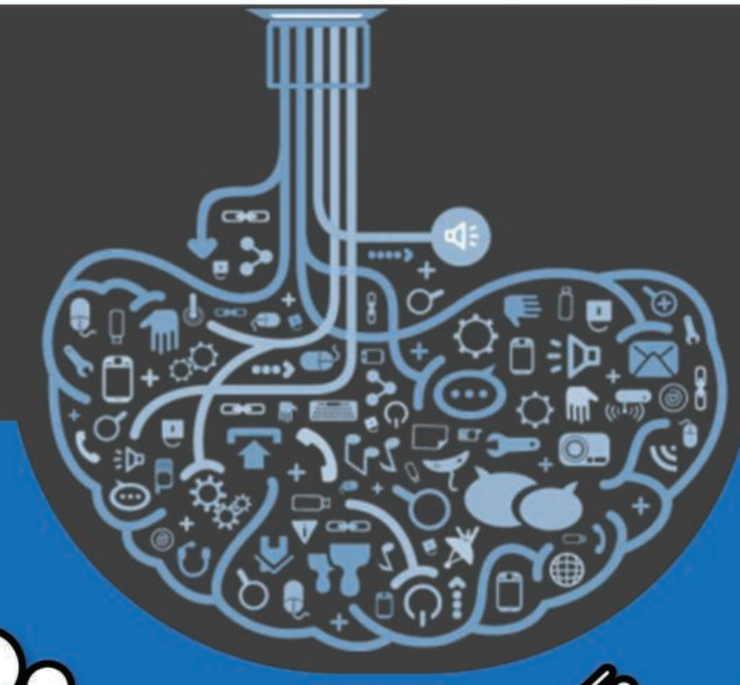


یادگیری عمیق

اصول، مفاهیم و رویکردها

تالیف و گردآوری: میلاد وزان

یادگیری عمیق

اصول، مفاهیم و رویکردها

میلاد وزان

آنچه در این کتاب می‌خوانید:

- اصول و مفاهیم یادگیری عمیق
- یادگیری بانظارت عمیق: شبکه‌های عصبی پیش‌خور، بازگشی و همگشتی
- یادگیری بازنمایی بدون‌نظارت عمیق: خودرمزنگارها و مدل‌های مولد
- یادگیری تقویتی عمیق
- یادگیری انتقالی عمیق
- یادگیری عمیق هندسی: یادگیری بازنمایی گراف

9 786222 314934

به نام خدا

یادگیری عمیق

اصول، مفاهیم و رویکردها

تالیف و گردآوری: میلاد وزان

سرشناسه	:	وزان، میلاد، ۱۳۷۱-
عنوان و نام پدیدآور	:	یادگیری عمیق: اصول، مفاهیم و رویکردها/ تالیف و گردآوری میلاد وزان.
مشخصات نشر	:	تهران: میعاد اندیشه،۱۳۹۹.
مشخصات ظاهری	:	۲۵۲ص.: مصور(رنگی)،جدول(رنگی)،نمودار(رنگی).
شابک	:	978-622-231-493-4
وضعیت فهرست نویسی	:	فیپا
یادداشت	:	کتابنامه:ص.[۲۴۷]-۲۵۲.
موضوع	:	یادگیری عمیق
موضوع	:	Deep Learning
موضوع	:	یادگیری
موضوع	:	Learning
رده بندی کنگره	:	LB۱۰۶۰
رده بندی دیویی	:	۳۷۰/۱۵۲۳
شماره کتابشناسی ملی	:	۷۵۲۲۲۴۳
وضعیت رکورد	:	فیپا

انتشارات میعاد اندیشه

miaadpub.ir

عنوان کتاب: یادگیری عمیق: اصول، مفاهیم و رویکردها

تالیف و گردآوری : میلاد وزان

ناشر : میعاد اندیشه

نوبت چاپ: اول - ۱۳۹۹

شمارگان: ۱۰۰۰ نسخه

قیمت: ۳۰۰/۰۰۰ریال

شابک: ۴- ۴۹۳- ۲۳۱ - ۶۲۲ - ۹۷۸

تلفن انتشارات : ۰۹۱۲۵۱۲۰۰۶۷ - ۰۲۱۶۶۰۱۷۴۴۸ - ۰۲۱۶۶۰۱۴۷۹۷

پیش‌گفتار

ساخت سیستمی هوشمند که توانایی استخراج بازنمایی سطح بالا از داده‌ها را داشته باشد، در بسیاری از مسائل مرتبط با هوش مصنوعی لازم و ضروری است. استدلال‌های نظری و بیولوژیکی نشان می‌دهد که برای ساخت چنین سیستم‌هایی، نیاز به مدل‌هایی با معماری عمیق است که شامل بسیاری از لایه‌های پردازش غیرخطی هستند.

یادگیری عمیق، زیرمجموعه‌ای از روش‌های یادگیری ماشین می‌باشد که از آن به عنوان یادگیری بازنمایی نیز نام برده می‌شود. یادگیری بازنمایی یا یادگیری ویژگی‌ها، تکنیکی است که به ماشین این توانایی را می‌دهد تا روابط را از داده‌های خام به‌صورت خودکار کشف کند. این توانایی مهم و مشخصه اصلی یادگیری عمیق، با یادگیری در لایه‌های مختلف در ساختار شبکه امکان‌پذیر می‌شود.

تا قبل از پیدایش یادگیری عمیق، روش‌های یادگیری ماشین سنتی، بیش‌از حد به بازنمایی‌هایی (انتخاب ویژگی‌ها) که از داده‌ها بدست می‌آورند، وابسته بودند. این روش‌ها، نیاز به یک متخصص در دامنه موضوع داشت تا استخراج ویژگی‌ها را به‌صورت دستی انجام دهد. حال آن‌که، این استخراج ویژگی‌ها به صورت دستی فرآیندی چالش‌انگیز و زمان‌بر است. پیدایش یادگیری عمیق توانست به‌سرعت جایگزین این روش‌های سنتی شود. چرا که می‌توانست استخراج ویژگی‌ها را به‌صورت خودکار متناسب با هر مساله بدست آورد.

در چندین سال اخیر، یادگیری عمیق به محرک اصلی راه‌حل‌های نوآورانه برای مسائل هوش مصنوعی تبدیل گردیده که این امر با افزایش مقدار داده‌های موجود، افزایش منابع محاسباتی و تکنیک‌های بهبودیافته در آموزش شبکه‌های عمیق امکان‌پذیر شده است. یادگیری عمیق امروزه فراهم‌ساز تغییرات گسترده در صنعت فناوری شده است، از همین‌رو درک و نحوه کار یادگیری عمیق برای افراد متخصص نرم‌افزاری در دنیای امروز، مفید، ضروری و لازم خواهد بود.

کتاب حاضر با بهره‌گیری از جدیدترین مراجع علمی و تجربیات مولف، برای طیف گسترده‌ای از پژوهشگران، دانشجویان، صاحبان صنایع و علاقمندان به یادگیری عمیق، تالیف و گردآوری گردیده تا بتوانند دانش کافی از اصول و مفاهیم اساسی در این زمینه را بدست آورند. همچنین، از آنجایی که این کتاب مطالب مورد نیاز برای درس یادگیری عمیق در مقطع تحصیلات تکمیلی را پوشش می‌دهد، می‌تواند مرجع مناسبی برای این درس و همچنین درس اختیاری دانشجویان سال آخر کارشناسی در رشته مهندسی کامپیوتر باشد. از همین‌رو تمام تلاش براین بود تا مطالب کتاب را به شیوه‌ای ساده، روان و قابل فهم، به‌همراه مثال‌هایی برای درک بهتر ارائه کنیم تا برای این طیف گسترده از خوانندگان قابل درک باشد.

تالیف این کتاب بر درک اصول و مفاهیم یادگیری عمیق، با چشم‌اندازی بر رویکردهای مختلف یادگیری تاکید دارد. اما، این بدان معنا نیست که بتوان در یک جلد کتاب به تشریح همه جوانب و اکتشافات یادگیری پرداخت؛ ما نیز چنین قصدی را نداریم. در واقع هدف ما از تالیف و گردآوری مطالب این کتاب، بدست آوردن دانش کافی از یادگیری عمیق و رویکردهای مختلف آن با تاکید بر مباحث مهم و نوین و ارائه دیدگاه جامعی از یادگیری عمیق می‌باشد.

هرچند که مطالب این کتاب قابل فهم بوده، اما این مطالب برمبنای اینکه خواننده دانش کافی از یادگیری ماشین را دارد، تالیف شده است. قطعا داشتن اطلاعات زمینه‌ای از یادگیری ماشین فهم مطالب این کتاب را دوچندان خواهد کرد. با این همه، در طول بیان مطالب، تمام تلاش بر این بود تا این پیش‌نیازها از یادگیری ماشین حداقل شود.

کتاب حاضر از ٦ فصل تشکیل شده است که در زیر به‌طور خلاصه به محتوای این فصل‌ها اشاره‌ای خواهیم داشت:

- **فصل اول ـ مقدمه‌ای بر یادگیری ماشین و یادگیری عمیق:** این فصل با تعریفی از یادگیری شروع می‌شود. سپس، مفهوم یادگیری ماشین و انواع رویکردهای مختلف آن همانند: یادگیری بانظارت، بدون‌نظارت، تقویتی، نیمه‌نظارتی، خودنظارت‌شده، فعال، برخط، چندوظیفه‌ای و انتقالی به‌صورت خلاصه بیان و با یک‌دیگر مقایسه می‌شوند. در ادامه، به تعریفی از یادگیری عمیق پرداخته و نحوه کار، اهمیت، چالش‌های

موجود در آن و تفاوت آن با یادگیری ماشین تشریح و در انتهای فصل کاربردهای آن در دنیای واقعی بیان می‌شود.

- **فصل دوم- یادگیری بانظارت عمیق:** در فصل دوم کتاب شبکه‌های عصبی مصنوعی و ساختار آن را شرح داده و فرآیند یادگیری و مفاهیم مورد نیاز آن همانند: تابع فعال‌سازی، تابع زیان، رویکردهای وزن‌دهی اولیه، بهینه‌سازی و چالش‌های موجود در یادگیری شبکه‌های عمیق بیان می‌شوند. در ادامه فصل، با دو نوع از شبکه‌های عمیق محبوب، یعنی شبکه‌های بازگشتی و همگشتی آشنا خواهید شد.
- **فصل سوم- یادگیری بازنمایی بدون‌نظارت عمیق:** در این فصل، ساختار خودرمزنگار و انواع آن را تشریح خواهیم کرد. سپس، به مقایسه بین مدل‌های مولد و تفکیک‌پذیر پرداخته و در انتها انواع مدل‌های مولد عمیق را مورد بررسی قرار خواهیم داد.
- **فصل چهارم- یادگیری تقویتی عمیق:** در این فصل، خواهیم دید که چرا برخی مسائل با رویکردهای یادگیری بانظارت و بدون‌نظارت قابل حل نیستند و نیاز به یادگیری تقویتی در این مسائل احساس می‌شود. در ادامه فصل، به انواع رویکردهای کلاسیک حل مسائل با یادگیری تقویتی پرداخته و در انتها هم علت بوجود آمدن زمینه تحقیقاتی جدید با عنوان یادگیری تقویتی عمیق را خواهیم دید و رویکردهای آن را بررسی خواهیم کرد.
- **فصل پنجم- یادگیری انتقالی عمیق:** فصل پنجم کتاب، یادگیری انتقالی، انگیزه استفاده، مزایا، استراتژی‌های استفاده از آن و رویکردهای مختلف آن را بررسی خواهیم کرد.
- **فصل ششم- یادگیری عمیق هندسی:** فصل آخر کتاب را به زمینه تحقیقاتی جدید با عنوان یادگیری عمیق هندسی اختصاص داده‌ایم. در این فصل با ساختار گراف، شبکه‌های عصبی گراف، یادگیری بازنمایی گراف و شبکه‌های همگشتی گراف آشنا خواهید شد. شایان ذکر است، به این فصل با دیدی کلی نگاه شده و به جزئیات آن پرداخته نشده است.

در پایان از خوانندگان محترم تقاضا می‌شود انتقادات، پیشنهادات و یا در صورت مشاهده هرگونه اشکال در کتاب، اینجانب را آگاه سازند.

میلاد وزان

زمستان ۱۳۹۹

vazanmilad@gmail.com

فهرست مطالب

عنوان **شماره صفحه**

فصل دوم: یادگیری بانظارت عمیق (شبکه‌های عصبی پیش‌خور، بازگشتی و همگشتی)

فصل سوم: یادگیری بازنمایی بدون‌نظارت عمیق (خودرمزنگارها و مدل‌های مولد)

فصل چهارم: یادگیری تقویتی عمیق

فصل ۱

مقدمه‌ای بر یادگیری ماشین و یادگیری عمیق

اهداف:

- **مروری بر یادگیری ماشین**
- **یادگیری عمیق چیست؟**
- **تفاوت یادگیری ماشین با یادگیری عمیق در چیست؟**
- **ارتباط بین هوش مصنوعی، یادگیری ماشین و یادگیری عمیق**
- **اهمیت و کاربرد یادگیری عمیق**

۰.۱ مقدمه

هدف اصلی در حوزه هوش مصنوعی دادن توانایی به رایانه‌ها به‌گونه‌ای است تا بتوانند دنیای اطراف خود را درک و به روشی هوشمندانه با آن تعامل کنند. در چندین سال گذشته یادگیری عمیق به عنوان یکی از امیدوارکننده‌ترین رویکردها برای دستیابی به این هدف ظاهر شده است.

یادگیری عمیق راهی برای یادگیری محاسباتی مفاهیم سطح بالا در داده‌ها و بازنمایی آن‌ها با استفاده از شبکه‌های عصبی سلسله‌مراتبی عمیق است و در زیرمجموعه روش‌های یادگیری ماشین قرار می‌گیرد. از همین‌رو، مروری بر مفاهیم یادگیری ماشین قبل از فراگیری یادگیری عمیق مفید خواهد بود. چرا که بسیاری از مفاهیم استفاده شده در شبکه‌های عصبی همانند: یادگیری بانظارت، بدون‌نظارت، تقویتی و بسیاری مباحث دیگر از یادگیری ماشین ناشی می‌شوند. بنابراین، در این بخش مروری کوتاه بر این مفاهیم جهت درک بهتر از یادگیری عمیق خواهیم داشت. اما قبل از آن که به آن‌ها بپردازیم کمی به عقب برمی‌گردیم و به این موضوع می‌پردازیم که یادگیری چیست؟

۱.۱ یادگیری چیست؟

هنگامی که در مورد یادگیری انسان صحبت می‌کنیم بین یادگیری[۱]، به یادسپاری[۲] و هوش[۳] تمایز قائل می‌شویم. بدون شک یادآوری شماره‌های تلفن نوعی یادگیری به‌حساب می‌آید، اما وقتی می‌گوییم یادگیری، اغلب به چیز دیگری اشاره می‌کنیم. یادگیری را می‌توان **بهتر شدن عملکرد در یک وظیفه خاص با استفاده تجربه و تمرین تعریف کرد**. رفتار هوشمندانه انسان از طریق یادگیری در تجربه‌ها برچسب می‌خورد، یادگیری همان پدید آورنده‌ی انعطاف‌پذیری در زندگی فردی است.

تصور کنید با استفاده از فلش کارت قصد داریم تفاوت بین گربه و سگ را به کودک آموزش دهیم. یک کارت را به کودک نشان می‌دهیم، کودک یکی را انتخاب می‌کند،

---

[۱] learning
[۲] memorization
[۳] intelligence

سپس کارت برای انتخاب درست یا اشتباه در یکی از دو ستون مربوط قرار می‌گیرد. با تمرین کودک، عملکرد وی بهبود می‌یابد و قادر به شناخت و تفکیک بین گربه و سگ خواهد شد. برای توانایی این ادراک و شناخت در انسان آنچه که تنها مورد نیاز خواهد بود نمونه‌ها هستند. پس از آنکه کودک از طریق این نمونه فلش‌کارت‌ها مهارت بدست آورد، نه‌تنها قادر خواهد بود تصاویر روی فلش‌کارت‌ها، بلکه بیشتر تصاویر سگ و گربه را از یک‌دیگر دسته‌بندی و تفکیک کند. این توانایی تعمیم‌دهی در به‌کار بردن دانش بدست آمده برای نمونه‌هایی که تاکنون آن‌ها را مشاهده نکرده است، همان ویژگی اصلی یادگیری در انسان و ماشین می‌باشد. البته، یادگیری در انسان چیزی فراتر و پیشرفته‌تر حتی از پیشرفته‌ترین الگوریتم‌های یادگیری ماشین است. اما یادگیری در رایانه‌ها چیست؟ تام میشل (۱۹۹۷) یادگیری را در برنامه‌های رایانه‌ای اینگونه توصیف می‌کند:

"**هرگونه برنامه رایانه‌ای که کارایی‌اش در یک کار خاص با تجربه بهبود یابد**".

و در تعریفی دقیق‌تر آن را اینگونه بیان می‌کند:

"یک برنامه رایانه‌ای با در نظر گرفتن تجربه E در مورد وظیفه T برحسب معیار کارایی P توانایی یادگیری خواهد داشت، اگر کارایی‌اش پس از تجربه E برای وظیفه T بهبود یابد."

بر طبق این تعریف دو پرسش بوجود می‌آید: اینکه، یک رایانه چگونه می‌داند که در وظیفه خاص مورد انجام در حال بهتر شدن عملکرداش هست یا نه، و دوم اینکه چگونه می‌داند که باید چگونه در انجام این وظیفه بهبود یابد. پاسخ به این پرسش‌ها دسته‌بندی از چند رویکرد متفاوت در یادگیری ماشین را بوجود می‌آورد که در ادامه به تشریح آن‌ها با عناوین یادگیری بانظارت، نیمه‌نظارتی و بدون‌نظارت خواهیم پرداخت.

۱.۱.۱ یادگیری ماشین

ما در طول روز اغلب با ماشین‌هایی مواجه هستیم که برای کاری که برنامه‌ریزی شده‌اند، به روشی مکانیکی آن را انجام می‌دهند. اما اگر این ماشین‌ها همانند ما انسان‌ها بتوانند از تجربه درس بگیرند و اگر ماشین آلات بتوانند در یک محیط کنترل شده و اخلاقی رفتار خود را تغییر دهند تا کارآیی بیشتری داشته باشند، چه می‌شود؟ در سال‌های اخیر،

تحولی در سیستم‌های فناوری در حال انجام است تا سیستم‌های منفعل ساکن به سیستم‌های خودکار و پویایی فعال شوند که با گذشت زمان بهبود پیدا می‌کنند. این رویکرد، یادگیری ماشین نام دارد.

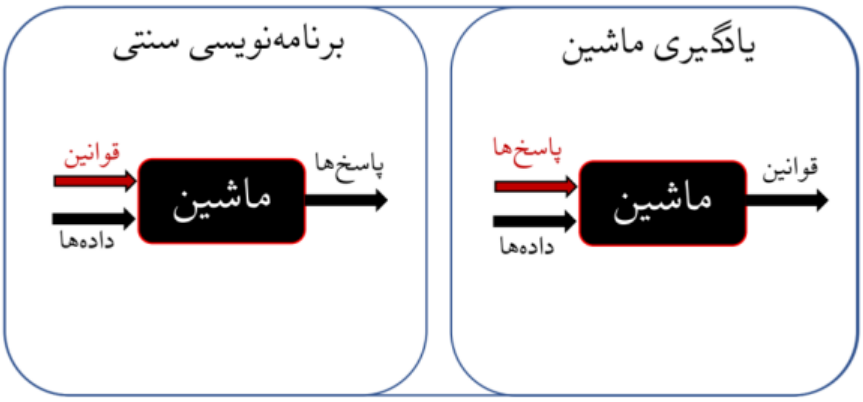
یادگیری ماشین یکی از شاخه‌های هوش مصنوعی است که در آن ماشین یاد می‌گیرد وظایفی را انجام دهد که به صراحت برنامه‌ریزی نشده و عملکرد ماشین به صورت خودکار از طریق تجربه در انجام این وظیفه، بهبود می‌یابد. تلاش برای ایجاد ماشینی که همانند انسان تفکر کند مسیری طولانی را طی کرده است. یادگیری ماشینی از یک رشته مبهم به یک نیروی عمده صنعتی و اجتماعی در تصمیم‌گیری خودکار، از تجارت برخط و تبلیغات گرفته تا آموزش و مراقبت‌های بهداشتی تبدیل شده است. یادگیری ماشینی در حال تبدیل شدن به یک فن‌آوری توانمند عمومی برای جهانیان به دلیل توانایی قوی یادگیری از طریق سازگاری با داده‌های برچسب‌دار و بدون برچسب است.

فرآیند یادگیری ماشین با استفاده از داده‌های خام در جهت استخراج اطلاعات مفید برای کمک به تصمیم‌گیری بهتر شروع می‌شود. پس در تعریفی دقیق‌تر یادگیری ماشین را می‌توان اینگونه بیان کرد: یادگیری ماشین بر طراحی مدل‌هایی متمرکز بوده که در یک حوزه خاص، الگوریتم‌های کامپیوتری برمبنای داده‌های آموزشی داده شده به مدل یادگیری را به‌صورت خودکار از طریق تجربه و آزمایش از داده‌ها بدست آورده، تا در مواجه با داده‌های جدید در همان حوزه بتواند رفتاری مشابه انسان از خود نشان دهد. یادگیری ماشین با رویکردهای متفاوتی قابل پیاده‌سازی و اجرا می‌باشد. از سه رویکرد اصلی آن می‌توان به **یادگیری بانظارت**، **یادگیری بدون‌نظارت** و **یادگیری تقویتی** نام برد. علاوه بر این سه رویکرد، رویکردهای دیگری نیز وجود دارد که در ادامه با آن‌ها آشنا خواهیم شد.

۱.۱.۱.۱ برنامه‌نویسی سنتی در مقابل یادگیری ماشین

هنگامی که صحبت استفاده از رایانه برای انجام وظایفی از طرف انسان می شود، همیشه باید دستورالعمل‌هایی را در قالب یک برنامه رایانه‌ای به رایانه دستور دهیم. زبان‌های برنامه‌نویسی سنتی یک فرآیند دستی است (به این معنا که یک شخص باید برنامه‌نویس

باید برنامه را ایجاد کند) و معمولا داده‌ها و مجموعه‌ای از قوانین را به عنوان ورودی می‌گیرند و با اعمال این قوانین بر داده‌ها پاسخ‌ها را به عنوان خروجی بدست می‌آورند. در طرف دیگر، در یادگیری ماشین داده‌ها و پاسخ‌ها (یا برچسب‌ها) به عنوان ورودی داده می‌شود و از قوانین (مدل‌ها) به عنوان خروجی استفاده می‌شود (شکل ۱ــ۱). الگوی یادگیری ماشین از ارزش بی‌نظیری برخوردار است. چرا که، به ماشین اجازه می‌دهد تا قوانین جدیدی را در فضای پیچیده و با ابعاد بالا که درک آن برای انسان دشوارتر است را بیاموزد.



شکل ۱ــ۱ برنامه‌نویسی سنتی در مقابل یادگیری ماشین

۲.۱.۱.۱ یادگیری بانظارت

خود را به عنوان دانش آموزی در کلاس درس ریاضی در نظر بگیرید که در آن معلم به چگونگی حل مساله یا حل درست یا غلط یک مساله بر شما نظارت دارد. این وضعیت مشابه آن چیزی است که الگوریتم بانظارت دنبال می‌کند. در حالت یادگیری بانظارت سیستم برای هر نمونه‌ی آموزشی یادگیری را براساس یک اِلمان دوتایی از ورودی و برچسب که خروجی مساله می‌باشد، فرا می‌گیرد. **مجموعه داده دارای برچسب به معنی این است که برای هر عضوی از مجموعه آموزشی، یک پاسخ یا راه‌حل نیز برای آن داده شده است.** هدف در اینجا تطبیق سیستم به‌گونه‌ای است که برای ورودی جدید سیستم بتواند خروجی درست را براساس آن‌چه تاکنون از داده‌های آموزشی فراگرفته است، پیش‌بینی کند. در یادگیری نظارتی اگر داده‌های مسئله جهت یادگیری به‌صورت

گسسته باشند، این مساله **دسته‌بندی**، و اگر مقادیر داده‌ها به‌صورت پیوسته باشند به آن **رگرسیون** گویند.

دسته‌بندی

دسته‌بندی سعی در ایجاد ارتباطی بین نمونه‌های آموزشی و دسته‌های از پیش تعریف شده برای مسئله مورد نظر دارد. برای مثال فرض کنید مجموعه‌ای از تصاویری حیوانات همانند: سگ، گربه، خرگوش، ببر وغیره وجود دارد. قرار دادن هر یک از این تصاویر حیوانات در دسته‌ی مربوط به خودش، عمل دسته‌بندی یا طبقه‌بندی نامیده می‌شود. برای این مثال مورد نظر، در این روش سیستم با تعداد زیادی از تصاویر به‌همراه برچسب‌ها، تا زمانی که سیستم به کارایی خوبی در دسته‌بندی تصاویر پیدا کند، آموزش داده می‌شود.

رگرسیون

فرض کنید، مجموعه‌ای از مقادیر را داریم. برای مثال، داده‌هایی در مورد شرایط آب و هوایی با ویژگی‌های مختلف موجود بوده، در این میان بعضی از مقادیر گمشده هستند. تخمین این مقادیر گمشده براساس ارتباط بین داده‌ها توسط رگرسیون انجام می‌گیرد. در بیانی دیگر، یک مقدار عددی را با در نظر گرفتن یک ورودی پیش‌بینی می‌کند.

۳.۱.۱.۱ یادگیری بدون‌نظارت

یادگیری بدون‌نظارت بدون کمک یک ناظر اتفاق می‌افتد، درست همانند این‌که یک ماهی به تنهایی شنا را یاد می‌گیرد. در یادگیری بدون‌نظارت، الگوریتم یادگیرنده برای ورودی‌های مشخص، بدون خروجی مشخص می‌باشد. هدف در این یادگیری پیدا کردن این خروجی‌ها به تنهایی است که این عمل توسط خود الگوریتم از طریق تجزیه و تحلیل داده‌ها و شناسایی الگوهای پنهان و خاصی که در ساختار داده‌ها وجود دارد انجام می‌گیرد. خوشه‌بندی نمونه‌ای از این نوع یادگیری می‌باشد، که هدف آن کشف گروهایی

از موارد مشابه براساس شباهت اندازه‌گیری شده، یا درک این شباهت‌ها در ساختار داده‌ها است.

مزایای یادگیری بدون‌نظارت

- برچسب‌گذاری داده‌ها مستلزم کار و هزینه زیادی است. یادگیری بدون‌نظارت با یادگیری از داده‌های بدون‌برچسب این مشکل را حل می‌کند.
- در یافتن الگوهایی از داده‌ها که یافتن آن‌ها با استفاده از روش‌های معمول امکان‌پذیر نیست، بسیار مفید است.
- کاهش ابعاد داده‌ها با استفاده از این نوع یادگیری به‌راحتی انجام می‌شود.

معایب یادگیری بدون‌نظارت

- نتیجه ممکن است در مقابل روش یادگیری بانظارت از دقت کم‌تری برخوردار باشد. چراکه ما هیچگونه برچسبی برای داده‌ها در اختیار نداریم و مدل باید با دانش بدست آمده از داده‌های خام یاد بگیرد.
- هرچه تعداد ویژگی‌ها بیشتر شود، پیچیدگی بیشتر می‌شود.
- فرآیندی زمان‌بر است. چراکه مرحله یادگیری الگوریتم ممکن است زمان زیادی را صرف تجزیه و تحلیل و محاسبه همه احتمالات کند.

۴.۱.۱.۱ یادگیری تقویتی

یادگیری تقویتی به نوعی متفاوت مساله را حل می‌کند. یادگیری تقویتی، مبتنی‌بر تعامل با محیط است و الگوریتم یاد می‌گیرد که به تنهایی نسبت به محیط واکنش نشان دهد. در این نوع یادگیری، الگوریتم با سازوکار بازخورد و تجارب گذشته یاد می‌گیرد و سعی می‌کند مسائل را همانند روشی که بشر در زندگی مدل‌سازی می‌کند به حل آن بپردازد و می‌آموزد که چگونه رفتار عامل را براساس وجود یا عدم وجود پاداش بهینه کند. **هدف آن یافتن مجموعه الگوهایی از اقدامات، با آزمایش و مقایسه همه آن‌ها تا بیشترین امتیاز پاداش را بدست آورد.** این نوع از یادگیری نیازی به مجموعه داده آموزشی ندارد. به عبارت دیگر، نه یادگیری بانظارت است و نه یادگیری بدون‌نظارت.

۵.۱.۱.۱ تفاوت یادگیری بانظارت، بدون‌نظارت و تقویتی

یادگیری بانظارت زمانی است که مدل از یک مجموعه داده برچسب‌دار برای راهنمایی در حل مساله کمک می‌گیرد. یادگیری بدون‌نظارت نیازی به داده‌های برچسب‌دار ندارد و این خود مدل است که به تنهایی و بدون یک ناظر خارجی با کشف الگوهای پنهان به حل مساله می‌پردازد. در مقابل این دو رویکرد، یادگیری تقویتی نیازی به مجموعه داده نداشته و ماشین یا عامل با محیط خود در تعامل است تا با آزمون و خطا و دریافت پاداش از محیط بهترین اقدام را در جهت حل مساله مورد نظر بدست آورد. به‌طور خلاصه، در یادگیری بانظارت، هدف تولید فرمولی براساس مقادیر ورودی و خروجی است. در یادگیری بدون‌نظارت، ارتباطی بین مقادیر ورودی و گروه‌بندی آن‌ها پیدا می‌شود. در یادگیری تقویتی، یک عامل از طریق تعامل با محیط یاد می‌گیرد. بر این اساس می‌توان تفاوت این سه رویکرد از یادگیری ماشین را در جدول ۱ ــ ۱ مشاهده کرد.

جدول ۱ ــ ۱ مقایسه یادگیری بانظارت، بدون‌نظارت و تقویتی

شاخص	یادگیری بانظارت	یادگیری بدون‌نظارت	یادگیری تقویتی
تعریف	از طریق مجموعه داده دارای برچسب یاد می‌گیرد.	بدون راهنما از طریق داده‌های بدون‌برچسب آموزش داده می‌شود.	در تعامل با محیط کار می‌کند.
نوع داده‌ها	داده‌های برچسب‌دار	داده‌های بدون برچسب	بدون تعریف داده
نوع مساله	دسته‌بندی و رگرسیون	قوانین انجمنی و خوشه‌بندی	مبتنی‌بر پاداش
ناظر	ناظر اضافی	بدون ناظر	بدون ناظر
هدف	نگاشت داده‌های ورودی به خروجی‌های مشخص	کشف الگو	آموختن یک سری اقدامات

۶.۱.۱.۱ یادگیری نیمه‌نظارتی

یادگیری نیمه‌نظارتی ترکیبی است از یادگیری بانظارت و یادگیری بدون‌نظارت، به عبارتی هم دارای داده‌های برچسب‌دار و هم بدون برچسب می‌باشد. در این روش ابتدا از داده‌های بدون برچسب، بازنمایی ویژگی‌های داده‌های ورودی را استخراج کرده و سپس از این بازنمایی ویژگی‌های فراگرفته شده در یادگیری نظارتی استفاده می‌کند. این رویکرد بیشتر هنگامی استفاده می‌شود که داده‌های برچسب‌دار کمی موجود بوده و بر کاهش

نواقص هر دو رویکرد بانظارت و بدون‌نظارت متمرکز می‌باشد. **هدف اصلی آن، استفاده موثر از همه داده‌های موجود است نه فقط داده‌های دارای برچسب.**

۷.۱.۱.۱ یادگیری خودنظارت‌شده

در مقایسه با یادگیری بانظارت و بدون‌نظارت، یادگیری خودنظارت‌شده بیشترین شباهت را با یادگیری بدون‌نظارت داشته اما به داده‌های برچسب‌دار هم نیاز دارد. در این نوع از یادگیری مدل روی برچسب‌هایی آموزش می‌بیند که به‌طور خودکار از خود داده‌ها، بدون حاشیه نویسی انسان بدست آمده و برچسب خورده‌اند. در این رویکرد، مدل یادگیری خود را با استفاده از یک قسمت از داده‌ها برای پیش‌بینی قسمت دیگر و تولید برچسب‌ها به طور دقیق آموزش می دهد. در پایان، این روش یادگیری یک مسئله یادگیری بدون‌نظارت را به یک مسئله نظارت‌شده تبدیل می‌کند. یادگیری خودنظارت‌شده را می‌توان در مقایسه با سایر رویکردها به صورت زیر مقایسه کرد:

- **یادگیری خودنظارت‌شده در مقابل یادگیری بانظارت.**

ویژگی مشترک یادگیری بانظارت و خودنظارت‌شده این است که هر دو روش مدل‌های یادگیری را از مجموعه داده های آموزشی برچسب‌دار می‌سازند. با این حال، یادگیری خودنظارت‌شده نیازی به افزودن دستی برچسب‌ها ندارد، چراکه خود آن‌ها را ایجاد می‌کند.

- **یادگیری خودنظارت‌شده در مقابل یادگیری نیمه‌نظارتی.**

یادگیری نیمه‌نظارتی از داده‌های آموزشی دارای برچسب دستی برای یادگیری بانظارت و از رویکردهای یادگیری بدون‌نظارت برای داده‌های بدون برچسب استفاده می کند تا مدلی تولید کند که از برچسب‌های موجود استفاده کند و مدلی را ایجاد کند که می‌تواند فراتر از داده‌های دارای برچسب پیش‌بینی کند. در مقابل، یادگیری خود نظارت شده کاملاً به داده‌هایی تکیه دارد که فاقد برچسب تولید شده به صورت دستی هستند.

- **یادگیری خودنظارت‌شده در مقابل یادگیری بدون‌نظارت.**

یادگیری خودنظارت‌شده مشابه یادگیری بدون‌نظارت است. چراکه هر دو رویکرد با مجموعه داده‌هایی کار می‌کنند که برچسب دستی ندارند. در برخی منابع، یادگیری خود نظارت شده به عنوان زیر مجموعه‌ای از یادگیری بدون‌نظارت قرار می‌گیرد. با این حال، یادگیری بدون‌نظارت بر خوشه‌بندی، گروه‌بندی و کاهش ابعاد متمرکز است. در حالی که یادگیری خودنظارت‌شده استفاده می‌شود تا نتیجه‌گیری در مورد مسائل طبقه‌بندی و رگرسیون را انجام دهد.

۸.۱.۱.۱ یادگیری فعال

نوعی یادگیری نیمه‌نظارتی می‌باشد و روشی است که مدل را قادر می‌سازد تا در حین فرآیند یادگیری، از یک کاربر انسانی به‌صورت تعاملی استفاده کند تا داده‌ها را با خروجی‌های مورد نظر برچسب‌گذاری کند. **یادگیری فعال اجازه می‌دهد تا نمونه برداری از دامنه به‌گونه‌ای انجام شود که تعداد نمونه‌ها به حداقل برسد و اثربخشی مدل را حداکثر کند.** این نوع یادگیری زمانی مفید خواهد بود که داده‌های زیادی در دسترس نباشد و جمع‌آوری و برچسب‌گذاری داده‌های جدید هزینه زیادی داشته باشد.

۹.۱.۱.۱ یادگیری برخط

یادگیری ماشین سنتی به‌صورت برون‌خط انجام می‌شود، به این معنا که یک دسته داده داریم و قصد داریم یک معادله را بهینه کنیم. با این حال، اگر داده‌های جریانی داشته باشیم استفاده از یادگیری برخط لازم است. در یادگیری برخط، یادگیرنده سعی می‌کند با استفاده از نمونه‌هایی از داده‌ها در هر مرحله، بهترین پیش‌بینی یا تصمیم‌گیری را انجام دهد. یادگیری برخط بر مشکالات یادگیری دسته‌ای غلبه می‌کند. چرا که، مدل ایجاد شده می‌تواند بلافاصله برای هر نمونه داده جدید بروز شود. یادگیری برخط برای آن دسته از مسائلی مناسب است که مشاهدات با گذشت زمان ارائه می‌شوند و انتظار می‌رود توزیع احتمال مشاهدات نیز با گذشت زمان تغییر کند. بنابراین، انتظار می‌رود مدل به‌منظور

مهار این تغییرات به همان دفعات تغییر کند. یادگیری برخط به یک روش امیدوار کننده برای یادگیری از داده‌های جریانی در بسیاری از برنامه‌های دنیای واقعی تبدیل شده است.

۱۰.۱.۱.۱ یادگیری چندوظیفه‌ای

یادگیری چندوظیفه‌ای نوعی یادگیری بانظارت است و سعی دارد با اشتراک گذاشتن اطلاعات دامنه درباره یک مساله خاص، مدلی را ایجاد کند که می‌توانند هم‌زمان وظایف مختلف را انجام دهد. انگیزه اصلی یادگیری چندوظیفه‌ای ایجاد یک مدل "عمومی" است که می‌تواند چندین کار را به‌جای ایجاد چندین مدل "تخصصی" که فقط برای یک کار خاص آموزش دیده‌اند، در یک مدل حل کند. از منظر زیست‌شناختی، یادگیری چندوظیفه‌ای الهام گرفته از روشی است که ما انسان‌ها یاد می‌گیریم. برای یادگیری وظایف جدید، معمولا ما دانشی را که از یادگیری کارهای مرتبط بدست آورده‌ایم، بکار می‌گیریم. علاوه براین، ما معمولا در ابتدا وظایفی را یاد می‌گیریم که مهارت‌های لازم برای تسلط بیشتر برای کارهای پیچیده را در اختیار ما قرار می‌دهد.

۱۱.۱.۱.۱ یادگیری انتقالی

یادگیری انتقالی نوعی از یادگیری ماشین است که در آن ابتدا یک مدل در یک کار خاص آموزش داده می‌شود، سپس برخی یا کل مدل به عنوان نقطه شروع یک کار مرتبط استفاده می‌شود. به عبارت دیگر، می‌خواهیم از دانش آموخته شده توسط یک کار منبع استفاده کنیم تا به یادگیری یک کار هدف دیگر کمک کنیم. **هدف از یادگیری انتقالی بهبود فرآیند یادگیری وظایف جدید با استفاده از تجربه به‌دست آمده از حل مسائل قبلی است که تا حدودی مشابه هستند.** یادگیری انتقالی به‌ویژه در مدل‌هایی که به تدریج آموزش می‌بینند بسیار مفید است و می توان از یک مدل موجود به عنوان نقطه شروع برای ادامه آموزش استفاده کند، مانند شبکه‌های یادگیری عمیق. یادگیری انتقالی بر استخراج داده‌ها از یک دامنه مشابه متمرکز است تا توانایی یادگیری را افزایش یا تعداد نمونه‌های برچسب‌دار مورد نیاز در یک دامنه هدف را کاهش دهد. توجه به این نکته ضروری است که، خروجی مدل‌های یادگیری انتقالی تحت تأثیر رابطه بین منبع و حوزه‌های هدف قرار

دارد. اگر دامنه منبع و دامنه هدف دارای دانش مشترک کم‌تری باشند، این مدل بر یادگیری هدف و دقت تأثیر منفی خواهد گذاشت، که به آن **انتقال منفی** می‌گویند.

۱۲.۱.۱.۱ تفاوت یادگیری انتقالی با یادگیری چندوظیفه‌ای

یادگیری چندوظیفه‌ای متفاوت از یادگیری انتقالی است، و تفاوت آن‌ها در نحوه انتقال دانش است. وظایف به‌طور متوالی در یادگیری انتقالی یاد گرفته می‌شوند و از یکی به یکی دیگر منتقل می‌شود. در حالیکه، یادگیری چندوظیفه‌ای با به‌اشتراک گذاشتن اطلاعات بین همه وظایف به‌دنبال عملکرد خوب در تمام وظایف در نظر گرفته شده توسط یک مدل واحد به‌صورت هم‌زمان به‌طور موازی است.

۱۳.۱.۱.۱ یادگیری بازنمایی[۱]

یادگیری بازنمایی زیرمجموعه‌ای از یادگیری ماشین است که هدف آن بدست آوردن ویژگی‌های خوب و مفید از داده‌ها به‌طور خودکار، بدون آن که یک طراح ویژگی درگیر با مساله باشد. از آن‌جا که این روش می‌تواند به عنوان یادگیری ویژگی‌های معنادار تفسیر شود، به آن یادگیری ویژگی نیز گفته می‌شود. هر چند این روش‌ها در بیشتر اوقات برای یافتن بازنمایی خوب در مسائل بانظارت همانند: طبقه‌بندی و رگرسیون استفاده می‌شوند؛ با این همه یادگیری بازنمایی بدون‌نظارت نیز امکان‌پذیر می‌باشد. یادگیری بازنمایی در داده‌های خام، مشاهده اطلاعات مفید هنگام ساخت هر مدلی برای مسائل طبقه‌بندی، پیش‌بینی و تولید را آسان‌تر می‌کند.

۱۴.۱.۱.۱ داده‌های آموزشی، آزمایشی و اعتبارسنجی

هر چند الگوریتم‌های یادگیری ماشین بانظارت ابزارهای شگفت‌انگیز و قدرتمند در پیش‌بینی و دسته‌بندی هستند، اما این سوال به‌وجود می‌آید که این پیش‌بینی‌ها تا چه اندازه دقیق هستند، و آیا راهی برای سنجش میزان کارایی مدل وجود دارد؟ از آنجایی که این الگوریتم‌ها دارای نمونه‌های برچسب خورده می‌باشند، این پرسش را می‌توان با

[۱] Representation learning

تقسیم نمونه‌های آموزشی به چندین بخش، پاسخ داد. با تقسیم‌بندی داده‌ها ابتدا، آموزش را روی بخشی از داده‌ها انجام داده، سپس برای سنجش میزان کارای مدل و قابلیت **تعمیم‌دهی** آن از داده‌های آزمایشی استفاده می‌کنیم. تعمیم‌دهی نشان‌دهنده میزان عملکرد مدل، در برخورد با داده‌هایی است، که تاکنون مدل آن‌ها را در فرآیند آموزش مشاهده نکرده است. البته در طراحی مدل‌های یادگیری ماشین در بیشتر اوقات مجموعه داده‌های مسئله مورد نظر را علاوه بر داده‌های آموزشی و آزمایشی به بخش دیگری نیز تقسیم می‌کنیم، نحوه این تقسیم‌بندی به‌صورت زیر می‌باشد:

- **مجموعه آموزشی:** به‌طور معمول بزرگ‌ترین در بین این سه دسته داده‌ها می‌باشد و برای یافتن پارامترهای مدل مورد استفاده قرار می‌گیرد. مجموعه داده آموزشی رابطه اساسی بین داده‌ها و برچسب‌های آن را به بهترین وجه ممکن توضیح می‌دهد.
- **مجموعه آزمایشی:** اندازه‌گیری عملکرد یک مدل را براساس توانایی مدل در پیش‌بینی داده‌هایی که در فرآیند یادگیری نقشی نداشته می‌سنجند، مجموعه آزمایشی همان داده‌های دیده نشده در فرآیند یادگیری هستند. این مجموعه کارایی مدل نهایی را می‌سنجد. اگر مدلی عملکرد خوبی در مجموعه آموزشی داشته باشد و همچنین متناسب با مجموعه آزمون باشد، یعنی برچسب درست را برای تعداد زیادی از داده‌های ورودی نادیده پیش‌بینی کند، حداقل بیش‌برازش صورت گرفته است. لازم به ذکر است که مجموعه آزمون معمولاً فقط یک بار به محض مشخص شدن کامل پارامترها و ابرپارامترهای مدل برای ارزیابی عملکرد تعمیم‌دهی مدل استفاده می‌شود. با این حال، برای تقریب عملکرد پیش‌بینی یک مدل در طول آموزش، از یک مجموعه اعتبارسنجی استفاده می‌شود.
- **مجموعه اعتبارسنجی:** در ارزیابی انواع مختلف مدل‌ها و الگوریتم‌ها برای مسئله مورد نظر از مجموعه اعتبارسنجی استفاده می‌شود. از این داده‌ها برای تنظیم ابرپارامترها و جلوگیری از بیش‌برازش مدل استفاده تا بهترین مدل انتخاب شود.

۲.۱ یادگیری عمیق

مغز باورنکردنی‌ترین عضو بدن یک انسان است. این امکان را به ما می‌دهد تا خاطرات را ذخیره کرده، احساسات و یا حتی رویاها را تجربه کنیم. بدون آن ما یک موجود زیستی ابتدایی هستیم که از ساده‌ترین واکنش‌ها ناتوانیم. مغز ذاتا همان چیزی است که ما را در زمره موجودات هوشمند قرار می‌دهد.

مغز یک نوزاد وزنی کم‌تر از نیم کیلو گرم دارد. اما، قادر به حل مسائلی است که حتی بزرگ‌ترین و قدرتمندترین ابر رایانه‌های ساخته‌ی دست بشر چنین توانایی را ندارند. با گذشت چند ماه پس از تولد، نوزاد می‌تواند چهره والدین خود را تشخیص داده، اشیا مخالف را شناسایی و حتی صداها را از یک‌دیگر تفکیک دهد. در اوایل کودکی، آن‌ها درک واضحی از دستور زبان داشته و هزاران واژه را به خاطر سپرده‌اند.

در طول چند سال گذشته انسان در تلاش برای ایجاد ماشین‌هایی هوشمند همانند: ربات‌هایی که خانه‌ها را تمیز کنند، اتوموبیل‌هایی که خودشان رانندگی کنند و سیستم‌هایی که به‌طور خودکار بیماری‌ها را تشخیص دهند، می‌باشد. ساختن چنین ماشین‌های مصنوعی باهوش، نیاز به حل برخی پیچیده‌ترین مسائل محاسباتی که تاکنون با آن‌ها دست و پنجه نرم کرده‌ایم را دارد؛ مسائلی که مغز انسان قادر به حل آن‌ها در میکروثانیه هستند. در جهت مقابله با این مسائل، باید از رویکردی کاملا متفاوت برای برنامه‌نویسی سنتی در رایانه که در طول دهه گذشته توسعه یافته‌اند، استفاده شود. این همان زمینه فعال هوش مصنوعی است که از آن به عنوان یادگیری عمیق یاد می‌شود.

یادگیری عمیق زیرمجموعه‌ای از یادگیری ماشین است و به مطالعه و توسعه ماشین‌هایی متمرکز بوده که می‌توانند یادگیری داشته باشند. در تعریفی دقیق‌تر: یادگیری عمیق با پردازش در داده‌ها و مشابه با یک انسان، از طریق یادگیری مثال‌های از قبل آموخته، سعی در استخراج ویژگی‌های خاص به‌صورت خودکار، از طریق تعداد لایه‌های دنباله‌داری که در ساختارش وجود دارد، در جهت ایجاد الگویی است، برای تصمیم‌گیری در حل یک مسئله. وجود این تعداد لایه‌های مختلف به یادگیری عمیق این امکان را می‌دهد که بتواند در هر لایه، ویژگی‌های خاصی از مسئله را کشف و از آن در جهت تصمیم‌گیری بهتر در حل مسئله استفاده کند.

۱.۲.۱ تاریخچه یادگیری عمیق

یادگیری عمیق برخلاف انتظار تاریخی طولانی دارد. شاید کمی تعجب‌آور به‌نظر برسد اما، یادگیری عمیق از دهه ۱۹۴۰با عناوین متفاوتی همانند: فرمان‌شناسی[۱]، ارتباط‌گرایی[۲] و مشهورترین آن یعنی شبکه‌های عصبی وجود داشته است.

اولین مدل از شبکه‌های عصبی توسط مک کلاچ[۳] و پیتز[٤] در سال ۱۹٤۳ ارائه گردید. این شبکه یک دسته‌بند دودویی بود که می‌توانست دو دسته مختلف را براساس مقادیر ورودی تشخیص دهد. مشکل این شبکه تنظیم وزن‌ها توسط یک اپراتور انسانی بود. پس از آن، در سال ۱۹۵۷ الگوریتم پرسپترون توسط روزنبلات[۵] ارائه شد که می‌توانست، وزن‌ها را برای دسته‌بندی داده‌ها در ساختار خودش بدون دخالت یک اپراتور انسانی یاد بگیرد. در حالی که روش پرسپترون در طول چند سال استفاده می‌شد، در سال ۱۹٦۹ مینسکی[٦] و پاپرت[۷] با انتشار مقاله‌ای نشان دادند که، پرسپترون تنها قادر به دسته‌بندی مسائل خطی بوده، و مسائل غیرخطی توسط این روش حل‌پذیر نیستند. همچنین نویسندگان این مقاله در همان سال ادعا کردند که، منابع محاسباتی لازم جهت ساخت شبکه‌های عصبی بزرگ و عمیق وجود ندارد، همین ادعا سبب شد تا مقالات روی شبکه‌های عصبی به سمت نابودی پیش رود. اما خوشبختانه ارائه الگوریتم پس‌انتشار توسط وربوس[۸] (۱۹۷٤)، روملهات[۹] (۱۹۸٦) و لکان[۱۰] (۱۹۹۸) سبب احیای زودهنگام و دوباره‌ی شبکه‌های عصبی گردید. در این تحقیقات آن‌ها توانستند شبکه عصبی چندلایه را آموزش دهند.

امرزوه الگوریتم پس‌انتشار پایه و اساس شبکه‌های عصبی هستند، که می‌توانیم شبکه را توسط آن آموزش داده، و همچنین توسط آن از اشتباهات خود یاد بگیرند. اما در آن

---

۱ cybernetics
۲ connectionism
۳ McCulloch
٤ Pitts
۵ Rosenblatt
٦ Minsky
۷ Papert
۸ Werbos
۹ Rumelhart
۱۰ Lecun

زمان به دلیل رایانه‌های ضعیف و نبود مجموعه داده‌های بزرگ، نتواستند شبکه عصبی با بیش از دو لایه پنهان آموزش دهند. اما امروزه با افزایش قدرت سخت‌افزارها و عصر داده‌های بزرگ که سبب فراهم اوردن داده‌های زیادی برای آموزش شبکه وجود دارد، می‌توان شبکه‌هایی با بیش از چند لایه پنهان را اموزش داد. شبکه‌های عصبی که از چندین لایه تشکیل شده باشند را، **شبکه‌های عمیق** گویند. امروزه وقتی در حال استفاده از شبکه‌های عمیق هستیم، منظور همان یادگیری عمیق می‌باشد.

۲.۲.۱ یادگیری عمیق چگونه کار می کند؟

مدل‌های یادگیری عمیق با تجزیه و تحلیل مداوم داده‌ها و با کشف ساختارهای پیچیده در داده‌ها یادگیری را فرا می‌گیرند. فرآیند یادگیری با ساخت مدل‌های محاسباتی به نام شبکه‌های عصبی که از ساختار مغز الهام گرفته شده، حاصل می‌شود. ساختار این شبکه از چندین لایه پردازشی تشکیل شده است. یادگیری عمیق، به دنبال بهره‌وری از ساختار ناشناخته در توزیع ورودی، به‌منظور کشف بازنمایی خوب توسط ساختاری سلسله مراتبی از مفاهیم که همان لایه‌های پردازشی هستند، می‌باشد.

در این ساختار با رفتن به لایه‌های سطح بعدی، قادر به حل مفاهیم پیچیده‌تری از مسئله می‌شود. لایه‌های اولیه داده‌های خام را پردازش و لایه‌های بعدی قادر به استفاده از اطلاعات نرون‌های موجود در لایه‌های قبلی جهت بدست آوردن اطلاعات پیچیده‌تر از داده‌ها را فراهم می‌سازند. به عنوان مثال، در پردازش یک تصویر، لایه‌ی ورودی هر یک پیکسل از تصویر را پردازش می‌کند. لایه‌های بعدی گروهی از پیکسل‌ها را پردازش و اطلاعاتی از داده‌ها را بدست می‌آورند. لایه‌های اولیه ممکن است متوجه شوند که برخی پیکسل‌ها تاریک‌تر از بقیه هستند، حال آن‌که لایه‌های بعدی ممکن است متوجه شوند که از گروهی پیکسل‌ها ساختار چشم به‌نظر را نشان می‌دهد، و لایه‌ای خیلی عمیق متوجه شود که کل یک تصویر مربوط به یک انسان است.

۳.۲.۱ علت محبوبیت یادگیری عمیق

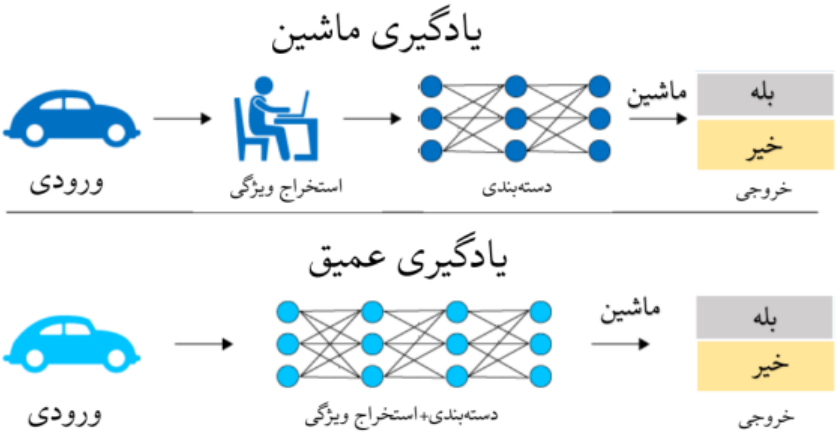
تا قبل از پیدایش یادگیری عمیق برای چندین دهه، سیستم‌های یادگیری ماشین نیازمند یک متخصص در دامنه مربوط به موضوع داشت، تا استخراج ویژگی‌ها را به‌صورت دستی انجام دهد. انتخاب ویژگی برای یک مجموعه داده تاثیر بسیار زیادی در موفقیت

یک مدل یادگیری ماشین دارد، حال آن که این استخراج ویژگی‌ها به‌صورت دستی فرآیندی زمان‌بر و پیچیده خواهد بود.

یادگیری عمیق این امکان را می‌دهد، تا داده‌های خام را به بردار تبدیل کرده و به شبکه تغذیه کرد بدون این‌که استخراج ویژگی را به‌صورت دستی روی این داده‌ها انجام دهیم. یادگیری عمیق این استخراج ویژگی‌ها را حتی بهتر از یک متخصص توسط تعداد لایه‌های مختلفی که در ساختارش وجود دارد، در جهت حل مسئله مورد نظر برمبنای داده‌های آموزشی مربوط به مسئله انجام می‌دهد. این لایه‌ها می‌توانند به صورت مستقیم و به تنهایی نمایشی انتزاعی از داده‌های خام را بیاموزند. شکل ۱ــ۲ تفاوت این عمل را در یادگیری ماشین و یادگیری عمیق نشان می‌دهد. در این‌جا، یک نمایش انتزاعی و فشرده از داده‌های خام در چندین لایه در شبکه تولید می‌شود. سپس، از این نمایش فشرده شده داده‌های ورودی برای تولید خروجی استفاده می‌شود. به عنوان مثال، طبقه‌بندی داده‌های ورودی به دسته‌های مختلف. همچنین در طی فرآیند آموزش شبکه، این مرحله توسط شبکه عصبی بهینه می‌شود تا بهترین نمایش انتزاعی ممکن از داده‌های ورودی را بدست آورد. این بدان معناست که مدل‌های یادگیری عمیق برای انجام و بهینه‌سازی فرآیند استخراج ویژگی، به تلاش دستی بسیار کمی نیاز دارند. برای مثال، اگر بخواهیم از یک مدل یادگیری عمیق برای شناسایی تصویر یک خودرو استفاده کنیم، در ابتدا باید ویژگی‌های خاصی از خودرو (شکل، اندازه، چرخ) را استخراج کرده و آن‌ها را به ورودی الگوریتم بدهیم. به این ترتیب الگوریتم، طبقه‌بندی را انجام می‌دهد. یعنی، یک برنامه‌نویس باید مستقیما وارد عمل شده تا مدل نتیجه مطلوب را بدست آورد. حال آن‌که، در یادگیری عمیق استخراج ویژگی‌ها در داخل خود مدل بدون دخالت انسانی تشخیص داده شده تا پیش‌بینی مطلوب را برای مسله انجام دهد.

مرحله استخراج ویژگی بخشی از فرآیندی است که در شبکه مصنوعی عمیق انجام می‌شود.

دومین مزیت بزرگ یادگیری عمیق، این است که با حجم زیادی از داده‌ها تامین می‌شود. مدل‌های یادگیری عمیق با افزایش مقدار داده‌های آموزشی میل به افزایش دقت دارند. در مقابل، مدل‌های سنتی یادگیری ماشین، پس از یک نقطه اشباع، مدل دیگر بهبودی را به همراه ندارد.



شکل ۱ ـ ۲ تفاوت یادگیری ماشین و یادگیری عمیق در استخراج ویژگی

۴.۲.۱ دلیل اهمیت یادگیری عمیق

ما در یک زمان بی‌سابقه زندگی می‌کنیم، جایی که فناوری یادگیری عمیق در دستیابی به بسیاری از موفقیت‌های جدید کمک کننده بوده و نقش موثری را در کشف سیارات فراخورشیدی، کشف داروهای جدید، تشخیص بیماری‌ها و ذرات زیراتمی داشته است. ما همچنین در دوره‌ای زندگی می‌کنیم که با چالش‌های بی‌وقفه روبه‌رو هستیم. تغییرات آب و هوایی تولید غذا را تهدید می‌کند و می تواند روزی به دلیل منابع محدود منجر به جنگ شود. چالش تغییرات محیط زیستی به سبب افزایش روز افزون جمعیت انسانی در حال تشدید است. دامنه و مقیاس این چالش‌ها نیاز به سطح جدیدی از هوش دارد که با یادگیری عمیق امکان‌پذیر می‌شود.

۵.۲.۱ چالش‌های موجود در یادگیری عمیق

یادگیری عمیق، پیشگام هوش مصنوعی و یکی از هیجان‌انگیزترین فناوری‌های دهه اخیر می‌باشد. و در حال حاضر در زمینه‌های مختلف همانند: تشخیص گفتار، تشخیص سرطان، اتوموبیل‌های خودران و دامنه‌هایی که قبلا بسته به نظر می‌رسید، کاربرد گسترده‌ای دارند. به عقیده برخی کارشناسان این روند با سرعت بیشتری ادامه خواهد یافت و دامنه‌های دیگری را نیز فتح خواهد کرد.

در برخی از این موارد این ترس وجود خواهد داشت که یادگیری عمیق، اساس اقتصاد و اجتماع زندگی جامعه بشری را تهدید، و انسان را به بیکاری یا برده‌داری سوق دهد. با این همه که یادگیری عمیق در بسیاری از کارها بسیار کارآمد عمل کرده است، آن‌ها هنوز نتوانسته‌اند بر همه فناوری‌ها غلبه کنند. دلیل آن هم محدودیت‌ها و چالش‌هایی است که در مقایسه با ذهن انسان دارد. انسان می‌تواند روابط انتزاعی و گسترده‌ای بین مفاهیم مختلف را با اندک اطلاعات بیاموزد و در تصمیم‌گیری از آن‌ها استفاده کند. در مقابل، الگوریتم‌های یادگیری عمیق در این قابلیت نیاز به داده‌های دقیق و زیاد دارند. گری مارکوس در مقاله‌ای با عنوان "یادگیری عمیق: ارزیابی انتقادی" در این مورد چنین می‌گوید:

"یادگیری عمیق امروزه فاقد سازوکاری برای یادگیری انتزاعی از طریق تعریف صریح و کلامی است و هنگامی بهترین عملکرد را دارد که هزارن، میلیون‌ها و یا حتی میلیاردها نمونه اموزشی وجود داشته باشد".

یکی دیگر از مشکلات الگوریتم‌های یادگیری عمیق آن است که آن‌ها تنها در نگاشت بین ورودی و خروجی بسیار خوب هستند اما، در درک زمینه داده‌هایی که از آن‌ها استفاده می‌کنند خوب نیستند. در حقیقت، کلمه "عمیق" در یادگیری عمیق بیشتر اشاره به مرجع معماری فناوری و تعداد لایه‌های پنهان است که در ساختار آن قرار دارد نه درک عمیقی از آنچه که در حال انجام آن است.

چالش دیگر در یادگیری عمیق را می‌توان، عدم شفافیت دانست. در حالی که، تصمیمات گرفته شده توسط مدل‌های مبتنی بر قاعده[۱] را می‌توان توسط دستورات if و $else$ ردیابی کرد، در یادگیری عمیق چنین چیزی امکان پذیر نخواهد بود. این عدم شفافیت همان چیزی است که در یادگیری عمیق از آن به عنوان "**جعبه سیاه**" یاد می‌شود.

الگوریتم‌های یادگیری عمیق، الگوها و همبستگی‌ها را از طریق داده‌های تغذیه شده به آن پیدا می‌کنند و گاهی تصمیماتی می‌گیرند که حتی برای مهندسانی که آن‌ها را ایجاد کرده‌اند، گیج کننده است. این امر زمانی که یادگیری عمیق کاری با اهمیت کم را انجام می‌دهد، مشکلی را به‌وجود نخواهد آورد. اما وقتی تصمیم به سرنوشت یک متهم در

[۱] rule-based

دادگاه یا معالجه پزشکی بیمار باشد، بسیار سرنوشت‌ساز خواهد بود. چرا که اشتباهات می‌تواند عواقب زیادی را به‌همراه داشته باشد. طبق گفته مارکوس:

"مسئله شفافیت هنوز حل نشده است، کاربران هنگام استفاده از یادگیری عمیق برای کار در حوزه‌های تشخیص پزشکی و تجارت مالی، دوست دارند درک کنند که چگونه یک سیستم مشخص یک تصمیم مشخصی را گرفته است."

همچنین در مقاله خود پیشنهاد می‌کند که یادگیری عمیق باید با فناوری‌های دیگری همانند برنامه نویسی مبتنی‌بر قانون ساده و سایر روش‌های هوش مصنوعی همانند یادگیری تقویتی ترکیب شوند. متخصصان دیگری همانند، پاسکال کافمن، علوم اعصاب را کلید اصلی هوش مصنوعی واقعی می‌دانند که قادر به دستیابی به حل مسئله همانند انسان خواهد بود.

با این همه یادگیری عمیق در حال حاضر، برای مسائل دسته‌بندی در صورت وجود مجموعه داده‌های آموزشی کافی عملکرد خوب و کارایی را از خود نشان می‌دهد.

۶.۲.۱ مقایسه بین یادگیری ماشین و یادگیری عمیق

یکی از مهم‌ترین تفاوت‌های که یادگیری عمیق را در تقابل با یادگیری ماشین قرار می‌دهد، عملکرد سیستم بر اساس افزایش تعداد نمونه‌های آموزشی است. یادگیری عمیق در صورت عدم وجود نمونه‌های اموزشی کافی نتایج مطلوبی را به‌همراه نخواهند داشت. در مقابل یادگیری ماشین می‌تواند با تعداد نمونه‌های کم هم نتایج خوبی را از خود نشان دهد. همچنین استفاده از یادگیری عمیق نیاز به سخت‌افزارهای پیشرفته داشته، حال آن‌که یادگیری ماشین می‌تواند با سخت‌افزار و رایانه‌هایی با قدرت پایین هم قابل استفاده باشد.

تفاوت مهم و کلیدی که قدرت یادگیری عمیق را در مقابل یادگیری ماشین نشان می‌دهد، استخراج ویژگی‌ها به‌صورت خودکار در این الگوریتم‌ها هستند. به‌طور خلاصه می‌توان مقایسه بین یادگیری ماشین و یادگیری عمیق را در جدول ۱ ــ ۲ مشاهده کرد.

جدول ۱ــ۲ مقایسه بین دو رویکرد یادگیری ماشین و یادگیری عمیق

معیار	یادگیری ماشین	یادگیری عمیق
مهندسی ویژگی‌ها	نیاز به درک ویژگی‌ها در داده‌ها	بدون نیاز به استخراج ویژگی دستی
وابستگی به داده‌ها	عملکرد عالی روی داده‌های کم و متوسط	عملکرد عالی روی داده‌های بزرگ
وابستگی به سخت‌افزار	روی ماشین‌های ضعیف هم کار می‌کند	نیاز به ماشین قدرتمند
زمان اجرا	از چند دقیقه تا چند ساعت	گاهی اوقات تا چند هفته

۳.۱ رابطه بین هوش مصنوعی، یادگیری ماشین و یادگیری عمیق

اگرچه اصطلاح هوش مصنوعی، یادگیری ماشین و یادگیری عمیق در بیشتر اوقات به‌جای هم مورد استفاده قرار می‌گیرند و به یک‌دیگر مرتبط هستند، اما به‌طور کامل به موارد مشابهی اشاره نمی‌کنند. در شکل ۱ــ۳ نحوه ارتباط آن‌ها با یک‌دیگر قابل نمایش است، و همان‌طور که مشاهده می‌شود، یادگیری عمیق زیرمجموعه‌ای از یادگیری ماشین و همچنین هوش مصنوعی است.

شکل ۱ــ۳ ارتباط بین هوش مصنوعی، یادگیری ماشین و یادگیری عمیق

برای درک تفاوت و تفکیک این سه گروه از یک‌دیگر می‌توان آن‌ها را اینگونه تشریح کرد:

هوش مصنوعی: همان‌طور که از نام آن پیداست هوش مصنوعی ترکیب هوش انسانی در ماشین است، به‌نحوی که رفتاری همانند انسان را تقلید و خلاقانه مسائل را حل کند. به‌طور دقیق‌تر هوش مصنوعی، **در تلاش برای روشی است که رونوشتی از مغز انسان را پیاده‌سازی کند، یعنی همان‌گونه که یک انسان فکر می‌کند، کار می‌کند، و عمل می‌کند**. برای مثال، چنین سیستم‌هایی قادرند، اشیا را شناسایی و تشخیص داده، آن را جابه‌جا کرده و یا کارهایی دیگری را انجام دهد. تاکنون سیستمی که بتواند به سطح کامل هوش انسانی دست پیدا کند ساخته نشده، و دلیل آن هم عدم درک کامل مغز انسان است. گرچه که ساخت چنین سیستمی روز به روز در تبدیل به واقعیت نزدیک‌تر می‌شود.

یادگیری ماشین: یادگیری ماشین که زیرمجموعه‌ای از هوش مصنوعی است، رایانه را به‌گونه‌ای توانمند می‌سازد، که قادر به یادگیری از طریق تجربه و بدون برنامه‌ریزی صریح می‌شود. این یادگیری توسط داده‌هایی که متناسب به هر مسئله به آن داده می‌شود، رابطه‌ای را میان ورودی و خروجی مسئله پیدا می‌کنند تا در مواجه با مسئله‌ای مشابه از آن استفاده کند. یادگیری ماشین اینگونه آموزش می‌بیند که، **چگونه یک تصمیم‌گیری برای مسئله انجام دهد**، و روشی برای تحقق بخشیدن به هوش مصنوعی است.

یادگیری عمیق: یادگیری عمیق زیرمجموعه‌ای از یادگیری ماشین بوده و از ساختار شبکه‌های عصبی برای تقلید در تصمیم‌گیری حل یک مسئله مشابه مغز انسان استفاده می‌کند، و همان کار یادگیری ماشین را انجام می‌دهد، ولی قابلیت‌های متفاوتی در آن وجود دارد. در مقایسه یادگیری ماشین با یادگیری عمیق اینگونه می‌توان بیان کرد، در حالی‌که یادگیری عمیق به‌طور خودکار ویژگی‌ها را از ساختار داده‌ها استخراج می‌کند، این عمل توسط یادگیری ماشین باید به‌صورت دستی انجام گیرد و اگر در تصمیم‌گیری حل مسئله پیش‌بینی‌های نادرستی را انجام دهد، آنگاه متخصص یا برنامه‌نویس باید صراحتا به حل این مشکل بپردازد. پس یادگیری عمیق را می‌توان نسخه‌ای تکامل یافته و پیشرفته‌ای از یادگیری ماشین دانست.

همان‌طور که دیدیم، یادگیری ماشین و یادگیری عمیق در راستای تحقق بخشیدن به هوش مصنوعی تلاش می‌کنند. با این امید که بتوانیم در آینده‌ای نزدیک به سطح کاملی از هوش مصنوعی دست پیدا کنیم. پس یادگیری ماشین و یادگیری عمیق ابزاری هستند، در جهت رسیدن به هوش مصنوعی. به بیانی دیگر هر الگوریتم یادگیری ماشین که یادگیری عمیق هم زیرمجموعه آن است، هوش مصنوعی به‌حساب می‌آید، اما هر الگوریتم هوش‌مصنوعی یادگیری ماشین نیست. به عنوان مثال سیستم‌های خبره، هوش مصنوعی می‌باشد ولی یادگیری ماشین به‌حساب نمی‌آیند.

۴.۱ کاربردهای یادگیری عمیق

مواردی که امروزه از یادگیری عمیق استفاده می‌کنند، انواع برنامه‌های تجزیه‌وتحلیل کلان داده‌ها را شامل می‌شود. زمینه‌های خاصی که از یادگیری عمیق بهره می‌برد شامل موارد زیر است:

- **پردازش زبان طبیعی:** درک پیچیدگی‌های مرتبط با زبان، ساختار، معنا، تفاوت‌های ظریف در لحن، عبارات و کنایه‌ها یکی از سخت‌ترین کارها برای یادگیری انسان است. آموزش مداوم از بدو تولد و قرار گرفتن در محیط‌های مختلف اجتماعی به انسان کمک می‌کند تا پاسخ مناسب و شکل بیان شخصی خود را نسبت به سناریوهای مختلف بدست آورد. استفاده از یادگیری عمیق در پردازش زبان طبیعی در تلاش است تا با آموزش ماشین، با در نظر گرفتن تفاوت‌های زبانی و پاسخ‌دهی مناسب به همان سطح انسانی دست یابد. پردازش زبان طبیعی در بسیاری از مسائل خود همانند: تحلیل احساسات، مدل‌سازی زبان، طبقه‌بندی متن، بازیابی اطلاعات، تعبیه‌ساز کلمات، درک زبان گفتار، ترجمه ماشینی، سیستم‌های پرسش و پاسخ و غیره از یادگیری عمیق استفاده می‌کند.
- **رباتیک:** تحولات اخیر در رباتیک ناشی از پیشرفت در هوش مصنوعی و یادگیری عمیق است. هوش مصنوعی ربات‌ها را قادر می‌سازد تا محیط خود را درک و به آن واکنش نشان دهند. این تحولات به این معنی است که ما می‌توانیم انتظار داشته

باشیم که از ربات‌ها در آینده‌ای نزدیک به‌طور فزاینده‌ای به عنوان دستیار انسانی استفاده به عمل آید.

- **دستیارهای مجازی:** یکی از کابردهای گسترده و مشهور یادگیری عمیق دستیارهای مجازی است. هر تعامل با این دستیارها سبب فرصتی برای آن‌ها می‌شود تا درباره صدا و لهجه شما بیشتر اطلاعات کسب کند. در نتیجه یک تجربه تعاملی انسانی ثانویه را برای شما به ارمغان می‌آورد. دستیاران مجازی از یادگیری عمیق برای کسب اطلاعات بیشتر در مورد موضوعات خود استفاده می کنند، از تنظیمات شام شما گرفته تا مکان های پر بازدید یا موسیقی‌های مورد علاقه. آن‌ها یاد می‌گیرند که دستورات شما را با ارزیابی زبان طبیعی انسان درک کنند. یکی دیگر از قابلیت های دستیاران مجازی این است که سخنرانی خود را به متن ترجمه کنید، برای شما یادداشت بنویسند و قرار ملاقات‌های شما را رزرو کنند. دستیاران مجازی به معنای واقعی کلمه آماده به خدمت شما هستند. چرا که، آن‌ها می‌توانند همه‌ی کارها را انجام دهند، از پاسخگویی خودکار به تماس‌های خاص شما تا هماهنگی بین شما و اعضای تیم.

- **اتوماسیون صنعتی:** یادگیری عمیق در ارتقا ایمنی کارگران در محیط‌هایی مانند کارخانه‌ها و انبارها خدماتی ارائه می‌دهد تا به‌طور خودکار زمان نزدیک شدن کارگر یا اشیا به ماشین را تشخیص دهد.

- **کشاورزی:** یادگیری عمیق می‌تواند انقلابی در کشاورزی ایجاد کند. یادگیری عمیق امروزه کشاورزان را قادر می‌سازد تا تجهیزات لازم را برای افتراق گیاهان زراعی و علف‌های هرز به‌کار گیرد. این قابلیت به ماشین‌ها این توانایی را می‌دهد تا به‌صورت انتخابی علف کش‌ها را روی علف‌های هرز پاشیده و گیاهان دیگر را دست نخورده بگذارد. همچنین، ماشین‌های کشاورزی که از بینایی رایانه‌ای با قابلیت یادگیری عمیق بهره می‌برند، می‌تواند با پاشش انتخابی علف کش‌ها، کودها، قارچ کش‌ها، حشره کش‌ها می‌توانند گیاهان بخصوصی را بهینه کنند. علاوه بر کاهش استفاده از علف کش و بهبود تولیدات مزرعه، از یادگیری عمیق می‌توان در سایر عملیات کشاورزی مانند: استفاده از کود، انجام آبیاری و برداشت نیز گسترش داد.

- **شناخت گونه‌های دریایی:** تحقیقات در مورد شناسایی گونه‌های دریایی بخش مهمی از اقدامات برای حفاظت از محیط اقیانوس است. با پیشرفت‌های قابل توجه یادگیری عمیق، علاقه به این موضوع بیشتر شده است.
- **تحقیقات پزشکی:** محققین سرطان شروع به استفاده از یادگیری عمیق در جهت شناسایی راهی برای شناسایی خودکار سلول‌های سرطانی کرده‌اند.
- **تصویربرداری پزشکی:** اخیرا استفاده از تکنیک‌های یادگیری عمیق به‌طور گسترده‌ای برای تجزیه و تحلیل تصاویر پزشکی مورد استفاده قرار گرفته و نتایج دلگرم‌کننده‌ای به‌ویژه برای مجموعه داده‌های بزرگ از خود نشان داده است.
- **سیستم‌های توصیه‌گر:** از یادگیری عمیق در سیستم‌های توصیه‌گر برای استخراج ویژگی‌های معنادار برای توصیه‌ها استفاده می‌شود.
- **تشخیص اشارات و حرکت:** تشخیص حرکت یکی از زمینه‌های جدید در یادگیری ماشین است و مربوط به شناخت حرکات صورت انسان است. سیگنال‌های ساطع شده از سنسورها قادرند با انرژی، تاخیر زمانی و تغییر فرکانس احساسات و یا حتی شی و خصوصیات آن را شناسایی کند.
- **تشخیص تاخیر رشد در کودکان:** اختلالات گفتاری، اوتیسم و اختلالات رشد می‌تواند کیفیت زندگی خوب کودک را از بین ببرد. تشخیص و درمان به‌موقع می‌تواند تاثیر شگفت‌انگیزی بر سلامت جسمی، روحی و روانی کودک داشته باشد. از همین رو، یکی از خاص‌ترین کابردهای یادگیری عمیق، تشخیص زودهنگام و دوره بهبودبخشی این مشکلات مربوط به کودکان است.

یک تفاوت عمده بین یادگیری ماشین و یادگیری عمیق این است که، از یادگیری ماشین اغلب فقط برای کارهای خاص استفاده می‌شود. در مقابل، یادگیری عمیق کمک کننده به حل جدی‌ترین مشکلات نژاد بشری است.

خلاصه فصل

- یادگیری را می‌توان بهتر شدن عملکرد در یک وظیفه خاص با استفاده تجربه و تمرین دانست.
- هرگونه برنامه رایانه‌ای که کارایی‌اش در یک کار خاص با تجربه بهبود یابد دارای یادگیری است.
- یادگیری ماشین با سه رویکرد متفاوت قابل پیاده‌سازی و اجرا خواهد بود: یادگیری بانظارت، یادگیری بدون‌نظارت و یادگیری تقویتی
- یادگیری ماشین بر طراحی مدل‌هایی متمرکز است که در یک حوزه خاص، الگوریتم‌های کامپیوتری برمبنای داده‌های آموزشی داده شده به مدل یادگیری را به‌صورت خودکار از طریق تجربه و آزمایش از داده‌ها بدست می‌آورد تا در مواجه با داده‌های جدید در همان حوزه بتواند رفتاری مشابه انسان از خود نشان دهد.
- یادگیری عمیق می‌تواند برخلاف یادگیری ماشین استخراج ویژگی‌ها را به‌صورت خودکار انجام دهد.

پرسش‌های مروری

۱. چرا با وجود یادگیری عمیق هنوز هم از یادگیری ماشین استفاده می‌شود؟

۲. یادگیری عمیق استخراج ویژگی‌ها را چگونه و از چه طریقی انجام می‌دهد؟

۳. تفاوت مهم یادگیری عمیق و یادگیری ماشین در چیست؟

٤. چالش‌های موجود در یادگیری عمیق را شرح دهید؟

٥. چه ارتباطی بین هوش مصنوعی، یادگیری ماشین و یادگیری عمیق وجود دارد؟

فصل ۲

یادگیری بانظارت عمیق:
شبکه‌های عصبی پیش‌خور، بازگشتی و همگشتی

اهداف

- فرآیند یادگیری در شبکه‌های عصبی
- آشنایی با شبکه‌های پیش‌خور، بازگشتی و همگشتی
- چالش‌های آموزش در شبکه‌های عمیق

۰.۲ مقدمه

بسیاری از الگوریتم های یادگیری ماشین موجود از معماری‌های "کم‌عمق" استفاده می‌کنند. از جمله شبکه‌های عصبی با یک لایه پنهان، رگرسیون هسته، ماشین‌های بردار پشتیبان و بسیاری دیگر. نتایج نظری نشان می‌دهد که بازنمایی‌های آموخته شده توسط چنین سیستم‌هایی لزوماً ساده بوده و در استخراج انواع ساختارهای پیچیده از ورودی‌ها ناتوان هستند.

استدلال‌های نظری و بیولوژیکی نشان می‌دهند که در راستای ساخت سیستمی هوشمند با توانایی استخراج بازنمایی‌های سطح بالا و قدرتمند از این داده‌ها، نیاز به مدل‌هایی با معماری عمیقی است که شامل بسیاری از لایه‌های پردازشی غیرخطی می‌باشد. شاید بتوان گفت، بهترین و پرکاربردترین نمونه از این شبکه‌ها، به‌دلیل سازگاری آن با انواع داده‌ها، شبکه‌های عصبی چند لایه هستند.

۱.۲ شبکه‌های عصبی مصنوعی

هدف از کشف علوم جدید توسط انسان افزایش توانایی‌های انسانی است. برای پخت‌وپز غذا، آتش را اختراع کردیم؛ بنابراین، وابستگی خود را به توانایی اولیه فرآوری غذا در معده کاهش دادیم. این امر، منجر به افزایش مصرف کالری و شاید رشد تمدن شد، چیزی که هیچ‌گونه شناخته شده دیگری موفق به انجام آن نشد. چرخ و دیگر وسایل نقلیه را اختراع کردیم تا سرعت سفر محدود به پاها نشود.

داستان اختراع بشر و رشد تکنولوژی روایتی از گونه‌های منحصربه‌فرد طبیعت است که به‌طور بی‌وقفه از توانایی‌های خود فراتر رفته و بی‌نهایت افق‌های خود را گسترش داده و به آینده می‌رود. بیشتر این پیشرفت‌ها مربوط به ساختار مغز انسان است. سیستم عصبی انسان و توانایی‌های آن گسترده و پیچیده است. انسان از یک سیستم عصبی بسیار پیچیده با قابلیت تفکر، استدلال، احساس، تخیل و فلسفه و برخوردار است. فرآیند یادگیری نیز در سیستم عصبی داخل مغز انسان اتفاق می‌افتد. اگر بتوانیم درکی از چگونگی نحوه کار مغز بدست آوریم، شاید بتوان این توانایی را بدست آورد که رونوشتی از آن را در ماشین‌ها پیاده‌سازی کرد.

با وجود اینکه مغز یکی از اجزای بسیار پیچیده و قدرتمند در ساختار بدن موجودات می‌باشد، ساختمان اساسی سازنده‌ی آن نسبتا ساده و قابل درک می‌باشد. شبکه‌های عصبی مصنوعی رونوشتی از همین ساختار می‌باشند. شبکه‌های عصبی مصنوعی مدل‌های محاسباتی بوده که سازوکار یادگیری را همانند شبکه عصبی طبیعی از ساختار مغز انسان شبیه‌سازی می‌کند.

ساختار مغز انسان که همان شبکه‌ی عصبی طبیعی است، خود از تعدادی زیادی واحد ساده به نام نرون تشکیل شده است. نرون‌ها سه نوع جز تشکیل دهنده دارند: **دندریت‌ها**، **سوما** و **آکسون**. نمایی از آن را در شکل ۲ ـ ۱ می‌توان مشاهده کرد.

نرون‌ها با استفاده از آکسون‌ها و دندریت‌ها با یک‌دیگر اتصال پیدا می‌کنند. به مناطق اتصال‌دهنده بین آکسون و دندریت‌ها سیناپس گفته می‌شود. این نقاط اغلب در پاسخ به محرک‌های خارجی تغییر می‌کنند، همین تغییرات است که یادگیری در موجودات زنده را به‌وجود می‌آورد. هر نرون در داخل خودش با ایجاد فرآیندهای شیمیایی، سیگنالی از خود خارج کرده، سیگنال‌های ارسال شده از هر نرون توسط آکسون نرون به دندریت نرونی دیگر منتقل می‌شود. بر اساس این سیگنال‌های ارسالی، مغز کار خاصی را انجام می‌دهد. شبکه‌های عصبی مصنوعی یک شبیه‌سازی از ساختار مغز انسان است و بر پایه این فرضیات می‌باشد:

۱. پردازش اطلاعات در ساختارهای ساده با تعداد زیاد، به نام نرون‌ها انجام می‌گیرد.
۲. سیگنال‌ها از طریق اتصالات بین نرون‌های شبکه منتقل می‌شوند.
۳. هر اتصال، وزن مربوط به خود را دارد، که در یک شبکه عصبی این وزن‌ها در سیگنال انتقالی ضرب می‌شوند.
٤. هر نرون از طریق یک تابع فعال‌ساز، بر ورودی‌های خود که جمع وزن‌دار سیگنال‌های ورودی می‌باشد را اعمال کرده تا سیگنال خروجی را ایجاد نماید.

شکل ۲ ـ ۱ نمایی از یک شبکه عصبی طبیعی

با توجه با این فرضیات، الگوی اتصال بین نرون‌های مختلف آن شبکه را معماری شبکه و روش تعیین وزن‌ها بر اتصالات را **الگوریتم آموزشی** گویند.

یک شبکه عصبی از کنار قرار هم دادن چندین نرون تشکیل یک لایه را می‌دهد. در شبکه عصبی مصنوعی محاسباتی از تابع را با انتشار مقادیر محاسبه شده از نرون‌های ورودی به نرون (های) خروجی و استفاده از وزن‌ها به عنوان پارامتر واسطه محاسبه می‌کند. یادگیری با استفاده از تغییر وزن‌های متصل به نرون‌ها اتفاق می‌افتد. مطابق آنچه که محرک‌های خارجی برای یادگیری در ساختار مغز مورد نیاز است، در شبکه‌های عصبی مصنوعی این محرک خارجی، داده‌های آموزشی هستند، که نمونه‌ای حاوی جفت‌هایی از ورودی‌ـ خروجی می‌باشند.

برای مثال؛ داده‌های آموزشی ممکن است حاوی بازنمایی پیکسلی از تصاویر که ورودی مسئله و برچسب‌هایی مانند، سگ،گربه و خرگوش به عنوان خروجی باشد. این جفت داده‌های آموزشی با استفاده از بازنمایی ورودی به ساختار شبکه عصبی تغذیه شده تا پیش‌بینی مربوط به برچسب‌های خروجی را انجام دهد. داده‌های آموزشی بسته به میزان خروجی پیش‌بینی شده (به عنوان مثال: گربه) برای یک ورودی خاص بازخوردی در وزن‌های موجود در شبکه ارائه می‌کنند. وزن بین نرون‌ها در پاسخ به خطاهای پیش‌بینی شده در یک شبکه عصبی تنظیم می‌شوند. هدف از تغییر وزن‌ها اصلاح عملکرد محاسباتی است، تا پیش‌بینی‌ها در تکرارهای بعدی صحیح‌تر باشند. بنابراین، وزن‌ها با دقت و به صورت ریاضی تعدیل و تصدیق می‌شوند، تا خطای محاسباتی کاهش یابد.

با تنظیم پی‌درپی وزن‌ها بین نرون‌ها براساس ورودی و خروجی داده‌های آموزشی عملکرد محاسبه شده توسط شبکه‌های عصبی به‌مرور زمان پالایش می‌شود تا پیش‌بینی دقیق‌تری صورت گیرد. بنابراین، اگر شبکه عصبی با تصاویر مختلف و زیادی آموزش پیدا کند، در نهایت قادر خواهد بود تصویر گربه را به‌طور صحیح در تصویری که تاکنون آن را مشاهده نکرده است، بشناسد.

رفتار یک شبکه عصبی را معماری آن شبکه شکل می‌دهد. این معماری براساس موارد زیر تعریف می‌شود:

- تعداد نرون‌ها
- تعداد لایه‌ها
- نحوه اتصالات بین لایه‌ها

شناخته شده‌ترین معماری یک شبکه عصبی که از آن به عنوان شبکه عصبی چند لایه که همچنین پرسپترون چندلایه نیز خوانده می‌شود، می‌توان نام برد، که از ۳ لایه تشکیل شده است: **لایه ورودی**، **لایه پنهان** و **لایه خروجی**. لایه ورودی اطلاعات را دریافت، یک یا چند لایه پنهان عمل پردازش را انجام داده و لایه خروجی نتایج را نشان می‌دهد. با افزایش تعداد لایه‌های پنهان، به سمت یک شبکه عمیق می‌رویم که توانایی حل مسائل پیچیده‌تر را نسبت به همتایان کم عمق خود دارا می‌باشد. در ادامه شبکه‌های عصبی تک‌لایه و چندلایه را معرفی خواهیم کرد.

۱.۱.۲ پرسپترون

پرسپترون[۱] که در سال ۱۹۵۷ در آزمایشگاه هوانوردی کرنل توسط فرانک روزنبلات[۲] اختراع شد، ساده‌ترین نوع یک شبکه عصبی مصنوعی و یک دسته‌بند دودویی می‌باشد. ساختار تشکیل دهنده این شبکه عصبی تنها یک لایه ورودی به همراه تنها یک خروجی است، از همین‌رو به آن شبکه عصبی تک‌لایه نیز گفته می‌شود. شمایی از آن در شکل ۲ــ۲ به تصویر کشیده شده است. همان‌طور که مشاهده می‌شود، در این شبکه تعداد زیادی ورودی وجود دارد که جمع وزن‌دار آن‌ها پس از محاسبه با استفاده از یک تابع فعال‌ساز خروجی را پیش‌بینی می‌کند. با داشتن لیستی از ورودی ها $x =$

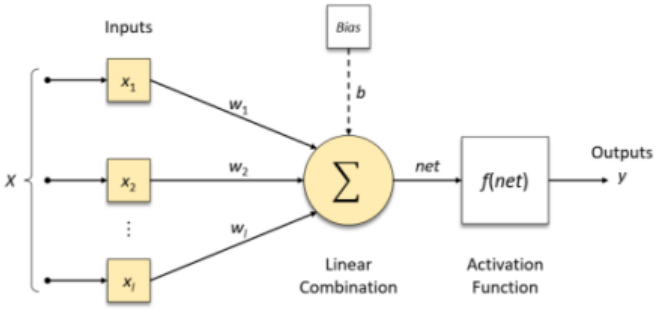
[۱] perceptron
[۲] Frank Rosenblatt

$\{x_1, x_2, \dots, x_n\}$ ،هر ورودی دارای یک بردار وزن به صورت $w = \{w_1, w_2, \dots, w_n\}$ خواهد بود. جمع وزن‌دار از طریق معادله ۲ ـ ۱ محاسبه می‌شود:

$$net = \sum_{i=0}^{n} w_i * x_i + b \qquad \text{معادله } ۲ - ۱$$

سپس تابع فعال‌ساز بر اساس حد آستانه θ خروجی y را بدست می‌آورد:

$$y = f(net) = \begin{cases} 1, if\ net \geq \theta \\ 0, if\ net < \theta \end{cases}$$



شکل ۲ ـ ۲ شمایی از یک پرسپترون

الگوریتم یادگیری پرسپترون

با داشتن فرضیات مناسب می‌توان نشان داد یادگیری در پرسپترون با تکرار الگوریتم آن، به وزن‌های درست همگرا خواهد شد. یعنی یادگیری شبکه منجر به تخمین وزن‌هایی خواهد شد که شبکه را قادر می‌سازد تا مقادیر درست را در خروجی براساس ورودی‌های مسئله تولید کند. در الگوریتم پرسپترون، برای هر بردار ورودی هنگام آموزش، شبکه خروجی را تولید می‌کند و آن را با مقدار درست مقایسه کرده تا مشخص شود آیا خطایی رخ داده است یا خیر. در این شبکه اگر خطایی رخ ندهد وزن‌ها تغییر نمی‌کند و آموزش تا زمانی که شبکه بدون خطا شود ادامه پیدا خواهد کرد.

فرض کنید یک بردار ورودی را به شبکه تغذیه کرده‌ایم و یکی از نرون‌ها جواب اشتباهی می‌دهد؛ یعنی مقدار ورودی با هدف یکسان نیست. m وزن برای این شبکه

وجود دارد، هر نرون متصل به گره ورودی یک وزن به خود اختصاص می‌دهد. اگر نرونی که خروجی اشتباه را تولید کرده است k بنامیم، وزن‌های مورد نظر w_{ik} هستند، که i می‌تواند از ۱ تا m باشد. بر این اساس می‌دانیم که کدام وزن‌ها تغییر می‌کنند. حال چگونگی تغییر وزن‌ها را بررسی می‌کنیم. اولین مسئله این است که آیا هر وزن خیلی بزرگ یا خیلی کوچک می‌باشد. محاسبه $y_k - t_k$ تفاوت بین خروجی(y_k) که نرون انجام داده و مقدار هدف (t_k) است که باید نرون انجام دهد. اگر این مقدار منفی شد، آن را بزرگتر می‌کنیم و برعکس اگر مثبت بود با کم کردن خطا می‌توان آن را کنترل کرد. نحوه تغییر وزن‌ها براساس معادله زیر انجام می‌گیرد:

$$\Delta w_{ik} = -(y_k - t_k) \times x_i$$

با این حال قبل از اینکه قانون یادگیری به پایان رسد، باید تصمیم بگیریم که وزن‌ها تا چه اندازه تغییر کنند. این کار با ضرب مقدار بالا در پارامتری به نام نرخ یادگیری انجام می‌شود، که معمولا توسط نماد η نمایش داده می‌شود. مقدار نرخ یادگیری سرعت یادگیری شبکه را تعیین می‌کند و بسیار مهم می‌باشد. بر این اساس معادله آخر به‌هنگام-سازی وزن‌ها را اینگونه می‌نویسیم:

$$w_{ij} \leftarrow w_{ij} - \eta(y_i - t_i).x_i$$

بر این اساس الگوریتم یادگیری پرسپترون را می‌توان به صورت زیر مشاهده کرد.

الگوریتم یادگیری پرسپترون

- **مقداردهی ابتدایی**

برای همه‌ی وزن‌ها $\boldsymbol{w_{ij}}$ مقادیر کوچکی تنظیم شود.

- **آموزش**

برای تکرار T تا زمانی که همه خروجی درست باشند:

-محاسبه تابع فعال‌سازی هر نرون j با استفاده از تابع فعال سازی $\boldsymbol{g}$:

$$y_j = g\left(\sum_{i=0}^{m} w_{ij}x_i\right) = \begin{cases} \mathbf{1}, \text{اگر } \sum_{i=0}^{m} w_{ij}x_i > 0 \\ \mathbf{0}, \text{اگر } \sum_{i=0}^{m} w_{ij}x_i \leq 0 \end{cases}$$

ـ هریک از وزن‌ها به‌صورت جداگانه به‌صورت زیر به‌هنگام می‌شود:

$$w_{ij} \leftarrow w_{ij} - \eta(y_i - t_i).x_i$$

- **فراخوانی**

ـ محاسبه تابع فعال‌سازی برای هر نرون j:

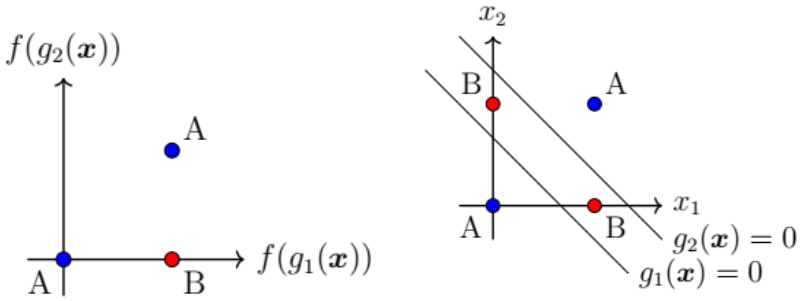
$$y_j = g\left(\sum_{i=0}^{n} w_{ij}x_i\right) = \begin{cases} \mathbf{1}, \text{اگر } w_{ij}x_i > 0 \\ \mathbf{0}, \text{اگر } w_{ij}x_i \leq 0 \end{cases}$$

مساله XOR و ناتوانی پرسپترون در حل آن

الگوریتم یادگیری پرسپترون وزن‌ها را تعدیل می‌کند تا این‌که، تمام نمونه‌های ورودی به درستی دسته‌بندی شوند. اگر ورودی‌ها به‌صورت خطی از یک‌دیگر قابل تفکیک و جداسازی نباشند، این الگوریتم یادگیری خاتمه‌پذیر نخواهد بود. این شبکه عصبی تک‌لایه در مسائل دنیای واقعی به دلیل اینکه، **الگوهای بین دسته‌ها لزوما تفکیک‌پذیر خطی نیستند**، قابل استفاده و کاربردی نخواهد بود. از همین‌رو به سراغ دسته‌ای از شبکه‌ها خواهیم رفت که قدرت بیشتری در حل مسائل دنیای واقعی دارد.

استفاده از یک لایه پنهان در ساختار شبکه‌ها گریزی است بر این محدودیت که در شبکه‌های تک‌لایه وجود دارد. برای درک بهتر از این موضوع شکل (آ) ۲ـ۳ را در نظر بگیرید؛ جایی که [۰،۰] و [۱،۱] به کلاس A تعلق دارد و [۰،۱] و [۱،۰] به کلاس B متعلق است. به‌راحتی قابل مشاهده است که هیچ خط مستقیمی وجود ندارد که بتوان

دو کلاس را به‌طور کامل از یک‌دیگر جدا کرد. از همین‌رو، یک دسته‌بند خطی همانند پرسپترون در این مورد عملکرد بسیار ضعیفی دارد.



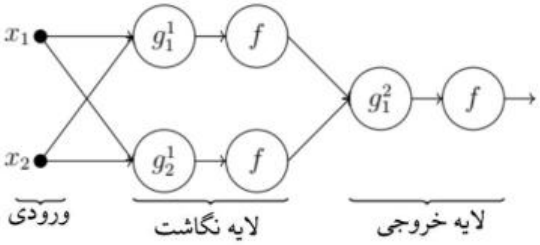
(آ) مساله دسته‌بندی XOR (ب) مساله XOR با نگاشت آن

شکل ۲ـ۳ مساله XOR در فضای اصلی و انتقال‌داده شده

با این حال، اگر به‌جای یکی از دو پرسپترون استفاده شود، چه اتفاقی خواهد افتاد؟ از شکل (آ) به راحتی می‌توان فهمید که فضای بین g_1 و g_2 به کلاس B اختصاص داده شود و فضای زیر g_1 یا بالاتر از g_2 باید به کلاس A اختصاص پیدا کند. حال این نگاشت را در نظر بگیرید:

$$x \rightarrow \begin{bmatrix} f(g_1(x)) \\ f(g_2(x)) \end{bmatrix} \quad , \quad f(x) = \begin{cases} 1, if\ net \geq \theta \\ 0, if\ net < \theta \end{cases}$$

اکنون $g_i(x)$ به شکلی است که در معادله ۲ ـ ۱ داده شده است. نتیجه این نگاشت در شکل (ب) ۲ ـ ۳ قابل مشاهده است. همان‌طور که به وضوح می‌بینیم، حال کلاس‌ها به‌صورت خطی در یک فضای جدید قابل تفکیک شدن می‌باشند. شکل ۲ ـ ٤ نمایی کلی از پرسپترون دوبعدی دولایه را ارائه می‌دهد. بسته به اینکه x به کدام کلاس تعلق داشته باشد، خروجی ۰ یا ۱ خواهد بود.



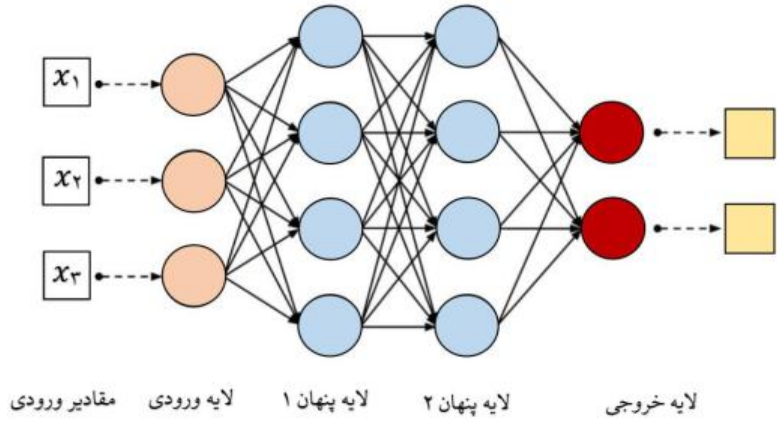
شکل ۲ ـ ٤ پرسپترون دو لایه

۲.۱.۲ شبکه‌های پیش‌خور عمیق

همان‌طور که بیان شد، یکی از محدودیت‌های اصلی شبکه‌های تک‌لایه تنها توانایی دسته‌بندی را داشتند، که داده‌ها جدایی‌پذیر خطی باشند، در غیر این صورت مسئله با این شبکه‌ها قابل حل نخواهد بود. در جهت حل این محدودیت می‌توان از لایه پنهان بین لایه ورودی و خروجی استفاده کرد. نمونه‌ای از این شبکه‌ها که اساس یادگیری عمیق هم هستند، شبکه‌های عصبی پرسپترون چندلایه بوده که همچنین از آن‌ها با عنوان شبکه‌های پیش‌خور عمیق نام برده می‌شود.

این شبکه‌ها از پرکابردترین‌ها شبکه‌ها در یادگیری عمیق به دلیل سازگاری آن با انواع مسائل است. چراکه برای ورودی آن هیچ محدودیتی وجود ندارد که داده‌ها تصویر، متن و یا ویدیو باشند. دلیل نام گذاری آن به پیش‌خور، نداشتن هیچگونه اتصال بازخوردی است، که از طریق آن بتوان خروجی‌های مدل دوباره به خود مدل بازگردنند. نرون‌های موجود در هر لایه از یک تابع فعال‌ساز مشترک استفاده می‌کنند، برای لایه ورودی، ورودی همان بردار خام از داده‌ها است.

در این شبکه‌ها با رفتن به هر لایه دیگر، جمع وزن‌دار مجموعه نرون‌های لایه قبلی محاسبه شده و پس از اعمال تابع فعال‌ساز غیرخطی به لایه دیگر منتقل می‌شوند تا در نهایت به لایه خروجی برسند. در شکل ۲ ـ ۵ شمایی از یک شبکه عصبی پیش‌خور عمیق تماما متصل قابل مشاهده است.



شکل ۲ ـ ۵ معماری یک شبکه عصبی پیش‌خور عمیق تماما متصل

با تکیه بر مفهوم لایه، معماری شبکه‌های عصبی پیش‌خور عمیق به صورت زیر خواهد بود:

- تنها یک لایه ورودی
- یک یا چند لایه پنهان، تماما متصل
- تنها یک لایه خروجی

اکنون به نحوه‌ی عمل و کاربرد هر یک از این لایه‌ها می‌پردازیم:

لایه ورودی: این لایه اولین لایه قابل مشاهده بوده و نحوه دریافت داده‌های ورودی (بردارها) به شبکه را تعیین می‌کند. در این لایه به‌طور معمول تعداد نرون‌ها نمایانگر تعداد ویژگی‌های در شبکه است. این لایه در شبکه‌های عصبی پیش‌خور در ارتباط به لایه‌های پنهان به‌صورت تماما متصل است. در سایر معماری‌های شبکه این امکان وجود دارد که، این ارتباط تماما متصل نباشد.

لایه پنهان: در شبکه‌های عصبی پیش‌خور یک یا چند لایه پنهان وجود دارد. مقادیر وزن‌ها در اتصالات بین لایه‌ها چگونگی رمزنگاری شبکه‌های عصبی در استخراج اطلاعات یادگرفته شده از داده‌های خام است. کلمه پنهان دلالت بر این موضوع دارد که، این لایه برای سیستم خارجی قابل رویت نیست. این لایه امکان مدل‌سازی توابع غیرخطی را می‌دهد. وجود یک لایه پنهان برای اکثر مسائل کافی خواهد بود؛ هر چه این تعداد لایه‌ها افزایش پیدا کند، تولید این شبکه به زمان بیشتری نیاز خواهد داشت، در

عوض می‌تواند مسائل پیچیده‌تری را حل کند. تعداد بهینه لایه‌ها و نرون‌ها در هر لایه عمدتا به مساله وابسته خواهد بود. انتخاب تعداد مناسب نرون‌ها در هر لایه‌ی پنهان بسیار مهم و ضروری است، چرا که در موفقیت فرآیند حل مسئله نقش کلیدی را دارند. این تعداد باید به‌صورت آزمایشی انتخاب شوند. انتخاب تعداد کم از نرون‌ها سبب کم‌برازش خواهد گردید. از سویی دیگر، تعداد زیاد نرون استفاده شده در لایه‌ی پنهان، منجر به افزایش غیرضروری زمان آموزش و امکان بیش‌برازش را بوجود خواهد آورد.

لایه خروجی: این لایه آخرین لایه‌ای است که قابل مشاهده بود و پیش‌بینی یا جواب مسئله در این لایه قابل صورت می‌گیرد. این خروجی بسته به نوع طراحی ما برای مسئله مور نظر می‌تواند، مجموعه‌ای از احتمالات باشد که مربوط به یک مسئله‌ای دسته‌بندی است، یا یک خروجی ارزیابی شده که مربوط به مسئله‌ای رگرسیون باشد. این خروجی با توجه به تابع فعال‌ساز تعیین می‌شود. تعداد نرون‌ها در این لایه متناسب با مسئله مورد نظر می‌باشد. برای مثال اگر مسئله دسته‌بندی دودویی باشد، لایه خروجی شامل دو نرون خواهد بود.

۲.۲ تابع فعال‌ساز

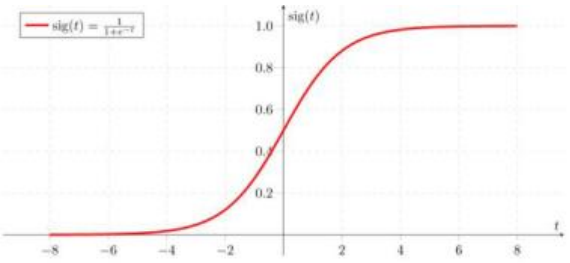
تابع فعال‌سازی نقش بسیار مهم و کلیدی در معماری یک مدل از شبکه عصبی را برخوردار است. از این تابع جهت انتشار خروجی هر لایه به لایه دیگر در پایان فرآیند محاسباتی در هر نرون استفاده می‌شود. در بیانی ساده، تابع فعال‌ساز تصمیم‌گیری در جهت این‌که کدام یک از نرون‌ها باید فعال و یا کدام یک غیرفعال شود را عهده‌دار است. به‌طور کلی، توابع فعال‌سازی غیرخطی در شبکه‌های عصبی بیشتر مورد استفاده قرار می‌گیرند.

تابع فعال‌سازی مورد استفاده در شبکه‌های پیش‌خور، برخلاف برخی شبکه‌های دیگر نمی‌تواند از هر تابعی باشد، بلکه باید ویژگی‌های خاصی را در خود به همراه داشته باشد. این تابع باید پیوسته، مشتق‌پذیر و یکنوا نزولی بوده، همچنین مشتق اول این تابع باید به‌راحتی قابل محاسبه باشد. در ادامه به بررسی چند نمونه از توابع فعال‌ساز پرکاربرد در یادگیری عمیق میپردازیم.

تابع سیگموید (sigmoid): تابع سیگموید که همچنین از آن به عنوان تابع منطقی نام برده می‌شود، یکی از کاربردی‌ترین توابع فعال‌سازی غیرخطی در شبکه‌های عصبی مصنوعی است. از این تابع جهت محاسبه احتمال برای مسائل دسته‌بندی دودویی در لایه خروجی استفاده می‌شود. این تابع خروجی احتمالاتی را به صورت مقادیر بین صفر تا یک برای هر دسته تولید می‌کند، و توسط معادله ۲ـ۲ قابل تعریف است:

$$\sigma(x) = sigmoid(x) = \frac{۱}{۱ + e^{-x}}$$ معادله ۲ – ۲

در شکل ۲ـ۶ شمایی از این تابع قابل مشاهده است.



شکل ۲ـ۶ تابع فعال‌سازی سیگموید

مزایا:

- غیرخطی است، بنابراین از آن در لایه‌های پنهان می‌توان استفاده کرد.
- در همه‌جا مشتق‌پذیر است.
- دامنه خروجی آن صفر و یک است، بنابراین می‌تواند برای مسائل دسته‌بندی استفاده شود.

معایب:

- گرادیان برای ورودی‌های دور از مبدا نزدیک به صفر است، بنابراین یادگیری مبتنی‌بر گرادیان برای نرون‌های اشباع (نرونی که به حداکثر یا حداقل مقدار خود رسیده باشد) شده با استفاده از سیگموید بسیار کند است.
- وقتی به عنوان تابع فعال‌سازی در لایه آخر برای مسائل دسته‌بندی استفاده می‌شود، جمع همه کلاس‌ها لزوما یک نمی‌باشد.

- صفر مرکز نیست (تابع صفر مرکز تابعی است که در برخی مواقع خروجی آن بیشتر از صفر و کمتر از صفر باشد). مقدار این تابع همیشه بین صفر تا یک است. بنابراین میانگین آن نمی‌تواند صفر باشد و همیشه مقداری بیشتر از صفر خواهد بود.
- نیاز به محاسبه نمایی دارد، بنابراین سرعت همگرایی کندی دارد.
- مشکل محو گرادیان دارد. برای مقادیر بسیار کوچک یا بسیار بزرگ x، تقریبا تغییری در پیش‌بینی وجود ندارد.

تابع تانژانت هذلولوی (Tanh): مزیت این تابع این است که می‌تواند با اعداد منفی راحت‌تر برخورد کند. خروجی این تابع مقداری بین ۱ تا ۱ــ می‌باشد، و به صورت معادله ۲ــ۳ قابل نمایش است:

$$\sigma(x) = tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$ معادله ۲ − ۳

مزایا:

- خروجی نرون را تا محدوده بین ۱ــ تا ۱ نرمال می‌کند.
- برخلاف سیگموید یک تابع صفرمحور[1] است تا بهینه‌سازی تابع زیان آسان‌تر شود.

معایب:

- مشکل محو گرادیان دارد.
- از لحاظ محاسباتی هزینه زیادی دارد.
- مشکل اشباع را دارد.

تابع یکسوساز خطی (Rectified Linear Unit): تابع یکسوساز خطی، که در لایه پنهان مورد استفاده قرار می‌گیرد، امروزه یکی از پرکاربردترین توابع در یادگیری عمیق است. این تابع در معادله ۲ــ۴ قابل نمایش است:

$$ReLU(x) = \max(0, x)$$ معادله ۲ − ۴

[1] zero-centered function

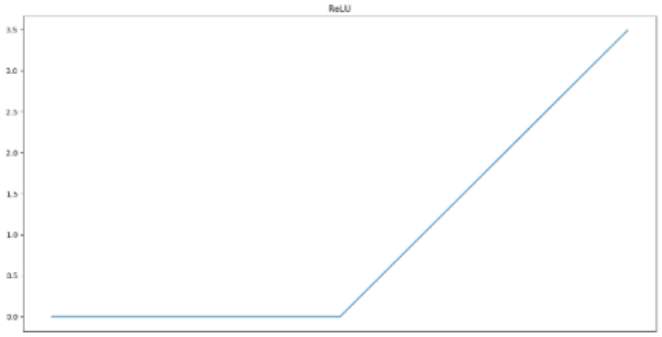
با تنظیم مقادیر ورودی منفی به صفر، ورودی را به مقدار بزرگتر یا مساوی از صفر تبدیل می‌کند. در بیانی دیگر، برای ورودی‌های منفی مقدار صفر و برای ورودی‌های مثبت حد بالا ندارد. شمایی از این تابع در شکل ۲ــ۷ قابل مشاهده است.

مزایا:

- از نظر محاسباتی خیلی کارآمد بوده و شبکه اجازه می‌دهد خیلی سریع همگرا می‌شود. نسبت به سیگموید و تانژانت هذلولوی محاسبات نمایی ندارد.
- از مشکل محو گرادیان جلوگیری می‌کند.

معایب:

- مشکل مرگ ReLU[۱] دارد. وقتی ورودی‌ها به صفر نزدیک می‌شوند یا صفر هستند، گرادیان تابع صفر می‌شود. بنابراین نمی‌تواند برای یادگیری از الگوریتم پس‌انتشار استفاده کند. مشکل مرگ ReLU دائمی نیست. اگر داده‌های آموزشی جدیدی اضافه شود، ممکن است این نرون‌ها دوباره فعال شوند.
- از مشکل انفجار گرادیان جلوگیری نمی‌کند.



شکل ۲ــ۷ تابع فعال‌سازی یکسوساز خطی

تابع یکسوساز خطی رخنه‌دار (LeakyReLU): این تابع کاملا مشابه با تابع فعال‌سازی یکسوساز خطی است. تفاوت آن معرفی یک پارامتر α است که اجازه می‌دهد گرادیان‌های کوچک در صورت عدم فعال‌سازی، فعال شوند. از همین رو، این تابع

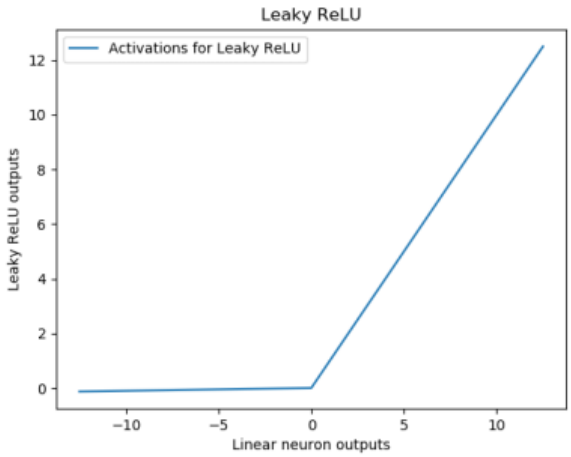
[۱] Dying ReLU

فعال‌سازی مشکل مرگ نرون‌ها در طول فرآیند آموزش را بین می‌برد. این تابع در معادله ۲ــ۵ قابل نمایش است:

$$LeakyReLU(x) = \begin{cases} x & \text{اگر : صفر} \geq x \\ \alpha x & \text{اگر : صفر} > x \end{cases}$$

معادله ۲ – ۵

شمایی از این تابع در شکل ۲ــ۸ قابل مشاهده است.



شکل ۲ــ۸ تابع فعال‌سازی یکسوساز خطی رخنه‌دار

مزایا:

- مشکل مرگ ReLU را رفع می‌کند. چرا که هنگام محاسبه مشتق اجازه یک گرادیان کوچک را می‌دهد.
- از لحاظ محاسباتی سریع‌تر است.

معایب:

- از مشکل انفجار گرادیان جلوگیری نمی‌کند.
- شبکه عصبی نمی‌تواند مقدار پارامتر α را یاد بگیرد.

تابع بیشینه‌هموار (Softmax): این تابع که از آن در لایه خروجی استفاده می‌شود، تعمیم‌یافته‌ای از تابع فعال‌ساز سیگموید بوده، و برای مشکلات مربوط به دسته‌بندی از آن استفاده می‌شود. این تابع این امکان را فراهم می‌سازد که یک پیش‌بینی احتمالاتی را برای مسئله دسته‌بندی بیش از دو دسته انجام دهد، و به‌صورت معادله ۲ــ۶ تعریف می‌شود:

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$ معادله ۲ — ۶

- **در مسائل دسته‌بندی دودویی، از تابع سیگموید در لایه خروجی استفاده می‌شود.**
- **در مسائل دسته‌بندی چندبرچسبی، در لایه خروجی از تابع سیگموید استفاده می‌شود.**
- **در مسائل دسته‌بندی چندکلاسه، از تابع بیشینه‌هموار در لایه خروجی استفاده می‌شود.**
- **تابع یکسوساز خطی در تمامی لایه‌های پنهان می‌توان استفاده کرد.**

مقایسه توابع فعال‌سازی

در جدول ۲ ــ ۱ مقایسه توابع فعال‌سازی را براساس معیارهای دامنه خروجی، صفرمحور بودن یا نه، مشکل اشباع، محاسبه و مشکل محو گرادیان خلاصه شده‌اند.

جدول ۲ ــ ۱ مقایسه توابع فعال‌سازی

تابع	دامنه	صفر ــ محور	اشباع	محو گرادیان	محاسبه
sigmoid	$[۰,۱]$	خیر	برای مقادیر مثبت و منفی	بله	کند و نمایی
Tanh	$[-۱,۱]$	بله	برای مقادیر مثبت و منفی	بله	کند و نمایی
ReLU	$[۰,+\infty]$	خیر	برای مقادیر منفی	بهتر از سیگموید و تانژانت هذلولوی	سریع
LeakyReLU	$[-\infty,+\infty]$	بله	ندارد	خیر	سریع

۳.۲ تابع زیان

تابع زیان که همچنین از آن به عنوان تابع هزینه نامبرده می‌شود، تعیین می‌کند که شبکه عصبی آموزش دیده شده تا چه اندازه نزدیک به معیار ایده‌آل ما بوده و یکی از جنبه‌های مهم آموزش پس از داده‌های آموزشی خوب و معماری مناسب در شبکه‌های عصبی است. انتخاب تابع زیان مناسب نقش مهمی در سرعت همگرایی شبکه دارد.

در الگوریتم‌های بانظارت، قصد داریم خطا را برای هر نمونه آموزشی در طول فرآیند یادگیری به حداقل برسانیم. این کار توسط الگوریتم‌های بهینه‌سازی انجام می‌گیرد. تابع زیان اندازه‌گیری این میزان خطای مدل در را انجام می‌دهد تا توانایی مدل را بسنجد. این مقدار اندازه‌گیری شده نمایانگر این است که چقدر شبکه عصبی به معیار ایده‌آل نزدیک است. با توجه با این مقدار بدست آمده شبکه می‌تواند با تکرار بروزرسانی وزن‌های خود، سبب به حداقل رساندن این میزان خطا شود.

به‌طور کلی، مفهوم اصلی تابع زیان، **اندازه‌گیری میزان خطا بین مقادیر هدف تخمین‌زده شده و مقدار واقعی مسئله می‌باشد.** با فرض این‌که y مقدار واقعی مسئله و $\tilde{y}$ خروجی تخمین زده باشد، جهت داشتن بهترین مدل باید خروجی تابع هزینه که به‌صورت $L(y, \tilde{y}) = y - \tilde{y}$ می‌باشد، به حداقل برسد. انتخاب تابع زیان به نوع مساله بستگی دارد و برای مسائل مختلف دسته‌بندی و رگرسیون این تابع زیان متفاوت خواهد بود. در یک مساله دسته‌بندی، قصد داریم تا یک توزیع احتمالاتی برای مجموعه کلاس‌ها پیش‌بینی کنیم. حال آن‌که، در مسائل رگرسیون، قصد داریم یک مقدار خاص را بیابیم.

۳.۱.۲ توابع زیان مربوط به دسته‌بندی

در این بخش توابع زیان مربوط به دسته‌بندی را پوشش خواهیم داد.

زیان آنتروپی متقاطع

آنتروپی متقاطع روشی ریاضی است که در مسائل گسسته مانند دسته‌بندی استفاده می‌شود. این تابع فاصله بین دو احتمال را محاسبه می‌کند و به صورت زیر تعریف می‌شود:

$$Cross\ Entropy(y, \hat{y}) = \frac{۱}{n}\sum_{i=۱}^{n} y_i \log(\hat{y})$$

برای دسته‌بندی‌های دودویی از آنتروپی متقاطع دودویی استفاده می‌شود که به صورت زیر تعریف می‌شود:

$$Binary\ Cross\ Entropy(y, \hat{y}) = -\frac{۱}{n}\sum_{i=۱}^{n} (y_i \log(\hat{y}) + (۱ - y_i)(۱ - \log(\hat{y})))$$

واگرایی کولبک‌ـ لیبر[۱]

این تابع مشابه با آنتروپی متقاطع در مسائل دسته‌بندی استفاده می‌شود و معیاری برای اندازه‌گیری تفاوت بین توزیع احتمال از توزیع احتمال پایه است. معادله آن به‌صورت زیر تعریف می‌شود:

$$KL(y, \hat{y}) = \sum_{i=1}^{n} y_i \log\left(\frac{y_i}{\hat{y}_i}\right)$$

۴.۱.۲ توابع زیان مربوط به رگرسیون

در این بخش توابع زیان مربوط به رگرسیون را پوشش خواهیم داد.

میانگین خطای مربعات[۲] **(MSE)**

از معروف‌ترین توابع زیان در رگرسیون می‌باشد، و میانگین اختلاف مربعات بین مقادیر واقعی و پیش‌بینی شده را توسط معادله زیر محاسبه می‌کند:

$$MSE(y, \hat{y}) = \frac{۱}{n}\sum_{i=۱}^{n} (y_i - \hat{y}_i)^۲$$

[۱] Kullback–Leibler divergence
[۲] Mean squared error loss

میانگین خطای قدرمطلق[۱] (MAE)

از این تابع برای مسائل مربوط به رگرسیون استفاده می‌شود. این تابع میانگین اختلاف قدرمطلق بین مقادیر واقعی و پیش‌بینی شده را توسط معادله زیر محاسبه می‌کند:

$$MAE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

۲.۲ رویکردهای وزن‌دهی اولیه

اولین مرحله در هنگام ساخت یک شبکه عصبی در جهت دستیابی به بهترین نتایج، مقداردهی اولیه به پارامترها می‌باشد. اگر این کار به‌درستی صورت پذیرد، بهینه‌سازی در کمترین زمان حاصل خواهد شد. در غیر این صورت همگرایی با استفاده از گرادیان کاهشی غیر ممکن خواهد شد. یکی از این پارامترها جهت مقداردهی اولیه، دادن مقادیر ابتدایی به وزن‌ها می‌باشد. وزن‌هایی که در طول آموزش شبکه استفاده می‌شوند، باید مقادیر ابتدایی برای شروع آموزش را داشته باشند. این مقداردهی اولیه وزن‌ها تاثیر بسیار زیادی در سرعت همگرایی و دقت شبکه دارند. مقادیر خودسرانه سبب ایجاد همگرایی کند یا از بین رفتن فرآیند یادگیری می‌شود. بنابراین انتخاب یک روش مناسب برای آموزش در شبکه‌ها عمیق فرآیندی مهم خواهد بود. روش‌های گوناگونی برای این کار وجود دارد که در ادامه به بررسی این روش‌ها می‌پردازیم.

۱.۲.۲ مقداردهی اولیه همه وزن‌ها با صفر

یکی از روش‌های ساده بوده و به این صورت عمل می‌کند که، در ابتدا به همه وزن‌ها مقادیر صفر بدهیم و در طول فرآیند آموزش وزن‌ها به‌هنگام شوند. این ایده گرچه مفید به نظر می‌رسد اما، مشکلی در این روش وجود دارد. اگر تمام وزن‌ها با صفر مقداردهی شوند، مشتق همه آن‌ها با توجه به تابع زیان یکسان خواهد بود. بنابراین، تمام وزن‌ها پس از تکرارهای پی در پی ارزش یکسانی دارند.

[۱] Mean absolute error loss

۲.۲.۲ مقداردهی اولیه تصادفی

در این روش، وزن‌دهی اولیه با مقادیر تصادفی شروع می‌شود تا هر وزن مقداری متفاوت داشته باشد. این روش هم با دو مشکل محو گردایان و انفجار گرادیان به همراه خواهد بود. اگر مقادیر وزن‌ها کوچک باشند، گرادیان به مرور کوچک و کوچک‌تر و سرانجام ناپدید خواهد شد. این عمل سبب همگرایی کند و یا بدترین حالت از بین رفتن فرآیند یادگیری را به همراه خواهد داشت. در مقابل مشکل محو گرادیان در این روش، امکان بوجود آمدن انفجار گرادیان نیز خواهد بود. اگر مقادیر وزن‌ها بسیار بزرگ باشند، گرادیان آن‌ها بزرگ خواهد شد و سبب به‌هنگام‌سازی بزرگ در وزن‌ها شبکه می‌شود. نتیجه این عمل ناپایداری و یا در بدترین حالت سرریز و خروجی را به‌همراه نخواهد داشت.

۳.۲.۲ یادگیری انتقالی

روشی دیگر استفاده از مقدار وزن‌های آموزش دیده از سایر مدل‌ها به مدل هدف است. به این ترتیب در این روش در شروع فرآیند آموزش مقداردهی اولیه وزن‌ها صورت نمی‌گیرد، بلکه وزن‌های مدل دیگر را می‌آموزد.

۴.۲.۲ مقداردهی اولیه یکنواخت گلوروت[۱]

مقداردهی اولیه یکنواخت گلوروت که همچنین با نام **مقداردهی اولیه خاویر**[۲] هم شناخته می‌شود، وزن‌های لایه L را براساس توزیع با میانگین صفر و انحراف معیار خاص در شبکه، به صورت یکنواخت در بازه:

$$\left[-\sqrt{\frac{۶}{(n_{l-۱}+n_l)}}, \sqrt{\frac{۶}{(n_{l-۱}+n_l)}}\right]$$

مقداردهی می‌کند. که در این معادله $n_{l-۱}$ و n_l تعداد نرون‌ها در لایه $L-۱$ و L می‌باشد.

[۱] Glorot Uniform Initialization
[۲] Xavier Initialization

۵.۲.۲ مقداردهی اولیه هنجار هی[۱]

در روش هی، وزن‌ها با در نظر گرفتن اندازه نرون‌های لایه‌ی قبلی در جهت دستیابی سریع‌تر به بهینه سراسری و به حداقل رساندن تابع زیان مقداردهی اولیه می‌شوند. در این روش وزن‌ها در لایه L با مقادیر تصادفی و بر اساس میانگین صفر و انحراف معیار $\sqrt{\frac{۲}{n_{l-۱}}}$ که $n_{l-۱}$ تعداد نرون‌ها در لایه $L-۱$ می‌باشد، مقداردهی اولیه می‌شوند.

۳.۲ بهینه‌سازها و به‌هنگام‌سازی وزن‌ها

بهینه‌سازها الگوریتمی‌هایی هستند که از طریق به‌هنگام‌سازی وزن‌ها در شبکه سعی در به حداقل رساندن تابع زیان دارند، یعنی همان هدف اصلی ما از آموزش شبکه‌ها برای مسئله مورد نظر است که سعی داریم، وزن‌های شبکه به‌گونه‌ای تنظیم شوند تا شبکه توانایی یادگیری را بدست آورد. انتخاب الگوریتم بهینه‌سازی مناسب نقش مهمی در سرعت همگرایی شبکه دارد. روش‌های مختلفی برای بهینه‌سازی وجود دارد.

گرادیان کاهشی یکی از محبوب‌ترین و متداول‌ترین این الگوریتم‌ها در شبکه‌های عصبی هستند. نمونه این دیگر این الگوریتم‌ها، روش نیوتن می‌باشد. این روش با استفاده از مشتق مرتبه دوم با یافتن ریشه‌های یک تابع در بهبود بهینه‌سازی نقش دارد. روش نیوتن در مقایسه با روش‌های مبتنی‌بر مشتق مرتبه اول، پیچیدگی محاسباتی را به میزان قابل توجهی افزایش می‌دهد. به همین دلیل، استفاده از روش‌های گرادیان کاهشی در فرآیند آموزش شبکه عصبی بیشتر مورد استفاده قرار می‌گیرد. حالت ساده این روش گرادیان خطا را برای تمام نمونه‌های آموزشی به‌صورت زیر محاسبه می‌کند:

الگوریتم گرادیان کاهشی

با فرض اینکه : **تابع زیان :** $L(w)$ **, نرخ یادگیری :** η **, بردار وزن :** w

۱ ــ مقداردهی اولیه وزن‌ها

---

[۱] He Normal Initialization

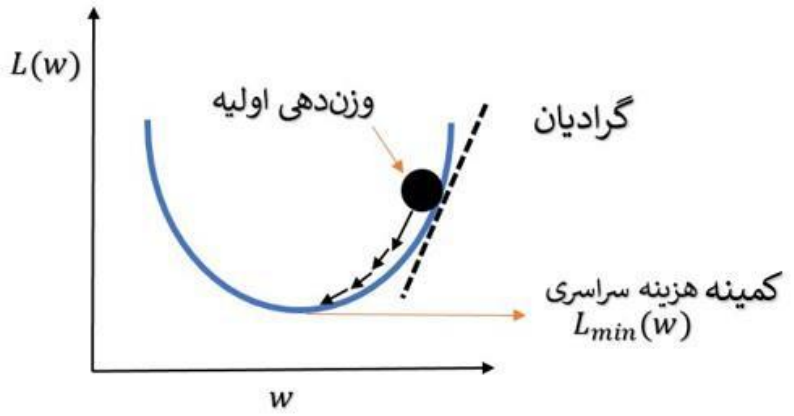
۲ـ تا زمان همگرایی حلقه را انجام بده:

گرادیان را محاسبه کن، $\frac{\partial L(w)}{\partial w}$

وزن‌ها را به‌هنگام کن، $w \leftarrow w - \eta \frac{\partial L(w)}{\partial w}$

۳ـ وزن‌ها را برگردان

این الگوریتم را مطابق شکل ۲ـ۹ می‌توان همانند یک کوهنورد (ضریب وزن) تصور کرد، که قصد دارد از کوه (تابع هزینه) به سمت دره (کمینه هزینه) و در هر مرحله با یک تندی شیب (گرادیان) با طول قدم‌ها (نرخ یادگیری) پایین بیاید.



شکل ۲ـ۹ الگوریتم گرادیان کاهشی

الگوریتم گرادیان کاهشی، یک روش بهینه‌سازی مبتنی‌بر تکرار بوده و با تغییر در وزن‌های داخلی شبکه و بروز کردن تدریجی آن‌ها در جهت به حداقل رساندن تابع زیان در تلاش است. اندازه گام در هر تکرار الگوریتم را نرخ یادگیری تعیین می‌کند. فرآیند تکرار تا جایی که در تابع زیان تغییری مشاهده نشود (به آن همگرایی گفته می‌شود)، انجام می‌گیرد.

در عمل وقتی تعداد نمونه‌های آموزشی زیاد باشد، استفاده از الگوریتم گرادیان کاهشی زمان زیادی را به همراه خواهد داشت. چرا که باید در هر تکرار الگوریتم برای تمام نمونه‌ها انجام شود. از همین رو استفاده از الگوریتم گرادیان کاهشی تصادفی به دلیل این‌که در هر تکرار الگوریتم تنها، دسته‌ای از نمونه‌ها را به‌هنگام می‌کند، مفیدتر خواهد بود. سه روش کلی برای استفاده از گرادیان کاهشی وجود دارد: گرادیان کاهشی با یک نمونه، گرادیان کاهشی کامل و گرادیان کاهشی ریزدسته‌ای. در ادامه به بررسی انواع مختلفی از الگوریتم‌های بهینه‌سازی گرادیان کاهشی مورد استفاده در شبکه‌های عمیق می‌پردازیم.

۱.۳.۲ گردایان کاهشی تصادفی (SGD)[۱] و ریز-دسته‌ای

فرض کنید در مجموعه داده‌های آموزشی میلیون‌ها نمونه وجود داشته باشد، در چنین وضعیتی استفاده از روش گرادیان کاهشی، در هر تکرار باید گرادیان تمام این میلیون نمونه محاسبه شود، که این عمل پردازش و محاسبات زیادی را به‌همراه خواهد داشت. این مشکل را الگوریتم گرادیان کاهش تصادفی حل کرده است. گرادیان کاهشی تصادفی، یک روش تقریب تصادفی از گرادیان کاهشی می‌باشد و در آن برخلاف گرادیان کاهشی، جهت بهینه‌سازی تابع هدف از همه‌ی نمونه‌های آموزشی استفاده نمی‌شود، بلکه با ورود هر نمونه که به‌صورت تصادفی برای بهینه‌سازی در هر دوره انتخاب می‌شود، بروزرسانی اعمال و وزن‌های جدید بدست می‌آیند. معادله آن به‌صورت زیر است:

$$\theta = \theta - \eta.\nabla_\theta J(\theta, x^i, y^i)$$

که در این معادله، θ پارمترهای مدل، J تابع زیان، x^i ورودی نمونه آموزشی و y^i برچسب آن می‌باشد. در این روش به دلیل پاسخ با هر ورود نمونه، می‌تواند نتایج ناپایداری را در شبکه به همراه داشته باشد. همچنین در این روش مشکل گیر افتادن در کمینه محلی نیز وجود دارد.

در روش گرادیان کاهشی کامل، شبکه با محاسبه خطا برای تمام نمونه‌های آموزشی بروزرسانی وزن‌ها را انجام می‌دهد. از همین رو، مشکل کمینه محلی را به‌همراه ندارد.

[۱] Stochastic gradient descent

در مقابل زمان آموزش طولانی‌تری در پی خواهد داشت. در خلا بین این دو روش، گرادیان کاهشی ریز_دسته‌ای استفاده می‌شود. در این روش اصطلاحی به نام دسته[۱] وجود دارد که، تعداد نمونه‌های مورد استفاده در هر دوره تکرار برای محاسبه گرادیان را نشان می‌دهد. گرادیان کاهشی ریز_دسته، کل مجموعه آموزشی را به ریز_دسته‌هایی به تعداد n تقسیم کرده، و براساس این ریز_دسته‌ها به‌هنگام‌سازی پارامترها را انجام می‌دهد. معادله آن به‌صورت زیر است:

$$\theta = \theta - \eta.\nabla_{\theta}J\left(\theta, x^{i:i+n}, y^{i:i+n}\right)$$

اندازه معمول ریز_دسته‌ها از ۵۰ تا ۲۵۶ متغییر بوده و باید معقول انتخاب شود:

- اندازه دسته‌های بزرگ، گرادیان دقیق‌تری را ارائه می‌دهند اما به حاظفه بیشتری نیاز دارد.
- اندازه دسته‌های کوچک، برای حفظ ثبات به دلیل واریانس زیاد در برآور گرادیان نیاز به نرخ یادگیری کوچکی دارد. انتخاب نرخ یادگیری کوچک به نوبه خود سبب کاهش فرآیند آموزش می‌شود.

استفاده از گرادیان کاهشی ریز_دسته، به دلیل استفاده و ترکیب دو روش گرادیان کاهشی کامل و گرادیان کاهشی تصادفی نسبت به نویز مقاوم‌تر بوده و واریانس کم‌تری داشته، در نتیجه همگرایی پایدارتری دارد. از همین رو، معمولا از این روش بهینه‌سازی در یادگیری عمیق استفاده می‌شود. **با این وجود، این روش‌ها دارای یک نقطه ضعف مهم هستند: انتخاب نرخ یادگیری**. انتخاب درست نرخ یادگیری همیشه کار آسانی نیست. جدای از این، انتخاب نرخ یادگیری یکسان در تمام مراحل آموزشی برای همه‌ی پارامترها عملی بهینه نخواهد بود. از همین رو، الگوریتم‌های متفاوتی برای حل این مشکل در جهت تطبیق نرخ یادگیری در مراحل مختلف الگوریتم ارائه تا همگرایی سریع‌تر شبکه را به‌وجود آورند. آدام نمونه‌ای بسیار کاربردی از الگوریتم‌های با نرخ یادگیری تطبیقی است.

[۱] batch

۲.۳.۲ آداگراد[۱]

آداگراد یک الگوریتم بهینه‌سازی مبتنی‌بر گرادیان می‌باشد. در روش گرادیان کاهشی تصادفی، ما به‌هنگام‌سازی هر پارامتر w_i را با استفاده از یک نرخ یادگیری مشترک انجام می‌دادیم. این عمل در بیشتر اوقات سبب بروز مشکلاتی می‌شود. آداگراد یکی از الگوریتم‌هایی است که از نرخ یادگیری تطبیقی استفاده می‌کند. در این الگوریتم، نرخ یادگیری هر یک از پارامترهای مدل از طریق تغییر مقیاس آن‌ها به‌صورت نسبت عکس با ریشه‌ی دوم مجموع همه‌ی مقادیر قبلی مربع آن‌ها، به صورت زیر تطبیق پیدا می‌کند:

$$w_{t+1,i} = w_{t,i} - \frac{\eta}{\sqrt{G_t}} \odot \nabla_{t,i}$$

که در این معادله $\nabla_{t,i}$ گرادیان تابع هزینه، G_t شامل مجموع مربع گرادیان‌های گذشته است.

مهم‌ترین مزیت استفاده از آداگراد این است که، نرخ یادگیری به‌طور خودکار تنظیم شده و نیازی به تنظیم دستی آن نیست. با این همه، مجموع مخرج موجود به تدریج سبب فروپاشی نرخ یادگیری می‌شوند. این میزان نرخ یادگیری در حال کاهش می‌تواند یادگیری را کند و یا حتی سبب آن شود که یادگیری را به‌طور کامل متوقف کند. الگوریتم‌های بعدی در جهت رفع این نقص ارائه شده‌اند.

۳.۳.۲ آدادلتا[۲]

آدادلتا گسترش‌یافته از الگوریتم آداگراد می‌باشد که مشکل کاهش نرخ یادگیری را برطرف می‌کند. این الگوریتم به جای مجموع مربع همه‌ی گرادیان‌های گذشته، تعداد گرادیان-های گذشته را محدود به اندازه x می‌کند و سپس میانگین این گرادیان‌های گذشته را برای بهره‌وری ذخیره می‌کند. میانگین مقدار ∇_t^2 در زمان t تنها وابسته به میانگن‌های گذشته و گرادیان کنونی دارد. به‌هنگام‌سازی پارامترها به صورت زیر انجام می‌شود:

---

[۱] Adagrad
[۲] Adadelta

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\text{میانگین}(\nabla_t^2)}} \odot \nabla_t$$

از آنجایی که مخرج فقط جذر متوسط مربع[1] گرادیان می‌باشد، می‌توان آن را به‌صورت زیر بازنویسی کرد:

$$w_{t+1} = w_t - \frac{\eta}{\text{جذر متوسط مربع}(\nabla_t)} \odot \nabla_t$$

۴.۳.۲ آرآم‌اس‌پراب[2]

آرآم‌اس‌پراب همانند آدادلتا با تغییر در الگوریتم آداگراد، به رفع مشکل کاهش نرخ یادگیری می‌پردازد. این نسخه اصلاح‌شده از الگوریتم آداگراد، از یک متوسط کاهنده نمایی برای حذف سوابق در گذشته‌های دور استفاده می‌کند. آرآم‌اس‌پراب در حقیقت با اولین بردار بدست آمده برای الگوریتم آدادلتا یکسان می‌باشد:

$$\text{میانگین}(\nabla_t^2) = \text{میانگین}۰.۹(\nabla_{t-1}^2) + ۰.۱\nabla_t^2$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\text{میانگین}(\nabla_t^2)}} \odot \nabla_t$$

به‌طور تجربی نشان داده شده است که، آرآم‌اس‌پراب الگوریتمی کارا و موثر در شبکه‌های عصبی عمیق می‌باشد.

۵.۳.۲ برآورد تکانه تطبیقی (ADAM)[3]

برآورد تکانه تطبیقی یا به اختصار آدام، روش دیگری جهت محاسبه نرخ یادگیری تطبیقی برای هر پارامتر می‌باشد. این الگوریتم از مزایای الگوریتم‌های آداگراد و آرآم-اس‌پراب استفاده می‌کند و میانگین فروپاشی نمایی از گرادیان‌های گذشته را در v_t

[1] root mean square
[2] RMSprop
[3] Adaptive Moment Estimation

ذخیره می‌کند. علاوه بر این آدام، میانگین تکانه‌ای دوم گرادیان را در m_t ذخیره می‌کند. m_t و v_t به ترتیب مقادیر میانگین و واریانس غیرمتمرکز هستند:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

آدام میانگین‌های تحرک نمایی گرادیان و گرادیان مربع را توسط معادلات زیر کنترل می‌کند:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

که ابرپامترهایی با مقادیر $\beta_2, \beta_1 \in [0,1]$ است. معادله نهایی به‌هنگام‌سازی به صورت زیر می‌باشد:

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t}} \odot \hat{m}_t$$

الگوریتم آدام از سایر روش‌های تطبیقی بهتر عمل می‌کند و خیلی سریع همگرا می‌شود. همچنین بر سایر مشکلاتی که الگوریتم‌های بهینه‌سازی همانند: فروپاشی نرخ یادگیری، واریانس بالا در به‌هنگام‌سازی و همگرایی آهسته غلبه کرده‌اند، غلبه می‌کند.

۴.۲ الگوریتم پس‌انتشار خطا

همان‌طور که در فصل ۱ بیان شد، در سال‌های اولیه پیدایش شبکه‌های عصبی آموزش شبکه‌های چندلایه ناشناخته باقی ماند. علت آن هم پافشاری مینسکی و پاپرت بر عدم توانایی آموزش شبکه‌های چندلایه بود. همین امر سبب شد، مقالات شبکه‌های عصبی تا دهه ۱۹۸۰ روبه نابودی رود. خوشبختانه اولین موفقیت در این رابطه توسط روملهارت و همکارانش در قالب ارائه الگوریتم پس‌انتشار، سبب احیا و علاقه مجدد پژوهشگران به شبکه‌های عصبی گردید. با این حال چند چالش محاسباتی، همگرایی و

بیش‌برازش در این شبکه‌ها یافت شد، تا دوباره تحقیقات در شبکه‌های عصبی یک سقوط را تجربه کند.

پیشرفت‌های اخیر، جان دوباره‌ای به شبکه‌های عصبی بخشید، تا دوباره این شبکه‌ها محبوبیت زیادی پیدا کنند. این پیشرفت‌ها محدود به الگوریتم نبودند، افزایش دسترسی به داده‌ها و قدرت محاسباتی سخت‌افزارها، نقش اساسی را در این بین داشتند. با این همه، ما در ادامه این بخش به بررسی الگوریتم پس‌انتشار همراه گرادیان کاهشی می‌پردازیم.

یادگیری در شبکه‌های عصبی براساس نمونه‌های آموزشی وارد شده به الگوریتم و تغییرات در وزن‌ها صورت می‌گیرد. الگوریتمی که برای یادگیری در وزن‌ها مورد استفاده قرار می‌گیرد، **الگوریتم پس‌انتشار خطا** نام دارد. این الگوریتم از قاعده زنجیره‌ای حساب دیفرانسیل استفاده می‌کند و گرادیان خطا را در مسیرهای مختلف از یک گره تا خروجی محاسبه می‌کند. این الگوریتم در رده روش‌های یادگیری بانظارت قرار می‌گیرد و از دو فاز اصلی به نام فاز جلورو و فاز عقبگرد تشکیل می‌شود. فاز جلورو جهت محاسبه مقادیر خروجی و مشتقات محلی در گره‌های مختلف، و فاز عقبگرد جهت جمع‌آوری حاصل این مقادیر محلی در تمام مسیرها از گره تا خروجی مورد نیاز هستند.

فاز جلورو: در این مرحله ورودی‌ها به عنوان نمونه‌های آموزشی به شبکه عصبی تغذیه می‌شوند. با استفاده از مجموعه وزن‌های فعلی، خروجی پیش‌بینی شده نهایی را می‌توان با نمونه آموزشی مقایسه کرد.

فاز عقبگرد: هدف اصلی مرحله عقبگرد، به‌هنگام‌سازی وزن‌ها است. از آنجایی که این مرحله از گره خروجی شروع می‌شود، این مرحله به عنوان مرحله عقبگرد نامیده می‌شود.

به‌طور خلاصه این الگوریتم به این صورت عمل می‌کند که در مرحله اول که فاز جلورو نامیده می‌شود، با اعدادی کوچیک تصادفی وزن‌دهی اولیه انجام می‌گیرد، سپس جمع وزن‌دار هر لایه توسط تابع فعال‌ساز محاسبه و به لایه دیگر منتقل می‌شود تا در نهایت در لایه خروجی، یک پیش‌بینی صورت گیرد. میزان خطای این خروجی پیش‌بینی شده با خروجی هدف مقایسه می‌شود و این احتمال بیشتر است که خطا در دور اول زیاد باشد.

مرحله دوم که فاز عقبگرد نامیده می‌شود، برمبنای میزان خطایی که از فاز جلورو بدست آمده، وزن‌ها به‌هنگام سازی می‌شوند تا میزان خطای پیش‌بینی شده با مقدار هدف کم شود. این عمل را در چندین دور انجام داده تا این میزان خطا برای تمام نمونه‌های آموزشی با مقادیر پیش‌بینی شده به حداقل رسد. در ادامه ابتدا قانون زنجیره‌ای راشرح می‌دهیم و سپس با مثالی به تشریح مرحله به مرحله این الگوریتم می‌پردازیم.

قانون زنجیره‌ای

استفاده از مدل های گرافیکی می‌تواند درک بهتری از قانون زنجیره ای را ایجاد کند. اساساً از مدل‌های گرافیکی برای توصیف روابط متغیرها و توابع در مدل‌های احتمالی استفاده می‌شود. فرض کنید که یک تابع به‌صورت $f = f(x) = f\big(x(y)\big)$ داشته باشیم و روابط توابع به عنوان مدل گرافیکی به‌صورت زیر باشد:

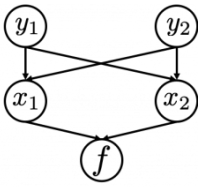
$$f = f(x) = f\big(x(y)\big)$$

$$\frac{df}{dy} = \frac{df}{dx}\frac{dx}{dy}$$

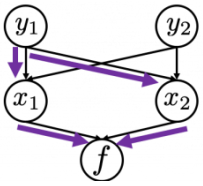
متغیرها نوعی تابع هستند، بنابراین باید تصور کنید که هر گره در مدل‌های گرافیکی یک تابع را نشان می‌دهد. پیکان‌ها در سمت راست تصویر بالا، نشان می‌دهند که چگونه اطلاعات در مشتق منتشر می‌شوند.

حال اگر یک تابع f داشته باشیم که دارای دو واریانس x_1 و x_2 و هر دو واریانس نیز دارای دو واریانس y_1 و y_2 باشند. وقتی از f نسبت به y_1 و y_2 مشتق جزئی بگیریم، معادله کمی مشکل‌تر می‌شود. حال بیاید تصور کنیم می‌خواهیم $\frac{\partial f}{\partial y_1}$ را محاسبه کنیم. واریانس y_1 از طریق x_1 و x_2 به f منتقل می‌شود. در این حالت مشتق جزئی دارای دو عبارت به‌صورت زیر خواهد بود.

$$f = f(x_1, x_2) = f\big(x_1(y_1, y_2), x_2(y_1, y_2)\big)$$

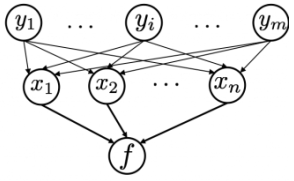


$$\frac{\partial f}{\partial y_1} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial y_1} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial y_1}$$

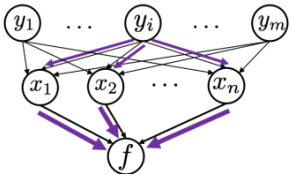


در قانون زنجیره‌ای، باید تمام مسیرهایی که یک واریانس از طریق آنها گسترش می یابد محاسبه شود. اگر قانون زنجیره‌ای را به عنوان مدل گرافیکی زیر تعمیم دهیم، مشتق جزئی f نسبت به y_i به‌صورت زیر محاسبه می‌شود. این درک از قانون زنجیره‌ای به فهم هر نوع پس‌انتشار کمک می‌کند.

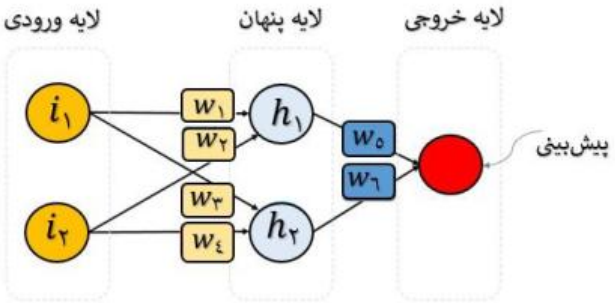
$$\begin{aligned} f(x) &= f(x_1(y_1, \ldots, y_m), \ldots, x_n(y_1, \ldots, y_m)) \\ &= f(x_1(\ldots, y_i, \ldots), \ldots, x_n(\ldots, y_i, \ldots)) \\ &= f(x_1(y_i), x_2(y_i), \ldots, x_n(y_i)) \end{aligned}$$



$$\begin{aligned} \frac{\partial f}{\partial y_i} &= \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial y_i} + \frac{\partial f}{\partial x_2}\frac{\partial x_2}{\partial y_i} + \cdots + \frac{\partial f}{\partial x_n}\frac{\partial x_n}{\partial y_i} \\ &= \sum_{j=1}^{n} \frac{\partial f}{\partial x_j}\frac{\partial x_j}{\partial y_i} \end{aligned}$$



حال،فرض کنید شبکه عصبی با ۳ لایه داریم، که در شکل ۲ ـ ۱۰ قابل نمایش است، و با داشتن مقادیر ورودی ۲ و ۳ و خروجی واقعی آن ۱ می‌باشد.



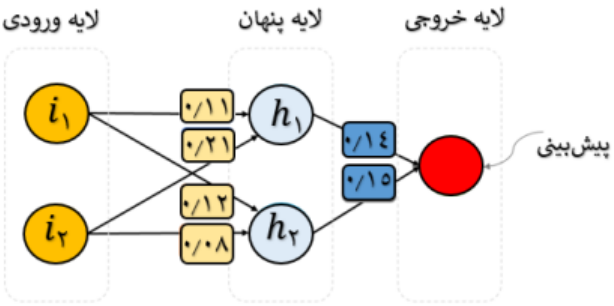
شکل ۲ ـ ۱۰ ساختار شبکه عصبی با یک لایه خروجی

در مرحله اول وزن‌دهی اولیه با مقادیر تصادفی برای وزن‌ها به‌صورت:

$$w_۱ = ۰.۱۱ , w_۲ = ۰.۲۱ , w_۳ = ۰.۱۲ , w_٤ = ۰.۰۸$$

$$w_٥ = ۰.۱٤ , w_٦ = ۰.۱٥$$

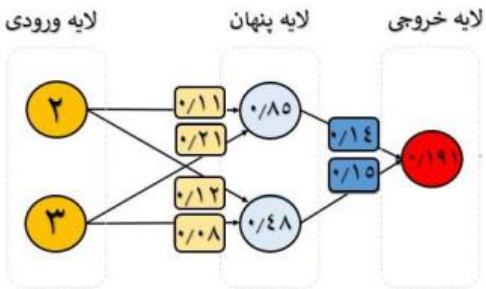
مقداردهی شده‌اند. این مقداردهی در ساختار شبکه در شکل ۲ ـ ۱۱ مشاهده می‌شود.



شکل ۲ ـ ۱۱ مقداردهی اولیه وزن‌های شبکه

پس از مقداردهی اولیه و تغذیه ورودی‌ها به شبکه، ورودی‌ها در وزن‌ها ضرب شده و به لایه دیگر منتقل می‌شوند. سپس جمع وزن‌دار آن‌ها محاسبه و در خروجی یک مقدار پیش‌بینی شده تولید می‌شود، این مرحله در شکل ۲ ـ ۱۲ قابل مشاهده و نحوه محاسبات در این مرحله به صورت زیر می‌باشد:

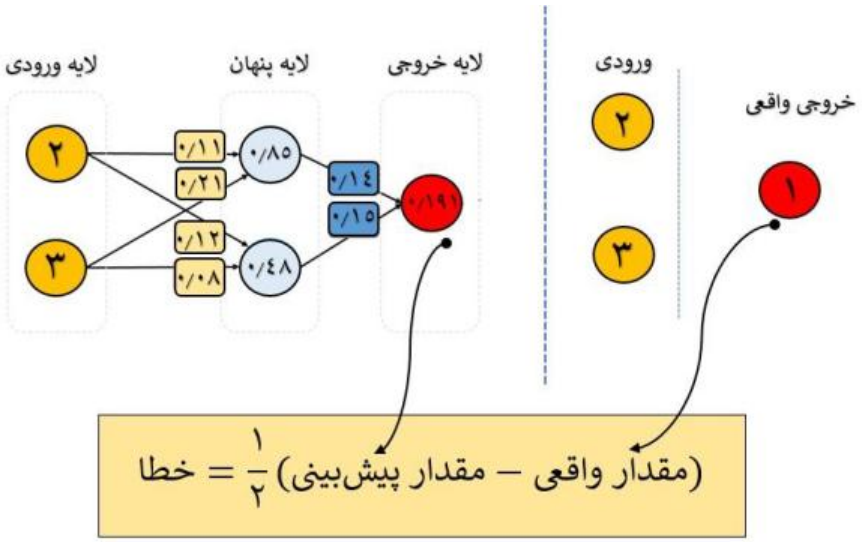
$$[۲ \quad ۳].\begin{bmatrix} ۰.۱۱ & ۰.۱۲ \\ ۰.۲۱ & ۰.۰۸ \end{bmatrix} = [۰.۸۵ \quad ۰.٤۸].\begin{bmatrix} ۰.۱٤ \\ ۰.۱٥ \end{bmatrix} = [۰.۱۹۱]$$



شکل ۲ ـ ۱۲ مرحله جلورو

حال زمان آن رسیده تا ببینیم که شبکه چگونه تفاوت بین خروجی پیش‌بینی شده را با خروجی واقعی در ارزیابی عملکرد شبکه محاسبه می‌کند. همان طور که گفته شد در این مرحله با احتمال زیاد این مقدار با مقدار واقعی اختلاف زیادی خواهد داشت. این تفاوت خطا در شکل ۲ــ۱۳ قابل مشاهده است. محاسبه این خطا در این مرحله پس از محاسبه به زیر بدست می‌آید:

$$\text{خطا} = \frac{۱}{۲}(۰.۱۹۱ - ۱)^{۲} = ۰.۳۲۷$$



شکل ۲ــ۱۳ محاسبه خطا

هدف اصلی ما در آموزش شبکه حداقل رساندن مقدار خطا بین مقادیر خروجی واقعی و پیش‌بینی شده است. از آن‌جا که مقدار خروجی واقعی مسئله مقداری ثابت است، تنها راکاهش خطا تغییر مقدار پیش‌بینی شده است، اما سوال اینجاست که چگونه این مقدار را تغییر دهیم؟ همان‌طور که قابل مشاهده است وزن‌ها تعیین کننده مقدار پیش‌بینی شده هستند، از همین‌رو باید آنها را تغییر داد تا میزان خطا کاهش یابد. این به‌هنگام سازی و تغییر وزن‌ها در مرحله عقبگرد در الگوریتم پس‌انتشار خطا توسط گرادیان کاهشی انجام می‌گیرد. نحوه محاسبه آن در به‌هنگام‌سازی وزن‌ها با فرض اینکه

$\dot{w}_x$ وزن جدید، w_x وزن قدیم و η نرخ یادگیری باشد، به صورت زیر قابل محاسبه است:

$$\dot{w}_x = w_x - \eta\left(\frac{\partial \text{خطا}}{\partial w_x}\right)$$

به عنوان مثال، نحوه به‌هنگام سازی وزن برای $w_۶$ به صورت زیر می‌باشد:

$$\dot{w}_۶ = w_۶ - \eta\left(\frac{\partial \text{خطا}}{\partial w_۶}\right)$$

$$\frac{\partial \text{خطا}}{\partial w_۶} = \frac{\partial \text{خطا}}{\partial \text{پیش‌بینی}} * \frac{\partial \text{پیش‌بینی}}{\partial w_۶}$$

$$\frac{\partial \text{خطا}}{\partial w_۶} = \frac{\partial \frac{۱}{۲}\left(\text{واقعی} - \text{پیش‌بینی}\right)^۲}{\partial \text{پیش‌بینی}} * \frac{\partial (i_۱w_۱ + i_۲w_۲)w_۵ + (i_۱w_۳ + i_۲w_٤)w_۶}{\partial w_۶}$$

$$\frac{\partial \text{خطا}}{\partial w_۶} = ۲ * \frac{۱}{۲}\left(\text{واقعی} - \text{پیش‌بینی}\right)\frac{\partial \left(\text{واقعی} - \text{پیش‌بینی}\right)}{\partial \text{پیش‌بینی}} * (i_۱w_۳ + i_۲w_٤)$$

$$\frac{\partial \text{خطا}}{\partial w_۶} = \left(\text{واقعی} - \text{پیش‌بینی}\right) * (h_۲)$$

$$\frac{\partial \text{خطا}}{\partial w_۶} = \Delta h_۲$$

بر این اساس $w_۶$ به صورت زیر به‌هنگام می‌شود:

$$\dot{w}_۶ = w_۶ - \eta \Delta h_۲$$

پس از به‌هنگام‌سازی برای همه وزن‌ها، به‌صورت زیر خلاصه می‌شوند:

$$\dot{w}_۶ = w_۶ - \eta(\Delta . h_۲)$$

$$\dot{w}_{۵} = w_{۵} - \eta(\Delta . h_{۱})$$

$$\dot{w}_{٤} = w_{٤} - \eta(\Delta w_{٦} . i_{۲})$$

$$\dot{w}_{۳} = w_{۳} - \eta(\Delta w_{٦} . i_{۱})$$

$$\dot{w}_{۲} = w_{۲} - \eta(\Delta w_{۵} . i_{۲})$$

$$\dot{w}_{۱} = w_{۱} - \eta(\Delta w_{۵} . i_{۱})$$

پس از بازنویسی و قرار دادن آن‌ها در ماتریس به صورت زیر قابل مشاهده هستند:

$$\begin{bmatrix} w_{۵} \\ w_{٦} \end{bmatrix} = \begin{bmatrix} w_{۵} \\ w_{٦} \end{bmatrix} - \eta\Delta \begin{bmatrix} h_{۱} \\ h_{۲} \end{bmatrix} = \begin{bmatrix} w_{۵} \\ w_{٦} \end{bmatrix} - \begin{bmatrix} \eta(\Delta . h_{۱}) \\ \eta(\Delta . h_{۲}) \end{bmatrix}$$

$$\begin{bmatrix} w_{۱} & w_{۳} \\ w_{۲} & w_{٤} \end{bmatrix} = \begin{bmatrix} w_{۱} & w_{۳} \\ w_{۲} & w_{٤} \end{bmatrix} - \eta\Delta \begin{bmatrix} i_{۱} \\ i_{۲} \end{bmatrix} . \begin{bmatrix} w_{۵} & w_{٦} \end{bmatrix}$$

$$\begin{bmatrix} w_{۱} & w_{۳} \\ w_{۲} & w_{٤} \end{bmatrix} = \begin{bmatrix} w_{۱} & w_{۳} \\ w_{۲} & w_{٤} \end{bmatrix} - \begin{bmatrix} \eta i_{۱}\Delta w_{۵} & \eta i_{۱}\Delta w_{٦} \\ \eta i_{۲}\Delta w_{۵} & \eta i_{۲}\Delta w_{٦} \end{bmatrix}$$

اکنون با استفاده از این معادله‌ها می‌توان وزن‌های جدید را برای مثال خود به‌هنگام کنیم، با فرض اینکه مقدار نرخ یادگیری برابر ۰/۰۵ مقادیر وزن‌های جدید به صورت زیر محاسبه می‌شوند:

$$\Delta = ۰.۱۹۱ - ۱ = -۰.۸۰۹$$

$$\begin{bmatrix} w_{۵} \\ w_{٦} \end{bmatrix} = \begin{bmatrix} ۰.۱٤ \\ ۰.۱۵ \end{bmatrix} - ۰.۰۵(-۰.۸۰۹) \begin{bmatrix} ۰.۸۵ \\ ۰.٤۸ \end{bmatrix} = \begin{bmatrix} ۰.۱۷ \\ ۰.۱۷ \end{bmatrix}$$

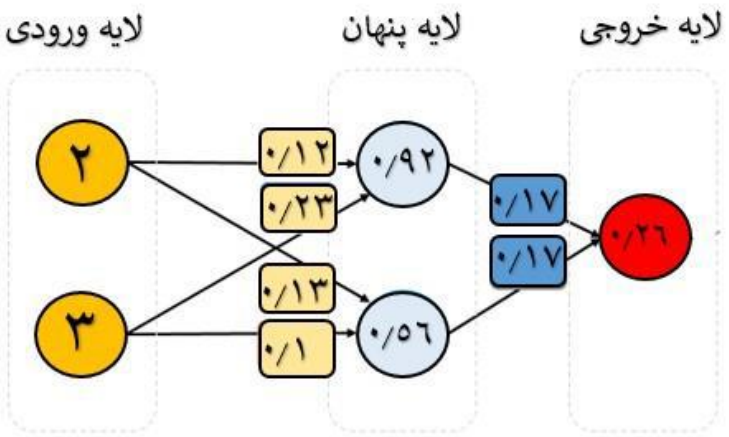
$$\begin{bmatrix} w_{۱} & w_{۳} \\ w_{۲} & w_{٤} \end{bmatrix} = \begin{bmatrix} ۰.۱۱ & ۰.۱۲ \\ ۰.۲۱ & ۰.۰۸ \end{bmatrix} - ۰.۰۵(-۰.۸۰۹) \begin{bmatrix} ۲ \\ ۳ \end{bmatrix} . \begin{bmatrix} ۰.۱٤ & ۰.۱۵ \end{bmatrix}$$

$$\begin{bmatrix} w_{۱} & w_{۳} \\ w_{۲} & w_{٤} \end{bmatrix} = \begin{bmatrix} ۰.۱۱ & ۰.۱۲ \\ ۰.۲۱ & ۰.۰۸ \end{bmatrix} - \begin{bmatrix} -۰.۰۱۱ & -۰.۰۱۲ \\ ۰.۰۱۷ & -۰.۰۱۸ \end{bmatrix} = \begin{bmatrix} ۰.۱۲ & ۰.۱۳ \\ ۰.۲۳ & ۰.۱۰ \end{bmatrix}$$

اکنون پس از به‌هنگام سازی وزن‌ها دوباره فاز جلورو را تکرار می‌کنیم، در شکل ۲ ـ ۱٤ این مرحله قابل مشاهده است.

نحوه محاسبه مقادیر خروجی جدید به‌صورت زیر می‌باشد.

$$[۲ \quad ۳].\begin{bmatrix} ۰.۱۲ & ۰.۱۳ \\ ۰.۲۳ & ۰.۱۰ \end{bmatrix} = [۰.۹۲ \quad ۰.۵۶].\begin{bmatrix} ۰.۱۷ \\ ۰.۱۷ \end{bmatrix} = [۰.۲۶]$$



شکل ۲ـ۱۴ تکرار دوباره فاز جلورو

همان‌طور که مشاهده می‌شود مقدار خروجی در تکرار دوم الگوریتم از ۰/۱۹۱ به ۰/۲۶ تغییر پیدا کرد، که این مقدار به خروجی واقعی نزدیک‌تر شده است. با تکرار چندین‌باره الگوریتم می‌توان میزان خطا را نزیک یا مساوی صفر کرد.

۷.۲ چالش‌های آموزش در شبکه‌های عمیق

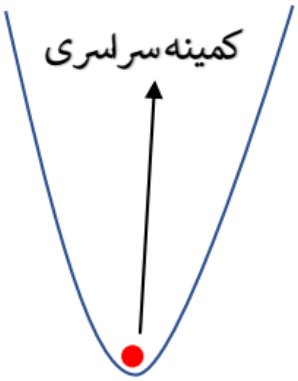
آموزش در شبکه‌های عمیق فرآیندی بسیار سخت بوده، از همین‌رو مشکلات و موانع پیچیده‌ای را به‌همراه خواهد داشت. در این بخش ما به چالش‌های مهمی که در آموزش شبکه‌های عمیق با آن مواجه می‌شویم، خواهیم پرداخت.

۶.۳.۲ محو گرادیان و انفجار گرادیان

فرآیند آموزش در شبکه‌های عمیق شامل یافتن مجموعه‌ای از وزن‌ها در شبکه است، این وزن‌ها نمایانگر یادگیری در شبکه برای مسئله مورد نظر هستند.

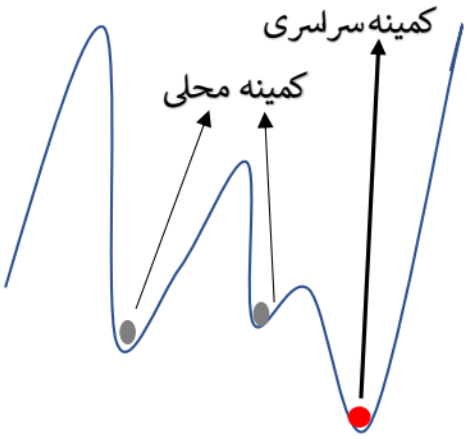
آموزش در شبکه‌های عصبی روندی تکرار شونده است، یعنی گام‌به‌گام با بروزرسانی‌های کوچک در وزن‌ها و با تکرار این عمل، عملکرد مدل در حل مسئله بهبود پیدا می‌کند. این فرآیند، یک مسئله بهینه‌سازی را بوجود می‌آورد، چرا که شبکه سعی می‌کند براساس وزن‌ها تابع زیان را حداقل کند. همین مسئله بهینه‌سازی سبب ایجاد چالش‌هایی در شبکه می‌شود. سوال اینجاست که دقیقا چه چیزی در این مسئله بهینه‌سازی چالش برانگیر می‌باشد؟

همان‌طور که دیدیم الگوریتم عمومی و متداول برای بهینه‌سازی در شبکه‌های عمیق گرادیان کاهشی است. که از این بهینه‌ساز در الگوریتم پس‌انتشار جهت به‌هنگام‌سازی وزن‌ها در هر تکرار استفاده می‌شود تا مقدار خطای تابع زیان حداقل شود. بهینه‌ساز در هر تکرار وزن‌ها را به‌هنگام و به دنبال یک کمینه سراسری می‌گردد. در مسائل بهینه‌سازی ساده، می‌توان آن را در تشابه یک کاسه بزرگ همانند شکل ۲ــ۱۵ مشاهده کرد، که پیدا کردن کف این کاسه به راحتی با الگوریتمی کار قابل انجام است. این مسایل در ریاضیات با عنوان بهینه‌سازی محدب شناخته می‌شوند.



شکل ۲ــ۱۵ بهینه‌سازی محدب

حال آنکه در بهینه‌سازی وزن‌های شبکه عصبی، مسئله شکلی همانند کاسه نخواهد داشت، بلکه منظره‌ای دارای تپه‌ها و دره‌ها است، همانند شکل ۲ــ۱۶. این نوع مسائل در ریاضیات به عنوان بهینه‌سازی غیر محدب شناخته‌شده هستند.



شکل ۲ـ۱۶ بهینه‌سازی غیر محدب

در حقیقت، این‌طور می‌توان بیان کرد که، یافتن مجموعه وزن‌های بهینه در شبکه‌های عصبی در زمان چندجمله‌ای قابل وجود ندارد. این نوع مسائل با عنوان ان‌پی کامل در علوم کامپیوتر شناخته شده هستند.

اکنون به بررسی مشکلاتی که در هنگام آموزش در شبکه عصبی عمیق با استفاده از گرادیان کاهشی و الگوریتم پس‌انتشار با آن مواجه خواهیم شد، می‌پردازیم. اضافه کردن لایه‌های پنهان بیشتر در شبکه‌های عصبی سبب می‌شود که شبکه بتواند توابع پیچیده‌تری را یاد بگیرد و این همان تفاوت بزرگی است که شبکه‌های عصبی با شبکه‌های عمیق دارند. اما هنگام استفاده از الگوریتم پس‌انتشار، در فاز عقبگرد محاسبه گرادیان کوچک و کوچک‌تر می‌شود. این اتفاق به این دلیل بوجود می‌آید که، گرادیان کاهشی در هر تکرار مشتقات جزئی را با طی کردن از لایه پایانی به سمت لایه ابتدایی با استفاده از قانون زنجیره‌ای می‌یابند. در شبکه‌ای با داشتن n لایه پنهان، مشتقات این n لایه در یک‌دیگر ضرب می‌شود. حال اگر این مشتقات کوچک باشند، با رفتن به لایه‌های اولیه به صورت نمایی کاهش پیدا و (یا در بدترین حالت صفر می‌شوند و یادگیری شبکه متوقف می‌شود) همین امر سبب پدیده محو گرادیان می‌شود. از آنجایی که این گرادیان کوچک در تکرار الگوریتم به‌هنگام‌سازی نمی‌شوند و این لایه‌های اولیه اغلب در شناخت داده‌ها موثر هستند، منجر به عدم دقت کافی شبکه می‌شوند. در مقابل این اگر این مقادیر

مشتقات بزرگ باشند با رشد نمایی از طریق انتقال به لایه‌ها، سبب سرریز شده و وزن‌ها دیگر توانایی به‌هنگام‌سازی نخواهند داشت، و شبکه‌ای ناپایدار را پدید می‌آورد.

۱.۶.۳.۲ راه‌های شناسایی مشکلات محو گرادیان و انفجار گرادیان؟

حال با این مسئله روبه‌رو می‌شویم که از کجا تشخیص دهیم شبکه ما دچار مشکلات محو گرادیان و انفجار گرادیان شده است! چندین روش برای شناسای این مشکلات وجود داشته که به صورت خلاصه در زیر آن‌ها را فهرست کرده‌ایم.

شناسایی انفجار گرادیان

- به دلیل عدم ثبات مدل، تغییرات زیادی در به‌هنگام‌سازی وزن‌ها مشاهده شود. وزن‌ها در هنگام آموزش به‌صورت نمایی رشد می‌کنند.
- در طول فرآیند آموزش تابع هزینه مقدار NaN بگیرد.
- مدل اطلاعات زیادی را در طول فرآیند آموزش فرا نمی‌گیرد، بنابراین تابع هزینه ضعیفی دارد.

شناسایی محو گرادیان

- در طول فرآیند آموزش بهبود مدل بسیار آهسته می‌باشد، و ممکن است فرآیند آموزش خیلی زود متوقف شود یعنی، هیچ آموزش دیگری سبب بهبود مدل نمی‌شود.
- وزن‌های نزدیک به لایه خروجی شاهد تغییراتی بیشتری نسبت به لایه‌های نزدیک به ورودی دارند.
- وزن‌های مدل به‌صورت نمایی کاهش پیدا کند.
- وزن مدل در هنگام آموزش صفر شود.

۲.۶.۳.۲ راه حل گریز از مشکلات محو گرادیان و انفجار گرادیان

روش‌های مختلفی جهت مقابله با محو گرادیان و انفجار گرادیان وجود دارد که در زیر چند مورد از آن‌ها ذکر شده‌اند.

۱. **استفاده از توابع فعال‌سازی دیگر (یکسوساز خطی)**
۲. **انتخاب روش وزن‌دهی اولیه دیگر**
۳. **بریدگی گرادیان**[۱]: این روش که برای انفجار گرادیان مناسب است، اندازه گرادیان را با یک حد آستانه محدود می‌کند. این امر سبب می‌شود، گرادیان‌هایی که حد آستانه‌ای بالاتر از هنجار تعیین شده دارند، قطع شده تا با هنجار مطابقت پیدا کنند.

۷.۳.۲ بیش‌برازش

از جنبه‌های قابل توجه شبکه‌های تماما متصل به‌یاد سپاری است. یعنی اگر زمان کافی به آن‌ها داده شود، این توانایی را بدست خواهند آورد که تمامی داده‌های آموزشی را به‌یاد بسپارند. از همین‌رو، همگرایی یک شبکه معیاری جهت ارزیابی عملکرد شبکه نمی‌باشد. چرا که اگر همگرایی بیش‌از حد صورت گیرد، شبکه تمام داده‌ها را حفظ کرده و دیگر قابلیت تعمیم‌دهی نخواهد داشت.

در شبکه‌های عمیق روند روبه کاهش به سمت صفر در تابع هزینه متداول بوده، و این امر دلیلی بر اثبات توانایی تعمیم‌دهی شبکه نبوده، و قدرتمندی یادگیری یک شبکه را نشان نمی‌دهد. چرا که این امکان وجود دارد شبکه، ویژگی و حالتی‌هایی از مجموعه داده‌ی تغذیه شده به آن را به‌یاد سپرده باشد، که در مجموعه داده‌ی دیگر کابردی ندارد. در مجموعه داده‌های با ابعاد بالا احتمال وجود همبستگی‌های عجیبی وجود دارد که، شبکه‌های تماما متصل قادرند آن‌ها را کشف و به‌کار ببرند، بنابراین برای موفقیت شبکه در داشتن عملکردی بهتر، باید از این رفتارها جلوگیری شود.

چالش اصلی در یادگیری ماشین این است که مدل باید در برخورد با داده‌های جدید نه فقط داده‌هایی که آن‌ها را یاد گرفته، عملکرد خوبی از خود نشان دهد به عبارتی توانایی تعمیم‌دهی داشته باشد. حال آنکه شبکه‌های یادگیری عمیق به‌دلیل تعداد زیاد پارامترهای یادگیری، توابع پیچیده را براساس داده‌های ورودی مدل می‌کند، و اگر تعداد این داده‌ها کم باشد مدل فقط روی همین داده‌ها عملکرد خوبی داشته و قابلیت تعمیم‌پذیری نخواهد داشت. به عبارت دیگر، آموزش یک شبکه عصبی را باید به‌اندازه‌ای

[۱] gradient clipping

تکرار کنیم تا قادر به نگاشت بین ورودی و خروجی شود. اما آموزش نباید آنقدر طولانی شود که شبکه نویزهای آماری را فراگیرد و فقط با داده‌های آموزشی همخوانی داشته باشد و تعیم‌دهی شبکه کاهش پیدا کند.

در جهت حل این مسئله دو رویکرد وجود دارد، نخست این‌که داده‌های بیشتری را جمع‌آوری کرده و به مدل تغذیه کنیم، حال آن‌که در بیشتر اوقات این روش امکان‌پذیر نبوده و همچنین این روش فرآیندی پرهزینه می‌باشد. روش دوم که از آن به عنوان **منظم‌سازی**[۱] نام برده می‌شود، رویکردی کاربردی در جهت مقابله با کاهش بیش‌برازش می‌باشد. منظم سازی باعث تغییر جزئی در الگوریتم یادگیری می‌شود، به‌طوری که مدل قابلیت تعمیم‌دهی بدست آورده تا در مواجه با داده‌هایی که تاکنون آن‌ها را مشاهده نکرده، عملکرد بهتری از خود نشان دهد. در ادامه به تشریح روش‌های منظم‌سازی می‌پردازیم.

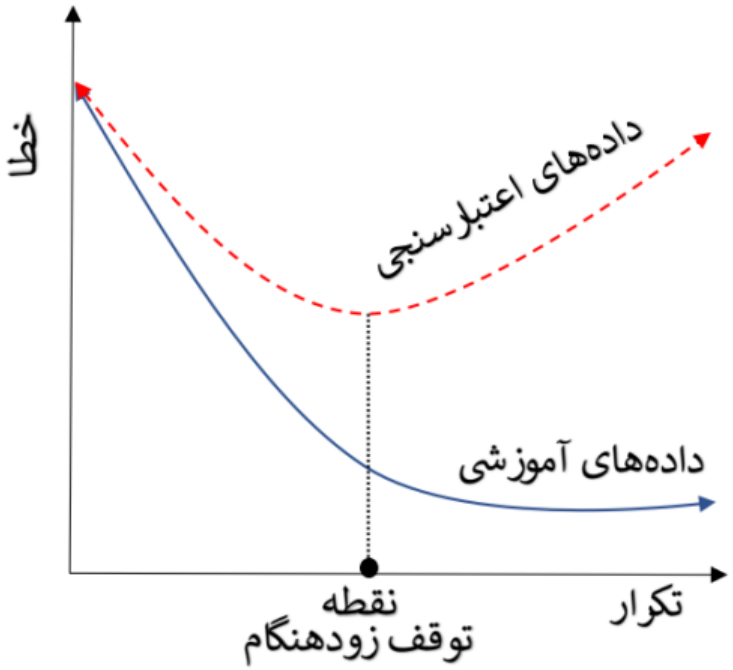
۱.۷.۳.۲ توقف زودهنگام[۲]

توقف زودهنگام یکی از روش‌های ساده و رایج در جهت مقابله با بیش‌برازش است. ایده اصلی این روش که در شکل ۲ـ۱۷ قابل مشاهده است. با استفاده از این روش می‌توان تعداد تکرارهای مناسب فرآیند آموزش شبکه را مشخص کرد. برای استفاده از این روش از داده‌های اعتبارسنجی جهت محاسبه عملکرد تابع زیان در پایان هر تکرار استفاده می‌شود، و تا نقطه‌ای (تکرار) که عملکرد داده‌های اعتبارسنجی شبکه بهبود می‌یابد فرآیند تکرار آموزش ادامه پیدا می‌کند.

توقف زودهنگام از آن جهت که تقریبا نیاز به اعمال هیچ نوع تغییری در فرآیند آموزش ندارد، روشی غیرتداخلی و ملایم منظم‌سازی است. بدان معنا که استفاده از این روش هیچ‌گونه خدشه‌ای در پویایی یادگیری شبکه نمی‌گذارد. این روش را می‌توان به تنهایی یا به‌همراه سایر روش‌های منظم‌سازی به‌کار برد.

---

[۱] Regularization
[۲] Early Stopping



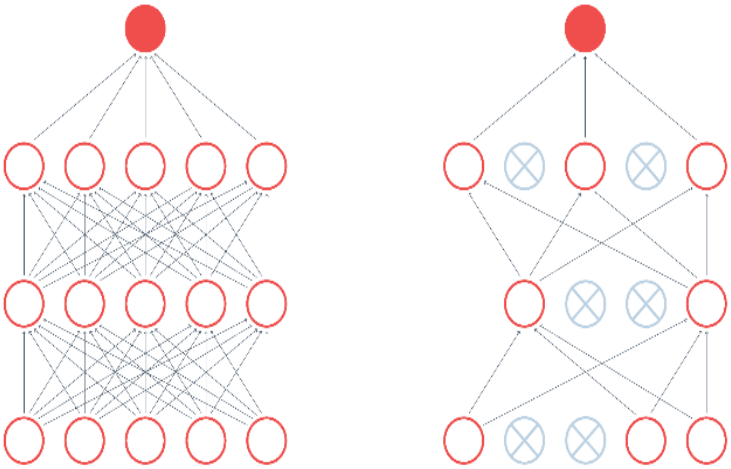
شکل ۲ـ۱۷ توقف زودهنگام

۲.۷.۳.۲ حذف تصادفی[۱]

فرض کنید که هر روز با تعداد زیادی از افراد ملاقات می کنید. درحالی که به صورت حضوری با آن ها صحبت می کنید، چهره آن‌ها را به‌خاطر می‌سپارید. در بعضی مواقع مجبورید از طریق تلفن ارتباط برقرار کنید. اما این‌بار همان افراد را تشخیص نمی دهید، چرا که فقط آن‌ها را دیده اید و تنها چهره‌شان را به یاد دارید. حال تصور کنید که فقط می توانید از طریق تلفن با شخص صحبت کنید. در این حالت باید یاد بگیرد براساس صدای آن‌ها، بتوانید آن‌ها را به‌خاطر بسپارید. بنابراین، شما با حذف تصادفی ویژگی های بصری، مجبور می شوید روی ویژگی های صوتی تمرکز نمایید. این همان کاری است که حذف تصادفی در شبکه‌های عصبی انجام می‌دهد، تا شبکه عصبی مجبور به یادگیری ویژگی‌های مفید بیشتری گردد.

[۱] Dropout

حذف تصادفی روشی قدرتمند با هزینه کم محاسباتی برای منظم‌سازی شبکه‌های عصبی است. ایده ساده این روش به این‌صورت است که در طول فرآیند آموزش در هر تکرار، هر نرون با احتمال p در شبکه حفظ و با احتمال $p - ۱$ از شبکه حذف (غیرفعال) شود. این میزان احتمال را معمولا مقدار ۰/۵ در نظر می‌گیرند، با مقدار $p = ۱$ هیچ‌گونه نرونی از شبکه حذف نخواهد شد. در شکل ۲ــ۱۸ شمایی از یک شبکه قبل و پس از حذف تصادفی قابل مشاهده است.



شکل ۲ــ۱۸ حذف تصادفی. شکل سمت راست یک شبکه پس از حذف تصادفی

در شبکه‌های عمیق، نورون‌های شبکه به سرعت وابستگی به نورونی که اطلاعات کاربردی را فراگرفته پیدا می‌کنند و اطلاعات را از آن می‌گیرند. این وابستگی به یک نورون بی‌ثباتی در شبکه ره به‌همراه دارد، چرا که شبکه به ویژگی‌های فراگرفته توسط این نورون وابسته است، حال آن‌که این ویژگی نمایانگر تمام داده‌ها نیست. با استفاده از روش حذف تصادفی، به دلیل این‌که، این امکان وجود خواهد داشت نورون حذف شده، همان نورون قدرمند ذکر شده باشد، این وابستگی را از بین خواهد برد. این عمل نورون‌ها را مجبور به یادگیری به‌صورت مستقل می‌کنند، که نتیجه آن عملکرد بهتر یادگیری شبکه را به‌همراه خواهد داشت.

۳.۷.۳.۲ یکسان‌سازی دسته‌ای[۱]

یکی از مشکلاتی که در آموزش شبکه‌های عصبی علاوه بر محو گرادیان و انفجار گرادیان وجود دارد، مشکل تغییر متغیرهای داخلی شبکه است. این مشکل از آن‌جا ناشی می‌شود که پارامترها در طول فرآیند آموزش مدام تغییر می‌کند، این تغییرات به نوبه خود مقادیر توابع فعال‌سازی را تغییر می‌دهد. تغییر مقادیر ورودی از لایه‌های اولیه به لایه‌های بعدی سبب همگرایی کندتر در طول فرآیند آموزش می‌شود، چرا که داده‌های آموزشی لایه‌های بعدی پایدار نیستند.

در بیانی دیگر، شبکه‌های عمیق ترکیبی از چندین لایه با توابع مختلف بوده، و هر لایه فقط یادگیری بازنمایی کلی از ابتدای آموزش را فرا نمی‌گیرد، بلکه باید با تغییر مداوم در توزیع‌های ورودی با توجه به لایه‌های قبلی تسلط پیدا کند. حال آن‌که بهینه‌ساز بر این فرض به‌هنگام‌سازی پارامترها را انجام می‌دهد که، در لایه‌های دیگر تغییر نکنند و تمام لایه‌ها را هم‌زمان به‌هنگام می‌کند، این عمل سبب نتایج ناخواسته‌ای هنگام ترکیب توابع مختلف خواهد شد. **یکسان‌سازی دسته‌ای** در جهت غلبه بر این مشکل برای کاهش ناپایداری و بهبود شبکه ارائه شده است. در این روش، یکسان‌سازی برروی داده‌های ورودی یک لایه را به گونه‌ای انجام می‌دهد، که دارای میانگین صفر و انحراف معیار یک شوند. این عمل فرآیند یادگیری را در مدل ساده‌تر می‌کند، چرا که پارامترها در لایه‌های قبلی در بیشتر موارد فاقد اثر خواهند شد. بدون انجام این یکسان‌سازی، هر به‌هنگام‌سازی یک تغییر شدید بر مدل خواهد گذاشت. به‌طور خلاصه می‌توان گفت، با قرار دادن یکسان‌سازی دسته‌ای بین لایه‌های پنهان و با ایجاد ویژگی واریانس مشترک، سبب کاهش تغییرات داخلی لایه‌های شبکه می‌شویم. مزایای استفاده از یکسان‌سازی دسته‌ای در شبکه به‌صورت زیر خلاصه شده است:

۱. **کمک به منظم‌سازی مدل.** آزمایش‌ها نشان داده است که استفاده از این روش، تا حدودی نیاز به سایر روش‌های منظم‌سازی همانند: حذف تصادفی را کاهش می‌دهد.
۲. **امکان استفاده از نرخ یادگیری بزرگ‌تر.** در شبکه‌هایی که از یکسان‌سازی دسته‌ای استفاده نمی‌کنند، یک نرخ یادگیری بزرگ سبب بروز نوسانات شود که نتیجه آن به‌جای

[۱] Batch Normalization

کاهش خطای تابع زیان، سبب افزایش آن شود. یکسان‌سازی دسته‌ای این مشکل را تا حدودی حل، و از این‌رو امکان استفاده از نرخ یادگیری بزرگ‌تر و نوبه خود افزایش سرعت یادگیری شبکه را به‌همراه خواهد داشت.

۳. **با استفاده از آن گرادیان بهتری از طریق شبکه امکان‌پذیر می‌شود، که در نتیجه آن امکان استفاده از لایه‌های پنهان بیشتر فراهم می‌شود.**
٤. **درکاهش وابستگی به مقداردهی اولیه پارامترها کمک می‌کند.**
٥. **احتمال بیش‌برازش راکاهش می‌دهد.** چراکه هنگام آزمایش کم‌تر تحت تاثیر نویز قرار می‌گیرد (داده‌ها یکسان‌سازی شده‌اند).
٦. **محو گرادیان کم‌تر.** به‌ویژه برای توابع فعال‌سازی دارای فضای اشباع (سیگموید و غیره)

۸.۳.۲ در دسترس بودن و کیفیت داده‌های آموزشی

الگوریتم‌های یادگیری عمیق، یادگیری را از طریق نمونه‌های آموزشی که همان داده‌های مسئله مورد نظر هستند، می‌آموزد. جهت اطمینان از اینکه، این شبکه کارکرد خوب و نتایج مطلوبی را ارائه دهد نیاز است تا پارامترهای زیادی را فراگیرند. هر چه شبکه پیچیده‌تر شود، این تعداد پارامترها بیشتر و به معنای انتزاع قوی‌تر است. فراگیری این تعداد پارامترها نیاز به داده‌های بسیار زیاد دارد و موفقیت شبکه وابسته به همین اندازه داده‌ها است. به عنوان مثال در ساخت مدلی برای شناسایی گفتار، به داده‌هایی از لهجه‌های مختلف مورد نیاز است. برای ساخت چنین سیستمی برای حتی یک زبان واحد، نیاز به داده‌های بسیار بزرگ جهت تغذیه به مدل است.

هر چند منظم‌سازی یکی از روش‌های رایج برای جلوگیری از بیش‌برازش می‌باشد. اما با افزایش مقدار داده‌ها نیز می‌توان بر این مشکل غلبه کرد. داده‌ها مهم‌ترین مولفه هر مدل یادگیری ماشین هستند. به‌طور خلاصه، شبکه‌های عصبی از تجربیاتی که با آن‌ها روبه‌رو هستند، آموزش را فرا می‌گیرند. تنظیم ابرپارامترها معمولا بهترین مرحله برای بهبود خطای تعمیم‌دهی است. اگر بازهم شکافی بین آموزش و خطای تعمیم وجود داشت، افزایش داده‌ها در بیشتر مواقع کمک کننده می‌باشد. نکته دیگر کیفیت داده‌های آموزشی است. چراکه تاثیر زیادی در آموزش مدل می‌گذارند. شبکه‌های عصبی می‌تواند تا حدی در یک مجموعه داده‌ها در زمان فرآیند آموزش، نویزها را کاهش دهند. با این

حال، داده‌ها غلط سبب بروز مسائل زیادی می‌شود. گاهی عملکرد ضعیف مدل در برنامه‌های واقعی ناشی از برچسب‌گذاری نادرست (داده‌های بی‌کیفیت) یا همان تعداد داده‌های کم می‌باشد. **در مدل آموزش دیده، اگر به‌نظر می‌رسد در تمام مراحل آموزش رفتار عجیب و غریبی وجود دارد، ممکن است نشانه عدم تطابق داده‌ها باشد.**

یادگیری عمیق از مجموعه داده‌های بزرگ در مقایسه با سایر الگوریتم‌های یادگیری ماشین بیشترین سود را می‌برد. بسیاری از پیشرفت‌های کیفی حاصل از یادگیری عمیق به‌طور مستقیم به افزایش اندازه مجموعه داده‌های آموزشی مربوط می‌باشد. مجموعه داده‌های بزرگ می‌تواند به عنوان یک روش قاعده‌مند عمل کنند تا از بیش‌برازش در یک مدل جلوگیری کند.

۴.۲ بهینه‌سازی ابرپارمترها

یادگیری یک شبکه عمیق با یادگیری یک شبکه عمیق که قابل استفاده باشد، بسیار متفاوت هستند. در جهت آموزش مدلی مناسب از یادگیری عمیق، مراحل زیادی را باید پشت سرگذاشت. نحوه تنظیم ابرپارامترها یکی از آن‌ها می‌باشد. فرآیند تنظیم پارامترهایی از مدل که مقدار آن‌ها قبل از شروع یادگیری شبکه باید انجام گیرد **بهینه‌سازی ابرپارامترها**[۱] گفته می‌شود، و در تلاش برای به‌دست آوردن حداکثر کارایی از سیستم است.

پارامترهایی همانند تعداد نرون‌ها در هر لایه پنهان، تعداد لایه‌های پنهان، اندازه دسته و نرخ یادگیری از این دست پارامترها می‌باشند. عملکرد شبکه‌های عمیق حساسیت بسیار زیادی به این پارامترها دارد. تنظیم نامناسب ابرپارامترها فرآیند یادگیری شبکه را به‌طور کامل با شکست مواجه خواهد کرد.

۱.۴.۲ تفاوت پارامترهای مدل با ابرپارمترها در شبکه‌های عصبی؟

در آموزش شبکه‌های عصبی دو نوع پارامتر نقش داشته و با یک‌دیگر متفاوت هستند، در ادامه نگاهی گذرا به تفاوت بین آن‌ها می‌اندازیم.

[۱] hyperparameter optimization

- **پارامترهای مدل:** این پارامترها متغیرهایی هستند که از مجموعه داده‌ها در طول فرآیند آموزش تخمین زده شده و مقادیر آن‌ها به‌صورت دستی تعیین نشده است. به عبارتی این پارامترها، متغیرهای داخلی در شبکه هستند. *مدل از این پارامترها برای پیش‌بینی استفاده می‌کند.*
- **ابرپارامترها:** ابرپارامترها متغیرهای خارجی در پیکره‌بندی شبکه هستند. به عبارت دیگر، مقدار این متغیرها قبل از شروع یادگیری تعیین می‌شود. این پارامترها تاثیر بسیار زیادی در سرعت و عملکرد شبکه دارند.

جهت درک بیشتر تفاوت بین پارامترها و ابرپارامترها ارائه مثالی تفکیک آن‌ها از یک‌دیگر را شفاف‌تر خواهد کرد. فرض کنید شما قصد یادگیری رانندگی را دارید. برای انجام این کار شما نیاز به یک مربی داشته تا در طی چندین جلسه به شما آموزش دهد. مربی با کمک دروس و تمرین به شما تا زمانی که مطمئن شود می‌توانید تنهایی رانندگی کنید، آموزش‌ها و تمرین‌ها را ادامه می‌دهد. پس از آموزش هنگامی که توانستید رانندگی کنید، دیگر نیازی به مربی نخواهید داشت. در این سناریو مربی نقش ابرپارامترها، و شما نقش پارامترها را دارید.

همان‌طور که قبلا مشاهده شد، پارامترها متغیرهایی هستند که در طول فرآیند آموزش برآورد می‌شوند، و در پایان آموزش از آن‌ها برای پیش‌بینی استفاده می‌شود. در مقابل ابرپارمترها وابسته به مجموعه داده‌ها نبوده و بخشی از مدل نهایی نیستند، و برای برآورد پارامترهای مدل از آن‌ها استفاده می‌شود. هرگاه در تفکیک آن‌ها با مشکل برخوردید از این قانون ساده استفاده کنید: **اگر مجبورید مقداری را قبل از آموزش تعیین کنید، آن یک ابرپارامتر خواهد بود.** مقایسه بین پارامترهای مدل و ابرپارامترها به‌طور خلاصه در جدول ۲ــ۲ قابل مشاهده است.

جدول ۲ــ۲ مقایسه پارامترها مدل با ابرپارامترها

ابرپارامتر	پارامتر
مقادیر قبل از آموزش تعیین می‌شوند.	مقادیر آن‌ها در طول فرآیند آموزش برآورد می‌شود.
متغیرهای خارجی هستند.	بخشی از مدل هستند.
وابسته به داده‌ها نیستند.	وابسته به داده هستند.

۲.۴.۲ تنظیم ابرپارامترها

تنظیم یا بهینه‌سازی ابرپارامترها در تلاش برای یافتن بهترین مقادیر هر ابرپارامتر جهت این‌که مدل بهترین و دقیق‌ترین پیش‌بینی را ارائه دهد. در تنظیم ابرپارامترها همیشه از یک معیار استفاده تا با امتحان مقادیر مختلف بهینه‌سازی صورت گیرد. در زیر لیستی از ابرپارامترهای رایج در شبکه‌های عمیق فهرست شده‌اند:

- تعداد لایه‌های پنهان
- نرخ یادگیری
- تابع فعال‌سازی
- اندازه ریزدسته
- دوره
- حذف تصادفی
- وزن‌دهی اولیه
- الگوریتم بهینه‌سازی

همان‌طور که در فصل ۱ بیان شد، برای تنظیم ابرپارامترها از داده‌های اعتبارسنجی استفاده می‌شود. در ادامه این بخش مروری کوتاه بر روش‌های مختلف بهینه‌سازی ابرپارمترها می‌پردازیم.

۱.۲.۴.۲ تنظیم دستی ابرپارامترها (آزمون و خطا)

ساده‌ترین راه تنظیم ابرپارمترها مقداردهی مقادیر مختلف و بررسی نتایج است. با تنظیم دستی در هر مرحله انتخاب فعلی پارامتر و تفاوت اختلاف با نتیجه قبلی قابل بررسی و مقایسه خواهد بود. هر چند ایده‌ای ساده به‌نظر می‌رسد، اما نتایج خوبی را به‌همراه خواهد داشت. یک شاغل در حوزه یادگیری عمیق باید در زمینه ساختار شبکه‌ها تجربه‌هایی کسب کند، این روش با امتحان کردن نتایج مختلف تجربیات ارزشمندی را انتقال می‌دهد و می‌تواند بسیار کمک کننده باشد. در این روش بهتراست که به‌صورت منظم عمل کرده و تمام نتایج خود را ثبت و فرآیند بهبود و عملکرد کنترل شود، تا با تحلیل و بررسی بفهمید کدام یک از پارامترها در عملکرد مدل تاثیر بیشتری می‌گذارند.

مزایا:

- می‌توانید جوهره رفتار ابرپارامترها را یاد بگیرید و از دانش خود در پروژه‌ای دیگر استفاده کرد.

معایب:

- نیاز به کار دستی دارد.
- ممکن است بدون امتحان زیاد براساس همین امتیاز گرفته شده بسنده کنید.

۲.۲.۴.۲ جستجوی شبکه‌ای[۱]

بارها و بارها آزمایش کردن و ترکیب مختلف از مقادیر برای ابرپارامترها به‌صورت دستی یک کار خسته‌کننده، وقت‌گیر و نیاز به تجربه‌ی زیاد در درک مدل دارد. جستجوی شبکه به‌سادگی سعی می‌کند تنظیم ابرپارامترها را در محدوده مشخصی از مقادیر انجام دهد. این روش مرتبا آزمایش مقادیر مختلف هر ابرپارامتر را به‌صورت خودکار با استفاده از مقدار متعدد متغیرها انجام می‌دهد.

مزایا:

- تمام مجموعه‌های احتمالی را پوشش می‌دهد.

معایب:

- زمان اجرای کل ابرپارامترها زیاد می‌باشد، از همین‌رو محدودیت تعداد پارامترها را خواهد داشت.

۳.۲.۴.۲ جستجوی تصادفی[۲]

یک روش ساده برای جایگزینی جستجوی شبکه‌ای، نمونه‌گیری از فضای ابرپارامترها به‌صورت تصادفی است. به عبارت دیگر، به جای آزمایش‌های مرتب در تمام مجموعه مقادیر در فضای مسئله، بهتر است مقادیر تصادفی از کل فضای نمونه انتخاب و آزمایش شوند. به‌طور تجربی و نظری سال ۲۰۱۲ برگسترا و بنجیو در مقاله‌ای با عنوان

---

[۱] Grid Search
[۲] Random Search

"جستجوی تصادفی برای بهینه‌سازی" نشان دادند که استفاده از جستجوی تصادفی برای بهینه‌سازی ابرپارامترها از جستجوی شبکه‌ای موثرتر است.

مزایا:

- نگرانی در مورد زمان اجرا وجود ندارد، چرا که می‌تواند تعداد جستجوی پارامترها را کنترل کند.

معایب:

- بسته به تعداد جستجوها و میزان بزرگی فضای پارامترها، برخی پارامترها ممکن است کاوش نشوند.

۴.۲.۴.۲ بهینه‌سازی بیزی[۱]

مفهوم اصلی بهینه‌سازی بیزی این است که: "**اگر به‌طور تصادفی برخی از نقاط را جستجو کردیم و بدانیم که برخی از این نقاط امیدوارکننده‌تر از سایرین هستند، چرا نگاهی به آن‌ها نیاندازیم؟**"

بهینه‌سازی بیزی، ارزیابی‌های گذشته را هنگام انتخاب مجموعه ابرپارامترها برای ارزیابی بعدی در نظر می‌گیرد. با انتخاب ترکیبات ابرپارمترها به روشی آگاهانه، این امکان را برای خود فراهم می‌سازد تا روی مناطقی از فضای ابرپارامترها که معتقد است، بیشترین امتیاز ممکن را دارند متمرکز می‌شود. این رویکرد به‌طور معمول در رسیدن به مجموعه مطلوب مقادیر ابرپارامترها به تکرار کم‌تری نیاز دارد، چرا که مناطقی از فضای پارامترهایی که به عقیده وی هیچ کمکی نمی‌کند را نادیده می‌گیرد

مزایا:

- نه لزوما اما، جستجو به‌طور بالقوه کارآمد است

معایب:

- امکان به دام افتادن در بهینه محلی را دارد.

[۱] Bayesian Optimization

۵.۲ شبکه‌های عصبی بازگشتی(RNN)[۱]

هنگامی که داده‌ها به‌گونه‌ای تنظیم شده باشند که هر قطعه به نوعی رابطه‌ای با قطعاتی که قبل و بعد از آن ایجاد می‌شوند، داشته باشد از آن‌ها به عنوان دنباله یاد می‌شود. داده‌های دنباله‌دار جالبی در جهان وجود دارد، همانند: قیمت سهام در چند روز اخیر، فریم‌های ساخت یک فیلم یا کلماتی از یک قطعه زبان گفتاری یا نوشتاری. این طبیعی است که بخواهیم درباره این دنباله از داده‌ها سوال‌هایی همانند: آیا این دنباله‌ها مانند یک دنباله دیگر هستند (به عنوان مثال: آیا این کتاب توسط همان نویسنده کتاب دیگر نوشته شده است؟)، چگونه می‌توان آن‌ها را با اصطلاحات دیگری بیان کرد (برای مثال: ترجمه رشته‌های کلمات به زبانی دیگر) و یا این‌که چگونه در آینده رفتار می‌کنند (برای مثال: قیمت سهام فردا چه خواهد بود؟).

شبکه‌های عصبی مورد بحث تا اکنون می‌توانند پاسخ درست به این پرسش‌ها را دهند، اما مشکلی در این شبکه‌ها وجود دارد و آن نداشتن حافظه می‌باشد. این نقص سبب ضعف استفاده از این شبکه‌ها برای این نوع ساختار داده می‌شود. جهت جبران این نقص می‌توان با جایگزینی نرون مصنوعی با پردازشی پیچیده‌تر به نام واحد بازگشتی، این کمبود حافظه را جبران کنیم. ما می‌توانیم با ترکیب لایه‌های استاندارد و لایه‌هایی که از واحدهای بازگشتی تشکیل شده هستند، شبکه‌های یادگیری عمیق با عنوان شبکه‌ی عصبی بازگشتی یا به اختصار RNN بسازیم. RNNها می‌توانند به تمامی سوالات بالا و بسیاری پرسش‌های دیگر پاسخ دهند. از آن‌ها می‌توان در ترجمه زبان گرفته تا شرح‌نویسی خودکار تصاویر و حتی تولید نثر جدید به سبک نویسندگان شناخته شده استفاده کرد.

۱.۵.۲ ساختار یک شبکه عصبی بازگشتی ساده

شبکه‌های عصبی بازگشتی یا همان RNN نوعی از شبکه‌های عصبی هستند، که می‌تواند داده‌های دنباله‌دار با طول متغیر را پردازش کند. نمونه‌ای از چنین داده‌هایی شامل کلمات یک جمله یا قیمت سهام در لحظه‌های مختلف از زمان می‌باشد. RNN

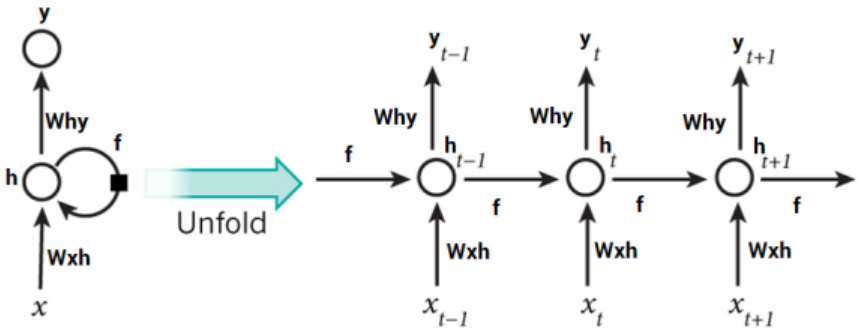
[۱] Recurrent Neural Network

را می‌توان به عنوان یک گرافی از واحدهای RNN تصور کرد، به‌گونه‌ای که هر عنصر دنباله یک عمل مشابه را انجام می‌دهد. همین عملکرد مکرر در یک دنباله سبب نام گذاری آن به شبکه عصبی بازگشتی شده است. شبکه‌های عصبی سنتی بر این فرض استوار هستند که همه ورودی‌ها از یک‌دیگر مستقل هستند و این فرض را برای داده‌های دنباله‌دار نیز استفاده می‌کنند. واحدهای RNN این وابستگی‌ها را با داشتن حالت پنهان یا همان حافظه، آنچه را که تاکنون دیده شده در خود جای می‌دهند.

RNN به عنوان یک رابطه بازگشتی به صورت زیر قابل تعریف می‌باشد:

$$h_t = f(h_{t-1}, x_t)$$

که در این رابطه h_t برداری از مقادیر به نام حالت شبکه داخلی در زمان t و x_t ورودی شبکه در زمان t است. برخلاف شبکه‌های معمولی که در آن حالت فقط به ورودی فعلی (و وزن‌های شبکه) بستگی دارند، در این شبکه‌ها هم به ورودی فعلی و هم به حالت قبلی بستگی دارد. می‌توان h_{t-1} را به عنوان خلاصه‌ای از ورودی‌های قبلی شبکه تصور کرد. رابطه بازگشتی چگونگی حالت تکامل را به‌صورت گام به گام بر روی دنباله از طریق یک حلقه بازخورد نسبت به حالت‌های قبلی در شکل ۲ ـ ۱۹ نشان می‌دهد.



شکل ۲ ـ ۱۹ شبکه عصبی بازگشتی ساده

RNN دارای سه مجموعه پارامتر است:

- U (w_{xh}) ورودی x_t را به حالت h_t تبدیل می‌کند.
- W (w_{hh}) حالت قبلی s_{t-1} را به حالت کنونی h_t تبدیل می‌کند.

- V (w_{yh}) وضعیت داخلی تازه محاسبه شده را به خروجی y_t نگاشت می‌کند.

تبدیل‌های خطی بر روی ورودی‌های مربوطه با استفاده W،U و V انجام می‌شود. بر این اساس می‌توان وضعیت داخلی و خروجی شبکه را به صورت زیر تعریف کرد:

$$h_t = f(h_{t-1} * W + x_t * U)$$

$$o_t = h_t * V + b_y$$

$$y_t = f(o_t)$$

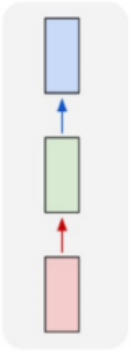
که در این رابطه f یک تابع فعال‌ساز غیرخطی می‌باشد.

همان‌طور که گفته شد در RNN، هر حالت به تمام محاسبات قبلی توسط معادله بازگشتی وابسته است. پیامد مهم این امر سبب ایجاد حافظه در طول زمان می‌شود، چرا که حالت‌ها مبتنی بر مراحل قبلی هستند. از منظر تئوری، RNNها می‌توانند اطلاعات را برای مدت طولانی در خود به خاطر بسپارند، اما در عمل فقط به چند مرحله نگاه می‌کنند.

۲.۵.۲ انواع معماری شبکه عصبی بازگشتی

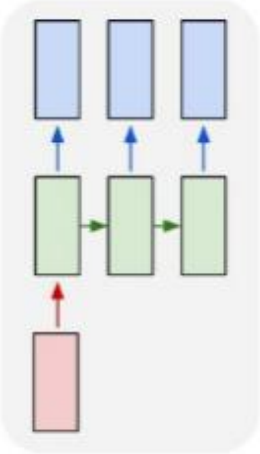
به‌طور معمول، RNNها معماری‌های متفاوتی را شامل می‌شوند. در این بخش، به برخی از معماری‌های اساسی RNN می‌پردازیم:

- **یک به یک:** همان‌طور که در شکل ۲ ـ ۲۰ قابل مشاهده است، در این معماری یک واحد ورودی RNN به یک واحد پنهان و یک واحد خروجی نگاشت شده است. این معماری یک پردازش بدون متوالی همانند، شبکه‌های عصبی پیش‌خور و شبکه‌ی عصبی همگشتی می‌باشد. نمونه‌ای از این پردازش دسته‌بندی تصاویر می‌باشد.



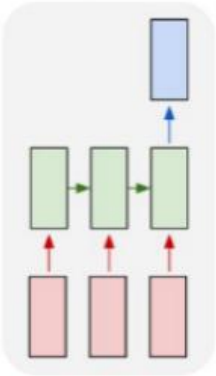
شکل ۲ـ ۲۰ RNN یک به یک

- **یک به چند:** همان‌طور که در شکل ۳ـ ۲۱ قابل مشاهده است، در این معماری یک واحد ورودی RNN به چند واحد پنهان و چند واحد خروجی نگاشت شده است. نمونه کابردی از این معماری شرح‌نویسی تصاویر می‌باشد. لایه ورودی یک تصویر را دریافت کرده و آن را به چندین کلمه نگاشت می‌کند.



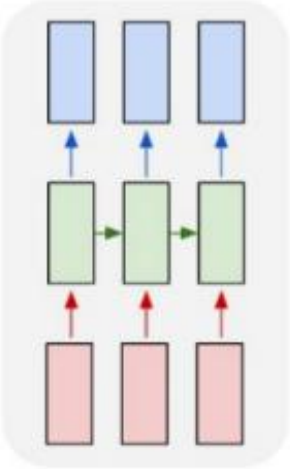
شکل ۲ـ ۲۱ RNN یک به چند

- **چند به یک:** همان‌طور که در شکل ۲ـ ۲۲ قابل مشاهده است، در این معماری چند واحد ورودی RNN به چند واحد پنهان و یک واحد خروجی نگاشت شده است. یک نمونه کاربردی از این معماری دسته‌بندی احساسات می‌باشد. لایه ورودی چندین نشانه از کلمات یک جمله را دریافت، و به صورت یک احساس مثبت یا منفی نگاشت می‌کند.



شکل ۲ـ۲۲ RNN چند به یک

- **چند به چند:** همان‌طور که در شکل ۲ ـ۲۳ قابل مشاهده است، در این معماری چند واحد ورودی RNN به چند واحد پنهان و چند واحد خروجی نگاشت شده است. نمونه کاربردی از این معماری ترجمه ماشینی است. لایه ورودی چندین نشانه از کلمات زبان مبدا را دریافت، و آن‌ها را به نشانه‌هایی از کلمات به زبان هدف نگاشت می‌کند.



شکل ۲ـ۲۳ RNN چند به چند

۳.۵.۲ آموزش شبکه عصبی بازگشتی

آموزش شبکه‌های عصبی بازگشتی اغلب با استفاده از الگوریتم گرادیان کاهشی تصادفی ریز_دسته‌ای انجام می‌گیرد. همان‌طور که در بخش الگوریتم‌های بهینه‌سازی بیان شد، این الگوریتم‌ها برای محاسبه گردایان و بروزرسانی وزن‌ها، زیرمجموعه‌ای تصادفی از نمونه‌های آموزشی را انتخاب می‌کنند. با در نظر گرفتن نمونه‌های کوچک‌تر از نمونه‌ها، آموزش پایدارتر و سازگارتری را نسبت به گرادیان که فقط با یک نمونه بروزرسانی می‌شود را داراست. همچنین، کارآیی بیشتری نسبت به بروزرسانی با تمام نمونه‌های آموزشی دارد.

۴.۵.۲ پس‌انتشار در طول زمان

شبکه‌های عصبی بازگشتی با نوعی خاصی از الگوریتم پس‌انتشار، به نام پس‌انتشار در طول زمان[۱] آموزش داده می‌شوند. همانند الگوریتم پس‌انتشار در شبکه عصبی پیش‌خور این الگوریتم نیز برای محاسبه گرادیان از قانون زنجیره‌ای استفاده می‌کند. پس‌انتشار در شبکه‌های عصبی بازگشتی به دلیل خاصیت بازگشتی وزن‌ها و از بین رفتن آن‌ها با گذشت زمان، کمی چالش برانگیزتر است. چرا که نیاز است گراف محاسباتی یک RNN را یکبار گسترش داده تا وابستگی‌ها را بین متغیرها و پارامترهای مدل بدست آوریم. سپس، از پس‌انتشار و با استفاده از قانون زنجیره‌ای محاسبه گرادیان‌ها و ذخیره گرادیان‌ها انجام گیرد. از آنجایی که توالی‌ها می‌توانند طولانی باشند، بنابراین وابستگی می‌تواند طولانی باشد.

به عنوان مثال، برای دنباله‌ای از ۱۰۰۰ نویسه، اولین نویسه به طور بالقوه می‌تواند تأثیر قابل توجهی در نویسه در مکان نهایی داشته باشد. این امر از نظر محاسباتی واقعاً امکان پذیر نیست. چرا که بیش از حد طول می‌کشد و به حافظه زیادی نیاز دارد. این فرایند مملو از عدم قطعیت محاسباتی و آماری است.

[۱] Backpropagation Through Time (BPTT)

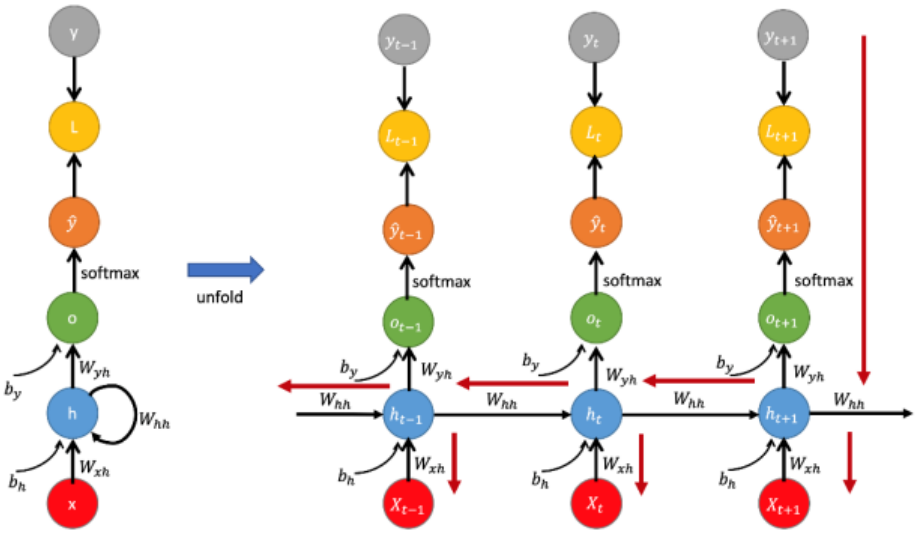
در ادامه، توضیح خواهیم داد که چگونه در عمل این مساله حل می‌شود. اما قبل از آن به تشریح پس‌انتشار در طول زمان به‌صورت ریاضی می‌پردازیم. برای درک این‌که این رویکرد چگونه عمل می‌کند نمایی از نحوه جریان اطلاعات را در تصویر ۲ــ۲٤ قابل مشاهده است.

برای این‌که بتوان از پس‌انتشار در طول زمان در فرآیند آموزش شبکه عصبی بازگشتی استفاده شود، ابتدا باید تابع زیان محاسبه شود:

$$L(\hat{y}, y) = \sum_{t=1}^{T} L_t(\hat{y}_t, y_t)$$

$$= -\sum_{t}^{T} y_t log\, \hat{y}_t$$

$$= -\sum_{t=1}^{T} y_t log\, [softmax(o_t)]$$



شکل ۲ــ۲۴ پس‌انتشار در شبکه عصبی بازگشتی ساده

از آنجایی که وزن W_{yh} در تمام توالی زمان تقسیم می‌شود. از همین‌رو، می‌توانیم در هر مرحله از آن مشتق گرفته و همه را با هم جمع کرد:

$$\frac{\partial L}{\partial w_{yh}} = \sum_{t}^{T} \frac{\partial L_t}{\partial w_{yh}}$$
$$= \sum_{t}^{T} \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial o_t} \frac{\partial o_t}{\partial w_{yh}}$$
$$= \sum_{t}^{T} (\hat{y} - y_t) \otimes h_t$$

که در این معادله $\frac{\partial o_t}{\partial w_{yh}} = h_t$ و $\otimes$ ضرب خارجی دو بردار می‌باشد.

به همین ترتیب، می‌توان گرادیان بایاس b_y را بدست آوریم:

$$\frac{\partial L}{\partial b_y} = \sum_{t}^{T} \frac{\partial L_t}{\partial \hat{y}_t} \frac{\partial \hat{y}_t}{\partial o_t} \frac{\partial o_t}{\partial b_y}$$
$$= \sum_{t}^{T} (\hat{y} - y_t)$$

به‌علاوه بیاید از L_{t+1} برای نشان دادن خروجی مرحله زمانی $t+1$ استفاده کنیم:

$$L_{t+1} = -y_{t+1} log \hat{y}_{t+1}$$

حال، جزئیات مربوط به گرادیان w_{hh} را با توجه به زمان $t+1$ را مرور می‌کنیم:

$$\frac{\partial L_{t+1}}{\partial w_{hh}} = \frac{\partial L_{t+1}}{\partial \hat{y}_{t+1}} \frac{\partial \hat{y}_{t+1}}{\partial h_{t+1}} \frac{\partial h_{t+1}}{\partial w_{hh}}$$

از آن‌جایی که حالت پنهان h_{t+1} با توجه به معادله بازگشتی h_t:

$$h_t = \tanh(w_{xh}^T . x_t + w_{hh}^T . h_{t-1} + b_h)$$

نیز بستگی دارد. بنابراین، در مرحله زمانی $t \to t-1$ می‌توان مشتق جزئی را با توجه به w_{hh} به‌صورت زیر بدست آورد:

$$\frac{\partial L_{t+1}}{\partial w_{hh}} = \frac{\partial L_{t+1}}{\partial \hat{y}_{t+1}} \frac{\partial \hat{y}_{t+1}}{\partial h_{t+1}} \frac{\partial h_{t+1}}{\partial h_t} \frac{\partial h_t}{\partial w_{hh}}$$

بنابراین، در مرحله زمانی $t+۱$، می‌توانیم گرادیان را محاسبه کرده و از طریق پس‌انتشار در طول زمان از $t+۱$ به t استفاده می‌کنیم تا محاسبه گرادیان کلی را با توجه به w_{hh} بدست آوریم:

$$\frac{\partial L_{t+۱}}{\partial w_{hh}} = \sum_{k=۱}^{t+۱} \frac{\partial L_{t+۱}}{\partial \hat{y}_{t+۱}} \frac{\partial \hat{y}_{t+۱}}{\partial h_{t+۱}} \frac{\partial h_{t+۱}}{\partial h_k} \frac{\partial h_k}{\partial w_{hh}}$$

توجه داشته باشید که $\frac{\partial h_{t+۱}}{\partial h_k}$ خود یک قانون زنجیره‌ای است. به عنوان مثال:

$$\frac{\partial h_۳}{\partial h_۱} = \frac{\partial h_۳}{\partial h_۲} \frac{\partial h_۲}{\partial h_۱}$$

همچنین، توجه داشته باشید که چون مشتق یک تابع را بردار در نظر می‌گیریم، نتیجه یک ماتریس است (ماتریس ژاکوبین•) که همه عناصر آن مشتقات جزئی هستند. می‌توانیم گرادیان فوق را دوباره بازنویسی کنیم:

$$\frac{\partial L_{t+۱}}{\partial w_{hh}} = \sum_{k=۱}^{t+۱} \frac{\partial L_{t+۱}}{\partial \hat{y}_{t+۱}} \frac{\partial \hat{y}_{t+۱}}{\partial h_{t+۱}} \left(\prod_{j=k}^{t} \frac{\partial h_{j+۱}}{\partial h_j} \right) \frac{\partial h_k}{\partial w_{hh}}$$

جایی که:

$$\prod_{j=k}^{t} \frac{\partial h_{j+۱}}{\partial h_j} = \frac{\partial h_{t+۱}}{\partial h_k} = \frac{\partial h_{t+۱}}{\partial h_t} \frac{\partial h_t}{\partial h_{t-۱}} \dots \frac{\partial h_{k+۱}}{\partial h_k}$$

گرادیان‌های را با توجه به w_{hh} در کل مراحل پس‌انتشار جمع می‌شود و در نهایت می‌توانیم گرادیان زیر را با توجه به w_{hh} بدست آورد:

• با توجه به تابعی از نگاشت $n-$ بعدی بردار x به یک بردار خروجی $m-$ بعدی، $\boldsymbol{f}: \mathbb{R}^n \to \mathbb{R}^m$ ، ماتریس تمام مشتقات جزئی درجه یک این تابع را ماتریس ژاکوبین ($\boldsymbol{J}$) گویند:

$$\boldsymbol{J} = \begin{bmatrix} \frac{\partial f_۱}{\partial x_۱} & \cdots & \frac{\partial f_۱}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_۱} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

$$\frac{\partial L}{\partial w_{hh}} = \sum_{t}^{T} \sum_{k=1}^{t+1} \frac{\partial L_{t+1}}{\partial \hat{y}_{t+1}} \frac{\partial \hat{y}_{t+1}}{\partial h_{t+1}} \frac{\partial h_{t+1}}{\partial h_k} \frac{\partial h_k}{\partial w_{hh}}$$

اکنون بیایید تا گرادیان را با توجه به w_{xh} استخراج کنیم. به‌طور مشابه، مرحله زمانی $t+1$ را در نظر می‌گیریم و گرادیان را با توجه w_{xh} به‌صورت زیر بدست می‌آوریم:

$$\frac{\partial L_{t+1}}{\partial w_{xh}} = \frac{\partial L_{t+1}}{\partial \hat{y}_{t+1}} \frac{\partial \hat{y}_{t+1}}{\partial h_{t+1}} \frac{\partial h_{t+1}}{\partial w_{xh}}$$

از آن‌جا که h_t و x_{t+1} هر دو در h_{t+1} مشارکت دارند، بنابراین برای پس‌انتشار به h_t نیاز داریم. اگر این مشارکت را در نظر بگیریم، خواهیم داشت:

$$\frac{\partial L_{t+1}}{\partial w_{xh}} = \frac{\partial L_{t+1}}{\partial \hat{y}_{t+1}} \frac{\partial \hat{y}_{t+1}}{\partial h_{t+1}} \frac{\partial h_{t+1}}{\partial w_{xh}} + \frac{\partial L_{t+1}}{\partial \hat{y}_{t+1}} \frac{\partial \hat{y}_{t+1}}{\partial h_t} \frac{\partial h_t}{\partial w_{xh}}$$

بنابراین، با جمع همه مشارکت‌ها از $t+1$ به t از طریق پس‌انتشار، می‌توان گرادیان را در مرحه زمانی $t+1$ بدست آوریم:

$$\frac{\partial L_{t+1}}{\partial w_{xh}} = \sum_{k=1}^{t+1} \frac{\partial L_{t+1}}{\partial \hat{y}_{t+1}} \frac{\partial \hat{y}_{t+1}}{\partial h_{t+1}} \frac{\partial h_{t+1}}{\partial h_k} \frac{\partial h_k}{\partial w_{xh}}$$

علاوه بر این، می‌توانیم مشتق را با توجه به w_{xh} در کل دنباله در نظر بگیریم:

$$\frac{\partial L}{\partial w_{xh}} = \sum_{t}^{T} \sum_{k=1}^{t+1} \frac{\partial L_{t+1}}{\partial \hat{y}_{t+1}} \frac{\partial \hat{y}_{t+1}}{\partial h_{t+1}} \frac{\partial h_{t+1}}{\partial h_k} \frac{\partial h_k}{\partial w_{xh}}$$

همچنین فراموش نشود که $\frac{\partial h_{t+1}}{\partial h_k}$ خود یک قانون زنجیره‌ای است.

همان‌طور که بیان شد مشکلات محو و انفجار گرادیان در شبکه عصبی بازگشتی معمولی وجود دارد. به‌طور کلی دو عامل وجود دارد که بر میزان گرادیان‌ها تاثیر گذار می‌باشد: **وزن‌ها و توابع فعال‌سازی یا به‌طور دقیق‌تر، مشتقات آن‌ها** که گرادیان از آن‌ها عبور می‌کند. در شبکه عصبی بازگشتی معمولی، محو و انفجار گرادیان از از اتصالات

بازگشتی (مکرر) ناشی می‌شود. واضح‌تر، این دو این مشکل به دلیل مشتق بازگشتی $\frac{\partial h_{t+1}}{\partial h_k}$ است که در معادله w_{xh} اتفاق می‌افتد وباید محاسبه شود:

$$\prod_{j=k}^{t} \frac{\partial h_{j+1}}{\partial h_j} = \frac{\partial h_{t+1}}{\partial h_k} = \frac{\partial h_{t+1}}{\partial h_t} \frac{\partial h_t}{\partial h_{t-1}} \dots \frac{\partial h_{k+1}}{\partial h_k}$$

و نمایانگر ضرب ماتریس روی دنباله می‌باشد.

از آن‌جایی که شبکه عصبی بازگشتی معمولی نیاز دارد تا گرادیان پس‌انتشار را در یک توالی طولانی (با مقادیر کوچک در ضرب ماتریس) بدست آورد، از همین رو مقدار گرادیان لایه به لایه کاهش می‌یابد و در نهایت پس از طی چند مرحله از بین می‌رود. بنابراین، حالاتی که از مرحله زمانی فعلی فاصله دارند، به محاسبه پارامترهای گرادیان که همان پارامترهای یادگیری در شبکه عصبی بازگشتی هستند، هیچ کمکی نخواهند کرد.

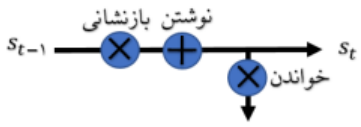
محو گرادیان منحصر به شبکه عصبی بازگشتی معمولی نیست. همان‌طور که در فصل دوم بیان شد، آن‌ها در شبکه‌های پیش‌خور نیز اتفاق می‌افتند. نکته فقط اینجاست که، شبکه عصبی بازگشتی به دلیل این‌که عمق زیادی دارد این مشکلات در آن رایج‌تر می‌باشد. این دو مشکل در نهایت نشان می‌دهند که اگر گرادیان از بین برود، به این معنا خواهد بود که حالت‌های پنهان قبلی هیچ تاثیر واقعی بر حالت‌های پنهان بعدی ندارند. به عبارت دیگر، هیچ وابستگی طولانی مدتی آموخته نمی‌شود. خوشبختانه، چندین روش برای رفع مشکل محو گرادیان وجود دارد. مقداردهی اولیه مناسب ماتریس‌های وزنی می‌تواند اثر شیب‌های محو شده را کاهش دهد. منظم سازی نیز می‌تواند کمک کننده باشد. راه حل دیگری که از دو مورد قبل بیشتر ترجیح داده می‌شود، استفاده از تابع فعال‌سازی ReLU به‌جای توابع فعال‌سازی تانژانت هذلولوی یا سیگموئید می‌باشد. مشتق ReLU یک ثابت ۰ یا ۱ است، بنابراین به احتمال زیاد مشکل محو گرادیان را ندارد. راه حل محبوب‌تر که امروزه بیشتر مورد استفاده می‌شود، استفاده از شبکه‌های حافظه طولانی کوتاه‌ـ مدت (LSTM) یا واحد بازگشتی دروازه‌دار (GRU) است.

۶.۲ حافظه طولانی کوتاه-مدت(LSTM)[۱]

شبکه‌های حافظه طولانی کوتاه ـ مدت که به اختصار از آن به عنوان LSTM یاد می‌شود، نوعی خاصی از RNN می‌باشند. این نوع از شبکه‌های عصبی در جهت یادگیری وابستگی‌های بلند مدت توسط هوچریتر[۲] و اشمیتبر[۳] در سال ۱۹۹۷ طراحی شدند. LSTMها به دلیل داشتن یک سلول حافظه خاص، می‌توانند مشکل وابستگی‌های بلند مدت را برطرف کنند. ایده اصلی LSTM یک منطق دروازه‌دار است، که یک معماری مبتنی‌بر حافظه را امکان‌پذیر می‌سازد. جهت توصیف بیشتر این مفهوم، به معماری حافظه LSTM می‌پردازیم. همانند هر سیستم مبتنی‌بر حافظه، یک سلول معمولی LSTM از سه ویژگی اصلی تشکیل شده است:

۱. نوشتن در حافظه
۲. خواندن از حافظه
۳. بازنشانی حافظه

در شکل ۲ ـ ۲۵ این ایده نمایش داده شده است. ابتدا مقدار یک سلول LSTM قبلی از طریق یک دروازه بازنشانی منتقل می‌شود، که مقدار حالت قبلی را در بازه ۰ تا ۱ ضرب اسکالر می‌کند. اگر حاصل به‌دست آمده نزیک به ۱ باشد، منجر به عبور مقدار حالت سلول قبلی(به‌یاد آوردن حالت قبلی) می‌شود. در صورت نزدیکی این مقدار به صفر، منجربه مسدود شدن حالت سلول قبلی(فراموش کردن حالت قبلی) می‌شود. در مرحله بعدی، دروازه نوشتن به سادگی خروجی تغییر یافته دروازه بازنشانی را مجدد بازنویسی می‌کند. سرانجام، دروازه خواندن، خروجی دروازه نوشتن را می‌خواند.

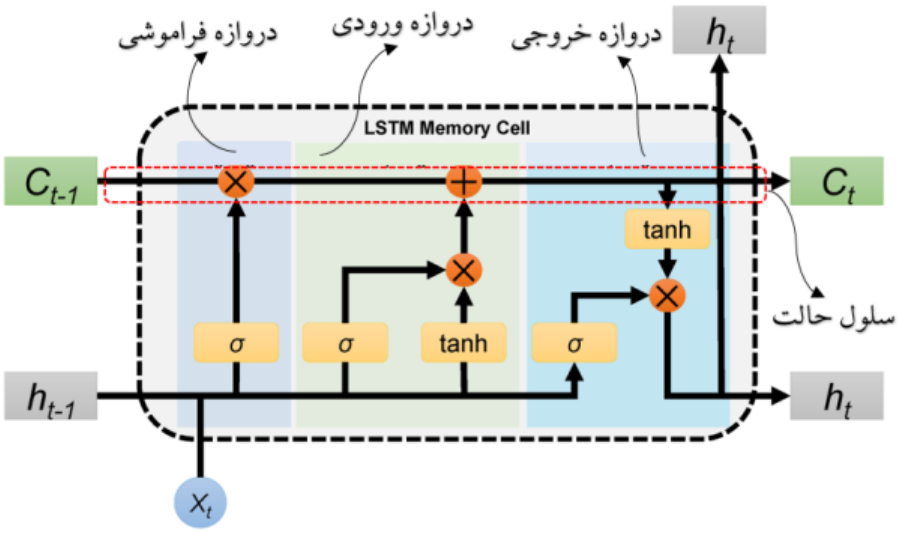


شکل ۲ ـ ۲۵ ایده اصلی LSTM

[۱] Long short term memory
[۲] Hochreiter
[۳] Schmidhuber

همان‌طور که بیان شد LSTM توانایی حذف یا افزودن اطلاعات به سلول حالت را توسط ساختار دروازه امکان‌پذیر می‌سازد. در ساختار یک LSTM که شمایی از آن را در شکل ۲ـ۲۶ مشاهده می‌کنید، سه نوع دروازه وجود دارد:

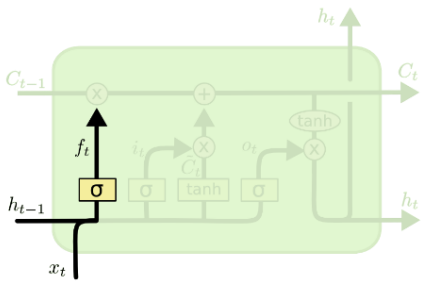
- **دروازه فراموشی:** در این دروازه یک تابع فعال‌ساز سیگموید بر مقادیر ورودی از سلول حالت قبلی اعمال می‌شود. از آنجا که تابع سیگموید هر مقدار بین ۰ تا ۱ را می‌گیرد، این دروازه به معنای فراموش کردن چه مقدار از مقدار سلول حالت قبلی را امکان پذیر می‌سازد. به عبارتی دیگر این دروازه کمک می‌کند تا محتوای گذشته فراموش شود.
- **دروازه ورودی:** دروازه ورودی محافظت در برابر ورودی‌های نامربوط را بر عهده دارد.
- **دروازه خروجی:** این دروازه با استفاده از یک تابع سیگموید خروجی در زمان t در h_t را تولید می‌کند. به عبارتی دیگر دروازه خروجی در نمایش یا عدم نمایش محتویات درون سلول نقش دارد.



شکل ۲ـ۲۶ ساختار LSTM

اکنون به بررسی قدم به قدم نحوه کار LSTM می‌پردازیم:

اولین قدم در LSTM این است که تصمیم بگیرم چه اطلاعاتی را از سلول حالت دور کنیم. این تصمیم توسط یک لایه سیگموید به نام **لایه دروازه فراموشی** که در شکل ۲ ـ ۲۷ قابل مشاهده است، گرفته می‌شود.



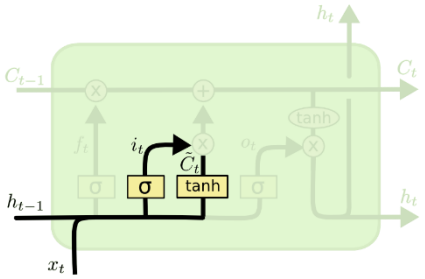
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

شکل ۲ ـ ۲۷ لایه دروازه فراموشی

ورودی‌های h_{t-1} و x_t فرآیند حذف یا ماندن اطلاعات از سلول حالت را انجام می‌دهند. اگر مقدار بدست آمده ۱ باشد، سلول حالت c_{t-1} به‌طور کامل به c_t انتقال داده می‌شود، و مقدار ۰ به معنای آن خواهد بود که هیچ چیزی از c_{t-1} انتقال پیدا نخواهد کرد. به عنوان مثال، در مدل سازی یک زبان که سعی در پیش‌بینی کلمه بعدی براساس همه کلمات قبل از خود دارد، اگر سلول حالت شامل جنسیت فاعل کنونی باشد، باید براساس آن تصمیم‌گیری برای انتخاب ضمیر را انجام دهد. از همین‌رو اگر فاعل جدیدی، ظاهر گشت، لازم است تا جنسیت فاعلی قبلی حذف گردد.

قدم بعدی تصمیم‌گیری در جهت این است که چه اطلاعات جدیدی در سلول حالت ذخیره شوند. این مرحله در شکل ۲ ـ ۲۸ قابل مشاهده بوده و شامل دو بخش می‌باشد.

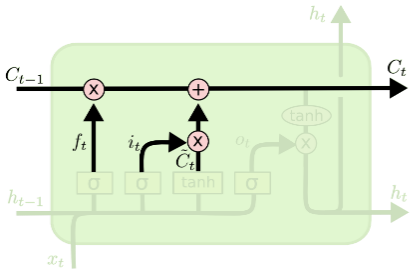
ابتدا یک لایه سیگموید به نام **لایه دروازه ورودی** تصمیم می‌گیرد که کدام مقادیر را بروزرسانی کنیم، سپس لایه تانژانت هذلولوی، یک بردار از مقادیر جدید نامزد c_t را ایجاد می‌کند. در انتها این دو باهم ترکیب شده تا سلول حالت بروز شود.



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

شکل ۲ـ۲۸ لایه دروازه ورودی

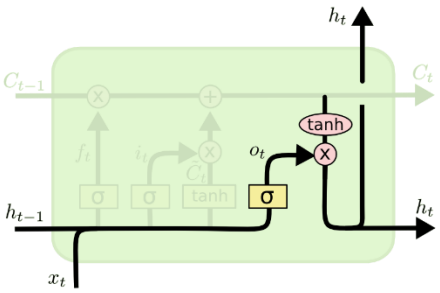
در مثال قبلی سعی در جایگزینی جنسیت فاعلی جدید به سلول حالت با جنسیت فاعلی گذشته بود. در جهت این کار سلول حالت قدیمی c_{t-1} با c_t بروزرسانی می‌شود، مطابق شکل ۲ـ۲۹ جهت فراموشی اطلاعات قبلی، مقدار قبلی سلول حالت را در f_t ضرب کرده، و پس از آن $i_t * C_t$ به آن اضافه می‌شود. این مرحله برای مثال ما، همان جایی است اطلاعات جنسیت فاعلی گذشته دور ریخته شده، و اطلاعات جدید به سلول حالت اضافه می‌شوند.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

شکل ۲ـ۲۹ بروزرسانی اطلاعات

سرانجام باید تصمیم گرفته شود که چه چیزی در خروجی باشد. این خروجی براساس سلول حالت با یک نسخه فیلترداری بدست می‌آید. در ابتدا، یک لایه سیگوید اجرا شده تا تصمیم بگیرد چه قسمت‌هایی از سلول حالت به خروجی فرستاده شوند، سپس مقدار سلول حالت را به یک لایه تانژانت هذلولوی می‌دهیم تا در نهایت مقدار آن را در خروجی لایه قبلی سیگموید ضرب کرده، تا قسمت‌های مورد نظر در خروجی به اشتراک گذاشته شوند. در شکل ۲ـ۳۰ نحوه انجام این عمل نمایش داده شده است.



$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

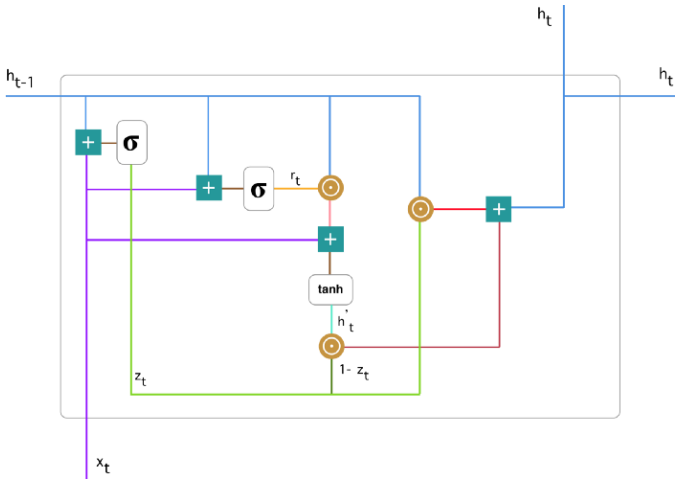
شکل ۲ـ ۳۰ دروازه خروجی

۷.۲ واحد بازگشتی دروازه‌دار(GRU)[۱]

در تلاش برای ساده‌سازی LSTM در سال ۲۰۱٤ واحد بازگشتی دروازه‌دار (GRU) معرفی شد. GRU در جهت حل مشکل محو گرادیان در RNN استاندارد در مقایسه با LSTM که دارای سه دروازه می‌باشد، از دو دروازه به نام‌های: دروازه **بروزرسانی** و **بازنشانی** استفاده می‌کند. این دو دروازه تصمیم‌گیرنده این هستند که چه اطلاعاتی باید در خروجی به نمایش گذاشته شوند.

برخلاف LSTM که به‌طور مستقیم از دروازه فراموشی و خروجی برای کنترل میزان تغییر اطلاعات در وضعیت پنهان استفاده می‌کند، واحد GRU همان هدف را تنها با دروازه بازنشانی انجام می‌دهد. اگر چه GRU بسیار مشابه با LSTM است، اما نباید آن را به عنوان مورد خاصی از LSTM قلمداد کرد. وجود تعداد پارامتر کم‌تر در واحد GRU، منجر به پیچیدگی محاسبات کم‌تر شد و نشان داده شده که GRU در بیشتر کارها می‌تواند از LSTM بهتر یا مشابه آن عمل کند. البته به نظر می‌رسد عملکرد نسبی به‌کار انجام شده بستگی دارد. GRU در مواقعی که داده‌های آموزشی کم‌تری در دسترس باشد، به دلیل تعداد پارمترهای کم‌تر مفیدتر است. اگر چه با افزایش تعداد داده‌های آموزشی LSTM ترجیح داده می‌شود. شمایی از یک سلول GRU در شکل ۲ـ ۳۱ قابل مشاهده است.

[۱] Gated recurrent unit



شکل ۲ـ۳۱ دروازه خروجی

در ادامه ریاضیات پشت فرآیند یک واحد GRU را مرحله‌به‌مرحله شرح می‌دهیم:

۱. **دروازه بروزرسانی:** در ابتدا دروازه بروزرسانی z_t در زمان t با استفاده از معادله زیر محاسبه می‌شود:

$$z_t = \sigma(w_z x_t + U_z h_{t-1})$$

دروازه بروزرسانی، براساس ورودی x_t و حالت پنهان گذشته h_{t-1} تصمیم می‌گیرد که چه اطلاعاتی باید دور ریخته و چه اطلاعات جدیدی را باید در خود جای دهد. هنگامی که x_t به شبکه وصل می‌شود، در وزن خود (w_z) ضرب می‌شود. h_{t-1} نیز در وزن خودش (U_z) ضرب شده و این دو باهم جمع می‌شوند. سپس تابع فعال‌ساز سیگموید نتیجه بین ۰ تا ۱ را تولید می‌کند. دروازه بروزرسانی به مدل کمک می‌کند تا باید چه مقدار از اطلاعات در آینده منتقل شوند. این عمل سبب جلوگیری از مشکل محو گرادیان می‌شود. چراکه مدل می‌تواند تصمیم بگیرد تا رونوشتی از همه اطلاعات گذشته داشته باشد.

۲. **دروازه بازنشانی:** دروازه بازنشانی r_t، از سلول حالت h_{t-1} و ورودی x_t جهت تصمیم‌گیری این‌که چه مقدار از اطلاعات گذشته باید فراموش شوند، استفاده می‌کند. این مرحله توسط معادله زیر محاسبه می‌شود:

$$r_t = \sigma(w_r x_t + U_r h_{t-۱})$$

۳. **محتوای حافظه فعلی:** اکنون نوبت به این می‌رسد تا ببینیم که دروازه‌ها چگونه خروجی را تعیین می‌کنند. برای این کار ابتدا یک حافظه جدید که دروازه بازنشانی از آن جهت ذخیره‌سازی اطلاعات مربوط به گذشته استفاده می‌کند، به صورت معادله زیر محاسبه می‌شود:

$$\widetilde{h_t} = \tanh\left(W x_t + r_t \bigodot U h_{t-۱}\right)$$

با این کار مشخص می‌شود که چه چیزی از مرحله قبلی حذف شود. برای مثال، فرض کنید در تحلیل احساسات نظری در مورد یک کتاب، جمله : " این یک کتاب تاریخی است که ..." در ابتدای پارگراف آمده و در چند پاراگراف بعد، جمله‌ی: "از این کتاب خوشم نیامد، جزئیات کاملی نداشت." آمده است. حال برای تعیین احساسات این نظر تنها به جملات انتهایی متن نیاز داریم. از همین رو، می‌توان بردار r_t را نزدیک به صفر تعیین کرد. این عمل سبب می‌شود اطلاعات گذشته پاک شوند و فقط روی جملات آخر تمرکز شود.

٤. **حافظه نهایی:** در آخرین مرحله، شبکه با محاسبه بردار $h_{t-۱}$ تصمیم می‌گیرد که اطلاعات مربوط به حافظه فعلی را در خود نگه داشته و به شبکه منتقل کند. برای این کار نیاز به دروازه بروزرسانی دارد تا تعیین کند که چه اطلاعاتی از محتوای حافظه فعلی h_t و چه چیزی از مرحله قبلی $h_{t-۱}$ جمع‌آوری شوند. این مرحله توسط معادله زیر محاسبه می‌شود:

$$h_t = (۱ - z_t) \odot h_{t-۱} + z_t \odot h_t$$

همان‌طور که مشاهده شد، GRU توانست اطلاعات را با استفاده از دروازه‌های بروزرسانی و بازنشانی ذخیره و فیلتر کند. همچنین، به دلیل این‌که مدل در هربار ورودی جدید را از شست‌وشو نمی‌دهد و آن‌ها را به مراحل بعد منتقل می‌کند، مشکل محو گردایان را ندارد.

در ادامه به‌طور خلاصه به مقایسه بین LSTM و GRU می‌پردازیم.

اشتراک بین LSTM و GRU

- LSTM و GRU هر دو دارای واحدهای بروزرسانی با مولفه‌های افزودنی از t تا t+۱ هستند که در RNN سنتی وجود ندارد.
- واحد LSTM و GRU هر دو محتوای موجود را حفظ می‌کنند و محتوای جدید را در بالای آن اضافه می‌کنند.
- LSTM و GRU در تلاش برای از بین بردن مشکل محو و انفجار گرادیان موجود در RNN سنتی هستند.

تفاوت بین LSTM و GRU

- GRU دارای دو دروازه بروزرسانی و بازنشانی می‌باشد. در مقابل LSTM دارای سه دروازه ورودی، فراموش و خروجی است. GRU مانند LSTM دروازه خروجی ندارد. دروازه بروزرسانی در GRU کار دروازه ورودی و فراموشی در مشابهت با LSTM را انجام می دهد.
- پارامترهای GRU کم‌تر است، بنابراین از نظر محاسباتی کارآمدتر بوده و برای تعمیم به داده‌های کم‌تری نسبت به LSTM نیاز دارد.
- GRU حافظه داخلی (c_t) ندارد تا با حالت پنهان در معرض تفاوت باشد. در حالی که، LSTM حالت حافظه داخلی c را حفظ می‌کند.

۸.۲ ماشین تورینگ عصبی (NTM)[۱]

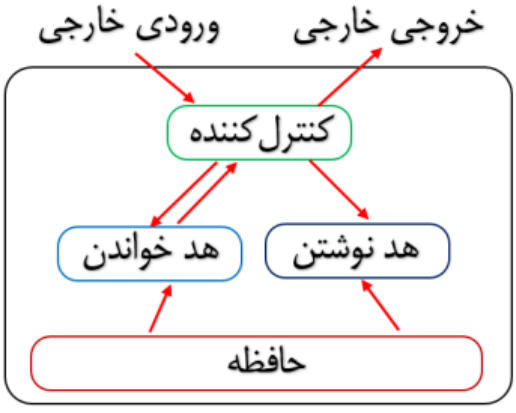
ماشین عصبی تورینگ اولین تلاش برای ساخت یک معماری یادگیری عمیق است تا قادر به یادگیری الگوریتم‌های دلخواه باشد. ماشین تورینگ عصبی سعی دارد روان‌شناسی شناختی "حافظه کاری[۲]" را را مدل‌سازی کند تا این امکان به آن اجازه دهد

[۱] Neural Turing machine

[۲] مفهومی در شناخت انسان است که نگهداری اطلاعات در ذهن و نحوه عملیات آن‌ها را توصیف می‌کند، مانند هنگام کار با یک مسئله ریاضی یا تفسیر زبان.

به شیوه‌ای که انسان با مسائل روبه‌رو می‌شود نزدیک شود. در این معماری یک بانک حافظه خارجی به یک سیستم مشابه LSTM اضافه می‌کند تا به معماری یادگیری عمیق اجازه استفاده از حافظه را در جهت محاسبه عملکردهای پیچیده را فراهم کند و اطلاعات را برای پردازش در نقاط مختلف محاسبه ذخیره کند. در مقایسه با ساختارهای RNN با حافظه داخلی، ماشین تورینگ عصبی از سازوکار توجه برای خواندن و نوشتن کارآمد حافظه خارجی بهره می‌برد. همین امر آن‌ها را به یک گزینه مطلوب‌تر برای گرفتن وابستگی‌های با درازمدت تبدیل می‌کند.

ایده اصلی معماری تورینگ عصبی جدا کردن محاسبه از حافظه است. این جدایی با هدف کاهش یک نقص در شبکه‌های عصبی بازگشتی می‌باشد، چرا که در شبکه‌های عصبی بازگشتی با افزایش ظرفیت حالت پنهان، پیچیدگی محاسباتی آن‌ها به‌طور تصاعدی رشد می‌کند. معماری یک ماشین تورینگ عصبی از دو مولفه اساسی تشکیل شده است: یک کنترل‌کننده شبکه عصبی و یک بانک حافظه. شکل ۲ـ۳۲ معماری سطح بالا از ماشین تورینگ عصبی را نمایش می‌دهد. همانند اکثر شبکه‌های عصبی، کنترل‌کننده از طریق بردارهای ورودی و خروجی با دنیای خارجی ارتباط برقرار کرده و بر خلاف شبکه عصبی استاندار، این شبکه با ارتباط بین ماتریس حافظه عملیات خواندن و نوشتن انتخابی را بدست می‌آورد. حافظه یک ماتریس $N \times W$ است، که N تعداد مکان‌های حافظه (سطرها) و W اندازه بردار در هر مکان است. کنترل‌کننده رابط بین لایه‌های دیگر شبکه و حافظه است.



شکل ۲ـ۳۲ معماری سطح بالا ماشین تورینگ عصبی

کنترل‌کننده: کنترل‌کننده یک شبکه عصبی می‌باشد که نمایش داخلی ورودی را که توسط هد خواندن و نوشتن برای تعامل با حافظه استفاده می‌شود، فراهم می‌کند. به عبارت دیگر، رابطی بین داده‌های ورودی و حافظه می‌باشد. نوع کنترل‌کننده مهم‌ترین انتخاب در معماری یک ماشین تورینگ عصبی است. این کنترل‌کننده می‌تواند یک شبکه عصبی بازگشتی یا حتی ممکن است یک شبکه عصبی پیش‌خور باشد.

سازوکارهای خواندن و نوشتن: هد خواندن و نوشتن سبب جالب شدن ماشین تورینگ عصبی شده است. آن‌ها تنها مولفه‌هایی هستند که همیشه با حافظه ارتباط متقابل دارند. براساس فرمان‌های دریافتی از کنترل‌کننده، یک هد با حضور یافتن در شکاف‌های حافظه با درجات مختلف، یک یا چند مکان از حافظه را برای خواندن یا نوشتن انتخاب می‌کنند.

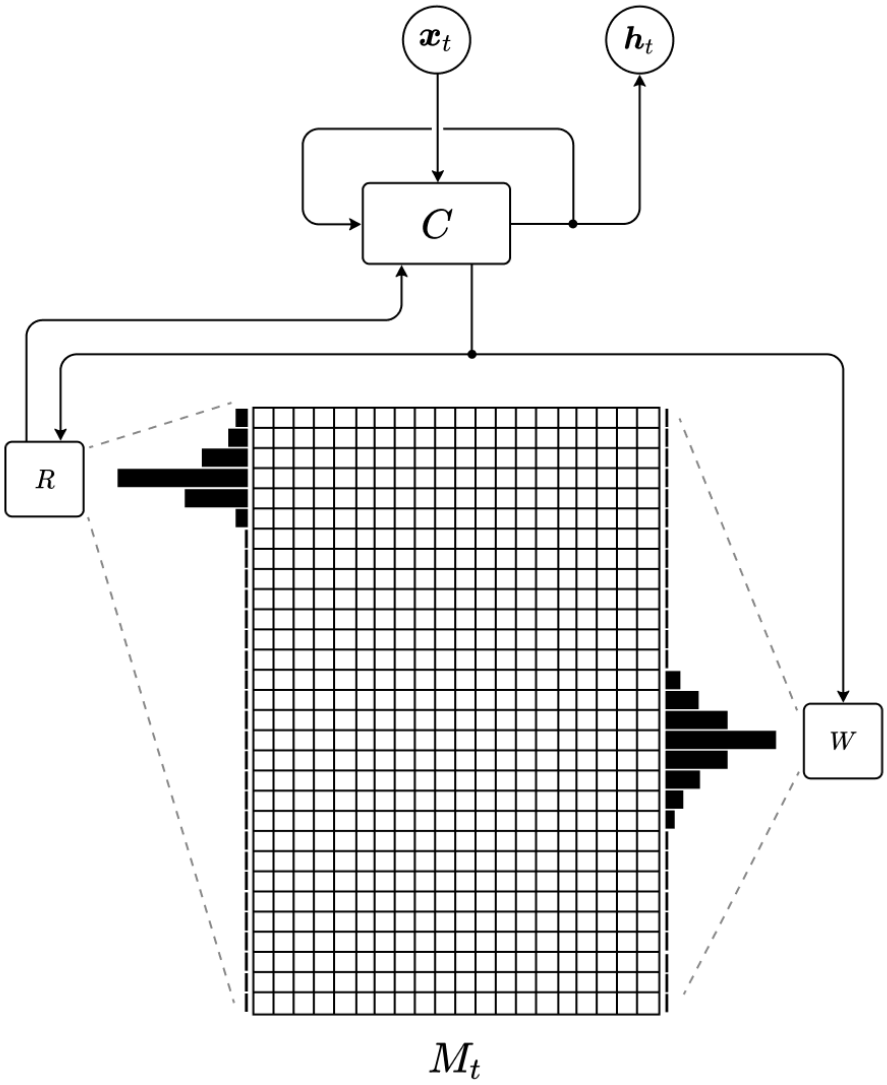
در ادامه قصد داریم به جزئیات ریاضی نحوه عملیات خواندن و نوشتن و نحوه تولید بردار توجه بپردازیم*. از همین رو، در شکل ۲ ـ ۳۳ معماری یک شبکه عصبی با نحوی که برای تشریح مناسب‌تر باشد ارائه شده است. در این شماتیک، کنترل کننده با حرف C، ماتریس حافظه با حرف M و هد خواندن و نوشتن با حروف R و W، t نمایانگر هر مرحله، x_t ورودی در هر مرحله و h_t حالت فعلی را نشان می‌دهند.

۲.۱.۸ خواندن

فرض کنید در مرحله زمانی مشخص t، یک هد خواندن R و حافظه M_t داریم. در مثال ما این حافظه دارای ٤ اسلات حافظه مجزا می‌باشد و هر کدام از آن‌ها یک بردار دودویی ٥ بعدی را ذخیره می‌کنند. براساس ورودی فعلی از کنترل کننده، هد خواندن یک بردار توجه یکسان‌سازی شده را با مقادیر بین ۰ و ۱ و مجموعه کل ۱ تولید می‌کند. هد تعیین می‌کند تا چه میزانی از تمرکز به هر مکان داده شود. با استفاده از این وزن‌ها می‌توانیم رویه خواندن را توسط یک بردار r_t که می‌تواند از حاظفه بخواند، تعریف کنیم:

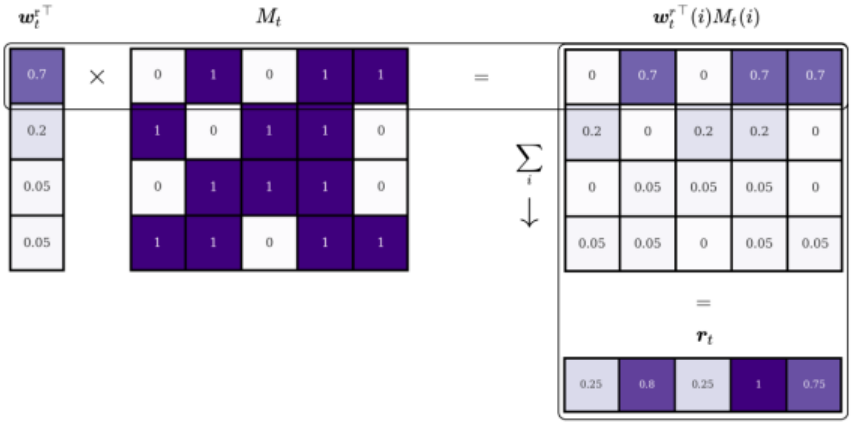
$$r_t = \sum_i w_t^r(i)\, M_t(i)$$

* برای جزئیات بیشتر و نحوه آدرس‌دهی می‌توان به این منبع مراجعه کنید:
https://www.niklasschmidinger.com/posts/۲۰۱۹-۱۲-۲۵-neural-turing-machines



شکل ۲ـ۳۳ معماری ماشین تورینگ عصبی با جزئیات بیشتر

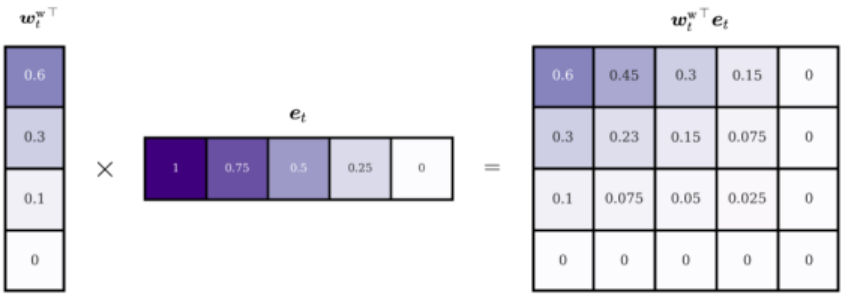
این عملیات را می‌توان در دو مرحله تجسم کرد. ابتدا هر عنصری از بردار توجه w_t^r با ردیف مربوط در حافظه ضرب می‌شود. دوم، ردیف‌ها برای تولید بردار خوانده شده r_t بهم اضافه می‌شوند، مطابق تصویر زیر:



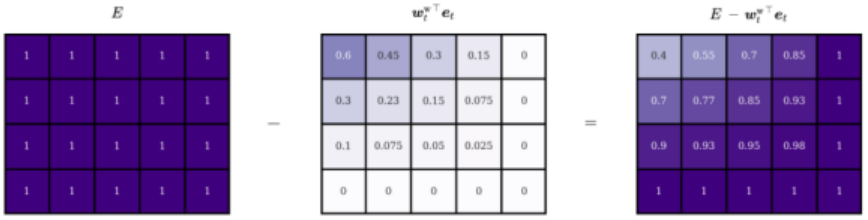
بردار خوانده شده حاصل یک جمع وزنی از محتویات مکان‌ها است.

۲.۸.۲ نوشتن

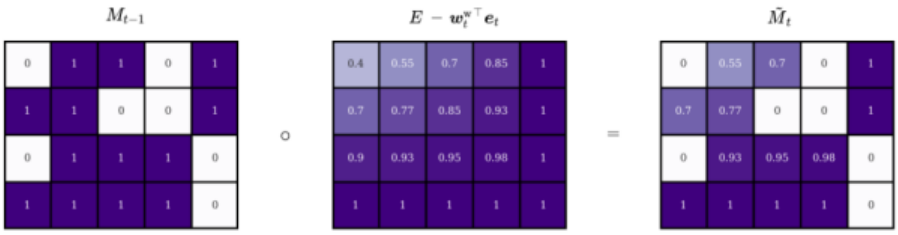
با الهام از LSTM، عملیات نوشتن به دو بخش قابل تجزیه می‌باشد: پاک کردن و اضافه کردن. در زمان مشخص t، ما یک هد نوشتن W با وزن‌های توجه w_t^w و یک ماتریس حافظه از مرحله زمانی قبلی M_{t-1} داریم. قبل از اینکه بتوانیم اطلاعات را حذف کنیم، باید تعیین کنیم که کدام یک از عناصر ذخیره شده باید در چه درجه‌ای پاک شوند. برای این منظور، بردار پاک‌کردن را با همان ابعاد محل حافظه واحد و مقادیر در محدوده $[0,1]$ معرفی می کنیم. هر عنصر مربوط به یک سلول در یک مکان مشخص است و نشان می‌دهد که چند درصد از مقدار سلول باید پاک شود. به عنوان مثال، اگر یک عنصر در e_t دارای مقدار ۱ باشد، مقدار سلول مربوطه در محل حافظه به طور کامل حذف می شود. با ضرب کردن وزن‌ها در بردار پاک کردن، یک ماتریس با ابعاد مشابه حافظه خود بدست می آوریم:



هر ردیف از ماتریس حاصل، نسخه متفاوتی از بردار اصلی پاک کردن را نشان می‌دهد که با توجه به مقدار توجه هد در آن مکان، مقیاس‌بندی می‌شود. می توانیم این ماتریس را به عنوان فیلتر پاک کردن تفسیر کنیم. چرا که، محتویات آن توصیف می‌کند چند درصد از یک سلول حافظه داده شده باید حذف شود. با کم کردن آن از ماتریس E دوم، آن را به یک فیلتر باقی‌مانده با اثر مخالف تبدیل می کنیم.



برای پایان دادن به مرحله پاک کردن، به‌سادگی ضرب عناصر ساخته‌شده بین فیلتر خود و حافظه مرحله قبلی می‌باشد را محاسبه می‌کنیم:

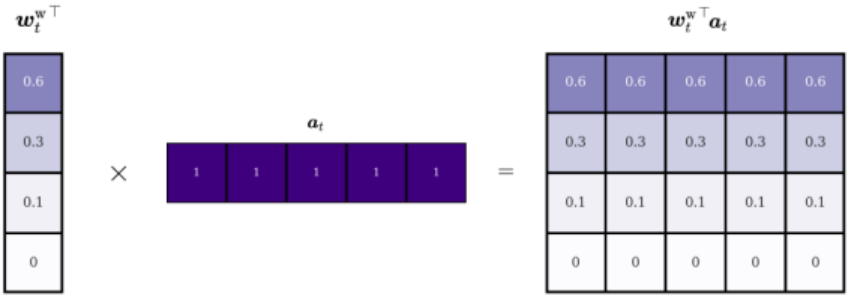


از منظر ریاضی، می‌توان مراحل بالا را با معادله زیر بیان کنیم:

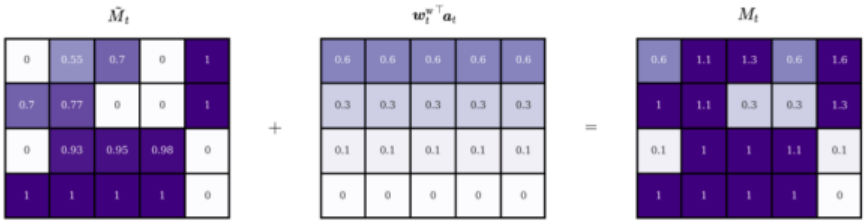
$$\bar{M}_t = M_{t-1} \circ [E - w_t^w e_t]$$

که در این معادله $\bar{M}_t$ ماتریس حافظه پاک‌شده است.

مرحله بعدی در فرآیند نوشتن، تولید اطلاعاتی است که حافظه قدیمی باید با آن‌ها به‌روز شود. برای دستیابی به این هدف، یک بردار a_t ارائه و آن را با ترانهاده بردار توجه ضرب می‌کنیم. این عملیات یک ماتریس بروزرسانی تولید می‌کند:



باز هم، هر ردیف از ماتریس، نسخه متفاوتی از بردار اصلی را نشان می‌دهد که با توجه به مقدار هد توجه در آن مکان، مقیاس‌بندی می‌شود. سرانجام، عملیات نوشتن با افزودن ماتریس بروزرسانی به حافظه پاک شده به پایان می رسد:



روند کامل مرحله نوشتن را می توان با مجموعه معادله‌های زیر بیان کرد:

$$\bar{M}_t = M_{t-1} \circ [E - w_t^w e_t]$$

$$M_t = \bar{M}_t + w_t^{wT} a_t$$

یا به‌طور فشرده‌تر به صورت:

$$M_t = M_{t-1} \circ \left[E - w_t^{wT} e_t\right] + w_t^{wT} a_t$$

۹.۲ شبکه‌های عصبی همگشتی

تصاویر نوع خاصی از داده‌های ورودی را تشکیل می‌دهند. ما از عکس‌ها و تصاویر برای برقراری ارتباط انواع چیزها به دلایل حرفه ای، اجتماعی و شخصی استفاده می کنیم. از برچسب‌گذاری چهره یک دوست در جهت راحت‌تر پیدا کردن آن در مجموعه‌ای از عکس‌ها تا داوری در مورد این‌که آیا لک روی رادیوگرافی یک وضعیت پزشکی است یا خیر، نیاز به نگاه دقیق‌تری دارد. استخراج معانی از تصاویر بسیار مهم می‌باشد. در این بخش به استخراج معانی از تصاویر با استفاده از ایده‌ای به نام همگشت پرداخته می‌شود.

برای برخی از انواع داده‌ها به ویژه تصاویر، شبکه‌های پیش‌خور کارکرد خوبی ندارند. همان‌طور که پیش‌تر بیان شد، در شبکه‌های پیش‌خور هر نورون به‌طور کامل به هر یک از نورون‌های موجود در لایه بعدی متصل است. به‌طور دقیق‌تر، هر نورون در لایه پنهان تابعی را محاسبه می‌کند که به مقادیر هر گره در لایه ورودی وابسته می‌باشد. با این حال، در بازشناسایی‌های بصری، اغلب استفاده از زیرساخت محلی در تصویر سودمند است. به عنوان مثال، پیکسل‌هایی که در یک تصویر نزدیک به‌هم هستند (پیکسل‌های مجاور) همبستگی شدیدی دارند. در حالی که، پیکسل‌هایی که در تصویر از یک‌دیگر بسیار دور هستند، همبستگی کم‌تر یا عدم همبستگی دارند. از همین‌رو، جای تعجب نیست که بسیاری از بازنمایی‌های ویژگی‌های مورد استفاده در مسائل بینایی رایانه براساس ویژگی‌های محلی در تصویر است. در معماری شبکه عصبی همگشتی، این زیر ساخت محلی را با محدود کردن هر نورون درگیر می‌کنیم تا فقط به یک زیرمجموعه محلی از متغیرهای لایه قبلی بستگی داشته باشد.

ویژگی دومی که شبکه عصبی همگشتی را از شبکه‌های عصبی معمولی متمایز می‌کند، این است که وزن‌های یال شبکه در نورون‌های مختلف لایه‌های پنهان مشترک است. حتما از پیش به یاد دارید که هر نورون در شبکه، ابتدا یک ترکیب خطی وزنی از ورودی‌های خود را محاسبه می‌کند. می‌توانیم این فرآیند را به عنوان ارزیابی یک فیلتر خطی بر روی مقادیر ورودی مشاهده کنیم. در این زمینه، تقسیم وزن در چندین نورون در یک لایه پنهان به معنی ارزیابی یک فیلتر در چندین زیر پنجره از تصویر ورودی است. در این راستا، می‌توانیم شبکه عصبی همگشتی را به عنوان یادگیری موثر مجموعه فیلترها ببینیم که هر کدام از آن‌ها برای همه‌ی زیر پنجره‌های تصویر ورودی اعمال می‌شود.

استفاده از همان فیلترها در کل تصویر، شبکه را مجبور به یادگیری یک رمزگذاری یا بازنمایی کلی از داده‌های زیربنایی می‌کند. یکی دیگر از مزایای تقسیم وزن این است که به‌طور قابل توجهی تعداد پارامترها در شبکه کاهش پیدا کرده و آموزش آن را آسان‌تر و کارآمدتر می‌کند.

قبل از توسعه یادگیری عمیق در بینایی رایانه، یادگیری مبتنی‌بر استخراج متغیرهای مورد علاقه با عنوان ویژگی‌ها بود. اما، این روش‌ها در جهت پردازش تصاویر نیاز به تجربه زیادی داشت. شبکه‌های عصبی همگشتی که توسط لکان[۱] معرفی شدند، پردازش تصاویر را متحول کرده و استخراج ویژگی‌ها به صورت دستی را حذف کرد.

شبکه‌های عصبی همگشتی، نوع خاصی از شبکه‌های عصبی در پردازش داده‌هایی است، که دارای یک ساختار مکانی معلوم و مشبکی هستند. این شبکه‌ها، ورودی‌هایی که از لحاظ ساختار مکانی نزیک به یک‌دیگر باشند را، به‌صورت معناداری به یک‌دیگر ارتباط می‌دهند. برای مثال، می‌توان پیکسل‌های موجود در تصاویر را مشبک دوبعدی در نظر گرفت. این فرض برای تصاویر درست است، چرا که پیکسل‌های نزدیک به‌هم از لحاظ معنایی با یک‌دیگر ارتباط دارند. شبکه‌های عصبی همگشتی از مطالعه قشر بینایی مغز پدید آمده و از سال ۱۹۸۰ تاکنون در تشخیص تصاویر مورد استفاده قرار می‌گیرد. در چندین سال گذشته، به لطف افزایش قدرت محاسباتی و افزایش میزان داده‌های آموزشی، CNNها موفق به دستیابی به عملکرد فوق بشری در برخی کاربردهای پیچیده دیداری شده‌اند.

شبکه‌های همگشتی، نقش مهمی را در تاریخچه یادگیری عمیق به‌همراه داشته است. آن‌ها نمونه‌ای مهم و موفق در فهم ما از مطالعه مغز در کاربردهای یادگیری ماشین هستند. شبکه‌های عصبی همگشتی جزو اولین شبکه‌های عصبی بودند که در حل و انجام کابردهای تجاری مهم مورد استفاده قرار گرفته و حتی تا امروز در صدر برنامه‌های کابردی تجاری یادگیری عمیق قرار دارند.

---

[۱] LeCun

۱.۹.۱ ساختار شبکه عصبی همگشتی

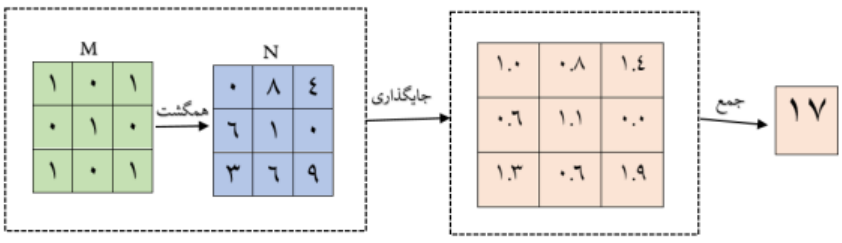
یک شبکه عصبی همگشتی توسط چندین لایه تشکیل شده است: **لایه همگشت**، **لایه ادغام** و **لایه تماما متصل**. در سطح اول برخورد با یک تصویر ورودی، شبکه عصبی همشگتی معمولا ویژگی‌های بصری ساده مانند، لبه‌ها یا لکه‌های رنگ را می‌آموزد. پس از آن، در سطح دوم ویژگی‌های سطح قبلی را ترکیب می‌کند. افزودن سطوح بیشتر، بسته به داده‌ها و کابرد مسئله ویژگی‌های سطح بالاتری مانند، چهره را می‌یابند. در ادامه به بررسی هر یک لایه‌های تشکیل‌دهنده در ساختار شبکه عصبی همگشتی می‌پردازیم.

۱.۱.۹.۲ لایه همگشت

لایه همگشت مهم‌ترین قسمت سازنده CNN بوده و همیشه به عنوان اولین لایه استفاده می‌شود. این لایه بیشترین بار محاسباتی را برعهده دارد. به‌طور کلی CNN یک شبکه عصبی است که حداقل دارای یک لایه همگشت در ساختارش است. **همگشت در عمومی‌ترین حالت تعریف خود، انجام عمل ریاضی بر روی دو تابع با مقادیر حقیقی است.** برای نشان عملگر همگشت فرض کنید دو ماتریس M و N به صورت زیر تعریف شده باشند:

$$M = \begin{bmatrix} ۱ & ۰ & ۱ \\ ۰ & ۱ & ۰ \\ ۱ & ۰ & ۱ \end{bmatrix} \quad , \quad N = \begin{bmatrix} ۰ & ۸ & ٤ \\ ٦ & ۱ & ۰ \\ ۳ & ٦ & ۹ \end{bmatrix}$$

عملگر همگشت تنها بر روی ماتریس‌های با تعداد سطر و ستون یکسان قابل اجرا است. برای دو M و N ماتریس عملگر همگشت به صورت زیر انجام می‌شود:



وظیفه اصلی لایه همگشت، شناسایی ویژگی‌های یافت شده در مناطق محلی تصویر ورودی بوده که این ویژگی‌ها در کل مجموعه داده مشترک هستند. این شناسایی ویژگی‌ها از طریق اعمال فیلترها منجر به تولید نقشه ویژگی می‌شود. لایه همگشت، یک فیلتر محلی را بر روی تصویر ورودی اعمال می‌کند. این امر منجر می‌شود طبقه‌بندی بهتری در پیکسل‌های همسایه‌ای که همبستگی بیشتری بین آن‌ها وجود دارد در همان تصویر صورت پذیرد. به عبارت دیگر، پیکسل‌های تصاویر ورودی می‌توانند با یک‌دیگر همبستگی داشته باشند. به عنوان مثال، در تصاویر صورت، بینی همیشه بین چشم‌ها و دهان قرار دارد. وقتی فیلتر را به زیرمجموعه‌ای از تصویر اعمال می‌کنیم، برخی از ویژگی‌های محلی را استخراج می‌کنیم.

از این لایه به عنوان لایه استخراج ویژگی نیز یاد می‌شود. چراکه، ویژگی‌های تصویر در این لایه استخراج می‌شوند. دو مفهوم مهم در لایه همگشت وجود دارد. **گام‌ها**[۱] و **لایه‌گذاری**[۲]. گام‌ها تعداد پیکسل هسته یا فیلتر بوده که روی ماتریس ورودی در حرکت است (اندازه جابه‌جایی فیلتر در راستای افقی و عمودی). لایه‌گذاری وقتی استفاده می‌شود که فیلتر با ماتریس ورودی متناسب نیست. دو نوع لایه‌گذاری وجود دارد: **لایه‌گذاری معتبر**[۳] و **لایه‌گذاری یکسان**[۴] یا صفر. در لایه‌گذاری معتبر پیکسل‌های حاشیه ماتریس ورودی کنار گذاشته می‌شود. لایه‌گذاری یکسان، صفرها را به حاشیه اضافه می‌کند تا فیلتر متناسب با ماتریس ورودی باشد.

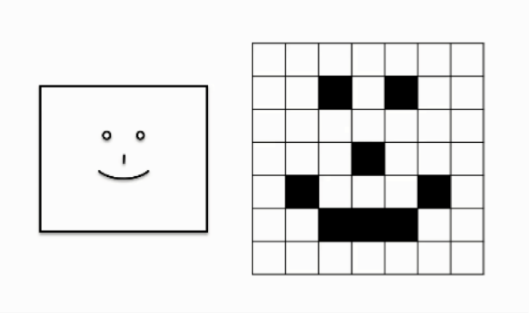
اکنون به بررسی این لایه همراه با مثال می‌پردازیم. در لایه همگشت سه مولفه مهم وجود دارد:

- تصویر ورودی
- آشکارساز ویژگی[۵]
- نقشه ویژگی[۶]

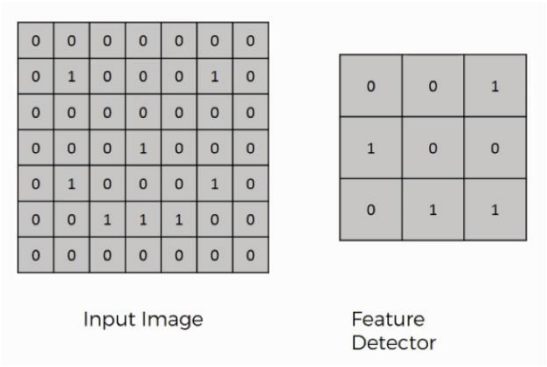
---

۱ strides
۲ padding
۳ valid padding
۴ same padding
۵ Feature detector
۶ Feature map

فرض کنید تصویر اصلی ما به‌صورت شکل ۲ ـ ۳٤ باشد. پس از تبدیل آن به فرم ورودی، همان‌طور که در تصویر ۲ ـ ۳۵ مشاهده می‌شود، الگوی صفر و یک‌ها نمایش چهره خندان را نشان می‌دهند.

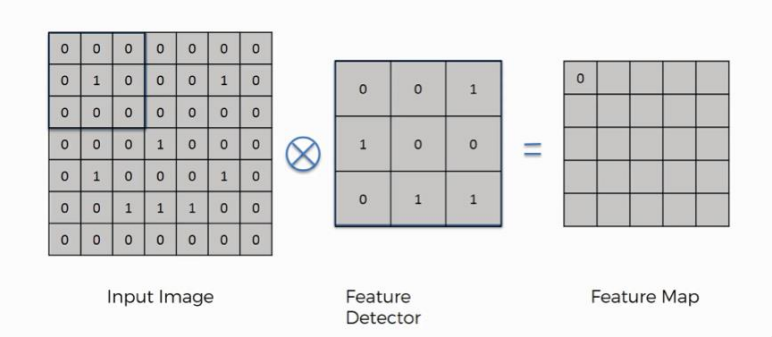


شکل ۲ ـ ۳۴ تصویر ورودی

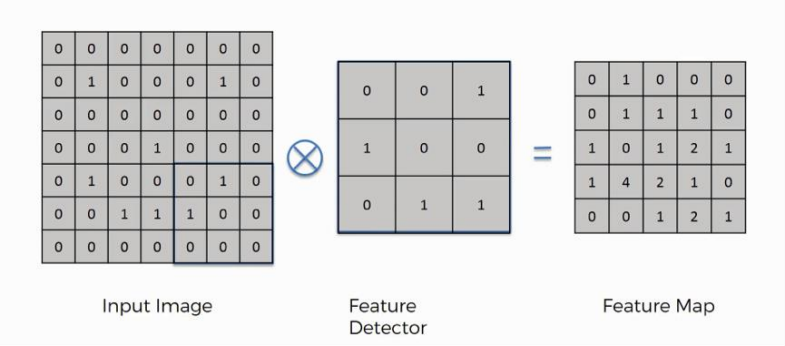


شکل ۲ ـ ۳۵ تصویر ورودی پس از تبدیل

در تصویر ۲ ـ ۳۵ از یک آشکارساز ویژگی که معمولا از آن به عنوان هسته یا فیلتر یاد می‌شود، با اندازه ۳*۳ استفاده شده است. این ماتریس ۳*۳ را از گوشه بالا سمت چپ تصویر مطابق تصویر ۲ ـ ۳٦ به تعداد سلول‌هایی که در آن آشکارساز ویژگی با تصویر ورودی مطابقت دارد، استفاده شده است. و با اعمال عملگر همگشت، اولین مقدار از نقشه ویژگی بدست می‌آید. پس از آن، آشکارساز ویژگی به سمت راست حرکت می‌کند و همان کار را انجام داده تا ردیف اول نقشه ویژگی کامل شود. بعد از پشت سر گذاشتن ردیف اول، به ردیف دیگر رفته و این عمل را تا زمان تکمیل شدن تمام نقشه ویژگی تکرار می‌کنیم. تا تصویر ۲ ـ ۳۷ حاصل شود.



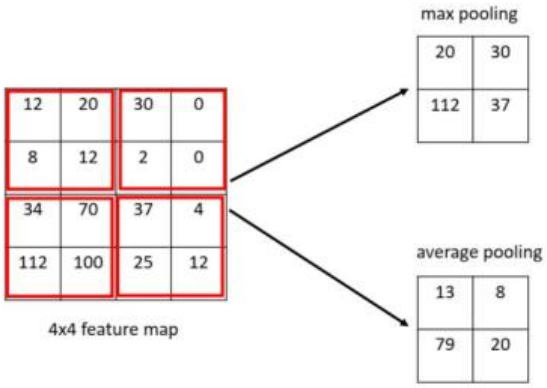
شکل ۲ـ۳۶ نحوه بدست آوردن نقشه ویژگی‌ها



شکل ۲ـ۳۷ نقشه ویژگی‌ها

۲.۱.۹.۲ لایه ادغام

معمولا از لایه ادغام به طور دوره‌ای بین دو لایه‌ی متوالی همگشت استفاده می‌شود. وظیفه آن کاهش ابعاد نقشه‌های ویژگی است. این کار علاوه بر استخراج ویژگی‌های مهم در نقشه ویژگی، با کاهش مقدار پارامترها قدرت محاسباتی مورد نیاز برای پردازش داده‌ها را نیز کاهش می‌دهد. دو لایه ادغام مهم وجود دارد: ادغام حداکثری و ادغام میانگین. تفاوت بین این دو در تصویر ۲ـ۳۸ قابل نمایش است. ادغام حداکثری عملکرد بهتری در استخراج ویژگی‌های غالب و مهم دارد.



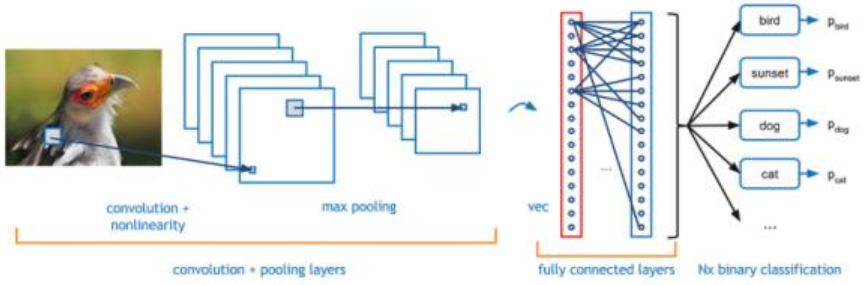
شکل ۲ـ۳۸ تفاوت ادغام حداکثری و ادغام میانگین

۳.۱.۹.۲ لایه تماما متصل

شبکه‌های عصبی همگشتی از دو مرحله اصلی ساخته می‌شوند: مرحله استخراج ویژگی و مرحله طبقه‌بندی. لایه‌های همگشت و ادغام وظیفه استخراج ویژگی‌ها را عهده‌دار هستند. لایه تماما متصل در شبکه عصبی همگشتی مانند لایه‌های مخفی یک شبکه عصبی استاندارد کار کرده و در انتهای شبکه عصبی همگشتی قرار می‌گیرند. این لایه دقیقا همان جایی است که طبقه‌بندی در آن اتفاق می‌افتد. بعد از استفاده چندین لایه مختلف می‌توان از لایه تماما متصل در انتهای شبکه CNN در جهت محاسبه ویژگی‌های دلخواه و امتیازات خروجی استفاده کرد. خروجی یک بردار N بعدی بوده که N در آن به تعداد کلاس‌ها اشاره می‌کند. به عنوان مثال، می‌خواهیم بین یک پرنده، غروب خورشید، سگ و گربه یک طبقه‌بندی انجام دهیم. مقدار N در این مثال ٤ می‌شود. هر عدد موجود در بردار احتمال یک کلاس خاص را نشان می‌دهد. لایه تماما متصل مشخص می‌کند که کدام ویژگی بیشترین ارتباط را با یک کلاس خاص دارد. به عنوان مثال، در تصویر پرنده، مقادیر سطح بالا در نقشه ویژگی‌ها وجود دارد که نمایانگر ویژگی بال پرنده است. فرض کنید مقادیر بردار بدست آمده در این لایه به‌صورت زیر باشد:

$$[۰/٤ \ , \ ۰ \ , \ ۰/۰۰۳ \ , \ ۰/۸۷]$$

این نتایج نشان می‌دهد که، به احتمال ۰/۸۷ درصد تصویر مربوط به یک پرنده می‌باشد. در تصویر ۲ ـ ۳۹ مثالی از شبکه عصبی همگشتی را در طبقه‌بندی تصویر نشان می‌دهد.



شکل ۲ ـ ۳۹ مثالی از شبکه عصبی همگشتی در طبقه‌بندی تصویر

۲.۹.۲ لایه‌گذاری و گام

در بسیاری از موارد، از روش‌های لایه‌گذاری و گام‌های همگشت که بر اندازه خروجی تاثیرگذار هستند، استفاده می‌کنیم. از آن‌جایی که هسته‌ها معمولا دارای ارتفاع و عرض بیشتر از ۱ هستند، استفاده از همگشت‌های پی‌درپی خروجی را نسبت به ورودی به‌طور چشم‌گیری کاهش می‌دهند. به عنوان مثال، اگر با یک تصویر ۲۴۰ × ۲۴۰ پیکسل شروع کنیم. ۱۰ لایه همگشت ۵ × ۵ تصویر را به ۲۰۰ × ۲۰۰ پیکسل کاهش می‌دهد. که نتیجه آن برش ۳۰٪ از تصویر و از بین بردن اطلاعات جالب در مرزهای تصویر اصلی را درپی دارد. لایه‌گذاری محبوب‌ترین ابزار برای حل این مشکل است، که راه حل آن اضافه کردن پیکسل‌های اضافی پرکننده اطراف مرز تصویر ورودی است. به‌طور معمول مقادیر این پیکسل‌ها را صفر قرار می‌دهیم. در موارد دیگر، ممکن است بخواهیم بعد را به شدت کاهش دهیم (برای مثال، اگر وضوح تصویر ورودی اصلی مناسب نباشد). گام‌های همگشت محبوب‌ترین روشی است که به این موضوع کمک می‌کند.

از گام‌ها و لایه‌گذاری به‌طور موثر می‌توان برای تنظیم ابعاد داده‌ها استفاده کرد

۳.۹.۲ آموزش در CNN

روند بهینه‌سازی در CNN مشابه با شبکه‌های عصبی پیش‌خور می‌باشد. به‌طور مشابه، یادگیری پارامترهای شبکه توسط الگوریتم پس‌انتشار و بهینه‌سازی گردایان کاهشی تصادفی صورت می‌گیرد. فاز اول، انتشار جلورو می‌باشد، جایی که سیگنال‌ها از ورودی شبکه به خروجی انتشار می‌یابند. در آخرین لایه، خروجی توسط تابع هزینه با مقدار دلخواه مقایسه و برآورد خطا صورت می‌گیرد. در فاز دوم، مجددا از الگوریتم پس‌انتشار برای جبران این خطا استفاده می‌شود. با این حال، فرآیند یادگیری در شبکه عصبی همگشتی در مشابهت با شبکه عصبی پیش‌خور وضعیتی پیچیده‌تر دارد، چرا که از انواع مختلفی لایه‌ها تشکیل شده و فاز انتشار جلورو و عقبگرد از قوانین ویژه‌ای در هر لایه پیروی می‌کنند. نرون‌ها در شبکه عصبی همگشتی برخلاف شبکه عصبی پیش‌خور که هر نرون دارای یک بردار وزن جداگانه هستند، وزنی مشترک دارند. این اشتراک وزن‌ها منجر به کاهش تعداد کلی وزن‌های قابل آموزش می‌شود.

انتشار جلورو در لایه همگشت

هر لایه همگشت، بر روی ورودی‌های خود در حال انجام عملگر همگشت می‌باشد. با فرض این‌که ورودی‌های یک لایه $N \times N$ و هسته آن $M \times M$ باشد. همگشت بدون استفاده از دنباله‌گذاری صفر $(N - m + ۱) \times (N - m + ۱)$ و محاسبه همگشت خروجی x_{ij}^{l} به این صورت خواهد بود:

$$x_{ij}^{l} = \sum_{a=۰}^{m-۱} \sum_{b=۰}^{m-۱} \omega_{ab} y_{(i+a)(j+b)}^{l-۱}$$

جایی که $i, j \in (۰, n - m + ۱)$، l شاخص لایه فعلی، ω_{ab} وزن هسته و $y_{(i+a)(j+b)}^{l-۱}$ خروجی لایه قبلی می‌باشد.

سپس، خروجی لایه همگشت y_{ij}^{l} با اعمال یک تابع فعال‌سازی غیرخطی بر روی خروجی همگشت x_{ij}^{l} محاسبه می‌شود:

$$y_{ij}^{l} = \sigma(x_{ij}^{l})$$

جایی که σ یک تابع فعال‌سازی غیرخطی است.

انتشار عقبگرد در لایه همگشت

انتشار عقبگرد در لایه همگشت از همان اصول الگوریتم پس‌انتشار که در فصل دوم شرح داده شد پیروی می‌کند. تنها تفاوت در این است که هسته همگشت وزن کل لایه را به اشتراک می‌گذارد و هسته دارای بایاس نیست. اگر مقدار تابع خطا را E فرض کنیم، با توجه به مقدار خطا در لایه همگشت و مشتق جزئی خطای لایه قبلی با توجه به هر خروجی نرون $\frac{\partial E}{\partial y_{ij}^{l}}$ باشد، تاثیر وزن هسته بر تابع هزینه باید محاسبه شود:

$$\frac{\partial E}{\partial w_{ab}} = \sum_{i=0}^{N-m} \sum_{j=0}^{N-m} \frac{\partial E}{\partial x_{ij}^{l}} \frac{\partial x_{ij}^{l}}{\partial w_{ab}}$$

از آنجایی که داشتیم:

$$x_{ij}^{l} = \sum_{a=۰}^{m-۱} \sum_{b=۰}^{m-۱} \omega_{ab} y_{(i+a)(j+b)}^{l-۱}$$

نتیجه می‌شود که:

$$\frac{\partial x_{ij}^{l}}{\partial w_{ab}} = y_{(i+a)(j+b)}^{l-۱}$$

بنابراین داریم:

$$\frac{\partial E}{\partial w_{ab}} = \sum_{i=۰}^{N-m} \sum_{j=۰}^{N-m} \frac{\partial E}{\partial x_{ij}^{l}} y_{(i+a)(j+b)}^{l-۱}$$

برای محاسبه گرادیان باید مقدار $\frac{\partial E}{\partial x_{ij}^{l}}$ را که اغلب دلتا نامیده می‌شود، بدانیم. محاسبه دلتا نسبتا ساده می‌باشد. از قانون زنجیره‌ای استفاده می‌کنیم:

$$\frac{\partial E}{\partial x_{ij}^{l}} = \frac{\partial E}{\partial y_{ij}^{l}} \frac{\partial y_{ij}^{l}}{\partial x_{ij}^{l}} = \frac{\partial E}{\partial y_{ij}^{l}} \frac{\partial}{\partial x_{ij}^{l}} \left(\sigma(x_{ij}^{l})\right) = \frac{\partial E}{\partial y_{ij}^{l}} \acute{\sigma}(x_{ij}^{l})$$

همان‌طور که مشاهده می‌شود، از آنجا که خطای لایه فعلی $\frac{\partial E}{\partial y_{ij}^{l}}$ قبلا داده شده است و مقدار آن را می‌دانیم، می‌توانیم به راحتی دلتا را $\frac{\partial E}{\partial x_{ij}^{l}}$ در لایه فعلی تنها با استفاده از مشتق تابع فعال سازی $\acute{\sigma}(x)$ محاسبه کنیم. از آنجا که خطاهای موجود در لایه فعلی را می‌دانیم، اکنون با توجه به وزن‌های استفاده شده توسط این لایه همگشت، هر آنچه برای محاسبه گرادیان نیاز است را داریم. علاوه بر این، برای محاسبه وزن برای این لایه همگشت، باید خطاها را به لایه قبلی منتقل کنیم. می‌توانیم یک بار دیگر از قانون زنجیره‌ای استفاده کنیم:

$$\frac{\partial E}{\partial y_{ij}^{l-1}} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \frac{\partial E}{\partial x_{(i-a)(j-b)}^{l}} \frac{\partial x_{(i-a)(j-b)}^{l}}{\partial y_{ij}^{l-1}}$$

از معادله

$$x_{ij}^{l} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \omega_{ab} y_{(i+a)(j+b)}^{l-1}$$

نتیجه می‌شود که $\frac{\partial x_{(i-a)(j-b)}^{l}}{\partial y_{ij}^{l-1}} = \omega_{ab}$ بنابراین داریم:

$$\frac{\partial E}{\partial y_{ij}^{l-1}} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \frac{\partial E}{\partial x_{(i-a)(j-b)}^{l}} \omega_{ab}$$

این مقدار بدست آمده خطای لایه قبلی را به ما می‌دهد.

۴.۹.۲ دلیل استفاده از CNN در دسته‌بندی تصاویر

دلایل مختلفی برای استفاده از به جای پرسپترون چندلایه در دسته‌بندی تصاویر وجود دارد:

- تصویر به عنوان ورودی معمولا به لایه‌های بسیار بزرگی منتهی می‌شود، چرا که هر پیکسل یک مقدار ورودی است. اگر تنها از شبکه عصبی معمولی استفاده شود، با اتصال کامل بین لایه‌ها برای ذخیره وزن‌ها مقدار زیادی حافظه

مورد نیاز است. در یک لایه همگشت هیچ اتصال کاملی وجود نداشته، در مقابل اتصال پراکنده وجود دارد. این اتصال پراکنده منجر به اتصال کمتر نسبت به یک لایه تماما متصل شده که نتیجه آن ذخیره وزن کمتر می‌باشد.

- در یک لایه تماما متصل، هر وزن نسبت به سایر موارد مستقل است، که این امر منجر به ذخیره و به‌هنگام‌سازی مقادیری زیادی می‌شود. حال آن‌که در یک لایه همگشت، وزن‌ها مشترک است، زیرا وزن‌ها مقادیر موجود در هسته هستند. هر زمان که همگشت به یک ناحیه مشخص از ورودی انجام می‌شود، با همان هسته انجام می‌شود، بنابراین وزن‌ها یکسان است.
- استفاده از هسته‌های چندگانه در لایه‌ی همگشت امکان‌پذیر است. یک هسته را می‌توان به عنوان فیلتر برای تشخیص یک الگوی کوچک خاص در تصویر ورودی استفاده کرد. با استفاده از چندین هسته می‌توان، چندین الگوی کوچک را هم‌زمان جستجو کرد.

۵.۹.۲ معماری‌های CNN

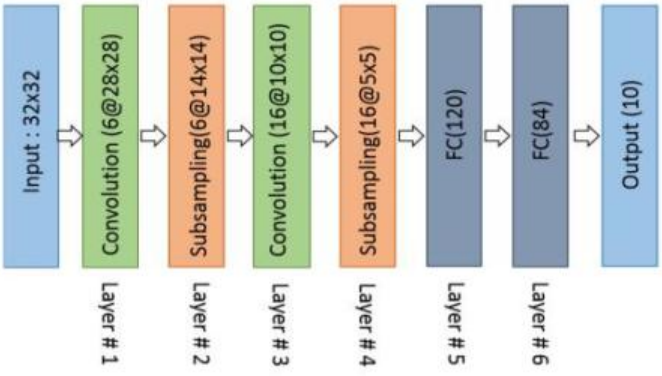
در این بخش به بررسی چند معماری از CNN می‌پردازیم.

LeNet ۱.۵.۹.۲

LeNet که در سال ۱۹۹۸ طراحی شد، اولین معماری مبتنی‌بر همگشت بود که از الگوریتم پس‌انتشار برای آموزش شبکه استفاده می‌کرد. این معماری برای دسته‌بندی اسناد دست‌نویس طراحی شده است. با وجود این‌که این معماری کار خود را به خوبی انجام می‌داد، موفقیت چندانی در آن زمان کسب نکرد و تا چندین دهه پس از ارائه آن در گمنامی به‌سر می‌برد، چرا که دارای مشکلاتی به شرح زیر بود:

- مجموعه داده‌های برچسب‌دار کم
- رایانه‌های کند
- استفاده از تابع فعال‌سازی غیرخطی اشتباه

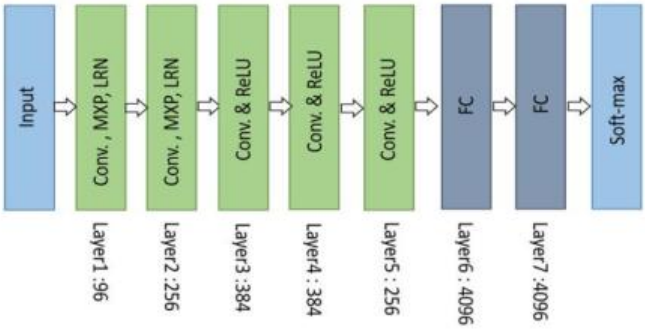
در شکل ۱ ـ ٤٠ معماری LeNet قابل مشاهده است.



شکل ۲ـ ۴۰ معماری LeNet

AlexNet ۲.۵.۹.۲

در سال ۲۰۱۲ الکس کریشوویسکی[۱] و همکارانش مدل AlexNet را ارائه دادند. این مدل توانست در سخت‌ترین چالش ImageNet با عنوان چالش تشخیص دیداری بزرگ مقیاس (ILSVRC) برنده شود و به موفقیت بزرگی در آن زمان دست پیدا کند. این مدل میزان خطا را ۲۶ درصد به ۱۵ درصد کاهش داد. این یک پیشرفت قابل توجه در تشخیص و طبقه‌بندی در بینایی ماشین محسوب می‌شد. این اتفاق نقطه‌ی از تاریخ است که علاقه به یادگیری عمیق به سرعت افزایش پیدا کرد. معماری AlexNet در شکل ۲ـ ۴۱ نشان داده شده است.

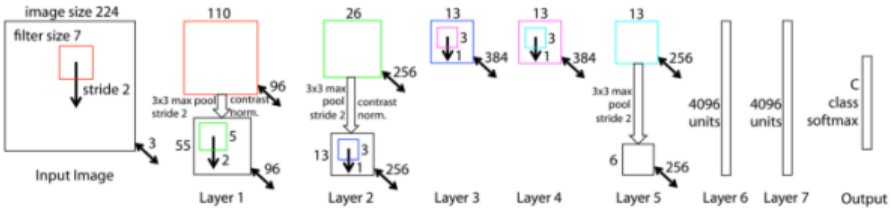


شکل ۲ـ ۴۱ معماری AlexNet

[۱] Alex Krizhevesky

ZFNet ۳.۵.۹.۲

Znet نسخه بهبودیافته از AlexNet بوده که در سال ۲۰۱۳ توسط زایلر[۱] و همکارنش ارائه شد. این معماری در مقایسه با AlexNet اندازه فیلتر را در لایه اول از ۱۱*۱۱ به ۷*۷ کاهش و از گام ۲ به جای ٤ استفاده کرد که منجر به کاهش قابل توجه تعداد پارامترها و افزایش دقت با استخراج ویژگی‌های متمایزتر شد. دلیل اصلی محبوبیت این معماری، درک بهتر از نحوه کار CNN بود. معماری این روش در شکل ۲ـ ٤۲ قابل مشاهده است.



شکل ۲ـ ٤۲ معماری AlexNet

چالش‌های CNN ۲.۹.۶

هرچند که CNNهای عمیق عملکرد خوبی را در پردازش داده‌های دارای ساختار مکانی مشخص و مشبکی کسب کرده‌اند، اما در طول فرآیند آموزش این شبکه‌ها برخی از چالش‌ها وجود دارند که در زیر فهرست شده‌اند:

- CNNهای عمیق عموما همانند جعبه سیاه بوده، از همین‌رو فاقد تفسیر و توضیحات می‌باشند. بنابرین، گاهی اوقات بررسی آن‌ها دشوار است.
- آموزش CNN در داده‌های تصاویر نویزدار زیاد می‌تواند سبب افزایش خطای دسته‌بندی شود. افزودن مقدار کمی نویز تصادفی در تصویر ورودی می‌تواند به فریب شبکه دام زند، به‌گونه‌ای که دسته‌بندی متفاوتی را به‌همراه داشته باشد.
- CNNهای عمیق مبتنی بر یادگیری نظارتی بوده، در همین راستا برای یادگیری درست نیاز به داده‌های بزرگ دارند.

[۱] Zeiler

- انتخاب ابرپارامتر مناسب بر عملکرد CNN تاثیر بسیار زیادی دارد.
- آموزش کارآمد CNN به منابع سخت افزاری قدرتمندی همانند GPUنیاز دارد.

خلاصه فصل

- **یادگیری در شبکه عصبی با استفاده از تغییر وزن‌های متصل به نرون‌ها اتفاق می‌افتد.**
- **ساده‌ترین شکل یک شبکه عصبی پرسپترون بوده و یک دسته‌بند دودویی است.**
- **شبکه‌های پیش‌خور عمیق، به دلیل این‌که برای ورودی‌ها هیچ محدودیتی قائل نیست، یکی از پرکابردترین شبکه‌ها در یادگیری عمیق می‌باشد.**
- **تابع فعال‌ساز نقش مهم و کلیدی در معماری یک شبکه عصبی دارد.**
- **تابع فعال‌سازی در جهت این‌که کدام یک از نرون‌ها باید فعال یا غیرفعال باشند، تصمیم‌گیری می‌کند.**
- **تابع زیان تعیین‌کننده این هستند که، شبکه آموزش داده شده تا چه اندازه به معیار ایده‌آل نزدیک است.**
- **مرحله مهم در زمان ساخت یک شبکه عصبی در جهت دستیابی به بهترین عملکرد، وزن‌دهی اولیه است.**
- **بهینه‌سازها در شبکه‌های عمیق، از طریق بروزرسانی وزن‌ها در شبکه سعی در به حداقل رساندن تابع زیان دارند.**
- **گرادیان کاهشی یکی از متداول‌ترین الگوریتم‌های بهینه‌سازی در شبکه‌های عصبی عمیق است.**

- الگوریتمی که برای یادگیری وزن‌ها در شبکه استفاده می‌شود، الگوریتم پس‌انتشار نام دارد. این الگوریتم در دو فاز جلورو و عقبگرد انجام می‌شود.
- آموزش شبکه‌های عمیق چالش‌هایی همانند: مشکل محو و انفجار گرادیان، بیش‌برازش و اندازه مجموعه آموزشی را دارد.
- منظم‌سازی رویکردی کاربردی در جهت جلوگیری از بیش‌برازش است.
- شبکه‌های عصبی همگشتی، با استخراج ویژگی‌ها به‌صورت خودکار سبب تحولی در پردازش تصویر شدند.
- شبکه‌های عصبی همگشتی، نقش مهمی در تاریخچه یادگیری عمیق دارند.
- یک شبکه عصبی همگشتی از سه لایه، همگشت، ادغام و تماما متصل تشکیل شده است.
- وظیفه اصلی لایه همگشت، شناسایی ویژگی‌های یافت شده در مناطق محلی تصویر ورودی است. از لایه همگشت، با عنوان لایه استخراج ویژگی یاد می‌شود.
- لایه ادغام، وظیفه کاهش ابعاد نقشه ویژگی را عهده‌دار است.
- مرحله طبقه‌بندی در شبکه‌های عصبی همگشتی در لایه تماما متصل اتفاق می‌افتد.
- شبکه‌های عصبی بازگشتی برای پردازش داده‌های متوالی طراحی شده‌اند.

پرسش‌های مروری

۱. لایه‌های مختلف در شبکه‌های پیش‌خور عمیق را نام برده و کابرد هر یک را شرح دهید.

۲. تعداد نرون‌ها در لایه ورودی شبکه چگونه تعیین می‌شود؟

۳. تعداد نرون‌ها در لایه پنهان چگونه تعیین می‌شود؟

٤. تعداد نرون‌ها در لایه خروجی چگونه تعیین می‌شود؟

٥. چند نمونه از توابع فعال‌سازی در یادگیری عمیق را نام ببرید؟

٦. در مقداردهی وزن‌های شبکه اگر همه مقادیر با صفر مقداردهی شوند، چه مشکلی به‌وجود خواهد آمد؟

۷. دلیل استفاده از گرادیان کاهشی تصادفی بر گرادیان کاهشی چیست؟

۸. چند نمونه از الگوریتم‌های بهینه‌سازی با نرخ یادگیری انطباقی را نام ببرید؟

۹. محو گرادیان و انفجار گرادیان به چه دلیلی بوجود می‌آیند؟ راه‌های گریز از این مشکلات چیست؟

۱۰. همگرایی بیش‌از حد در شبکه‌های عصبی سبب بروز چه پدیده‌ای می‌شود؟

۱۱. استفاده از یکسان‌سازی دسته‌ای در شبکه‌های عمیق چه کابردی دارد؟

۱۲. از لایه‌گذاری در شبکه‌های عصبی همگشتی به چه علتی استفاده می‌شود؟

۱۳. دو لایه ادغام را نام ببرید؟

۱٤. استخراج ویژگی‌ها در کدام لایه‌های شبکه عصبی همگشتی انجام می‌گیرد؟

۱٥. دلیل استفاده از شبکه‌های عصبی همگشتی در دسته‌بندی تصاویر چیست؟

۱٦. مشکلات مربوط به شبکه عصبی بازگشتی ساده را شرح دهید؟

۱۷. انواع معماری‌های شبکه عصبی بازگشتی را نام برده و کاربرد هر یک نام ببرید؟

۱۸. انواع دروازه‌های تشکیل دهنده در ساختار LSTM را نام برده و وظیفه هر کدام را شرح دهید؟

فصل ۳

یادگیری بازنمایی بدون‌نظارت عمیق:
خودرمزنگارها و مدل‌های مولد

اهداف

- آشنایی با نحوه کار خودرمزنگار و انواع آن
- تفاوت مدل‌های تفکیک‌پذیر با مدل‌های مولد
- آشنایی با انواع مدل‌های مولد عمیق
- دلیل استفاده از یادگیری عمیق در مسائل

۰.۳ مقدمه

موفقیت شگفت‌آور یادگیری عمیق، عمدتا با الگوریتم‌های بانظارت حاصل شده است. در جهت دستیابی به عملکرد خوب، برای آموزش این شبکه‌ها نیاز به مجموعه داده‌های برچسب‌دار بسیاری زیاد می‌باشد. مساله‌ای که وجود دارد، بدست آوردن چنین مجموعه داده با مقیاس بزرگ دارای برچسب لزوما قابل دسترسی نیست و فرآیند حاشیه‌نویسی زمان‌بر و نیاز به دانش دامنه از موضوع دارد.

علیرغم موفقیت یادگیری بانظارت در سال‌های اخیر، در همان روزهای آغازین دوره یادگیری عمیق، هینتون و همکارش نتایج قابل قبولی را در کاهش ابعاد با استفاده از خودرمزنگار نشان دادند. پس از آن، موفقیت یادگیری بانظارت در مسائل اساسی، مانند تشخیص گفتار و طبقه‌بندی تصویر، علاقه جامعه تحقیقاتی را به سوی این نوع از یادگیری سوق داد، در حالی که یادگیری بدون‌نظارت تا حدی نادیده گرفته شد.

اخیراً، علاقه مجدد به یادگیری بدون‌نظارت با معرفی دو مدل مولد عمیق جدید به‌ویژه، شبکه مولد تخاصمی و شبکه خودرمزنگار متغیر بوجود آمد. پیش‌بینی می‌شود که یادگیری بدون‌نظارت نقش مهمی را در آینده یادگیری عمیق داشته باشد. ذکر این نکته خالی از لطف نیست که، یادگیری بدون‌نظارت هنوز هم یک زمینه بسیار چالش‌برانگیز است و در بسیاری از مسائل، محققین بیشتر ترجیحشان این است تا از رویکردهای بانظارت استفاده کنند. با این همه، جالب است بدانیم که روش انسان در برابر واکنش به مشاهدات غیرمنتظره جهان، تا حد زیادی شباهت به یادگیری بدون‌نظارت دارد. در آخر این‌که، به عقیده یان‌لکان[۱]: "انقلاب بعدی هوش مصنوعی، بدون‌نظارت خواهد بود".

۱.۳ یادگیری فعال و یادگیری بازنمایی بدون‌نظارت

یادگیری بازنمایی یک دامنه فعال در یادگیری ماشین است و هدف آن بدست آوردن نمایشی مفید از داده‌هاست، و از آن جهت که می‌تواند به عنوان یادگیری ویژگی‌های

[۱] https://engineering.nyu.edu/news/revolution-will-not-be-supervised-promises-facebooks-yann-lecun-kickoff-ai-seminar

معنادار تفسیر شود، به عنوان **یادگیری ویژگی** نیز شناخته می‌شود. انگیزه این روش‌ها در بیشتر مواقع، یافتن نمایش خوب داده‌ها برای استفاده در مسائل طبقه‌بندی و رگرسیون به‌روشی خودکار و تعیم‌یافته است. کارهای اخیر در حوزه یادگیری عمیق، متمرکز بر توسعه الگوریتم‌هایی است که می‌توانند به‌طور خودکار بازنمایی‌های اساسی یا ویژگی‌ها را از خود داده‌ها یاد بگیرند. به عنوان مثال شبکه‌های عصبی عمیق را می‌توان به عنوان یادگیری بازنمایی بانظارت مشاهده کرد؛ یادگیری سلسله‌مراتبی از نمایش‌های توزیع شده که همان لایه‌های پنهان می‌باشند و از بازنمایی‌های سطح پایین به سطح بالادر جهت رسیدن به هدف نهایی نظارت شده ایجاد می‌شوند. در مورد الگوریتم‌های بدون‌نظارت، این ویژگی‌ها از داده‌های بدون برچسب یاد گرفته می‌شوند. از انوع مختلف الگوریتم‌های شناخته شده یادگیری بازنمایی بدون‌نظارت، می‌توان به خودرمزنگارها و ماشین‌های بولتزمن اشاره کرد.

اکثر مدل‌های یادگیری ماشین بانظارت برای آموزش و بدست آوردن نتایج خوب به دادهای زیادی احتیاج دارند. در بیشتر موارد، در اختیار محققین علوم داده یک مجموعه داده بزرگ و بدون برچسب قرار می‌گیرد و از آن‌ها خواسته می‌شود تا مدل‌هایی با عملکرد خوب آموزش دهند. برچسب‌گذاری دستی حجم بسیار زیاد داده‌ها یک چالش بزرگ را بوجود می‌آورد. اینجاست که یادگیری فعال به کمک آن‌ها می‌آید. یادگیری فعال، دامنه‌ای از یادگیری ماشین است که با وضعیتی سروکار دارد تا مشکل داده‌های بدون برچسب را با اولویت‌بندی داده‌هایی که باید در جهت تاثیرگذاری بیشتر مدل برچسب‌گذاری شوند، مورد استفاده قرار دهد. به عبارت دیگر، یادگیری فعال بهینه‌سازی نقاط داده‌ای که باید برای برچسب‌گذاری و آموزش مدل انتخاب شوند، می‌باشد.

با وجود پیشرفت اخیر در یادگیری بازنمایی بانظارت، سوالی بوجود می‌آید؛ آیا می‌توان از داده‌های بدون برچسب و بدون هیچ‌گونه ناظری بازنمایی "قدرتمندی" از این رویکرد بدست آورد؟ می‌توان به این سوال پاسخی مثبت داد، چرا که چیزهای زیادی (ویژگی‌های مفید) را می‌توان از داده‌های بدون برچسب، به‌ویژه داده‌های با ابعاد بالا همانند تصویر (داده‌های تصویر به‌طور کلی با مقادیر پیکسل نمایش داده می‌شوند و بیشتر معانی در آن پنهان می‌شود) آموخت و می‌توان حتی قبل از استفاده سناریو یادگیری فعال برای بدست آوردن برچسب در جهت یادگیری نظارتی، از روش‌های بدون‌نظارت برای یادگیری بهتر نمایش داده‌های بدون برچسب (بدست آوردن آن اغلب بسیار ساده‌تر

است) بهره‌مند شویم. روش‌های زیادی برای کار با داده‌های بدون برچسب وجود دارد. در این فصل، ما بر این فرضیه تمرکز می‌کنیم که می‌توان از داده‌های بدون برچسب برای یادگیری بازنمایی خوب استفاده کرد. یادگیری بدون‌نظارت یکی از حوزه‌های تحقیقاتی فعال در یادگیری ماشین می‌باشد و حل آن گام مهمی در جهت توسعه هوش مصنوعی عمومی تلقی می‌شود. در ادامه، به بررسی رویکردهای بدون‌نظارت در یادگیری عمیق خواهیم پرداخت.

۲.۳ خودرمزنگار

در شبکه‌های عصبی مصنوعی، پس‌انتشار برای بهبود یادگیری بازنمایی توسعه داده شد. فرآیند پس‌انتشار شامل تنظیم مجدد وزن‌ها وابسته به خروجی‌های مورد انتظار است. در دهه ۱۹۸۰، خودرمزنگارها معرفی شدند تا بدون معلم، پس‌انتشار را انجام دهند. یعنی خودرمزنگارها روشی را برای یادگیری خودکار ویژگی‌ها از داده‌های بدون برچسب ارائه می‌دهند که امکان یادگیری بدون‌نظارت را فراهم می‌کند. در شبکه‌های عصبی که تاکنون مورد بحث قرار گرفتند، داده‌های دارای برچسب مورد نیاز بودند تا به عنوان مثال‌های آموزشی ضروری برای تنظیم‌ـ دقیق پس‌انتشار عمل کنند، چراکه از این برچسب‌ها برای تنظیم مجدد پارامترها استفاده می‌شود. با این حال، خودرمزنگارها فرصتی برای یادگیری، بدون وابستگی به داده‌های دارای برچسب را فراهم می‌کنند.

یک شبکه عصبی خودرمزگذار با تنظیم مقادیر خروجی هدف برابر با مقادیر ورودی، پس‌انتشار را انجام می‌دهد و بدین ترتیب خودرمزنگار آموزش داده می‌شود تا اختلاف بین داده‌ها و بازسازی آن را به حداقل برساند (یعنی تفاوت بین بردار واقعی خروجی و بردار خروجی مورد انتظار، که در آن خروجی مورد انتظار همان بردار ورودی است). در نتیجه، خودرمزنگارها قادر به یادگیری بدون معلم (ناظر) هستند.

به‌طور کلی، خودرمزنگار نوعی شبکه عصبی مصنوعی می‌باشد که در تلاش است با دریافت ورودی تا آن‌جایی که امکان دارد خروجی شبیه به آن را تقلید کند. هدف آن، بازسازی ورودی اصلی به دقیق‌ترین شکل ممکن است (به جای تلاش برای پیش‌بینی یک نتیجه خاص، سعی می‌کنند ورودی‌های خودشان را بازسازی کنند). به عبارت دیگر، ورودی را رونوشت می‌کند. معمولاً خودرمزنگارها به روشی محدود می‌شوند که تنها به

آن‌ها اجازه رونوشت می‌دهد. از آنجا که مدل مجبور است اولویت بندی کند که کدام یک از ویژگی‌های ورودی باید رونوشت شوند، اغلب ویژگی‌های مفید داده را می‌آموزد. در ظاهر بنظر می‌رسد که تهیه رونوشتی از ورودی در خروجی (خروجی شبکه با ورودی برابر است)، یک کار یادگیری بی‌اهمیت می‌باشد، اما خواهیم دید که اینگونه نیست. ایده این است که علاوه بر آموزش شبکه برای رونوشتی از ورودی در خروجی، برخی محدودیت‌ها نیز اعمال شود.

یک محدودیت معمول این است که یک لایه گلوگاه در وسط شبکه قرار گیرد. به این لایه گلوگاه ابعادی بسیار کوچک‌تر از ورودی و خروجی داده می‌شود. این امر، شبکه را مجبور می‌کند که تنها ورودی را در خروجی منتقل نکند و نتواند تمام اطلاعات موجود در ورودی را در خود داشته باشد. لایه گلوگاه را می‌توان محاسبه یک نسخه فشرده‌شده از ورودی دانست. این نسخه فشرده شده ورودی، بازنمایی، رمزگذاری یا گاهی اوقات به سادگی رمز گفته می‌شود.

همان طور که دیدیم این ایجاد محدودیت در شبکه می‌تواند نمایانگر ساختار جالب داده‌ها باشد. این روش امکان کشف بازنمودهای داخلی داده‌ها را فراهم می‌کند که به ویژگی‌های کم‌تری متکی هستند. به عنوان مثال، در تشخیص چهره، ممکن است هر پیکسل از تصویر در لایه ورودی نشان داده شود. این داده‌ها در لایه پنهان به ویژگی‌هایی مانند "دهان کوچک" یا "چشمانی درشت" فشرده می‌شوند. یعنی، داده‌های ورودی صورت را می توان با استفاده از داده‌های کم‌تر از آنچه در تصویر داده شده توصیف کرد. سپس، داده‌های فشرده شده می‌توانند به منظور بازنمایی مجدد داده‌های ورودی در لایه خروجی، فشرده نشوند و اجازه دهند تصویر چهره کاملاً از ویژگی‌های آموخته شده بازسازی شود.

این نوع یادگیری در خودرمزنگار، یادگیری **خودنظارت‌شده** نیز نامیده می‌شود، چراکه سیستم در واقع با استفاده از یک تابع هزینه و پس‌انتشار، به روشی نظارت شده آموزش می‌بیند، اما به داده‌های دارای برچسب نیازی ندارد. در نتیجه، هنگامی که شما فقط به داده‌های دارای برچسب بسیار کم اما مقادیر زیادی از داده‌های بدون برچسب دسترسی داشته باشید، استفاده از خودرمزنگار یک روش معمول برای آموزش شبکه

می‌باشد. از خودرمزنگارها به طور سنتی برای پیش‌آموزش استفاده می‌شود: یعنی، ابتدا خودرمزنگار را روی یک مجموعه داده بدون برچسب آموزش داده و سپس چند لایه کاملاً متصل به هم اضافه و وزن‌های اصلی را ثابت[۱] می‌کنید. سپس آخرین لایه‌ها را روی مجموعه کوچک‌تری از داده های دارای برچسب آموزش می‌دهید. به این ترتیب، از یک خودرمزنگار به عنوان پایه آموزش یک طبقه‌بندی استفاده می‌شود.

۱.۲.۳ معماری خودرمزنگار

خود رمزگذارها نوع خاصی از شبکه‌های عصبی پیش‌خور هستند که ورودی آن‌ها همان خروجی است. در این شبکه ورودی به یک رمز در ابعاد پایین‌تر فشرده و سپس خروجی حاصل از این بازنمایی بازسازی می‌شود. رمز یک "خلاصه" یا "فشرده‌سازی" ورودی است، که بازنمایی فضای‌ـ نهفته[۲] نیز نامیده می‌شود. همان‌طور که بیان شد، ایده پشت این شبکه‌ی عصبی بازسازی داده‌های ورودی با کم‌ترین کژریختی ممکن می‌باشد.

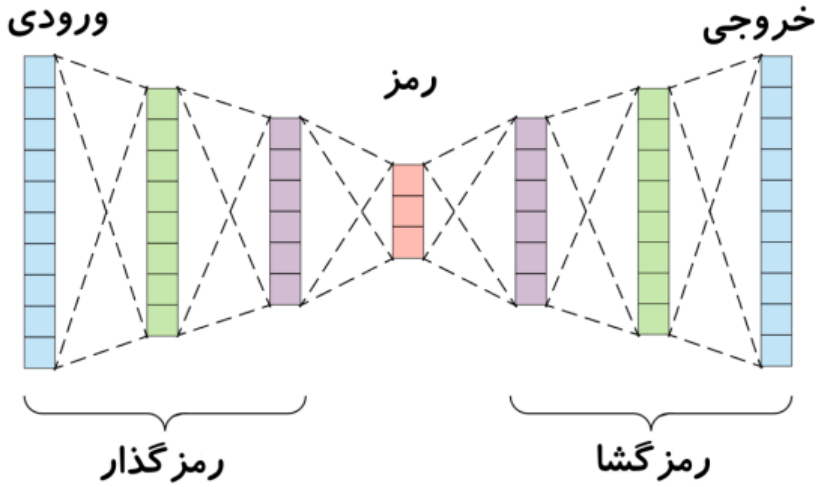
یک خودرمزنگار به ترتیب از سه مولفه تشکیل شده است: رمزگذار، رمز و رمزگشا. رمزگذار ورودی را فشرده و رمز را تولید می‌کند، رمزگشا ورودی را براساس رمز بازسازی می‌کند (شکل ۳ـ ۱). رمزگذار و رمزگشا هر دو شبکه عصبی پیش‌خور هستند و در اکثر مواقع به‌صورت متقارن در ساختار خودرمزنگار قرار می‌گیرند. رمز، تک‌لایه‌ای با ابعاد متناسب با انتخاب ما می‌باشد (تعداد نرون‌ها در لایه رمز، یک ابرپارامتر است).

حال نگاهی دقیق‌تر به ساختار خودرمزنگار می‌اندازیم. ابتدا ورودی از یک رمزگذار که شبکه عصبی مصنوعی کاملا متصل است، عبور کرده تا رمز را تولید کند. سپس، رمزگشا که ساختاری مشابه رمزگذار دارد و متقارن با آن می‌باشد، تنها با استفاده از رمز خروجی را تولید می‌کند. هدف، بدست آوردن خروجی یکسان با ورودی داده شده به شبکه می‌باشد. اجزای رمزگذار و رمزگشا می‌توانند از هر نوعی باشند. در ساده ترین حالت، می‌توان از یک شبکه عصبی تنها با یک لایه مخفی استفاده کرد. با این حال، نشان داده شده که شبکه‌های عمیق‌تر در مقایسه با انواع کم عمق خود قادر به بازنمایی

---

[۱] freeze
[۲] latent-space representation

بهتری هستند. علاوه بر این، معمولاً استفاده از لایه‌های همگشتی برای کارهای پردازش تصویر که به عنوان خودرمزنگار همگشتی شناخته شده هستند، ترجیح داده می‌شود. در ساده‌ترین شکل ممکن، یک خودرمزنگار شبکه‌ای سه لایه می‌باشد. یعنی، یک شبکه عصبی با یک لایه پنهان که به آن خودرمزنگار معمولی[۱] یا ساده گفته می‌شود. این خودرمزنگار گاهی اوقات در مقایسه با سایر خودرمزنگارها عملکرد پایین‌تری دارد. انواع مختلفی از خودرمزنگارها وجود دارند، که در بخش‌های بعد آن‌ها را مورد بررسی قرار می‌دهیم.



شکل ۳ـ۱ معماری خودرمزنگار

۲.۲.۳ پارامترهای خودرمزنگار

در هنگام آموزش یک خودرمزنگار، با چندین پارامتر روبه‌رو خواهید شده که بر عملکرد مدل تاثیرگذار و باید از قبل تنظیم شوند. این پارامترها مربوط به معماری (اندازه رمز و تعداد لایه‌ها) و پارامترهای آموزش (تابع هزینه) هستند. در این بخش، این پارامترها مورد بحث قرار می‌گیرند.

[۱] Vanilla autoencoder

- **اندازه رمز:** تعداد گره‌ها در لایه میانی یک خودرمزنگار است. همیشه بهتر است تعداد گره‌های موجود در این لایه کمتر از اندازه ورودی باشد. اندازه کوچکتر لایه رمز منجر به فشرده‌سازی بیشتر می‌شود. با توجه به اندازه رمز، خودرمزنگارها را می‌توان به دو گروه طبقه بندی کرد: خودرمزنگارهای ناقص و مافوق‌کامل.
- **تعداد لایه‌ها:** می‌توان به اندازه مورد نیاز تعداد لایه‌های رمزگذار و رمزگشا را تعیین کرد. بسته به تعداد لایه‌های مخفی، خودرمزنگارها را می توان به عمیق و کم‌عمق تقسیم کرد. همچنین می‌توان تعداد گره‌ها در این لایه‌ها را تعیین کرد. به‌طور معمول، با افزایش تعداد لایه‌ها، تعداد گره‌ها کاهش می‌یابد.
- **تابع هزینه:** اثربخشی آموزش شبکه‌های عصبی را ارزیابی می‌کند. نمره‌ای را برمی‌گرداند که نشان‌دهنده عملکرد خوب شبکه است. در مورد خودرمزنگارها، میزان خوب بودن بازسازی را اندازه گیری می‌کند. تابع هزینه مورد استفاده در خودرمزنگار معمولا خطای میانگین مربع یا آنتروپی متقاطع می‌باشد.

۳.۲.۳ خودرمزنگار چگونه کار می‌کند؟

هر ورودی در خودرمزنگار با بردار $x \epsilon \mathbb{R}^n$ و ابعاد n که برابر اندازه ورودی می‌باشد، نمایش داده می‌شود. خودرمزنگار ورودی را می‌گیرد و ابتدا آن را به بازنمایی پنهان (رمز) توسط رمزگذار نگاشت می‌کند که می‌تواند به عنوان یک تابع مشاهده شود:

$$h = f_\theta(x) = \sigma(xW + b)$$

جایی که $\theta = \{W, b\}$؛ $W \epsilon \mathbb{R}^{mxn}$ ماتریس وزن؛ $b \epsilon \mathbb{R}^m$ بردار بایاس؛ $f_\theta(x)$ رمزگذار و σ تابع فعال‌سازی می‌باشد.

بازنمایی پنهان توسط رمزگشا که لایه خروجی $\acute{x} \epsilon \mathbb{R}^n$ را با همان ابعاد ورودی n تولید می‌کند، بار دیگر نگاشت می‌شود. این فرآیند را می‌توان به عنوان یک تابع نوشت:

$$\acute{x} = g_{\acute{\theta}}(h) = \sigma(h\acute{W} + \acute{b})$$

جایی که $\acute{\theta} = \{\acute{W}, \acute{b}\}$ و $g_{\acute{\theta}}(h)$ رمزگشا می‌باشد.

۴.۲.۳ خودرمزنگار انقباضی[۱]

خودرمزنگار انقباضی، ارائه یک رویکرد منظم‌سازی است که شبکه را مجبور به یادگیری بازنمایی‌های مفیدی می‌کند که نسبت به تغییرات کوچک در ورودی (داده‌ها) حساسیت کم‌تری (قوی‌تر هستند) دارند. این امر با افزودن یک جریمه برای تابع زیان حاصل می‌شود. این جریمه، مجموع عناصر مربع شده ماتریس ژاکوبین از مشتقات جزئی مربوط به تابع رمزگذار است.

$$L\left(x, g\left(f(x)\right)\right) + \varepsilon(h)$$

جایی که $g(h)$ خروجی رمزگشا، $h = f(x)$ خروجی رمزگذار و $\varepsilon(h)$ مجموع عناصر مربع شده ماتریس ژاکوبین به صورت زیر می‌باشد:

$$\varepsilon(h) = \lambda \left\| \frac{\partial f(x)}{\partial x} \right\|_F^2$$

جایی که λ یک ابرپارامتر برای کنترل قدرت منظم‌سازی استفاده می‌شود.

نتیجه نهایی، کاهش حساسیت بازنمایی آموخته شده نسبت به ورودی‌های آموزشی است. به عبارت دیگر، خودرمزنگار انقباضی آموزش می‌بیند تا در برابر اغتشاش ورودی خود مقاومت کند.

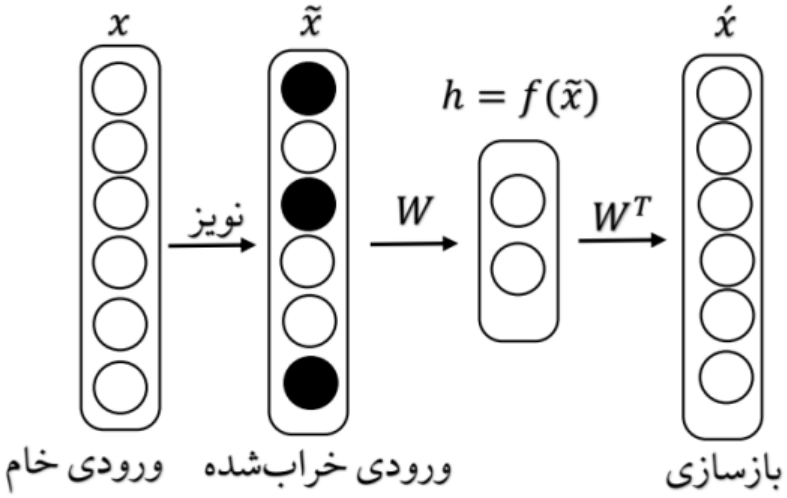
۵.۲.۳ خودرمزنگار حذف‌نویز[۲]

در خودرمزنگار حذف نویز به‌جای افزودن جریمه به تابع زیان، می‌توان یک خودرمزنگار بدست آورد تا با تغییر خطای بازسازی تابع زیان، چیزهای مفیدی را بیآموزد. این کار را می‌توان با افزودن عمدی مقداری نویز به لایه ورودی انجام داد. با وارد کردن این مقادیر نویزی خودرمزنگار حذف‌نویز، رونوشتی نویزدار از ورودی را ایجاد می‌کند. این عمل

[۱] Contractive Autoencoder
[۲] Denoising autoencoder

کمک می‌کند تا از رونوشت ورودی و خروجی بدون یادگیری ویژگی‌های داده‌ها توسط خودرمزنگار جلوگیری و شبکه مجبور به آموختن قوی‌ترین ویژگی‌ها شود.

ورودی در این شبکه، نسخه خراب‌شده (نویزدار) $\tilde{x} \in \mathbb{R}^n$ از ورودی اصلی $x \in \mathbb{R}^n$ است. این خودرمزنگار، به سادگی ورودی را در خروجی کپی نمی‌کند بلکه داده‌ها را از نویز پاک کرده و پس از آن، ورودی را از نسخه خراب تولید می‌کند (شکل ۳ـ۲).



شکل ۳ـ۲ معماری خودرمزنگار

تابع زیان، خطا را در ورودی خراب‌شده به حداقل می رساند نه در ورودی اصلی و به‌صورت زیر می‌باشد:

$$L\left(x, g\left(f(\tilde{x})\right)\right)$$

جایی که $g\left(f(\tilde{x})\right)$ خروجی رمزگشا، $f(\tilde{x})$ خروجی رمزگذاری شده از ورودی خراب‌شده است.

۶.۲.۳ خودرمزنگار ناقص[۱]

یکی از راه‌های یادگیری بازنمایی‌های مفید با خودرمزنگارها، محدود کردن اندازه رمز آن خودرمزنگار است. در این حالت، خودرمزنگار مجبور به استخراج ویژگی‌های برجسته از داده‌ها می‌شود. خودرمزنگارهای ناقص، دارای اندازه رمز کوچک‌تری از اندازه ورودی هستند. همین امر، کمک می‌کند تا ویژگی‌های مهم و برجسته از داده‌ها بدست آید. این خودرمزنگارها برای بازنمایی ویژگی‌های مفید داده‌ها و کاهش ابعاد طراحی شدند. فرآیند یادگیری به سادگی با حداقل‌رساندن تابع زیان توصیف می‌شود:

$$L\left(x, g\left(f(x)\right)\right)$$

۷.۲.۳ خودرمزنگار پراکنده[۲]

برخلاف خودرمزنگار ناقص که اندازه رمز آن از ابعاد ورودی کوچک‌تر است، خودرمزنگار پراکنده معمولاً مافوق‌کامل هستند. با این حال، آن‌ها هنوز هم می‌توانند ویژگی‌های مهم را از داده‌ها کشف کنند.

معماری مافوق‌کامل خودرمزنگار پراکنده، اجازه داشتن تعداد بیشتری از واحدهای پنهان را در لایه رمز می‌دهد. اما این امر مستلزم این است که برای ورودی داده شده، برای هر نورون پنهان، مقدار تابع فعال‌سازی متوسط باید نزدیک به صفر باشد (اگر از تابع فعال‌سازی سیگموید استفاده شود یا مقدار ۱ ــ هنگامی که از تانژانت هذلولوی استفاده شود). اگر خروجی نزدیک به ۱ باشد، نورون فعال و در غیر این صورت، غیر فعال در نظر گرفته می‌شود. حال، سوالی بوجود می‌آید. هدف از داشتن واحدهای پنهان با بیشتر از صفر چیست؟ ایده این است که یک نورون فقط برای بخش کوچکی از نمونه‌های آموزشی فعال شود. از آنجایی که نمونه‌ها دارای ویژگی‌های مختلفی هستند، بنابراین فعال‌سازی نورون‌ها نباید در همه نورون‌ها به یک شکل انجام گیرد و باید

[۱] Undercomplete Autoencoder
[۲] Sparse autoencoder

هماهنگ باشد. هدف، نمایش پنهان با صفرهای زیاد و تنها چند عنصر غیرصفر تا برجسته‌ترین ویژگی‌ها نشان داده شوند.

فرآیند آموزش در این خودرمزنگار، با اضافه کردن یک جریمه به تابع زیان در لایه رمز صورت می‌گیرد:

$$L\left(x, g\big(f(x)\big)\right) + \varepsilon(h)$$

جایی که $g(h)$ خروجی رمزگشا، $h = f(x)$ خروجی رمزگذار و $\varepsilon(h)$ یک جریمه پراکندگی با تابعی لگاریتمی به صورت زیر می‌باشد:

$$\varepsilon(h) = \sum_{j=1}^{s} KL(p||\acute{p}_j)$$

جایی که p پارامتر پراکندگی و به‌طور معمول یک مقدار کوچک نزدیک به صفر است، $\acute{p}_j$ فعال‌سازی متوسط واحد پنهان j که تقریبی از p می‌باشد، s تعداد نرون‌ها در لایه پنهان، KL واگرایی کولبک‌ـ لیبلر بین یک متغیر تصادفی برنولی با میانگین p و یک متغیر تصادفی برنولی با میانگین $\acute{p}$ است:

$$KL(p||\acute{p}_j) = plog\frac{p}{\acute{p}_j} + (۱ - p)log\frac{۱ - p}{۱ - \acute{p}_j}$$

۸.۲.۳ خودرمزنگار همگشتی

خودرمزنگار همگشتی یک نوع شبکه عصبی همگشتی است که به عنوان ابزاری پیشرفته در یادگیری بدون نظارت برای فیلترهای همگشت استفاده می‌شود. خودرمزنگار همگشتی از منظر دیگری به وظیفه تعریف فیلتر می پردازد؛ به جای فیلترهای همگشتی با مهندسی دستی، به مدل اجازه می‌دهیم فیلترهای بهینه را بیاموزد که خطای بازسازی را به حداقل می‌رساند. وقتی این فیلترها یاد گرفته شدند، می‌توانند برای استخراج ویژگی‌ها روی هر ورودی اعمال شوند. بنابراین، از این ویژگی‌ها می‌توان برای انجام هر کاری که نیاز به نمایش فشرده ورودی دارد، همانند طبقه‌بندی استفاده کرد.

از این نوع شبکه‌ها به‌طور کلی در کار بازسازی تصویر استفاده می‌شود تا با یادگیری فیلترهای بهینه، خطای بازسازی به حداقل برسد. خودرمزنگار همگشتی، یاد می‌گیرد ورودی را در مجموعه‌ای از سیگنال‌های ساده رمزگذاری کرده و سپس ورودی را از آن‌ها بازسازی کند. در این نوع از خودرمزنگار، لایه‌های رمزگذار به عنوان لایه‌ی همگشت و لایه‌های رمزگشا به عنوان لایه‌ی واهمگشت[۱] نامیده می‌شوند.

۳.۳ مدل‌های تفکیک‌پذیر و مولد

یکی از اهداف یادگیری ماشین، توسعه الگوریتم‌های آماری است که با استفاده از مشاهدات گذشته، امکان استنتاج بر وضعیت داده‌های آینده را فراهم می‌کند. در بسیاری از کاربردها، مانند طبقه‌بندی یا رگرسیون، توصیف فرم متغیر هدف y به عنوان تابعی از متغیر پیش‌بینی کننده x کافی است. با توجه به داشتن یک مجموعه داده D، هدف تعیین پارامترهای θ است، به این صورت که $p_\theta(y|x)$ مقدار y را برای یک داده آزمون داده شده، $x = x^*$ را به درستی استنباط می‌کند. این احتمال شرطی $p_\theta(y|x)$ یک مدل تفکیک‌پذیر می‌باشد.

در مقابل، یک مدل مولد (تولیدی) توزیع احتمالی مشترک $p_\theta(y, x)$ را بر روی تمام متغیرها نشان می‌دهد. مزیت این کار، این است که می‌توانیم از مدل‌های مولد برای تولید نمونه‌های مشابه با نمونه‌های تهیه‌شده از توزیع واقعی استفاده کنیم. این توانایی نمونه‌برداری در جهت افزایش داده‌ها در سناریوهای تصمیم‌گیری بسیار مفید است.

مدل‌های مولد[۲] و تفکیک‌پذیر[۳] دو رویکرد متفاوت هستند که به‌طور گسترده در مسائل طبقه‌بندی مورد مطالعه قرار می‌گیرند و در جهت رسیدن به نتیجه نهایی مسیری کاملا متفاوت را از یک‌دیگر انتخاب می‌کنند. کار مدل‌های تفکیک‌پذیر در مقابل مدل‌های مولد ساده‌تر است، چرا که اگر داده‌هایی با دسته‌های مختلف به آن نشان داده شود باید بتواند بین آن‌ها تفاوت قائل شود و اگر داده‌های مناسب وجود داشته باشد، این رویکرد محبوب‌تر و کارایی بهتری را به نمایش می‌گذارد. در مقابل، مدل‌های مولد

---

[۱] deconvolution: از همگشت در آوردن

[۲] Generative

[۳] Discriminative

کار سخت‌تری در پیش‌رو دارند، چرا که باید توزیعی از داده‌ها را بدست آورده، درک کند و پس از آن به انجام طبقه‌بندی بپردازد. همچنین، **از آن جهت که این مدل‌ها توزیع داده‌ها را آموخته‌اند، توانایی ایجاد داده‌های مشابه با داده‌های آموزشی را نیز دارند.** به عنوان مثالی از این دو رویکرد، فرض کنید متغیر تصادفی x یک تصویر و y برچسبی است که محتوای تصویر را توصیف می‌کند. از یک مدل تفکیک‌پذیر $p_\theta(y|x)$ تنها می‌توان برای استنباط برچسب یک تصویر جدید استفاده کرد. در مقابل، مدل مولد می‌تواند برای نمونه‌برداری در جهت تولید نمونه‌ای مشابه با تصویر حاوی برچسب y استفاده شود. یک مدل مولد، با گرفتن احتمال مشترک داده‌های ورودی و برچسب $p_\theta(y,x)$ به‌طور هم‌زمان، می‌تواند نمونه‌های مشابه را تولید کند. به عنوان مثال، با در نظر گرفتن تصاویر به عنوان داده‌های ورودی، هر نمونه (تصویر) دارای هزاران بعد (پیکسل) است و وظیفه مدل مولد، بدست آوردن وابستگی‌های بین پیکسل‌ها می‌باشد.

مدل‌های مولد را می‌توان به عنوان یک کلاس از مدل‌ها تعریف کرد که هدف آن‌ها یادگیری نحوه تولید نمونه‌های جدیدی است که به نظر می‌رسد از همان مجموعه داده های آموزشی هستند. در طول مرحله آموزش، یک مدل مولد در تلاش است یک مساله تخمین چگالی را حل کند. در تخمین چگالی، مدل می‌آموزد تا یک تخمین تا حد امکان شبیه به تابع چگالی احتمال غیرقابل مشاهده بسازد. نکته مهم این است که، مدل مولد باید بتواند نمونه‌های جدیدی از توزیع را تشکیل دهد و نه فقط نمونه‌های موجود را رونوشت و ایجاد کند.

مدل‌های مولد مجبورند توزیع‌ها و خصوصیات اساسی داده‌ها را برای بازسازی یا تولید نمونه‌های مشابه کشف و به‌طور کارآمدی یاد بگیرند. می‌توانیم مدل‌های مولد را به عنوان یک ماشین در نظر بگیریم که قادر است هر شی را به عنوان مثال یک اتوموبیل را نگاه کند، و با وارسی کردن از مقدار زیادی نمونه‌های اتومبیل، مدل سرانجام یک طرح تولید را یاد می‌گیرد که چگونه انواع جدیدی از اتومبیل‌ها را با انواع رنگ‌ها، اشکال، ارتفاع، تعداد درها و غیره را بسازد.

اگر مدلی واقعاً قادر به تولید نمونه‌های جدیدی باشد که از پیدایش اشیا دنیای واقعی پیروی می‌کنند، در واقع می‌توان گفت که مفهومی را بدون آموزش یاد گرفته و درک کرده

است. از همین رو این دسته از مدل‌ها، در رده مدل‌های بدون‌نظارت (مدل‌های مولد را می‌توان در رده مدل‌های خودنظارت شده نیز قرار داد) قرار می‌گیرد.

۱.۳.۳ انواع مدل مولد

مدل‌های مولد، به عنوان رویکردی بدون‌نظارت دسته‌بندی می شوند و به‌طور کلی مدل مولد، مدلی است که با در نظر گرفتن تعدادی از نمونه‌های آموزشی که از توزیع $p(x)$ تهیه شده، توانایی یادگیری تخمین $p_\theta(x)$ از چنین توزیعی را دارد. روش‌های تخمین متفاوتی وجود دارد که مدل‌های مولد براساس آن‌ها ساخته می‌شوند. با این حال، با فرض این‌که از درست‌نمایی بیشینه برای تخمین استفاده شود، مدل‌های مولد را می‌توان به‌طور کلی به دو دسته: مدل‌های ضمنی و آشکار تقسیم‌بندی کرد.

درست‌نمایی بیشینه

یکی از روش‌های مناسب برای آموزش یک مدل مولد بر روی یک مجموعه داده آموزشی $X = \{x^{(i)}\}_{i=1}^{N}$ و یافتن پارامترهای مناسب برای توزیع مدل $p_\theta(x)$ استفاده از برآورد درست‌نمایی بیشینه[1] است. ایده پشت این روش، مدل‌سازی تقریب توزیع پیشین داده‌ها از طریق برخی پارامترهای $\theta: p_\theta(x)$ است و سپس انتخاب پارامترهایی که درست‌نمایی را بیشینه می‌کنند.

به عبارت دیگر، برآورد درست‌نمایی بیشینه یک پارامتر θ^* بهینه‌ای را تعیین می‌کند که براساس آن درست‌نمایی برای هر نقطه‌ای از داده X تا جای ممکن زیاد باشد.

$$\theta^* = arg_\theta max \prod_{i=1}^{n} p_\theta(x^{(i)})$$

[1] Maximum Likelihood Estimation

در عمل، برای سادگی محاسبات و پایداری عددی، مطلوب‌تر است که به‌جای استفاده از $p_\theta(x)$ مقدار $log\ p_\theta(x)$ آن استفاده شود.

$$\theta^* = arg_\theta max \sum_{i=1}^{n} log\ p_\theta(x^{(i)})$$

مدل‌های ضمنی

مدل‌های مولد ضمنی، ابزاری برای دستیابی به توزیع $p_\theta(x)$ فراهم نمی‌کند و چگالی احتمال را تخمین نمی‌زنند اما در عوض یاد می‌گیرند که مستقیما نمونه (داده) تولید کنند. در این روش، مقایسه‌ای بین داده‌های واقعی با نمونه‌های تولیدی صورت می‌گیرد. از معروف‌ترین این مدل‌ها، می‌توان به شبکه‌های مولد تخاصمی اشاره کرد.

مدل‌های آشکار

با استفاده از مدل‌های آشکار، می‌توان توزیع $p_\theta(x)$ را از چارچوب مدل بدست آورد. سه کلاس اصلی مدل‌های آشکار: مدل‌های خودبرگشتی[۱]، مدل‌های مبتنی‌بر جریان[۲] و مدل‌های متغیر نهفته احتمالی[۳] هستند.

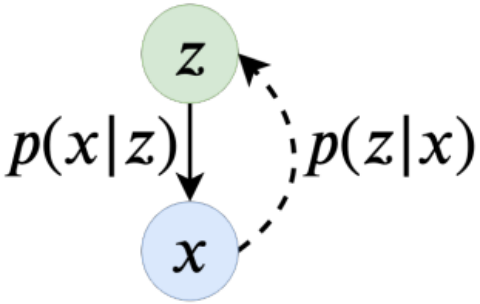
- **مدل‌های خودبرگشتی:** این مدل‌ها تنها با استفاده از قانون زنجیره‌ای احتمالات، خروجی خود را به داده‌های مشاهده شده در گذشته مشروط می‌کنند و نه به داده‌های آینده. به عنوان مثال، هر پیکسل جدید از تصویر، به پیکسل‌های مشاهده شده قبلی بستگی دارد.
- **مدل‌های مبتنی‌بر جریان:** مدل مبتنی‌بر جریان به عنوان دنباله‌ای از تبدیلات وارون[۴] که جریان یکسان‌ساز[۵] نامیده می‌شوند و اجازه جایگزینی مکرر متغیرها را با توجه به قضیه تغییر متغیرها می‌دهد، ساخته می‌شوند. درنتیجه، مدل‌های

---

۱ autoregressive
۲ low-based models
۳ probabilistic latent variable models
۴ invertible transformations
۵ Normalizing Flows

مبتنی بر جریان دقیقاً توزیع واقعی داده‌ها را فرا می‌گیرند و ارزیابی دقیق احتمال را امکان پذیر می‌کنند.

- **مدل‌های متغیر نهفته احتمالی:** این مدل‌ها طبقه گسترده‌ای از مدل‌های آشکار را تشکیل می‌دهند و از متغیرهای کمکی برای توزیع‌های پیچیده که جنبه‌های واقعی‌تری از جهان است، استفاده می‌کنند. یک مدل متغیر نهفته، یک مدل گرافیکی جهت‌دار (شکل ٦ــ١) احتمالی از متغیرهای x که متغیرهای نهفته z را در خود جای داده است. گنجاندن متغیرهای نهفته به ما این امکان را می‌دهد تا وابستگی‌های پنهان در میان متغیرهای مشاهده شده را بدست آوریم و ساختار زیربنایی سازوکار تولید داده را بیآموزیم و از همه مهم‌تر، متغیرهای نهفته می‌توانند یک نمایش جایگزین، با ابعاد کم‌تر برای متغیرهای مشاهده شده ارائه دهند. از همین‌رو، این مدل‌ها یادگیری بازنمایی را انجام می‌دهند. این مدل بر این فرض استوار است که متغیر مشاهده x توسط یک فرآیند تصادفی مبتنی‌بر یک متغیر پیوسته مشاهده نشده z ایجاد می‌شود. به عبارتی دیگر، در ابتدا z نهفته از توزیع پیشین $p(z)$ تولید می‌شود. سپس، x از توزیع شرطی $p(x|z)$ تولید می‌شود. متغیر مشاهده نشده z را می‌توان به عنوان بازنمایی نهفته تفسیر کرد. به‌طور کلی، این مدل‌ها هدفی دوگانه دارند: مدل‌سازی توزیع مشترک $p_\theta(x,z)$ و استنباط توزیع $p_\theta(z|x)$ در جهت یادگیری بازنمایی است. مدل‌های متغیر نهفته، این پتانسیل را دارا هستند که به‌طور خودکار اصول فرآیند تولید را کشف و بازنمایی‌های پنهان قابل تفسیری را ارائه دهند. خودرمزنگار متغیر، نمونه‌ای از یک مدل متغیر نهفته احتمالی است.



شکل ۳ــ۳ شمایی از یک مدل متغیر نهفته احتمالی

قضیه تغییر متغیرها

با توجه به یک متغیر مشاهده شده $x \in X$، یک توزیع احتمال پیشین ساده p_z بر روی یک متغیر نهفته $z \in Z$، و نگاشت یک‌به‌یک $f: X \to Z$ (با $g = f^{-1}$)، معادله تغییر متغیر، توزیع مدل برروی X را اینگونه تعریف می‌کند:

$$p_X(x) = p_Z(f(x)) \left| det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|$$

$$log\ (p_X(x)) = \log\left(p_Z(f(x))\right) + \log\left(\left| det\left(\frac{\partial f(x)}{\partial x^T}\right)\right|\right)$$

جایی که $\frac{\partial f(x)}{\partial x^T}$ ژاکوبین f روی x است. با استفاده از قانون نمونه برداری معکوس، می‌توان نمونه‌هایی دقیق از توزیع بدست آمده تولید کرد. یک نمونه $z \sim p_z$ در فضای نهفته ترسیم می‌شود و تصویر معکوس آن $x = f^{-1}(z) = g(z)$ نمونه‌ای را در فضای اصلی ایجاد می‌کند.

۴.۳ مدل مولد عمیق

یادگیری عمیق اکنون قادر است داده‌هایی جدیدی را پس از یادگیری از داده‌های ورودی بدون برچسب ایجاد کند. از همین‌رو، به یک "هوش خلاق" تبدیل شده است. به عنوان مثال، شبکه‌های مولد تخامصی که امروزه محبوب‌ترین مدل مولد عمیق می‌باشد، می‌تواند تصاویر با کیفیت بالا تولید کند، کیفیت عکس‌ها را بهبود بخشد، تصویر را به متن تبدیل کند، با افزایش سن ظاهر تصویر صورت را تغییر دهد، در امنیت سایبری برای شبیه‌سازی حملات استفاده شود، به پزشکی در تشخیص سرطان با ایجاد اسکن‌های واقع‌گرایانه جدید کمک کند و در طیف گسترده‌ی دیگری با امکاناتی بی‌پایان مورد استفاده قرار گیرند.

ذکر این نکته خالی از لطف نیست، از آن‌جا که شبکه‌های مولد تخاصمی توانایی تولید داده‌های جدید را دارند، این امر گاهی خطرناک نیز خواهد بود. به عنوان مثال، می‌تواند تصاویر جعلی ایجاد کند که شاید واقعی به نظر برسند و این واقعیت می‌تواند پیامدهای اخلاقی، اجتماعی و سیاسی جدی بین کشورها عمدتا با توجه به تجربه محدود سیاست‌مداران با هوش مصنوعی داشته باشد. علاوه‌براین، اگر توسط یک کلید

خصوصی رمزگذاری شده به سرورهای میزبان ارسال شود، می‌توان آن‌ها را برای کشف رمزهای عبور کاربر آموزش داد. از هش می‌توان به عنوان نویز تغذیه شده به مولد استفاده کرد و با زمان کافی، شبکه قادر به کشف رمز عبور کاربر خواهد شد.

علیرغم این خطراتی که می‌تواند ناشی از استفاده غیراخلاقی و غیرمسئولانه از شبکه‌های مولد تخاصمی باشد، آن‌ها یک فناوری جدید بسیار خلاقانه هستند که بسیاری از فرآیندهای خودکار روزانه که در جامعه ما انجام می‌شود را افزایش می‌دهد تا بتوانیم روی موضوعات دیگر تمرکز کنیم.

توانایی یادگیری بازنمودهای مفید از داده‌ها بدون راهنمایی انسان هنوز هم یک چالش اساسی برای پیشرفت‌های تحقیقاتی در هوش مصنوعی است. پیشرفت در الگوریتم‌های مولد از اهمیت زیادی برخوردار است. چرا که، انسان‌ها دقیقا همانند مدل‌های تفکیک‌پذیر عمل نمی‌کنند و توانایی‌های تولیدی با تخیل بسیاری بالا را دارا هستند. به عنوان مثال اگر ویژگی‌های خاصی از یک ماشین مثلا ماشینی با رنگ آبی را در جاده به یک انسان بدهیم، می‌تواند بلافاصله تصویری از آن را در ذهن خود ایجاد کند. هوش مصنوعی به دنبال ارائه همین نوع از هوش برای ماشین‌ها می‌باشد. استفاده از تکنیک‌های یادگیری عمیق، طی چندین سال گذشته پیشرفت‌های بزرگی را در ساخت مدل‌های مولد بوجود آورده است.

مدل‌های مولد در زیرمجموعه روش‌های بدون‌نظارت قرار می‌گیرد چرا که سعی در یادگیری توزیع داده‌های مجموعه داده آموزشی را دارد. مدل‌های مولد طی دهه گذشته در خط مقدم یادگیری بدون‌نظارت عمیق قرار داشته‌اند. دلیل این امر این است که آن‌ها روشی بسیار کارآمد را برای تجزیه و تحلیل و درک داده‌های بدون برچسب ارائه می‌دهند.

به‌طور خلاصه ایده پشت مدل‌های مولد، گرفتن توزیع احتمالاتی درونی از یک کلاس داده در جهت تولید داده‌های مشابه با آن می‌باشد. از آن‌جا که مدل‌های مولد توزیع مشترک را اغلب برروی متغیرهای مشاهده شده و نهفته نشان می‌دهند، استنباط برروی پارامترهای مدل و متغیرهای نهفته می‌تواند مشکل ساز و یا حتی غیرقابل حل باشد (به‌ویژه در فضاهای ورودی با ابعاد بالا همانند تصاویر، طراحی یک فضای

مشخصه به اندازه کافی توانا برای توضیح داده‌های موجود). در جهت غلبه براین مشکل، به‌طور موفقیت‌آمیزی از مدل‌های مولد عمیق استفاده شده است.

به طور کلی، مدل‌های مولد عمیق با کمک تکنیک‌های پس‌انتشار آموزش می‌یابند تا توزیع احتمالی را که تا حد ممکن نزدیک به توزیع تولید داده است را بیاموزند. یک رویکرد معمول این است که یک متغیر نویز را از یک توزیع ساده، همانند توزیع نرمال استاندارد نمونه‌برداری کنیم و این نمونه را با کمک ساختارهای شبکه عصبی به گونه‌ای تبدیل کنیم تا به نمونه‌ای از توزیع تولید داده شباهت داشته باشد.

مدل‌های مولد عمیق را می‌توان به سه دسته اصلی تقسیم‌بندی کرد:

۱. **مدل‌های مبتنی‌بر تابع هزینه همانند خودرمزنگار و شبکه مولد تخاصمی.**
۲. **مدل‌های مبتنی‌بر انرژی که در آن احتمال مشترک با استفاده از یک تابع انرژی تعریف می‌شود. انواع مختلف ماشین بولتزمن و شبکه‌های باور عمیق در این دسته قرار می‌گیرند.**
۳. **مدل‌های مبتنی‌بر جریان که از توسط تبدیلات وارون ساخته می‌شوند. مدل‌های مبتنی‌بر جریان یکسان‌ساز و جریان خودبرگشتی نمونه‌ای از این مدل‌های مولد هستند.**

۱.۴.۳ خودرمزنگار متغیر

خودرمزنگارها به‌طور کلی عمدتا برای کاهش ابعاد و یادگیری بازنمایی استفاده می‌شوند. با این حال، ارتباطات نظری بین مدل‌های متغیر نهفته منجر به ایجاد خودرمزنگار متغیر شده است که می‌تواند به عنوان مدل مولد استفاده شود. خودرمزنگار متغیر نمونه‌ای از یک مدل متغیر نهفته عمیق است که از شبکه‌های عصبی برای استنباط پسین تقریبی متغیرهای نهفته و تولید نمونه‌های داده استفاده می‌کند.

یک خودرمزنگار متغیر یک مدل مولد احتمالی است که در آن چگالی احتمال $p(x)$ از طریق یک متغیر نهفته z مدل می‌شود. هدف ما مدل‌سازی $p(x)$ است به گونه‌ای که با نمونه‌برداری از توزیع، منجر به تولد یک نمونه متقاعدکننده از مجموعه داده ما شود

که در این مجموعه داده وجود ندارد (قادر به تولید نمونه داده‌های جدیدی است که مشابه نمونه داده‌ای است که مدل در طول فرآیند آموزش آن را دیده است).

سناریو مساله

برای درک بهتر خودرمزنگار متغیر، سناریوی شفافی از مساله را تعریف می‌کنیم. فرض می‌کنیم مجموعه داده $X = \{x^{(1)}, x^{(2)}, \dots, x^{(n)}\}$ متشکل از n نمونه از برخی متغیرهای پیوسته یا گسسته x است و داده‌ها توسط برخی فرآیندهای تصادفی که شامل یک متغیر نهفته تصادفی z می‌باشد، تولید می شوند. در این فرآیند، یک مقدار نهفته $z^{(i)}$ از برخی توزیع‌های پیشین $p_{\theta^*}(z)$ تولید می‌شود. سپس، $x^{(i)}$ از برخی توزیع‌های شرطی $p_{\theta^*}(x|z)$ تولید می‌شود. در این سناریو، فرض می‌کنیم که $p_{\theta^*}(z)$ و همچنین درست‌نمای $p_{\theta^*}(x|z)$ به ترتیب از خانواده‌های توزیع پارامتری $p_\theta(z)$ و $p_\theta(x|z)$ می‌آیند و توابع چگالی احتمال آن‌ها با توجه به θ و z تقریبا در همه‌جا قابل تغییر است. پارامترهای واقعی (راستین) θ^* و همچنین مقادیر نهفته $z^{(i)}$ برای ما ناشناخته هستند.

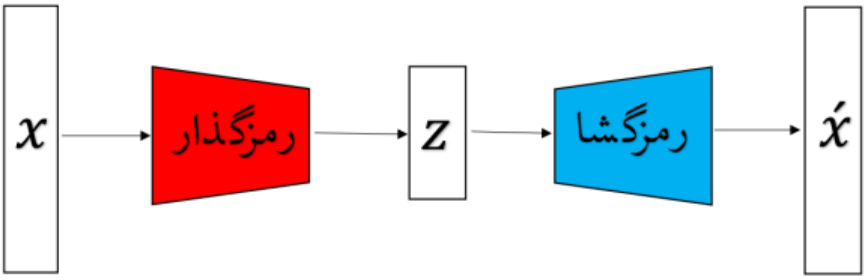
برای توصیف توزیع داده‌های اصلی و ارتباط آن‌ها با متغیرهای نهفته، علاقمند به استفاده از درست‌نمایی مرزی $p_\theta(x^{(i)}) = \int p_\theta(x^{(i)}|z)p_\theta(z)dz$ و همچنین چگالی پسین واقعی $p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)}$ هستیم. این انتگرال و چگالی پسین برای یک تابع احتمال (درست‌نمایی) $p_\theta(x|z)$ بیان شده توسط یک شبکه عصبی با یک لایه پنهان غیرخطی غیرقابل‌حل (رام‌نشدنی) است. بنابراین، علاقمند به تقریب‌های کارآمد برای این توزیع‌ها هستیم. برای این کار یک مدل $q_\varphi(z|x)$ معرفی می‌کنیم، که تقریبی از $p_\theta(z|x)$ پسین واقعی غیرقابل‌حل است.

معماری خودرمزنگار متغیر

خودرمزنگار متغیر یک رویکرد احتمالی برای توصیف نمونه‌های داده x در فضای نهفته z است. بنابراین، هر متغیر نهفته در رمز نهفته z با توزیع احتمال توصیف می‌شود. در یک خودرمزنگار متغیر می توانیم متغیرهای نهفته z را به عنوان رمز خودرمزنگار مشاهده کنیم. $q_\varphi(z|x)$ را می‌توان به عنوان یک رمزگذار احتمالی مشاهده کرد؛ با دادن یک داده x، توزیعی در مقادیر احتمالی رمز z تولید می‌شود که x می‌تواند از آن‌جا تولید شود.

به‌طور مشابه، $p_\theta(x|z)$ را می‌توان به عنوان یک رمزگشا احتمالی در نظر گرفت. با توجه به رمز z، توزیعی در مقادیر متناظر با x تولید می‌شود. باید توجه داشت که رمزگذار در این حالت، در واقع مقدار رمز z را تولید نمی‌کند. بلکه، یک توزیع برای چنین مقادیری می‌باشد. در جریان کار خودرمزنگار می‌توانیم مقدار z را از این توزیع نمونه بگیریم و وارد رمزگشا کنیم.

معماری یک مدل خودرمزنگار متغیر در شکل ۳ـ۵ قابل نمایش است. شبکه رمزگذار و رمزگشا به‌طور کلی می‌تواند هر نوع شبکه عصبی باشد. اما، یک انتخاب معمول استفاده از پرسپترون چندلایه می‌باشد.

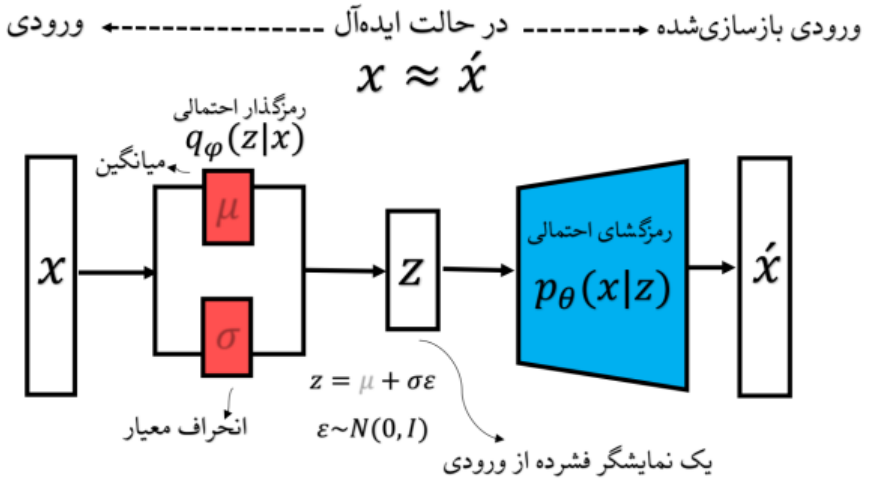


شکل ۳ـ۵ معماری خودرمزنگار متغیر

اگر توزیع یکسان‌ساز گاوسی چندمتغیره $N(0, I)$ را به عنوان $p_{\theta^*}(z)$ پیشین و $q_\varphi(z|x^{(i)})$ پسین تقریبی را با توزیع یکسان‌ساز گاوسی چندمتغیره $N(\mu, dig(\sigma))$ با پارامترهای $\mu = (\mu_1, \ldots, \mu_k)$ و $\sigma = (\sigma_1, \ldots, \sigma_k)$ را اعمال کنیم، با بازکردن رمزگشا در فرآیند نمونه‌برداری تصادفی می‌توان در شکل ۳ـ۶ جزئیات بیشتری را از معماری این خودرمزنگار مشاهده کرد.

خودرمزنگار متغیر، یک داده x را به عنوان ورودی دریافت می‌کند و آن را از طریق یک شبکه عصبی به پارامترهای μ و $log\ \sigma$ تبدیل می‌کند که تقریبی از $q_\varphi(z|x^{(i)})$ پسین می‌باشد. بنابراین، در این حالت $\varphi = (\mu, \sigma)$ است. برای سهولت و دقت عددی، به‌جای استفاده از σ مقدار $log\ \sigma$ آن را یاد می‌گیریم. سپس، یک مقدار نهفته z از توزیع

$N(\mu, dig(\sigma))$ نمونه‌برداری می‌شود. سرانجام، این z از طریق یک رمزگشا به $\acute{x}$ از x ورودی بازسازی (تبدیل) می‌شود. این بازسازی نشان‌دهنده مقدار میانگین توزیع $p_\theta(x|z)$ است که از آن برای ارزیابی خطای بازسازی استفاده می‌کنیم.



شکل ۳ـ۶ معماری خودرمزنگار متغیر با روند دقیق رمزگذار

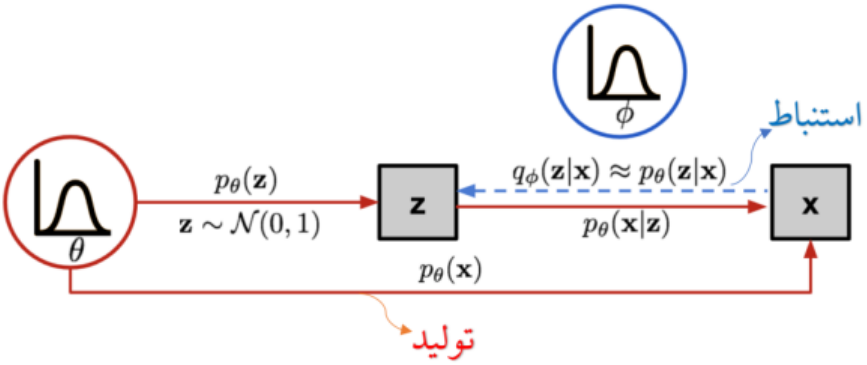
آموزش خودرمزنگار متغیر

رابطه بین داده ورودی x و بردار رمزگذاری نهفته z را می‌توان به طور کامل توسط احتمالات زیر تعریف کرد:

- $p(z)$ پیشین
- درست‌نمایی $p_\theta(x|z)$؛ که توسط رمزگشا تعریف می‌شود.
- $q_\varphi(z|x)$ پسین؛ که توسط رمزگذار تعریف می‌شود.

که در آن φ مجموعه پارامترهای متعلق به تابع رمزگذار و θ پارامترهای تابع رمزگشا است.

با دانستن تعاریف بالا، می‌توانیم مدل گرافیکی احتمالی را همانند شکل ۳ـ۷ ترسیم کنیم.



شکل ۳ـ۷ مدل احتمالی گرافیکی خودرمزنگار متغیر (استنباط + تولید)

با فرض این‌که θ^* را پارامترهای راستین این توزیع بدانیم، با انجام دو مرحله زیر می‌توانیم نمونه داده‌های جدیدی را که به‌نظر می‌رسد مشابه یک نقطه داده از $x^{(i)}$ است را تولید کنیم:

۱. نمونه‌برداری از بردار متغیرهای نهفته $z^{(i)}$ از طریق توزیع پسین $p_{\theta^*}(z)$
۲. استفاده از رمزگشا به عنوان شبکه مولد و بازسازی بردار نهفته نمونه با تابع احتمال (درست‌نمایی) شرطی $p_{\theta^*}(x|z = z^{(i)})$ و تولید یک مقدار $x^{(i)}$

پارامترهای بهینه θ^* پارامتری است که احتمال بازسازی را برای هر نقطه $x^{(i)}$ بیشینه می‌کند. از همین رو هدف ما این است که:

$$\theta^* = arg_\theta max \prod_{i=1}^{n} p_\theta(x^{(i)})$$

همان‌طور که قبلا نیز بیان شد، برای سادگی محاسبات و پایداری عددی می‌توان آن را به شکل زیر بازنویسی کنیم:

$$\theta^* = arg_\theta max \sum_{i=1}^{n} log\, p_\theta(x^{(i)})$$

اگر بخواهیم توزیع پسین واقعی فضای نهفته $p_\theta(z|x)$ را محاسبه کنیم، باید $p_\theta(x)$ را با توجه به قضیه بیز تعیین کنیم:

$$p_\theta(z|x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)}$$

حالا بیایید معادله را بروز کنیم تا روند تولید داده بهتر نشان داده شود و بردار رمزگذار را درگیر کنیم:

$$p_\theta\left(x^{(i)}\right) = \int p_\theta\left(x^{(i)}|z\right)p_\theta(z)dz$$

بدیهی است که این روش خوبی نیست. چرا که محاسبه $p_\theta\left(x^{(i)}\right)$ پسین برای هر نمونه داده $x^{(i)}$ هزینه زیادی را در پی دارد. بنابراین، برای محدود کردن فضا در جهت جستجوی سریع‌تر، خودرمزنگار متغیر از استنباط تقریبی توزیع رام‌نشدنی[۱] پسین استفاده می‌کند، که توسط تابع رمزگذار از طریق $q_\varphi(z|x)$ نمایش داده می شود.

از آنجا که $q_\varphi(z|x)$ تنها برآورد پسین واقعی رام‌نشدنی است، ما باید تفاوت این دو احتمال را بدست آوریم. یا به عبارت دیگر، $q_\varphi(z|x)$ باید بسیار نزدیک به $p_\theta(z|x)$ واقعی باشد. برای تعیین کمیت فاصله بین این دو توزیع، می‌توان اندازه‌گیری اختلاف را با کمک واگرایی کولبک ـ لیبلر بدست آورد.

برای دو توزیع احتمال واگرایی کولبک ـ لیبلر به‌صورت زیر محاسبه می‌شود:

$$D_{KL} = \left(q_\varphi(z|x) \parallel p_\theta(z|x)\right) = \int q_\varphi(z|x) \log\left(\frac{q_\varphi(z|x)}{p_\theta(z|x)}\right) dz$$

و با نابرابری ینسن، واگرایی کولبک ـ لیبلر همیشه یک غیرمنفی $D_{KL}(p\|q) \geq 0$ است. از همین‌رو ما می‌توانیم تعیین کنیم که $q_\varphi(z|x)$ پسین تقریبی با $p_\theta(z|x)$ واقعی چقدر تفاوت دارد.

---

[۱] intractable

حال با تجزیه و تحلیل معادله واگرایی کولبک ـ لیبلری که در بالا بدست آوردیم، تابع هدف یا همان تابع هزینه را برای خودرمزنگار متغیر بدست می‌آوریم.

$$D_{KL} = \left(q_\varphi(z|x) \parallel p_\theta(z|x)\right) = \int q_\varphi(z|x)\, log\left(\frac{q_\varphi(z|x)}{p_\theta(z|x)}\right) dz$$

$$= \int q_\varphi(z|x)\, log\left(\frac{q_\varphi(z|x)p_\theta(x)}{p_\theta(z,x)}\right) dz \quad \text{"زیرا: } p(z|x) = \frac{p(z,x)}{p(x)} \text{"}$$

$$= \int q_\varphi(z|x)(log\, p_\theta(x) + log\left(\frac{q_\varphi(z|x)}{p_\theta(z,x)}\right))dz$$

$$= log\, p_\theta(x) + \int q_\varphi(z|x) log\left(\frac{q_\varphi(z|x)}{p_\theta(z,x)}\right) dz \text{ "زیرا: } \int q(z|x)dz = ۱ \text{"}$$

$$= log\, p_\theta(x) + \int q_\varphi(z|x) log\left(\frac{q_\varphi(z|x)}{p_\theta(x|z)p_\varphi(z)}\right) dz \text{ "زیرا: } p(z,x) = p(z|x)p(z) \text{"}$$

$$= log\, p_\theta(x) + \mathbb{E}_{z\sim q_\varphi(z|x)}[log\left(\frac{q_\varphi(z|x)}{p_\varphi(z)}\right) - log p_\theta(x|z)]$$

$$= log\, p_\theta(x) + D_{KL}\left(q_\varphi(z|x) \parallel p_\theta(z)\right) - \mathbb{E}_{z\sim q_\varphi(z|x)}[log p_\theta(x|z)]$$

حال با مرتب کردن معادله فوق داریم:

$$log\, p_\theta(x) - D_{KL}\left(q_\varphi(z|x) \parallel p_\theta(z)\right) =$$

$$\mathbb{E}_{z\sim q_\varphi(z|x)}[log p_\theta(x|z)] - D_{KL}\left(q_\varphi(z|x) \parallel p_\theta(z)\right) = -ELBO$$

سمت چپ معادله فوق دقیقا همان چیزی است که می‌خواهیم آن را هنگام یادگیری توزیع‌های واقعی بیشینه کنیم. در واقع ما قصد داریم احتمال تولید داده‌ها واقعی را بیشینه، چیزی که در قسمت اول معادله است $log\, p_\theta(x)$؛ و همچنین تفاوت بین توزیع تقریبی را با توزیع برآورد شده کمینه کنیم (قسمت دوم معادله).

با توجه به این‌که بسیاری از الگوریتم‌های بهینه‌سازی همانند، گرادیان کاهشی با به حداقل‌رساندن تابع هدف کار می‌کنند، از همین‌رو تابع هدف نهایی خودرمزنگار متغیر به‌صورت زیر خواهد بود:

$$L_{\text{خودرمزنگار متغیر}}(\varphi, \theta, x, z) = -ELBO$$

$$= \mathbb{E}_{z\sim q_\varphi(Z|X)}[logp_\theta(x|z)] - D_{KL}\big(q_\varphi(z|x)\|\, p_\theta(z)\big)$$

$ELBO$ عبارتی است که برای روش‌های متغیر بیزی تعریف شده است. این تابع زیان با عنوان حد پایین متغیر[۱] یا حد پایین مشاهدات[۲] شناخته می‌شود. حد پایین در این نام‌گذاری در واقع از آن‌جا ناشی می‌شود که واگرایی کولبک ـ لیبلر همیشه غیرمنفی است. بنابراین، تابع هدف خودرمزنگار متغیر همیشه کوچک‌تر از $logp_\theta(x|z)$ می‌باشد:

$$log\ p_\theta(x) - D_{KL}\big(q_\varphi(z|x)\|\, p_\theta(z)\big) = L_{\text{خودرمزنگار متغیر}} = -ELBO \le logp_\theta(x)$$

اکنون مشاهده می‌شود که با کمینه‌کردن لگاریتم درست‌نمایی منفی، که اولین قسمت در معادله قبل می باشد، درحال بهینه‌سازی هستیم. از همین‌رو، با کمینه‌کردن تابع زیان $L_{\text{خودرمزنگار متغیر}}$ حد پایین احتمال تولید نمونه داده‌های واقعی را با استفاده از گرادیان نزولی حداکثر می‌کنیم و بهترین پارامترها را بازیابی می‌کنیم:

$$\theta^*, \varphi^* = arg_{\theta,\varphi} \min L_{\text{خودرمزنگار متغیر}}$$

همان‌طور که مشاهده می شود، با کمینه کردن $L_{\text{خودرمزنگار متغیر}}$، به‌طور هم‌زمان $ELBO$ را که متناسب با درست‌نمایی آن رفتار می‌کنیم، بیشینه کرده و بنابراین احتمال بازسازی را افزایش می‌دهیم.

---

[۱] variational lower bound
[۲] evidence lower bound

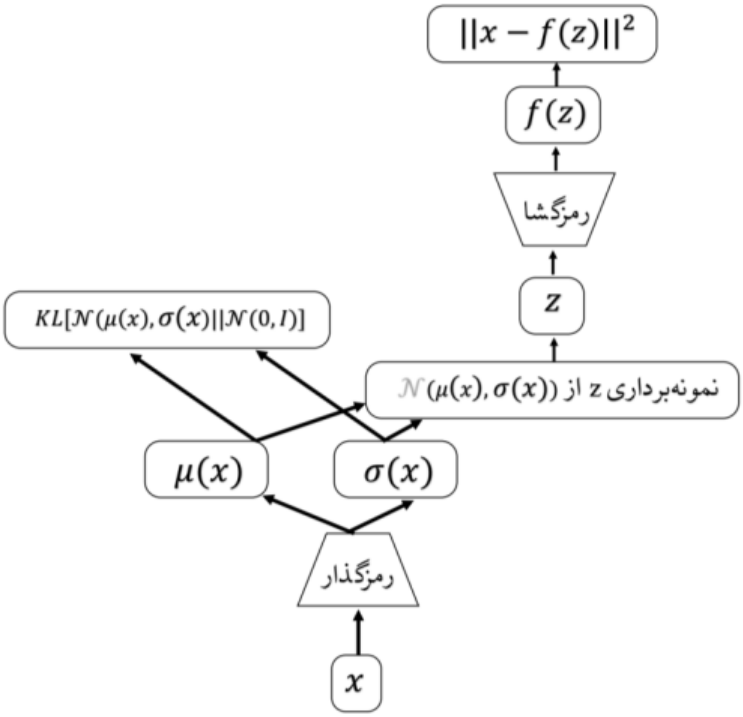
ترفند پارامتری‌سازی مجدد[۱]

در خودرمزنگار متغیر، رمزگشا به‌طور تصادفی از پسین واقعی $z \sim q_\varphi(z|x)$ نمونه‌برداری می‌کند. هنگام آموزش مدل با استفاده از گرادیان کاهشی تصادفی، این مساله منجر به بروز مشکلی می گردد. چرا که مشتق‌گرفتن با توجه به پارامترهای متغیر φ امکان‌پذیر نیست. به عبارت دیگر، گرادیان‌ها را نمی‌توان از طریق متغیر پنهان z به عقب انتشار داد. این مشکل از آنجا ناشی می‌شود که پس‌انتشار از طریق گره‌های تصادفی قابل جریان نیست و پس‌انتشار برای تعیین این پارامترها، انتظار مقادیر قطعی را دارد. این مشکل در شکل ۳ـ۸ قابل نمایش است. در جهت غلبه با این مشکل از ترفند پارامترسازی مجدد استفاده می‌شود.

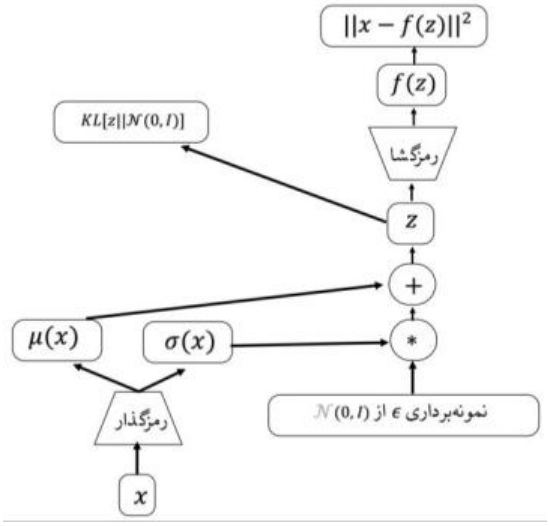
این ترفند شامل نمونه‌برداری از یک متغیر تصادفی کمکی از یک توزیع ثابت، یکسان‌ساز $\mathcal{N}(۰, I)$ است. به عبارت دیگر، می‌توان با تعریف متغیرهای تصادفی کمکی $\mathcal{N}(۰, I)$ با آن‌ها به عنوان ورودی رفتار کنیم. سپس، با محاسبه $z = \mu + \sigma \odot \epsilon$ نمونه‌برداری $z \sim q_\varphi(z|x) = \mathcal{N}(z, \mu, \sigma^۲ * I)$ را شبیه‌سازی می‌کنیم. که در آن μ و σ خروجی قطعی تولید شده توسط لایه رمزگذار و $\epsilon \sim \mathcal{N}(۰, I)$ است.

پارامترسازی‌مجدد تنها به پارامترهای تعیین کننده از شبکه استنباط بستگی دارد. بنابراین، می‌توانیم گرادیان‌های خروجی رمزگشا $f(z)$ را با توجه به پارامترهای توزیع متغیر پنهان μ و σ محاسبه و این اطلاعات را به رمزگذار پس‌انتشار دهیم. از همین‌رو، روند نمونه‌گیری تصادفی با استفاده از ترفند پارامترسازی‌مجدد همان‌طور که در شکل ۹ـ۳ نشان داده شده، می باشد.

[۱] Parameterization Trick

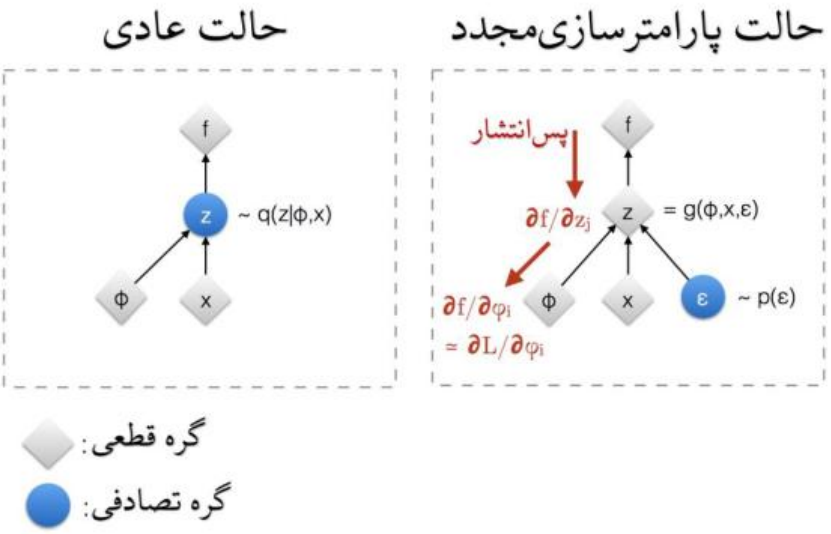


شکل ۳ـ۸ پس‌انتشار و نیاز به ترفند پارامترسازی‌مجدد



شکل ۳ـ۹ پارامترسازی‌مجدد بدون هیچ گره تصادفی در مسیر خروجی به ورودی و امکان‌پذیری پس‌انتشار

شکل ۳ــ۱۰ مقایسه گراف محاسباتی در خودرمزنگار متغیر را با دیدی کلی‌تر در حالت عادی و حالت پارامترسازی‌مجدد نشان می‌دهد.



شکل ۳ــ۱۰ مقایسه حالت عادی و حالت پارامترسازی مجدد. پارامترهای پسین تقریبی φ با استفاده از متغیر نهفته $z{\sim}q_{\varphi}(z|x)$ بر تابع هدف f تاثیر می‌گذارند. اما از آن‌جایی که نمی‌توان مشتق f را نسبت به φ (زیرا گرادیان‌ها از طریق متغیرهای تصادفی z قابلیت پس‌انتشار را ندارند) بدست آورد، یک متغیر تصادفی خارجی از توزیع ثابت $p(\epsilon) = \mathcal{N}(۰, I)$ نمونه‌برداری و سپس یک تبدیل متغیر ساده اجرا می‌شود.

۲.۴.۳ شبکه‌های مولد تخاصمی

همان‌طور که در بخش قبلی دیدیم، آموزش مدل‌های مولد، استنباط از فضای نهفته را امکان‌پذیر می‌ساخت. یک خودرمزنگار متغیر، شبکه را مجبور می‌کرد تا نگاشتی از یک فضای مستقل گوسی به توزیع هدف را بیآموزد. این نوع مدل مولد با کمینه‌کردن تابع زیان بین داده‌های اصلی و داده‌های فشرده‌شده آموزش پیدا می‌کرد. اما، در این بخش قصد داریم با نمونه‌ای دیگر از شبکه‌های مولد عمیق به نام شبکه‌های مولد تخاصمی آشنا شویم که با روشی خصمانه آموزش می‌بینند.

شبکه‌های مولد تخاصمی روشی محاسباتی براساس تئوری بازی‌ها هستند و در هسته آن ترکیبی از دو شبکه عصبی وجود دارد. یکی از این شبکه‌ها مولد و دیگری تمایزگر نامیده می‌شود و یک رابطه‌ی خصمانه‌ای بین این دو شبکه وجود دارد. به زبان ساده، یکی از آن‌ها برای ایجاد داده آموزش می‌بیند و دیگری برای تشخیص این‌که چه داده‌ای واقعی و چه داده‌ای جعلی می‌باشد. مولد که یک جعل‌کننده است سعی می‌کند، شبکه دوم را (یک کارآگاه است که سعی می‌کند جعلی را از واقعی تشخیص دهد) فریب دهد و با هر تلاش ناموفق، با دریافت بازخورد از آن، می‌تواند بهتر از پس این کار برآید. به صورت خلاصه می‌توان گفت، یکی تولید می‌کند و دیگری عیب‌جویی می‌کند و در کنار هم و در یک همکاری کامل، نتایج بسیار خوبی به‌دست می‌آورند.

برای تجسم بهتر روند تخاصم، فرض کنید فردی (مولد) می‌خواهد اشعاری بنویسد که مشابه اشعار حافظ است؛ بدون این‌که هرگز یکی از اشعار حافظ به او نشان یا حتی کوچک‌ترین اطلاعاتی در مورد آن به او اطلاع داده شود. به فرد دیگر (متمایزگر) که به‌همان اندازه نسبت به حافظ ناآگاه است، مواردی که توسط مولد نوشته شده و بخش‌هایی از اشعار حافظ به‌صورت تصادفی نشان داده می‌شود. سپس، متمایزگر با احتمالی در مقیاس ۰ تا ۱ باید بگوید که فکر می‌کند شعر نوشته شده واقعی است. اگر طبقه‌بندی غلطی انجام دهد تنبیه می‌شود.

متمایزگر به‌زودی شروع به یادگیری به نحوی می‌کند تا از تنبیه اجتناب و در تشخیص بهتر اشعار نوشته شده توسط مولد از اشعار حافظ بهتر عمل کند. همچنین، مولد هر زمان که متمایزگر به درستی خط‌خطی‌ها آن را به عنوان جعلی نشان دهد، تنبیه می‌شود. با انگیزه بیذاری از تنبیه، مولد شروع به یادگیری نحوه تولید اشعار متقاعد کننده همانند حافظ می‌کند که متمایزگر را فریب می‌دهد. این بازی تا جایی ادامه می‌یابد که امیدواریم تعادلی حاصل شود، که نه مولد و نه متمایزگر عملکرد خود را بهبود نمی‌بخشند. وجود همین دو حلقه تنبیه منجر به یادگیری می‌شود. پس از این مرحله، متمایزگر به خانه فرستاده می‌شود، در حالی‌که مولد فرمول پیچیده‌ای را برای تولید اشعاری به سبک حافظ را بدست می‌آورد. فرمول تولیدی که توسط مولد ساخته می‌شود از تغییرات و فشار انتخاب (متمایزگر) حاصل می‌شود. معیار موفقیت در این روش هم این است که مولد چقدر در فریب دادن متمایزگر، برای باور واقعی بودن یک نمونه خوب عمل کرده است.

قابلیت‌های شبکه‌های مولد تخاصمی، برای تولید داده‌های جدیدی که از داده‌های یادگیری تقلید می‌کند، چشمگیر بوده و قبلاً هرگز دیده نشده است. شایان ذکر است که، یان لکان، درباره‌ی شبکه‌های مولد تخاصمی اینگونه بیان می‌کند: "**شبکه‌های مولد تخاصمی یکی از جالب‌ترین ایده‌های ۱۰ سال گذشته در یادگیری ماشین است**".

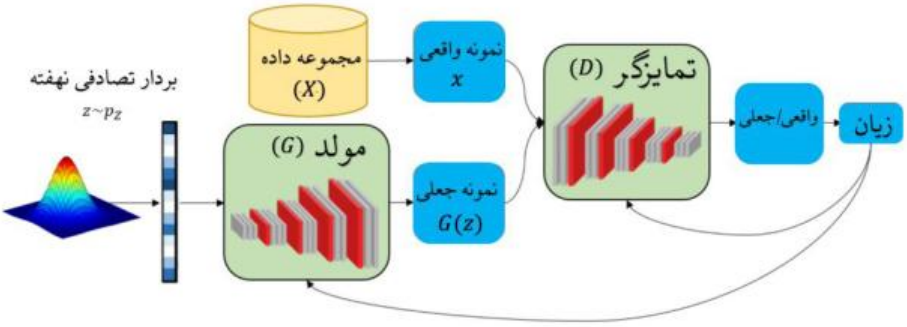
حال که با مفهوم کلی شبکه‌های مولد تخاصمی آشنا شدیم، قصد داریم به‌صورت ریاضی آن را تشریح کنیم. هدف شبکه مولد تخاصمی، یادگیری یک مولد (G) است که می‌تواند از توزیع داده (p_x)، با تبدیل بردارهای نهفته از یک فضای نهفته با ابعاد پایین (Z)، به نمونه‌ها در فضای داده با ابعاد بالاتر (x)، نمونه تولید کند. معمولاً از بردارهای نهفته با استفاده از توزیع یکنواخت یا عادی از Z نمونه‌برداری می‌شود. به‌منظور آموزش G، یک تمایزگر (D)، آموزش داده می‌شود تا نمونه‌های آموزشی واقعی را از نمونه‌های جعلی تولید شده توسط G، متمایز کند. بنابراین، تمایزگر مقدار $D_x \epsilon [۰,۱]$ را برمی‌گرداند که می‌تواند به عنوان احتمال اینکه نمونه ورودی x، یک نمونه واقعی از توزیع داده باشد، تفسیر شود. در این پیکربندی، مولد برای ایجاد انسداد در متمایزگر با تولید نمونه‌هایی که بیشتر به نمونه های واقعی آموزش شباهت دارند، آموزش داده می‌شود. در حالی که، متمایزگر به طور مداوم آموزش می‌یابد تا نمونه‌های واقعی را از جعلی تشخیص دهد.

نکته مهم در این پیکربندی این است که، مولد دسترسی مستقیمی به نمونه‌های واقعی مجموعه آموزشی ندارد. چرا که، تنها از طریق تعامل با متمایزگر آموزش می‌بینند. شکل ۳ـ۱۱ ساختار شبکه مولد تخاصمی را نشان می‌دهد. از منظر ریاضی، هدف شبکه مولد تخاصمی به‌صورت زیر بیان می‌شود:

$$\min_G max_D L(D,G) = \mathbb{E}_{x\sim p_r(x)} logD(x) + \mathbb{E}_{z\sim p(z)} \log(۱ - D(G(z)))$$

$$= \mathbb{E}_{x\sim p_r(x)} logD(x) + \mathbb{E}_{z\sim p_g(x)} \log(۱ - D(x))$$

که در معادله فوق $p_r(x)$ توزیع داده‌های واقعی و $p_g(x)$ توزیع داده‌های تولید شده توسط مولد است.



شکل ۳ـ۱۱ ساختار شبکه مولد تخاصمی

علاوه براین، به منظور آموزش مولد و تمایزگر، خطاهای موجود در خروجی آن‌ها دوباره در مدل منتشر می‌شود.

معادله بروزرسانی برای تمایزگر به‌صورت زیر:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} [logD\left(x^{(i)}\right) + \log\left(1 - D(G(z^{(i)}))\right)]$$

همچنین، معادله بروزرسانی برای مولد به‌صورت زیر می‌باشد:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} log\left(1 - D(G(z^{(i)}))\right)$$

که در آن m نمایانگر تعداد کل نمونه‌های آزمایش‌شده به‌صورت دسته‌ای قبل از بروزرسانی هر دو مدل، و θ_d و θ_g نمایانگر وزن هر مدل است.

شایان ذکر است، از آنجا که شبکه مولد تخاصمی تنها یک ساختار سیستم است، انتخاب عناصر برای ساخت این سیستم (مولد و تمایزگر) به عهده کاربر است.

۳.۴.۳ ماشین بولتزمن

در این بخش قصد داریم نوعی خاصی از مدل‌های یادگیری ماشین را به نام ماشین بولتزمن را که یک مدل مبتنی‌بر انرژی است، شرح دهیم. اما، قبل از اینکه به تشریح این مدل بپردازیم، برای درک بیشتر این مدل‌ها، بیاید کمی به گذشته در دوران مدرسه برگردیم. شاید به یاد داشته باشید که معلم شما در دوران مدرسه، علت این‌که چرا گازها به‌طور مساوی در اتاق پخش و در یک گوشه جمع نمی‌شوند را بیان کرده باشد. آیا اصطلاح توزیع ماکسول‌ـ بولتزمن را شنیده‌اید؟ معادله ماکسول‌ـ بولتزمن، اساس تئوری جنبش گازها را تشکیل می‌دهد؛ توزیع سرعت یک گاز را در یک درجه حرارت تعریف می‌کند. به عبارت دیگر:

"**در دمای اتاق، گازها به احتمال زیاد به طور مساوی در فضای اتاق پخش می شوند، چرا که این پیکربندی انرژی کل سیستم را به حداقل می رساند.**"

سرمنشا اصول اساسی ماشین بولتزمن همین جاست. انرژی اصطلاحی است که ممکن است در وهله اول با یادگیری عمیق همراه نباشد. بلکه، انرژی از ویژگی‌های کمی فیزیک است. با این وجود، برخی از معماری‌های یادگیری عمیق از ایده انرژی به عنوان معیاری برای سنجش کیفیت مدل استفاده می‌کنند. هرچند، این ارتباط ممکن است در حال حاضر مبهم به نظر برسد، اما در ادامه، منطقی به‌نظر خواهد آمد.

یکی از اهداف مدل‌های یادگیری عمیق، رمزگذاری وابستگی‌ها بین متغیرها است. مدل‌های مبتنی‌بر انرژی، وابستگی‌های بین متغیرها را با پیوند دادن یک انرژی مقیاس‌دار به هر پیکربندی متغیرها بدست می‌آورند که به عنوان معیار سازگاری عمل می‌کند. انرژی زیاد به معنی سازگاری بد است. یک مدل مبتنی بر انرژی سعی می‌کند همیشه یک تابع انرژی از پیش تعریف شده را به حداقل برساند.

یادگیری در مدل مبتنی‌بر انرژی، شامل یافتن یک تابع انرژی است که در آن پیکربندی مشاهده شده متغیرها، انرژی کم‌تری نسبت به موارد مشاهده نشده داشته باشند. هسته اصلی مدل‌های داده مبتنی‌بر انرژی بر این اصل استوار است: **یافتن پیکربندی از مدل تا انرژی سیستم کاهش یابد.** به عبارت دیگر، هدف از یادگیری یافتن یک تابع انرژی (در

یک فضای عملکردی از پیش تعیین شده) است که مقادیر کوچک‌تر را به پیکربندی‌های درست و مقادیر بالاتر را برای موارد نادرست، چه در داخل و چه در خارج از نمونه‌های آموزشی، مرتبط کند. سپس، با انتخاب پیکربندی که انرژی را کمینه می‌کند، پیش‌بینی‌ها انجام می‌شود. برای سنجش کیفیت توابع انرژی موجود، از تابع زیان استفاده می‌شود که در حین یادگیری به حداقل می رسد. با در نظر گفتن این تفکر،حال بیاید نگاهی به ماشین بولتزمن بیندازیم:

ماشین بولتزمن، یک معماری مبتنی‌بر انرژی است که توزیع احتمال را برروی متغیرهای ورودی خود می‌آموزد. ایده پشت ماشین بولتزمن، این است که یک مدل باید بتواند رفتار آماری پارامترهای ورودی را به تنهایی و بدون هیچ نظارتی از نمونه‌های ورودی بیاموزد. هنگامی که مجموعه‌ای از نمونه‌ها ارائه می‌شود، باید بتواند توزیع آماری را از طریق ویژگی‌های ورودی بیاموزد. هنگامی که مدل تمام این نمونه‌های ورودی را مشاهده کرد، باید بتواند به تنهایی نمونه‌هایی تولید کند که از توزیع احتمال مشترک با نمونه‌های ورودی اولیه بدست آید و از این رو رفتار آماری یکسانی داشته باشند.

انرژی در ماشین بولتزمن

ماشین‌های بولتزمن، از لحاظ نظری این توانایی را دارند که هر توزیع داده شده را تنها با نشان دادن نمونه‌های، نمونه‌برداری شده از آن یاد بگیرند. اساسا، شبکه ارتباط‌های قوی بین گره‌ها را برای گرفتن همبستگی‌هایی که آن‌ها را بهم متصل می‌کند، تنظیم تا یک شبکه مولدی را ایجاد که قادر است نمونه‌های جدیدی از همان توزیع تولید کند. یادگیری در این مدل‌ها ماهیت هبین دارد. به عبارت دیگر، برای بروز کردن وزن‌ها، تنها نیاز به اطلاعات نرون‌های همسایه داریم. در عمل، نرون‌ها براساس این‌که، چندبار در خروجی با یک‌دیگر توافق دارند، تصمیم می‌گیرند که پیوند خود را تقویت یا تضعیف کنند.

در ماشین بولتزمن، نورون‌ها نه تنها به سلول‌های عصبی در لایه‌های دیگر، بلکه به سلول‌های عصبی درون همان لایه نیز متصل هستند.

از منظر معماری، ماشین بولتزمن شبکه‌ای از گره‌هایی دودویی است. همه گره‌ها با یک یال بدون جهت که دارای وزنی می‌باشد، به یک دیگر ارتباط دارند. اگر وزن ۰ باشد،

می‌توان گره‌ها را بدون اتصال تصور کرد و اگر ۱ باشد عکس آن اتفاق می‌افتد. گره‌ها در ساده‌ترین حالت ماشین بولتزمن، تصادفی و دودویی هستند. یعنی، یک گره بسته به پیکربندی بقیه گره‌ها و وزن مربوطی که آنها را بهم متصل می‌کند، می‌تواند تصمیمی به‌صورت تصادفی بگیرد که روشن یا خاموش باشد. این متفاوت از شبکه‌های عصبی پیش‌خور است که در آن تابع فعال‌سازی یک گره تعیین‌کننده برای فعال‌سازی گره‌ها و وزن‌های مدل است.

هر گره در ماشین بولتزمن، دارای بایاس و با وزنی متقارن به گره‌های همسایه متصل می‌شود. گره i با گره j همسایه است، اگر و تنها اگر گره i و گره j دارای وزن غیرصفر باشند. حالت (یا فعال‌سازی) گره فقط به فعال شدن گره‌های همسایه آن و وزن متصل به آن‌ها بستگی دارد و به طور مشروط از سایر گره‌ها و وزن شبکه مستقل است.

برای تعیین حالت گره، یک محاسبه آماری انجام می‌دهیم:

$$Z_i = b_i + \sum_j s_j W_{ij}$$

که در این معادله b_i بایاس، $s_{j=1}$ اگر گره j روشن و برعکس و W_{ij} وزن یال بین گره i با گره j است.

سپس، بسته به مقدار Z_i، گره i با احتمال زیر روشن می‌شود:

$$P(s_i = 1) = \frac{1}{1 + e^{-z_i}}$$

اگر گره‌های ماشین بولتزمن، به‌طور متوالی انتخاب و به‌طور تصادفی بروز شوند، ماشین بولتزمن در نهایت به یک تعادل می‌رسد که در آن توزیع احتمال روی گره‌های x تثبیت می‌شود. از آن‌جا که بروزرسانی‌ها تصادفی است، نمی‌توان انتظار داشت شبکه به یک پیکربندی واحد از گره‌ها تثبیت شود. اما می‌توان انتظار داشت که در یک توزیع احتمال واحد روی آن گره‌ها تثبیت شود.

انرژی ماشین بولتزمن توسط معادله زیر تعریف می‌شود:

$$E(x) = -\sum_{i=1}^{N} b_i x_i - \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} w_{i,j} x_i x_j$$

$$E(x) = -x^T W_x - b^T x$$

جایی که W ماتریس وزن $d \times d$ بین گره i با گره j و b بردار بایاس برای گره i است.

براساس انرژی، برای گره‌های x ماشین بولتزمن در تعادل، توزیع احتمال بر روی گره‌ها به این صورت تعریف می‌شود:

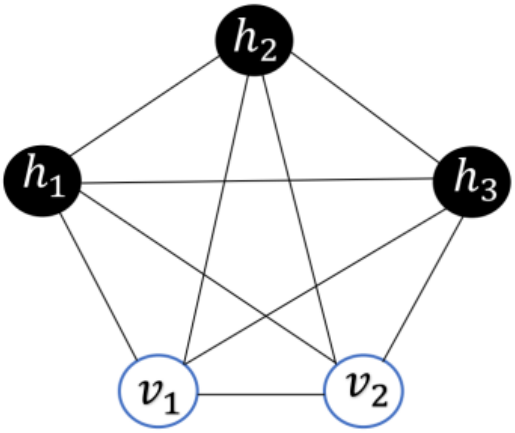
$$P(x) = \frac{1}{Z} e^{(-E(x))}$$

که در معادله فوق $z = \sum_x exp\,(-E(x))$ یک ثابت یکسان‌ساز برای اطمینان از اینکه $\sum_x P(x) = 1$ است. Z را تابع تقسیم[1] نیز می‌نامند. از معادله فوق می‌توان دریافت که احتمال x با افزایش انرژی به‌صورت یکنواخت کاهش پیدا می‌کند.

در ماشین بولتزمن، یک گره می‌تواند پنهان یا قابل رویت باشد. هر گره قابل رویت مربوط به یک ویژگی منحصربه‌فرد از ورودی و خروجی مشاهده شده است. مقادیر گره‌های قابل رویت از ویژگی‌های نمونه ورودی حاصل می‌شود و می‌توان مقادیر گره‌های نهفته را استنباط کرد. شکل ۳ـ۱۲ معماری یک ماشین بولتزمن را نشان می‌دهد.

در ادامه، مجموعه گره‌های قابل رویت را با عنوان v و مجموعه گره‌های پنهان را با h نمایش می دهیم. همچنین مجموعه همه گره‌ها را به صورت $x = v \cup h$ تعریف می‌کنیم.

[1] partition function



شکل ۳ـ۱۲ معماری یک ماشین بولتزمن با دو گره قابل رویت و ۳ گره پنهان

معادله انرژی در ماشین بولتزمن را می‌توان براساس پیکربندی مشترک گره‌های پنهان و قابل رویت، جایی که $v \in \{۰,۱\}^a$ و $h \in \{۰,۱\}^b$ و $a + b = d$ است، به‌صورت زیر بازنویسی کرد:

$$E(v,h) = -v^T w_{۱v} - v^T w_{۲h} - h^T w_{۳h} - b_۱^T v - b_۲^T h$$

که در معادله فوق، $w_۱$ بردار وزن برای اتصالات بین جفت گره‌های قابل رویت، $w_۲$ بردار وزن برای اتصالات بین جفت گره‌های پنهان و قابل رویت و $w_۳$ بردار وزن برای اتصالات بین جفت گره‌های پنهان است.

- **ماشین‌های بولتزمن مدل‌های مولد عمیق غیرقطعی (یا تصادفی) هستند، که فقط دو نوع گره دارند: گره‌های پنهان و قابل رویت و در آن‌ها هیچ گره خروجی وجود ندارد!**
- **برخلاف سایر شبکه‌های عصبی که هیچ ارتباطی بین گره‌های ورودی ندارند، یک ماشین بولتزمن دارای اتصالات بین گره‌های ورودی است. این به آنها امکان می‌دهد تا اطلاعات را بین خود به اشتراک بگذارند و داده‌های بعدی را خود تولید کنند.**

۱.۳.۴.۳ ماشین بولتزمن محدود

ماشین بولتزمن محدود، نوعی خاصی از ماشین بولتزمن با دولایه شامل: یک لایه پنهان و یک لایه قابل رویت، در جهت حل مشکل توزیع مشترک ماشین بولتزمن طراحی شده است. آنچه ماشین بولتزمن محدود را از ماشین بولتزمن متفاوت می‌کند، این است که اتصال قابل‌رویت ـ قابل‌رویت و اتصال پنهان ـ پنهان در ساختار آن‌ها وجود ندارد. غیر از این، ماشین بولتزمن محدود دقیقا همان ماشین بولتزمن است. ماشین بولتزمن محدود، به‌دلیل استقلال گره‌های درون لایه‌ای، جایگزینی قدرتمند برای ماشین‌های کاملاً متصل بولتزمن هنگام ساخت یک معماری عمیق است، چرا که امکان آزادی و انعطاف پذیری بیشتر را فراهم می کند.

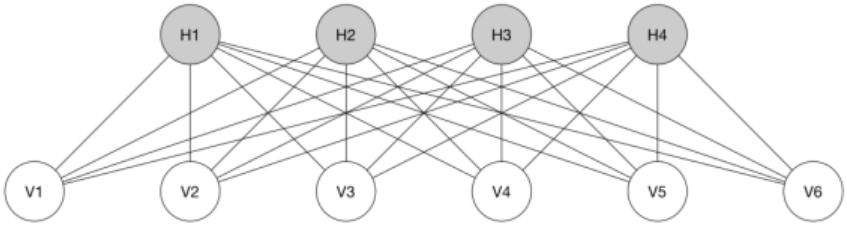
ماشین بولتزمن محدود را می‌توان با استفاده از مفهوم مدل گرافیکی احتمالی توصیف کرد، که با استفاده از یک نمایش مبتنی بر گراف، وابستگی بین متغیرهای تصادفی را بیان می‌کند. در یک ماشین بولتزمن محدود، واحدهای پنهان و قابل‌رویت یک گراف دوبخشی را تشکیل می‌دهد (همانند شکل ۱۱). از منظر احتمال، این بدان معنی است که متغیرهای پنهان با توجه به وضعیت متغیرهای قابل‌رویت مستقل هستند و بالعکس. این ویژگی استقلال شرطی، عبارات شرطی زیر را ارائه می دهد:

$$p(h|v) = \prod_{i=1}^{n} p(h_i|v)$$

و

$$p(v|h) = \prod_{j=1}^{m} p(v_i|h)$$

که در این معادلات تعداد n گره‌های پنهان و m تعداد گره‌های قابل‌رویت است.



شکل ۳ـ۱۳ مثالی از یک ماشین بولتزمن محدود با ٦ گره قابل‌رویت و ٤ گره پنهان

در ماشین بولتزمن محدود، هیچ ارتباط درون لایه‌ای بین گره‌های قابل رویت وجود ندارد. همچنین هیچ ارتباط درون لایه‌ای بین گره‌های پنهان وجود ندارد. تنها ارتباطات بین گره‌های ورودی و پنهان وجود دارد. دلیل نام‌گذاری آن‌ها، به ماشین بولتزمن محدود، وجود همین محدودیت ارتباط درون لایه‌ای بین گره‌ها است. این محدودیت امکان استفاده از الگوریتم های آموزشی کارآمدتر را فراهم می کند.

توزیع احتمال و تابع انرژی در ماشین بولتزمن محدود

تابع انرژی در ماشین بولتزمن محدود به‌صورت زیر تعریف می‌شود:

$$E(v,h) = -v^T W h - b^T v - c^T h$$

که در این معادله، W ماتریس وزنی است که w_{ij} وزن یال‌های بین گره قابل‌رویت v_i با گره پنهان h_i است. b بردار بایاس گره‌های قابل‌رویت جایی که b_i در آن بایاس گره v_i است. به‌طور مشابه c بردار بایاس برای گره‌های پنهان است.

همانند ماشین بولتزمن، توزیع احتمال مشترک ماشین بولتزمن محدود برروی h و v به‌صورت زیر است:

$$P(v,h) = \frac{1}{Z} e^{(-E(v,h))}$$

که در آن Z تابع تقسیم به‌صورت $Z = \sum_{v,h} e^{(-E(v,h))}$ تعریف شده است.

۲.۳.۴.۳ شبکه‌های باور عمیق

شبکه‌های باور عمیق، یکی از اولین مدل‌های غیرهمگشتی بود که با موفقیت آموزش معماری‌های عمیق را پذیرفت. معرفی شبکه‌های باور عمیق در سال ۲۰۰۶، نوزایی یادگیری عمیق فعلی را آغاز کرد. چراکه تا قبل از آن‌ها، بهینه‌سازی مدل‌های عمیق بسیار دشوار تلقی می‌شد. ماشین‌های هسته با توابع هدف محدب بر فضای تحقیق مسلط می‌شوند. شبکه‌های باور عمیق نشان دادند که معماری عمیق می‌تواند با عملکرد بهتر از ماشین‌های بردار پشتیبان هسته‌ای در مجموعه داده MNIST موفق باشند.

امروزه، شبکه‌های باور عمیق عملا مورد پسند واقع نمی‌شوند و به ندرت مورد استفاده قرار می‌گیرند. اما، به دلیل نقش مهمی که در تاریخچه یادگیری عمیق دارند، هنوز هم شناخته شده هستند.

۴.۴.۳ مدل‌های مولد مبتنی‌بر جریان

توانایی انجام تخمین چگالی مناسب در بسیاری از مسائل یادگیری ماشین کاربرد مستقیم دارد، اما بسیار سخت است. تخمین خوب، امکان انجام کارآمد بسیاری مسائل از جمله: نمونه داده‌های مشاهده نشده جدید اما واقع‌بینانه (تولید داده)، پیش‌بینی وقایع آینده، استنباط متغیرهای نهفته، پرکردن نمونه داده‌های ناقص و غیره را فراهم می‌کند.

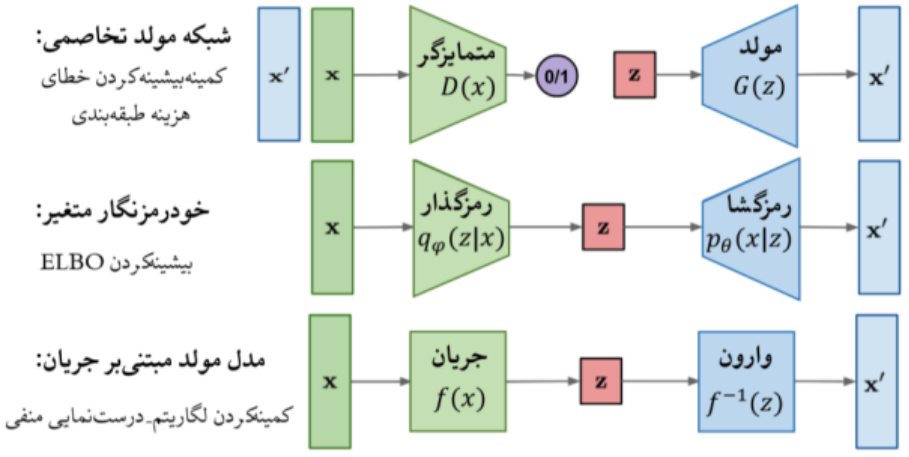
شبکه‌های مولد تخاصمی و خودرمزنگارهای متغیر، عملکرد چشم‌گیری را در کارهای چالش‌برانگیر همانند، یادگیری توزیع تصاویر واقعی از خود نشان داده‌اند. با این حال، چندین نقطه ضعف در این شبکه‌ها وجود دارد. هیچ یک از آن‌ها اجازه ارزیابی دقیق چگالی احتمالات نقاط جدید را نمی‌دهند. علاوه براین، آموزش شبکه‌های مولد تخاصمی، به دلیل انواع پدیده‌ها از جمله، محو گرادیان، بی‌ثباتی آموزش، فروپاشی حالت[۱] و فروپاشی پسین[۲] چالش‌برانگیر بوده و نیاز به تنظیم دقیق ابرپارمترها دارد.

[۱] mode collapse
[۲] posterior collapse

فروپاشی حالت

معمولا در شبکه مولد تخاصمی قصد داریم که خروجی‌های متنوعی تولید شود. به عنوان مثال، برای هر ورودی تصادفی به مولد، چهره متفاوتی می‌خواهیم. با این حال، اگر مولد یک خروجی قابل‌قبول تولید کند، ممکن است مولد یاد بگیرد که فقط همان خروجی را تولید کند. به عبارت دیگر، مولد همیشه در تلاش است تا خروجی را پیدا کند که برای متمایزگر منطقی به‌نظر برسد. اگر مولد بارها و بارها تولید همان خروجی را شروع کند، بهترین استراتژی متمایزگر این است که همیشه این نتیجه را رد کند. اما اگر نسل بعدی متمایزگر در کمینه محلی گیر کند و بهترین استراتژی را پیدا نکند، برای تکرار سازنده بعدی یافتن قابل قبول‌ترین نتیجه برای متمایزگر فعلی خیلی آسان است. هر تکرار از مولد برای یک متمایزگر خاص بیش‌ازحد بهینه می‌شود و متمایزگر هرگز موفق به یادگیری راه خود برای خروج از تله نمی‌شود. در نتیجه، مولدها از طریق مجموعه کوچکی از انواع خروجی دوران می‌کنند. به این شکل از شکست، فروپاشی حالت گفته می‌شود.

مدل‌های مبتنی‌بر جریان، خانواده‌ای از مدل‌های مولد با توزیع‌های رام‌شدنی هستند که در آن‌ها نمونه‌برداری و ارزیابی چگالی توسط جریان‌های یکسان‌ساز، می‌تواند کارآمدتر و دقیق‌تر باشد. مدل‌های مبتنی‌بر جریان، احتمال را مستقیما از طریق جریان‌های یکسان‌ساز بهینه می‌کنند. به‌عبارت دیگر، مدل‌های مبتنی‌بر جریان دقیقاً توزیع واقعی داده‌ها را فرا می‌گیرند و ارزیابی دقیق احتمال را امکان‌پذیر می‌کنند. در شکل ۳ـ ۱٤ مقایسه بین مدل‌های مولد عمیق مشاهده می‌شود.



شکل ۳ـ ۱٤ مقایسه مدل‌های مولد عمیق

مدل‌های مولد مبتنی‌بر جریان از توالی‌های وارون در جهت ایجاد تبدیل چگالی احتمال برای تقریب توزیع پسین استفاده می‌کند. جریان با یک متغیر اولیه شروع می‌شود و با اعمال مکرر قضیه تغییر متغیر آن را به یک متغیر با یک توزیع ساده نگاشت می‌کند. از آنجایی که مدل مبتنی‌بر جریان وارون‌پذیر است، تولید نمونه‌های مصنوعی با نمونه‌برداری از "توزیع ساده" و "جریان" از طریق نگاشت به صورت معکوس ساده است.

شایان ذکر است، مدل‌های مبتنی‌بر جریان، علی‌رغم ارائه ویژگی‌های بسیار جذاب از جمله، توانایی تخمین دقیق لگاریتم‌درست نمایی، سنتز کارآمد و استنباط دقیق متغیرهای نهفته در مقایسه با خودرمزنگارهای متغیر و شبکه‌های مولد تخاصمی کم‌تر مورد توجه قرار گرفته‌اند. این مدل‌ها برخلاف شبکه‌های مولد تخاصمی و خودرمزنگارهای متغیر، می‌توانند احتمال هر نمونه تولیدشده را محاسبه کنند.

تحقیقات در زمینه مدل‌های مولد مبتنی‌بر جریان را می‌توان به دو دسته تقسیم‌بندی کرد: مدل‌های کاملا مبتنی‌بر جریان یکسان‌ساز و مدل‌های مبتنی‌بر جریان خودبرگشتی. در ادامه به تشریح این مدل‌ها خواهیم پرداخت.

۱.۴.۴.۳ مدل‌های با جریان یکسان‌ساز

هدف مدل‌های مولد مبتنی‌بر جریان، تقریب یک توزیع از داده‌های واقعی $x \sim p(x)$ از مجموعه محدود از مشاهدات $\{x^{(i)}\}_{i=1}^{N}$ است. داده با یادگیری یک تبدیل وارون $z = f^{-1}(x)$ نگاشت از یک فضای نهان با چگالی شناخته‌شده $\pi(z)$ را به x مدل‌سازی و با استفاده از یک تابع نگاشت یک‌به‌یک $x = f(z)$ یک متغیر تصادفی جدید را می‌سازد:

$$z \sim \pi(z) \quad , \quad x = f(z) \ , \ z = f^{-1}(x)$$

که با استفاده از قضیه تغییر متغیرها داریم:

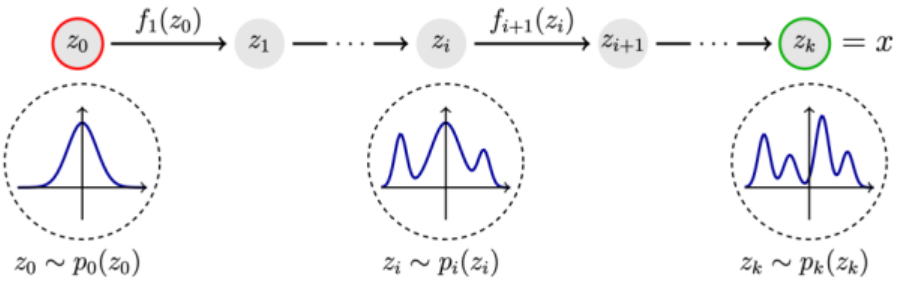
$$p(x) = \pi(z) \left| det \frac{dz}{dx} \right| = \pi\left(f^{-1}(x)\right) \left| det \frac{df^{-1}}{dx} \right|$$

جایی که در آن z یک متغیر نهفته و $\pi(z)$ به‌طور معمول یک توزیع گوسی $N(z; ۰, I)$ است. روش استنباط $p(x)$ از $\pi(z)$ را جریان یکسان‌ساز گویند. به عبارت دیگر، جریان یکسان‌ساز، توزیع احتمال (به عنوان مثال گوسی) را به یک توزیع پیچیده‌تر توسط دنباله‌ی از تبدیلات وارون امکان‌پذیر می‌کند.

حال بیاید با توجه به تصویر ۳ـ۱۵ قدم به قدم به نحوه تبدیل یک توزیع ساده همانند، $z_۰$ به یک توزیع پیچیده z_k بپردازیم.

با فرض این‌که داشته باشیم:

$$z_۰ \sim p_۰(z_۰) \ , \qquad z_k \sim p_k(z_k)$$



شکل ۳ـ۱۵ نحوه تبدیل توزیع ساده $z_۰$ به توزیع پیچیده z_k

برطبق تصویر ۳ـ۱۵ داریم:

$$z_{i-۱} \sim p_{i-۱}(z_{i-۱})$$
$$z_i = f_i(z_{i-۱})$$

بنابراین:

$$z_{i-۱} = f_i^{-۱}(z_i)$$

حال برای این‌که بتوان، از توزیع پایه استنتاج کرد؛ معادله را به تابعی از z_i تبدیل می‌کنیم:

$$p_i(z_i) = p_{i-۱}(f_i^{-۱}(z_i)) \left| det \frac{df_i^{-۱}}{dz_i} \right|$$

برطبق قضیه وارون تابع اگر؛ $y = f(x)$ و $x = f^{-1}(y)$. اگر داشته باشیم:

$$\frac{df^{-1}(y)}{dy} = \frac{dx}{dy} = \left(\frac{dy}{dx}\right)^{-1} = \left(\frac{df(x)}{dx}\right)^{-1}$$

پس داریم:

$$p_i(z_i) = p_{i-1}(z_{i-1})|\det\left(\frac{df_i}{dz_{i-1}}\right)^{-1}|$$

و برطبق خاصیت وارون تابع ماتریس ژاکوبین، داریم:

$$p_i(z_i) = p_{i-1}(z_{i-1})|\det\frac{df_i}{dz_{i-1}}|^{-1}$$

$$logp_i(z_i) = logp_{i-1}(z_{i-1}) - log|\det\frac{df_i}{dz_{i-1}}|^{-1}$$

با توجه به چنین زنجیره‌ای از توابع چگالی احتمال، رابطه بین هر جفت متغیر متوالی را می‌دانیم، پس می‌توانیم معادله خروجی x را گام به گام گسترش دهیم تا زمانی که به توزیع اولیه z_x برگردیم:

$$x = z_k = f_k \circ f_{k-1} \circ \dots f_1(z_0)$$

$$logp(x) = log\pi_k(z_k) = log\pi_{k-1}(z_{k-1}) - log|\det\frac{df_k}{dz_{k-1}}|$$

$$= log\pi_{k-2}(z_{k-2}) - log\left|\det\frac{df_{k-1}}{dz_{k-2}}\right| - log|\det\frac{df_k}{dz_{k-1}}|$$

$$= \cdots$$

$$= log\pi_0(z_0) - \sum_{i=1}^{k} log\left|\det\frac{df_i}{dz_{i-1}}\right|$$

مسیری که توسط متغیرهای تصادفی $z_i = f_i\ (z_{i-1})$ پیموده می‌شود را جریان و زنجیره کامل تشکیل‌شده توسط توزیع‌های متوالی π_i را جریان یکسان‌ساز می‌نامند.

۲.۴.۴.۳ مدل‌های با جریان خودبرگشتی

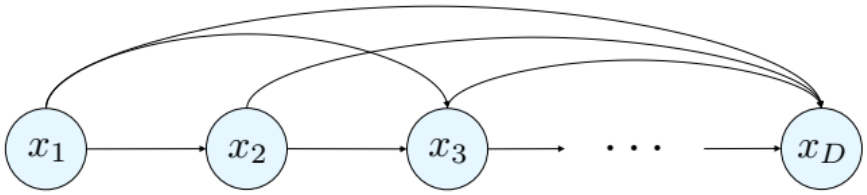
محدودیت خودبرگشتی، راهی برای مدل‌سازی داده‌های دنباله‌دار $X = [x_1, \dots, x_D]$ است. در مدل‌های خودبرگشتی، هر عنصر تنها به عناصر گذشته بستگی داشته و به عناصر آینده بستگی ندارد (ساختار این مدل در شکل ۳ـ۱۶ قابل مشاهده است).

به عبارت دیگر، احتمال مشاهده x_i مشروط به $x_1, \dots, x_{i-1}$ است و حاصل‌ضرب این احتمالات شرطی، احتمال مشاهده توالی کامل را به ما می‌دهد:

$$p(x) = \prod_{i=1}^{D} p(x_i | x_1, \dots, x_{i-1}) = \prod_{i=1}^{D} p(x_i | x_{1:i-1})$$

اگر یک تبدیل جریان در یک جریان یکسان‌ساز به عنوان یک مدل خودبرگشتی تنظیم شود، به عبارت دیگر، هر بعد در یک بردار متغیر به ابعاد قبلی مشروط شود، این یک جریان خودبرگشتی است.

جریان‌های خودبرگشتی، ترکیبی از تبدیلات خودبرگشتی برای ایجاد یک کلاس جدید از جریان‌های یکسان‌ساز به نام جریان‌های خودبرگشتی هستند که دستیابی به نتایج پیشرفته در انواع معیارها را برای تخمین و چگالی متغیرها را بدست آورده است.



شکل ۳ـ۱۶ نمایش گراف نشان‌دهنده وابستگی‌ها در یک مدل خودبرگشتی

خلاصه فصل

- موفقیت شگفت‌آور یادگیری عمیق، عمدتا با الگوریتم‌های بانظارت حاصل شده است. با این همه، در جهت دستیابی به عملکرد خوب برای آموزش این شبکه‌ها نیاز به مجموعه دادههای برچسب‌دار بسیار زیاد می‌باشد.

- علاقه مجدد به یادگیری بدون‌نظارت با معرفی دو مدل مولد عمیق جدید به‌ویژه، شبکه مولد تخاصمی و شبکه خودرمزنگار متغیر بوجود آمده است.

- خودرمزنگارها روشی را برای یادگیری خودکار ویژگی‌ها از داده‌های بدون برچسب ارائه می‌دهند که امکان یادگیری بدون‌نظارت را فراهم می‌کند.

- یک شبکه عصبی خودرمزگذار با تنظیم مقادیر خروجی هدف برابر با مقادیر ورودی، پس‌انتشار را انجام می‌دهد و بدین ترتیب خودرمزنگار آموزش داده می‌شود تا اختلاف بین داده‌ها و بازسازی آن را به حداقل برساند.

- یادگیری در خودرمزنگار، یادگیری خودنظارت‌شده نیز نامیده می‌شود. چراکه سیستم در واقع با استفاده از یک تابع هزینه و پس‌انتشار، به روشی نظارت‌شده آموزش می‌بیند؛ اما به داده‌های دارای برچسب نیازی ندارد.

- معماری یک خودرمزنگار به ترتیب از سه مولفه تشکیل می‌شود: رمزگذار، رمز و رمزگشا. رمزگذار ورودی را فشرده و رمز را تولید و رمزگشا ورودی را براساس رمز بازسازی می‌کند.

- رمزگذار و رمزگشا هر دو شبکه عصبی پیش‌خور هستند و در اکثر مواقع به‌صورت متقارن در ساختار خودرمزنگار قرار می‌گیرند.

- یکی از راه‌های یادگیری بازنمایی‌های مفید با خودرمزنگارها، محدود کردن اندازه رمز آن خودرمزنگار است. در این حالت، خودرمزنگار مجبور به استخراج ویژگی‌های برجسته از داده‌ها می‌شود.

- مدلهای مولد و تفکیک‌پذیر دو رویکرد متفاوت هستند که به‌طور گسترده در مسائل طبقه‌بندی مورد مطالعه قرار می‌گیرند.
- کار مدل‌های تفکیک‌پذیر در مقابل مدل‌های مولد ساده‌تر است. چرا که، اگر داده‌هایی با دسته‌های مختلف به آن نشان داده شود باید بتواند بین آنها تفاوت قائل شود. در مقابل، مدل‌های مولد کار سخت‌تری در پیش‌رو دارند. چرا که، باید توزیعی از داده‌ها را بدست آورده، درک کند و پس از آن به انجام طبقه‌بندی بپردازد.
- مدل‌های مولد را می‌توان به عنوان یک کلاس از مدل‌ها تعریف کرد که هدف آن‌ها یادگیری نحوه تولید نمونه‌های جدیدی است که به نظر می‌رسد از همان مجموعه داده‌های آموزشی هستند.
- مدل‌های مولد در زیرمجموعه روش‌های بدون‌نظارت قرار می‌گیرد. چرا که، سعی در یادگیری توزیع داده‌های مجموعه داده آموزشی را دارد.
- یک مدل مولد در تلاش است یک مساله تخمین چگالی را حل کند.
- مدل‌های مولد را می‌توان به‌طور کلی به دو دسته: مدل‌های ضمنی و آشکار تقسیم‌بندی کرد.
- مدل‌های مولد ضمنی یاد می‌گیرند که مستقیما نمونه (داده) تولید کنند.
- مدل‌های مولد عمیق را می‌توان به سه دسته اصلی تقسیم‌بندی کرد: مدل‌های مبتنی‌بر تابع هزینه، مدل‌های مبتنی‌بر انرژی و مدل‌های مبتنی‌بر جریان.
- خودرمزنگار متغیر نمونه‌ای از یک مدل مولد عمیق است.
- شبکه‌های مولد تخاصمی روشی محاسباتی براساس تئوری بازی‌ها هستند و در هسته آن ترکیبی از دو شبکه عصبی وجود دارد. یکی از این شبکه‌ها مولد و دیگری تمایزگر نامیده می‌شود.

- یادگیری در مدل مبتنی‌بر انرژی، شامل یافتن یک تابع انرژی است که در آن پیکربندی مشاهده شده متغیرها، انرژی کمتری نسبت به موارد مشاهده نشده داشته باشند.
- ماشین بولتزمن، یک معماری مبتنی‌بر انرژی است‌که توزیع احتمال را برروی متغیرهای ورودی خود می‌آموزد
- مدل‌های مبتنی‌بر جریان، احتمال را مستقیما از طریق جریانهای یکسان‌ساز بهینه می‌کنند.
- مدل های مبتنی‌بر جریان دقیقاً توزیع واقعی داده‌ها را فرا می‌گیرند.
- مدل‌های مولد مبتنی‌بر جریان، از توالی‌های وارون در جهت ایجاد تبدیل چگالی احتمال برای تقریب توزیع پسین استفاده می‌کند.

پرسش‌های مروری

۱. یادگیری بازنمایی چیست؟
۲. یادگیری فعال چیست و چه کابردی دارد؟
۳. آیا می‌توان از داده‌های بدون‌برچسب بازنمایی قدرتمندی بدست آورد؟
٤. خودرمزنگار روشی بانظارت است یا بدون‌نظارت؟
٥. قرار دادن لایه گلوگاه در خودرمزنگار چه مزیتی دارد؟
٦. چه عواملی در آموزش یک خودرمزنگار نقش دارند؟
۷. خودرمزنگار متغیر چه کاربردی دارد؟
۸. چرا از ترفند پارامترسازی مجدد در خودرمزنگار متغیر استفاده می‌شود؟

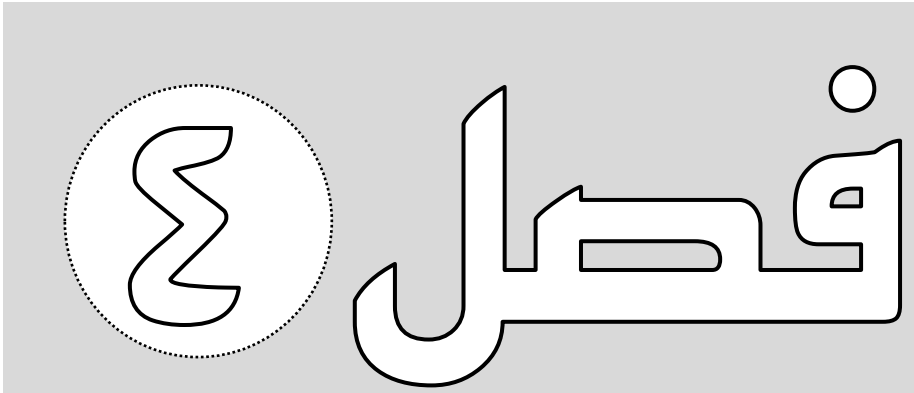
۹. نحوه کار شبکه مولد تخاصمی را شرح دهید؟

۱۰. مشکلات مدل مولد تخاصمی را شرح نام ببرید؟

۱۱. هدف اصلی مدل‌های مولد مبتنی‌بر انرژی چیست؟

۱۲. مزیت ماشین بولتزمن محدود در مقایسه با ماشین بولتزمن در چیست؟

۱۳. مزیت‌های مدل‌های مبتنی‌بر جریان را نام ببرید؟



یادگیری تقویتی عمیق

اهداف

- آشنایی با یادگیری تقویتی و تفاوت آن با یادگیری ماشین
- نحوه کار یادگیری تقویتی
- رویکردهای کلاسیک حل مسائل یادگیری تقویتی
- آشنایی با یادگیری تقویتی عمیق

۰.۴ مقدمه

یادگیری از طریق تعامل با محیط، احتمالا اولین رویکردی است که وقتی به ماهیت یادگیری فکر می‌کنیم به ذهن ما خطور می‌کند. این روش شناخته شده‌ای است که می‌دانیم، نوزاد از طریق آن یاد می‌گیرد. چنین تعاملات بدون شک منبع اساسی دانش بین محیط و ما، و نه فقط نوزادان در طول زندگی می‌باشد. به عنوان مثال، هنگامی که یاد می‌گیریم رانندگی اتومبیل را انجام دهیم، کاملا از نحوه پاسخگویی محیط به آنچه که انجام می‌دهیم آگاه هستیم و همچنین با اقدامات خود می‌خواهیم آنچه را که در محیط اتفاق می‌افتد را تحت تأثیر قرار دهیم. یادگیری از طریق تعامل یک مفهوم اساسی است که تقریباً در همه نظریه‌های یادگیری نهفته و پایه و اساس یادگیری تقویتی است.

اگرچه یادگیری نظارتی یکی از انواع مهم یادگیری به‌حساب می‌آید، اما به تنهایی برای یادگیری تعاملی کافی نیست. در مسائل تعاملی به دست آوردن نمونه‌هایی از رفتار مطلوب که هم صحیح باشد و هم نمایانگر همه موقعیت‌هایی که عامل باید در آن عمل کند، غالباً غیرعملی است. در جهت حل این نوع مسائل از یادگیری تقویتی استفاده می‌شود.

رویکرد یادگیری تقویتی، بیش از سایر رویکردهای یادگیری ماشین متمرکز بر یادگیری هدفمند از طریق تعامل است. در یادگیری تقویتی به مولفه یادگیرنده گفته نمی‌شود که همانند اشکال دیگر یادگیری ماشین، چه کارهایی باید انجام شود؛ بلکه عامل باید با آزمون و خطا و دریافت پاداش و تنبیه، کشف کند که کدام اعمال با تلاش آن بیشترین پاداش را به‌همراه دارد. به عبارت دیگر، یادگیری تقویتی نه با توصیف روش‌های یادگیری، بلکه با مشخص کردن یک مسئله، یادگیری تعریف می‌شود.

هنگام برخورد با مسائل دامنه با ابعاد بالا یا عامل‌های پیوسته ، یادگیری تقویتی از مشکل نمایش ناکارآمد ویژگی رنج می‌برد. بنابراین، زمان یادگیری کندی را به‌همراه دارد و باید تکنیک‌هایی برای سرعت بخشیدن به روند یادگیری طراحی شود. از همین‌رو، زمینه جدیدی با عنوان یادگیری تقویتی عمیق برای کمک به حل یادگیری تقویتی در مسائل با ابعاد بالا ظاهر شد. مهم‌ترین ویژگی یادگیری عمیق این است که شبکه‌های

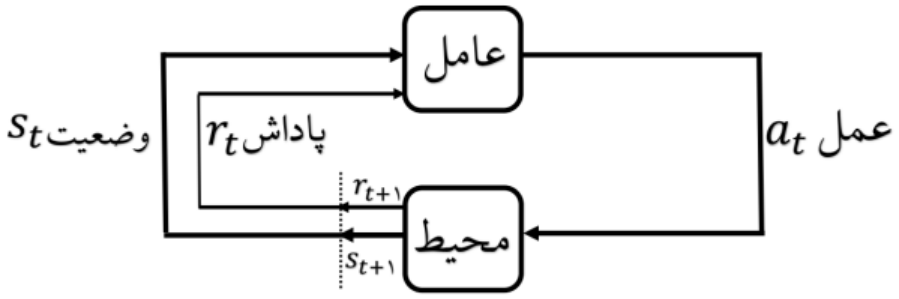
عصبی عمیق می‌توانند به صورت خودکار نمایش‌های کم حجم و فشرده از داده‌های با ابعاد بالا را پیدا کنند.

۱.۴ یادگیری تقویتی

تصمیم‌گیری‌ها در انسان، یک یادگیری مبتنی‌بر تجربه می‌باشد و از پاداش یا تنبیه دریافتی از محیط در جهت یادگیری برای تصمیم‌گیری در آینده استفاده می‌شود. به عبارت دیگر، یادگیری در موجودات هوشمند با آزمون و خطای هدایت شده توسط پاداش صورت می‌گیرد. علاوه براین، بسیاری از هوش و غریزه انسان در ژنتیک رمزنگاری شده که این فرآیند طی میلیون‌ها سال با یک روند محیط‌محور، که از آن با عنوان تکامل یاد می‌شود، تکامل یافته است.

با وجود اینکه یادگیری عمیق توانایی بازنمایی قدرتمند داده‌ها را دارد و در بسیاری از مسائل طبقه‌بندی و پردازش تصویر از بسیاری روش‌های دیگر بهتر عمل می‌کند، اما برای ساخت یک سیستم هوشمند هوش مصنوعی کافی نیست. این امر از آن‌جایی ناشی می‌شود که یک سیستم هوش مصنوعی نه تنها باید توانایی آموختن از داده‌ها را داشته باشد، بلکه باید همانند انسان از تعاملات با محیط دنیای واقعی نیز بیاموزد. یادگیری تقویتی یکی از حوزه‌های یادگیری ماشین می‌باشد و تمرکز بر این موضوع دارد تا ماشین را قادر به تعامل با محیط دنیای واقعی کند.

یادگیری تقویتی از طریق عامل در تلاش است با آزمون و خطا مساله را از طریق تعامل با محیطی که برای عامل ناشناخته می‌باشد، حل کند. عامل می تواند ضمن انجام بازخورد فوری از محیط، با اقدامات خود وضعیت محیط را تغییر دهد. بازخورد معمولا به عنوان **پاداش** در یادگیری تقویتی نامیده می‌شود. عامل با دریافت پاداش‌های مثبت بیشتر از محیط، توانایی یادگیری بهتر را کسب می‌کند. به‌طور کلی، **هدف عامل پیدا کردن یک زنجیره بهینه از اقدامات است تا مساله را حل کند.** یادگیری تقویتی، معمولا به عنوان یک فرآیند تصمیم‌گیری مارکوف مدل‌سازی می‌شود و می‌توان آن را همانند شکل ٤ ـ ١ توصیف کرد.



شکل ٤ـ ۱ نمایش یادگیری تقویتی

همان‌طور که در شکل ٤ـ ۱ قابل مشاهده است، واحد یادگیرنده که عامل نامیده می‌شود، با انتخاب عمل a از مجموعه اعمال ممکن، محیط خود را به‌طور فعال تغییر می‌دهد. پس از اجرای عملی، محیط مطابق با آن تغییر می‌کند و وضعیت جدید s را به عامل می‌گوید. علاوه براین، محیط با انتشار یک سیگنال پاداش r به عامل، بازخوردی درباره‌ی عمل انتخاب شده با توجه حالات داده‌شده ارائه می‌دهد. یک عامل همچنان به فعالیت خود ادامه می‌دهد، در نتیجه از یک وضعیت به وضعیت دیگر منتقل، تا زمانی که به یک حالت نهایی برسد.

عناصر تشکیل‌دهنده یک سیستم یادگیری تقویتی را می‌توان به‌صورت زیر فهرست کرد:

- **عامل:** برنامه‌ای که با هدف انجام کاری مشخص، آموزش می‌بیند.
- **محیط:** جهانی واقعی یا مجازی، که در آن عامل اقداماتی را انجام می‌دهد.
- **عمل:** حرکتی که توسط عامل انجام می‌شود و سبب تغییر وضعیت (حالت) در محیط می‌شود.
- **پاداش:** یک تابع پاداش، هدف را در یک مساله یادگیری تقویتی تعریف می‌کند و هر وضعیت مشاهده شده از محیط را به یک عدد واحد نگاشت می‌کند که مطلوبیت ذاتی آن وضعیت را نشان می‌دهد. هدف از پاداش در یادگیری تقویتی، ارزیابی یک عمل که می‌تواند مثبت یا منفی باشد و بازخوردی است که عامل پس از هر عمل از محیط دریافت می‌کند. پاداش واقعی یک عمل درست انجام شده در یک وضعیت خاص، ممکن است بلافاصله محقق نشود.

- **وضعیت:** کلیه اطلاعاتی که عامل در محیط فعلی خود در اختیار دارد. برای مثال در یک بازی شطرنج، وضعیت، موقعیت همه مهره‌ها در صفحه است.
- **مشاهدات:** با توجه به اینکه در برخی مسائل، عامل بر وضعیت کامل محیط دسترسی ندارد، معمولا مشاهده بخشی از وضعیتی است که عامل می‌تواند مشاهده کند. به عبارت دیگر، مشاهدات اطلاعاتی هستند که محیط در اختیار عامل قرار می‌دهد و آنچه در اطراف عامل رخ می دهد را نشان می‌دهد. با این حال، اغلب در ادبیات، وضعیت و مشاهده به‌جای یک‌دیگر استفاده می‌شوند.
- **خط‌مشی:** مشخص می‌کند که عامل با توجه به وضعیت فعلی چه اعمالی انجام خواهد داد. در زمینه یادگیری عمیق، می‌توانیم یک شبکه عصبی را آموزش دهیم تا این تصمیمات را بگیرد. در طول دوره آموزش، عامل سعی می‌کند خط‌مشی خود را اصلاح کند تا تصمیمات بهتری بگیرد. وظیفه یافتن خط مشی بهینه، بهبود خط‌مشی (کنترل) نامیده می‌شود و یکی از اصلی ترین مسائل در یادگیری تقویتی است.
- **تابع مقدار:** مشخص می‌کند که چه چیزی برای عامل در درازمدت خوب است. به عبارت دیگر، وقتی تابع مقدار را روی یک حالت معین اعمال می کنیم، اگر از آن حالت شروع کنیم، کل بازدهی را که می‌توان در آینده انتظار داشت را به ما می‌دهد.

مثال‌های ساده زیر کمک می‌کنند تا سازوکار یادگیری تقویتی را بهتر درک کنید:

فرض کنید گربه‌ای دارید و قصد دارید به آن آموزش دهید تا اقدامات خاصی را انجام دهد. از آن‌جا که گربه فارسی یا هر زبان دیگری را نمی‌فهمد، نمی‌توان به‌طور مستقیم به او گفت که چه کاری را باید انجام دهد. در عوض، می‌توان یک استراتژی متفاوتی را دنبال کرد. ما یک وضعیت را ارائه و گربه سعی می‌کند به طرق مختلف به آن پاسخ دهد. اگر پاسخ گربه روش مطلوبی باشد، به او ماهی می‌دهیم. اکنون هر زمان که گربه در معرض همان وضعیت قرار گیرد، گربه نیز با اشتیاق بیشتر انتظار مشابهی را برای دریافت پاداش (غذا) دارد. چراکه یادگرفته است که اگر یک عمل خاص را انجام دهد، پاداش دریافت می‌کند.

در این مثال:

- گربه عاملی است که در معرض محیط که در این مورد خانه می‌باشد، قرار می‌گیرد.

- وضعیت می‌تواند نشستن گربه باشد و شما از گفتن کلمه‌ای خاص برای راه رفتن گربه استفاده می‌کنید.
- عامل با انجام یک عمل، با انتقال از یک وضعیت به وضعیت دیگر واکنش نشان می‌دهد. به عنوان مثال، گربه از حالت نشستن به راه رفتن می‌رود.

می‌توان مثال دیگری از کودکان ارائه کرد. بچه ها اغلب اشتباهاتی را انجام می‌دهند. بزرگسالان سعی می‌کنند اطمینان حاصل کنند که کودک از این اشتباه درس گرفته و سعی می‌کنند دیگر آن را تکرار نکنند. در این حالت می‌توانیم از مفهوم بازخورد استفاده کنیم. اگر والدین سختگیر باشند، فرزندان را برای هرگونه خطا سرزنش می‌کنند، که بازخوردی منفی است. کودک از این پس با انجام این خطا به یاد می‌آورد که اقدام اشتباهی انجام داده است، چرا که با سرزنش والدین همراه خواهد بود. پس از آن بازخورد مثبتی نیز وجود دارد، جایی که والدین ممکن است آن‌ها را بخاطر انجام کاری درست، تحسین کنند. در اینجا، ما یک عمل صحیح را به طریقی خاص اجرا یا در تلاش برای اجرای آن هستیم.

به‌طور خلاصه می‌توان این‌گونه بیان کرد که، یادگیری تقویتی نوعی روش شناسی یادگیری است که در آن به الگوریتم با پاداش بازخورد می‌دهیم تا از آن یاد گرفته، تا بهبود نتایج را در آینده به‌همراه داشته باشد.

۱.۱.۴ یادگیری تقویتی در مقایسه با یادگیری ماشین

اگرچه یادگیری تقویتی زیرمجموعه‌ای از یادگیری ماشین می‌باشد، اما از جنبه‌های مختلفی با روش‌های یادگیری ماشین بانظارت و بدون‌نظارت، تفاوت اساسی دارد. اول اینکه، یادگیری تقویتی به استفاده از داده بستگی ندارد. در مقابل، در یادگیری تقویتی عامل از تجربیات خود به شیوه آزمون و خطا در خلال تعامل با محیط یاد می‌گیرد و به ناظر بستگی ندارد تا چه اقدامی را انجام دهد. ثانیاً، یادگیری تقویتی به‌جای تجزیه و تحلیل داده‌ها، در یافتن یک خط‌مشی مطلوب متمرکز است. همچنین، از آنجایی که عامل مستقیما از محیط یاد می‌گیرد، برای یادگیری به مقادیر زیادی از داده‌های آموزشی از قبل موجود در دامنه خاص، نیاز ندارد. بلکه، داده‌های آموزشی را خود صرفا با تعامل با محیطی که در آن قرار دارد و آموختن از تجربه‌ها ایجاد می‌کند.

۲.۴ فرآیندهای تصمیم‌گیری مارکوف

فرآیندهای تصمیم‌گیری مارکوف، یک مدل ریاضی تصادفی برای یک سناریوی تصمیم‌گیری است. در هر مرحله، تصمیم گیرنده یا به عبارت‌دیگر همان عامل، عملی را انتخاب می‌کند. در این مدل، بخشی از نتایج آن تصادفی و بخشی دیگر نیز نتیجه عمل است. فرآیندهای تصمیم‌گیری مارکوف، برای مدل‌سازی انواع مسائل بهینه‌سازی استفاده و از طریق برنامه‌نویسی پویا و یادگیری‌تقویتی قابل حل هستند.

فرآیند تصمیم‌گیری مارکوف، همانند یک نمودار جریانی با دایره‌هایی که وضعیت‌ها را نشان می‌دهند. از هر دایره فلش‌های خارج می‌شود که نمایانگر تمام اقدامات احتمالی قابل انجام از آن وضعیت می‌باشد. به عنوان مثال، یک فرآیند تصمیم‌گیری مارکوف در بازنمود یک بازی شطرنج، دارای وضعیت‌هایی است که نشان‌دهنده‌ی محل قرارگیری مهره‌ها صفحه شطرنج است و اعمالی که نمایانگر حرکات احتمالی براساس مهرهای در صفحه‌ی شطرنج می‌باشند.

یک ویژگی اصلی یک فرآیند تصمیم‌گیری مارکوف، این است که هر وضعیت باید تمام اطلاعات مورد نیاز عامل را برای تصمیم‌گیری آگاهانه در اختیار داشته باشد، الزامی که "دارایی مارکوف" نامیده می‌شود. اساساً دارایی مارکوف می‌گوید که نمی‌توان انتظار داشت عامل از خود حافظه تاریخی خارج از وضعیت خود داشته باشد. به عنوان مثال، وضعیت فعلی تخته شطرنج، همه مواردی را که برای انجام بهترین حرکت بعدی است را می‌گوید و نیازی به حرکاتی که قبلا انجام شده نیست تا آن‌ها را به‌خاطر بیاورد.

در عمل، برای اینکه یادگیری تقویتی بتواند مساله‌ای را حل کند، الزامی به الگو قرار دادن مساله دنیای واقعی نیست. به عنوان مثال، خاطره من از چگونگی بازی شطرنج توسط یک حریف خاص ممکن است در روند تصمیم‌گیری من در دنیای واقعی نقش داشته باشد، اما می‌توان با یادگیری تقویتی یک عامل بازی شطرنج برنده ساخت که نیازی به این اطلاعات نداشته باشد.

فرآیندهای تصمیم گیری مارکوف، توسط یک ۵تایی از عناصر $<S,A,P,R,\gamma>$ تعریف می‌شود، جایی که:

- **S:** مجموعه‌ای از وضعیت‌ها که شامل تمام نمایش‌های احتمالی محیط است.
- **A:** در هر وضعیت، محیط مجموعه‌ای از اعمال را در یک فضای عملیاتی در اختیار عامل قرار می‌دهد تا عامل، از این اعمال انتخاب نماید. عامل از طریق اعمال بر محیط تاثیر می‌گذارد.
- **P:** $P = (s, a, \acute{s}) = P_r(s_{t+1} = \acute{s} | s_t = s, a_t = a)$ ماتریس انتقالی است که احتمال می‌دهد عمل a در وضعیت s در زمان t منجر به وضعیت $\acute{s}$ در زمان $t + 1$ می‌شود.
- **R:** $R = (s, a, \acute{s})$ پاداش مورد انتظار است که عامل پس از عمل a در وضعیت s و رسیدن به وضعیت $\acute{s}$ دریافت می‌کند.
- **γ:** فاکتور نزول است و نشان‌دهنده اهمیت بین پاداش‌های کوتاه‌مدت و درازمدت می‌باشد.

مساله اساسی در فرآیندهای تصمیم‌گیری مارکوف، یافتن "خط‌مشی" برای تصمیم‌گیرنده است؛ یک تابع P که وضعیت‌ها را به اعمال $a = \pi(s)$ نگاشت می‌کند. این خط‌مشی می‌تواند قطعی یا تصادفی باشد. هدف این است که یک خط‌مشی را بیابیم که مجموع پاداش فاکتور نزول از هر وضعیت به وضعیت بعد را بیشینه کند:

$$G_t = \sum_{i=t}^{\infty} \gamma^i . R(s_i, a_i, s_{t+1})$$

از G_t به عنوان بازده نام برده می‌شود.

۳.۴ عامل

عامل کسی یا چیزی است که با انجام برخی اعمال، مشاهدات و دریافت پاداش‌های نهایی با این محیط ارتباط برقرار می‌کند. عامل مولفه‌ای است که براساس پاداش و تنبیه تصمیم می‌گیرد، چه اقدامی باید صورت پذیرد. برای تصمیم‌گیری، عامل مجاز است از هرگونه مشاهده از محیط و هرگونه قانون داخلی استفاده کند. این قوانین داخلی می‌توانند هر چیزی باشند، اما به‌طور معمول در یادگیری تقویتی، انتظار براین است که وضعیت فعلی توسط محیط تامین تا وضعیت، خصوصیت تصمیم‌گیری مارکوف را داشته باشد

و پس از آن، با استفاده از یک تابع خط‌مشی تصمیم می‌گیرد چه تصمیمی باید صورت گیرد.

در بیشتر سناریوهای عملی یادگیری تقویتی، عامل نرم‌افزار ماست که قرار است برخی از مسائل و مشکلات را به روشی کم‌وبیش کارآمد حل کند. عامل یکی از مهم‌ترین عناصر تشکیل‌دهنده یک سیستم مبتنی‌بر یادگیری تقویتی است. چرا که شامل هوش در جهت تصمیم‌گیری و توصیه اقدامات (اعمال) بهینه در هر شرایط است. از آنجا که عامل در یادگیری تقویتی نقش بسیار مهمی را داراست، تحقیقات زیادی در مورد معماری یادگیری و مدل‌های مرتبط انجام شده است. در ادامه الگوریتم‌ها را بر اساس عامل‌ها به الگوریتم‌های **مبتنی‌بر مقدار**، **مبتنی‌بر خط‌مشی** و **مبتنی‌بر مدل** تقسیم‌بندی می‌کنیم.

۱.۳.۴ الگوریتم‌های مبتنی‌بر مقدار

در الگوریتم‌های مبتنی‌بر مقدار، توابع مقدار $V^{\pi}(s)$ به وضعیت‌ها اختصاص داده می‌شوند و عامل تصمیمات خود را برمبنای مقادیر وضعیت‌ها می‌گیرد. تابع مقدار، تابعی است که میزان خوب بودن وضعیت را برمبنای پیش‌بینی پاداش آینده ارزیابی می‌کند. دو نوع متفاوت از تابع مقدار وجود دارد:

- **تابع وضعیت‌ـ‌مقدار** که معمولا به عنوان تابع مقدار خوانده می‌شود، بازده G_t مورد انتظار با شروع از یک وضعیت s و پیروی از خط‌مشی π است و توسط معادله بلمن به‌صورت زیر تعریف می‌شود:

$$V^{\pi}(s) = \mathbb{E}[G_t|S_t = s]$$

$$= \sum_{a\in A} \pi(a|s) \sum_{\acute{s}\in S} \pi(\acute{s}|s,a)[\mathrm{R} = (s,a,\acute{s}) + \gamma^{V^{\pi}(\acute{s})}]$$

- **تابع وضعیت‌ـ‌عمل** که معمولا با عنوان Qـ‌مقدار خوانده می‌شود، بازده G_t مورد انتظار یک جفت وضعیت‌ـ‌عمل در زمان t و پیروی از خط‌مشی π است و به‌طور مشابه توسط معادله بلمن به‌صورت زیر تعریف می‌شود:

$$Q^{\pi}(s,a) = \mathbb{E}[G_t|S_t = s, A_t = a]$$

$$= \sum_{\acute{s} \in S} P(\acute{s}|s,a)[\mathrm{R} = (s,a,\acute{s}) + \gamma \sum_{a \in A} \pi(\acute{a},\acute{s}) + Q^{\pi}(\acute{s},\acute{a})]$$

معادلات بلمن

معادلات بلمن به مجموعه معادلاتی گفته می‌شود که تابع مقدار را به پاداش فوری به علاوه مقادیر آتی تنزیل‌یافته تجزیه می‌کند.

$$V^{\pi}(s) = \mathbb{E}[G_t|S_t = s]$$
$$= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^{2} R_{t+3} + \ldots |S_t = s]$$
$$= \mathbb{E}[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \ldots)|S_t = s]$$
$$= \mathbb{E}[R_{t+1} + \gamma G_{t+1}|S_t = s]$$
$$= \mathbb{E}[R_{t+1} + \gamma V(S_{t+1})|S_t = s]$$

به‌طور مشابه برای Q ـ مقدار:

$$Q^{\pi}(s,a) = \mathbb{E}[R_{t+1} + \gamma V(S_{t+1})|S_t = s, A_t = a]$$
$$= \mathbb{E}[R_{t+1} + \gamma \mathbb{E}_{a \sim \pi} Q(S_{t+1}, a)|S_t = s, A_t = a]$$

معادله زیر بین توابع وضعیت ـ مقدار و وضعیت ـ عمل برقرار است:

$$V^{\pi}(s) = \sum_{a} \pi(a|s)\, Q^{\pi}(s,a)$$

هدف عامل بیشینه‌کردن کل پاداش تجمعی در بلند مدت است. به عبارتی دیگر، هدف از یادگیری تقویتی، یافتن خط‌مشی بهینه است. خط‌مشی که حداکثر پاداش تجمعی را بیشینه می‌کند، خط‌مشی بهینه نامیده می‌شود و با π^{*} نشان داده می‌شود. خط‌مشی بهینه π^{*} به این صورت است که مقدار هر وضعیت s تحت π^{*} بیشتر یا برابر با مقدار وضعیت s تحت خط‌مشی دیگر π' برای همه‌ی $s \in S$ باشد:

$$V^{\pi^{*}}(s) = V^{*}(s) \geq V^{\pi'} \; \forall \, s \in S, \pi'$$

اگر تابع وضعیت_مقدار بهینه باشد، عامل از خط‌مشی بهینه استفاده کرده و ممکن است چندین خط‌مشی بهینه وجود داشته باشد که منجر به همان تابع وضعیت_مقدار بهینه شود. تابع وضعیت_مقدار بهینه V^* را می توان به شرح زیر تعریف کرد:

$$V^*(s) = max_\pi V^\pi(s) \; \forall \; s \in S$$

علاوه براین، خط‌مشی بهینه منجر به تابع وضعیت_عمل بهینه Q^* نیز می‌شود:

$$Q^*(s,a) = max_\pi Q^\pi(s,a) \; \forall \; s \in S, a \in A$$

$$= \mathbb{E}[R_{t+1} + \gamma V^*(\acute{s})|S_t = s, A_t = a]$$

در پایان، می‌توان معادله بهینه‌سازی بلمن را از معادلات معرفی شده قبلی استخراج کرد:

$$V^*(s) = max_{a \in A(s)} Q^{\pi^*}(s,a)$$

$$= max_a \; \mathbb{E}[R_{t+1} + \gamma V^*(\acute{s})|S_t = s, A_t = a]$$

$$= max_a \sum_{\acute{s} \in S} P(\acute{s}|s,a)[\mathrm{R}(s,a,\acute{s}) + \gamma V^*(\acute{s})]$$

$$= max_a \sum_{\acute{s} \in S} P(\acute{s}|s,a)[\mathrm{R}(s,a,\acute{s}) + \gamma max_{\acute{a}} Q^{\pi^*}(\acute{s},\acute{a})]$$

نمودار پشتیبان‌گیری در یادگیری تقویتی

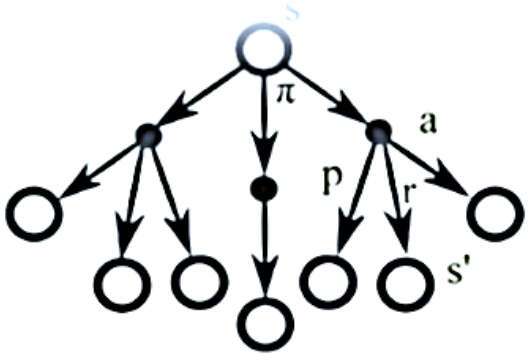
معادلات بلمن با تجسم یک گام مجزا در زمان با استفاده از درختی از حالات و کنش‌ها (نمودار پشتیبان‌گیری) بهتر قابل درک هستند. نمودار پشتیبان‌گیری، یک نمایش بصری از الگوریتم‌ها و مدل‌های مختلف در یادگیری تقویتی را نشان می‌دهد.

فرآیند پشتیبان‌گیری (عملیات بروزرسانی)، نمایش گرافیکی الگوریتم، با نشان‌دادن وضعیت، عمل، وضعیت انتقال، پاداش و غیره است. در این نمودار وضعیت_مقدار

با یک دایره توخالی درحالیکه عمل_مقدار با یک دایره توپر نمایش داده می‌شود. همچنین، عمل با یک فلش شروع‌شده از وضعیت نمایش داده می‌شود.

حال بیایید نحوه نمایش یک وضعیت_مقدار را با استفاده از نمودارهای پشتیبان‌گیری بررسی کنیم (شکل ٤_٢).

۱. s حالت شروع و گره اصلی است.
۲. از حالت s سه عمل می‌تواند صورت گیرد. همان‌طور که توسط فلش نشان‌داده شده و عامل طبق خط‌مشی π عمل می‌کند.
۳. در صورتی که یک محیط تصادفی با احتمال انتقال معین باشد، این عامل می‌تواند در حالت‌های مختلفی به پایان برسد. همان‌طور که در شکل مشاهده می‌شود ۳ حالت احتمالی نشان داده شده که عامل می‌تواند پس از انجام بیشترین عمل درست به آن وارد شود.



شکل ٤_٢ نمودار پشتیبان‌گیری وضعیت_مقدار $V^{\pi}(s)$

۲.۳.۴ الگوریتم‌های مبتنی‌بر خط‌مشی

خط‌مشی، استراتژی‌ای است که سازوکار یادگیری برای تعیین بهترین اقدام بعدی را براساس وضعیت فعلی امکان‌پذیر می‌کند. خط‌مشی، هسته اصلی عامل یادگیری تقویتی است؛ به این معنا که به تنهایی برای تعیین رفتار کافی است.

الگوریتم‌های مبتنی‌بر خط‌مشی، در مقابل با الگوریتم‌های مبتنی‌بر مقدار، رویکرد مستقیم‌تری را در پیش می‌گیرند. در این رویکرد، به‌جای یافتن مقدار هر وضعیت ممکن و سپس بدست آوردن خط‌مشی بهینه، به دنبال یافتن مستقیم خط‌مشی هستند تا بازده مورد انتظار بیشینه شود.

به عبارت دیگر، هسته اصلی روش‌های مبتنی‌بر خط‌مشی، بروزرسانی مجدد پارامترهای خط‌مشی θ است؛ به‌طوری که بازده G_t مورد انتظار افزایش یابد. به عبارت ساده‌تر، در رویکرد مبتنی‌بر خط‌مشی، در ابتدا یک خط‌مشی تصادفی انتخاب و در مرحله ارزیابی تابع مقدار آن خط‌مشی پیدا می‌شود. سپس، خط‌مشی جدید را با استفاده از تابع مقدار محاسبه تا در هر مرحله بهبود پیدا کند. این روند را تا زمانی که خط‌مشی بهینه پیدا شود تکرار می‌کند. روند بهینه‌سازی به‌صورت زیر توصیف می‌شود:

$$\theta_{i+1} = \theta_i + \Delta\theta_i$$

جایی که θ_i مجموعه‌ای از پارامترهای خط‌مشی است که برروی خط‌مشی π پارامترسازی می‌شوند و $\Delta\theta_i$ تغییرات پارمترهای خط‌مشی است.

۳.۳.۴ استخراج[۱] در مقابل اکتشاف[۲]:

انسان‌ها قبل از انجام هرکاری سعی می‌کند هرچه بیشتر اطلاعات کسب کنند. به عنوان مثال، قبل از اینکه یک رستوران جدید را امتحان کند، سعی می‌کند نظرات را بخواند یا از دوستانی که قبلاً آن را امتحان کرده‌اند پرسش‌وجو کند. از طرف دیگر، در یادگیری تقویتی، انجام این کار امکان پذیر نیست. اما برخی از تکنیک‌ها وجود دارد که به کشف بهترین استراتژی کمک می‌کند.

در یادگیری تقویتی، عامل در زمان تصمیم گیری در محیط، می‌تواند دو استراتژی را اعمال کند:

- **اکتشاف:** یک عمل تصادفی را انتخاب کند. با پیروی از این روش، عامل می‌تواند از وضعیت‌های جدید بازدید کند و خط‌مشی‌های جدید و بهتری را

[۱] exploitation
[۲] exploration

پیدا کند. نتیجه آن، جمع‌آوری اطلاعات بیشتر که ممکن است به تصمیم‌گیری بهتر در آینده منجر شود.

- **استخراج:** حریصانه عمل کند. با استفاده از پاداش کلی، بهترین اعمال شناخته شده را با توجه دانش موجود انتخاب کند. به عبارت دیگر، قصد دارد با توجه به اطلاعات فعلی بهترین تصمیم را بگیرد.

در یادگیری تقویتی به این نوع دو از تصمیم‌گیری: زمانی که ادامه کار قبلی را انجام می‌دهید استخراج و در هنگام امتحان کارهای دیگر اکتشاف گفته می‌شود.

یکی از ویژگی‌های کلیدی یادگیری تقویتی، معضل استخراج در مقابل اکتشاف است. اگر عامل بخواهد اعمال بهتری بیآموزد یا به عبارت دیگر، اعمالی که در نهایت به پاداش انباشته بیشتری منجر شود، باید اعمال جدید را امتحان کند. همچنین اگر عامل از دانش فعلی خود استخراج کرده و اقدامات قبلاً شناخته شده برای بازخورد پاداش‌های خوب را دنبال کرده باشد، تضمین نمی شود که بازدهی بالاتر از پاداشی که عامل می‌توانست دریافت کند، داشته باشد. بنابراین این معضلی است که یک عامل هنگام تصمیم‌گیری برای اعمال بعدی با آن روبرو می‌شود: یا اعمال تصادفی را امتحان کند و حدس بزند که پاداش‌های بیشتری می‌گیرد، اما با خطر بدتر شدن امتیاز مواجه است. یا اینکه با عملکرد مطابق با شرایط فعلی خود به پاداش احتمالاً پایین‌تر اما مطمئن دست پیدا کند. به عبارت دیگر، اگر عامل تنها اکتشاف را انجام دهد، ممکن است در این کار به امتیازات بالاتری نرسد و اعمال خود را بهبود ندهد. در طرف دیگر، اگر فقط از استخراج استفاده شود، ممکن است در خط‌مشی فعلی خود با دیدن تمام خط سیرهای احتمالی گیر کند. از همین رو، عامل احتمالا خط‌مشی بهینه را از دست خواهد داد. بنابراین، باید یک تعادل خوب بین اکتشاف و استخراج وجود داشته باشد.

این معضل از آن‌جایی ناشی می‌شود که، فرآیند یادگیری در یادگیری تقویتی به‌صورت برخط صورت می‌گیرد. به عبارت دیگر، به یادگیری تقویتی هیچ داده‌ای همانند یادگیری بانظارت داده نمی‌شود. بنابراین، عامل خود به نوعی به‌دنبال جمع‌آوری داده‌ها است و از طریق اعمالی که انجام می‌دهد، بر داده‌های مشاهده‌شده اثر می‌گذارد. از همین رو، گاهی ارزش دارد که اعمال مختلفی را برای بدست آوردن داده‌های جدید انجام دهد.

معضل اکتشاف در مقابل استخراج یک موضوع تکرارشونده در یادگیری تقویتی و به‌طور کلی هوش مصنوعی است. آیا باید از دانش به دست آمده استخراج کنیم، یعنی باید یک مسیر شناخته شده با پاداش بالا را دنبال کنیم؟ یا باید در جستجوی خط‌مشی جدید بهتر، وضعیت‌های ناشناخته را کشف کنیم؟ تعادل بین هر دو استراتژی تصمیم‌گیری، عملکرد یادگیری عامل بسیار بهبود می‌بخشد. یک پاسخ قابل قبول برای حل آن، این است که ابتدا یک عامل لازم است بیشترین وضعیت‌ها را کشف و به دنبال آن با استخراج از دانش جمع‌آوری شده، نتایج بهتری را کسب کند. اما نباید فراموش کرد که، در محیط‌های پویا، بدست‌آوردن این‌که آیا اکتشاف کافی صورت گرفته یا نه بسیار پیچیده می‌باشد. با این وجود، چند روش برای انتخاب اعمال ارائه شده است که در ادامه به تشریح آن‌ها می‌پردازیم.

استخراج: به‌معنای بیشنه‌کردن عملکرد عامل با استفاده از دانش موجود است که عملکرد آن معمولاً با پاداش مورد انتظار ارزیابی می‌شود. به عنوان مثال، یک جوینده طلا اکنون یک سنگ معدن دارد که روزانه دو گرم طلا برای او فراهم می‌کند و او می‌داند که بزرگترین سنگ طلا می‌تواند پنج گرم طلا در روز به او بدهد. با این حال، همچنین می‌داند که یافتن یک سنگ معدن جدید نه تنها او را مجبور به متوقف کردن استخراج از سنگ معدن فعلی می‌کند، بلکه برای او هزینه اضافی را نیز در پی دارد چرا که در نهایت خطر پیدا نکردن هیچ چیز را نیز به همراه دارد. با توجه به این موارد، او تصمیم می‌گیرد سنگ معدن فعلی را حفر کند تا حداکثر پاداش (در این حالت طلا) از طریق استخراج بیشتر شود و با توجه به خطرات بزرگ اکتشاف، از اکتشاف صرف نظر می‌کند. خط‌مشی که وی در اینجا اتخاذ کرده خط‌مشی حریصانه است، به این معنی که عامل بطور مداوم عملی را انجام می‌دهد که بالاترین پاداش مورد انتظار را براساس اطلاعات فعلی بدست می آورد، نه اینکه آزمایش‌های خطرناکی را انجام دهد که منجر به کاهش پاداش‌های مورد انتظار شود.

اکتشاف: به‌معنای افزایش دانش موجود با اقدامات و تعامل با محیط است. به مثال حفار طلا برگردیم، او آرزو دارد مدتی را صرف یافتن سنگ معدن جدید کند و اگر سنگ معدن طلای بزرگتری پیدا کرد، می‌تواند روزانه جایزه بیشتری دریافت کند. برای داشتن بازده بلندمدت بیشتر، ممکن است بازده کوتاه مدت فدا شود. حفار طلا همیشه با معضل استخراج و اکتشاف روبرو است، چراکه باید تصمیم بگیرد که معدن طلا چه میزان بازده برای ماندن دارد و بازده معدن طلا برای ادامه کار در او چقدر است. تعامل بین اکتشاف و استخراج تعادل بین میزان تلاش عامل به ترتیب در اکتشاف و بهره برداری را توصیف می‌کند. تعامل بین اکتشاف و استخراج، موضوع اصلی تحقیقات یادگیری تقویتی و توسعه الگوریتم‌های یادگیری تقویتی است.

استراتژی انتخاب حریصانه ـ اپسیلون

یک روش ممکن، ساده و محبوب و در عین حال موثر برای انتخاب عمل در هر مرحله به عنوان استراتژی انتخاب، انتخاب حریصانه ـ اپسیلون است. در این روش، یک پارامتر e بین ۰ تا ۱ اقدام عامل مبنی‌بر اینکه، استخراج یا اکتشاف انجام می‌دهد را کنترل می‌کند. با استفاده از این روش در هر زمان عامل به‌طور احتمالی بین اکتشاف و استخراج یکی را انتخاب می‌کند. با احتمال e و انتخاب تصادفی از بین تمام اعمال موجود و با احتمال e ـ ۱ اکتشاف می‌کند.

مقادیر بالای e عامل را وادار به جستجوی بیشتر و در نتیجه، احتمال اعمال بهینه را کاهش می‌دهد. در حالی که، به عامل توانایی واکنش سریع به تغییراتی را می‌دهد که در محیط ایجاد می‌شود. در مقابل، مقادیر کم e عامل را به سوی اعمال بهینه‌تر سوق می‌دهد.

اکتشاف بولتزمن

روش دیگر برای انتخاب اعمال، خط‌مشی توزیع بولتزمن است. توزیع بولتزمن، یک خط‌مشی یادگیری است که با گذشت زمان تمایل به اکتشاف را کاهش می‌دهد. و بر این فرض است که با پیشرفت یادگیری، مدل فعلی بهبود می‌یابد. توزیع بولتزمن، احتمالی را با استفاده از یک پارامتر T به نام دما به هر عمل اختصاص می‌دهد.

توزیع بولتزمن، با استفاده از معادله زیر یک احتمال مثبت را برای هر عمل ممکن $a\epsilon A$ اختصاص می‌دهد:

$$P(a|s) = \frac{e^{\frac{Q(s,a)}{T}}}{\sum \acute{a}\epsilon A e^{\frac{Q(s,\acute{a})}{T}}}$$

جایی که

$$T_{\text{جدید}} = e^{-dj} * T_{\text{بیشینه}} + ۱$$

عمل با $Q(s,a)$ بیشتر با احتمال P بیشتری همراه است. T با افزایش تکرار j با گذشت زمان کاهش پیدا می‌کند. بنابراین، با پیشرفت یادگیری گرایش به اکتشاف در عامل کاهش پیدا می‌کند و در نتیجه، خط‌مشی یادگیری توزیع بولتزمن تمایل به استخراج

از اعمال با $Q(s,a)$ بالا را دارد. پارامترهای بیشینه T و نرخ تنزل dj در ابتدا تنظیم می‌شوند.

۴.۳.۴ مبتنی‌بر مدل در مقابل بدون مدل

همان‌طور که پیش‌تر بیان شد، هدف اصلی عامل در یادگیری تقویتی، جمع‌آوری بیشترین میزان پاداش در "درازمدت" است. در جهت انجام این کار، عامل باید یک خط‌مشی بهینه برای رفتار در محیط پیدا کند. محیط می‌تواند قطعی یا تصادفی (غیرقطعی) باشد. یعنی اگر عامل در یک وضعیت خاص عملی را انجام دهد، وضعیت بعدی حاصل از محیط ممکن است لزوما همیشه یکسان نباشد. قطعا این عدم اطمینان‌ها، مساله را برای یافتن خط‌مشی بهینه دشوارتر می‌کند.

پیش‌بینی در مقابل بازبینی

پیش‌بینی و بازبینی به دو مساله اساسی اشاره دارد که یک عامل یادگیری تقویتی باید آن را حل کند. پیش‌بینی، مستلزم توانایی محاسبه یا برآورد پیامدهای یک عمل است. مسائل پیش‌بینی معمولا به معنای برآورد مقادیر وضعیت‌ها یا مقادیر عمل جفت‌ها وضعیت_عمل است. در مقابل، بازبینی نیاز به توانایی تصمیم‌گیری دارد. بدون بازبینی، عامل هرگز عملی را انجام نمی‌دهد.

مساله پیش‌بینی: با توجه به فرآیند تصمیم‌گیری مارکوف <S,A,P,R,γ> و خط‌مشی π، باید تابع مقدار $v(\pi)$ پیدا شود. به عبارت دیگر، هدف این است که بفهمیم یک خط‌مشی چقدر خوب است.

مساله بازبینی: با توجه به فرآیند تصمیم‌گیری مارکوف <S,A,P,R,γ>، باید تابع مقدار بهینه $v(\pi)$ و خط‌مشی بهینه π^* پیدا شود. به عبارت دیگر، هدف این است که خط‌مشی پیدا شود که بیشترین پاداش را با بهترین عمل برای انتخاب در اختیار قرار دهد.

همان‌طور که می‌دانیم، در یادگیری تقویتی مساله اغلب از منظر ریاضی به عنوان یک فرآیند تصمیم‌گیری مارکوف صورت می‌گیرد. فرآیند تصمیم‌گیری مارکوف، روشی برای نمایش "پویایی" محیط است؛ یعنی، نحوه واکنش محیط به اقدامات احتمالی عامل در یک وضعیت خاص. به عبارت دقیق‌تر، فرآیند تصمیم‌گیری مارکوف به یک تابع انتقال مجهز شده که تابعی است که با توجه به وضعیت فعلی محیط و علمی که ممکن است

عامل انجام دهد، احتمال انتقال به هر یک از خروجی‌ها را ایجاد می‌کند. یک تابع پاداش نیز با فرآیند تصمیم‌گیری مارکوف مرتبط است.

تابع پاداش با توجه به وضعیت فعلی محیط و احتمالا عملی که توسط عامل و وضعیت بعدی محیط انجام می‌شود، پاداش دریافت می‌کند. توابع پاداش و انتقال اغلب الگوی محیط نامیده می‌شوند. با این حال، گاهی اوقات توابع پاداش و انتقال را نداریم. از همین رو، نمی‌توانیم خط‌مشی را تخمین بزنیم، چرا که ناشناخته است. در غیاب این توابع، در جهت تخمین خط‌مشی بهینه، نیاز به تعامل با محیط و مشاهده پاسخ‌های آن است که اغلب به عنوان "مشکل یادگیری تقویتی" نامیده می‌شود. چرا که، عامل باید با تقویت باورهای خود در مورد پویایی محیط، یک خط‌مشی را تخمین بزند.

با گذشت زمان، عامل شروع به درک نحوه واکنش محیط به اقدامات خود می‌کند و می‌تواند خط‌مشی بهینه را تخمین بزند. بنابراین، در مسائل یادگیری تقویتی، عامل خط‌مشی بهینه را برای رفتار در یک محیط ناشناخته با تعامل با آن، با استفاده از روش "آزمون و خطا" تخمین می‌زند. بر این اساس، الگوریتم‌های یادگیری تقویتی را می‌توان به الگوریتم‌های مبتنی‌بر مدل یا بدون مدل است تقسیم‌بندی کرد.

در الگوریتم‌های مبتنی‌بر مدل، عامل به یک مدل کامل از محیط دسترسی دارد، یا سعی می‌کند آن را از طریق تعامل بیاموزد و به‌منظور برآورد درست خط‌مشی بهینه از تابع انتقال و پاداش استفاده می‌کند. به عبارت دیگر، عامل سعی می‌کند مدل احتمالاتی را نمونه‌گیری کرده و بیاموزد و از آن برای تعیین بهترین اعمال استفاده کند. عامل ممکن است فقط به تقریبی از توابع انتقال و پاداش دسترسی که توسط عامل آموخته می‌شود، دسترسی داشته باشد. در حالی که، با محیط تعامل دارد یا می‌تواند به عامل مثلا توسط عامل دیگری داده شود.

به‌طور کلی، در یک الگوریتم مبتنی‌بر مدل، عامل می‌تواند به‌طور بالقوه پویایی محیط را در طول یا بعد از مرحله یادگیری پیش‌بینی کند. چراکه، تخمینی از توابع انتقال و تابع پاداش را دارد و اگر احتمال انتقال با موفقیت آموخته شود، عامل می‌داند که با توجه به وضعیت فعلی و عمل، چقدر احتمال دارد به یک وضعیت خاص وارد شود. با این حال، باید توجه داشت که توابع انتقال و پاداشی که عامل برای بهبود تخمین خط‌مشی بهینه

خود استفاده می‌کند، ممکن است فقط تقریبی از توابع "واقعی" باشد. از این‌رو خط‌مشی بهینه به دلیل این تقریب‌ها ممکن است هرگز یافت نشود.

در مقابل الگوریتم‌های مبتنی‌بر مدل، الگوریتم‌های بدون مدل هیچ دانش اولیه‌ای در مورد تابع انتقال ندارد و باید ضمن یادگیری در یافتن مسیرهای کارآمد، آن را بیآموزد. به عبارت دیگر، یک الگوریتم بدون مدل، یا "تابع مقدار" یا "خط‌مشی" را مستقیما از تجربه، یعنی با تعامل بین عامل و محیط تخمین می‌زند، بدون اینکه از توابع انتقال و پاداش استفاده کند.

یکی از راه‌های تمیز دادن بین روش‌های مبتنی‌بر مدل و بدون مدل این است که: آیا عامل می‌تواند پیش از شروع هر اقدامی پس از یادگیری، وضعیت و پاداش بعدی را پیش‌بینی کند. به عبارت دیگر، راه تمایز بین الگوریتم‌های مبتنی‌بر مدل یا بدون مدل این است که الگوریتم ها را بررسی کنید و ببینید آیا از توابع انتقال و پاداش استفاده می‌کنند یا نه. اگر استفاده می‌کند، یک الگوریتم یادگیری تقویتی مبتنی‌بر مدل است.

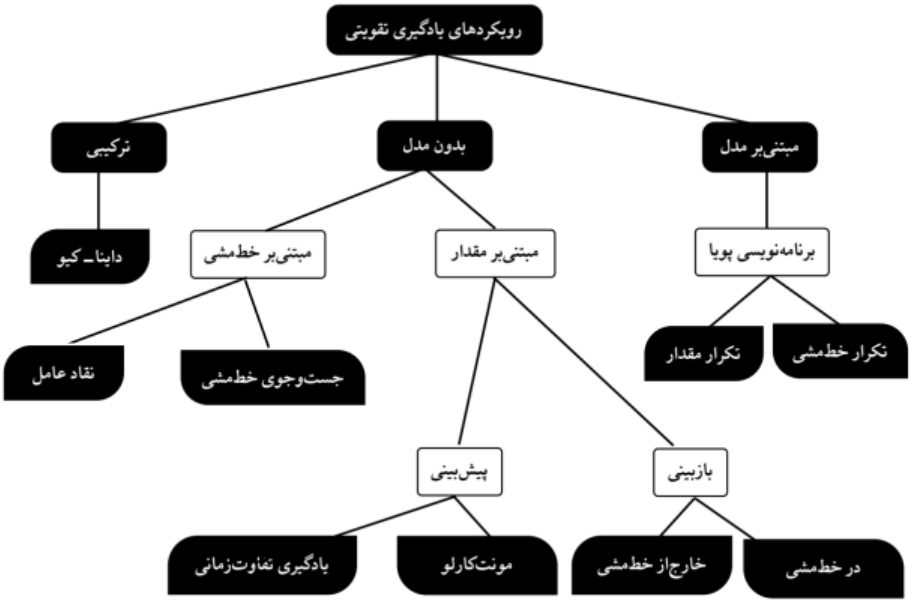
هر دو روش دارای نقاط قوت و ضعف هستند. روش‌های بدون مدل تاحدودی تضمین می‌کنند که در نهایت خط‌مشی بهینه را پیدا می‌کنند و زمان محاسبه بسیار کمی در هر آزمایش دارند. با این حال، آن‌ها از داده‌ها در طول آزمایشات بسیار ناکارآمد استفاده می‌کنند و بنابراین اغلب برای دستیابی به عملکرد خوب به تجربه زیادی نیاز دارند. در مقابل، الگوریتم‌های مبتنی‌بر مدل می‌توانند بر این مشکل فائق آیند، اما عامل تنها برای یک مدل خاص یاد می‌گیرد و گاهی اوقات برای برخی از مدل‌های دیگر مناسب نیست. همچنین، برای یادگیری یک مدل دیگر نیاز به زمان دارد.

مزیت مهم در داشتن مدل این است که به عامل اجازه می‌دهد تا از قبل برنامه‌ریزی کند و ببیند برای طیف وسیعی از اعمال مختلفی که می‌تواند در وضعیت فعلی خود انجام دهد چه اتفاقی می‌افتد. پس از آن، هنگام تصمیم‌گیری برای عمل به مقایسه این نتایج بپردازد. این امر می‌تواند در مقایسه با الگوریتم‌هایی که از مدل استفاده نمی‌کنند، منجر به بهبود قابل توجهی در کارایی شود. الگوریتم‌های بدون مدل برای بروزرسانی دانش خود، تنها به "آزمون و خطا" اعتماد می‌کنند. از همین‌رو، به ذخیره تمام ترکیبات وضعیت‌ها و اعمال، نیازی ندارند.

رویکرد بدون مدل را می‌توان به‌صورت در_خط‌مشی[۱] و خارج‌از_خط‌مشی[۲] طبقه‌بندی کرد. روش‌های در_خط‌مشی، از خط‌مشی فعلی برای ایجاد اقدامات استفاده می‌کنند و از آن برای بروزرسانی خط‌مشی فعلی استفاده می‌کنند. در حالیکه، در روش‌های خارج‌از_خط‌مشی، از یک خط‌مشی اکتشافی متفاوت برای ایجاد اقدامات در مقایسه با خط‌مشی که بروز می‌شود، استفاده می‌کنند.

۴.۴ رویکردهای کلاسیک حل مسائل یادگیری تقویتی

اکنون که با مفاهیم پایه یادگیری تقویتی آشنا شده‌اید، در این بخش قصد داریم تا رویکردهای کلاسیک یادگیری تقویتی در حل مسائل را تشریح کنیم. این رویکردها را می‌توان براساس رویکردهای مبتنی‌بر مدل، بدون مدل و ترکیبی از این دو همانند شکل ٤_۳ تقسیم‌بندی کرد. در ادامه به بررسی این رویکردها می‌پردازیم.



شکل ٤_۳ نحوه تقسیم‌بندی رویکردهای یادگیری تقویتی

[۱] on-policy
[۲] off-policy

۱.۴.۴ برنامه‌نویسی پویا

اصطلاح برنامه نویسی پویا، به مجموعه‌ای از الگوریتم‌ها گفته می‌شود که می‌تواند برای محاسبه خط‌مشی بهینه با توجه به مدل کاملی از محیط به عنوان فرآیند تصمیم‌گیری مارکوف مورد استفاده قرار گیرد. ایده اصلی برنامه نویسی پویا و به طور کلی یادگیری تقویتی، استفاده از توابع مقدار برای سازماندهی و ساختار جستجوی خط‌مشی‌های خوب است. الگوریتم های کلاسیک برنامه‌نویسی پویا، به دلیل فرض داشتن یک مدل کامل و همچنین به دلیل هزینه محاسباتی زیاد، کاربرد محدودی در یادگیری تقویتی دارند. با این همه، این روش‌ها از لحاظ نظری هنوز هم مهم هستند.

برنامه‌نویسی پویا، شامل دو نسخه متفاوت از چگونگی پیاده‌سازی است: **تکرار خط‌مشی** و **تکرار مقدار**. در ادامه به‌طور خلاصه این دو رویکرد را شرح خواهیم داد.

تکرار خط‌مشی

هنگامی که یک خط‌مشی π با استفاده از v_π بهبود یافته تا یک خط‌مشی بهتر $\acute{\pi}$ داشته باشد، می‌توان $v_{\acute{\pi}}$ را محاسبه کرده و دوباره آن را بهبود بخشید تا خط‌مشی $\pi^{"}$ حتی بهتر بدست آورد. در نتیجه می‌توان به یک ترتیب به طور یکنواخت خط‌مشی‌ها و توابع مقدار را بدست آورد:

$$\pi_0 \xrightarrow{\text{E}} v_{\pi_0} \xrightarrow{\text{I}} \pi_1 \xrightarrow{\text{E}} v_{\pi_1} \xrightarrow{\text{I}} \pi_2 \xrightarrow{\text{E}} \cdots \xrightarrow{\text{I}} \pi_* \xrightarrow{\text{E}} v_*,$$

تضمین شده است که هر خط‌مشی نسبت به خط‌مشی قبلی اکیدا بهبود می‌یابد؛ مگر این‌که از قبل بهینه باشد. از آن‌جا که فرآیند تصمیم‌گیری مارکوف محدود، تنها تعداد محدودی از خط‌مشی‌ها را دارد، این فرآیند باید در یک تعداد تکرار محدود، به یک خط‌مشی و تابع مقدار بهینه همگرا باشد.

این روش برای یافتن یک خط‌مشی بهینه، تکرار خط‌مشی نامیده می‌شود. باید توجه داشت که هر ارزیابی خط‌مشی، که خود یک محاسبه تکراری است، با تابع مقدار برای خط‌مشی قبلی شروع می‌شود. این امر، معمولا منجر به افزایش بسیار زیاد سرعت همگرایی ارزیابی سیاست می‌شود؛ احتمالا به این دلیل که تابع مقدار از یک خط‌مشی به خط‌مشی دیگر تغییر کمی می‌کند.

تکرار مقدار

یکی از اشکالاتی که در روش تکرار خط‌مشی وجود دارد، این است که هر یک از تکرارهای آن شامل ارزیابی خط‌مشی است، که این امر ممکن است خود یک محاسبه تکراری طولانی مدت را به‌همراه داشته باشد که مستلزم جابجایی‌های متعدد در مجموعه وضعیت‌ها است. اگر ارزیابی خط‌مشی به‌صورت تکراری انجام شود، همگرایی دقیقا به v_π فقط در محدوده مجاز اتفاق می‌افتد. در نتیجه، سوالی بوجود می‌آید، آیا ما باید منتظر همگرایی دقیق باشیم، یا می‌توانیم از آن دست بکشیم؟

مرحله ارزیابی خط‌مشی، تکرار خط‌مشی می‌تواند به چندین روش بدون از دست دادن تضمین همگرایی تکرار خط‌مشی کوتاه شود. یک مورد خاص مهم هنگامی است که ارزیابی خط‌مشی فقط پس از یک بار حرکت متوقف شود. این الگوریتم تکرار مقدار نامیده شده که می‌تواند به عنوان یک عملیات پشتیبان‌گیری ساده نوشته شود که ترکیبی از بهبود خط‌مشی و مراحل ارزیابی خط‌مشی است:

$$v_{k+1}(s) = max_a \mathbb{E}[R_{t+1} + \gamma v_k(S_{t+1})|S_t = s, A_t = a]$$
$$= max_a \sum_{\acute{s},r} p(\acute{s}, r|s, a)[r + \gamma v_k(\acute{s})]$$

برای هر v_0 دلخواه، می‌توان نشان داد که توالی $\{v_k\}$ می‌تواند تحت همان شرایطی که وجود v^* را تضمین می‌کند به v^* همگرا می‌شود.

در نهایت، بیایید در نظر بگیریم که تکرار مقدار چگونه به پایان می‌رسد. همانند ارزیابی خط‌مشی، تکرار مقدار رسما به تعداد نامحدودی از تکرارها نیاز دارد تا دقیقا در عمل همگرا شود، ما زمانی متوقف می‌شویم که تابع مقدار تنها با مقدار کمی در یک جابه‌جایی تغییر می‌کند.

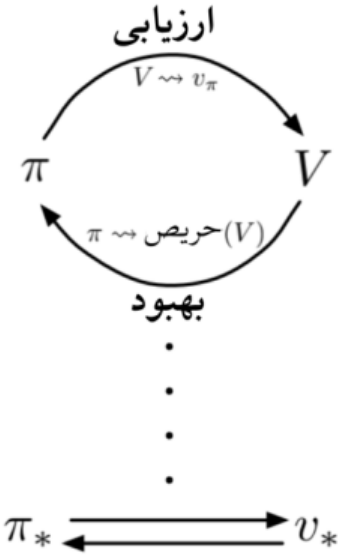
تکرار خط‌مشی تعمیم‌یافته[1]

تکرار خط مشی شامل دو فرآیند همزمان و متقابل است که یکی تابع مقدار را با خط‌مشی فعلی (ارزیابی خط‌مشی) سازگار می‌کند، و دیگری خط‌مشی را با توجه به تابع مقدار

[1] generalized policy iteration (GPI)

فعلی (بهبود سیاست) حریص می‌کند. در تکرار خط مشی، این دو فرآیند متناوب هستند و هر یک قبل از شروع روند دیگر تکمیل می شوند، اما این واقعا لازم نیست. به عنوان مثال، در تکرار مقدار، تنها یک تکرار ارزیابی خط‌مشی بین هر بهبود خط‌مشی انجام می‌شود.

از اصطلاح تکرار خط‌مشی تعمیم‌یافته برای اشاره به ایده کلی اجازه دادن به تعامل فرآیندهای ارزیابی خط‌مشی و بهبود خط‌مشی، مستقل از جزئیات دو فرآیند استفاده می‌شود. طرح کلی برای تکرار خط‌مشی تعمیم‌یافته در شکل ٤ ــ ٤ نشان داده شده است.



شکل ٤ ــ ٤ تکرار خط‌مشی تعمیم‌یافته. توابع مقدار و خط‌مشی تا زمانی که بهینه و در نتیجه با هم سازگار باشند تعامل دارند.

به راحتی می توان دریافت که اگر هم فرآیند ارزیابی و هم فرآیند بهبود پایدار شوند، یعنی دیگر تغییری ایجاد نکنند، آنگاه تابع مقدار و خط‌مشی باید بهینه باشد. تابع مقدار تنها زمانی پایدار می‌شود که با خط‌مشی فعلی سازگار باشد، و خط‌مشی تنها زمانی پایدار می‌شود که با توجه به تابع مقدار فعلی، حریص باشد. بنابراین، هر دو فرآیند فقط وقتی ثبات پیدا می‌کنند که خط‌مشی پیدا شود که نسبت به تابع ارزیابی خود حریصانه عمل کند. این نشان می‌دهد که معادله بهینه‌سازی بلمن برقرار است و بنابراین خط‌مشی و تابع مقدار بهینه هستند.

فرایندهای ارزیابی و بهبود در تکرار خط‌مشی تعمیم‌یافته را می‌توان هم به عنوان یک رقابت و هم همکاری در نظر گرفت. در درازمدت، این دو فرآیند برای یافتن یک راه حل مشترک تعامل دارند: تابع مقدار بهینه و خط‌مشی بهینه.

۲.۴.۴ مونت کارلو

برخلاف برنامه‌نویسی پویا، روش‌های مونت‌کارلو تنها از تجربه می‌آموزند. از بسیاری جهات می‌توان آن را ساده‌ترین روش یادگیری تقویتی دانست. یک روش مونت‌کارلو مقدار یک وضعیت را با گذراندن چندین مرتبه و میانگین کل پاداش دریافتی پس از تصویب وضعیت، تعیین می‌کند. از آنجا که کل پاداش دریافتی قبل از سپری‌شدن مرحله ناشناخته است، روش‌های مونت‌کارلو فقط برای کارهای مرحله‌ای قابل اجرا هستند. بروزرسانی تنها پس از پایان حرکت در فضای وضعیت که بروزرسانی خارج‌از خط نامیده می‌شود (بروزرسانی در حالی‌که هنوز در فضای وضعیت حرکت می‌کنید، بروزرسانی برخط نامیده می‌شود)، صورت می‌گیرد. علاوه براین، در روش مونت‌کارلو، مقادیر براساس تجربه واقعی است نه بر اساس مقادیر وضعیت‌های جانشین.

روش‌های مونت کارلو بر مبنای ایده تکرار خط‌مشی تعمیم‌یافته کار می‌کنند. همان‌طور که پیش‌تر بیان شد، تکرار خط‌مشی تعمیم‌یافته یک طرح تکراری است و از دو مرحله تشکیل شده است. در اولین گام، سعی می‌شود تقریب تابع مقدار را براساس خط‌مشی فعلی ایجاد کند که به عنوان مرحله ارزیابی خط‌مشی شناخته می‌شود. در گام دوم، خط‌مشی با توجه به تابع مقدار فعلی که به عنوان مرحله بهبود خط‌مشی شناخته می‌شود، بهبود می‌یابد. در روش‌های مونت‌کارلو، برای تخمین تابع مقدار، اجرای برنامه‌ها با اجرای خط‌مشی فعلی برروی سیستم انجام می‌شود. پاداش تجمعی در کل مرحله و توزیع حالاتی که با آن روبرو می‌شوند برای تشکیل تخمینی از تابع مقدار استفاده می‌شود. سپس، خط‌مشی فعلی به‌صورت حریصانه، با توجه به تابع مقدار فعلی تخمین‌زده می‌شود. با استفاده از این دو مرحله به صورت تکراری، می‌توان نشان داد که الگوریتم به تابع مقدار و خط‌مشی بهینه همگرا می‌شود. اگرچه اجرای روش‌های مونت‌کارلو ساده است، اما برای همگرایی آن‌ها به تعداد تکرار زیادی نیاز دارد و از واریانس بالا در تخمین تابع مقدار رنج می‌برد.

۳.۴.۴ یادگیری تفاوت‌زمانی

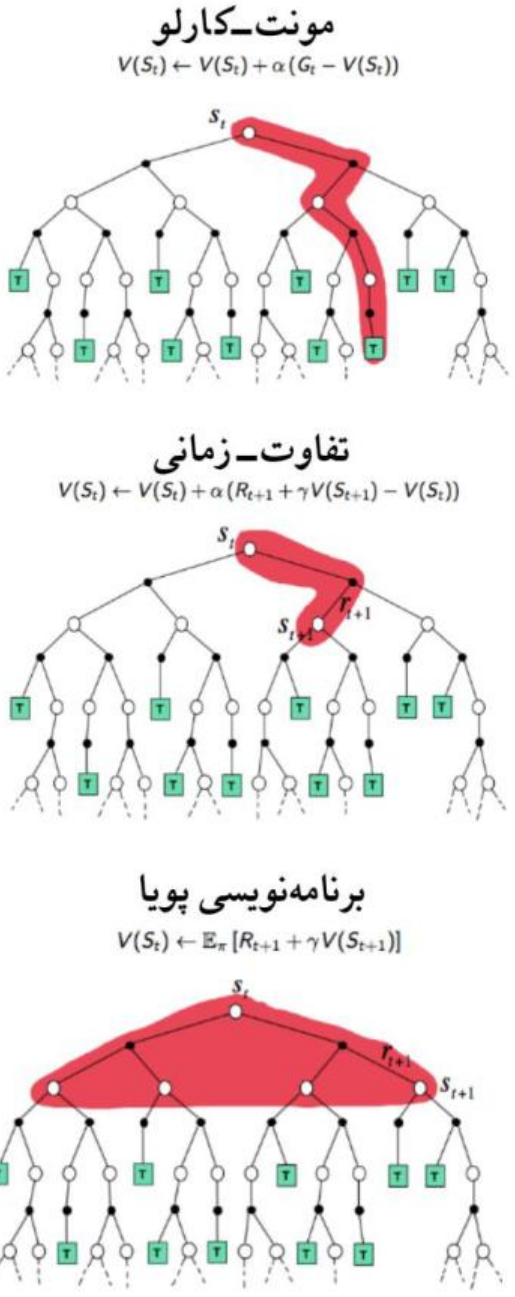
یادگیری تفاوت‌زمانی، ایده‌های برنامه‌نویسی پویا و مونت‌کارلو را در خود جای می‌دهد. رویکرد تفاوت‌زمانی با مقایسه تخمین‌ها در دو نقطه از زمان، مقدار جفت وضعیت‌ـ عمل را تقریب می زند، از همین‌رو نام تفاوت‌زمانی را به خود گرفته است. همانند برنامه‌نویسی پویا، الگوریتم‌های یادگیری تفاوت‌زمانی، برآورد مقادیر را براساس سایر تخمین‌ها یاد می‌گیرد (که به آن بوت‌استرپینگ[۱] گویند). همچنین، یادگیری تفاوت زمانی همانند روش مونت‌کارلو، می‌تواند بدون اطلاع قبلی از محیط و به‌طور مستقیم از تجربیات بیآموزد. این بدان معناست که یادگیری تفاوت‌زمانی یک رویکرد یادگیری بدون مدل (یا می‌توان آن را معادل با یادگیری بدون‌نظارت دانست) است.

ایده اصلی در رویکرد یادگیری تفاوت‌زمانی، یادگیری براساس تفاوت بین پیش‌بینی‌های پی‌درپی زمانی است و برای بروزرسانی نیازی به صبرکردن تا پایان مسیر نیست. به عبارت دیگر، هدف از یادگیری این است که پیش‌بینی کنونی یادگیرنده برای الگوی فعلی ورودی، بیشتر با پیش‌بینی بعدی در مرحله بعدی مطابقت داشته باشد. بینش این روش بر این است که، پس از مشاهده برخی از پاداش‌هایی که عامل پس از بازدید یک وضعیت و انجام یک عمل معین بدست آورده، می‌توان تخمین بهتری برای مقدار یک جفت وضعیت‌ـ عمل ارائه کرد.

روش‌های مونت‌کارلو و برنامه‌نویسی پویا دارای نقاط ضعف جدی هستند؛ مونت‌کارلو قادر به حل مسائل پیوسته نیست و برنامه‌نویسی پویا به یک مدل از محیط نیاز دارد. یادگیری تفاوت‌زمانی، از نسخه پشتیبان‌گیری تهیه نمونه در روش مونت‌کارلو و بوت‌استرپینگ از روش برنامه‌نویسی پویا همزمان استفاده می‌کند. از همین‌رو، تنها قسمت‌های خوب روش‌های قبلی را با یک‌دیگر ترکیب می‌کند.

روش مونت‌کارلو، از پاداش کل برای بروزرسانی تابع مقدار استفاده می‌کند. در حالی‌که، روش تفاوت‌زمانی، به‌جای بروزرسانی مقادیر در پایان مرحله (با استفاده از مجموع پاداش)، مقادیر را به‌صورت برخط در هر مرحله بروز می‌کند.

[۱] bootstrapping



شکل ٤ ـ ٥ مقایسه نمودارهای پشتیبان‌گیری مونت‌کارلو ، یادگیری تفاوت‌زمانی و برنامه‌نویسی پویا برای توابع وضعیت مقدار

برای مقایسه سه رویکرد، مونت‌کارلو، برنامه‌نویسی پویا و یادگیری تفاوت‌زمانی می‌توان از نمودار پشتیبان‌گیری استفاده کرد. مقایسه بین این سه رویکرد براساس نمودار پشتیبان‌گیری وضعیت ـ مقدار در شکل ٤ ـ ٥ قابل مشاهده است.

یادگیری تفاوت زمانی اغلب به یک مساله پیش‌بینی با یک قانون بروزرسانی برای تابع مقدار داده شده، اشاره دارد:

$$V(s) \leftarrow V(s) + \alpha\big(r + \gamma V(\acute{s}) - V(s)\big)$$

جایی که α نرخ یادگیری و γ فاکتور تنزیل است. قسمت داخل پرانتز به عنوان خطای تفاوت‌زمانی شناخته می شود:

$$\delta_t = r + \gamma V(\acute{s}) - V(s)$$

روش یادگیری تفاوت‌زمانی برای پیش‌بینی در دو روش مختلف برای انجام بازبینی استفاده می‌شود. تفاوت اصلی این دو روش این است که یکی در خط‌مشی و دیگری خارج از خط‌مشی است. این دو الگوریتم که به‌طور گسترده در یادگیری تقویتی استفاده می‌شود، سارسا و کیو ـ یادگیری می‌باشد. در ادامه به بررسی این دو نوع از الگوریتم تفاوت‌زمانی می‌پردازیم.

سارسا (در خط‌مشی)

سارسا یک روش یادگیری تقویتی تفاوت‌زمانی است که در تلاش برای تخمین یک تابع مقدار عمل را به‌جای یک تابع مقدار یاد بگیرد. این وراثت نام از: ("وضعیت"، "عمل"، "پاداش"، "وضعیت بعدی"، "عمل بعدی") گرفته شده است. این رویکرد، در خط‌مشی است، چرا که $Q_\pi(s,a)$ را برای π برای خط‌مشی فعلی تخمین می‌زند. قانون بروزرسانی وضعیت ـ مقدار به‌صورت زیر می‌باشد:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma Q(\acute{s},\acute{a}) - Q(s,a)\big)$$

می‌توان نشان داد که سارسا هنگامی که همه جفت‌های وضعیت ـ عمل به تعداد نامحدودی بازدید شود، به یک خط‌مشی بهینه همگرا می‌شود.

کیو_یادگیری (خارج‌از خط‌مشی)

یادگیری تفاوت‌زمانی خارج‌از خط‌مشی کیو_یادگیری (Q-learning) نامیده می‌شود که یکی از اساسی‌ترین و محبوب‌ترین روش‌ها برای تخمین توابع مقدار Q به روشی بدون مدل است. قانون بروزرسانی این روش به صورت زیر است:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\big(r + \gamma max_{\acute{a}} Q(\acute{s},\acute{a}) - Q(s,a)\big)$$

کیو_یادگیری در تلاش است با توجه به شرایط فعلی، بهترین اقدامات را انجام دهد. این الگوریتم خارج‌از خط‌مشی در نظر گرفته می‌شود. چراکه، تابع کیو_یادگیری از اقداماتی خارج‌از خط‌مشی فعلی یاد می‌گیرد. به‌طور کلی می‌توان گفت که، کیو_یادگیری به‌دنبال یادگیری خط‌مشی است تا مجموع پاداش را بیشینه کند. امروزه، بسیاری از الگوریتم‌های یادگیری تقویی عمیق مبتنی‌بر کیو_یادگیری هستند.

> کیو_یادگیری بی‌تفاوت به اکتشاف است. این بدان معناست که، بدون توجه به خط‌مشی اکتشافی که دنبال می‌شود، به خط‌مشی مطلوب همگرا می‌شود؛ با این فرض که هر جفت اقدام وضعیت به تعداد نامحدودی بازدید شده و پارامتر یادگیری α به‌طور مناسب کاهش می‌یابد.

۴.۴.۴ جست‌وجوی خط‌مشی

روش‌های جست‌وجوی خط‌مشی نیازی به حفظ یک مدل تابع مقدار ندارند، بلکه به‌طور مستقیم یک خط‌مشی بهینه π^* را جست‌وجو می‌کنند. از بین الگوریتم‌های جست‌وجوی خط‌مشی در یادگیری تقویتی، الگوریتم گرادیان خط‌مشی محبوب‌ترین هستند.

هدف از یادگیری تقویتی، یافتن یک استراتژی رفتار بهینه برای عامل برای به‌دست آوردن پاداش‌های بهینه است. روش‌های گرادیان خط‌مشی به طور مستقیم به مدل‌سازی و بهینه‌سازی خط‌مشی می‌پردازند. این خط مشی با انجام اقدامات ارائه شده از خط‌مشی فعلی و محاسبه پاداش ارزیابی می‌شود. سپس پارامترهای خط‌مشی در جهت افزایش بازده مورد انتظار با استفاده از گرادیان نزولی بروز می‌شوند. قانون بروزرسانی برای پارامترهای خط‌مشی را می توان با توجه به بازده مورد انتظار J به‌صورت زیر نوشت:

$$\theta^{\pi}_{i+1} = \theta^{\pi}_{i} + \alpha \nabla_{\theta^{\pi}} j \ , j = \mathbb{E}_{\pi}(\sum_{k=0}^{\infty} \gamma^{k} r_{k})$$

جست‌وجوی خط‌مشی، همگرایی بهتری دارد و می‌تواند خط‌مشی‌های تصادفی را بیاموزد که با روش‌های مبتنی بر مقدار امکان‌پذیر نیست. اشکال عمده الگوریتم‌های خط‌مشی، مرحله ارزیابی خط‌مشی آن‌ها است که از واریانس زیادی رنج می‌برد و بنابراین می‌تواند در یادگیری خط‌مشی‌های خوب، کند باشد.

۵.۴.۴ نقاد عامل

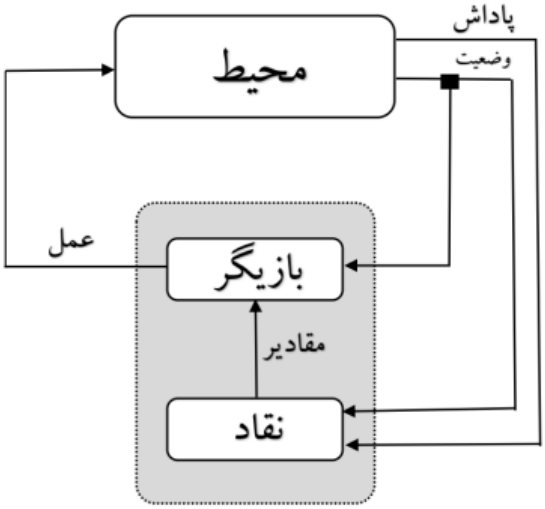
روش‌های گرادیان خط‌مشی محض، به دلیل تخمین‌های با واریانس بالا، به آهستگی یاد می‌گیرند و برای اجرای مسائل برخط نامناسب هستند. با این حال، می‌توان از روش‌های تفاوت‌زمانی برای مقابله با این مشکلات استفاده کرد.

رویکردهای تابع مقدار از لحاظ نظری، به کل پوشش فضای وضعیت و مقادیر تقویت‌شده مربوطه از همه اعمال ممکن در هر وضعیت نیاز دارند. بنابراین، هنگام کار با برنامه‌های کاربردی با ابعاد بالا، پیچیدگی محاسباتی زیادی داشته و همچنین، یک تغییر کوچک در مقادیر تقویت‌کننده محلی ممکن است باعث تغییر زیادی در خط‌مشی شود.

در مقابل روش‌های تابع مقدار، روش‌های جست‌وجوی خط‌مشی، خط‌مشی فعلی و خط‌مشی بعدی را به خط‌مشی فعلی در نظر می‌گیرند و سپس تغییرات پارامترهای خط‌مشی را محاسبه می‌کنند که در نتیجه، پیچیدگی محاسباتی بسیار کم‌تری در مقابل روش‌های تابع مقدار است. با این حال، رویکردهای جستجوی خط‌مشی ممکن است سبب بهینه‌سازی محلی شود و نتواند به بهینه سراسری برسد.

روش‌های نقاد عامل با ترکیب این دو رویکرد به‌صورت همزمان یک خط‌مشی و یک تابع وضعیت ـ مقدار را یاد می‌گیرند. این رویکرد که از تلفیق تابع مقدار و رویکرد جست‌وجوی خط‌مشی برای بهره‌وری از مزیت هر دو روش ارائه شده است، با ساختار نقاد عامل شناخته می‌شود. می‌توان گفت، روش نقاد عامل، روش‌های یادگیری

تفاوت‌زمانی است که به‌طور صریح خط‌مشی را ذخیره می‌کند. همان‌طور که در شکل ٤ـ٦ نشان داده شده است، انتخاب اقدام توسط عامل کنترل و از نقاد برای انتقال مقادیر به عامل استفاده می‌شود. بنابراین، تصمیم‌گیری در مورد به روزرسانی خط‌مشی براساس این نقاد می‌باشد.



شکل ٤ـ٦ ساختار نقاد عامل

۶.۴.۴ رویکرد ترکیبی (دینا-کیو)

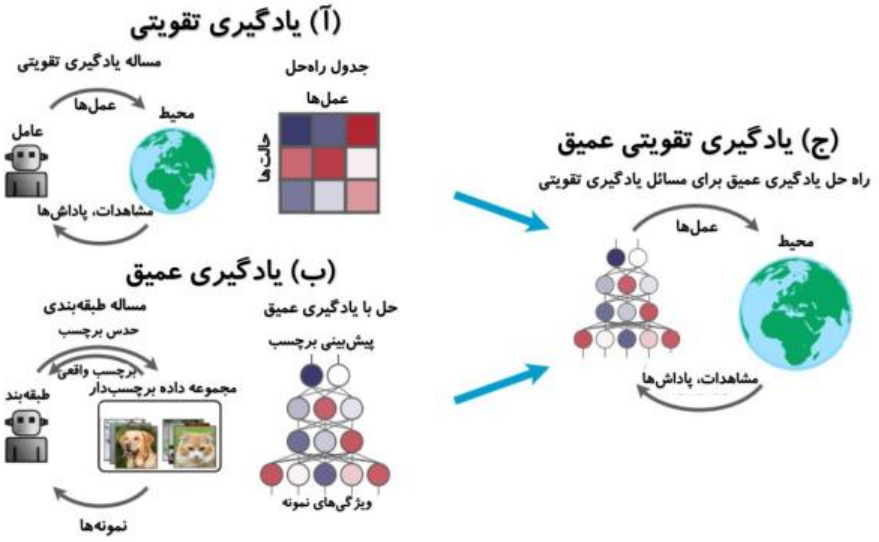
داینا_کیو، یکی از معماری‌هایی است که توانایی تلفیق قابلیت یادگیری در روش بدون مدل و قابلیت برنامه‌ریزی در روش‌های مبتنی‌بر را دارد. الگوریتم با دیدن این‌که کدام وضعیت‌ها بازدید می‌شود و چه پاداشی در یک وضعیت خاص می‌گیرد، یک مدل را یاد گرفته و از اطلاعات برای بروزرسانی احتمالات انتقال و تابع پاداش استفاده می‌کند.

۵.۴ یادگیری تقویتی عمیق

هنگام برخورد با مسائل دامنه با ابعاد بالا یا عامل‌های پیوسته، یادگیری تقویتی از مشکل نمایش ناکارآمد ویژگی رنج می‌برد. بنابراین، زمان یادگیری کندی را به‌همراه دارد و باید تکنیک‌هایی برای سرعت بخشیدن به روند یادگیری طراحی شود. از همین‌رو، زمینه

جدیدی با عنوان یادگیری تقویتی عمیق برای کمک به حل یادگیری تقویتی در مسائل با ابعاد بالا ظاهر شد. مهم‌ترین ویژگی یادگیری عمیق این است که شبکه‌های عصبی عمیق می‌توانند به صورت خودکار نمایش‌های کم حجم و فشرده از داده‌های با ابعاد بالا را پیدا کنند. یادگیری تقویتی عمیق، با ترکیب مزایای یادگیری عمیق و یادگیری تقویتی در جهت ساخت سیستم‌های هوش مصنوعی تلاش می‌کند.

یادگیری تقویتی عمیق از قدرت بازنمایی یادگیری عمیق برای مقابله با مشکلاتی که در یادگیری تقویتی وجود دارد، بهره می‌برد. یک سیستم یادگیری تقویتی عمیق را می‌توان، به عنوان سیستمی که حداکثر پاداش طولانی مدت را در یک مساله یادگیری تقویتی با استفاده از بازنمایی‌هایی که خود آن‌ها توسط شبکه‌ی عمیق (به‌جای اینکه توسط طراح تعریف شده باشند) آموخته می‌شود، تعریف کرد. در شکل ٤ ـ ۷ نحوه کار یادگیری تقویتی عمیق و مقایسه آن‌ها قابل مشاهده است. همان‌طور که در تصویر مشاهده می‌شود، در یادگیری تقویتی، از شبکه‌های عصبی به عنوان عامل برای حل مساله یادگیری تقویتی استفاده می‌شود.



شکل ٤ ـ ۷ یادگیری تقویتی، یادگیری عمیق و یادگیری تقویتی عمیق

در چندین سال گذشته، یادگیری تقویتی عمیق مورد توجه بسیاری در جامعه هوش مصنوعی قرار گرفته است. یادگیری تقویتی عمیق به استفاده از شبکه‌های عصبی عمیق

به عنوان تقریب توابع در تابع مقدار یا خط‌مشی در یک چارچوب یادگیری تقویتی اشاره دارد و در روش‌های گرادیان خط‌مشی، کیو_یادگیری و نقاد عامل با موفقیت اعمال شده است. در ادامه این بخش، به تشریح الگوریتم‌های مدرن یادگیری تقویتی عمیق مبتنی‌بر کیو_یادگیری می‌پردازیم.

۱.۵.۴ شبکه کیو عمیق (DQN)

کیو_یادگیری یک الگوریتم ساده و در عین حال کاملا قدرتمند برای ایجاد صفحه فهرست برای عامل است. این کار به عامل کمک می‌کند تا دقیقا مشخص کند که کدام عمل باید انجام شود، اما اگر این صفحه فهرست خیلی طولانی باشد چه می‌شود؟ محیطی را تصور کنید که دارای ۱۰ هزار وضعیت و ۱۰۰۰ عمل در هر وضعیت است. با این کار جدولی از ۱۰ میلیون سلول ایجاد می شود و همه چیز به سرعت از کنترل خارج می‌شود! واضح است که نمی‌توانیم مقدار Q وضعیت‌های جدید را از وضعیت‌های قبلا بررسی شده استنباط کنیم. این امر دو مشکل را نشان می دهد: اول این‌که، مقدار حافظه مورد نیاز برای ذخیره و بروزرسانی جدول با افزایش تعداد وضعیت‌ها افزایش می‌یابد. دوم، مقدار زمان لازم برای کاوش در هر وضعیت برای ایجاد جدول Q مورد نیاز، عملی نیست. اینجاست که یک تفکر بوجود می‌آید: اگر این مقادیر Q را با مدل‌های شبکه‌های عصبی تخمین بزنیم، چه می‌شود؟ خوب، این ایده پشت الگوریتم $DeepMind$ بود که منجر به خرید آن توسط گوگل به مبلغ ۵۰۰ میلیون دلار شد!

شبکه کیو عمیق یا به اختصار DQN یک الگوریتم مبتنی‌بر مقدار است که از یک شبکه عصبی $Q(s,a|\theta)$ برای تقریب بهینه عمل_مقدار Q^* برای هر عمل در یک وضعیت معین استفاده می‌کند (جدول کیو_یادگیری را با یک شبکه عصبی جایگزین خواهیم کرد که سعی می‌کند مقادیر کیو را تقریب بزند). وضعیت به عنوان ورودی داده می‌شود و مقدار Q تمام اقدامات احتمالی به عنوان خروجی تولید می‌شود. اهداف آموزشی برای Q از طریق معادله بلمن محاسبه می‌شود:

$$Q(s,a,\theta) = r + \gamma max_{\acute{a}} Q(\acute{s},\acute{a},\acute{\theta})$$

برای آموزش شبکه عصبی، نیاز به یک تابع زیان داریم که در شبکه کیو عمیق از خطای میانگین مربع در معادله بلمن استفاده می‌کنیم:

$$C(\theta\,|\theta') = \frac{1}{m}\sum_j \Big(\underbrace{r_j + \gamma\, Q(s_{j+1}, \pi(s_{j+1}|\theta')\,|\theta')}_{y_j} - Q(s_j, \pi(s_j|\theta)\,|\theta)\Big)^2$$
$$= \frac{1}{m}\sum_j \Big(\underbrace{r_j + \gamma \max_a Q(s_{j+1}, a\,|\theta')}_{y_j} - \max_a Q(s_j, a\,|\theta')\Big)^2.$$

با این حال، وابستگی اهداف Q به خودش می‌تواند منجر به بی‌ثباتی یا حتی واگرایی در هنگام یادگیری گردد. داشتن مجموعه دوم پارامترهای شبکه $\theta' = LP(\theta)$، جایی که LP یک فیلتر کم‌گذر است (به عنوان مثال، میانگین متحرک نمایی)، یادگیری را تثبیت می‌کند.

بی‌ثباتی‌های اضافی می‌توانند ناشی از آموزش مستقیم در مورد وضعیت‌های دریافتی و پاداش‌ها باشند. چرا که، برخلاف یادگیری بانظارت، داده‌های ورودی (جفت وضعیت–عمل) بسیار همبسته هستند و بخشی از یک مسیر هستند. نتیجه این اتفاق، منجر به بیش‌برازش در شبکه شده و شبکه قادر به یادگیری موثر نخواهد بود. علاوه براین، خط‌مشی و در نتیجه توزیع داده‌ها ممکن است به سرعت با تکامل Q تغییر کند. DQN با ذخیره تمام انتقال‌ها (s_t, a_t, r_t, s_{t+1}) در یک مجموعه داده حافظه و سپس یادگیری در ریزدسته‌ها که متشکل از انتقال‌های تصادفی از این مجموعه است، هر دو مشکل را حل می‌کند. به عبارت دیگر، در هر لحظه، تعداد مشخصی از نمونه‌ها به طور تصادفی از حافظه استخراج و برای آموزش شبکه استفاده می‌شوند، که منجر به یادگیری بهتر شبک می‌شود. این ترفند همبستگی داده‌های ورودی را می‌شکند و تغییرات در توزیع ورودی را هموار می‌کند.

تنها تفاوت بین کیو–یادگیری و شبکه کیو عمیق مغز عامل است. مغز عامل در کیو–یادگیری جدول کیو است، اما در شبکه کیو عمیق مغز عامل یک شبکه عصبی عمیق است. ورودی شبکه عصبی وضعیت‌ها خواهد بود و تعداد نورون‌های خروجی تعداد اعمالی است که یک عامل می‌تواند انجام دهد.

۲.۵.۴ شبکه کیو عمیق دوگانه[۱]

یکی از مشکلات الگوریتم DQN این است که پاداش واقعی را بیش‌از حد ارزیابی می‌کند. مقادیر Q فکر می‌کنند عامل قصد دارد بازدهی بالاتر از آنچه در واقعیت بدست خواهد

[۱] Double

آمد را بدست آورد. شبکه کیو عمیق دوگانه، یک افزایش شبکه کیو عمیق برای کاهش بیش‌از حد ارزیابی با ترفند ساده‌ی، جداسازی انتخاب عمل از ارزیابی عمل است. در این نوع شبکه، معادله بلمن در DQN را به‌صورت زیر تغییر می‌دهند:

$$Q(s,a,\theta) = rQ(\acute{s}, argmax_{\acute{a}} Q(\acute{s},\acute{a},\theta);\acute{\theta})$$

ابتدا، شبکه عصبی اصلی θ تصمیم می‌گیرد که کدام یک از بهترین اقدامات بعدی $\acute{a}$ در میان تمام اعمال بعدی موجود است و سپس شبکه عصبی هدف این عمل را ارزیابی می‌کند تا مقدار Q خود را بداند. این ترفند ساده نشان داده است که ارزیابی‌های بیش‌از حد را کاهش می‌دهد، که نتیجه آن خط‌مشی‌های نهایی بهتر است.

شبکه کیو عمیق دوگانه از دو مدل شبکه عصبی یکسان استفاده می‌کند. یکی دقیقا همانند شبکه کیو عمیق در طول بازپخش تجربه یاد می‌گیرد و دیگری رونوشتی از آخرین قسمت از مدل اول است. مقدار Q در واقع با این مدل دوم محاسبه می‌شود، چرا؟ در DQN مقدار Q با پاداش اضافه شده به حداکثر مقدار Q حالت بعدی محاسبه می‌شود. بدیهی است، اگر هر بار که مقدار Q برای یک وضعیت خاص عدد بالایی را محاسبه کند، مقداری که از خروجی شبکه عصبی برای آن وضعیت خاص بدست می‌آید، هر بار بیشتر خواهد شد. هر مقدار نورون خروجی بیشتر و بیشتر می‌شود تا زمانی که اختلاف بین هر مقدار خروجی زیاد باشد. حال اگر فرض کنیم برای وضعیت s عمل a مقدار بالاتری نسبت به عمل b دارد، از همین‌رو عمل a هر بار برای حالت s انتخاب می‌شود. سپس، از آنجا که شبکه عصبی به روشی آموزش داده می‌شود که مقدار بسیار بالاتری را برای عمل ارائه می‌دهد، آموزش شبکه برای یادگیری این که عمل b عمل بهتری در برخی شرایط است، دشوار است.

بنابراین، برای پایین آوردن اختلاف بین مقادیر خروجی (اعمال)، از یک مدل ثانویه استفاده می‌شود که رونوشت مدل اصلی از قسمت آخر باشد. بدیهی است، از آنجا که تفاوت بین مقادیر مدل دوم کمتر از مدل اصلی است، برای دستیابی به مقدار Q از مدل دوم استفاده می‌کنیم.

۳.۵.۴ شبکه کیو عمیق هم‌آورد

برای برخی از وضعیت‌ها، اعمال مختلف مربوط به مقدار مورد انتظار نیستند و نیازی به یادگیری اثر هر عمل برای چنین وضعیت‌هایی نداریم. به عنوان مثال، تصور کنید که روی کوه بایستید و طلوع خورشید را تماشا کنید. چشم‌انداز دلپذیر، حس راحتی را برای شما به‌وجود می‌آورد و پاداش بالایی را فراهم می‌کند. می‌توانید در اینجا بمانید و مقادیر Q اعمال مختلف مهم نیست. بنابراین، جدا کردن مقدار مستقل عمل از وضعیت و مقدار Q ممکن است منجر به یادگیری قوی‌تر شود. شبکه کیو عمیق هم‌آورد، برای دستیابی به این ایده معماری شبکه‌ی جدیدی را پیشنهاد می‌کند. به‌طور دقیق‌تر، مقدار Q را می‌توان به بخش مقدار وضعیت و تابع سود تقسیم کرد:

$$Q(s,a) = V(s) + A(s.a)$$

تابع مقدار $V(s)$ به ما می‌گوید که چه مقدار پاداش از حالت s بدست خواهیم آورد و تابع مزیت $A(s.a)$ می‌گوید که یک عمل در مقایسه با اعمال دیگر چقدر بهتر است. بر این اساس، مقدار یک وضعیت مستقل از عمل است. اما مزیت آن چیست؟ عامل ممکن است در وضعیتی باشد که هر یک از اعمال مقدار Q یکسانی داشته باشد. بنابراین، عمل خوبی در این وضعیت وجود ندارد. چه اتفاقی می‌افتد اگر مقدار Q را به مقدار یک وضعیت و سود هر عمل تقسیم کنیم. اگر هر عملی نتیجه یکسانی داشته باشد، سود هر عمل نیز همان مقدار را خواهد داشت. حال، اگر میانگین همه سودها را از هر سود کم کنیم، صفر (یا نزدیک به صفر) بدست خواهیم آورد و مقدار Q واقعا همان مقداری است که وضعیت دارد:

$$Q(s,a) = V(s) + A(s.a) - \frac{1}{n}\sum_{\acute{a}} A(s.\acute{a})$$

البته باید توجه داشت که خروجی مدل، مقدار وضعیت به‌علاوه سود اعمال خواهد بود. اما، برای آموزش مدل از مقدار Q برای اهداف مشابه استفاده می‌کنیم:

$$Q(s_t, a_t) = R_t + \gamma max Q(s_{t+1}, \acute{a})$$

خلاصه فصل

- اگرچه یادگیری بانظارت یکی از انواع مهم یادگیری به‌حساب می‌آید، اما به تنهایی برای یادگیری تعاملی کافی نیست.
- رویکرد یادگیری تقویتی، بیش از سایر رویکردهای یادگیری ماشین متمرکز بر یادگیر هدفمند از طریق تعامل است.
- یادگیری تقویتی از طریق عامل در تلاش است با آزمون و خطا مساله را از طریق تعامل با محیطی که برای عامل ناشناخته می‌باشد، حل کند.
- یادگیری تقویتی، معمولا به عنوان یک فرآیند تصمیم‌گیری مارکوف مدل‌سازی می‌شود.
- عامل یکی از مهم‌ترین عناصر تشکیل‌دهنده یک سیستم مبتنی‌بر یادگیری تقویتی است. چراکه، شامل هوش در جهت تصمیم‌گیری و توصیه اقدامات بهینه در هر شرایط است.
- هدف اصلی عامل در یادگیری تقویتی، جمع‌آوری بیشترین میزان پاداش در "درازمدت" است. در جهت انجام این کار، عامل باید یک خط‌مشی بهینه برای رفتار در محیط پیدا کند.
- خط‌مشی، هسته اصلی عامل یادگیری تقویتی است. چراکه، به تنهایی برای تعیین رفتار کافی است.
- یکی از ویژگی‌های کلیدی یادگیری تقویتی، معضل استخراج در مقابل اکتشاف است. این معضل از آنجایی ناشی میشود که، فرآیند یادگیری در یادگیری تقویتی به‌صورت برخط صورت می‌گیرد.
- راه تمایز بین الگوریتم‌های مبتنی‌بر مدل یا بدون مدل این است که الگوریتم‌ها را بررسی‌کنید و ببینید آیا از توابع انتقال و پاداش استفاده می‌کنند یا نه.

- الگوریتم های کلاسیک برنامه‌نویسی پویا، به دلیل فرض داشتن یک مدل کامل و همچنین به دلیل هزینه محاسباتی زیاد، کاربرد محدودی در یادگیری تقویتی دارند.
- برخلاف برنامه‌نویسی پویا، روش‌های مونت‌کارلو تنها از تجربه می‌آموزند. از بسیاری جهات می‌توان آن را ساده‌ترین روش یادگیری تقویتی دانست.
- روش‌های مونت کارلو برمبنای ایده تکرار خط‌مشی تعمیم‌یافته کار می‌کنند.
- روش‌های مونت‌کارلو تنها برای کارهای مرحله‌ای قابل اجرا هستند.
- یادگیری تفاوت‌زمانی، ایده‌های برنامه‌نویسی پویا و مونت‌کارلو را در خود جای می‌دهد.
- یادگیری تفاوت‌زمانی خارج‌از خط‌مشی کیو ـ یادگیری نامیده می‌شود که یکی از اساسی‌ترین و محبوب‌ترین روش‌ها برای تخمین توابع مقدار Q به روشی بدون مدل است.
- هنگام برخورد با مسائل دامنه با ابعاد بالا یا عامل‌های پیوسته، یادگیری تقویتی از مشکل نمایش ناکارآمد ویژگی رنج می‌برد.
- یادگیری تقویتی عمیق از قدرت بازنمایی یادگیری عمیق برای مقابله با مشکلاتی که در یادگیری تقویتی وجود دارد، بهره می‌برد.

پرسش‌های مروری

۱. آیا یادگیری بانظارت به تنهایی قادر به حل یادگیری تعاملی می‌باشد؟

۲. یادگیری تقویتی برای حل چه مسائلی مناسب است؟ نحوه کار آن را شرح دهید؟

۳. هدف عامل در یادگیری تقویتی چیست؟

٤. عناصر تشکیل‌دهنده یک سیستم یادگیری تقویتی را نام ببرید؟

٥. نقش تابع مقدار و تابع پاداش را شرح دهید؟

٦. یادگیری تقویتی را با یادگیری ماشین مقایسه کنید؟

۷. مساله اساسی در فرآیند یادگیری تقویتی چیست؟

۸. خط‌مشی در یادگیری تقویتی چه کاربردی دارد؟

۹. تفاوت الگوریتم‌های مبتنی‌بر خط‌مشی و مبتنی‌بر مقدار در چیست؟

۱۰. معضل استخراج در مقابل اکتشاف را در یادگیری تقویتی شرح دهید؟

۱۱. روش‌های مبتنی‌بر مدل را با روش‌های بدون مدل مقایسه کنید؟

۱۲. چرا الگوریتم‌های برنامه‌نویسی پویا در یادگیری تقویتی کاربرد محدودی دارند؟

۱۳. ساده‌ترین روش یادگیری تقویتی چیست و چگونه کار می‌کند؟

۱٤. روش‌های مونت‌کارلو و برنامه‌نویسی پویا را با یک‌دیگر مقایسه کنید؟

۱٥. مزیت الگوریتم یادگیری تفاوت زمانی نسبت به روش‌های مونت‌کارلو و برنامه‌نویسی پویا در چیست؟

۱٦. ایده اصلی یادگیری تفاوت زمانی چیست؟

۱۷. مزایا و معایب روش گرادیان خط‌مشی در مقابل روش تابع مقدار چیست؟

۱۸. چند الگوریتم یادگیری تقویتی عمیق مبتنی‌بر کیو یادگیری را نام ببرید؟

فصل ۵

یادگیری انتقالی عمیق

اهداف

- آشنایی با یادگیری انتقالی و انگیزه استفاده از آن
- آشنایی با یاگیری انتقالی عمیق، رویکردها و استراتژی‌های آن

۰.۵ مقدمه

همانند هوش مصنوعی و یادگیری ماشین، مفهوم یادگیری انتقالی دهه‌های تکاملی را پشت سر گذاشته است. از همان سال‌های اولیه پیدایش هوش مصنوعی، محققان توانایی انتقال دانش را به عنوان یکی از ارکان اساسی هوش در نظر گرفته‌اند. خارج از هوش مصنوعی و علوم کامپیوتر، مفهوم یادگیری انتقالی نیز تحت اصطلاحات مختلفی ابداع شده است. به عنوان مثال، در زمینه‌ی روانشناسی یادگیری، مفهوم یادگیری انتقالی یک موضوع مهم در مدل‌سازی چیزی است که یادگیری موثر را تشکیل می‌دهد و آموزش معلمان را می‌سازد. این باور وجود دارد که بهترین تدریس دانش‌آموز را قادر می‌سازد "چگونه یاد بگیرد" و دانش آموخته‌شده را در موقعیت‌های آینده تطبیق دهد. با وجود نام‌های مختلف، کالبد و درون‌مایه اساسی آن‌ها شبیه هم است: اینکه بتوانند از تجربه گذشته خود استفاده و به تصمیمات موثرتر در آینده کمک کنند.

یادگیری عمیق در مقایسه با روش‌های سنتی یادگیری ماشین، وابستگی شدیدی به داده‌های بسیار زیاد برای آموزش دارند. چراکه، آن‌ها ویژگی‌ها یا همان الگوهای نهفته را از داده‌های آموزشی به‌طور خودکار یاد می‌گیرند و این امر نیاز به حجم زیادی از داده دارد. علاوه براین، بیشتر مدل‌های یادگیری عمیق برای یک حوزه خاص یا حتی یک کار خاص تخصص بدست آورده‌اند.

اگرچه ممکن است این مدل‌ها پیشرفته و دقت بسیار بالایی داشته باشند، این دقت فقط در مجموعه داده‌های بسیار خاص خواهد بود و در صورت استفاده در یک کار جدید که ممکن است مشابه آن هم باشد، عملکرد خود را به‌صورت قابل توجهی از دست می‌دهند. از همین‌رو در جهت حل این این مشکلات، انگیزه استفاده از یادگیری انتقالی را در یادگیری عمیق شکل می‌دهد تا فراتر از وظایف و حوزه‌های خاص سعی کند تا ببیند که چگونه می‌توان دانش را از مدل‌های از پیش‌آموزش‌دیده استفاده و از آن‌ها برای حل مشکلات جدید بهره ببرد.

۱.۵ یادگیری انتقالی

انسان‌ها توانایی ذاتی انتقال دانش در بین وظایف را دارا هستند. آن‌چه را که هنگام یادگیری یک کار به عنوان دانش کسب می‌کنند، به همان روش برای برای حل کارهای مرتبط استفاده می‌کند. به عنوان مثال، انسانی که دانش رانندگی اتومبیل را داشته باشد، رانندگی با اتوبوس را سریع‌تر از دیگران یاد می گیرد. مثال عینی‌تر آن یک دانش‌آموز در کلاس اول نحوه خواندن و نوشتن را یاد می‌گیرد و در کلاس‌های بالاتر از این خواندن و نوشتن در جهت بهره‌وری در یاد گرفتن درس‌ها استفاده می‌کند.

به‌طور خلاصه، انسان‌ها همه چیز را از پایه نمی‌آموزند و دانش خود را از حوزه‌های قبلاً آموخته شده به حوزه‌ها و وظایف جدیدتر منتقل می‌کنند. حال، هرچه این کارها ارتباط بیشتری با یک‌دیگر داشته باشند، انتقال دانش یا استفاده متقابل از آن آسان‌تر خواهد بود. این توانایی انتقال دانش در یادگیری ماشین، یعنی اینکه بتوان در آن الگویی که برای یک کار ایجاد و آموزش داده شده است، به عنوان نقطه شروع یک کار ثانویه مورد استفاده مجدد قرار گیرد را یادگیری انتقالی گویند. در منابع مختلف تعریف‌های متفاوتی برای یادگیری انتقالی ارائه شده است. در این کتاب، ما یادگیری انتقالی را اینگونه تعریف می‌کنیم:

استفاده از مدلی پیش‌آموزش داده شده در راستای انتقال دانش از این مدل برای وظیفه‌ای مشابه، در جهت بهبود عملکرد این وظیفه جدید.

برای درک تعریف رسمی یادگیری انتقالی، لازم است در ابتدا دامنه و وظیفه تعریف شوند. دامنه مجموعه داده‌ای است که برای آموزش استفاده و دامنه به صورت $D = \{\chi, P(X)\}$ نشان داده می‌شود که شامل دو مولفه: χ فضای ویژگی و $P(X)$ یک توزیع احتمال که در این تعریف $X = \{x_1, \dots, x_n\} \epsilon \chi$ است. وظیفه را می‌توان با فضای برچسب y و یک تابع مدل هدف $f(x)$ و به صورت $T = \{y, f(x)\}$ نشان داد. $f(x)$ را همچنین می‌توان به عنوان یک تابع احتمال شرطی $P(y|x)$ نوشت. حال می‌توان یادگیری انتقالی را به‌طور رسمی به صورت زیر تعریف کرد:

با توجه به دامنه منبع D_s و وظیفه یادگیری منبع T_s، دامنه هدف D_t و وظیفه یادگیری هدف T_t، جایی که حجم D_sها بیشتر از حجم D_tها باشد، یادگیری انتقالی روشی است برای بهبود عملکرد مدل هدف $f_T(.)$ برای وظیفه یادگیری هدف T_t با کسب دانش ضمنی از D_s و T_s، جایی که $D_s \neq D_t$ و $T_s \neq T_t$ است.

یادگیری انتقالی به این موضوع می‌پردازد که چگونه سیستم‌ها می‌توانند به سرعت خود را با شرایط جدید، وظایف جدید و محیط‌های جدید تطبیق دهند. این سیستم به سیستم‌های یادگیری ماشین امکان استفاده از داده‌ها و مدل‌های کمکی برای کمک به حل مسائلی را می‌دهد که تنها مقدار کمی داده در دامنه هدف موجود است. این امر سبب می‌شود که این سیستم‌ها از قابلیت اطمینان بیشتری برخوردار شود.

یادگیری انتقالی یک حوزه مهم در ویژه یادگیری ماشین است که می‌توانیم از زوایای مختلف به آن نگاه کرد. اول این‌که، توانایی یادگیری از داده‌های کوچک به نظر یک جنبه بسیار قوی از هوش انسانی است. به عنوان مثال، ما مشاهده می‌کنیم که نوزادان تنها از چند نمونه یاد می‌گیرند و می‌توانند به سرعت و به طور موثر از چند مثال به مفاهیم تعمیم دهند. این توانایی یادگیری از داده‌های کوچک را می‌توان تا حدودی با توانایی بشر در استفاده از تجربه قبلی و مدل‌های آموزش دیده‌شده برای کمک به حل مسائل هدف آینده توضیح داده شود. سازگاری یک توانایی ذاتی موجودات هوشمند است و مسلما عوامل هوشمند مصنوعی باید از توانایی یادگیری انتقالی برخوردار باشند.

دوم این‌که، در عمل در یادگیری ماشین اغلب با مجموعه داده‌هایی با اندازه کوچک احاطه شده‌ایم. بسیاری از سازمان‌ها به دلیل محدودیت‌های مختلفی از جمله، محدودیت‌های منابع گرفته تا منافع سازمان‌ها و قوانین و مقررات مربوط به حریم خصوصی کاربران، توانایی جمع‌آوری حجم عظیمی از داده‌ها را ندارند. این چالش با داده‌های کوچک، یک مشکل جدی است که بسیاری از سازمان‌ها با استفاده از فناوری هوش مصنوعی در مسائل خود با آن روبرو هستند. یادگیری انتقالی یک راه حل مناسب برای رفع این چالش است. چراکه، می‌تواند بسیاری از داده‌های کمکی و مدل‌های خارجی را استفاده کرده و آن‌ها را برای حل مسایل هدف تطبیق دهد.

علاوه براین، هنگام مواجهه با تغییرات غیرقابل پیش‌بینی و عبور از یک مدل آموخته شده از مرزهای دامنه، یادگیری انتقالی همچنان اطمینان حاصل می‌کند که عملکرد مدل بیش‌از حد از عملکرد مورد انتظار منحرف نمی‌شود. به این ترتیب، یادگیری انتقالی امکان استفاده مجدد از دانش را فراهم می‌کند. بنابراین، تجربه یکبار بدست آمده، می‌تواند به طور مکرر در دنیای واقعی اعمال شود. از منظر یک سیستم نرم افزاری، اگر سیستمی قادر باشد خود را از طریق یادگیری انتقالی در دامنه‌های جدید تطبیق دهد، گفته می‌شود که وقتی محیط خارجی تغییر می‌کند، از مقاومت و اطمینان بیشتری برخوردار است. چنین سیستم‌هایی اغلب در عمل ترجیح داده می‌شوند.

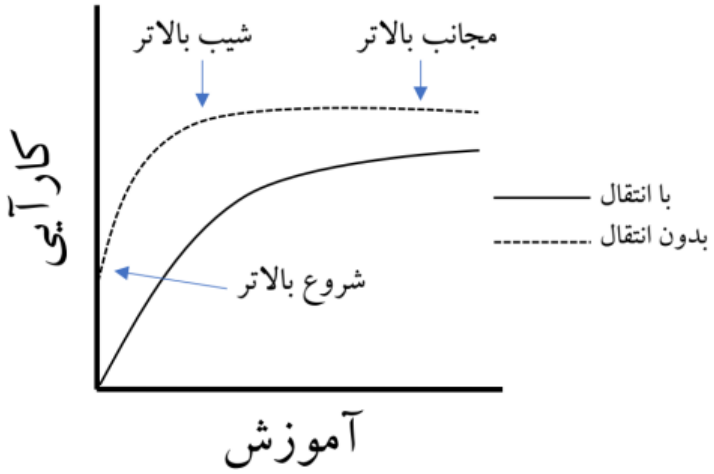
یادگیری انتقالی بر استخراج داده‌ها از یک دامنه مشابه متمرکز است تا توانایی یادگیری را افزایش یا تعداد نمونه‌های برچسب‌دار مورد نیاز در یک دامنه هدف را کاهش دهد. در یادگیری انتقالی، یک مدل از دانش به دست آمده از کار قبلی برای بهبود تعمیم در مورد دیگری بهره می‌برد. هدف از یادگیری انتقالی بهبود فرآیند یادگیری وظایف جدید با استفاده از تجربه به‌دست آمده از حل مسائل قبلی است که تا حدودی مشابه هستند.

یادگیری انتقالی به‌ویژه در مدل‌هایی که به تدریج آموزش می‌بینند بسیار مفید است و می‌توان از یک مدل موجود به عنوان نقطه شروع برای ادامه آموزش استفاده کند، مانند شبکه‌های یادگیری عمیق. توجه به این نکته ضروری است که، خروجی مدل‌های یادگیری انتقالی تحت تأثیر رابطه بین منبع و حوزه‌های هدف قرار دارد. اگر دامنه منبع و دامنه هدف دارای دانش مشترک کم‌تری باشند، این مدل بر یادگیری هدف و دقت تأثیر منفی خواهد گذاشت، که به آن **انتقال منفی** می‌گویند.

می‌توان سه معیار مشترک تعریف کرد که با استفاده از آن‌ها نشان داد انتقال می‌تواند اثربخشی یادگیری را بهبود بخشد (شکل ۵ ـ ۱).

- **شروع بالاتر:** عملکرد اولیه قابل دستیابی در کار هدف فقط با استفاده از دانش منتقل شده از منبع، قبل از انجام هرگونه یادگیری بیشتر، در مقایسه با عملکرد اولیه یک عامل جاهل بسیار بهتر است.

- **شیب بالاتر:** این معیار نشان‌دهنده مقدار زمان کمتری است که برای یادگیری کامل وظیفه مورد نظر با توجه به دانش منتقل شده در مقایسه با مدت زمان یادگیری آن از ابتدا لازم است.
- **مجانب بالاتر:** سطح عملکرد نهایی قابل دستیابی نسبت به وظیفه هدف در مقایسه با سطح نهایی بدون انتقال.



شکل ۵ ـ ۱ سه معیار که در آن انتقال ممکن است یادگیری را بهبود بخشد.

در طی فرآیند یادگیری انتقالی، سه سوال مهم به‌وجود می‌آید که باید به آن‌ها پاسخ داده شود:

۱. **چه چیزی باید انتقال داده شود:** اولین و مهم‌ترین مرحله در تمام فرآیند یادگیری انتقالی می‌باشد و اشاره به این موضوع دارد که کدام دانش می‌تواند منتقل شود. باید در جستجوی پاسخی باشیم که کدام قسمت از دانش را می‌توان از منبع به هدف منتقل کرد تا عملکرد کار مورد نظر بهبود یابد. باید سعی کنیم بفهمیم کدام قسمت از دانش مورد نیاز منبع بوده و چه چیزی بین منبع و هدف مشترک است.
۲. **زمان انتقال:** انتقال دانش همیشه بهبود نتایج را به‌همراه ندارد و گاهی ممکن است حتی سبب بدتر شدن نتایج شود. این اتفاق با انتقال منفی شناخته می‌شود. هدف ما از یادگیری انتقالی بهبود نتایج است نه کاهش آن. از همین رو، باید مراقب این مساله بود و بدانیم چه زمانی از یادگیری انتقالی استفاده و چه زمانی استفاده نکنیم.

۳. **نحوه انتقال:** پس از پاسخ به دو پرسش قبلی، حال نوبت به روش انتخابی برای انتقال دانش در دامنه مورد نظر می‌باشد.

۱.۱.۵ چه زمانی از یادگیری انتقالی استفاده کنیم؟

بزرگترین مزیت یادگیری انتقال زمانی مشاهده می‌شود که مجموعه داده‌های هدف نسبتاً کوچک باشد. در بسیاری از این موارد، مدل ممکن است مستعد بیش‌برازش باشد و ممکن است همیشه افزایش داده‌ها این مشکل را حل نکند. بنابراین، یادگیری انتقالی در مواردی که مدلی برای کار منبع در یک مجموعه آموزشی بسیار بزرگتر از آنچه برای کار هدف به دست می‌آید، آموزش دیده باشد، بهترین کاربرد را دارد. هر چند که در یادگیری ماشین، یک قاعده کلی که در تمام موارد قابل اجرا باشد، وجود ندارد. با این همه، به‌صورت خلاصه می‌توان زمانی که با موارد زیر مواجه شدیم از یادگیری انتقالی استفاده کنیم:

- **داده‌های آموزشی برچسب‌دار کافی برای آموزش شبکه وجود ندارد.**
- **در حال حاظر، شبکه‌ای وجود دارد که با حجم عظیمی از داده، از قبل برای کار مشابه آموزش داده شده است.**

۲.۵ یادگیری انتقالی عمیق

الگوریتم‌های متداول در یادگیری ماشین و یادگیری عمیق، به‌طور سنتی برای یک کار به‌صورت جداگانه طراحی می‌شوند. این مدل‌ها به محض تغییر توزیع فضای مشخصه، باید از ابتدا طراحی و بازسازی شوند. یادگیری انتقالی ایده‌ای برای غلبه بر یادگیری جداگانه و استفاده از دانش کسب شده برای یک کار در جهت حل کار مرتبط است. یادگیری انتقالی با یادگیری ماشین سنتی متفاوت است. چراکه شامل استفاده از یک مدل از قبل آموزش‌دیده به عنوان سکوی پرشی برای شروع یک کار ثانویه است.

مدلی را در نظر بگیرید که قادر به تمایز بین سیب رسیده و سیب پوسیده است. حتی اگر گلابی از نظر شکل متفاوت باشد، خصوصیاتی که سبب پوسیدگی میوه‌ها می‌شوند، می‌تواند مشترک باشد. در نظر بگیرید که یک سازمان برای سیب‌ها مدلی دارد

که می‌تواند بین سیب های تازه و پوسیده تمایز قائل شود، اما همان سازمان فاقد داده کافی برای گلابی است. در این حالت، مدل یادگیری عمیق می‌تواند با استفاده از مجموعه داده‌های سیب آموزش داده شود و سپس با استفاده از مجموعه داده کوچک‌تر گلابی به‌روز شود. استفاده از یادگیری انتقالی در یادگیری عمیق در حال حاظر بسیار محبوب است. چرا که می‌تواند شبکه‌های عمیق را با داده‌های نسبتا کم آموزش دهد.

یادگیری انتقالی عمیق را می‌توان به‌طور رسمی به صورت زیر تعریف کرد:

با توجه به تعریف وظیفه یادگیری انتقالی توسط $\{\boldsymbol{D_s}, \boldsymbol{T_s}, \boldsymbol{D_t}, \boldsymbol{T_t}, \boldsymbol{f}_T(.)\}$ این یک وظیفه یادگیری انتقالی عمیق است که در آن $\boldsymbol{f}_T(.)$ یک تابع غیرخطی می‌باشد که شبکه عصبی عمیق را منعکس می‌کند.

۱.۲.۵ انگیزه استفاده از یادگیری انتقالی عمیق

هرچند شبکه‌های عمیق به حل مسائلی پرداخته که برای دهه‌ها غیرممکن بوده، اما این شبکه‌ها برای حل مسائل پیچیده با چالش‌هایی مواجه هستند. وابستگی به داده‌ها یکی از جدی‌ترین این مشکلات است. یادگیری عمیق در مقایسه با روش‌های سنتی یادگیری ماشین، وابستگی شدیدی به داده‌های بسیار زیاد برای آموزش دارند. چراکه، آن‌ها ویژگی‌ها یا همان الگوهای نهفته را از داده‌های آموزشی به‌طور خودکار یاد می‌گیرند و این امر نیاز به حجم زیادی از داده دارد. با این حال، داده‌های آموزشی ناکافی در برخی حوزه‌های خاص (برای مثال، تشخیص بیماری‌های نادر) یک مساله گریزناپذیر می‌باشد. علاوه بر این، بیشتر مدل‌های یادگیری عمیق برای یک حوزه خاص یا حتی یک کار خاص تخصص بدست آورده‌اند.

اگرچه ممکن است این مدل‌ها پیشرفته و دقت بسیار بالایی داشته باشند، این دقت فقط در مجموعه داده‌های بسیار خاص خواهد بود و در صورت استفاده در یک کار جدید که ممکن است مشابه آن هم باشد، عملکرد خود را به‌صورت قابل توجهی از دست می‌دهند. این انگیزه استفاده از یادگیری انتقالی را شکل می دهد که فراتر از وظایف و حوزه‌های خاص می‌باشد، و سعی می‌کند ببیند که چگونه می‌توان دانش را از مدل‌های از قبل آموزش دیده استفاده و از آن برای حل مشکلات جدید بهره برد.

در یادگیری انتقالی، لازم نیست که داده‌های آموزشی و داده‌های آزمون از یک دامنه باشند. همچنین، مدل دامنه هدف نیازی به آموزش از ابتدا را ندارد. شبکه‌های عصبی عمیق با میلیون‌ها وزن قابل تنظیم در فرآیند آموزش به حل یک مساله می‌پردازند. ایده اصلی یادگیری انتقالی این است که با یک شبکه یادگیری عمیق فرآیند یادگیری آغاز شود که از قبل برای یک مساله مشابه آموزش داده شده است. همین امر، می‌تواند به میزان قابل توجهی نیاز به داده‌های آموزشی و زمان آموزش در دامنه هدف را کاهش دهد.

۲.۲.۵ مزایای استفاده از یادگیری انتقالی

استفاده از مدل‌هایی که قبلاً در زمینه خاصی آموزش دیده و استفاده مجدد از آن در زمینه دیگر، مزایای زیادی دارد. برخی از مزایای اصلی در زیر فهرست شده است.

- **آموزش با تعداد داده‌های کمتر:** شروع به آموزش یک مدل از ابتدا کار هزینه‌بر و نیاز به داده‌های زیادی دارد. به عنوان مثال، اگر بخواهیم الگوریتمی جدید ایجاد کنیم که بتواند اخم را تشخیص دهد، به داده‌های آموزشی زیادی نیاز داریم. مدل ما ابتدا باید نحوه تشخیص چهره را بیاموزد، و فقط در این صورت می تواند نحوه تشخیص حالات چهره مانند اخم را بیاموزد. در عوض، اگر ما از مدلی استفاده کنیم که قبلاً نحوه تشخیص چهره را یاد گرفته باشد و برای تشخیص اخم‌ها این مدل را دوباره آموزش دهیم، می توانیم با استفاده از داده‌های بسیار کم‌تر، نتیجه مشابه را بدست آوریم.
- **تعمیم‌دهی بهتر مدل:** استفاده از یادگیری انتقال در یک مدل، مدل را برای عملکرد مناسب با داده‌هایی که آموزش داده نشده‌اند، آماده می‌کند. مدل‌هایی که با استفاده از یادگیری انتقال آموزش داده شده‌اند، بهتر می‌توانند از یک کار به کار دیگر تعمیم داده شوند. چراکه آن‌ها برای یادگیری شناسایی ویژگی‌هایی که می توانند در زمینه‌های جدید استفاده شوند، آموزش دیده‌اند.
- **دسترسی به یادگیری عمیق را بیشتر می‌کند:** کار با یادگیری انتقالی استفاده از یادگیری عمیق را آسان‌تر می‌کند. با استفاده از مدلی که توسط یک متخصص یادگیری عمیق ایجاد شده و استفاده از آن برای یک مسئله جدید می توان بدون داشتن مهارت در یادگیری عمیق به نتایج خوبی دست یافت.

۳.۲.۵ استراتژی‌های استفاده از یادگیری انتقالی عمیق

یادگیری انتقالی را می‌توان از طریق چندین استراتژی مختلف در فضای یادگیری عمیق و ماشین به‌کار برد. در این بخش، تنها تکنیک‌های یادگیری عمیق را بررسی می‌کنیم. در راستای استفاده از یادگیری انتقالی عمیق، سه استراتژی اصلی وجود دارد:

- **انتقال ویژگی:** یکی از ساده‌ترین روش‌های یادگیری انتقالی، انتقال ویژگی است. همان‌طور که می‌دانیم، شبکه یادگیری عمیق از تعداد بسیاری لایه تشکیل شده است. این لایه‌ها مهم هستند. چراکه، یادگیری عمیق یک معماری لایه‌لایه است که ویژگی‌های مختلفی را در لایه‌های مختلف یاد می‌گیرد. در ابتدا لایه ورودی وجود دارد که ورودی را به لایه بعدی نگاشت می‌کند. سپس، لایه استخراج ویژگی که دارای لایه‌های درونی مختلفی است. خروجی لایه استخراج ویژگی "ویژگی‌هایی" هستند که می‌توانند به عنوان مثال، ویژگی‌های از یک چهره شخص همانند، چشم، بینی یا دهان را نشان دهند و سپس می‌توانند به صورت سلسله مراتبی برای ترجمه به ویژگی‌های سطح بالاتر استفاده شوند. لایه طبقه‌بندی نهایی ویژگی‌های یافت‌شده در لایه استخراج ویژگی را درهم می‌کشد و طبقه‌بندی می‌کند. به عنوان مثال، تصویر ورودی نشان‌دهنده یک چهره انسان است یا خیر. هر کدام از ویژگی‌های متفاوتی که ساخته شده‌اند در لایه طبقه‌بندی به خروجی تبدیل می‌شوند. ایده پشت انتقال ویژگی آن است که از لایه‌های ورودی و استخراج ویژگی که در یک مجموعه داده مشخص آموزش دیده شده‌اند، یک لایه طبقه‌بندی جدید را برای دامنه مساله مربوط آموزش دهد. به این ترتیب، به عنوان مثال، یک شبکه یادگیری عمیق مورد استفاده برای تشخیص خودرو در یک تصویر می‌تواند یک لایه طبقه‌بندی آموزش‌دیده برای تشخیص دوچرخه داشته باشد. این روش در صورتی ایده‌آل است که دو حوزه مسائل مشابه باشند.
- **تنظیم‌ـ دقیق:** تنظیم‌ـ دقیق می‌تواند به این معنی باشد که ما لایه‌های بعدی شبکه یادگیری عمیق را آموزش می‌دهیم در حالی که لایه‌های پیشین را ثابت نگه می‌داریم. به این ترتیب، لایه‌هایی را که مختص ویژگی‌های طبقه‌بندی در مقایسه با لایه‌های پیشین که عمومی‌تر هستند، تنظیم‌ـ دقیق می‌کنیم. این روش

زمانی ایده‌آل است که دامنه‌های مساله دارای فاصله باشند و نیاز به ویژگی‌های جدیدی داشته باشند که طبقه‌بندی شوند.

- **رویکرد مدل از پیش‌آموزش دیده:** ساده ترین استراتژی حل یک مساله هدف استفاده مستقیم از یک مدل پیش‌آموزش‌دیده در یک کار منبع است. چنین مدل‌هایی به طور معمول شبکه‌های عصبی بزرگی با میلیون‌ها پارامتر هستند که روزها یا حتی هفته‌ها روی دستگاه‌های پیشرفته آموزش دیده‌اند.

مدل از پیش‌آموزش دیده چیست؟

یکی از الزامات اساسی برای یادگیری انتقالی، وجود مدل‌هایی است که به خوبی در انجام وظایف منبع عمل می‌کنند. خوشبختانه، دنیای یادگیری عمیق معتقد به اشتراک‌گذاری است. بسیاری از پیشرفته‌ترین معماری‌های یادگیری عمیق توسط تیم‌ها در دامنه‌های مختلفی همانند، بینایی رایانه و پردازش زبان طبیعی که دو حوزه محبوب یادگیری عمیق وجود هستند، به اشتراک‌گذاری شده‌اند.

مدل از پیش‌آموزش‌دیده شبکه‌ای ذخیره شده است که قبلا روی یک مجموعه داده بزرگ، با میلیون‌ها پارامتر در یک وضعیت پایدار آموزش دیده و به اشتراک گذاشته شده است. برای حل یک مسئله ما باید یک مدل از پیش‌آموزش‌دیده، از یک مسئله مشابه را داشته باشیم. به جای ساختن یک مدل از ابتدا برای حل یک مساله مشابه، ما از پیش‌آموزش‌دیده روی مسئله دیگر به عنوان نقطه شروع استفاده می‌کنیم. می‌توان از مدل پیش‌آموزش‌دیده به همان صورت که هست استفاده کرد یا از یادگیری انتقالی برای شخصی‌سازی این مدل برای یک کار معین استفاده کرد.

شایان ذکر است، در زمان انتخاب یک مدل پیش‌آموزش‌دیده، باید دقت کرد. اگر بیان مسئله‌ای که در دست داریم بسیار متفاوت از گزاره‌ای است که در آن مدل از پیش‌آموزش‌دیده در آن مساله آموزش دیده است، پیش‌بینی که می‌گیریم بسیار نادرست خواهد بود.

بسته به اندازه مجموعه داده جدید و شباهت مجموعه داده جدید به مجموعه داده اصلی، روش استفاده از یادگیری انتقال متفاوت خواهد بود. چهار سناریو زیر

به شما کمک می‌کند تا در مورد نحوه استفاده از مدل پیش‌آموزش‌دیده در مورد خود تصمیم بگیرید:

۱. **اندازه مجموعه داده کوچک، درحالیکه شباهت داده‌ها بسیار زیاد است.** در این مورد خاص، نیازی به حفظ مدل نیست. چراکه تشابه داده‌ها بسیار زیاد است. هرچند، طبق صورت مسئله، باید لایه های خروجی را سفارشی‌سازی و اصلاح کنیم. در این مورد، از مدل از پیش‌آموزش‌دیده، به عنوان یک استخراج کننده ویژگی استفاده کرده‌ایم.
۲. **اندازه مجموعه داده کوچک و همچنین شباهت داده‌ها بسیار کم است.** در این مورد، در ابتدا باید لایه های اولیه مدل از پیش‌آموزش‌دیده را ثابت نگه داریم. سپس، دوباره لایه های باقیمانده را آموزش دهیم. شایان ذکر است که لایه های بالایی با توجه به مجموعه داده‌های جدید سفارشی‌سازی می‌شوند. همچنین، لایه های اولیه به دلیل اندازه کوچکترشان به صورت از پیش‌آموزش داده شده باقی مانده و وزن‌هایش را ثابت نگه می‌داریم.
۳. **اندازه مجموعه داده زیاد اما شباهت داده‌ها بسیار کم است.** در این مورد، آموزش شبکه عصبی موثرتر خواهد بود. چراکه یک مجموعه داده بزرگ داریم و از آن‌جایی که داده‌هایی که استفاده می‌کنیم با داده‌های آموزشی متفاوت‌اند، از همین‌رو، بهتر است شبکه عصبی را با توجه به داده‌های خود از ابتدا آموزش دهیم.
٤. **اندازه مجموعه داده زیاد و همچنین شباهت داده‌ها بسیار زیاد است.** می‌توان گفت این وضعیت پایانی و ایده‌آل است. مدل‌های از پیش‌آموزش‌دیده در این مورد موثرتر هستند.

۳.۵ رویکردهای یادگیری انتقالی عمیق

یادگیری انتقالی عمیق نحوه استفاده از دانش سایر زمینه‌ها را توسط شبکه‌های عصبی عمیق بررسی می‌کند. براساس تکنیک‌های مورد استفاده در یادگیری انتقالی عمیق، می‌توان یادگیری انتقالی عمیق را به چهار دسته طبقه‌بندی کرد: **یادگیری انتقالی عمیق**

مبتنی‌بر نمونه، یادگیری انتقالی عمیق مبتنی‌بر نگاشت، یادگیری انتقالی عمیق مبتنی‌بر شبکه و **یادگیری انتقالی عمیق مبتنی‌بر تخاصم**.

۱.۳.۵ یادگیری انتقالی عمیق مبتنی‌بر نمونه

رویکردهای یادگیری انتقالی مبتنی‌بر نمونه استفاده مجدد از داده‌های دارای برچسب از دامنه منبع برای آموزش مدل دقیق‌تری برای یک وظیفه یادگیری هدف است. اگر دامنه منبع و دامنه هدف کاملا مشابه باشند، می‌توانیم مستقیما داده‌های دامنه منبع را در دامنه هدف ادغام کنیم.

انگیزه رایج در پشت رویکردهای یادگیری انتقالی مبتنی‌بر نمونه این است که برخی از داده‌های دارای برچسب دامنه منبع، هنوز برای یادگیری یک مدل دقیق برای دامنه هدف مفید و برخی از آن‌ها قابل استفاده نبود و یا حتی در صورت استفاده می‌توانند به عملکرد مدل هدف آسیب برسانند. برای درک بهتر می‌توانیم از تحلیل واریانس و بایاس استفاده کنیم.

وقتی مجموعه داده‌های دامنه هدف کوچک باشد، مدل ممکن است از سطح واریانس بالایی برخوردار باشد. بنابراین، خطای تعمیم‌دهی مدل زیاد است. با افزودن بخشی از داده‌های دامنه منبع به عنوان مجموعه داده‌های کمکی، می توان واریانس مدل را به طور بالقوه کاهش داد. با این حال، اگر توزیع داده‌های این دو حوزه بسیار متفاوت باشد، مدل یادگیری جدید ممکن است دارای واریانس بالایی باشد. بنابراین، اگر بتوانیم نمونه‌هایی از دامنه منبع را که از توزیع مشابه دامنه هدف پیروی می‌کنند را جدا کنیم، می‌توان از آن‌ها استفاده مجدد کنیم و هم واریانس و هم بایاس مدل یادگیری هدف را کاهش دهیم.

یادگیری انتقال عمیقی مبتنی‌بر نمونه، به استفاده از یک استراتژی تنظیم وزن خاص اشاره دارد تا نمونه‌های جزئی را از دامنه منبع به عنوان مکمل به مجموعه آموزشی در دامنه هدف انتخاب کند و بر این فرض استوار است که، **اگر چه بین دو حوزه تفاوت وجود دارد، نمونه‌های جزئی در حوزه منبع را می‌توان با دامنه هدف با وزن‌های مناسب مورد استفاده قرار داد.**

۲.۳.۵ یادگیری انتقالی عمیق مبتنی‌بر تخاصم

یکی از راه‌های استفاده از یادگیری انتقالی، استفاده از مدل‌سازی مولد در یادگیری عمیق است که منجر به مدل‌های خصمانه می‌شود. استفاده از مدل‌های مولد بدون‌نظارت برای کاهش وابستگی به داده‌های دارای برچسب است. در یک دامنه هدف، داده‌های دارای برچسب محدود هستند، اما ممکن است داده‌های بدون برچسب فراوانی در یک دامنه منبع موجود باشد. می‌توان از یادگیری ویژگی بدون‌نظارت برای بازنمایی از داده‌های بدون برچسب استفاده و از مدل‌های مولد می‌توان برای امکان انتقال دانش به یک دامنه هدف استفاده کرد.

یادگیری انتقالی عمیق مبتنی‌بر تخاصم به فناوری خصمانه الهام گرفته از شبکه‌های مولد تخامصی، برای یافتن بازنمایی قابل انتقال که هم برای دامنه منبع و هم دامنه هدف قابل استفاده است، اشاره دارد. یادگیری خصمانه به طور طبیعی با یادگیری انتقالی کار می‌کند. به عنوان یک مدل تولیدی ، مدل‌های مولد تخاصمی می‌توانند داده های دامنه هدف را تولید کنند و در نوع جدیدی از یادگیری انتقالی معروف به "داده‌افزایی" داده‌ها را افزایش دهند. یادگیری خصمانه می‌تواند برای "ترجمه" یک نمونه دامنه منبع دارای برچسب به یک نمونه دامنه هدف ضمن حفظ برچسب آن استفاده و می‌تواند بین نمونه‌های دامنه منبع و هدف به صورت کاملا بدون نظارت ارتباط ایجاد کند.

از آنجا که شبکه مولد تخاصمی و همچنین یادگیری خصمانه به‌عنوان یک چارچوب جدید و قدرتمند ظاهر شده‌اند، محققان سعی کرده‌اند مدل‌های یادگیری انتقالی را براساس چارچوب یادگیری خصمانه توسعه دهند.

۳.۳.۵ یادگیری انتقالی عمیق مبتنی‌بر نگاشت

یادگیری انتقالی عمیق مبتنی‌بر نگاشت به ترسیم نمونه‌ها از دامنه منبع و دامنه هدف به یک فضای داده جدید اشاره دارد. در این فضای داده جدید، نمونه‌ها از دو دامنه به‌طور مشابه و مناسب در یک شبکه عصبی مرکزی هستند. این رویکرد، براساس این فرض است که، **اگر چه بین دو دامنه اصلی تفاوت وجود دارد، اما آن‌ها می‌توانند در یک فضای داده‌ای جدید، مشابه باشند.**

۴.۳.۵ یادگیری انتقالی عمیق مبتنی‌بر شبکه

یادگیری انتقالی عمیق مبتنی‌بر شبکه به استفاده مجدد از شبکه جزئی که در دامنه منبع از پیش‌آموزش دیده است (از جمله ساختار شبکه و پارامترهای ارتباطی آن) و انتقال آن به بخشی از شبکه عصبی عمیق که در دامنه هدف مورد استفاده می‌شود، اشاره دارد. این رویکرد بر این فرض است که، **شبکه عصبی شبیه سازوکار پردازش مغز انسان و یک فرایند انتزاعی تکراری و مداوم است. لایه‌های جلویی شبکه می‌توانند به عنوان یک استخراج‌کننده ویژگی عمل کنند و ویژگی‌های استخراج‌شده چندمنظوره هستند.**

خلاصه فصل

- **یادگیری عمیق در مقایسه با روش‌های سنتی یادگیری ماشین، وابستگی شدیدی به داده‌های بسیار زیاد برای آموزش دارند.**
- **بیشتر مدل‌های یادگیری عمیق برای یک حوزه خاص یا حتی یک‌کار خاص تخصص بدست آورده‌اند. از همین‌رو در جهت حل این این مشکلات، انگیزه استفاده از یادگیری انتقالی را در یادگیری عمیق شکل می‌دهد تا فراتر از وظایف و حوزه‌های خاص عمل کند.**
- **یادگیری انتقالی به این موضوع می‌پردازد که چگونه سیستم‌ها می‌توانند به سرعت خود را با شرایط جدید، وظایف جدید و محیط‌های جدید تطبیق دهند.**
- **خروجی مدل‌های یادگیری انتقالی تحت تأثیر رابطه بین منبع و حوزه‌های هدف قرار دارد.**
- **بزرگترین مزیت یادگیری انتقال زمانی مشاهده می‌شود که مجموعه داده‌های هدف نسبتاً کوچک باشد.**
- **استفاده از مدل‌هایی که قبلاً در زمینه خاصی آموزش دیده و استفاده مجدد از آن در زمینه دیگر، مزایای زیادی دارد.**

پرسش‌های مروری

۱. یادگیری انتقالی چیست؟

۲. یادگیری انتقالی چه کمکی به یادگیری ماشین می‌کند؟

۳. معیارهایی که نشان می‌دهند یادگیری انتقالی اثربخش بوده است را نام ببرید؟

٤. سه سوالی که در طی فرآیند یادگیری انتقالی وجود دارد را نام ببرید؟

۵. استفاده از یادگیری انتقالی چه زمانی مناسب است؟

٦. چرا از یادگیری انتقالی در یادگیری عمیق استفاده می‌شود؟ مزایای آن را شرح دهید؟

۷. مدل از پیش‌آموزش‌دیده چیست و چه مزایایی دارد؟

فصل ۶

یادگیری عمیق هندسی:
یادگیری بازنمایی گراف

اهداف

- آشنایی با یادگیری عمیق هندسی و علت استفاده از آن
- نحوه کار شبکه‌های عصبی گراف
- آشنایی با شبکه‌ی همگشتی گراف

۰.۶ مقدمه

ساختار داده گراف در حوزه‌های مختلفی همانند، بیوشیمی، پردازش‌تصویر، سیستم‌های توصیه‌گر، تحلیل شبکه‌های اجتماعی و غیره، کاربرد گسترده‌ای دارد. رویکردهای گوناگونی برای آموزش مدل‌های یادگیری ماشین برروی داده‌هایی با ساختار گراف با استفاده از تکنیک‌های پیش‌پردازش وجود دارد. با این همه، آن‌ها انعطاف‌پذیری برای انطباق کامل با مجموعه داده و وظیفه موجود را ندارند و اثبات شده است که، استفاده از این این داده‌ها در مدل یادگیری ماشین به دلیل ابعاد بالا و ویژگی غیراقلیدسی داده‌های گراف، دشوار است. در مجموع می‌توان گفت که این رویکردهای سنتی به عنوان یک مرحله پیش‌پردازش استفاده می‌شوند، نه بخشی از فرآیند آموزش.

شبکه های عصبی عمیق در دهه گذشته موفقیت چشم‌گیری کسب کرده‌اند. با این حال، انواع اولیه شبکه‌های عصبی تنها با استفاده از داده‌های منظم یا اقلیدسی قابل پیاده‌سازی هستند. درحالی‌که، بسیاری از داده‌ها در دنیای واقعی دارای ساختار گراف‌اند که غیر اقلیدسی است. همین غیرمنظم بودن ساختار داده‌ها منجر به پیشرفت‌های اخیر در شبکه‌های عصبی گراف شده است.

شبکه‌های عصبی گراف اجازه ایجاد یک مدل یادگیری ماشین انتها_به_انتها را می‌دهد که به‌طور هم‌زمان آموزش داده می‌شود تا بازنمایی از داده‌های با ساختار گراف را یاد بگیرد. شبکه‌های عصبی گراف را می‌توان برای وظایفی مختلف، از خوشه‌بندی گرفته تا طبقه‌بندی یا رگرسیون، بر روی داده‌های با ساختار گراف اعمال کرد و همچنین می‌توانند بازنمایی در سطح گره، یال و یا گراف را یاد بگیرند.

۱.۶ یادگیری عمیق هندسی

در حالی‌که مدل‌های یادگیری عمیق در دهه گذشته، در برخورد با ورودی‌هایی به شکل تصاویر، گفتار یا ویدیو که اساس ساختار آن‌ها اقلیدسی است، موفقیت‌آمیز عمل کرده، اخیرا، علاقه محققین در تلاش برای استفاده از یادگیری بر روی داده‌های غیراقلیدسی افزایش یافته است. یادگیری عمیق هندسی، زمینه نوظهور تحقیقاتی است که سعی در

تعمیم معماری یادگیری عمیق برای کار با داده‌های غیراقلیدسی دارد، تا این شکاف را پر کند.

داده‌های غیراقلیدسی می‌توانند مفاهیم پیچیده‌تری را با دقت بیشتری نسبت به نمایش‌های یک‌ـ بعدی و دوـ بعدی نشان دهند. یکی از ساختارهای غیراقلیدسی مهم، گراف می‌باشد. گراف‌ها نوع خاصی از ساختار داده‌ها هستند، که از رئوسی تشکیل شده که با یال‌ها متصل هستند. از این ساختار داده انتزاعی می‌توان تقریبا در مدل‌سازی هر چیزی استفاده کرد. به عنوان مثال، در شبکه‌های اجتماعی، ویژگی‌های کاربران می‌تواند به‌عنوان سیگنال‌هایی بر روی رئوس گراف اجتماعی مدل شود. در علوم اعصاب، از مدل‌سازی گراف برای نشان‌دادن ساختارهای تشریحی و عملکردی مغز استفاده می‌شود.

۱.۱.۶ گراف

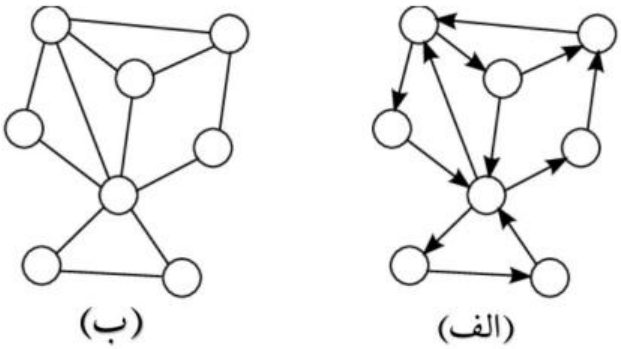
در علوم کامپیوتر، گراف یک ساختمان داده با دو مولفه راس* و یال است. گراف G را می‌توان به‌صورت $G = (V, E)$ توصیف کرد؛ جایی که V مجموعه‌ای از رئوس و E مجموعه‌ای از یال‌هایی است که بین این گره‌ها ارتباط برقرار می‌کند. بسته به وابستگی‌های بین این رئوس، یال‌ها می‌تواند جهت‌دار یا بدون‌جهت باشند (شکل ٦ـ ١).

یک روش مناسب برای نمایش گراف‌ها از طریق **ماتریس همجواری** $\boldsymbol{A \in \mathbb{R}^{|v|\times|v|}}$ است. برای نشان دادن یک گراف با ماتریس همجواری، گره‌ها را در گراف مرتب می‌کنیم تا هر گره یک ردیف و ستون خاص را در ماتریس همجواری نمایه کند.

داده‌ها با ساختار گراف در دنیای واقعی فراوان و در همه‌جا دیده می‌شوند. به‌عنوان مثال، شبکه‌های اجتماعی، شبکه‌های زیستی، ساختارهای مولکولی، گراف دانش و غیره. به‌طور کلی، هر مجموعه داده‌ای که شامل گره و یال شود گراف می‌باشد. همچنین، طیف گسترده‌ای از مسائل یادگیری مربوط به گراف‌ها وجود دارد، همانند طبقه‌بندی گره‌های نیمه‌نظارتی، طبقه‌بندی گراف، پیش‌بینی پیوند، تشخیص جامعه، خوشه بندی گراف و

*در اغلب موارد از راس با عنوان گره نیز یاد می‌شود. در این کتاب، ما از هر دو اصطلاح استفاده می‌کنیم.

غیره. به دلیل فراوانی داده‌هایی با ساختار گراف و مسائل یادگیری گراف، مطالعه چگونگی یادگیری از گراف‌ها بسیار مهم است. علاوه براین ، گراف یک موضوع مهم در یادگیری ماشین است. چراکه، بسیاری از مدل های یادگیری ماشین، مانند شبکه های عصبی و شبکه‌های بیزی، با محاسبات برروی گراف‌ها تحقق می‌یابند.



شکل ٦ ـ ١ (الف) گراف جهت‌دار (ب) گراف بدون‌جهت

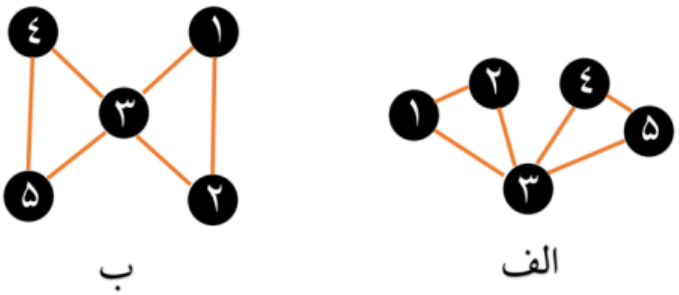
با این همه، داده‌ها با ساختار گراف به‌قدری پیچیده هستند که چالش‌های زیادی را برای الگوریتم‌های موجود یادگیری ماشین ایجاد می‌کنند. این مشکل از آن‌جایی ناشی می‌شود که، ابزارهای معمول یادگیری ماشین و یادگیری عمیق در انواع داده‌های ساده تخصص دارند؛ همانند تصاویر با ساختار و اندازه یکسان، که می‌توانیم آن‌ها را به عنوان گراف‌هایی با اندازه ثابت تصور کنیم، یا متن و گفتار که داده‌هایی دارای توالی بوده و می‌توانیم آن‌ها را به عنوان گراف‌های خطی تصور کنیم. اما، گراف‌های پیچیده‌تری نیز وجود دارد؛ بدون شکل ثابت و با اندازه متغیری از گره‌های نامرتب، که در آن گره‌ها می‌توانند همسایگان مختلفی داشته باشند.

چرا تجزیه و تحلیل گراف‌ها دشوار است؟

گراف‌ها ویژگی غیراقلیدسی دارند؛ به این معنی که نمی‌توانیم آن‌ها را توسط هیچ سیستم مختصاتی که با آن آشنا هستیم، نمایش دهیم. این امر تفسیر داده‌های گراف را در مقایسه با داده‌هایی با فرم تصاویر، امواج و سری‌های زمانی دشوارتر می‌کند. علاوه براین،

گراف‌ها شکل ثابتی ندارند. به عنوان مثال شکل ٦ــ۲ را نگاه کنید. گراف (الف) و گراف (ب) از منظر ساختاری و بصری کاملا متفاوت هستند. اما وقتی آن‌ها را به ماتریس مجاورتی تبدیل کنیم، دو گراف ماتریس مجاورتی یکسانی دارند (با فرض این‌که وزن یال‌ها را در نظر نگیریم). بنابراین، آیا باید این دو گراف را متفاوت یا مشابه در نظر بگیریم؟

در آخر این‌که، به‌طور کلی تجسم یک گراف برای تفسیر انسان دشوار است. منظور ما گراف‌های کوچک همانند مثال‌های بالا نیست. گراف‌های عظیمی وجود دارند که شامل صدها یا هزاران گره هستند. گراف‌هایی که بعد بسیار بالا داشته و گره‌ها به‌طور متراکم گروه‌بندی شده و درک گراف حتی برای یک انسان دشوار است. بنابراین، آموزش یک ماشین برای این کار چالش‌برانگیز است.



شکل ٦ــ۲ یک گراف با نمایش ساختاری و بصری متفاوت

۲.۶ شبکه‌های عصبی گراف

برخلاف تصاویر، داده‌ها با ساختار گراف، دارای نمایشی تنسوری نیستند که بتوان به‌راحتی توسط شبکه‌های عصبی معمولی آن‌ها را خواند. همین امر استفاده از یادگیری عمیق را برای کار با گراف محدود می‌کند.

معماری‌های معمولی شبکه عصبی، همانند شبکه‌های عصبی پیش‌خور، شبکه‌های عصبی همگشتی و شبکه‌های عصبی بازگشتی، نیاز به سیگنال‌های ورودی دارند که در

فرم‌هایی با اندازه نمایش داده می‌شوند. براین اساس، لایه های شبکه عصبی قادر به استخراج ویژگی‌ها به‌صورت سلسه‌مراتبی و یادگیری الگوها از داده‌ها هستند. اگرچه این شبکه‌های عصبی، در انواع مختلف داده‌ها به موفقیت‌های بزرگی دست یافته‌اند، اما این شبکه‌های عصبی معمولی نمی‌توانند مستقیما برروی گراف‌ها اعمال شوند. در عین حال، محققین بر این عقیده هستند که، یادگیری عمیق برای کار با گراف‌ها یک زمینه عالی است که می‌توان در آن روش‌های جدید شبکه‌های عصبی را امتحان کرد.

شبکه‌های عصبی گراف، دسته‌ای از روش‌های یادگیری عمیق هستند که به‌طور خاص، برای استنباط بر داده‌های توصیف‌شده توسط گراف‌ها طراحی شده‌اند. ایجاد مدل‌هایی که مستقیما بروی گراف‌ها کار می‌کنند، مطلوب‌تر است. چراکه، می‌توانیم اطلاعات بیشتری در مورد ساختار و خصوصیات آن‌ها را بدست آوریم.

شبکه‌های عصبی گراف، به‌طور مستقیم برروی گراف‌ها اعمال می‌شوند و روشی آسان برای انجام وظایفی همانند، پیش‌بینی سطح گره، یال و گراف ارائه می‌کنند. تا پیش از، توسعه شبکه‌های عصبی گراف، روش‌های یادگیری عمیق توانایی اعمال برروی یال‌ها در جهت استخراج دانش و پیش‌بینی را نداشتند. در عوض، تنها بر اساس ویژگی‌های گره عمل می‌کردند.

هر گره در یک گراف مجموعه‌ای از ویژگی‌هایی را نشان می‌دهد که توسط گره شناسایی می‌شوند و با مجموعه‌ای از برچسب‌ها در ارتباط است. سپس، از شبکه عصبی گراف برای آموزش وزن‌هایی استفاده می‌شود که می‌توان از آن‌ها برای پیش‌بینی برچسب‌ها برای گره‌های جدید استفاده کرد. شبکه‌های عصبی گراف با بازنمایی وضعیت گره‌ها به‌صورت تکراری و استفاده از شبکه‌های عصبی پیش‌خور کار می‌کنند و انتقال پیام را انجام می‌دهند.

۱.۲.۶ شبکه‌های عصبی انتقال پیام[۱]:

شبکه‌های عصبی گراف، دارای فرموله‌بندی مستقل بسیاری هستند. با این همه، می‌توان آن‌ها را در یک چارچوب انتقال پیام[۲] یکپارچه‌سازی کرد. شبکه عصبی انتقال پیام، نوعی از مدل‌های شبکه عصبی است که به‌طور خاص برای کار بروی گراف‌ها طراحی شده است. با توجه به یک گراف بدون‌جهت G با ویژگی‌های گره x_v، انتقال روبه‌جلو یک شبکه عصبی گراف، معمولا شامل دو فاز است: فاز انتقال پیام که برای استخراج ویژگی‌های زیرساخت محلی در اطراف گره‌ها استفاده می‌شود، و فاز بازخوانی[۳] که یک فاز تجمیع برای خلاصه کردن ویژگی‌های جداگانه گره در یک بردار از ویژگی‌های سطح گراف است.

فاز انتقال پیام (گراف همگشتی)، به تعداد T تکرار اجرا می‌شود و شامل توابع پیام M_t و توابع بروزرسانی U_t است. در هر فاز انتقال پیام، وضعیت‌های پنهان رئوس h_v^t براساس پیام‌های m_v^t بروز می‌شوند:

$$m_v^{t+} = \sum_{u \epsilon \Gamma(v)} M_t(h_v^t, z_u^t)$$

$$h_v^{t+۱} = U_t(h_v^t, m_v^{t+۱})$$

جایی که $\Gamma(v)$ مجموعه همسایگان v را در گراف نشان می دهد. M_t و U_t هر دو توابع مشتق‌پذیر با پارامترهای قابل یادگیری است.

در فاز بازخوانی، یک تابع بازخوانی R به مجموعه وضعیت‌های نهفته نهایی h_v^T اعمال می‌شود تا پیش‌بینی را به‌صورت زیر انجام دهد:

$$\hat{y} = R(\{h_v^T | v \epsilon G\})$$

---

۱ Message Passing Neural Networks (MPNN)
۲ message passing framework
۳ readout phase

تابع بازخوانی، معمولا با ساختن یک نمایش واحد از کل گراف با درنظر گرفتن وضعیت‌های پنهان نهایی h کار می‌کند:

$$h = \sum_{v \epsilon G} h_v^T$$

سپس، یک شبکه عصبی پیش‌خور f به h برای خروجی اعمال می‌شود:

$$\hat{y} = f(h)$$

انتقال پیام را می‌توان به‌عنوان یک عملگر همگشت که برای استخراج ویژگی‌های محلی بروی گره‌ها اعمال می‌شود، مشاهده کرد. مشابه همان کاری که یک همگشت در شبکه عصبی همگشتی سنتی برای هر پیکسل انجام می‌دهد. بیشتر شبکه‌های عصبی گراف موجود را می‌توان در این چارچوب گنجاند. تفاوت‌ها در طراحی منحصربه فرد M_t و U_t در کارهای مختلف نهفته است. به‌عنوان مثال، M_t می‌تواند به‌سادگی مجموع یا میانگین باشد، یا از معماری‌های پیچیده شبکه‌های عصبی مانند، سازوکار توجه و RNN باشد. تابع بروزرسانی U_t نیز می‌تواند از یک لایه خطی تک‌لایه به پرسپترون چندلایه تا GRU محدوده‌بندی شود.

۲.۲.۶ یادگیری بازنمایی گراف

یادگیری عمیق در کار با داده‌های به فرم گراف به عنوان **یادگیری عمیق هندسی**[۱]، **یادگیری بازنمایی گراف**[۲] و یا **تعبیه‌سازی گراف**[۳] نیز شناخته می‌شوند که به دنبال یادگیری بازنمایی اطلاعات ساختاری در مورد گراف است. هدف از یادگیری بازنمایی گراف، ساخت مجموعه‌ای از ویژگی‌ها است که ساختار گراف و داده‌های موجود در آن را نشان دهد. ایده پشت این روش، یادگیری نگاشتی است که گره‌ها یا گراف را به عنوان نقاطی در یک فضای برداری با ابعاد کم تعبیه کند، به‌طوری که، این نگاشت بهینه شود

[۱] geometric deep learning
[۲] graph representation learning
[۳] graph embedding

تا روابط هندسی در این فضای آموخته شده، ساختار گراف اصلی را منعکس کند. پس از بهینه‌سازی فضای تعبیه‌شده، می‌توان از این تعبیه‌سازی آموخته شده به‌عنوان دریافت ویژگی‌های ورودی یادگیری ماشین استفاده کرد.

تمایز کلیدی بین رویکردهای یادگیری بازنمایی و کارهای گذشته در این است که، چگونه با مساله ثبت اطلاعات ساختاری در مورد گراف‌ها رفتار می‌کنند. در کارهای گذشته، این مساله را به عنوان مرحله‌ای پیش‌پردازش، با استفاده از مهندسی‌های آماری دستی در جهت استخراج اطلاعات ساختاری، مورد بررسی قرار می‌دادند. در مقابل، رویکردهای یادگیری بازنمایی با استفاده از یک رویکرد داده‌محور برای یادگیری تعبیه‌سازهایی که ساختار گراف را رمزگذاری می‌کند، این مساله را به خود یادگیری عمیق واگذار می‌کنند.

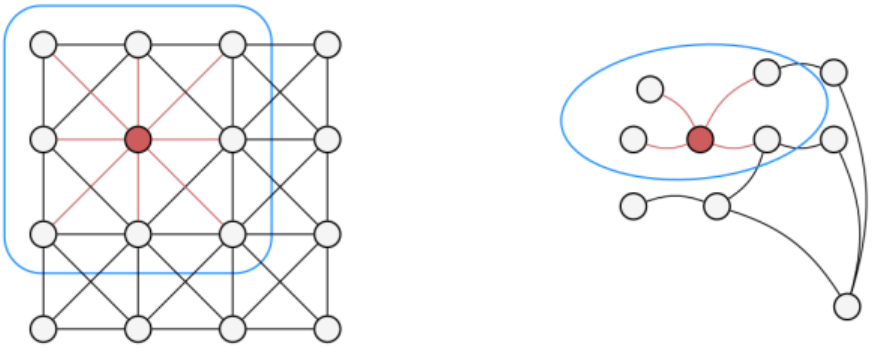
۳.۲.۶ شبکه‌های همگشتی گراف

همانند معرفی شبکه‌های عصبی همگشتی که به سرعت بخشیدن یادگیری و افزایش دقت با پردازش سلسله‌مراتبی داده کمک می‌کند، شبکه‌های همگشتی گراف نیز برای انجام همان کار اما در داده‌های گراف است. شبکه‌های همگشتی گراف، دسته‌ای از شبکه‌های عصبی بسیار قدرتمند هستند که برای استخراج الگوهای آماری معنادار از داده‌های ساختاریافته گراف، از عملیات همگشت که اصلی‌ترین عملگر در مدل‌های شبکه‌های عصبی همگشتی می‌باشد، استفاده کرده و قابلیت اجرای بسیار خوب با حداقل آموزش را دارند.

در حقیقت، آن‌ها آنقدر قدرتمند هستند که حتی یک شبکه همگشتی گراف با دولایه که به‌صورت تصادفی مقداردهی اولیه شده‌اند، می‌توانند بازنمایی ویژگی‌های مفید از گره‌ها را بدست آورند. به‌طور کلی، شبکه‌های همگشتی گراف، یک بازنمایی جدید از هر راس گراف را با تجمیع ویژگی‌های همسایه‌های خود پیدا می‌کند. همگشت دوبعدی را می‌توان به یک همگشت گراف تعمیم داد.

همان‌طور که در شکل ٦ـ٣ نشان داده شده است، یک تصویر را می‌توان یک مورد خاص از گراف دانست که پیکسل‌ها توسط پیکسل‌های مجاور بهم متصل می‌شوند. همگشت گراف، هر راس را به عنوان یک پیکسل در نظر گرفته و ویژگی‌های راس هدف و همسایگان راس هدف را تجمیع می‌کند.

شبکه‌های عصبی گراف، بدون شک داغ‌ترین موضوع در یادگیری عمیق مبتنی‌بر گراف هستند. این شبکه‌های مدرن، با تقلید از شبکه‌های عصبی همگشتی، الگوهای ساختاری محلی و سراسری گراف‌ها را از طریق توابع همگشت و بازخوانی طراحی شده، می‌آموزند. تفاوت عمده بین شبکه‌های عصبی همگشتی و شبکه‌های همگشتی گراف این است که، شبکه‌های عصبی همگشتی به‌طور خاص ساخته شده‌اند تا برروی داده‌های با ساختاری منظم (اقلیدسی) کار کنند. درحالی‌که، شبکه‌های همگشتی گراف، نسخه تعمیم‌یافته شبکه‌های عصبی همگشتی هستند که در آن داده‌ها ساختاری نامنظم دارد.



شکل ٦ـ٣ همگشت دو بعدی (شکل سمت چپ) در مقابل همگشت گراف (شکل سمت راست).

شبکه‌های همگشتی گراف را می‌توان در ۲ گروه عمده: شبکه‌های همگشتی گراف طیفی[۱] و شبکه‌های همگشتی گراف فضایی[۲] دسته‌بندی کرد. رویکردهای طیفی، با معرفی فیلترها از منظر پردازش سیگنال گراف که مبتنی‌بر تئوری طیفی گراف، جایی که در آن

[۱] Spectral
[۲] Spatial

عملیات همگشت به عنوان حذف نویز از سیگنال‌های گراف است، تفسیر می‌شود. رویکردهای فضایی، با جمع‌آوری اطلاعات گره‌های همسایه، به‌طور مستقیم همگشت گراف را فرموله می‌کند. همگشت مبتنی‌بر فضا، تجمع گره و همسایگان آن را می‌گیرد تا بازنمایی جدیدی برای آن بدست آورد. یک روش معمول برای این‌کار این است که، چندین لایه همگشت گراف را روی هم قرار دهید.

از نظر مقیاس‌پذیری و موازی‌سازی، روش‌های طیفی با اندازه گراف به‌صورت نمایی افزایش می‌یابند و به کل گراف در حافظه نیاز دارند. بنابراین، این روش‌ها برای داده‌های با مقیاس بزرگ با میلیاردها گره و یال (مانند، گراف شبکه‌های اجتماعی) یا معماری‌های موازی مناسب نیستند. در مقابل، روش‌های فضایی از این مشکل رنج نمی‌برند. چراکه آن‌ها از طریق تجمیع ویژگی‌های همسایگان، همگشت را به‌طور مستقیم در حوزه گراف انجام می‌دهند تا بتوانند گراف‌های بزرگ را اداره کنند.

از منظر تعمیم مدل‌ها، مدل‌های مبتنی‌بر طیف، یک گراف ثابت را در نظر می‌گیرند و در تعمیم گراف‌های دیده نشده تبحر ندارند. روش‌های فضایی این محدودیت را ندارند زیرا همگشت آنها به صورت محلی انجام می‌شود. از همین رو می‌تواند به‌راحتی وزن‌های استفاده شده در همگشت را در مکان‌ها وساختارهای مختلف به اشتراک بگذارند.

سرانجام این‌که، روش‌های طیفی محدود به کار بر روی گراف‌های بدون‌جهت هستند. در حالی که، روش‌های فضایی می‌توانند با تغییر در تابع تجمیع با ورودی‌های چندمنبعی و گراف‌های جهت‌دار مقابله کنند. با توجه به عواملی که بیان شد، مدل‌های مبتنی‌بر فضا، بیشتر از مدل‌های طیفی، توجه جامعه را به خود جلب کرده‌اند.

خلاصه فصل

- **یادگیری عمیق هندسی، زمینه نوظهور تحقیقاتی است که سعی در تعمیم معماری یادگیری عمیق برای کار با داده‌های غیراقلیدسی دارد.**
- **یکی از ساختارهای غیراقلیدسی مهم، گراف می‌باشد.**

- داده‌ها با ساختار گراف در دنیای واقعی فراوان و در همه‌جا دیده می‌شوند.
- شبکه‌های عصبی گراف، به‌طور مستقیم برروی گراف‌ها اعمال می‌شوند و روشی آسان برای انجام وظایفی همانند، پیش‌بینی سطح گره، یال و گراف ارائه می‌کنند.
- شبکه عصبی انتقال پیام، نوعی از مدل‌های شبکه عصبی است که به‌طور خاص برای کار بروی گراف‌ها طراحی شده است.
- هدف از یادگیری بازنمایی گراف، ساخت مجموعه‌ای از ویژگی‌ها است که ساختار گراف و داده‌های موجود در آن را نشان دهد.

پرسش‌های مروری

۱. چرا مطالعه برروی داده‌ها با ساختار گراف مهم است؟

۲. چالش‌های موجود کار با گراف‌ها را شرح دهید؟

۳. شبکه عصبی انتقال پیام از چند فاز تشکیل شده است؟ نام برده و نحوه کار هر یک از این فازها را شرح دهید؟

٤. هدف از یادگیری بازنمایی گراف چیست؟

٥. تمایز بین رویکردهای یادگیری بازنمایی و کارهای گذشته در چیست؟

٦. دو گروه عمده از شبکه‌های همگشتی گراف را نام برده و با یک‌دیگر مقایسه کنید؟

مراجع

Aggarwal, Charu C. *Neural Networks and Deep Learning a Textbook*. Cham Springer, ٢٠١٨.

Alexander Alexander Zai. *Deep Reinforcement Learning in Action.* Manning Publications Company, ٢٠٢٠.

Alom, Md Zahangir, Tarek M. Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C. Van Essen, Abdul A. S. Awwal, and Vijayan K. Asari. "A State-of-the-Art Survey on Deep Learning Theory and Architectures." *Electronics* ٨, no. ٣ (March ٥, ٢٠١٩): ٢٩٢. https://doi.org/١٠,٣٣٩٠/electronics٨٠٣٠٢٩٢.

Andriy Burkov. *The Hundred-Page Machine Learning Book*. Quebec, Canada] Andriy Burkov, ٢٠١٩.

Arbones, Marc. "Deep Learning: Creating Bridges between DMPs in Auto Encoders and Recurrent Neural Networks," ٢٠١٧.

ARMACKI, Aleksandar. "Application of Autoencoders on Single-Cell Data," ٢٠١٨.

Aurélien Géron. *Hands-on Machine Learning with Scikit-Learn and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems*, ٢٠١٩.

Barto, Andrew G, and Richard S Sutton. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. Mit Press, ٢٠١٨.

Bertsekas, Dimitri P. *Reinforcement Learning and Optimal Control*. Belmont, Massachusetts: Athena Scientific, ٢٠١٩.

BÜTEPAGE, JUDITH. "Generative Models for Action Generation and Action Understanding," ٢٠١٩.

Charniak, Eugene. *Introduction to Deep Learning*. Cambridge, Ma: Mit Press, ۲۰۱۸.

Chen, Gang. “A Gentle Tutorial of Recurrent Neural Network with Error Backpropagation.” *ArXiv:1610.02583 [Cs]*, January ۱٤, ۲۰۱۸. https://arxiv.org/abs/۱٦۱۰,۰۲٥۸۳.

Chollet, François. *Deep Learning with Python*. Shelter Island (New York, Estados Unidos): Manning, Cop, ۲۰۱۸.

Di, Wei, Anurag Bhardwaj, and Jianing Wei. *Deep Learning Essentials : Your Hands-on Guide to the Fundamentals of Deep Learning and Neural Network Modeling*. Birmingham: Packt, ۲۰۱۸.

Dong, Hao, Zihan Ding, and Shanghang Zhang. *Deep Reinforcement Learning : Fundamentals, Research and Applications*. Singapore: Springer, ۲۰۲۰.

Faradonbeh, Soroor Malekmohammadi, and Faramarz Safi-Esfahani. “A Review on Neural Turing Machine.” *ArXiv:1904.05061 [Cs]*, November ۱٥, ۲۰۲۰. https://arxiv.org/abs/۱۹۰٤,۰٥۰٦۱.

Foster, David. *Generative Deep Learning : Teaching Machines to Paint, Write, Compose, and Play*. Beijing ; Boston ; Farnham ; Sebastopol ; Tokyo O’reilly, ۲۰۱۹.

Ghosh, Tomojit. “SUPERVISED AND UNSUPERVISED TRAINING OF DEEP AUTOENCODER,” ۲۰۱۷.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Cambridge, Massachusetts: The Mit Press, ۲۰۱۷.

Graesser, Laura, and Wah Loon Keng. *Foundations of Deep Reinforcement Learning : Theory and Practice in Python*. Boston: Addison-Wesley, ۲۰۲۰.

Gravelines, Céline. “Deep Learning via Stacked Sparse Autoencoders for Automated Voxel-Wise Brain Parcellation Based on Functional Connectivity,” ۲۰۱٤.

Greve, Rasmus Boll, Emil Juul Jacobsen, and Sebastian Risi. "Evolving Neural Turing Machines for Reward-Based Learning." *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, July ٢٠, ٢٠١٦. https://doi.org/١٠,١١٤٥/٢٩٠٨٨١٢,٢٩٠٨٩٣٠.

Gulli, Antonio, and Sujit Pal,. *Deep Learning with Keras : Implement Neural Networks with Keras on Theano and TensorFlow*. Birmingham, Uk ; Mumbai: Packt Publishing, ٢٠١٧.

Hamilton, William L. *Graph Representation Learning*. S.L.: Morgan & Claypool Publish, ٢٠٢٠.

Ho, Jonathan, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. "Flow++: Improving Flow-Based Generative Models with Variational Dequantization and Architecture Design." *ArXiv:1902.00275 [Cs, Stat]*, May ١٥, ٢٠١٩. https://arxiv.org/abs/١٩٠٢,٠٠٢٧٥.

Hope, Tom, Yehezkel S Resheff, and Itay Lieder. *Learning TensorFlow : A Guide to Building Deep Learning Systems*. Sebastopol, Ca: O'reilly Media, ٢٠١٧.

Kazak, Veronica. "UNSUPERVISED FEATURE EXTRACTION WITH AUTOENCODER FOR THE REPRESENTATION OF PARKINSON'S DISEASE PATIENTS," ٢٠١٨.

Kobyzev, Ivan, Simon J. D. Prince, and Marcus A. Brubaker. "Normalizing Flows: An Introduction and Review of Current Methods." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ٢٠٢٠, ١–١. https://doi.org/١٠,١١٠٩/TPAMI.٢٠٢٠,٢٩٩٢٩٣٤.

Lupo, Francesco. "Variational Autoencoder for Unsupervised Anomaly Detection," ٢٠١٩.

Makhzani, Alireza. "Unsupervised Representation Learning with Autoencoders," ٢٠١٨.

MANELA, Binyamin. “Deep Reinforcement Learning for Complex Manipulation Tasks with Sparse Feedback,” ۲۰۲۰.

Marcus, Gary. “Deep Learning: A Critical Appraisal.” arXiv.org, ۲۰۱۸. https://arxiv.org/abs/۱۸۰۱,۰۰۶۳۱.

Marsland, Stephen. *Machine Learning : An Algorithmic Perspective*. Boca Raton, Fl: Crc Press, ۲۰۱۵.

Maxim Lapan. *Deep Reinforcement Learning Hands-on : Apply Modern RL Methods to Practical Problems of Chatbots, Robotics, Discrete Optimization, Web Automation, and More*. Birmingham ; Mumbai Packt January, ۲۰۲۰.

Mohit Sewak, and Springerlink (Online Service. *Deep Reinforcement Learning : Frontiers of Artificial Intelligence*. Singapore: Springer Singapore, ۲۰۱۹.

Nielsen, Michael A. *Neural Networks and Deep Learning.* Determination Press, ۲۰۱۵.

Nikhil Buduma. *Fundamentals of Deep Learning : Designing Next-Generation Artificial Intelligence Algorithms*. Beijing, Boston, Farnham, Sebastopol, Tokyo: O’reilly, ۲۰۱۷.

Osinga, Douwe. *Deep Learning Cookbook : Practical Recipes to Get Started Quickly*. Sebastopol, Ca: O’reilly Media, ۲۰۱۸.

Oussidi, Achraf, and Azeddine Elhassouny. “Deep Generative Models: Survey.” *2018 International Conference on Intelligent Systems and Computer Vision (ISCV)*, April ۲۰۱۸. https://doi.org/۱۰,۱۱۰۹/isacv.۲۰۱۸,۸۳٥٤۰۸۰.

Patterson, Josh, and Adam Gibson. *Deep Learning : A Practitioner’s Approach*. Beijing ; Sebastopol: O’reilly Media, ۲۰۱۷.

Pong, Vitchyr, Shixiang Gu, Murtaza Dalal, and Sergey Levine. “Temporal Difference Models: Model-Free Deep RL for Model-Based Control.” *ArXiv:1802.09081 [Cs]*, February ۲٤, ۲۰۲۰. https://arxiv.org/abs/۱۸۰۲,۰۹۰۸۱.

Ragav Venkatesan, and Baoxin Li. *Convolutional Neural Networks in Visual Computing : A Concise Guide*. Boca Raton, Fl: Crc Press, ٢٠١٨.

Ramsundar, Bharath, and Reza Bosagh Zadeh. *TensorFlow for Deep Learning : From Linear Regression to Reinforcement Learning*. Beijing: O'reilly Media, ٢٠١٨.

Rodríguez Esmerats, Pau. "Graph Neural Networks and Its Applications," ٢٠١٩.

Rosebrock, Adrian. *Deep Learning for Computer Vision with Python : Starter Bundle*. United States: Pyimagesearch, ٢٠١٧.

Ruder, Sebastian. "An Overview of Gradient Descent Optimization Algorithms." arXiv.org, ٢٠١٦. https://arxiv.org/abs/١٦٠٩,٠٤٧٤٧.

Rungta, Krishna. *TensorFlow in 1 Day : Make Your Own Neural Network*. United States?, ٢٠١٨.

Sandro Skansi. *Guide to Deep Learning Basics : Logical, Historical and Philosophical Perspectivces*. Cham, Switzerland: Springer, ٢٠٢٠.

SEXTON, Conor. "Advancing Neural Turing Machines: Learning a Solution to the Shortest Path Problem," ٢٠١٧.

Soetens, Robin. "Reinforcement Learning Applied to Keepaway, a RoboCup-Soccer Subtask," ٢٠١٠.

Tan, Chuanqi, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. "A Survey on Deep Transfer Learning." arXiv.org, ٢٠١٨. https://arxiv.org/abs/١٨٠٨,٠١٩٧٤.

Uday Kamath, Zhanjiang Liu, and James Whitaker. *Deep Learning for NLP and Speech Recognition*. Cham, Switzerland Springer, ٢٠١٩.

Vasilev, Ivan, Daniel Slater, Gianmario Spacagna, Peter Roelants, and Valentino Zocca. *Python Deep Learning : Exploring Deep*

Learning Techniques and Neural Network Architectures with PyTorch, Keras and TensorFlow. Birmingham ; Mumbai Packt January, ۲۰۱۹.

Vath, Dirk. "Deep Reinforcement Learning in Dialog Systems," ۲۰۱۸.

Ven, van de. "A Deep Graph Convolutional Neural Network Aiding in Finding Feasible Shunt Plans," ۲۰۱۸.

Wani, M A, Farooq Ahmad Bhat, Saduf Afzal, and Asif Iqbal Khan. *Advances in Deep Learning*. Singapore: Springer, ۲۰۲۰.

Wu, Zonghan, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. "A Comprehensive Survey on Graph Neural Networks." *IEEE Transactions on Neural Networks and Learning Systems*, ۲۰۲۰, ۱–۲۱. https://doi.org/۱۰.۱۱۰۹/TNNLS.۲۰۲۰.۲۹۷۸۳۸٦.

Yang, Qiang. *Transfer Learning*. Cambridge Cambridge University Press, ۲۰۲۰.

Yu, Chao, Jiming Liu, and Shamim Nemati. "Reinforcement Learning in Healthcare: A Survey." *ArXiv:1908.08796 [Cs]*, April ۲٤, ۲۰۲۰. https://arxiv.org/abs/۱۹۰۸.۰۸۷۹٦.

Zhang, Ziwei, Peng Cui, and Wenwu Zhu. "Deep Learning on Graphs: A Survey." *ArXiv:1812.04202 [Cs, Stat]*, March ۱۳, ۲۰۲۰. https://arxiv.org/abs/۱۸۱۲.۰٤۲۰۲.