کاربرد یادگیری ماشین در مسیریابی برای شبکه‌های نرم‌افزار محور

علی باقری، دکتر محمد رضا پاکروان، دانشکده ی مهندسی برق، دانشگاه صنعتی شریف

چکیده- در سال‌های اخیر، به دلیل توسعه سریع اینترنت زیرساخت‌ها، تجهیزات و منابع شبکه پیچیده‌تر شده‌اند. برای سازمان‌دهی، بهینه کردن و مدیریت موثرتر شبکه، نیاز به هوشمندی بیشتری است. اما به دلیل ماهیت توزیع شده شبکه‌های مرسوم، استفاده از تکنیک‌های یادگیری ماشین دشوار است. ولی شبکه‌های نرم‌افزار محور به دلایلی از جمله کنترل متمرکز، دید کلی به شبکه، آنالیز نرم‌افزار محور ترافیک و اپدیت پویای قوانین ارسالی، امکان استفاده از تکنیک‌های یادگیری ماشین را راحت‌تر می‌کنند.

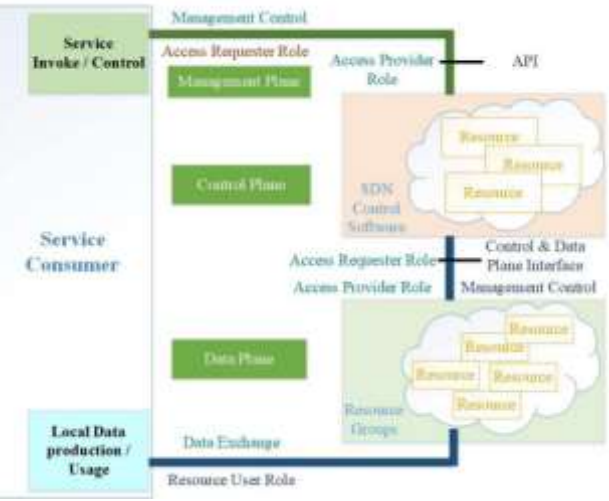
شبکه‌های نرم‌افزار محور، کنترل، قابل برنامه‌ریزی بودن و خودکار بودن شبکه را بهتر می‌کنند. هم‌چنین در حوزه‌های مختلفی از جمله در مسیریابی فوایدی دارد. به عبارت دیگر برای مدیریت موثرتر و مسیریابی بهینه‌تر در شبکه به هوشمندی نیاز است که شبکه‌های نرم‌افزار محور امکان انجام این کار را راحت‌تر می‌کنند. برای رسیدن به این مقصود محققان بسیاری تکنیک‌های مختلف یادگیری ماشینی را روی شبکه‌های نرم‌افزار محور برای کاربردهای مسیریابی اعمال کرده‌اند. ما در ابتدا مقدمه‌ای از این حوزه را بررسی می‌کنیم و سپس مروری بر شبکه‌های نرم‌افزار محور انجام داده و در نهایت تکنیک‌های مختلف یادگیری ماشین که برای بهینه‌سازی در مسیریابی در شبکه‌های نرم‌افزار محور استفاده می‌شوند را بررسی می‌کنیم.

کلمات کلیدی– شبکه نرم‌افزار محور، یادگیری ماشین، هوش مصنوعی، بهینه سازی، مسیریابی

1. مقدمه

تا چند سال قبل، اکثر شبکه‌های شرکت‌ها یک روند قدیمی داشتند. به طوری که در معماری دستگاه‌های شبکه‌های قدیمی Control Plane و Data Plane از هم جدا نشده بودند. به همین دلیل شبکه‌ها پیچیده شدند و همچنین مدیریت و نگهداری از آن‌ها نیز(به خصوص در مواردی که سریع رشد می‌کردند) دشوار بود. هم‌چنین با توسعه سریع و دستگاه‌های هوشمند(مثل گوشی‌های هوشمند، خودروهای هوشمند، دستگاه‌های خانه‌های هوشمند) و تکنولوژی‌های شبکه‌ها(مثل محاسبات ابری و...) ترافیک زیادی داریم که برای بهینه کردن آن‌ها شبکه‌ها در حال پیچیده‌تر شدن هستند که برای بهینه کردن آن‌ها می‌توان از الگوریتم‌های یادگیری ماشین استفاده کرد[۱۱].

زمانی که نرافزار با سخت‌افزار همراه هستند، دستگاه‌های مرتبط خیلی به شرکت سازنده آن‌ها محدود می‌شوند که در نتیجه‌ی این امر، شبکه‌ها غیرپویا و انعطاف ناپذیر می‌شوند. شبکه‌های نرم‌افزار محور با بردن قسمت کنترل به یک قسمت مرکزی(یعنی کنترل کننده شبکه نرم‌افزار محور) بر این مشکلات غلبه می‌کنند. این جدا کردن قسمت کنترلی با استفاده از پروتکل OpenFlow انجام می‌شود. شکل زیر معماری شبکه نرم‌افزار محور را نشان می‌دهد که در آن Data Plane (توابع رو به جلو و ارسالی) و Control Plane (کنترل شبکه) از هم جدا شده‌اند.



شکل ۱ : معماری شبکه نرم‌افزار[۱]

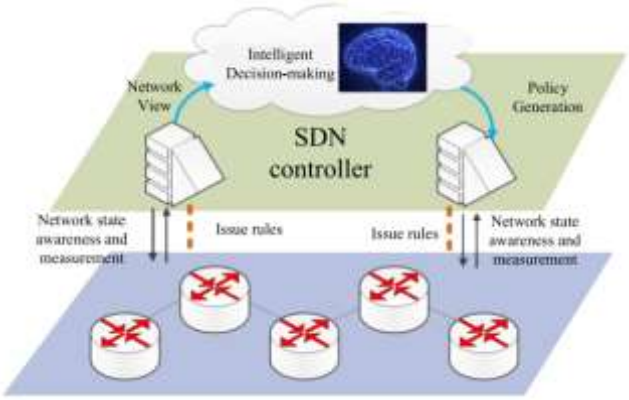
الگوریتم‌های مسیریابی قدیمی مثل OSPF برای شبکه‌های نرم‌افزار محور مناسب نیستند زیرا همگرایی و سرعت آن‌ها کم است و همچنین این الگوریتم‌ها از یک روند توزیع شده پیروی می‌کنند.

امروزه استفاده از تکنیک‌های یادگیری ماشین در حال افزایش است. این تکنیک‌ها بهتر از الگوریتم‌های قدیمی، به خصوص برای مواقعی که داده زیادی داریم، می‌باشد. در حوزه شبکه، محققان در حال بررسی کاربردهای این تکنیک‌ها هستند.

در حوزه شبکه‌های نرم‌افزار محور، یادگیری ماشین در موارد مختلفی از جمله مهندسی ترافیک، مدیریت منابع، سیستم‌های تشخیص intrusion، استفاده شده است.

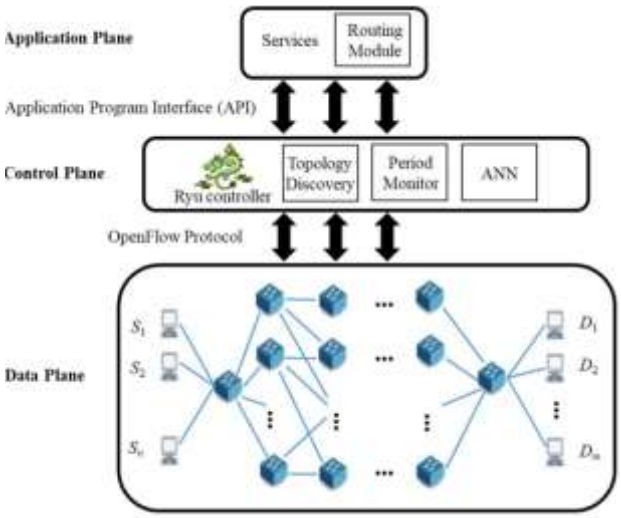
در نتیجه، در شبکه‌های نرم‌افزار محور نقش یادگیری ماشین به دلیل کاربردهای زیادش، افزایش یافته است. منطق معماری شبکه نرم‌افزار محور با یادگیری ماشین بهتر انطباق دارد. به طور خاص، تحقیق‌های زیادی، تکنیک‌های یادگیری ماشین را با شبکه‌های نرم‌افزار محور برای مسیریابی بهینه ترکیب می‌کند. هم‌چنین، یادگیری ماشین به عنوان یک تکنولوژی کلیدی برای 6G و فراتر از آن کاربرد دارد[۲].

شکل زیر ساختار یک شبکه نرم‌افزار محور که از یادگیری ماشین استفاده می‌کند را نشان می‌دهد. همانطور که در این شبکه دیده می‌شود، یک ماجول هوشمند برای تصمیم‌گیری در قسمت control plane شبکه نرم‌افزار محور وجود دارد. این ماجول هوشمند تصمیم‌گیری به طور موثر سیاست‌های شبکه را تولید می‌کند تا بتواند کنترل و مدیریت کلی و real time شبکه را انجام دهد. به طور خاص ماجول هوشمند تصمیم‌گیری با استفاده از دید کلی‌ای که از شبکه نرم‌افزار محور بدست آورده است استراتژی شبکه را تعیین می‌کند. با این معماری می‌توانیم به صورت هوشمند مسیریابی را بهینه کنیم.



شکل ۲ : ساختار یک فریم‌ورک شبکه نرم‌افزار محوری که از یادگیری ماشینی استفاده می‌کند

این ساختار را می‌توان از منظری دیگر نیز دید که در شکل زیر نمایش داده شده است:



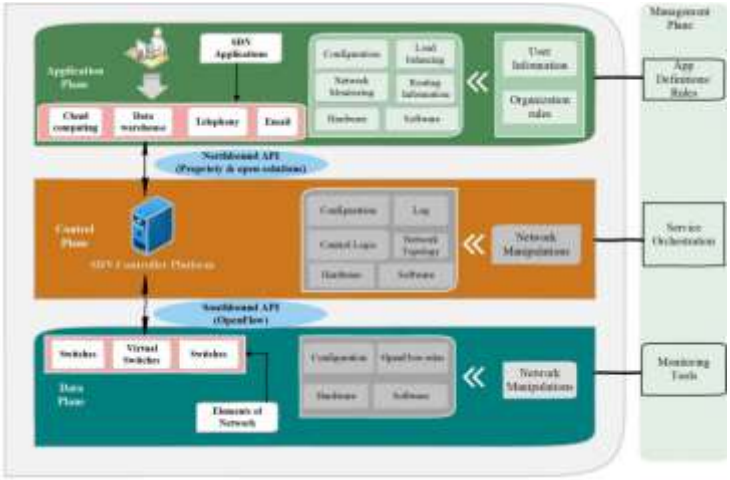
شکل ۳: یک روش پیشنهادی برای مسیریابی با استفاده از هوش مصنوعی در شبکه‌های نرم‌افزار محور[۱۲]

کنترل‌کننده شبکه نرم‌افزار محور می‌تواند شبکه را به صورت پویا برنامه‌ریزی کند. همچنین کنترل‌کننده مرکزی با نظارت و جمع‌آوری real-time حالت شبکه و نیز اطلاعات بسته‌ها به یک دید کلی به شبکه می‌رسد. اعمال کردن تکنیک‌های یادگیری ماشینی به شبکه‌های نرم‌افزار محور به دلایل زیر برای شبکه مفید است[۱۱]:

- پیشرفت‌های اخیر در حوزه تکنولوژی‌های computing مانند GPU و TPU امکانات زیادی را برای آسانی در اعمال تکنیک‌های یادگیری ماشینی فراهم می‌کند.
- داده جزء مهم‌ترین عوامل در کارهای مربوط به یادگیری ماشین می‌باشد. از آنجا که در شبکه‌های نرم‌افزار محور کنترل‌کننده به صورت متمرکز است و یک دید کلی به شبکه وجود دارد، بنابراین می‌تواند تمام داده‌ها را دیده و جمع‌آوردی کند. در نتیجه برعکس شبکه‌های مرسوم که ساختاری توزیع‌شده داشتند، در شبکه‌های نرم‌افزار محور می‌توان دیتا را جمع‌آوری کرده و سیستم را با کمک الگوریتم‌های یادگیری ماشین آموزش داد.
- به دلیل اینکه شبکه‌های نرم‌افزار محور قابلیت برنامه‌ریزی دارند، برای همین می‌توان پاسخ‌های بهینه برای شبکه(مثل مسیریابی بهینه، تخصیص بهینه منابع یا ...) را به کمک الگوریتم‌های یادگیری ماشین، به صورت real time بدست آورد.

2. شبکه نرم‌افزار محور

در دهه گذشته، به دلیل شبکه‌های نرم‌افزار محور، در حوزه شبکه، موجی جدید از نوآوری شروع شد. در ابتدا، به طور کلی یک پروتکل بود(OpenFlow) که Data plane و Control plane را جدا می‌کرد که این امر سبب شکوفا شدن پروتکل‌ها و طراحی‌های شبکه جدیدی شد. اگر چه خیلی زود به یک روند معماری جدیدی تبدیل شد که که کلیدها (Data Plane) توسط یک قسمت مرکزی، کنترل کننده شبکه نرم افزار محور(Control Plane)، از طریق پروتکل OpenFlow مدیریت شد. البته جدا کردن این دو ایده جدیدی نبود اما شبکه نرم‌افزار محور این مهم را توانست وارد بازار سخت‌افزار کرده و تجاری سازی کند. علاوه بر این، شبکه‌های نرم افزار محور فرصت‌های زیادی برای همکاری محققین و شرکت‌های سازنده ایجاد کرد. طبق تعریف، شبکه‌های نرم افزار محور، پیچیدگی شبکه را پنهان می‌کند. معماری آن، که در شکل ۱ آمده است، کنترل پویا، مقرون به صرفه، قابل مدیریت و سازگار با شبکه را تامین می‌کند. تعریف دیگر برای معماری شبکه نرم‌افزار محور در شکل زیر نشان داده شده است که شامل چهار plane می‌باشد.



شکل ۴: معماری، توابع و روابط plane های شبکه نرم افزار محور[۱]

در پایین این معماری، Data Plane وجود دارد که شامل مجموعه‌ای از دستگاه‌های شبکه(فیزیکی یا مجازی) است که ترافیک داده را انتقال می‌دهد. Data Plane فریم‌های جدیدی که رسیده‌اند را طبق منطق Control Plane بررسی می‌کند

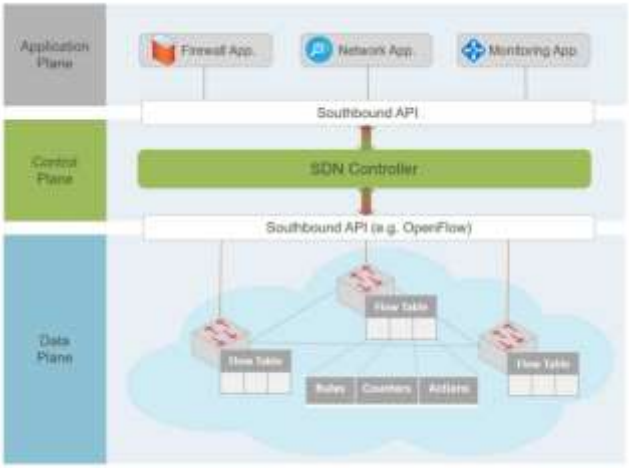
Control Plane مغز شبکه است که مسئول تصمیم‌گیری‌هایی مثل مسیریابی، سیگنالینگ‌های ترافیک است. ارتباط این دو plane از طریق Southbound Interface (SBI) انجام می‌شود که در ابتدا از OpenFlow طبعیت می‌کرد، اما امروزه از پروتکل‌های دیگری مثل P4Runtime استفاده می‌شود.

در بالای آن، Application Plane از طریق Northbound Interface (NBI)، معمولا به صورت ناهمزمان(مثل Rest API)، متصل شده است. برخی محققین Application Plane و Control Plane را جدا در نظر می‌گیرند اما برخی دیگر یکی در نظر می‌گیرند. ملاکی که این دو Plane را جدا می‌کنند این است که Control Plane شامل هسته عملکردهای شبکه‌ای است(مثلا پیدا کردن توپولوژی، محاسبه کوتاه‌ترین مسیر و ...) در حالی که Application Plane، application های جداگانه‌ای است که از Control Plane برای اجرا استفاده می‌کند.

در پایان، Management Plane نیز برای این است که وسیله‌ای برای مدیریت شبکه برای جنبه‌هایی مثل پیکربندی، نظارت، کارهای مربوط به صورت حساب و ... می‌باشد[۳]. در این معماری، ناهمگون‌ترین Plane همین Management Plane است و چالش‌های گوناگونی را در بر می‌گیرد[۴].

به طور خلاصه، مزیت اصلی شبکه‌های نرم‌افزار محور این است که امکانات جدیدی را برای کنترل شبکه‌های متمرکز فراهم می‌کند. مثلا، با کمک شبکه‌های نرم‌افزار محور، به دلیل ماهیت مجازی control و data plane، کاربران می‌توانند برای یک مکان خاص، به المان‌های فیزیکی و مجازی دسترسی پیدا کنند. همچنین، در شبکه‌های نرم‌افزار محور، مدیران می‌توانند همه چیز را به صورت متمرکز مدیریت کنند که با کمک آن می‌توان دیدی کلی برای مدیریت شبکه داشت در حالی که در شبکه‌های سنتی این گونه نیست.

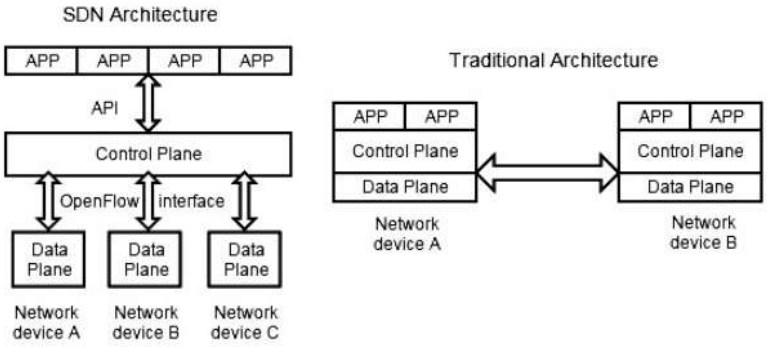
با توجه به ماهیت متمرکز سیستم‌های نرم‌افزار محور، استفاده از از تکنیک‌های یادگیری ماشین، که جزء روش‌های متمرکز است، در حال افزایش است.



شکل ۵: معماری SDN [۱۳]

2-1- *مقایسه شبکه‌های نرم‌افزار محور با شبکه‌های قدیمی*

حال که با معماری شبکه نرم‌افزار محور آشنا شدیم، می‌توانیم از منظری دیگر معماری شبکه‌های معمولی را با معماری شبکه‌های نرم‌افزار محور مقایسه کنیم که این مقایسه، در شکل زیر نشان داده شده است:



شکل ۶ :مقایسه معماری شبکه نرم‌افزار محور و شبکه‌های قدیمی[۱۴]

2-2- *کاربردها و چالش‌های مسیریابی در شبکه‌های نرم‌افزار محور*

مسیریابی بهینه جزء اهداف اصلی شبکه‌های کامپیوتری می‌باشد. به طور خاص، این هدف مستقیما به مهندسی ترافیک شبکه مرتبط است. زیرا این حوزه بر اساس یک هدف خاص شکل گرفته است: ترافیک دقیقا بر اساس تقاضای ترافیکی مسیریابی شده است. بنابراین می‌توانیم ادعا کنیم که مهندسی ترافیک نوعی از انواع مختلف بهینه‌سازی مسیریابی است. به علاوه، این نیازهای ترافیکی، بسته به اینکه ترافیک داده یا ترافیک کنترلی را در نظر بگیریم، متفاوت هستند. در همین حوزه، منطق متمرکز دیدن کنترل کننده شبکه نرم‌افزار محور، از زوایای دید مختلفی، بهتر و راحت‌تر از شبکه‌های سنتی می‌باشد. به عنوان مثال، توپولوژی گراف‌ها می‌توانند به آسانی از شبکه استخراج شوند و الگوریتم‌های کوتاه‌ترین مسیر مثل دایکسترا، به طور موثر و پویا می‌توانند حساب شوند تا بهترین مسیرها پیدا شوند. این موضوع باعث کاربرد مستقیم الگوریتم‌های علوم کامپیوتر در شبکه‌های کامپیوتری شده است[۵] بدون اینکه نیاز باشد که این الگوریتم‌ها را به پروتکل‌های توزیع شده تبدیل کنیم. مثل تولید مسیرهای جدا از هم برای مدیریت ترافیک که امروزه از هر زمانی راحت‌تر است[۶]. در نتیجه با توجه به ویژگی‌های مفید شبکه‌های نرم‌افزار محور، مسیریابی را می‌توان بر اساس انواع پارامترها مثل مسیریابی بهینه(کوتاه‌ترین مسیر، کوتاه‌ترین مسیر محدود شده و ...)، توابع هزینه، منابع و ... انجام داد. این امر سبب آسان شدن انطباق و سازگاری و گسترش آسان برای سناریوهای خاص می‌شود[۵].

با وجود اینکه شبکه‌های نرم‌افزار محور جوابی بهینه برای گسترش تکنولوژی ارتباطات و اطلاعات، تامین کننده‌های ابری و ... هستند، اما یک سری چالش‌هایی دارند[۷] که برخی از آن‌ها عبارت‌اند از:

- مکان کنترل‌کننده: شبکه نرم‌افزار محور یک کانال ارتباطی جدید بین data plane و control ایجاد می‌کند که ممکن است کاملا واضح نباشد(به خصوص در شبکه‌های بزرگ که ارتباط out-of-band شاید ممکن نباشد) بنابراین مکان دقیق کنترل کننده باید مشخص باشد.
- مقیاس‌پذیری: مقیاس‌پذیری با مورد قبلی کاملا مرتبط است زیرا شبکه نرم‌افزار محور به صورت منطقی متمرکز می‌باشد و در نتیجه مدیران شبکه باید تصمیم بگیرند که کنترل تا چه حدی باید به کنترل‌کننده داده شود تا بتوانند از مشکلات مقیاس‌پذیری جلوگیری کنند.
- امنیت: از آنجا که شبکه‌های نرم‌افزار محور منطقی متمرکز دارند، می‌توان به آسانی مورد مشکلاتی قرار گیرد.
- قابلیت اطمینان: مشابه شبکه‌های سنتی، قابلیت اطمینان چالش مهمی است. اما در شبکه‌های نرم‌افزار محور به دلیل کانال کنترلی که یک نقطه

با پتانسیل نفوذ و شکست است، خیلی چالش جدی‌تری می‌باشد.

یکی از نتایج این است که کنترل کننده شبکه‌های نرم‌افزار محور باید به طرز درستی چیده شوند که از اشتباهات دستی جلوگیری شود. برای حل دیگر مشکلات نیز، تحقیقات باید روی کنترل کننده‌های توزیع شده برای شبکه‌های نرم‌افزار محور (همراه با تضمین امنیت) تمرکز کند. در حال حاضر، از بین تمام کنترل‌کننده‌های شبکه‌های نرم‌افزار محور[۸]، می‌توان به دو مورد از آنها اشاره کرد: Ryu و ONOS

2-3- *یادگیری ماشین در محیط شبکه‌های نرم‌افزار محور*

اگر چه یادگیری ماشین(و به طور کلی تر هوش مصنوعی) حدود دو دهه است که در شبکه استفاده می‌شود اما استفاده از آن در عمل هنوز در مراحل ابتدایی است[۹]. به لطف نرم‌افزاری شدن شبکه‌ها، استفاده از هوش مصنوعی و یادگیری ماشین در شبکه‌ها آسان‌تر شده است.

2-4- *تکنیک‌های یادگیری ماشین*

در بررسی‌هایی که انجام می‌دهیم یادگیری ماشین را به سه دسته تقسیم می‌کنیم: یادگیری نظارت شده، یادگیری نظارت نشده و یادگیری تقویتی که هر کدام نیز انواعی دارند که در شکل زیر بررسی شده‌اند:

شکل ۷: دسته‌بندی تکنیک‌های یادگیری ماشین

تقریبا تمام تحقیقات این حوزه از ترافیک داده و سیگنال داده برای امنیت نیز استفاده می‌کنند. برای امنیت نیز بحث‌های مسیریابی اهمیت دارد[۱۵]

3. تکنیک‌های یادگیری ماشین برای بهینه‌سازی مسیریابی در شبکه‌های نرم‌افزار محور

در این قسمت ما الگوریتم‌های یادگیری ماشین را در سه دسته کلی که در بخش قبل گفتیم برای بهینه‌سازی مسیریابی در شبکه‌های نرم‌افزار محور تقسیم می‌کنیم.

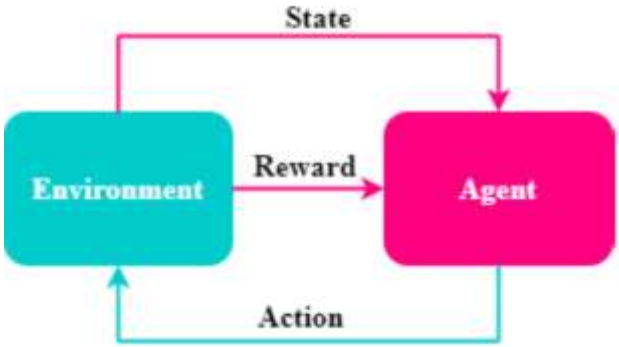
برای مسیریابی می‌توان از روش های یادگیری نظارت شده، یادگیری نظارت نشده و نیز یادگیری تقویتی استفاده کرد. از بین این سه نوع روش یادگیری ماشینی، اینکه کدام روش بهتر است به یه سری پارامترهایی بستگی داره که عبارت‌اند از:

- نوع دیتاست: در سناریوهایی که دیتاست برچسب دار در دسترس است امکان استفاده از روش های یادگیری نظارت شده وجود دارد که معمولا نتایج دقیقتری میدهند. البته اکثر کارهایی که برای مسیریابی در شبکه‌های نرم‌افزار محور با استفاده از یادگیری نظارت شده انجام شده از دیتاست های شبیه سازی شده استفاده شده است و روندهای محدودی از دیتاست واقعی استفاده کرده اند. البته چون بیشتر اوقات دیتاست برچسب دار نداریم به همین دلیل گاهی از یادگیری نظارت نشده استفاده می‌شود. از یادگیری تقویتی نیز در مسائل بهینه سازی پویا مثل همین بهینه سازی مسیریابی نیز استفاده میشود
- اندازه دیتاست: دیتاست های بزرگ برای روشهایی مثل ANN یا DNN مناسبتر است و برای دیتاست هایی که خیلی بزرگ نیستند

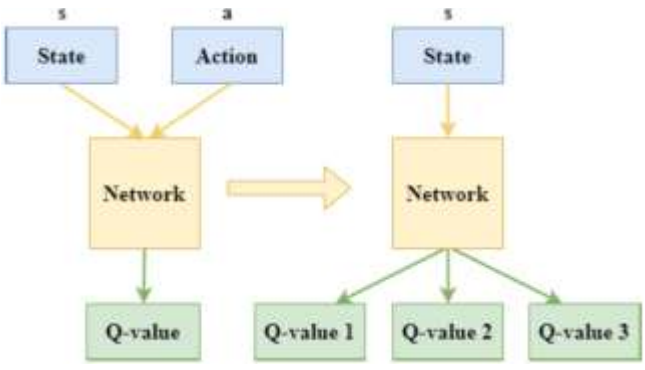
روش هایی مثل SVM احتمالا مناسب‌تر باشند. برای دیتاست های کوچک نیز روش هایی مثل regression بیشتر استفاده شده است.

- نوع مساله بهینه سازی: و در پایان، سومین پارامتری که در تعیین الگوریتم یادگیری ماشینی موثر است، نوع مساله بهینه‌سازی می‌باشد.

در بخش بررسی یادگیری تقویتی، ما در زیر بخشی از این بخش، به بررسی یادگیری تقویتی عمیق می‌پردازیم. جهت توضیح کلی این دو روش، در دو شکل بعدی، به ترتیب روند کلی یادگیری تقویتی و یادگیری تقویتی عمیق نشان داده شده است:



شکل ۸: یادگیری تقویتی



شکل ۹: یادگیری تقویتی عمیق

حال به بررسی چندتا از روشهای یادگیری ماشینی برای مسیریابی می‌پردازیم.

3-1- *یادگیری نظارت شده*

NeuRoute: یک فریم‌ورک برای مسیریابی پویا برای شبکه‌های نرم‌افزار محور است که از یادگیری ماشین برای حل مساله Maximum Throughput minimum cost dynamic routing استفاده می‌شود که در زمانی کم‌تر به جوابی مشابه با روش‌هایی که در پیش از ارائه این روش مطرح شده بود می‌رسد. NeuRoute از شبکه‌های عصبی برای یادگیری ویژگی‌های ترافیکی استفاده می‌کند. دو بلوک اصلی این روش بر اساس DNN می‌باشد:

- Traffic Matrix Predictor: یک LSTM است که مرحله‌ی بعدی را پیش‌بینی می‌کند.
- Traffic Routing Unit: با FNN طراحی شده است و اینکه چگونه درخواست ترافیکی را با مسیرهای مسیریابی شده، match کند را یاد می‌گیرد.

علاوه بر NeuRoute، روش دیگری که از یادگیری نظارت شده استفاده می‌کند، یک سیستم load balance resolution ارائه می‌دهد که به دلیل دید کلی به شبکه نرم‌افزار محور، برای broadcasting بهتر است. در شبکه‌های قدیمی دید کلی به شبکه وجود ندارد و routerها فقط مقصد و hop بعدی را دارند. اما در این روش کنترل‌کننده شبکه نرم‌افزار محور، همه‌ی مسیرهای بین مبدا و مقصد را یافته و سپس یک تراز کننده بار را اعمال می‌کند که ترافیک به صورتی بهینه توزیع می‌کند. این ترازکننده بار در لحظه تمام مسیرها را بررسی می‌کند و بعد از انتخاب مسیر بهینه، شبکه نرم‌افزار محور برای کلیدهای OpenFlow جداول flow را اختصاص می‌دهد. برای انجام این کار، در این روش از ANN با یک لایه پنهان استفاده می‌شود که ۴ ویژگی بار را به عنوان ورودی

می‌گیرد که عبارت‌اند از: bandwidth utilization ratio – packet loss rate – transmission latency – transmission hop

حال برخی از روش‌های یادگیری نظارت شده که در شبکه‌های نرم‌افزار محور کاربرد دارند را معرفی می‌کنیم:

KNN(K-Nearest Neighbors): در این روش فاصله‌ی بین مقادیر ویژگی‌های مختلف محاسبه می‌شود. اگر K تا نمونه که نزدیک‌ترین فاصله را دارند به یک دسته خاصی تعلق داشته باشند، آنگاه این نمونه خاص به آن دسته تعلق خواهد داشت. نتایج این طبقه‌بندی فقط به تعداد نمونه کمی که در اطراف آن هستند بستگی دارد. پیاده‌سازی این روش ساده بوده و به داده‌های پرت حساسیت کمتری دارد. همچنین این روش برای دسته‌بندی‌های چند کلاسه نیز کاربرد دارد. اگر چه برای داده‌های بزرگ از نظر زمانی به صرفه نیست[۱۶]. Owusu برای طبقه‌بندی ترافیک در شبکه‌های نرم‌افزار محور که از IOT استفاده می‌کنند برای رسیدن به این مقصود از سه روش Random Forest Classifier و Decision Trees Classifier و KNN استفاده کرده و نتایج را مقایسه کرده‌اند. همچنین از دو روش انتخاب ویژگی SFS و SHAP نیز استفاده کرده‌اند و بهترین نتیجه برای Random Forest Classifier با SFS بود که دقتی برابر با 0.83 داشت.

SVM(Support Vector Machines): این روش، تعمیم طبقه‌بند خطی است که در حالت نظارت شده طبقه‌بندی دوتایی را انجام می‌دهد. مرز تصمیم‌گیری برابر حداکثر مرز ابرصفحه داده‌های آموزشی است. این روش پایدار است زیرا در این روش بهینه‌سازی، هم ریسک empirical و هم ریسک structural مینیمم می‌شود. البته باید توجه شود که این روش فقط برای موارد طبقه‌بندی دوتایی استفاده می‌شود و برای طبقه‌بندی چند کلاسه باید از روش‌های دیگر بهره جست. اگر در کنترل‌کننده شبکه نرم‌افزار محور از SVM استفاده شود، پیچیدگی SVM، تاثیر خیلی کمی روی شبکه نرم‌افزار محور خواهد گذاشت[۱۶].

DT(Decision Tree): DT یک مدل predictive می‌باشد که ارتباط بین اشیا خواص و اشیا مقادیر را نشان می‌دهد. این روش یک ساختار درختی دارد که هر نود داخلی در درخت یک شی را نشان می‌دهد که هر مسیر شاخه یک مقدار صفت احتمالی را نشان می‌دهد. کاربرد اصلی این روش در شبکه، برای طبقه‌بندی بسته‌ها می‌باشد.

در مقایسه با SVM و KNN، مهم‌ترین ویژگی این روش این است که به سادگی قابل فهم بوده و به سرعت می‌توان آن‌را روی شبکه اجرا کرد. همچنین آماده کردن داده برای این روش ساده است و گاهی حتی نیاز هم نیست. البته باید توجه داشت که اگر تعداد کلاس‌ها زیاد شود ممکن است در این روش خطا به سرعت افزایش یابد[۱۶].

Ensemble Learning: در الگوریتم‌های روش یادگیری نظارت شده، هدف، یادگیری یک مدل پایدار است که در تمام جنبه‌ها، حتی اگر حقایق خوب بیان نشده است، خوب عمل کند. Ensemble Learning ترکیبی از چندین روش ضعیف یادگیری نظارت شده است که با هم نتیجه و مدلی قوی‌تر ارائه می‌دهند. الگوریتم‌های اصلی این روش عبارتند از: bagging و boosting

تا اینجا ما برخی از روش‌های یادگیری نظارت شده را معرفی کردیم. برخی از روش‌های یادگیری نظارت شده دیگر که در مسیریابی استفاده می‌شود می‌توان به ANN، Linear Regression، DNN و Naïve Bayes اشاره کرد.

توجه شود که در استفاده از این روش‌ها برای یافتن مسیریابی بهینه، لزوما مسیری که کمترین تاخیر را ایجاد می‌کند انتخاب نمی‌کنیم. اینکه مسیریابی بهینه به چه معنیست باید گفت بستگی دارد. مثلا از نظر فیزیکی شبکه باید بتواند به دو بخش سیمی و بیسیم تقسیم شود و پروتکل های مسیریابی متفاوتی برای شروع نیاز دارند. به عنوان مثال latency و throughput پارامتر های معتبری برای اندازه گیری کیفیت

مسیریابی در محیط سیمی هستند اما در سناریو های بیسیم، مثلا در حالت های توان پایین شاید به توان پایین نیاز بیشتری داشته باشند. بعلاوه، توپولوژی شبکه نیز بسته به کاربرد متفاوتند. مثال مسیریابی بهینه در مراکز داده ممکن است با برخی دیگر از شبکه ها متفاوت باشد. همچنین شبکه‌ها پویا هستند و تغییر می‌کنند.

حال بسته به این تفاوت در تعریف مسیریابی بهینه می‌توان با توجه به معیارهای مختلفی از الگوریتم‌های روش‌های یادگیری نظارت شده استفاده کرد. برخی از این معیارها که در مقالات از آن‌ها برای بهینه‌سازی مسیریابی از یادگیری نظارت شده استفاده کرده‌اند عبارتند از:

- حداکثر throughput و حداقل هزینه
- پاسخ Load balance با دید کلی به شبکه
- کمترین احتمال تراکم
- بهترین عملکرد
- پیش‌بینی ماتریس ترافیک
- مدیر پهنای باند وفقی
- جفت افزایش داده شده از نظر هر فرستنده یا مقصد و میانگین تاخیر و تخمین jitter
- بهینه سازی ماتریس پیش‌بینی ترافیک
- QoE افزایش داده شده
- تاخیر کم و انعطاف‌پذیری بیشتر
- مسیریابی flow بهینه
- مسیریابی چند مسیره با محدودیت‌های QoS و قوانین برای محدودیت flow
- پیش بینی پهنای باند
- و...

حال در شکل زیر، روش‌های یادگیری نظارت شده برای این پارامترهای بهینه‌سازی گفته شده نشان داده شده است:

Techniques	Objective	Implementation & Evaluation	Advantage	Disadvantage
ANN	Maximum throughput & minimum cost	POX controller + Mininet. GÉANT network topology and traffic	Fast execution. Min cost. Max throughput	Large networks and datasets not tested yet
ANN	Load balance solution with global network view for SDN	Floodlight controller + Mininet	Enhanced bandwidth utilization ratio, transmission latency, packet loss rate and transmission hop.	DLB strategy not select best path, ignores the load condition, global path lead to the bad impacts.
ANN	Minimum congestion probability	Ryu controller + Mininet	Improvements in the average throughput, packet loss ratio, and packet delay versus data rate	Simplicity of the layout and the model. The model is not scalable.
ANN + Evolutionary (HIA)	Maximum performance	POX controller + Mininet, and MATLAB	Cost effective, time effective, good performance index	It lacks proper / reproducible implementation details
LSTM-RNN	Traffic matrix prediction	POX controller. GÉANT network topology and traffic	Successfully applied. Best suited for sequence labeling task and sequence modeling	Traditional non-linear prediction models (ARIMA, ARAR, HW) cannot meet the accuracy requirements
ARIMA, LSTM	Adaptative bandwith manager	Floodlight controller + Mininet	Dynamic changes of QoS policy when the traffic flood the forwarding elements	The time series forecasting is an optional module
GNN	Enhanced per-source/destination pair mean delay and jitter estimation	OMNeT++. GÉANT, NSFNet, 50-node Germany50 topologies	Significant delay and jitter reduction	Large amount of data
Logistic Regression	Optimized routing. Traffic matrix prediction	ONOS controller + Mininet	Improves shortest path algorithm. Dynamically reduces network congestion.	Real datasets are needed for advance models and predictions for industrial applications

Techniques	Objective	Implementation & Evaluation	Advantage	Disadvantage
NSGA-II (Genetic algorithm)	Multi-objective: 1) minimize path delay, 2) maximize path reliability	POX controller + Mininet	Optimal paths for each type of traffic (UDP or TCP). Focus on real AI-based network applications (IoT and fog computing)	Missing comprehensive evaluation using different network topologies (only one fixed custom topology is tested).
Random Forest, Decision Trees, K-nearest neighbors	Classify traffic in SDN-IoT networks	Dataset from a real-world ToR network	High accuracy rates above 0.8 with six features	Lack of comparison with any ANNs. Accuracy rates above 0.9? Lack of detail in the SDN implementation
ARIMA, SVR, Decision Trees, Linear Regression, Random Forest	Bandwidth prediction	Ryu controller + Mininet. GENI testbed	Simplicity of the regressors. Real traffic traces	Missing comparison with the DNN-based regressor
Q-learning, Genetic algorithm, Particle swarm optimization, Hidden Markov model	Architectural design for load balancing and segment routing	Theoretical analysis	It compares four supervised and reinforcement learning techniques	Simple architectural design. No thoughts on implementation details and implications
Linear Regression	Enhanced QoE	Theoretical analysis based on the SDN architecture	Best management strategy and performance. Ensures user requirements are met	Missing practical implementation and dataset
MACCA2-RF&RF	Intelligent routing by leveraging flow classification and avoiding congested links with local routing	Floodlight controller + Mininet. Moore and Li datasets	It can accurately classify flows to their obtain QoS requirements. Local routing adapts to provided QoS.	Evaluated with a relatively old dataset. Requires many entries in the SDN tables.
Random forest	Managing IP and SDN-enabled optical networks	Theoretical proof-of-concept study	Cost effective, better accuracy, enhanced robustness and dynamic capacity.	Missing practical implementation and dataset
Naive Bayes	Reduced delay and enhaced resilience	Ryu controller + Mininet-WiFi	Delay reduction compared to Q-learning, multipath, and OLSR routing protocols	Simplistic layout and model. Requires much data
DNN	Optimized flow routing	P4 switches	Low average delays achieved. It uses programmable P4 switches	Missing detailed implementation
DNN	Multipath routing framework with QoS constraints and flow rule space constraints	Keras (TensorFlow). TOTEM toolbox	High prediction accuracy of the heuristic routing solution and low computation time	Missing comparison with other algorithms. No thoughts on SDN implementation details and implications

Techniques	Objective	Implementation & Evaluation	Advantage	Disadvantage
K-means	Minimum privacy risk, while achieving compliance requirements of data transmission	Ryu controller + Risk simulations in Python	Provides privacy and risk compliance with low complexity	Delays in communication
K-means / Vector Space Model (cosine similarity)	Least congested route for routing traffic	Ryu controller + Mininet	Best Round Trip Time in comparison to Dijkstra	Simplistic network layout

شکل ۱۰: مقایسه روش‌های یادگیری نظارت نشده برای مسیریابی[۱]

3-2 *یادگیری نظارت نشده*

در روش‌های یادگیری نظارت نشده، برچسب اطلاعات و داده‌های آموزشی شناخته‌شده نیست. هدف یافتن اطلاعات ذاتی و قوانین داده از طریق بررسی داده‌های آموزشی بدون برچسب است. پراستفاده‌ترین روش استفاده شده clustering نام دارد و ساده‌ترین و معروف‌ترین الگوریتم K-means است[۱۷].

در clustering، داده‌ها به زیرمجموعه‌های مختلفی تقسیم می‌شوند که هر کدام از آن‌ها یک cluster نام دارد که یک دسته است. همچنین باید توجه شود که clustering داده نمی‌داند که قبل از آن به کدام دسته تعلق دارد.

روش‌های عادی مسیریابی برای شبکه‌های نرم‌افزار محور معمولا برای موارد مربوط به حریم خصوصی مناسب نیستند. به خصوص که مسیرهای شبکه نرم‌افزار محور معمولا ایستا هستند یا به طور خاص برای یک flow ارتباطی تعیین می‌شوند که برای حمله‌های امنیتی مثل DoS مناسب نیستند. از آنجا که تعداد زیادی از packetها از یک مسیر داده یکسان ارسال می‌شوند برای همین آن مسیر به عنوان مسیر ریسکی شناخته می‌شود. در نتیجه از route randomization استفاده می‌شود. به این ترتیب که از یادگیری ماشین برای محاسبه ریسک در شبکه نرم افزار محور استفاده شده و از مسیریابی توزیع شده با استفاده از الگوریتم‌های swarm استفاده می‌شود و در نهایت با مینیمم کردن route randomization و ریسک کردن برای رسیدن به privacy مورد نظر استفاده می‌شود. در این روش از بسته داده قبلی برای آموزش استفاده می‌شود و با کمک آن‌ها بسته‌های داده جدید مسیریابی می‌شوند. برای شناسایی ریسک نیز از الگوریتم کلاسترینگ k-means استفاده می‌شود.

در شکل زیر دو روش یادگیری نظارت نشده که برای مسیریابی با استفاده از یادگیری نظارت نشده استفاده شده است را مقایسه می‌کنیم:

3-3 *یادگیری تقویتی*

یادگیری تقویتی یک روش هدایت با جایزه است که یک عامل در یک حالت آزمون و خطا یاد گرفته و از طریق ارتباط با محیط جایزه می‌گیرد. هدف یک سیستم یادگیری تقویتی این است که پارامترها را به صورت پویا تنظیم کند طوری که حداکثر سیگنال تقویتی را دریافت کند.

سیگنال تقویتی‌ای که از محیط دریافت می‌شود یک ارزیابی خوب یا بد را از عملی که انجام داده شده می‌دهد.

یادگیری تقویتی به طور کلی برای بهبود انعطاف‌پذیری و مقیاس‌پذیری استفاده می‌شود.

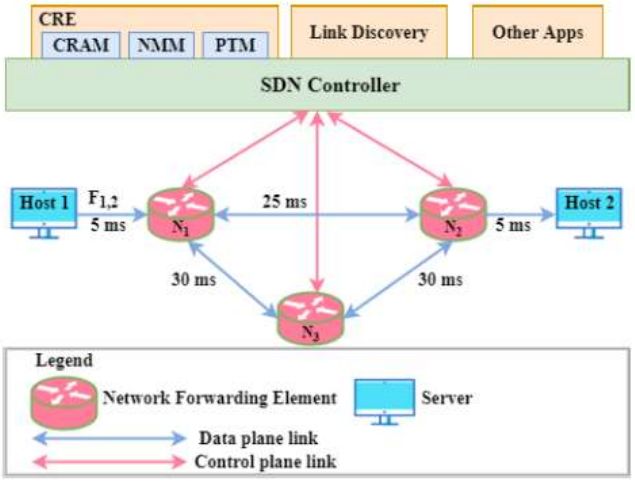
الگوریتم‌های یادگیری تقویتی به طور کلی برای حل مسائل تصمیم‌گیری استفاده می‌شوند. زمانی که از یادگیری تقویتی برای حل مساله مسیریابی بهینه استفاده می‌شود، کنترل‌کننده به عنوان یک عامل است و شبکه نیز به عنوان یک محیط می‌باشد. فضای حالت نیز از شبکه و ترافیک تشکیل شده است. عمل هم جواب مسیریابیست. جایزه نیز بر اساس متریک‌های بهینه‌سازی مثل تاخیر تعیین می‌شود.

در شکل زیر شش روش یادگیری تقویتی که برای مسیریابی با استفاده از یادگیری نظارت نشده استفاده شده است را مقایسه می‌کنیم:

Techniques	Objective	Implementation & Evaluation	Advantage	Disadvantage
SARSA	Time-efficient and QoS-aware routing in large scale SDN	Simulated scenario + OpenFlow-compliant. It outperforms conventional Q-learning	Time-efficient, minimum signal delay	Missing experiments with fully SDN-compliant networks, including a controller
Q-learning	Model-free proposal. Routing paths in its state-action space	Ryu SDN controller + Mininet	Multipath routing and reduced latencies. Comprehensive evaluation	Still lacks scalability in large scenarios
Q-learning	Use of the Knowledge Plane concept. Best throughput, loss ratio, and delay + Obtaining best set of shortest paths	Ryu controller + Mininet. GÉANT network topology and traffic	Best metrics and enhanced set of shortest paths in comparison with Dijkstra. Very complete implementation and evaluation	Application to commercial SDN solutions (e.g. ONOS) would be desirable
Q-learning + Deep Q-learning	Improved network performance based on QoS	Simulated scenario + RIP protocol. After a certain training period, the algorithm can find a route with better QoS efficiency with almost 100 per cent accuracy	Better QoS connection and stronger link selection trend	The specific features must be designed manually, which is not trivial. No integration in real SDN scenarios
Unspecified	Improved network performance with decision-making based on QoS and security	No controller (distributed, OSPF) + Mininet. Better jitter performance than delay results	Routing based on the open source tool Quagga, hence easily reproducible	Missing experiments with fully SDN-compliant networks, including a controller
Data-driven model	Development of a data-driven model for routing optimization	Theoretical analysis + Results via simulation	Minimizes routing link utilization.	No integration with SDN or additional discussion about it.

شکل ۱۱ :مقایسه تکنیک‌های یادگیری تقویتی برای مسیریابی[۱]

مشابه چیزی که در یادگیری نظارت شده نیز گفتیم، در اینجا نیز بسته به هدف مسیریابی، پارامترهای خاصی را باید بهینه‌سازی کنیم.



شکل ۱۲ : معماری CRE که کارایی شبکه را با جمع‌آوری حالات شبکه بر اساس ملزومات QoS، افزایش می‌دهد

روش‌های یادگیری تقویتی مثل Q-Learning دو مشکل دارند. اولا نمی‌توانند بطور مستقیم استفاده شوند زیرا به Q table بزرگی نیاز دارند. دوما در گسسته زمان استفاده می‌شوند که برای تغییرات پویا مناسب نیست. برای حل این دو مشکل از یادگیری تقویتی عمیق استفاده می‌کنیم. DROM یک الگوریتم بهینه‌سازی مسیریابی برای شبکه‌های نرم‌افزار محور است که از یادگیری تقویتی عمیق برای این منظور استفاده می‌کند[۱۰].

DROM چهار مزیت دارد:

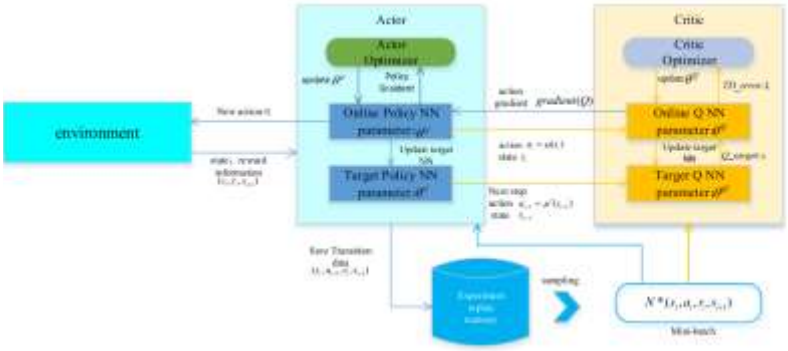
- DROM با تنظیم جایزه، پارامترهای پویای شبکه را تعیین می‌کند.
- DROM به جای استفاده از Q-table ها از شبکه‌های عصبی استفاده می‌کند و به همین دلیل سربار حافظه و هزینه زمانی آن را کاهش می‌دهد.
- چون DROM برای شبکه شرایط خاصی را در نظر نمی‌گیرد و به همین دلیل می‌تواند برای اکثر موارد استفاده شود.
- DROM برعکس روش‌های یادگیری تقویتی می‌تواند برای زمان پیوسته نیز استفاده شود.

قاعده کلی DDPG:

همانطور که گفتیم، یادگیری تقویتی عامل‌ها را قادر می‌کند تا از اقدامات در محیط با استفاده از جایزه آموزش ببینند تا به مرور این جایزه را تا جای ممکن حداکثر کنند. یادگیری تقویتی از قابلیت perceptron یادگیری عمیق و قابلیت تصمیم‌گیری یادگیری تقویتی استفاده می‌کند. اگر چه یادگیری تقویتی عمیق که بر اساس مقدار عمل می‌کند، مثل DQN، نمی‌توانند اعمال پیوسته را کنترل و مدیریت کنند که برای شبکه‌های پویا و real time مناسب نیست.

روش‌های یادگیری تقویتی بر اساس سیاست مثل DPG، می‌توانند اعمال پیوسته را نیز کنترل و بهینه‌سازی کنند اما این‌ها فقط توابع سیاست و خط مشی را برای توابع خطی را تعیین می‌کنند و مشکلاتی از قبیل بیش برازش(به دلیل هم‌بستگی داده‌ها) دارند. Deep Mind گوگل این مشکل را به

روشی حل کرده است که در ادامه می‌گوییم. DeepMind یک روند یادگیری تقویتی عمیق به نام DDPG ارائه داده است که با استفاده از شبکه‌های عصبی، توابع Q و توابع استراتژی را ساخته و با کمک آن‌ها یک عمل گسسته پایدار و کارا ایجاد می‌کند. شکل زیر فریمورک DDPG را نشان می‌دهد که میو و Q به ترتیب تابع مشخصه استراتژی و تابع Q هستند.



شکل ۱۳: فریمورک DDPG[۱]

عامل روش DPG و critic نیز DQN را اتخاذ می‌کنند. هر عامل شامل دو شبکه عصبی است. اولی شبکه عصبی آنلاینیست که برای آموزش کاربرد دارد و دومی شبکه هدف بوده و برای بلاک کردن همگرایی داده‌های آموزشی استفاده می‌شود. ساختار این دو مشابه‌اند ولی دومی پارامترهای قبلی شبکه آنلاین را استفاده می‌کند.

بروزرسانی پارامترهای DDPG شامل بروزرسانی عامل شبکه عصبی و critic می‌باشد و از backpropagation استفاده می‌شود. سیاست گرادیان با استفاده از فرمول زیر بدست می‌آید:

$$\nabla_{\theta^\mu} J = grad[Q] * grad[\mu] \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i} \quad (1)$$

برای اپدیت ماجول critic از شبکه DQN نیز خطای TD را محاسبه می‌کنیم. داریم:

$$L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2 \quad (2)$$

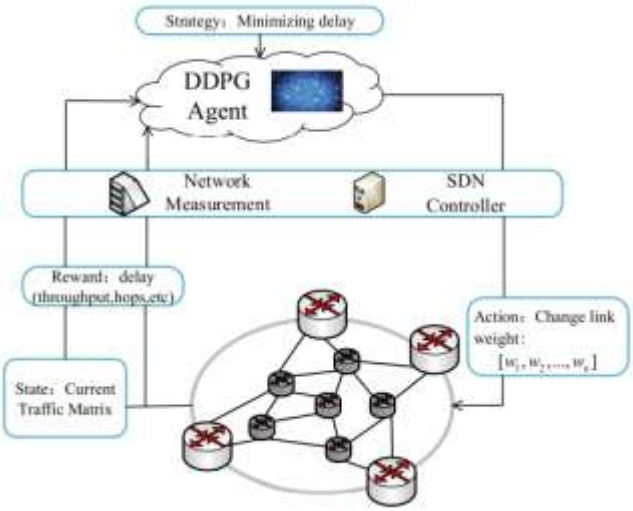
و در پایان، فرمول محاسبه y_i نیز به ترتیب زیر خواهد بود:

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'}) \quad (3)$$

حال می‌خواهیم مکانیزم DROM را بررسی کنیم.

مکانیزم DROM:

با اجرای DROM، پارامترهای عملکردی customize شده مثل تاخیر، ارسال طول مسیر، throughput به طور خودکار بهینه می‌شوند تا بتوانند کنترل real time را به صورت پیوسته بفهمند. شکل زیر، فریمورک DROM را نشان می‌دهد



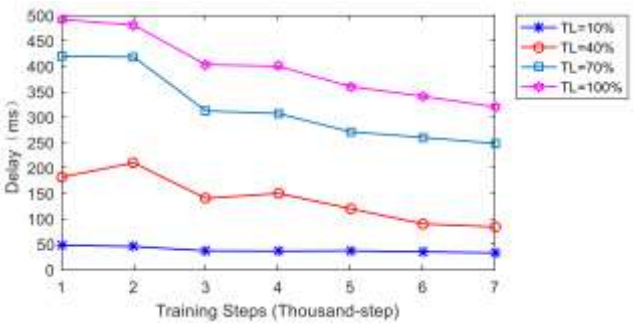
شکل ۱۴: فریمورک DROM

در این فریمورک عوامل DROM با محیط از سه طریق ارتباط می‌گیرند: حالت، اقدام و جایزه. از بین این‌ها، حالت s، ماتریس ترافیک بار شبکه فعلی(TM) است و اقدام a که توسط عامل به محیط انجام شده است، عبارت از تغییر وزن لینک‌های شبکه است. با تغییر وزن لینک‌ها، عامل می‌تواند مسیرهای data flow را تغییر دهد. جایزه r عامل، به عملکردهای شبکه و استراتژی نگهداری مرتبط است.

$$R_{i->j} = R(i->j|s_t, a_t) = -h(a_t) + \alpha delay_{ij} + \beta BW_{ij} + \gamma loss_{ij} + \theta TP_{ij} \quad (4)$$

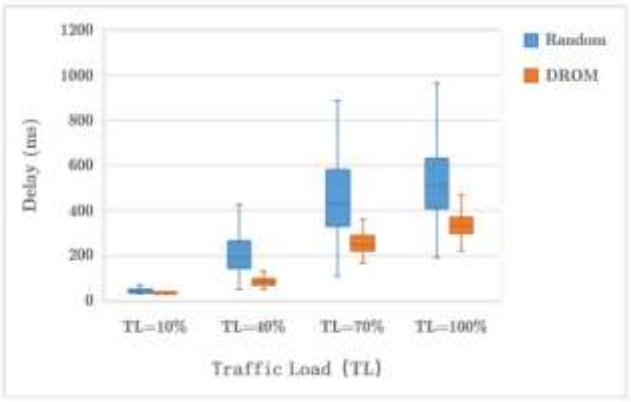
این رابطه نشان می‌دهد که شبکه در حالت s_t با اقدام گرفته شده r است. هدف از آموزش DDPG این است که اقدام بهینه a را بر اساس ورودی حالت s بیابد طوری که جایزه r ماکسیمم شود. فرآیند کلی DROM را می‌توان به صورت خلاصه این‌گونه بیان کرد: با تحلیل شبکه و اندازه‌گیری کنترل‌کننده شبکه نرم‌افزار محور، عامل DROM می‌تواند حالت شبکه s را یافته و عملی بهینه را تعیین کند: مجموعه‌ای از وزن لینک‌های $[w_1,w_2,\ldots,w_n]$. مسیرهای جدید flow مجددا از روی وزن‌های بروز شده ، محاسبه می‌شوند. کنترل‌کننده شبکه نرم‌افزار محور نیز قوانین جدیدی را تولید می‌کند تا مسیرهای جدید ساخته شوند. بعد از بروزرسانی مسیرها، جایزه r و حالت شبکه از روی آنالیز بعدی شبکه بدست می‌آید. عملکرد شبکه به صورت مکرری بهبود یافته و بهینه می‌شود.

نتایج عملکرد DROM از نظر همگرایی:



شکل ۱۵: تاخیر شبکه بر حسب مرحله‌های آموزشی برای بارهای ترافیکی مختلف[۱۰]

مزایای DROM از نظر performance:



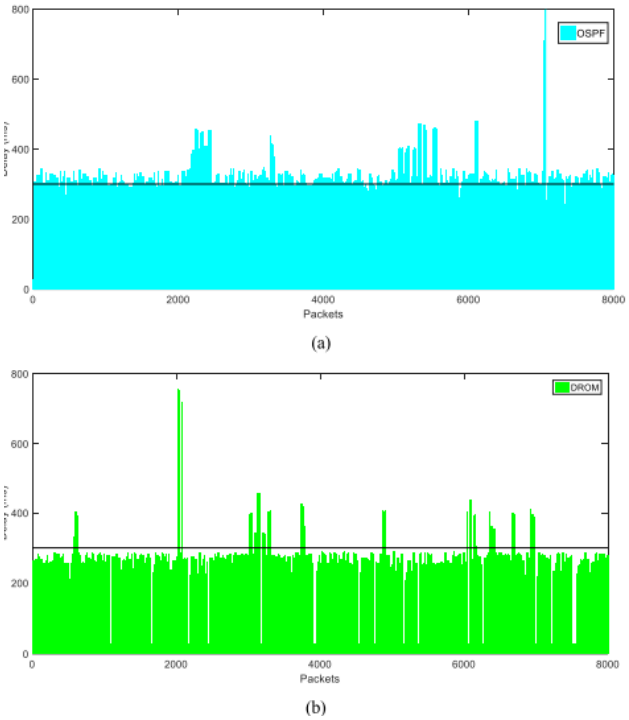
شکل ۱۶: مقایسه DROM و مسیریابی رندوم برای بارهای ترافیکی متفاوت[۱۰]

در این نمودار، قسمت بالایی و پایینی مستطیل به ترتیب نشان‌دهنده چارک بالا و چارک پایین تاخیر هستند و نیز خطی که در وسط مستطیل است نیز نشان‌دهنده‌ی میانه تاخیرها می‌باشد. برای سادگی داده‌های پرت نشان داده نشده‌اند.

همانطور که نتایج نشان می‌دهد، تاخیرهای DROM کمتر از چارک پایین تاخیر مسیریابی‌های تصادفی می‌باشد و حداقل تاخیر DROM نیز به بهترین جواب مسیریابی تصادفی بسیار نزدیک است. این نتایج نشان دهنده کارایی و موثر بودن DROM هستند.

هم‌چنین مقایسه DROM با OSPF نیز نتایج زیر را دارد:

DDPG	Enhanced routing based on a hybrid approach (distributed+centralized)	OMNeT++ simulator	Quick average delivery time. Promising architecture	Evaluated only via simulation. Huge amount of data and training iterations
DDPG	Content-aware traffic engineering for SDN	Event-driven simulator. GÉANT, NSFNET and BRITE-generated network topologies	Best throughput and bandwidth utilization compared to classic algorithms (e.g. shortest path)	Evaluated only via simulation
DDPG	Enhanced cluster stability and route selection method for routing in VANETs	OMNeT++ simulator + SUMO simulator	Improves delay, throughput and computational overhead.	Evaluated only via simulation. Lacks study including effects like driver behaviour, road conditions, and real-world scenarios.
DDPG + LSTM	Enhanced throughput and delay, focused on topology changes in space-ground integration networks	OMNeT++ simulator. CERNET+NSFNET topologies + 3-layer satellite network. Compared to OSPF	Better results than OSPF in terms of throughput and delay	Evaluated only via simulation
DDPG + Convolutionary Neural Networks	Reduced latency and packet loss rate	OMNeT++ simulator. BtEurope network	It admits diverse configuration as input parameters	Evaluated only via simulation. No comparison with other approaches is performed
DDPG + Convolution layer	Reduced latency and packet loss rate	OMNeT++ simulator. Compared to OSPF	It outperforms OSPF in terms of latency and packet loss	Evaluated only via simulation
DDPG-EREP	Optimized routing (no specific parameters involved)	Ryu controller + Mininet	Improves the original DDPG algorithm.	Slow reading of information on complex topologies. More tests should on more topologies and traffic workloads.



شکل ۱۷: مقایسه‌ی DROM با OSPF بر حسب بارهای ترافیکی مختلف. (a) تاخیر OSPF (b) تاخیر DROM [۱۰]

Techniques	Objective	Implementation & Evaluation	Advantage	Disadvantage
DDPG + Deep Q-network (DQN)	High performance routing in data center networks	OMNeT++ simulator. Fat-Tree topology. Compared to OSPF and TIDE [157]	It outperforms OSPF and TIDE in terms of throughput, flow completion time and load balance	Evaluated only via simulation
Deep Q-network (DQN)	High performance routing in data center networks, differentiating mice and elephant flows	Ryu controller + Mininet. Fat-tree data center topology. Compared to ECMP [174] and SRL+FlowFit [175]	It outperforms ECMP and SRL+FlowFit in terms of throughput, delay and packet loss	Missing a wider range of topologies. No detail about traffic matrices
Dueling Deep Q-learning (Dueling DDQN)	Computing path based on multiple QoS metrics (delay, bandwidth, loss, cost)	Ryu controller + Mininet. NSFNet and 10-node topologies. Compared to other greedy routing	Good results in terms of cost, loss and bandwidth, with acceptable delay	Overall gain is low. Missing detail about traffic matrices
Dueling Double Deep Q-learning (Dueling DQN)	Enhanced throughput and delay	Ryu controller + Mininet. Fat-tree, NSFNet and ARPANet topologies. Compared to OSPF and LL	Good results in terms of reward, file transmission time, and utilization rate metrics	Missing analysis of monitoring cost
Deep Q-learning + SARSA	Energy-efficient routing and guaranteed QoS	N/A (Only architectural design)	Detailed explanation of the architecture	Missing synthetic or real experiments and comparison

حال به مقایسه الگوریتم‌های مطرح شده برای یادگیری تقویتی عمیق می‌پردازیم. در جداول زیر، روش‌های مختلف یادگیری تقویتی نوشته و مقایسه شده‌اند:

Techniques	Objective	Implementation & Evaluation	Advantage	Disadvantage
Random Neural Networks (RNNs)	Secure traffic engineering based on QoS for cloud providers with low monitoring	Floodlight controller + Mininet. GÉANT network topology and traffic. Compared to IP (shortest path)	Better round trip time than IP. Reduced monitoring overhead	Relatively small evaluated network. More QoS parameters should be measured to prove the approach
Recurrent Neural Network + DDPG	Reduced delay	POX controller + Pica8 switches. OSNE network topology	Reduced delay in comparison with shortest path. Realistic evaluation scenario	Poor scalability
DDPG	Enhancing overall scalability in comparison to other DRL approaches	OMNeT++ simulator. OS3E, NSF and BRITE-generated network topologies	Partial control shows very good preliminary results	Evaluated only via simulation
DDPG	Traffic engineering via combination of DRL and pinning control theory focused on scalability	OMNeT++ simulator. OS3E, NSF and BRITE-generated network topologies	Improves delay	Throughput is not tested. Traffic workload is not real. Evaluated only via simulation
DDPG	Reduced network delay via a DRL agent for routing optimization	OMNeT++ simulator. Scale-free network topology	One-step, model-free, black-box optimization	Evaluated only via simulation. Few details about the design
DDPG	Enhanced throughput and delay, while keeping reduced convergence time	OMNeT++ simulator. Sprint backbone network. Compared against OSPF	DROM dynamically adjusts the reward function, it does not rely on specific network states and achieves better results than OSPF	DROM requires the definition of a strategy, which cannot be defined automatically (and requires human intervention)

rning	Optimized multipath routing in DCNs (DRL to compute links weight and Dijkstra's to select optimal paths)	POX controller + Mininet. Fat-tree topology	Improves ECMP	Evaluated only with a few tests and not using DC-based workloads. Missing in-depth design details
d	Optimized load balancing	C++/WILL API. Fixed mesh topology	Beter results than OSPF an DL	Evaluated with a few tests and not considering the usual performance metrics
(BM)	Optimized load balancing in DCNs	Floodlight controller + Mininet. Fat tree topology. Traffic generated with Iperf	Compared with other ML techniques such as: ANN, SVM and logistic regression (all worse than the author's proposal)	Missing details of the DRL technique implemented. Limited to DCNs

[6] D. Lopez-Pajares, E. Rojas, J. A. Carral, I. Martinez-Yelmo, and J. Alvarez-Horcajo, ‘‘The disjoint multipath challenge: Multiple disjoint paths guaranteeing scalability,’’ IEEE Access, vol. 9, pp. 74422–74436, 2021.

[7] A. Shirmarz and A. Ghaffari, ‘‘Performance issues and solutions in SDNbased data center: A survey,’’ J. Supercomput., vol. 76, pp. 7545–7593, Oct. 2020.

[8] S. Badotra and S. N. Panda, ‘‘Experimental comparison and evaluation of various OpenFlow software defined networking controllers,’’ Int. J. Appl. Sci. Eng., vol. 17, pp. 317–324, Dec. 2020, doi: 10.6703/IJASE.202012_17(4).317.

[9] P. Casas, ‘‘Two decades of AI4NETS–AI/ML for data networks: Challenges & research directions,’’ in Proc. IEEE/IFIP Netw. Operations Manage. Symp. (NOMS), Apr. 2020, pp. 1–6.

[10] C. Yu, J. Lan, Z. Guo, and Y. Hu, ‘‘DROM: Optimizing the routing in software-defined networks with deep reinforcement learning,’’ IEEE Access, vol. 6, no. 18, pp. 54539–64533, 2018.

[11] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, “A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges,” IEEE Communications Surveys Tutorials, vol. 21, no. 1, pp. 393–430, Firstquarter 2019.

[12] Y.-J. Wu, P.-C. Hwang, W.-S. Hwang, and M.-H. Cheng, ‘‘Artificial intelligence enabled routing in software defined networking,’’ Appl. Sci., vol. 10, no. 18, p. 6564, Sep. 2020, doi: 10.3390/app10186564

[13] M. Latah and L. Toker, ‘‘Artificial intelligence enabled software-defined networking: A comprehensive overview,’’ IET Netw., vol. 8, no. 2, pp. 79–99, 2018.

مراجع

[1] R. Amin, E. Rojas, A. Aqdus, S. Ramzan, D. Casillas-Perez, and J. M. Arco, ‘‘A survey on machine learning techniques for routing optimization in SDN,’’ IEEE Access, vol. 9, pp. 104582–104611, 2021.

[2] A. Mourad, R. Yang, P. H. Lehne, and A. de la Oliva, ‘‘Towards 6G: Evolution of key performance indicators and technology trends,’’ in Proc. 2nd 6G Wireless Summit (6G SUMMIT), 2020, pp. 1–5.

[3] Z. Latif, K. Sharif, F. Li, M. M. Karim, S. Biswas, and Y. Wang, ‘‘A comprehensive survey of interface protocols for software defined networks,’’ J. Netw. Comput. Appl., vol. 156, Apr. 2020, Art. no. 102563.

[4] E. Rojas, R. Doriguzzi-Corin, S. Tamurejo, A. Beato, A. Schwabe, K. Phemius, and C. Guerrero, ‘‘Are we ready to drive software-defined networks? A comprehensive survey on management tools and techniques,’’ ACM Comput. Surveys, vol. 51, no. 2, pp. 1–35, Jun. 2018, doi: 10.1145/3165290.

[5] J. W. Guck, A. Van Bemten, M. Reisslein, and W. Kellerer, ‘‘Unicast QoS routing algorithms for SDN: A comprehensive survey and performance evaluation,’’ IEEE Commun. Surveys Tuts., vol. 20, no. 1, pp. 388–415, 1st Quart., 2018.

[14] IECON.2017.8217065 [26] M. Latah and L. Toker, ‘‘Application of artificial intelligence to software defined networking: A survey,’’ Indian J. Sci. Technol., vol. 9, no. 44, pp. 1–7, Nov. 2016. [Online].

[15] L. Cui, S. Yang, F. Chen, Z. Ming, N. Lu, and J. Qin, ‘‘A survey on application of machine learning for Internet of Things,’’ Int. J. Mach. Learn. Cybern., vol. 9, no. 8, pp. 1399–1417, 2018.

[16] Y. Zhao, Y. Li, X. Zhang, G. Geng, W. Zhang, and Y. Sun, ‘‘A survey of networking applications applying the software defined networking concept based on machine learning,’’ IEEE Access, vol. 7, pp. 95397–95417, 2019.

[17] M. Usama, J. Qadir, A. Raza, H. Arif, K. L. A. Yau, Y. Elkhatib, A. Hussain, and A. Al-Fuqaha, ‘‘Unsupervised machine learning for networking: Techniques, applications and research challenges,’’ IEEE Access, vol. 7, pp. 65579–65615, 2019.