانینی که برای محدودیت‌ها و اهداف پروژه مناسب‌تر هستند، افزایش می‌دهد. الگوریتم ژنتیک GPPH به عنوان بهترین الگوریتم یافته شده در طی سالیان گذشته می‌باشد.

1-3 یادگیری ماشین

یادگیری ماشینی (ML) زیرمجموعه‌ای از هوش‌مصنوعی (AI) است که بر توسعه الگوریتم‌هایی متمرکز است که رایانه‌ها را قادر می‌سازد الگوهایی را از داده‌ها یاد بگیرند و بدون برنامه‌ریزی صریح تصمیم‌گیری یا پیش‌بینی کنند. با استفاده از تکنیک‌های آماری، سیستم‌های یادگیری ماشینی از مجموعه داده‌های بزرگ یاد می‌گیرند تا عملکرد خود را در طول زمان بهبود بخشند. یادگیری نظارت‌شده، که در آن مدل بر روی داده‌های برچسب دار آموزش داده‌می‌شود، و یادگیری بدون نظارت، که در آن مدل الگوها را در داده‌های بدون برچسب شناسایی می‌کند، دو رویکرد اصلی هستند. همچنین یادگیری تقویتی وجود دارد که از یک سیستم مبتنی بر پاداش برای مدل برای یادگیری رفتارهای بهینه از طریق آزمون و خطا استفاده می‌کند.

در هسته خود، یادگیری ماشین مدل‌هایی را برای یافتن همبستگی‌ها، روندها و بینش‌ها از مجموعه داده‌های بزرگ و پیچیده ایجاد می‌کند که برای الگوریتم‌های سنتی بسیار چالش‌برانگیز است. به‌عنوان مثال، در وظایف یادگیری تحت نظارت مانند تشخیص تصویر،

مدل‌ها بر روی تصاویر برچسب دار آموزش می‌بینند و یاد می‌گیرند که تصاویر جدید را بر اساس آنچه دیده‌اند طبقه‌بندی کنند. در یادگیری بدون نظارت، وظایفی مانند خوشه‌بندی شامل الگوریتم گروه بندی نقاط داده با ویژگی‌های مشابه بدون دسته‌های از پیش تعریف شده‌است. یادگیری ماشینی به‌ویژه در زمینه‌هایی مانند مراقبت‌های بهداشتی، مالی و بازاریابی مفید بوده است، جایی که می‌تواند به پیش‌بینی، تحلیل ریسک و مدل سازی رفتار مشتری کمک کند.



پیشرفت‌ها در یادگیری ماشینی، به‌ویژه در یادگیری عمیق (زیرمجموعه‌ای شامل شبکه‌های عصبی)، پیشرفت‌هایی را در کارهایی مانند پردازش زبان طبیعی، تشخیص صدا و حتی رانندگی مستقل فراهم کرده است. سیستم‌های مدرن می‌توانند حجم زیادی از داده‌ها را در زمان واقعی پردازش کنند، و آنها را به‌شدت با محیط‌های پویا سازگار می‌سازد. این قابلیت‌ها یادگیری ماشینی را به ابزاری حیاتی برای نوآوری در فناوری، اتوماسیون و فرآیندهای تصمیم گیری تبدیل می‌کند.

یادگیری ماشینی (ML) را می‌توان به‌طور کلی به سه نوع اصلی تقسیم کرد و هر یک از این انواع رویکردهای متفاوتی برای الگوریتم‌های آموزشی دارند و بر اساس ماهیت داده‌ها و کار، اهداف مشخصی را دنبال می‌کنند.

1. یادگیری با نظارت: در یادگیری نظارت‌شده، مدل بر روی داده‌های برچسب دار آموزش داده‌می‌شود، جایی که ورودی و خروجی (پاسخ صحیح) هر دو مشخص است. الگوریتم یاد می‌گیرد که با به حداقل رساندن خطا بین پیش بینی‌های خود و نتایج واقعی، ورودی‌ها را به خروجی‌های صحیح نگاشت کند. وظایف یادگیری تحت نظارت

متداول شامل طبقه‌بندی (به‌عنوان مثال، تشخیص هرزنامه بودن یا نبودن ایمیل) و پسرفت (مثلاً پیش‌بینی قیمت خانه) است. الگوریتم‌هایی مانند رگرسیون خطی، ماشین‌های بردار پشتیبانی (SVM[1]) و جنگل‌های تصادفی اغلب در یادگیری نظارت‌شده استفاده می‌شوند.

2. یادگیری بدون نظارت: برخلاف یادگیری تحت نظارت، یادگیری بدون نظارت با داده‌های بدون برچسب کار می‌کند، به این معنی که الگوریتم باید الگوها یا روابط پنهان را در داده‌های ورودی بدون دستورالعمل صریح پیدا کند. اغلب برای کارهایی مانند خوشه‌بندی (گروه بندی نقاط داده مشابه با هم) و کاهش ابعاد (کاهش تعداد ویژگی‌ها با حفظ اطلاعات ضروری) استفاده می‌شود. این نوع یادگیری در تجزیه و تحلیل داده‌های اکتشافی، جایی که ساختار داده‌ها به‌خوبی درک نشده است، ارزشمند است.
3. یادگیری تقویتی: در یادگیری تقویتی، یک عامل از طریق تعامل با محیط خود و دریافت بازخورد به‌صورت پاداش یا جریمه یاد می‌گیرد. هدف، به حداکثر رساندن پاداش تجمعی با یادگیری یک خط‌مشی یا توالی اقدامات بهینه است. این نوع یادگیری به‌طور گسترده در روباتیک، بازی و سیستم‌های مستقل استفاده می‌شود، جایی که عامل باید در یک محیط پویا تصمیم بگیرد. تکنیک‌هایی مانند Q-learning و یادگیری تقویتی عمیق معمولاً در این زمینه استفاده می‌شوند.

هر یک از این انواع یادگیری ماشینی دارای نقاط قوت منحصر به فردی است و برای مشکلات مختلف مناسب است، خواه پیش‌بینی کردن، پیدا کردن الگوهای پنهان یا یادگیری از تعامل با محیط باشد.

[1] Support Vector Machine

1-4 یادگیری تقویتی

یادگیری تقویتی (RL[1]) نوعی از یادگیری ماشینی است که در آن یک عامل می‌آموزد که با تعامل با یک محیط و دریافت بازخورد به‌شکل پاداش یا جریمه تصمیم گیری کند. هدف عامل به حداکثر رساندن پاداش تجمعی در طول زمان با یافتن بهترین توالی از اقدامات است که اغلب به‌عنوان یک سیاست از آن یاد می‌شود. برخلاف یادگیری تحت نظارت، که در آن پاسخ‌های صحیح ارائه می‌شود، RL بر آزمون و خطا تأکید می‌کند. عامل استراتژی‌های مختلف را بررسی می‌کند و از بازخورد محیط برای بهبود فرآیند تصمیم گیری خود استفاده می‌کند. این باعث می‌شود RL برای کارهایی که نیاز به بهینه‌سازی نتایج بلندمدت در محیط‌های پویا و نامطمئن وجود دارد، مناسب باشد.

هسته اصلی یادگیری تقویتی، فرآیند تصمیم گیری مارکوف (MDP[2]) است که محیط را بر حسب حالت‌ها، اقدامات، پاداش‌ها و انتقال‌ها تعریف می‌کند. در هر حالت، عامل اقدامی را انتخاب می‌کند، پاداشی از محیط دریافت می‌کند و به‌حالت جدیدی منتقل می‌شود. با گذشت زمان، عامل مدلی از رفتار محیط می‌سازد و از الگوریتم‌هایی مانند Q-Learning و PPO برای یافتن بهترین عملکرد برای هر حالت استفاده می‌کند. یکی از ویژگی‌های کلیدی RL، مبادله بین اکتشاف (آزمودن اقدامات جدید برای کشف اثرات آنها) و بهره‌برداری (استفاده از اقدامات شناخته شده که بالاترین پاداش را به‌همراه دارد) است. ایجاد تعادل در این مبادله برای یادگیری استراتژی‌های بهینه بسیار مهم است.

یادگیری تقویتی به‌طور موفقیت آمیزی در زمینه‌های مختلف از جمله روباتیک، بازی کردن و سیستم‌های خودمختار به کار گرفته شده‌است. RL همچنین به‌طور گسترده در سیستم‌های توصیه، تجارت مالی و بهینه‌سازی منابع استفاده می‌شود، جایی که تصمیم گیری پویا ضروری است. توانایی RL برای انطباق با محیط‌های پیچیده و در حال تغییر، آن را به ابزاری قدرتمند برای حل مسائل دنیای واقعی که شامل تصمیم گیری متوالی است تبدیل می‌کند.

تاریخ یادگیری تقویتی دو رشته اصلی دارد که هر دو طولانی و غنی هستند و به‌طور مستقل دنبال شده‌اند تا اینکه در یادگیری تقویتی مدرن به‌هم پیوستند. یکی از این رشته‌ها به یادگیری از طریق آزمون و خطا مربوط می‌شود و در روانشناسی یادگیری حیوانات آغاز شد.

---

[1] Reinforcement Learning
[2] Markov Decision Process

این رشته در برخی از نخستین کارها در هوش‌مصنوعی وجود دارد و منجر به احیای یادگیری تقویتی در اوایل دهه 1980 شد. رشته دیگر به مسئله کنترل بهینه و حل آن با استفاده از توابع ارزش و برنامه‌ریزی دینامیک مربوط می‌شود. عمدتاً، این رشته شامل یادگیری نبود. اگرچه این دو رشته عمدتاً مستقل بوده‌اند، استثنائات حول یک رشته سوم، کمتر مشخص، مربوط به روش‌های تفاوت زمانی می‌چرخد که در مثال دوز بازی در این فصل استفاده شده‌است. هر سه رشته در اواخر دهه 1980 با هم جمع شدند تا زمینه مدرن یادگیری تقویتی را تولید کنند که ما در این کتاب ارائه می‌دهیم.

رشته‌ای که بر یادگیری آزمون و خطا تمرکز دارد، همان رشته‌ای است که ما بیشتر با آن آشنا هستیم و درباره آن در این تاریخچه مختصر بیشتر صحبت خواهیم کرد. اما قبل از آن، به‌طور مختصر به رشته کنترل بهینه اشاره می‌کنیم.

اصطلاح "کنترل بهینه" در اواخر دهه 1950 برای توصیف مشکل طراحی یک کنترل‌کننده برای حداقل کردن اندازه‌گیری رفتار یک سیستم دینامیکی در طول زمان به کار رفت. یکی از رویکردهای این مشکل در اواسط دهه 1950 توسط ریچارد بلمن و دیگران با گسترش نظریه‌ای از قرن نوزدهم توسط همیلتون و جاکوبی توسعه یافت. این رویکرد از مفاهیم حالت یک سیستم دینامیکی و تابع ارزش یا "تابع بازگشت بهینه" برای تعریف یک معادله تابعی استفاده می‌کند که اکنون اغلب معادله بلمن نامیده می‌شود. کلاس روش‌ها برای حل مشکلات کنترل بهینه با حل این معادله به‌عنوان برنامه‌ریزی دینامیک شناخته شد) بلمن، همچنین نسخه تصادفی گسسته‌ای از مشکل کنترل بهینه را معرفی کرد که به‌عنوان فرآیندهای تصمیم‌گیری مارکوف (MDP) شناخته می‌شود، و ران هاوارد (1960) روش تکرار سیاست را برایMDP ها طراحی کرد. همه این‌ها عناصر اساسی نظریه و الگوریتم‌های یادگیری تقویتی مدرن هستند.

1-4-1 برنامه ریزی پویا

مفهوم برنامهریزی پویا به‌طور رسمی توسط ریچارد بلمن، ریاضی‌دان آمریکایی، در دهه ۱۹۵۰ معرفی شد. بلمن در مؤسسه رند(RAND Corporation) ، یک اندیشکده متمرکز بر دفاع و برنامهریزی استراتژیک در دوران جنگ سرد، کار می‌کرد. او این روش را برای حل مسائل بهینهسازی شامل تصمیمگیریهای متوالی، مانند تخصیص منابع و زمان‌بندی، توسعه داد.

بلمن در سال ۱۹۵۳ عبارت "برنامهریزی پویا" را ابداع کرد. او واژه "پویا" را برای نشان دادن ماهیت متغیر با زمان مسائلی که به آنها می‌پرداخت انتخاب کرد و "برنامهریزی" به استفاده از تکنیک‌های بهینهسازی ریاضی اشاره داشت، نه برنامه‌نویسی کامپیوتری به‌معنای امروزی. بلمن بعدها شوخی کرد که این عبارت را تا حدی برای تأثیرگذاری بیشتر و جلوگیری از انتقادات مافوق‌های خود انتخاب کرده است.

یکی از پایه‌های برنامهریزی پویا، **اصل بهینگی** است که بلمن در سال ۱۹۵۷ آن را بیان کرد. این اصل بیان می‌کند که یک راه‌حل بهینه برای یک مسئله را می‌توان از راه‌حلهای بهینه زیر مسائل آن ساخت. این اصل امکان حل بازگشتی مسائل را فراهم می‌کند و برنامهریزی پویا را به‌ویژه برای مسائل با زیر مسائل همپوشان و ساختار بهینه مؤثر می‌سازد.

در سال‌های اولیه، برنامهریزی پویا در مسائل اقتصادی، لجستیک و استراتژی نظامی به کار گرفته شد. برای مثال، از آن برای بهینهسازی مدیریت موجودی، زمان‌بندی تولید و سیستم‌های هدایت موشک استفاده شد. این کاربردها نشان‌دهنده انعطاف‌پذیری برنامهریزی پویا در فرآیندهای تصمیم‌گیری چندمرحله‌ای بود.

در دهه‌های ۱۹۶۰ و ۱۹۷۰، محققان شروع به فرموله کردن الگوریتم‌های برنامهریزی پویا و بررسی ویژگی‌های محاسباتی آنها کردند. از جمله پیشرفت‌های کلیدی، معرفی **مموایزیشن** (ذخیره نتایج میانی برای جلوگیری از محاسبات تکراری) و استفاده از **جدول‌سازی** (پر کردن جداول به‌صورت تکراری برای حل مسائل) بود. این تکنیک‌ها کارایی الگوریتم‌های برنامهریزی پویا را بهبود بخشیدند و آنها را برای کاربردهای واقعی عملی‌تر کردند.

با ظهور علوم کامپیوتر به‌عنوان یک رشته در اواسط قرن بیستم، برنامهریزی پویا به یک ابزار اساسی در طراحی الگوریتم‌ها تبدیل شد. از آن برای حل مسائلی مانند ترازسازی دنباله‌ها، الگوریتم‌های کوتاه‌ترین مسیر و پیمایش گراف استفاده شد. توسعه زبان‌های برنامه‌نویسی و منابع محاسباتی نیز امکان پیاده‌سازی الگوریتم‌های برنامهریزی پویا روی کامپیوترها را فراهم کرد.

یکی از مسائل کلاسیک حل شده با برنامهریزی پویا، **مسئله کوله‌پشتی** است که شامل انتخاب زیرمجموعه‌ای از اقلام با حداکثر ارزش بدون محدودیت وزن است. این مسئله، همراه با مسائل دیگری مانند مسئله فروشنده دوره‌گرد و ضرب زنجیرهای ماتریس‌ها،

به‌عنوان معیارهایی برای آزمایش و بهبود تکنیک‌های برنامهریزی پویا مورد استفاده قرار گرفتند.

در دهه‌های ۱۹۷۰ و ۱۹۸۰، برنامهریزی پویا به یک ابزار کلیدی در مدل‌سازی اقتصادی تبدیل شد. اقتصاددانان از آن برای مطالعه تصمیمگیریهای بینزمانی، مانند رفتار مصرف-پس‌انداز، استراتژی‌های سرمایه‌گذاری و تخصیص منابع در طول زمان استفاده کردند. کار اقتصاددلنانی مانند رابرت لوکاس و ادوارد پرس کات برنامهریزی پویا را در نظریه اقتصاد کلان محبوب‌تر کرد.

در دهه‌های ۱۹۸۰ و ۱۹۹۰، برنامهریزی پویا کاربردهای جدیدی در هوش‌مصنوعی و یادگیری ماشین پیدا کرد. این روش به یک جزء اصلی در **یادگیری تقویتی** تبدیل شد، جایی که عامل‌ها با تعامل با محیط، سیاستهای بهینه را یاد می‌گیرند. الگوریتم‌هایی مانند تکرار ارزش و تکرار سیاست، که ریشه در برنامهریزی پویا دارند، برای حل فرآیندهای تصمیم‌گیری مارکوف (MDPs) استفاده می‌شوند.

با وجود قدرت برنامهریزی پویا، این روش با چالش‌هایی در رابطه با پیچیدگی محاسباتی مواجه است. مسائل با فضای حالت بزرگ یا ابعاد بالا میتوانند منجر به "نفرین ابعاد" شوند، جایی که هزینه محاسباتی به‌طور نمایی با اندازه مسئله افزایش می‌یابد. محققان تکنیک‌های تقریبی، مانند برنامهریزی پویای تقریبی(ADP[1]) ، را برای مقابله با این محدودیت‌ها توسعه داده‌اند.

امروزه، برنامهریزی پویا در طیف گسترده‌ای از حوزه‌ها استفاده می‌شود. در بیوانفورماتیک، از آن برای ترازسازی دنباله‌ها (مانند الگوریتم نیدلمن-و ان) استفاده می‌شود. در مالی، برای بهینهسازی سبد سهام و قیمتگذاری اختیارات به کار می‌رود. در رباتیک، برنامهریزی پویا به برنامه‌ریزی مسیر و کنترل کمک می‌کند. انعطاف‌پذیری آن همچنان آن را به ابزاری ارزشمند برای حل مسائل دنیای واقعی تبدیل کرده است.

برنامهریزی پویا بخشی اساسی در برنامه‌های درسی علوم کامپیوتر و تحقیق در عملیات است. به‌عنوان یک تکنیک بنیادی برای حل مسئله آموزش داده‌می‌شود و اصول آن در برنامه‌نویسی رقابتی و مصاحبه‌های کدنویسی به کار می‌رود. کتاب‌هایی مانند *برنامه‌ریزی پویا* بلمن (۱۹۵۷) و *مقدمه‌ای بر الگوریتم‌ها* کورمن و همکاران به گسترش دانش برنامهریزی پویا کمک کرده‌اند.

[1] Approximation of Dynamic Programming

در طول سال‌ها، محققان گسترش‌ها و انواعی از برنامهریزی پویا را برای مقابله با چالش‌های خاص توسعه داده‌اند. از جمله این موارد می‌توان به برنامهریزی پویای تصادفی (برای مسائل با عدم قطعیت)، برنامهریزی پویای تفاضلی (برای مسائل کنترل پیوسته) و برنامهریزی پویای چندهدفه (برای مسائل با اهداف متضاد) اشاره کرد. برنامهریزی پویا ارتباط نزدیکی با سایر روش‌های بهینهسازی، مانند الگوریتم‌های حریصانه، تقسیم و حل و برنامهریزی خطی دارد. ماهیت بازگشتی و تمرکز آن بر ساختار بهینه، آن را به ابزاری مکمل در چشم‌انداز گسترده‌تر حل مسئله الگوریتمی تبدیل کرده است.

با وجود تاریخچه طولانی، برنامهریزی پویا همچنان یک حوزه فعال پژوهشی است. از جمله مسائل باز، توسعه الگوریتم‌های کارآمدتر برای مسائل با ابعاد بالا، ادغام برنامهریزی پویا با تکنیک‌های یادگیری ماشین و بررسی کاربردهای آن در حوزه‌های نوظهور مانند محاسبات کوانتومی است.

داستان برنامهریزی پویا بازتابی از تعامل بین ریاضیات، علوم کامپیوتر و کاربردهای دنیای واقعی است. کار بلمن نمونه‌ای از این است که چگونه بینش‌های نظری می‌توانند به ابزارهای عملی منجر شوند که صنایع را شکل می‌دهند و فناوری را پیش می‌برند . مشارکت‌های ریچارد بلمن در برنامهریزی پویا و بهینهسازی میراثی ماندگار از خود به‌جای گذاشته‌است. او جوایز متعددی از جمله مدال افتخار IEEE دریافت کرد و کار او همچنان الهام‌بخش نسل‌های جدیدی از محققان و متخصصان است.

با افزایش قدرت محاسباتی و ظهور چالش‌های جدید، برنامهریزی پویا احتمالاً به تکامل خود ادامه خواهد داد. ادغام آن با هوش‌مصنوعی، داده‌های بزرگ و محاسبات موازی نویدبخش گشودن افق‌های جدید برای حل مسائل پیچیده‌است.

تاریخچه برنامهریزی پویا گواهی بر قدرت تفکر ریاضی و تولنایی آن در تبدیل ایده‌های انتزاعی به راه‌حلهای عملی است. از خاستگاه آن در دوران جنگ سرد تا کاربردهای مدرن در هوش‌مصنوعی و فراتر از آن، برنامهریزی پویا به‌عنوان ابزاری جاودانه و ضروری برای مقابله با برخی از چالش برانگیزترین مسائل در علم و مهندسی ثابت شده‌است.

دو ویژگی کلیدی وجود دارد که یک مسئله را برای برنامه ریزی پویا مناسب می‌کند: زیرساخت بهینه و مسائل فرعی همپوشانی. زیرساخت بهینه به این معنی است که راه‌حل مشکل کلی را می‌توان از راه‌حل‌های زیر مشکلات آن ساخت، که مشخصه مشکلاتی مانند کوتاه‌ترین مسیرها یا کوله‌پشتی است. مسائل فرعی همپوشانی به این معنی است که همان

مسائل فرعی چندین بار در ساختار بازگشتی مشکل ظاهر می‌شوند. DP با حل هر زیر مسئله فقط یک‌بار و ذخیره نتیجه آن، از این امر استفاده می‌کند و آن را به یک جایگزین کارآمد برای الگوریتم‌های بازگشتی brute-force تبدیل می‌کند.
نمونه‌هایی از مسائلی که می‌توان با استفاده از برنامه نویسی پویا حل کرد شامل دنباله فیبوناچی، الگوریتم‌های کوتاه‌ترین مسیر و مسئله کوله پشتی است. در عمل، مسائل DP اغلب با استفاده از جداول یا شبکه‌ها برای ذخیره راه‌حل‌های زیر مسئله نشان داده می‌شوند و یک استراتژی رایج این است که با کوچک‌ترین مسائل فرعی شروع و به یکی از بزرگ‌ترها اضافه شود. این امر از محاسبه مجدد جلوگیری می‌کند و در بسیاری از موارد پیچیدگی زمانی را از نمایی به چند جمله‌ای کاهش می‌دهد.
معادله **بلمن** (Bellman Equation)یک مفهوم اساسی در برنامه‌ریزی پویا و یادگیری تقویتی است که توسط ریچارد بلمن معرفی شد. این معادله یک تجزیه بازگشتی برای مسائل تصمیم‌گیری ارائه می‌دهد و آن‌ها را به مسائل کوچک‌تر و قابل مدیریت‌تر تقسیم می‌کند. معادله بلمن برای حل مسائل بهینه‌سازی که شامل تصمیم‌گیری‌های متوالی در طول زمان هستند، به‌ویژه در فرآیندهای تصمیم‌گیری مارکوف(MDPs) ، بسیار مهم است.

2-4-1 معادله بلمن

معادله بلمن ارزش یک مسئله تصمیم‌گیری را در یک حالت خاص بر اساس ارزش حالت‌های ممکن آینده بیان می‌کند. این معادله اصل بهینگی را نشان‌می‌دهد که بیان می‌کند یک سیاست بهینه این ویژگی را دارد که بدون توجه به‌حالت اولیه و تصمیم اولیه، تصمیم‌های بعدی باید یک سیاست بهینه با توجه به حالتی که از تصمیم اولیه حاصل می‌شود، تشکیل دهند.

اجزای کلیدی این معادله عبارتند از:

1. **حالت :(s)** نمایشی از وضعیت یا محیط فعلی.
2. **عمل :(a)** تصمیم یا انتخابی که عامل در یک حالت خاص انجام می‌دهد.
3. **پاداش :(r)** بازخورد یا سود فوری که پس از انجام یک عمل در یک حالت دریافت می‌شود.

4. **تابع ارزش :(V (s))** ارزش مورد انتظار تجمعی (یا ســود) از بودن در یک حالت و دنبال کردن یک سیاست پس از آن.

5. **ســیاســت :(π)** یک اســتراتژی یا قاعده که عمل انجام‌شــده در هر حالت را تعیین می‌کند.

معادله بلمن برای تابع ارزش $V(s)$ در یک فرآیند تصمیم‌گیری مارکوف (MDP) به‌صورت زیر تعریف می‌شود:

$$V(s) = max\ a\ (R(s,a) + \gamma \sum s' P(s' \mid s,a)\ V(s'))$$

جایی که:

- $V(s)$: ارزش بودن در حالت s.
- $R(s, a)$: پاداش فوری برای انجام عمل a در حالت s
- γ برابر با فاکتور تخفیف (بین ۰ و ۱) که اهمیت پاداش‌های آینده را تعیین می‌کند.
- $P(s'|s, a)$: برابر با احتمال انتقال به‌حالت s' از حالت s پس از انجام عمل a/*ست*.
- $max\ a$:بیشینه‌سازی بر روی تمام اقدامات ممکن برای اطمینان از سیاست بهینه.

معادله بلمن بیان می‌کند که ارزش یک حالت ssبرابر است با بیشینه (بر روی تمام اقدامات ممکن) مجموع دو مورد زیر می‌باشد:

1. پاداش فوری $R(s, a)$ برای انجام عمل aدر حالت s
2. ارزش مورد انتظار تخفیف‌یافته حالت بعدی $s's'$، که با احتمال انتقال $P(s' \mid s, a)$ وزن‌دهی شده‌است.

معادله بهینگی بلمن

معادله بهینگی بلمن شــکل خاصــی از معادله بلمن اســت که تابع ارزش بهینه $V*(s)$ را تعریف می‌کند:

$$V(s')\ V(s) = amax\ (R(s,a) + \gamma s' \sum P(s' \mid s,a)\ V(s'))$$

این معادله اطمینان می‌دهد که تابع ارزش مربوط به سیاست بهینه است، که پاداش تجمعی را به حداکثر می‌رساند.

کاربردها

معادله بلمن به‌طور گسترده در موارد زیر استفاده می‌شود:

- **یادگیری تقویتی :**برای محاسبه توابع ارزش و استخراج سیاست‌های بهینه.
- **کنترل بهینه :**برای حل مسائل کنترل بهینه.
- **اقتصاد :**در مدل‌های تصمیم‌گیری در طول زمان.
- **تحقیق در عملیات :**برای مسائل تخصیص منابع و برنامه‌ریزی.

معادله بلمن یک رابطه بازگشتی است که یک مسئله تصمیم‌گیری را به مسائل کوچک‌تر تقسیم می‌کند. این معادله یکی از پایه‌های برنامه‌ریزی پویا و یادگیری تقویتی است و امکان محاسبه سیاست‌های بهینه را با حل تکراری تابع ارزش فراهم می‌کند. کلیت و زیبایی آن، معادله بلمن را به یک ابزار قدرتمند برای حل مسائل تصمیم‌گیری متوالی تبدیل کرده است.

1-5 یادگیری عمیق

یادگیری عمیق زیرشاخه‌ای از یادگیری ماشینی است که شامل آموزش شبکه‌های عصبی مصنوعی بر روی حجم وسیعی از داده‌ها می‌شود تا آنها را قادر به یادگیری الگوهای پیچیده و تصمیم گیری کند. این شبکه‌ها که از ساختار مغز انسان الهام گرفته‌شده‌اند، از لایه‌های به‌هم پیوسته‌ای از «نورون‌ها» تشکیل‌شده‌اند که برای پردازش و تبدیل داده‌ها با هم کار می‌کنند. برخلاف یادگیری ماشینی سنتی، که استخراج ویژگی اغلب به مداخله انسان نیاز دارد، شبکه‌های یادگیری عمیق قادر به شناسایی خودکار ویژگی‌های مرتبط در داده‌های خام هستند، که آنها را به‌ویژه برای مدیریت داده‌های بدون ساختار مانند تصاویر، صدا و متن مؤثر می‌سازد.

ریشه‌های شبکه‌های عصبی و یادگیری عمیق به دهه ۱۹۴۰ بازمی‌گردد، زمانی که محققان شروع به بررسی ایده ایجاد ماشین‌هایی کردند که بتوانند مغز انسان را تقلید کنند. در سال ۱۹۴۳، وارن مک‌کلاچ و وال‌تر پیتس مقاله‌ای seminal منتشر کردند که مدلی ساده‌شده از یک شبکه عصبی با استفاده از مدارهای الکتریکی را توصیف می‌کرد. کار آنها پایه‌هایی برای نورون‌های مصنوعی ایجاد کرد که می‌توانستند عملکردهای منطقی پایه را انجام دهند. در سال ۱۹۵۸، فرانک روزنبلات *پرسپترون* را معرفی کرد، الگوریتمی که برای کارهای طبقه‌بندی دودویی طراحی شده بود. پرسپترون یک شبکه عصبی تک‌لایه بود که می‌توانست

از داده‌ها یاد بگیرد و یکی از اولین مدل‌هایی بود که قادر به یادگیری نظارت‌شـــده بود. کار روزنبلات هیجان زیادی ایجاد کرد، اما پرسپترون محدودیت‌هایی داشت—فقط می‌توانست مسائل جدا پذیر خطی را حل کند.

محـدودیـت‌هـای پرســـپترون، کـه توســـط مـاروین مینســـکـی و ســـیمور پـاپرت در کتاب *پرسپترون‌ها* در سـال ۱۹۶۹ برجسـته شـد، منجر به دوره‌ای از شـک و تردید و کاهش بودجه برای تحقیقات شبکه‌های عصبی شد. این دوره، که به‌عنوان "زمستان هوش‌مصنوعی" شـناخته می‌شود، شاهد تغییر تمرکز به‌سمت هوش‌مصنوعی نمادین و سیسـتم‌های مبتنی بر قاعده بود که در آن زمان امیدوارکننده‌تر به نظر می‌رسیدند.

توسعه الگوریتم *پس‌انتشار* در دهه ۱۹۷۰ و ۱۹۸۰ نقطه عطفی بود. پس‌انتشار به شـبکه‌های عصـــبی اجازه می‌داد تا وزن‌های خود را با انتشـــار خطاها به عقب در شـــبکه به‌طور کارآمد تنظیم کنند. این نوآوری آموزش شبکه‌های چندلایه را ممکن ساخت که می‌توانستند مسائل پیچیده‌تر و غیرخطی را حل کنند. محققانی مانند جفری هینتون، دیوید روملهارت و رونلد ویلیامز نقش کلیدی در محبوبیت این رویکرد داشتند.

دهه ۱۹۸۰ شـاهد ظهور *اتصالی‌گرایی* بود، جنبشـی که بر اهمیت شـبکه‌های به‌هم‌پیوسـته از واحدهای سـاده (نورون‌ها) در مدل‌سـازی فرآیندهای شـناختی تأکید داشـت. انتشـار کتاب دو جلدی *پردازش توزیع‌شـــده موازی* در ســـال ۱۹۸۶ توســـط روملهارت، هینتون و دیگران به تثبیت پایه‌های نظری شبکه‌های عصبی کمک کرد و علاقه به این زمینه را دوباره برانگیخت.

علی‌رغم پیشـــرفت‌های دهه ۱۹۸۰، شـــبکه‌های عصـــبی در دهه ۱۹۹۰ با چالش‌هایی مواجه شـدند. آنها به مقدار زیادی داده و قدرت محاسـباتی نیاز داشـتند که در آن زمان به‌راحتی در دســـترس نبودند. علاوه بر این، روش‌های دیگر یادگیری ماشـــین، مانند ماشـــین‌های بردار پشتیبان(SVMs) ، به‌دلیل تضمین‌های نظری قوی و کارایی‌شان محبوبیت یافتند.

اصطلاح "یادگیری عمیق" در دهه ۲۰۰۰ شروع به جلب توجه کرد و به شبکه‌های عصبی با چندین لایه پنهان اشـاره داشـت. پیشـرفت‌های سـخت‌افزاری، به‌ویژه اسـتفاده ازGPU ها (واحدهای پردازش گرافیکی)، آموزش شـبکه‌های عمیق‌تر را ممکن سـاخت. محققانی مانند یان لکون، یوشوا بنجیو و جفری هینتون به پیشبرد مرزهای شبکه‌های عصبی ادامه دادند و معماری‌هایی مانند شبکه‌های عصبی کانولوشنال(CNN) [1] را برای تشخیص تصویر توسعه دادند.

[1] Convolutional Neural Network

دهه ۲۰۱۰ شـــاهد دســـتاوردهای انقلابی در یادگیری عمیق بود. در ســـال ۲۰۱۲، الکس کریژفسکی، ایلیا ساتسکور و جفری هینتون *الکس‌نت* را معرفی کردند، یک CNN عمیق که با اختلاف زیادی در رقابت ImageNet برنده شـــد. این پیروزی قدرت یادگیری عمیق را برای تشخیص تصویر نشان داد و باعث پذیرش گسترده آن شد. در همین زمان، مدل‌های یادگیری عمیق شروع به دستیابی به نتایج پیشرفته در تشخیص گفتار، پردازش زبان طبیعی و سایر حوزه‌ها کردند.

موفقیت یادگیری عمیق در دهه ۲۰۱۰ توســط در دســترس بودن مجموعه‌داده‌های عظیم)مانند (ImageNet و افزایش قدرتGPU [1]ها وTPU [2]ها (واحدهای پردازش تنســور) تقویت شـد. این منابع به محققان امکان دادند مدل‌های بزرگ‌تر و پیچیده‌تر را آموزش دهند که منجر به عملکرد بی‌سابقه در کارهایی مانند تشخیص اشیا، ترجمه ماشینی و بازی‌های فکری شد.

معرفی معماری *ترنسفورمر* در سال ۲۰۱۷ انقلابی در پردازش زبان طبیعی (NLP) ایجاد کرد. ترنســفورمرها، که از مکانیزم‌های خودتوجهی اســتفاده می‌کنند، به مدل‌هایی مانند BERT، GPTو T5 امکان دادند تا به نتایج پیشرفته در کارهایی مانند ترجمه زبان، تولید متن و پاســخ به ســؤالات دســت یابند. این مدل‌ها مقیاس‌پذیری و تطبیق‌پذیری یادگیری عمیق را نشان دادند.

دهه ۲۰۲۰ شـــاهد ظهور مدل‌های تولیدی، مانند شـــبکه‌های مولد (GAN)و مدل‌های انتشـار، بوده اسـت که می‌توانند تصـاویر، موسـیقی و متن واقع‌گرایانه ایجاد کنند. مدل‌هایی مانندDALL·E ، Stable Diffusionو ChatGPT مرزهای خلاقیت هوش‌مصنوعی را جابه‌جا کرده‌اند و امکان کاربردهایی در هنر، طراحی و تولید محتوا را فراهم کرده‌اند.

با قدرتمندتر شدن یادگیری عمیق، نگرانی‌ها درباره پیامدهای اخلاقی و اجتماعی آن افزایش یافته‌اسـت. مسـائلی مانند سـوگیری در سـیسـتم‌های هوش‌مصـنوعی، تأثیر محیطی آموزش مدل‌های بزرگ و امکان سـوءاسـتفاده، بحث‌هایی درباره توسـعه مسـئولانه هوش‌مصـنوعی و مقررات را برانگیخته است.

در دسـترس بودن چارچوب‌های یادگیری عمیق متن‌باز مانندTensorFlow ، PyTorch و Keras دســترســی به ابزارهای هوش‌مصــنوعی را دموکراتیک کرده اســت و به محققان،

[1] Graphic Processing Unit
[2] Tensor Processing Unit

توسعه‌دهندگان و کسب‌وکارها امکان می‌دهد شبکه‌های عصبی را راحت‌تر بسازند و مستقر کنند. این امر نوآوری و پذیرش را در صنایع مختلف تسریع کرده است.
هسته اصلی یادگیری عمیق استفاده از شبکه‌های عصبی با لایه‌های پنهان متعدد است، از این رو اصطلاح "عمیق" نامیده می‌شود. هر لایه در یک شبکه عصبی داده‌ها را در سطوح فزاینده‌ای از انتزاع پردازش می‌کند. برای مثال، در تشخیص تصویر، لایه‌های اولیه ممکن است لبه‌های ساده را شناسایی کنند، در حالی که لایه‌های بعدی اشکال پیچیده‌تر و در نهایت اشیا را می‌گیرند. این پردازش سلسله‌مراتبی به مدل‌های یادگیری عمیق اجازه می‌دهد تا به‌خوبی در طیفی از ورودی‌ها تعمیم پیدا کنند و آنها را برای کارهایی مانند ترجمه زبان، تشخیص تصویر، رانندگی مستقل و حتی انجام بازی‌های پیچیده بسیار متنوع می‌کند.

1-5-1 پرسپترون‌های چندلایه

پرسپترون چندلایه (MLP[1]) نوعی شبکه عصبی مصنوعی است که از چندین لایه گره (نورون) تشکیل شده‌است که در یک لایه ورودی، یک یا چند لایه پنهان و یک لایه خروجی سازماندهی‌شده‌اند. هر نورون در یک MLP به نورون‌های لایه بعدی متصل است و به هر اتصال وزنی اختصاص می‌یابد که تأثیر آن را تعیین می‌کند. این اتصالات و وزن‌های مرتبط به MLP اجازه می‌دهد تا داده‌های ورودی را به روش‌های انتزاعی فزاینده‌ای در هنگام عبور از لایه‌های شبکه تبدیل کند. این ساختار MLP ها را برای کارهایی که نیاز به شناسایی الگوها در داده‌ها دارند، مانند مشکلات طبقه بندی و رگرسیون قدرتمند می‌کند.
MLP بر اساس یادگیری نظارت شده عمل می‌کند، جایی که بر روی داده‌های برچسب دار آموزش می‌بیند تا نگاشت بین ورودی‌ها و خروجی‌های مورد نظر را بیاموزد. در طول آموزش، MLP از فرایندی به‌نام backpropagation برای تنظیم وزن‌ها بر اساس خطای بین پیش بینی‌های شبکه و خروجی‌های واقعی استفاده می‌کند. با به‌روزرسانی مکرر این وزن‌ها، MLP خطا را در طول زمان به حداقل می‌رساند و عملاً پیش‌بینی‌های دقیق روی داده‌های جدید و دیده نشده را «یاد می‌گیرد». توابع فعال‌سازی مانند sigmoid، ReLU (واحد خطی اصلاح‌شده)، یا tanh اغلب در هر لایه برای معرفی تبدیل‌های غیرخطی اعمال می‌شوند و MLP را قادر می‌سازد تا الگوهای پیچیده‌تری را نسبت به خطی ساده بیاموزد.

[1] Multi Layer Perceptron

$$ReLU(x) = \max(0, x)$$

$$\boldsymbol{\sigma(x) = \frac{1}{1 + e^{-x}}}$$

$$\boldsymbol{tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}}$$

در حالی که MLPها یکی از اولین انواع شبکه‌های عصبی بودند و در یادگیری عمیق پایه‌ای هستند، آنها برای داده‌های ساختاری، جدولی یا وظایفی که در آن روابط بین متغیرها نسبتاً ساده است، مناسب‌تر هستند. در مواردی مانند داده‌های تصویر یا توالی، که وابستگی‌های مکانی یا زمانی بسیار مهم هستند، دیگر معماری‌های شبکه عصبی، مانند شبکه‌های عصبی کانولوشنال (CNN) یا شبکه‌های عصبی مکرر (RNN[1]) مؤثرتر هستند. با این حال، MLPها یک بلوک ساختمانی مهم در یادگیری عمیق، به‌ویژه به‌عنوان جزئی از مدل‌های پیچیده‌تر، باقی می‌مانند و همچنان در کاربردهای عملی مختلف، از جمله مالی، مراقبت‌های بهداشتی و سیستم‌های توصیه استفاده می‌شوند.

آموزش مدل‌های یادگیری عمیق به مجموعه داده‌های بزرگ و منابع محاسباتی قابل توجه و همچنین سخت افزارهای تخصصی مانند GPU نیاز دارد. فرآیند یادگیری معمولاً تکراری است و از الگوریتم‌هایی مانند انتشار پس‌انداز و بهینه‌سازها برای تنظیم وزن در لایه‌های شبکه استفاده می‌کند و خطاها را در طول زمان به حداقل می‌رساند. با پیشرفت یادگیری عمیق، معماری‌های جدیدتری مانند شبکه‌های عصبی کانولوشنال (CNN) و ترانسفورماتورها پدیدار شدند که هر کدام برای انواع داده‌ها و وظایف مناسب هستند. این نوآوری‌ها منجر به پیشرفت‌هایی در هوش‌مصنوعی شده‌است و ماشین‌ها را قادر می‌سازد تا کارهایی را انجام دهند که زمانی تصور می‌شد نیاز به هوش در سطح انسان دارند.

یادگیری عمیق تأثیر عمیقی بر حوزه‌های مختلف، از جمله مراقبت‌های بهداشتی (مانند تصویربرداری پزشکی، کشف دارو)، خودروهای خودران، مالی و رباتیک داشته است. ماهیت بین‌رشته‌ای آن همچنان همکاری بین دانشمندان کامپیوتر، مهندسان و متخصصان حوزه‌های مختلف را تقویت می‌کند.

علی‌رغم موفقیت‌هایش، یادگیری عمیق با چالش‌هایی مانند نیاز به مجموعه‌داده‌های بزرگ برچسب‌دار، آسیب‌پذیری در برابر حملات متخاصم و مشکلات در تعمیم دادن به کارهای مختلف مواجه است. حل این چالش‌ها برای پیشرفت ادامه‌دار این زمینه حیاتی خواهد بود.

[1] Recurrent Neural Network

تاریخچه یادگیری عمیق همچنین داستانی از همکاری جهانی است. محققان و مؤسسات از سراسر جهان به توسعه آن کمک کرده‌اند و جامعه‌ای پویا و فراگیر ایجاد کرده‌اند.
دستاوردهای یادگیری عمیق نسل جدیدی از دانشمندان، مهندسان و کارآفرینان را الهام بخشیده‌است تا پتانسیل هوش‌مصنوعی را کشف کنند. ابتکارات و منابع آموزشی یادگیری عمیق را برای مردم آسان‌تر کرده‌اند تا درباره آن بیاموزند و به این زمینه کمک کنند.
از آغاز فروتنلنه‌اش در دهه ۱۹۴۰ تا وضعیت کنونی‌اش به‌عنوان یک فناوری تحول‌آفرین، تاریخچه یادگیری عمیق و شبکه‌های عصبی گواهی بر نبوغ و پشتکار انسان است. با ادامه تکامل این زمینه، وعده شکل‌دهی به آینده‌ای را دارد که تنها می‌توانیم شروع به تصور آن کنیم.

2-5-1 شبکه‌های کانولوشنی

یک شبکه کاملاً متصل (FCN[1]) که به عنوان شبکه عصبی متراکم نیز شناخته می‌شود، یکی از ساده‌ترین و در عین حال قدرتمندترین معماری‌ها در یادگیری عمیق است. از چندین لایه نورون تشکیل شده است که در آن هر نورون در یک لایه از طریق اتصالات وزنی به هر نورون در لایه بعدی متصل می شود. این شبکه ها به طور گسترده در وظایف یادگیری ماشین از جمله طبقه بندی، رگرسیون و تقریب تابع استفاده می شوند. تولنایی آن‌ها در مدل‌سازی روابط پیچیده بین ورودی‌ها و خروجی‌ها، آنها را به یک جزء اساسی در چارچوب‌های یادگیری عمیق تبدیل می‌کند.
یک شبکه FC به سه نوع لایه اصلی ساختار یافته است: لایه ورودی که داده های خام را دریافت می کند. یک یا چند لایه پنهان، جایی که محاسبات و استخراج ویژگی انجام می شود. و یک لایه خروجی که پیش بینی نهایی را تولید می کند. هر ارتباط بین نورون ها دارای یک وزن مرتبط است و هر نورون یک اصطلاح سوگیری دارد. در طول آموزش، این پارامترها از طریق الگوریتم های بهینه سازی، مانند نزول گرادیان تصادفی (SGD[2]) یا Adam تنظیم می شوند تا خطا بین خروجی های پیش بینی شده و واقعی به حداقل برسد.
یکی از مزایای اصلی شبکه های FC جهانی بودن آنهاست. با توجه به نورون‌ها و لایه‌های کافی، یک شبکه FC می‌تواند هر تابع پیوسته را تقریب بزند، این ویژگی به عنوان قضیه تقریب جهانی شناخته می‌شود. این باعث می شود آنها برای طیف گسترده ای از برنامه ها، از تشخیص تصویر گرفته تا پیش

[1] Fully Connected Network
[2] Stochastic Gradient Descendant

بینی سری های زمانی، مفید باشند. با این حال، این انعطاف پذیری هزینه دارد: شبکه های FC تمایل دارند به تعداد زیادی پارامتر نیاز داشته باشند که منجر به حافظه و تقاضاهای محاسباتی بالا می شود. با افزایش ابعاد ورودی، این موضوع بارزتر می شود.

برای بهبود کارایی یادگیری، شبکه‌های FC اغلب از توابع فعال‌سازی مانند ReLU (واحد خطی اصلاح‌شده)، سیگموئید یا tanh استفاده می‌کنند. این توابع غیرخطی بودن را وارد شبکه می‌کنند و به آن اجازه می‌دهند تا روابط پیچیده در داده‌ها را بیاموزد. بدون توابع فعال‌سازی، یک شبکه FC مانند یک مدل خطی رفتار می‌کند و توانایی آن را برای گرفتن الگوهای پیچیده محدود می‌کند. ReLU به دلیل سادگی محاسباتی و توانایی کاهش مشکل Vanishimg Gradient، که در شبکه های عمیق آموزش دیده با فعال سازی سیگموئید یا tanh رخ می دهد، محبوبیت خاصی دارد.

در Deep Reinforcement Learning (DRL)، شبکه‌های FC اغلب برای تقریب توابع ارزش، خط‌مشی‌ها یا نگاشت‌های عملکرد حالت استفاده می‌شوند. به عنوان مثال، در Deep Q-Networks (DQN)، یک شبکه FC نمایشی از وضعیت محیط را به عنوان ورودی دریافت می کند و مقادیر Q را برای اقدامات مختلف خروجی می دهد. به همین ترتیب، در روش‌های Actor-Critic، از دو شبکه FC مجزا استفاده می‌شود: یکی برای بازیگر (سیاست) و دیگری برای منتقد (عملکرد ارزش). این شبکه ها به عوامل کمک می کنند تا استراتژی های تصمیم گیری بهینه را در محیط های پیچیده بیاموزند.

شبکه‌های FC هنگامی که برای مسئله زمان‌بندی پروژه با محدودیت منابع (RCPSP) اعمال می‌شوند، در یادگیری سیاست‌های زمان‌بندی کارآمد نقش دارند. ورودی شبکه می تواند نشان دهنده وضعیت فعلی پروژه، از جمله وظایف برنامه ریزی شده و برنامه ریزی نشده، منابع موجود و ضرب الاجل باشد. سپس شبکه می تواند اقدامات بهینه را پیش بینی کند، مانند اینکه کدام کار بعدی را برنامه ریزی کند یا چگونه منابع را به طور موثر تخصیص دهد. با این حال، به دلیل ماهیت ترکیبی RCPSP، شبکه‌های FC به تنهایی ممکن است برای تعمیم خوب مشکل داشته باشند، به ویژه در نمونه‌های مقیاس بزرگ با فضاهای ورودی با ابعاد بالا.

یکی از چالش های کلیدی شبکه های FC در چنین برنامه هایی ناتوانی آنها در بهره برداری از ساختار داده های ورودی است. برخلاف شبکه‌های عصبی گراف (GNN[1])، که به صراحت وابستگی‌های بین وظایف و منابع را مدل‌سازی می‌کنند، شبکه‌های FC همه ویژگی‌های ورودی را مستقل تلقی می‌کنند که به طور بالقوه منجر به ناکارآمدی در یادگیری می‌شود. برای پرداختن به این موضوع، محققان اغلب

[1] Graph Neural Network

شبکه‌های FC را با معماری‌های دیگر، مانند مکانیسم‌های توجه یا لایه‌های کانولوشن، ترکیب می‌کنند تا وابستگی‌های ساختاری در زمان‌بندی پروژه را بهتر درک کنند.
با وجود این محدودیت‌ها، شبکه‌های FC همچنان یک ابزار اساسی در یادگیری عمیق و برنامه‌های زمان‌بندی مبتنی بر DRL هستند. سادگی، سهولت اجرا و سازگاری آنها با چارچوب های یادگیری عمیق مدرن، آنها را به نقطه شروع ارزشمندی برای بسیاری از مشکلات یادگیری ماشین تبدیل می کند. با پیشرفت تحقیقات، رویکردهای ترکیبی که شبکه‌های FC را با مدل‌های ساختاریافته‌تر ادغام می‌کنند (به عنوان مثال، GNN یا ترانسفورماتور) ممکن است اثربخشی آنها را در مسائل پیچیده زمان‌بندی و بهینه‌سازی افزایش دهد.

3-5-1 شبکه‌های گرافی

شبکه‌های عصبی گراف (GNN) دسته‌ای از شبکه‌های عصبی هستند که برای کار بر روی داده‌های ساختار یافته گرافی طراحی‌شده‌اند. نمودارها ساختارهای ریاضی متشکل از گره‌ها (یا رئوس) و یال‌ها (یا اتصالات) هستند که روابط بین گره‌ها را نشان می‌دهند. GNN ها به‌ویژه برای کارهایی که داده‌ها ذاتاً رابطه‌ای هستند، مانند شبکه‌های اجتماعی، ساختارهای مولکولی، سیستم‌های توصیه و نمودارهای دانش مفید هستند. برخلاف شبکه‌های عصبی سنتی که بر روی داده‌های شبکه‌مانند (مانند تصاویر یا توالی‌ها) کار می‌کنند، GNNها می‌توانند ساختارهای داده‌ای نامنظم و غیراقلیدسی را مدیریت کنند، که آنها را برای نمایش داده‌های پیچیده بسیار متنوع می‌کند.

ایده اصلی پشت GNN ها یادگیری نمایش گره‌ها، لبه‌ها یا کل نمودارها با جمع آوری اطلاعات از محله‌های محلی آنهاست. این امر از طریق فرایندی به نام ارسال پیام به‌دست می‌آید که در آن هر گره اطلاعات را از گره‌های همسایه خود دریافت و پردازش می‌کند. مکانیسم ارسال پیام به GNN ها اجازه می‌دهد تا هم اطلاعات ساختاری گراف و هم ویژگی‌های گره‌های جداگانه را ضبط کنند. در چندین تکرار، گره‌ها اطلاعات جهانی را جمع‌آوری می‌کنند و شبکه را قادر می‌سازد تا نمایش‌های سلسله مراتبی گراف را بیاموزد.
شبکه‌های عصبی گراف (GNN) ریشه در زمینه وسیع‌تر نظریه گراف و یادگیری ماشین دارد، که طی چندین دهه برای رسیدگی به چالش‌های پردازش داده‌های ساختاریافته نمودار تکامل یافته‌است. اولین پایه‌های GNN ها را می‌توان به دهه 1990 ردیابی کرد، زمانی که

محققان شروع به کشف راه‌هایی برای گسترش شبکه‌های عصبی برای مدیریت داده‌های غیر اقلیدسی کردند. یکی از کارهای پیشگام در این زمینه، توسعه شبکه‌های عصبی بازگشتی (RecNNs) توسط Sperduti و Starita در سال 1997 بود که هدف آن پردازش نمودارهای غیر چرخه‌ای جهت‌دار (DAGs) بود. اگرچه دامنه محدودی داشت، اما این کار زمینه را برای پیشرفت‌های بعدی در یادگیری مبتنی بر نمودار فراهم کرد.

مفهوم GNN ها همانطور که امروزه می‌شناسیم در اوایل دهه 2000 با کار مارکو گوری و همکارانش که اصطلاح "شبکه عصبی گراف" را در سال 2005 معرفی کردند، شکل گرفت. چارچوب آنها ایده استفاده از شبکه‌های عصبی برای پردازش گراف را رسمیت بخشید. داده‌ها با به‌روزرسانی مکرر نمایش گره‌ها بر اساس همسایگانشان. این رویکرد از این باور الهام گرفته شده‌است که بسیاری از مشکلات دنیای واقعی، مانند تحلیل شبکه‌های اجتماعی و مدل‌سازی مولکولی، می‌توانند از مدل‌هایی بهره ببرند که به صراحت ساختارهای رابطه‌ای را توضیح می‌دهند. با این حال، این GNN های اولیه از نظر محاسباتی گران بودند و فاقد مقیاس پذیری مورد نیاز برای برنامه‌های کاربردی در مقیاس بزرگ بودند.

عصر مدرن GNN ها در اواسط دهه 2010 با ظهور یادگیری عمیق و معرفی شبکه‌های کانولوشن گراف (GCNs[1]) توسط توماس کیپف و مکس ولینگ در سال 2016 آغاز شد. از شبکه‌های عصبی سنتی گرفته تا نمودارها با استفاده از فیلترهای طیفی، GCN ها می‌توانند به‌طور مؤثر اطلاعات را از همسایگی محلی یک گره جمع آوری کنند، و آنها را برای کارهایی مانند طبقه بندی گره و پیش‌بینی پیوند بسیار مؤثر می‌کند. سادگی و اثربخشی GCN ها منجر به پذیرش گسترده آنها شد و باعث افزایش علاقه به تحقیقات GNN شد.

در یک GCN، نمایش هر گره با ترکیب ویژگی‌های خود با مجموع وزنی از ویژگی‌های همسایه‌اش به‌روز می‌شود. این فرآیند شبیه به نحوه جمع آوری اطلاعات از پیکسل‌های مجاور در یک تصویر توسط فیلترهای کانولوشن است. GCN ها به‌دلیل سادگی و اثربخشی در کارهایی مانند طبقه بندی گره‌ها، پیش‌بینی پیوندها و طبقه بندی گراف به‌طور گسترده مورد استفاده قرار گرفته‌اند.

به‌دنبال موفقیت GCN ها، محققان شروع به بررسی معماری‌های جایگزین برای رفع محدودیت‌های آنها کردند. در سال 2017، شبکه‌های توجه گراف (GAT) توسط Veličković و همکاران معرفی شدند، که مکانیزم توجهی را برای سنجش اهمیت گره‌های

[1] Graph Convolutional Network

همســـایه به‌طور متفاوت در خود جای دادند. این نوآوری به GATها اجازه داد تا بر روی ارتباطات مرتبط‌تر تمرکز کنند و عملکرد را در وظایفی مانند سیستم‌های توصیه و تحلیل شبکه‌های اجتماعی بهبود بخشند. تقریباً در همان زمان، GraphSAGE توسط همیلتون و همکاران پیشـــنهاد شـــد، که یک رویکرد مبتنی بر نمونه‌گیری را برای مدیریت کارآمد نمودارهای مقیاس بزرگ معرفی کرد. این پیشـــرفت‌ها کاربرد GNN ها را برای مشـــکلات دنیای واقعی با مجموعه داده‌های عظیم گسترش داد.

اواخر دهه 2010 و اوایل دهه 2020 شـاهد ظهور معماری‌ها و تکنیک‌های پیچیده GNN بودیم. مدل‌هایی مانند Graph Isomorphism Networks (GIN) و DiffPool راه‌های جدیدی را برای ثبت ساختارها و سلسله‌مراتب‌های گراف معرفی کردند که عملکرد بهتری را در وظایفی مانند طبقه‌بندی و خوشـــه‌بندی گراف‌ها ممکن می‌ســـازد. محققان همچنین شـروع به پرداختن به چالش‌هایی مانند هموارسـازی بیش از حد کردند، که در آن گام‌های مکرر ارسال پیام باعث می‌شود نمایش گره‌ها غیرقابل تشخیص شوند. تکنیک‌هایی مانند اتصالات پرش، شبکه‌های باقیمانده و ادغام سلسله مراتبی برای کاهش این مشکل و بهبود عمق و مقیاس پذیری GNN ها توسعه داده شد.
به موازات پیشـرفت‌های معماری، GNN ها در حوزه‌های کاربردی مختلف شـروع به جذب کردند. در زیســـت‌شـــناســـی، GNN ها برای کشـــف دارو، پیش‌بینی برهمکنش پروتئین و پیش‌بینی خواص مولکولی اسـتفاده می‌شـدند. در بینایی کامپیوتری، آنها برای تولید نمودار صحنه و تشخیص اشیا سه بعدی استفاده شدند. در پردازش زبان طبیعی، GNN ها برای کارهایی مانند برچسب‌گذاری نقش معنایی و طبقه بندی اسناد به کار گرفته شدند. تطبیق پذیری GNN ها در این زمینه‌ها پتانسـیل آنها را برای ایجاد تحول در یادگیری ماشـینی و هوش‌مصنوعی نشان داد.
علی‌رغم پیشرفت سریع، GNN ها همچنان با چالش‌هایی به‌ویژه در مقیاس پذیری و تعمیم مواجه هســـتند. پردازش گراف‌های بزرگ با میلیون‌ها گره و یال همچنان از نظر محاســـباتی فشـــرده اســـت و محققان را بر آن می‌دارد تا تکنیک‌هایی مانند نمونه‌گیری زیرگراف، تقســـیم‌بندی گراف و آموزش توزیع‌شـــده را توســـعه دهند. علاوه بر این، اطمینان از تعمیم GNN ها به ساختارهای گراف غیرقابل مشاهده، یک حوزه تحقیقاتی مداوم است. تلاش‌ها برای رسیدگی به این چالش‌ها باعث ایجاد نوآوری و گسترش قابلیت‌های GNN می‌شود.

امروزه، GNN ها یک حوزه تحقیقاتی پر رونق با جامعه‌ای پر جنب و جوش از محققان و پزشکان هستند. کنفرانس‌هایی مانند NeurIPS، ICML، و ICLR به‌طور منظم کارهای پیشرفته‌ای را روی GNN ها ارائه می‌دهند و کتابخانه‌های منبع باز مانند PyTorch Geometric و DGL پیاده سازی و آزمایش GNN ها را برای توسعه دهندگان آسان‌تر کرده‌اند. از آنجایی که داده‌های ساختاریافته گراف به‌طور فزاینده‌ای در زمینه‌هایی مانند مراقبت‌های بهداشتی، مالی و حمل و نقل رایج می‌شوند، GNN ها آماده هستند تا نقشی مرکزی در شکل دادن به آینده یادگیری ماشین ایفا کنند.

یکی از مزایای کلیدی GNN ها توانایی آنها در تعمیم ساختارها و اندازه‌های مختلف نمودار است. برخلاف مدل‌های سنتی که به ورودی‌های با اندازه ثابت نیاز دارند، GNNها می‌توانند نمودارهایی را با تعداد گره‌ها و یال‌های مختلف مدیریت کنند. این انعطاف پذیری آنها را برای طیف گسترده‌ای از کاربردها، از تجزیه و تحلیل مولکول‌های کوچک گرفته تا مدل سازی شبکه‌های مقیاس بزرگ مانند اینترنت یا سیستم‌های حمل و نقل، مناسب می‌کند. علاوه بر این، GNN ها می‌توانند هر دو ویژگی گره (به‌عنوان مثال، نمایه‌های کاربر در یک شبکه اجتماعی) و ویژگی‌های لبه (مانند نقاط قوت تعامل) را در خود جای دهند و قدرت بیان آنها را بیشتر افزایش دهند.

با وجود نقاط قوت، GNN ها با چالش‌های متعددی روبه‌رو هستند. یکی از مسائل مهم صاف کردن بیش از حد است، که در آن مراحل مکرر ارسال پیام باعث می‌شود که نمایش گره‌ها غیر قابل تشخیص شوند. این مشکل به‌ویژه در GNNهای عمیق، جایی که لایه‌های زیادی برای گرفتن وابستگی‌های دوربرد مورد نیاز است، مشهود است. محققان راه‌حل‌های مختلفی مانند اتصالات پرش، شبکه‌های باقیمانده و ادغام سلسله مراتبی را برای کاهش هموارسازی بیش از حد و بهبود عمق و عملکرد GNN پیشنهاد کرده‌اند. چالش دیگر مقیاس پذیری است، زیرا پردازش گراف‌های بزرگ با میلیون‌ها گره و لبه می‌تولند از نظر محاسباتی فشرده باشد. تکنیک‌هایی مانند نمونه‌گیری زیرگراف، تقسیم‌بندی گراف، و آموزش توزیع‌شده برای رسیدگی به این موضوع توسعه داده‌شده‌اند.

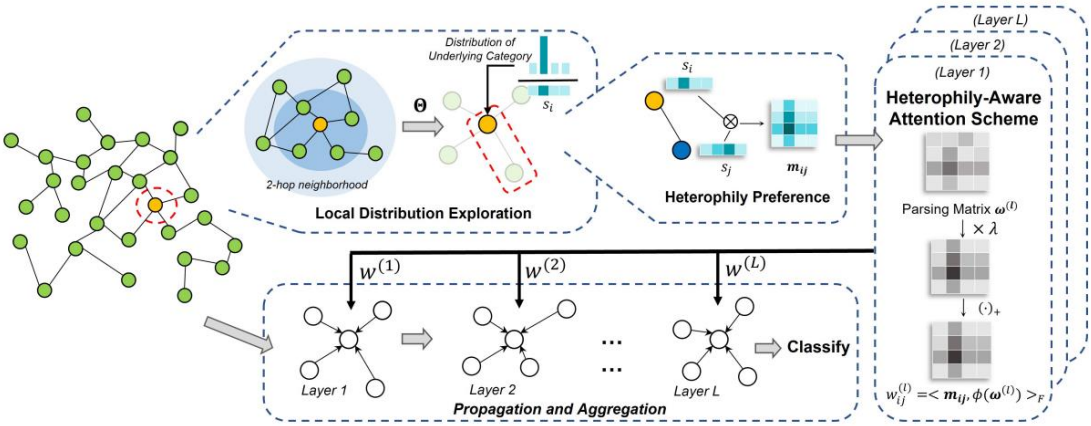
در سال‌های اخیر، GNN ها پیشرفت‌های سریع داشته‌اند و در تعداد فزاینده‌ای از دامنه‌ها به کار گرفته‌شده‌اند. در زیست‌شناسی، آنها برای کشف دارو، پیش‌بینی تداخل پروتئین و تشخیص بیماری استفاده می‌شوند. در بینایی کامپیوتری، GNN ها برای تولید نمودار صحنه و تشخیص اشیا سه بعدی استفاده می‌شوند. در پردازش زبان طبیعی، از آنها برای

برچسب‌گذاری نقش معنایی و طبقه بندی اسناد استفاده می‌شود. تطبیق پذیری GNN ها به نوآوری ادامه می‌دهد، با معماری‌ها و تکنیک‌های جدید در حال توسعه برای مقابله با مشکلات فزاینده پیچیده.

GNN ها ابزار قدرتمندی برای تجزیه و تحلیل و یادگیری از داده‌های ساختار یافته نمودار هستند. با استفاده از انتقال پیام و تجمع همسایگی، آنها می‌توانند الگوهای محلی و جهانی را در نمودارها ثبت کنند و آنها را برای طیف گسترده‌ای از برنامه‌ها مناسب کند. در حالی که چالش‌هایی مانند هموارسازی بیش از حد و مقیاس پذیری باقی مانده‌است، تحقیقات در حال انجام به این مسائل پرداخته و قابلیت‌های GNN ها را گسترش می‌دهد. از آنجایی که داده‌های ساختاریافته گراف در زمینه‌های مختلف رایج‌تر می‌شوند، GNNها آماده هستند تا نقشی مرکزی در پیشرفت یادگیری ماشین و هوش‌مصنوعی ایفا کنند.

4-5-1 شبکه گرافی توجه

شبکه‌های گرافی توجه (GAT) یکی دیگر از انواع محبوب شبکه توجه گراف (GAT) است که مکانیزم توجهی را برای سنجش اهمیت گره‌های همسایه به‌طور متفاوت معرفی می‌کند. برخلاف GCN ها که با همه همسایگان به‌طور مساوی رفتار می‌کنند، GAT ها یاد می‌گیرند سطوح مختلفی از توجه را به همسایگان مختلف اختصاص دهند و به مدل اجازه می‌دهند بر روی ارتباطات مرتبط تری تمرکز کند. این سازگاری GATها را به‌ویژه در سناریوهایی قدرتمند می‌کند که در آن روابط خاص از سایرین مهم‌تر هستند، مانند تجزیه و تحلیل شبکه‌های اجتماعی یا سیستم‌های توصیه.



GNN ها همچنین شـــامل معماری‌هایی مانند GraphSAGE هســـتند که ایده تجمع همسایگی را با نمونه برداری از یک زیرمجموعه با اندازه ثابت از همسایگان گسترش می‌دهد. این رویکرد به‌ویژه برای نمودارهای مقیاس بزرگ که پردازش کل محله از نظر محاســـباتی گران اســـت مفید اســـت. GraphSAGE با کاهش تعداد گره‌های درگیر در هر مرحله تجمیع، یادگیری مقیاس‌پذیر را امکان‌پذیر می‌کند و استفاده از GNNها را در مجموعه‌های داده عظیم مانند نمودارهای دانش در مقیاس وب یا شبکه‌های اجتماعی امکان‌پذیر می‌سازد. فراتر از وظایف ســـطح گره، GNN ها می‌توانند در ســـطح گراف نیز عمل کنند، جایی که هدف پیش‌بینی ویژگی‌های کل گراف اســـت. به‌عنوان مثال، در پیش‌بینی ویژگی‌های مولکولی، کار ممکن است شامل پیش‌بینی سمیت یا حلالیت یک مولکول بر اساس نمایش نمودار آن باشد. برای دستیابی به این هدف، GNN ها اغلب از یک تابع بازخوانی استفاده می‌کنند که ویژگی‌های گره و لبه را در یک نمایش در سطح نمودار جمع می‌کند. سپس این نمایش می‌تواند به یک طبقه بندی کننده یا رگرسیون برای پیش بینی‌ها وارد شود.

GAT

شبکه توجه گراف (GAT[1]) نوعی معماری شبکه عصبی است که به‌طور خاص برای پردازش داده‌های ساختار یافته گراف طراحی شده‌است. برخلاف شبکه‌های عصبی سنتی که بر روی داده‌های شبکه مانند تصاویر یا توالی‌ها کار می‌کنند، GATها قادر به مدیریت داده‌ها هستند که در آن روابط بین موجودیت‌ها به‌صورت لبه‌ها در یک نمودار نمایش داده‌می‌شود. این امر باعث می‌شود که GATها برای کارهایی مانند طبقه‌بندی گره‌ها، پیش‌بینی پیوندها و طبقه‌بندی نمودار مفید باشند، جایی که روابط بین گره‌ها به‌اندازه خود گره‌ها مهم است.

در هسته معماری GAT مفهوم مکانیسم‌های توجه است که در ابتدا در زمینه پردازش زبان طبیعی (NLP) توسط مدل‌هایی مانند Transformer رایج شد. مکانیسم‌های توجه به مدل اجازه می‌دهد تا هنگام پیش‌بینی‌ها، بر مرتبط‌ترین بخش‌های داده‌های ورودی تمرکز کند. در زمینه GATها، توجه برای ســـنجش اهمیت گره‌های همســـایه هنگام جمع‌آوری اطلاعات برای به‌روزرســـانی نمایش یک گره معین اســـتفاده می‌شـــود. این یک انحراف از شبکه‌های عصبی گراف قبلی (GNN) مانند شبکه‌های کانولوشن گراف (GCNs) است که اطلاعات همسایگان را با استفاده از وزن‌های ثابت جمع‌آوری می‌کنند.

[1] Graph Attention Networks

نوآوری کلیدی در GAT ها، معرفی لایه‌های خودتوجهی است که بر روی ساختار نمودار عمل می‌کنند. برای هر گره در نمودار، GAT ضرایب توجه را محاسبه می‌کند که تعیین می‌کند چقدر باید به هر یک از همسایگان آن اهمیت داده شود. این ضرایب در طول آموزش آموخته می‌شوند و با استفاده از مکانیزم توجه مشترک محاسبه می‌شوند. به‌طور خاص، برای یک گره معین، مکانیسم توجه ویژگی‌های گره و همسایه‌های آن را به‌عنوان ورودی می‌گیرد و مجموعه‌ای از ضرایب نرمال‌شده را که مجموع آن‌ها به یک می‌شود، خروجی می‌دهد. سپس از این ضرایب برای محاسبه مجموع وزنی ویژگی‌های گره همسایه استفاده می‌شود که از یک تابع فعال‌سازی غیرخطی برای به‌روزرسانی نمایش گره عبور می‌کند.

یکی از مزایای GAT ها توانایی آنها در مدیریت نمودارها با درجات مختلف اتصال است. برخلاف GCN ها که اندازه همسایگی ثابتی را در نظر می‌گیرند، GAT ها می‌توانند با تنظیم پویا وزن‌های توجه، با ساختارهای نمودارهای مختلف سازگار شوند. این باعث می‌شود GAT ها انعطاف پذیرتر شوند و بتوانند الگوهای پیچیده را در داده‌ها ثبت کنند. علاوه بر این، GAT ها را می‌توان به‌راحتی برای رسیدگی به توجه چند سر گسترش داد، که در آن مکانیسم‌های توجه متعدد به‌صورت موازی اعمال می‌شوند و خروجی‌های آنها به‌هم‌پیوسته یا میانگین می‌شوند. این به مدل اجازه می‌دهد تا انواع مختلفی از روابط بین گره‌ها را ثبت کند و می‌تواند منجر به بهبود عملکرد در وظایف خاص شود.

یکی دیگر از ویژگی‌های مهم GAT ها قابلیت تفسیر آنهاست. از آنجایی که ضرایب توجه در طول آموزش آموخته می‌شوند، می‌توانند بینش‌ی در مورد اینکه کدام همسایه‌ها برای نمایش یک گره معین مهم‌تر هستند، ارائه دهند. این می‌تولند به‌ویژه در برنامه‌هایی مفید باشد که درک روابط بین موجودیت‌ها مهم است، مانند تجزیه و تحلیل شبکه‌های اجتماعی یا پیش‌بینی ویژگی‌های مولکولی. با بررسی وزن‌های توجه، محققان می‌توانند درک بهتری از نحوه پیش‌بینی مدل به‌دست‌آورد و سوگیری‌ها یا خطاهای بالقوه را شناسایی کنند.

شبکه های گرافی توجه با وجود مزایایی که دارند محدودیت‌هایی نیز دارند. یک چالش مقیاس پذیری است، زیرا محاسبه ضرایب توجه می‌تواند از نظر محاسباتی گران باشد، به‌خصوص برای نمودارهای بزرگ با گره‌ها و لبه‌های زیاد. این می‌تولند GATها را برای کاربردهای بسیار بزرگ در مقایسه با معماری‌های ساده‌تر GNN مانند GCNها کمتر کاربردی کند. علاوه بر این، GAT ها ممکن است با نمودارهایی که داده‌های پر سر و صدا یا ناقص دارند مشکل داشته‌باشند، زیرا مکانیسم توجه ممکن است وزن‌های بالایی را به

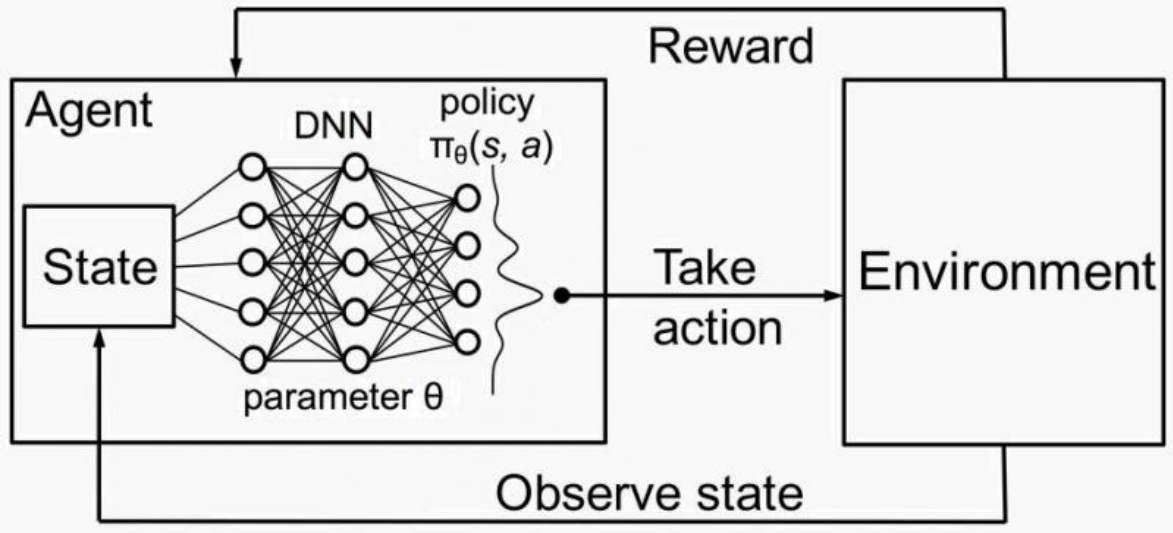
همسـایگان نامربوط یا گمراه کننده اختصـاص دهد. پیش‌پردازش دقیق و تکنیک‌های منظم اغلب برای کاهش این مسائل مورد نیاز است.

در عمل، GATها با موفقیت در طیف وسیعی از وظایف در دامنه‌های مختلف اعمال‌شده‌اند. به‌عنوان مثال، در بیوانفورماتیک، GAT ها برای پیش‌بینی برهمکنش‌های پروتئین-پروتئین و طبقه بندی نمودارهای مولکولی اسـتفاده‌شـده‌اند. در تجزیه و تحلیل شـبکه‌های اجتماعی، GAT ها برای شـناسـایی کاربران تأثیرگذار و شـناسـایی جوامع به کار گرفته‌شـده‌اند. در سیستم‌های توصیه، GAT ها برای مدل سازی تعاملات کاربر- آیتم و بهبود دقت توصیه‌ها اسـتفاده شـده‌اسـت. انعطاف پذیری و قدرت GAT ها آنها را به ابزاری ارزشـمند برای هر برنامه کاربردی که شامل داده‌های ساختار یافته گرافی است تبدیل می‌کند.
شـبکه‌های گرافی توجه با معرفی مکانیسـم‌های توجهی که به مدل اجازه می‌دهد به‌صـورت پویا اهمیت گره‌های همسـایه را بسـنجید، پیشـرفت قابل‌توجهی در زمینه شـبکه‌های عصـبی نمودار نشـان‌می‌دهد. این GATها را قادر می‌سـازد تا روابط پیچیده را در داده‌های سـاختار یافته گراف ثبت کنند و با ساختارهای گراف متفاوت سازگار شوند. در حالی که GAT ها از نظر مقیاس پذیری و حسـاسـیت به داده‌های پر سـر و صـدا دارای محدودیت‌هایی هسـتند، GAT ها ثابت کرده‌اند که در طیف گسترده‌ای از کاربردها، از بیوانفورماتیک گرفته تا تجزیه و تحلیل شبکه‌های اجتماعی، بسیار مؤثر هستند. همان طور که تحقیقات در این زمینه ادامه دارد، این احتمال وجود دارد که پیشـرفت‌ها و توسـعه‌های بیشـتری در معماری GAT ایجاد شـود و آنها را به ابزارهای قدرتمندتر و همه‌کاره‌تر برای یادگیری ماشـین مبتنی بر نمودار تبدیل کند.

1-6 یادگیری تقویتی عمیق

یادگیری تقویتی عمیق (DRL) شـاخه‌ای از یادگیری ماشـینی اسـت که یادگیری تقویتی (RL) را با یادگیری عمیق برای حل وظایف تصـمیم گیری پیچیده ترکیب می‌کند. در DRL، یک عامل یاد می‌گیرد که یک کار را با تعامل با یک محیط، دریافت بازخورد به‌شکل پاداش یا جریمه انجام دهد. هدف عامل این است که پاداش انباشته خود را در طول زمان با یادگیری یک خط‌مشـی بهینه به حداکثر برسـاند - یک اسـتراتژی برای انتخاب اقدامات در موقعیت‌های مختلف. یادگیری عمیق به عامل اجازه می‌دهد تا محیط‌هایی با داده‌های با ابعاد

بالا (مانند پیکســـل‌های خام در بازی‌های ویدیوئی) را مدیریت کند، جایی که می‌تواند مستقیماً از ورودی‌های حسی پیچیده یاد بگیرد.



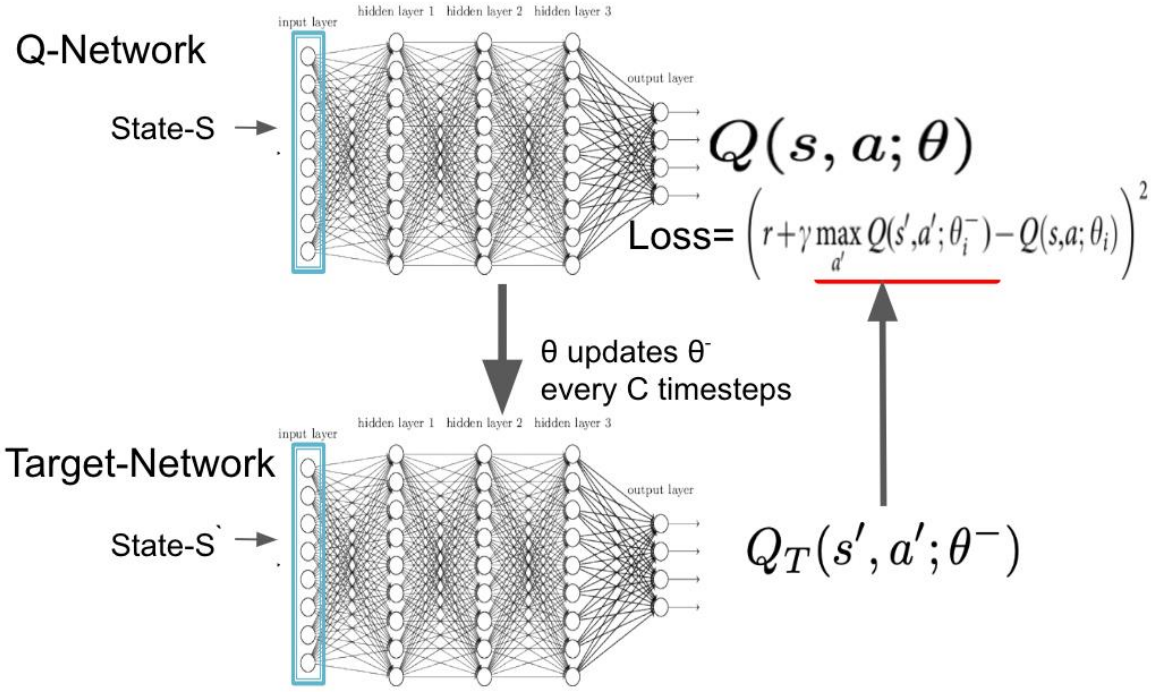
در هسته DRL، استفاده از شبکه‌های عصبی عمیق، که اغلب به‌عنوان «تقریب‌ساز توابع» شناخته می‌شوند، برای تخمین مؤلفه‌های کلیدی یادگیری تقویتی، مانند تابع Q-value یا تابع خط‌مشی، قرار دارد. به‌عنوان مثال، در Deep Q-Networks (DQNs)، یک شبکه عصبی برای تقریب مقادیر Q استفاده می‌شود که نشان دهنده پاداش‌های مورد انتظار از انجام اقدامات خاص از حالت‌های خاص است. در روش‌های گرادیان خط‌مشی، مانند بهینه‌سازی سیاست پروگزیمال (PPO) یا A3C (Asynchronous Advantage Actor-Critic)، یک شبکه عصبی مستقیماً خط‌مشی بهینه را یاد می‌گیرد. این رویکرد یادگیری عمیق به عوامل DRL اجازه می‌دهد تا فضاهای گسترده و مستمر حالت و عمل را مدیریت کنند، که روش‌های سنتی RL با آنها دست و پنجه نرم می‌کنند.

DRL به‌دلیل موفقیت در طیف وسیعی از وظایف چالش‌برانگیز، از انجام بازی‌های ویدیوئی و بازی‌های تخته‌ای در سطح مافوق بشری (مانند ربات‌های AlphaGo و Dota 2) گرفته تا برنامه‌های کاربردی در رباتیک، رانندگی مستقل و امور مالی، محبوبیت پیدا کرده است. این سیستم‌ها نه‌تنها قادر به یادگیری استراتژی‌های پیچیده هستند، بلکه می‌توانند با محیط‌های پویا و نامطمئن سازگار شوند. با این حال، آموزش عوامل DRL از نظر محاسباتی و داده فشرده‌است، زیرا آنها اغلب به میلیون‌ها تعامل با محیط برای رسیدن به عملکرد مطلوب نیاز دارند. علی‌رغم این چالش‌ها، DRL یک چارچوب قدرتمند برای مقابله با مشکلات دنیای واقعی است که شامل تصمیم گیری متوالی تحت عدم اطمینان است.

1-6-1 شبکه‌های عمیق q

شـبکه‌های عمیق q نشـان دهنده پیشـرفت قابل توجهی در زمینه یادگیری تقویتی (RL) است که یادگیری Q ابتدایی را با شبکه‌های عصبی عمیق برای حل مشکلات تصمیم گیری پیچیده ترکیب می‌کند. معرفی شده توسط منیه و همکاران. در سال 2013 و بعداً در مقاله مهم خود در سـال 2015، DQN یکی از اولین کاربردهای موفق یادگیری عمیق در RL بود. این تولنایی یادگیری سـیاسـت‌ها برای اجرای بازی‌های Atari 2600 در سـطحی مافوق‌بشری، تنها با استفاده از ورودی‌های پیکسل خام و بدون دانش قبلی از قوانین بازی را نشـان داد. این پیشـرفت نقطه عطفی در RL بود و پتانسـیل یادگیری عمیق برای مدیریت فضاهای حالت با ابعاد بالا را به نمایش گذاشت.

DQN در هسته خود مبتنی بر یادگیری Q است، یک الگوریتم کلاسیک RL که یک تابع ارزش عمل، Q (s, a) را یاد می‌گیرد، که پاداش تجمعی مورد انتظار انجام اقدام a را در حالت s و پیروی از خط مشـی بهینه پس از آن تخمین می‌زند. یادگیری Q سـنتی از یک جدول برای ذخیره مقادیر Q برای هر جفت حالت-عمل استفاده می‌کند که برای مشکلات با فضاهای حالت بزرگ یا پیوسته غیرممکن می‌شود. DQN با استفاده از یک شبکه عصبی عمیق برای تقریب تابع Q، این محدودیت را برطرف می‌کند و آن را قادر می‌سـازد تا بین حالت‌ها تعمیم دهد و ورودی‌های با ابعاد بالا مانند تصاویر را مدیریت کند.
یکی از نوآوری‌های کلیدی DQN اسـتفاده از بازپخش تجربه اسـت، مکانیزمی که تجربیات عامل (وضـعیت، اقدام، پاداش، حالت بعدی) را در یک بافر پخش مجدد ذخیره می‌کند. در طول آموزش، نماینده مجموعه‌ای کوچک از تجربیات را از این بافر برای به‌روزرسـانی شـبکه Q نمونه‌برداری می‌کند. این رویکرد همبسـتگی زمانی بین تجربیات متوالی را می‌شـکند و منجر به یادگیری پایدارتر و کارآمدتر می‌شـود. علاوه بر این، بازپخش تجربه به عامل اجازه می‌دهد تا از تجربیات گذشته استفاده مجدد کند، کارایی داده‌ها را بهبود می‌بخشد و امکان کاوش بهتر در فضای حالت را فراهم می‌کند.



یکی دیگر از اجزای حیاتی DQN استفاده از یک شبکه هدف است که یک شبکه عصبی مجزا با معماری مشابه شبکه Q اصلی اما با پارامترهای ثابت است. شبکه هدف برای محاسبه مقادیر Q هدف در طول به‌روزرسانی بلمن استفاده می‌شود، در حالی که شبکه Q اصلی با استفاده از نزول گرادیان به‌روز می‌شود. با جدا کردن مقادیر Q هدف از پارامترهای در حال بهینه‌سازی، شبکه هدف ناپایداری را کاهش می‌دهد و به جلوگیری از واگرایی در طول آموزش کمک می‌کند. این تکنیک عامل اصلی موفقیت DQN بود و از آن زمان به یک روش استاندارد در RL عمیق تبدیل شده‌است.

DQN همچنین چندین تکنیک پیش‌پردازش را برای مدیریت ورودی‌های پیکسل خام از بازی‌های Atari معرفی کرد. اینها عبارتند از انباشتن فریم، که در آن فریم‌های متوالی به‌هم متصل می‌شوند تا اطلاعات زمانی را ثبت کنند، و رنگ خاکستری، که ابعاد ورودی را کاهش می‌دهد. این مراحل پیش‌پردازش، همراه با شبکه‌های عصبی کانولوشن (CNN)، DQN را قادر می‌سازد تا به‌طور مؤثر ویژگی‌های مکانی و زمانی را از داده‌های بصری با ابعاد بالا بیاموزد. این قابلیت یک نقطه عطف بزرگ بود، زیرا نشان داد که عوامل RL می‌توانند مستقیماً از ورودی‌های حسی خام بدون ویژگی‌های مهندسی شده دست یاد بگیرند.

علی‌رغم موفقیت، DQN چندین محدودیت دارد. یکی از مسائل مهم تمایل آن به بیش از حد برآورد کردن مقادیر Q است که می‌تواند منجر به سیاست‌های غیر بهینه شود. این مشکل به این دلیل به وجود می‌آید که از همان شبکه برای انتخاب و ارزیابی اقدامات

استفاده می‌شود و باعث سوگیری مثبت در تخمین‌های Q-value می‌شود. برای رسیدگی به این موضوع، محققان DQN دوگانه (DDQN) را پیشنهاد کردند، که انتخاب و ارزیابی عملکرد را با استفاده از شبکه Q اصلی برای انتخاب اقدامات و شبکه هدف برای ارزیابی آنها جدا می‌کند. DDQN نشان داده است که تخمین بیش از حد را کاهش می‌دهد و عملکرد را در بسیاری از محیط‌ها بهبود می‌بخشد.
یکی دیگر از چالش‌های DQN ناکارآمدی نمونه آن است، زیرا اغلب به میلیون‌ها تعامل با محیط برای یادگیری سیاست‌های مؤثر نیاز دارد. این محدودیت تحقیقات را به‌سمت الگوریتم‌های RL کارآمدتر، مانند تکرار تجربه اولویت‌دار، که تجربیات نمونه‌گیری با خطاهای اختلاف زمانی بالا (TD) را در اولویت قرار می‌دهد، و Dueling DQN، که تخمین مقادیر حالت و مزایای عمل را از هم جدا می‌کند، تحریک کرده است. این پیشرفت‌ها عملکرد و پایداری روش‌های مبتنی بر DQN را بیشتر بهبود بخشیده‌است.

اما به صورت کلی الگوریتم DQN یک ادغام پیشگامانه از یادگیری عمیق و یادگیری تقویتی را نشان‌می‌دهد، که عوامل را قادر می‌سازد مستقیماً از ورودی‌های با ابعاد بالا بیاموزند و مسائل پیچیده تصمیم گیری را حل کنند. نوآوری‌های آن، مانند پخش مجدد تجربه و شبکه‌های هدف، به ابزارهای استاندارد در جعبه‌ابزار RL تبدیل‌شده‌اند. در حالی که چالش‌هایی مانند تخمین بیش از حد و ناکارآمدی نمونه همچنان باقی مانده‌است، تحقیقات در حال انجام بر پایه‌های DQN ادامه می‌یابد و مرزهای آنچه RL می‌تواند به آن دست یابد را تغییر می‌دهد. در نتیجه، DQN نه‌تنها در زمینه RL پیشرفت کرده است، بلکه راه را برای پیشرفت‌های آینده در زمینه مصنوعی هموار کرده است.

1-6-2 شبکه های دو عامل عمیق q

شبکه Q-Double Deep (DDQN) یک رویکرد یادگیری تقویتی است که به طور فزاینده ای برای حل مشکلات زمانبندی پروژه با محدودیت منابع (RCPSP) مورد بررسی قرار گرفته است. برخلاف روش‌های ابتکاری یا فراابتکاری سنتی، DDQN به صورت پویا یک خط‌مشی زمان‌بندی بهینه را با تعامل با یک محیط پروژه شبیه‌سازی شده می‌آموزد. این به مدل اجازه می دهد تا با محدودیت های مختلف پروژه سازگار شود و کارایی زمان بندی را در طول زمان بهبود بخشد.

در راه‌حل‌های RCPSP مبتنی بر DDQN، نمایش وضعیت اطلاعات زمان‌بندی ضروری، مانند پیشرفت کار، در دسترس بودن منابع، و روابط اولویت را به تصویر می‌کشد. فضای اقدام تصمیمات زمان بندی ممکن را تعریف می کند، از جمله انتخاب فعالیت بعدی برای اجرا یا تخصیص کارآمد منابع. یک عملکرد پاداش که به خوبی طراحی شده است، تضمین می کند که عامل به حداقل رساندن زمان ساخت و در عین حال رعایت محدودیت های پروژه اولویت بندی می کند.

مزیت کلیدی DDQN نسبت به شبکه استاندارد Deep Q-Networks (DQN) توانایی آن در کاهش تعصب بیش از حد تخمین است. در DQN استاندارد، از شبکه یکسانی برای انتخاب و ارزیابی اقدامات استفاده می شود که منجر به تخمین های ارزش بیش از حد خوش بینانه می شود. DDQN با تفکیک انتخاب کنش و ارزیابی به دو شبکه به این موضوع می پردازد – یکی اقدامات را انتخاب می کند، در حالی که دیگری مقادیر Q آنها را تخمین می زند. این منجر به یادگیری پایدارتر و بهبود تصمیمات برنامه ریزی می شود.

در طول آموزش، مدل DDQN با محیط زمان‌بندی تعامل دارد و خط‌مشی خود را از طریق پخش مجدد تجربه و به‌روزرسانی‌های شبکه هدف اصلاح می‌کند. با تکرارهای متعدد، عامل یاد می‌گیرد که کدام استراتژی‌های زمان‌بندی منجر به بهترین نتایج بلندمدت می‌شوند. این امر آن را قادر می سازد در نمونه های مختلف RCPSP تعمیم دهد و از اکتشافات سنتی، به ویژه در سناریوهای زمان بندی پیچیده و پویا، بهتر عمل کند.

با استفاده از DDQN، محققان می‌توانند استراتژی‌های زمان‌بندی تطبیقی را توسعه دهند که به صورت پویا با شرایط پروژه تنظیم می‌شود و آن را به ابزاری قدرتمند برای برنامه‌های RCPSP در دنیای واقعی تبدیل می‌کند. مطالعات آینده ممکن است این رویکرد را با ادغام محدودیت‌های اضافی، همکاری چند عاملی، یا تکنیک‌های یادگیری ترکیبی برای بهبود کارایی زمان‌بندی بیشتر تقویت کند.

3-6-1 بهینه‌سازی سیاست ابتدایی

الگوریتم بهینه‌سازی خط‌مشی مجانبی (PPO[1]) یکی از روش‌های یادگیری تقویتی (RL) است که برای حل **مسئله زمان‌بندی پروژه با منابع محدود (RCPSP) مورد بررسی قرار گرفته است. برخلاف روش‌های مبتنی بر ابتکار، PPO یک خط‌مشی تطبیقی برای زمان‌بندی فعالیت‌ها را از طریق تعامل با محیط پروژه یاد می‌گیرد و به‌طور پویا وظایف را بهینه می‌کند. از آنجا که RCPSP شامل برنامه‌ریزی وظایف متعدد با محدودیت‌های تقدم و منابع است، PPO رویکردی مبتنی بر داده را برای یافتن راه‌حل‌های نزدیک به بهینه ارائه می‌دهد.

PPO یک الگوریتم مبتنی بر خط‌مشی است که به جای تخمین مقادیر Q (مانند روش‌های مبتنی بر ارزش مانند DDQN)، مستقیماً یک خط‌مشی زمان‌بندی را یاد می‌گیرد. این الگوریتم خط‌مشی خود را به‌طور تدریجی بهبود می‌بخشد و از هدف تابع مجانبی برش‌یافته برای پایداری آموزش استفاده می‌کند. اجزای اصلی یک راه‌حل مبتنی بر PPO برای RCPSP شامل موارد زیر است:

1. نمایش وضعیت– عامل وضعیت فعلی پروژه را درک می‌کند، از جمله پیشرفت وظایف، منابع در دسترس، محدودیت‌های تقدم و هرگونه تأخیر.

2. انتخاب اقدام– PPO اقدامات زمان‌بندی را انتخاب می‌کند، مانند تصمیم‌گیری برای اجرای کدام وظیفه در مرحله بعد یا نحوه تخصیص منابع محدود.

3. تابع پاداش– الگوریتم بازخوردی بر اساس مدت‌زمان پروژه (makespan)، استفاده از منابع یا سایر معیارهای عملکردی دریافت می‌کند تا خط‌مشی خود را اصلاح کند.

4. به‌روزرسانی خط‌مشی– PPO به‌طور تدریجی خط‌مشی خود را به‌روزرسانی می‌کند تا از تغییرات ناگهانی جلوگیری کرده و یادگیری پایدار را تضمین کند.

PPO هنگام حل RCPSP مزایای متعددی ارائه می‌دهد که عبارتند از:

- پایداری و کارایی برخلاف روش‌های قدیمی‌تر گرادیان ، PPO تعادلی بین کاوش و بهره‌برداری حفظ می‌کند و به بهبود تدریجی خط‌مشی بدون نوسانات بزرگ کمک می‌کند.

[1] Policy Proximal Optimization

- قابلیت تعمیم PPO یک خط‌مشـــی زمان‌بندی یاد می‌گیرد که می‌تولند برای پروژه‌های مختلف اعمال شود و آن را برای سناریوهای متنوع RCPSP مفید می‌کند.
- یادگیری مداوم برخلاف روش‌های ابتکاری ایســتا، مدل آموزش‌دیده PPO می‌تواند تصــمیمات زمان‌بندی را به‌صــورت پویا تنظیم کندو به تغییرات در دســترس بودن منابع یا وابســتگی‌های وظایف پاسخ دهد.

برای آموزش یک عامل PPO در RCPSP، معمولاً یک **محیط شــبیه‌ســازی‌شــده پروژه** ایجاد می‌شـود. مدل در طی اپیزودهای مختلف آموزش داده می‌شـود و فعالیت‌ها را زمان‌بندی می‌کند، بازخورد دریافت می‌کند و خط‌مشـــی خود را اصـــلاح می‌کند. با اســـتفاده از تکنیک‌هایی مانند بازپخش تجربه (Experience Replay) و برش خط‌مشی (Policy Clipping) ، PPO اطمینان حاصل می‌کند که همگرایی پایدار به یک استراتژی زمان‌بندی بهینه دارد.

PPO می‌تولند با یادگیری چندعاملی (Multi-Agent Learning) گســـترش یلبد، جایی که چندین عامل RL برای برنامه‌ریزی بخش‌های مختلف یک پروژه با یکدیگر همکاری کنند. علاوه بر این، ترکیب PPO با روش‌های ابتکاری (مانند قوانین اولویت یا الگوریتم‌های ژنتیک) می‌تولند کارایی زمان‌بندی را بیشتر بهبود بخشـد. با پیشـرفت یادگیری تقویتی عمیق (Deep Reinforcement Learning)، انتظار می‌رود که مدل‌های زمان‌بندی مبتنی بر PPO نقش مهمی در مدیریت پروژه‌های خودکار و تطبیقی ایفا کنند.

1-6-4 مقایسه روش‌ها

با توجه به تحقیقات انجام شده و تجارب گذشته الگوریتم DDQN تا حدی می تواند عامل را به رسیدن به مقدار بهینه بهتر سوق دهد و همچنین نسبت به پاسخ های بهینه روش الگوریتم ژنتیک بهتر عمل کند. اما به صورت کلی و در فضای پیوسته الگوریتم PPO بهترین الگوریتم موجود خواهدبود. مهمترین چالش به وجود آمده و مورد بحث درباره مقایسه کاربرد این سه الگوریتم در روند حل مسائل برنامه‌ریزی پروژه خواهد بود. زیرا تجارب نشان داده‌است در فضای گسسته الگوریتم‌های DQN و DDQN می‌توانند بهتر از PPO در رسیدن به مقدار بهینه عمل کنند.

چارچوب این پروژه مبتنی بر یافتن بهترین الگوریتم است، در فصل سوم این پروژه به تعریف مسئله کلی می‌پردازیم و روند حل مسئله با اهداف آن را بررسی خواهیم کرد. در فصل چهارم بر روی بستر مناسب الگوریتم را در قالب کد پایتون پیدا سازی خواهیم کرد و نتایج اولیه را ثبت خواهیم کرد. در فصل

پنجم با توجه به نتایج به دست آمده به سوال مطرح شده پاسخ داده و ایده هایی بهتر در آینده برای حل مسائل برنامه‌ریزی پروژه بر مبنای بهترین الگوریتم انتخاب شده خواهیم پرداخت.

فصل2 مبانی نظری و پیشینه پژوهش

در مقالات اشـاره شـده به ارائه راهکاری برای رسـیدن به پاسـخ نزدیک به مقدار بهینه در مسـئله بهینه‌سـازی برنامه‌های پروژه با در نظر گرفتن محدودیت منابع اشـاره شـده‌اسـت، در مقاله‌ای از روش ژنتیک برای رسـیدن به پاسـخ نزدیک اشـاره شـده‌اسـت. در مقاله‌ای از روش forward-backward برای حل مسئله برنامه ریزی پروژه استفاده شده‌است. در مقاله‌ای از سیستم‌های چند عاملی برای کمک به حل مسئله استفاده کرده است. در مقاله‌ای از روش branch and bound برای حل مسئله برنامه ریزی پروژه استفاده شده‌است. در مقاله‌ای از روش مبتنی بر q-learning برای حل مسئله استفاده شده‌است.

همچنین دو مقاله زیر از روش‌هایی استفاده کرده‌اند که در ادامه با تأکید بر آن به حل مسئله می‌پردازیم:

1- در این مقاله از روش‌های یادگیری عمیق برای حل مسئله برنامه ریزی پروژه استفاده شده‌است. در این مقاله از روش یادگیری عمیق با استفاده از شبکه‌های نورونی گرافی برای پیش‌بینی با بالاترین درصد دقت اسـتفاده شـده‌اسـت. همچنین در این مقاله از روش پیش‌بینی سـیاسـت برای انتخاب روش بهینه با کمترین زمان ممکن اسـتفاده شده‌است.

2- در این مقاله از روش‌های مختلف یادگیری برای حل مسـئله برنامه ریزی پروژه اسـتفاده شـده‌اسـت. سـپس با تلفیق سـه روش برنامه ریزی بهینه در قالب یک روش

حل مســـئله برنامه ریزی پروژه در قالب یادگیری تقویتی چند عاملی مســـئله را حل کرده است.

در صـــفحه بعد در جدولی اطلاعات تمامی مقالات و مرور جزئی آن در دســـترس قرار گرفته است.

نام مقاله	یادگیری تقویتی	یادگیری عمیق	سیستم چند عاملی	روش‌های دیگر
A multi-agent optimization algorithm for resource constrained project scheduling problem(2015)			✓	
Optimal resource allocation using reinforcement learning for IoT content-centric services(2018)	✓			
A Heuristic Algorithm for Solving Resource Constrained Project Scheduling Problems(2017)				✓
Fast and Robust Resource-Constrained Scheduling with Graph Neural Networks(2023)		✓		
Reinforcement Learning for Constrained Project Scheduling Problem with Activity Iterations and Crashing (2020)	✓			
Cooperative Multi-Agent Control Using Deep Reinforcement Learning(2017)	✓	✓	✓	
A Q-Learning-based method applied to stochastic resource constrained project scheduling with new project arrivals(2006)	✓			
A Deep Reinforcement Learning Approach for Resource-Constrained Project Scheduling(2022)	✓	✓		
A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations(1997)				✓
Reinforcement Learning strategies for A-Team solving the Resource-Constrained Project Scheduling Problem(2014)	✓		✓	
Multi-Agent Reinforcement Learning: A Review of Challenges and Applications(2021)	✓		✓	

این موضوعات حوزه‌های ناشناخته یا ناشناخته را نشان می‌دهند که ترکیب RL و RCPSP می‌تولند به روش‌های جدیدی برای حل مشـــکلات زمان‌بندی پروژه پیچیده‌تر، کارآمدتر، سازگارتر و در مقیاس‌های بزرگ‌تر منجر شود.

فصل3 روش پژوهش

روش‌شناسی این پژوهش را می‌توان به عنوان یک رویکرد ترکیبیتوصیف کرد که یادگیری تقویتی (RL)** و الگوریتم ژنتیک به عنوان یکی از بهترین الگوریتم‌های فعلی بهینه‌سازی را برای حل مسئله زمان‌بندی پروژه با محدودیت منابع (RCPSP) ترکیب می‌کند.

این پژوهش به مسئله RCPSP می‌پردازد که یک مسئله بهینه‌سازی NP–سخت در مدیریت پروژه است. هدف، زمان‌بندی مجموعه‌ای از وظایف با مدت زمان، نیازهای منابع و محدودیت‌های پیشنیازی مشخص برای کمینه‌کردن مدت زمان کل پروژه (Makespanاست. مسئله به‌صورت یک **گراف جهت‌دارمدل‌سازی می‌شود که در آن وظایف، گره‌ها و وابستگی‌ها، یال‌ها هستند. هر وظیفه دارای ویژگی‌هایی مانند مدت زمان، نیازهای منابع و یک پرچم دودویی است که نشان می‌دهد آیا زمان‌بندی شده است یا خیر.

مسئله RCPSP به‌صورت یک گراف با استفاده از کتابخانه `networkx` نمایش داده می‌شود. هر وظیفه (گره) با ویژگی‌هایی مانند مدت زمان، نیازهای منابع و وضعیت زمان‌بندی همراه است. ساختار گراف، وابستگی‌های وظایف را نشان می‌دهد و امکان استفاده از شبکه‌های نورونی مختلف را برای اعمال شرایط مسئله مهیا کرده است. در این روش در ابتدا تمامی شبکه های نورونی ذکر شده در فصل قبل را در این روش پیاده سازی می‌کنیم. سپس بهترین و موثرترین شبکه نورونی مورد نیاز برای مسئله RCPSP مورد استفاده قرار می‌گیرد.

دو الگوریتم یادگیری تقویتی پیاده‌سازی شده‌اند: شبکه Q عمیق (DQNو بهینه‌سازی سیاست نزدیک‌شونده (PPOهر دو الگوریتم از GAT به‌عنوان تقریب‌زننده تابع سیاست یا Q-value استفاده می‌کنند. عامل‌های RL با محیط RCPSP تعامل می‌کنند، اقدامات (وظایف برای زمان‌بندی) را انتخاب می‌کنند و بر اساس مدت زمان کل پروژه، پاداش دریافت می‌کنند. محیط، بازخورد را به‌صورت حالت بعدی، پاداش و یک پرچم "تمام‌شده" ارائه می‌دهد که نشان می‌دهد آیا اپیزود کامل شده است یا خیر.

مسئله برنامه ریزی پروژه به عنوان فضای موجه عامل در یک کلاس تعریف می‌شود. این کلاس روش‌هایی برای بازنشانی محیط، اجرای اقدامات، محاسبه مدت زمان کل پروژه و زمان‌بندی وظایف بر

اســـاس یک ترتیب یا بردار وزن ارائه می‌دهد. محیط اطمینان حاصـــل می‌کند که محدودیت‌های منابع و وابسـتگی‌های وظایف در طول زمان‌بندی رعایت می‌شـوند. همچنین، اسـتفاده از منابع را در طول زمان با استفاده از یک DataFrame پانداس برای بررسی‌های تخصیص منابع به‌صورت کارآمد پیگیری می‌کند.

عامل‌های RL (DQN و PPO) در طول تعداد ثابتی از تکرارها آموزش می‌بینند. در طول آموزش، عامل‌ها با محیط تعامل می‌کنند، تجربیات را ذخیره می‌کنند و ســـیاســـت‌ها یا شـــبکه‌های Q خود را به‌روزرسـانی می‌کنند. عملکرد هر عامل بر اسـاس میانگین مدت زمان کل پروژه در چندین اپیزود ارزیابی می‌شـود. برای GPHH، فرآیند بهینه‌سـازی برای تعداد ثابتی از نسـل‌ها اجرا می‌شـود و بهترین بردار وزن و مدت زمان کل پروژه مربوطه ثبت می‌شود.

این پژوهش عملکرد DQN، PPO و GPHH را از نظر توانایی آن‌ها در کمینه‌کردن مدت زمان کل پروژه مقایسـه می‌کند. میانگین مدت زمان کل پروژه برای هر روش گزارش می‌شـود و بهترین بردار وزن یافت‌شده توسط GPHH نیز ارائه می‌شود. این مقایسه نقاط قوت و ضعف هر رویکرد را برجسته می‌کند و بینش‌هایی در مورد مناسـب‌بودن آن‌ها برای حل مسـئله RCPSP ارائه می‌دهد. پیاده‌سـازی از پایتون و کتابخانه‌های محبوبی مانند `torch` برای یادگیری عمیق، `networkx` برای نمایش گراف و `pandas` برای پیگیری اسـتفاده از منابع اسـتفاده می‌کند. کد به‌صـورت ماژولار اسـت و کلاس‌های جداگانه‌ای برای GAT، عامل‌های RL، محیط و GPHH دارد. این ماژولاریته امکان گسترش و آزمایش آسان با الگوریتم‌ها یا پیکربندی‌های مختلف مسئله را فراهم می‌کند.

نتایج برای تعیین این‌که کدام روش (DQN، PPO یا GPHH) در کمینه‌کردن مدت زمان کل پروژه بهتر عمل می‌کند، تحلیل می‌شود. این تحلیل شامل موارد زیر است:

- میزان زمان بهینه پروژه
- میانگین فاصله زمان بهینه پروژه با مسیر بحرانی

در هر کدام از شرایط مربوط به پروژه برای هر الگوریتم 15 تکرار اجرا شده است و نتایج آن در فصل بعد قابل ذکر و تحلیل است.

.

فصل4 تجزیه و تحلیل یافته‌ها

4-1 مقدمه

تمامی متدولوژی فصل گذشته در فایل پایتون پیوست شده به مستندات قرار دارد. در اعمال اولیه سعی شد که با یک مدل ساده Dqn تمامی شبکه های نورونی مورد نظر(تماما متصل، شبکه نورونی گرافی و شبکه نورونی توجه) برای اعمال بهتر الگوریتم DRL استفاده می‌کنیم.

سپس با انتخاب شبکه نورونی مناسب الگوریتم های یادگیری عمیق با دو رویکرد زیر بررسی خواهندشد:

1. تحلیل حساسیت بر روی محدودیت منابع: در این الگوریتم به ازای 20 فعالیت و تعداد 2 الی 7 منبع نتایج الگوریتم بررسی می‌شوند.
2. تحلیل حساسیت بر روی تعداد فعالیت‌ها: در این الگوریتم به ازای 2 منبع الگوریتم ها به ازای 20، 30، 40 و 50 فعالیت تقسیم خواهند شد.

لازم به ذکر است که تمامی فعالیت‌ها با توجه به زیرساخت در دسترس برای پردازش کد(کگل) به ازای 15 تکرار انجام شده‌است و زیرساخت مناسب و پایدار برای اجرای بیش از صد تکرار مهیا نبود.

همچنین تمامی فضای مربوط به RCPSP به صورت شبیه سازی شده و الگو گرفته از آخرین مقالات این حوزه که در منابع موجود است، تعریف شده‌است.

4-2 انتخاب شبکه نورونی مناسب

4-3 انتخاب الگوریتم یادگیری تقویتی

4-3-1 تحلیل الگوریتم ها بر مبنای تعداد منابع

سه الگوریتم یادگیری تقویتی(شبکه عمیق ساده q، شبکه عمیق دو عاملی q و بهینه سازی سیاست از مبدا) مورد بررسی قرار گرفت. نتایج کلی در کد قرار گرفته شده است. نتایج خلاصه به صورت زیر می‌باشد.

تعداد فعالیت	تعداد منابع	DQN	DDQN	PPO	GPPH
20	3	74	81	**75**	**71**
		36.26	25.4	39.27	
20	4	106	**86**	99	100
		90.4	**41.8**	59.6	
20	5	101	**96**	109	108
		65.6	**51.93**	72.73	
20	6	122	**105**	108	99
		91.06	**63.86**	86.0	
20	7				
20	8				

4-3-2 تحلیل الگوریتم ها بر مبنای تعداد فعالیت ها

تعداد فعالیت	تعداد منابع	DQN	DDQN	PPO	GPPH
20	3				
20	4				

				5	20
99				6	20
				7	20
				8	20

4-4 نتایج استخراج شده

در فاز اول حل مسئله شبکه نورونی توجه به صورت کاراتری شبکه بهتری خواهدبود و از الگوریتم آن برای تعامل عامل در فضای پروژه استفاده می‌شود. در فاز دوم نتایجی به دست می آید که نشان میدهد هر الگوریتم در شرایط خاصی از تعداد فعالیت ها و تعداد منابع محدود دارای مزیتی نسبت به یکدیگر هستند.

فصل 5 نتیجه‌گیری و پیشنهادها

5-1 نتیجه‌گیری

با تحلیل شبکه‌های مختلف نورونی در ابتدا برداشت می‌شود که با اعمال شبکه کانولوشنی پردازش بسیار طولانی و بدون نتیجه است. چنین چیزی کارا نبودن مدل را نشان‌می‌دهد.

در شبکه gnn پردازش با سرعت بیشتر و زمان کمتر انجام‌می‌شود اما مقدار پیشرفت عامل بعد از هر تکرار آنقدر مطلوب نیست.

در شبکه gat پردازش با سرعت کمتری نسبت به شبکه‌های عصبی گرافی انجام‌می‌شود و میزان پیشرفت عامل بعد از هر تکرار مطلوب‌تر است.

در نتیجه با تفاسیر داده شده شبکه نورونی توجه به‌عنوان شبکه نورونی مطلوب برای تعریف فضای مورد نیاز عامل در حل مسئله RCPSP تعریف می‌شود.

درباره سیاست های بهینه پروژه در برنامه های کوچک و دارای محدودیت الگوریتم PPO بهتر عمل می‌کند اما با افزایش فعالیت یا اضافه شدن منابع جدید الگوریتم DDQN بهتر عمل خواهند کرد.

همچنین میزان فاصله الگوریتم DDQN از مسیر بحرانی در هر شرایطی نسبت به الگوریتم های دیگر کمتر است که این مورد پایدار و قابل اتکا بودن این الگوریتم را نشان می دهد.

5-2 پیشنهادها

با توجه به پروژه انجام شده می‌توان تحقیقات را در پاسخ به سه مسئله زیر ادامه‌داد:

اعمال شبکه نورونی کاراتر: یکی از پیچیدگی‌های مربوط به حل چنین مسائلی در ابعاد بالا شبکه‌های نورونی با پردازش پیچیده و سنگین است. از این رو لازم است که برای

انتخاب شبکه عصبی بهینه برای کاربردهای صنعتی یک چالش چند وجهی است که تحت‌تأثیر عوامل مختلفی مانند کیفیت داده، منابع محاسباتی و الزامات خاص مشکل قرار

دارد. در اینجا مروری بر چالش‌ها و جهت گیری های بالقوه تحقیقات آینده برای رسیدگی به آنها آورده شده‌است:

پیچیدگی مدل در مقابل مبادله عملکرد

شبکه‌های عصبی عمیق (DNN[1]) اغلب در کارهایی مانند تشخیص تصویر و پردازش زبان طبیعی به‌دلیل توانایی آنها در گرفتن الگوهای پیچیده از مدل‌های ساده‌تر بهتر عمل می‌کنند. با این حال، آنها به منابع محاسباتی قابل توجهی نیاز دارند و تفسیر آنها سخت‌تر است، که باعث می‌شود برای برنامه‌های بلادرنگ یا استقرار در محیط‌های محدود مانند دستگاه‌های اینترنت اشیا یا پلتفرم‌های تلفن همراه کمتر امکان‌پذیر باشند. صنایع باید بین عملکرد و کارایی تعادل ایجاد کنند. به‌عنوان مثال، در خودروهای خودران، دقت بالا بسیار مهم است، اما تأخیر کم برای اطمینان از ایمنی به همان اندازه مهم است. چگونه می‌توان مدل‌ها را بدون از دست دادن قدرت پیش‌بینی بحرانی ساده کرد؟ تکنیک‌هایی مانند هرس، کوانتیزاسیون یا معماری‌های فشرده (مثلاً MobileNet) امیدوارکننده هستند، اما همچنان ممکن است دقت را برای موارد استفاده خاص به خطر بیندازند.

2. بهینه‌سازی Hyperparameter

پارامترهایی مانند نرخ یادگیری، انتخاب‌های بهینه‌ساز، اندازه دسته و معماری شبکه عصبی به‌طور قابل توجهی بر عملکرد مدل تأثیر می‌گذارند. انتخاب نادرست فراپارامتر می‌تواند منجر به مدل‌های نابهینه یا ناپایدار، به‌ویژه در مسائل برنامه‌ریزی شود. خودکارسازی کارآمد جستجوی فراپارامترها با در نظر گرفتن محدودیت‌های صنعتی مانند زمان، منابع و محیط‌های استقرار یکی از چالش‌های این حوزه است که بایستی به آن رسیدگی شود.

3. کیفیت و کمیت داده‌ها

بسیاری از صنایع با داده‌های ناقص کار می‌کنند، مانند داده‌های از دست رفته (خوانش حسگرها در سیستم‌های اینترنت اشیا ممکن است به‌دلیل اختلالات شبکه از بین برود) داده‌های پر سر و صدا: داده‌های تولید شده توسط کاربر یا داده‌های محیط‌های فیزیکی اغلب شامل خطا هستند.

4-تحلیل حساسیت مقدار بهینه به‌دست‌آمده به‌ازای پارامترهای مختلف

[1] Deep Neural Network

یکی از عناوین تحقیقات آتی می‌تواند انجام آزمایش‌های بیشتر جهت سنجش تغییر مقدار بهینه به‌ازای پارامترهای مختلف شبکه‌های نورونی، یادگیری تقویتی و بهینه‌سازی پروژه شود.

تحلیل زمان به‌دست‌آمده به‌ازای مقادیر مختلف اپسیلون و نرخ یادگیری می‌تواند بر روی نتیجه اثربخش باشد و تحلیل اثربخشی چنین عواملی می‌تواند موضوع جذابی برای تحقیق در آینده باشد.

یکی از مهم‌ترین چالش‌های پروژه طراحی سیستمی برای agent است که بتواند با استفاده از آزمون و خطا و روش‌های متداول سیاست بهینه را انتخاب کند در سیستم رسم شده سعی شده‌است که اگر agent فعالیتی را انجام دهد که پیش‌نیاز پس نیازی آن فراهم نشده است، با دریافت پاداش منفی بسیار زیاد در تکرارهای بعدی از این عمل اجتناب کند. مقدار جریمه و مقیاس جریمه و پاداش طی کردن مسیر درست می‌تواند موضوعات تحقیق مفید و جالبی برای آینده باشد.

5-نحوه اعمال چندین کاربر در یک الگوریتم

نحوه تعامل agentهای مختلف و نوع تعامل آنها با یکدیگر و بیشترین تأثیر تعامل آنها با یکدیگر می‌تواند از موضوعاتی باشد که با تحقیق آن در آینده بتوان به نتایج مفیدی رسید و از آن برای بهینه‌تر کردن برنامه ریزی پروژه استفاده‌کرد.

منابع یا مراجع

- Andrew G. Barto & R. S. Sutton & C. J. C. H. Watkins(1989) ,Learning and Sequential Decision Making
- Bert De Reyck & Willy Herroelen(1997), A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations
- P. Je˛drzejowicz, E. Ratajczak-Ropel(2013) , Reinforcement Learning strategies for A-Team solving the Resource-Constrained Project Scheduling Problem
- Lorenzo Canese & Gian Carlo Cardarilli & Luca Di Nunzio & Rocco Fazzolari & Daniele Giardino (2021), Multi-Agent Reinforcement Learning: A Review of Challengesand Applications
- Jayesh K. Gupta & Maxim Egorov & Mykel Kochenderfer (2017), Cooperative Multi-Agent Control Using Deep Reinforcement Learning
- Florent Teichteil-Königsbuch & Guillaume Povéda & Guillermo González de Garibay Barba & Tim Luchterhand & Sylvie Thiébaux (2023), Fast and Robust Resource-Constrained Scheduling with Graph Neural Networks.
- Xiaohan Zhao & Wen Song1 & Qiqiang Li & Huadong Shi & Zhichao Kang & Chunmei Zhang (2022). A Deep Reinforcement Learning Approach for Resource-Constrained Project Scheduling
- Keke Gai a & Meikang Qiu (2018). Optimal resource allocation using reinforcement learning for IoT
- Shelvin Chand & Hemant Kumar Singh & Tapabrata Ray (2017). A Heuristic Algorithm for Solving ResourceConstrained Project Scheduling Problems
- Inkyung Sung & Bongjun Choi & Peter Nielsen (2020). Reinforcement Learning for Resource Constrained Project Scheduling Problem with Activity Iterations and Crashing
- Jaein Choi & Matthew J. Realff & Jay H. Lee (2006). A Q-Learning-based method applied to stochastic resource constrained project scheduling with new project arrivals
- Hua Zhang & Hao Xu & Wuliang Peng (2008).A Genetic Algorithm for Solving RCPSP
- Jian Lina & Lei Zhua,& Kaizhou Gao (2020). A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem
- L. Peng & P. Wuliang (2014). An Efficient Simulation Algorithm for Resource-Constrained Project Scheduling Problem

پیوست