هوش مصنوعی چطور کار می‌کند؟

از جادوگری تا علم

مترجم: منصور تقی‌زاده || Author: Ronald T. Kneusel

The Best People in Life Are FREE

فهرست

مقدمه نویسنده

بسیاری از کتاب‌ها به شما آموزش می‌دهند چگونه هوش مصنوعی (AI) بسازید. به همین ترتیب، بسیاری از کتاب‌های محبوب درباره هوش مصنوعی صحبت می‌کنند. با این حال، آنچه به نظر می‌رسد گم شده است، کتابی است که به شما در سطح مفهومی آموزش دهد هوش مصنوعی چگونه کار می‌کند. هوش مصنوعی جادو نیست؛ می‌توانید بدون غرق شدن در ریاضیات پیچیده، درک کنید که چه می‌کند.

این کتاب آن خلأ را با توضیحی بدون ریاضی درباره نحوه کار هوش مصنوعی پر می‌کند. در حالی که برخی کتاب‌ها در جزئیات غرق می‌شوند و برخی دیگر دیدی کل‌نگرانه ارائه می‌دهند، کتاب حاضر شما را به نوک درخت می‌برد تا از آن جا همه چیز را به خوبی مشاهده کنید. هدف ارائه جزئیات کافی برای درک رویکرد، بدون گرفتار شدن در ریاضیات دقیق است. اگر این موضوع کنجکاوی شما را برانگیخته، دعوتتان می‌کنم ادامه دهید.

من برای اولین بار در سال 1987، در یک دوره کارشناسی با همین نام، درباره هوش مصنوعی یاد گرفتم. آنچه مردم معمولاً هوش مصنوعی می‌خوانند، طی دهه‌های گذشته کمی تغییر کرده است. با این حال، هدف همچنان یکسان باقی مانده است: تقلید از رفتار هوشمند در یک ماشین.

در دهه 1980، تعداد کمی از افراد دلیلی برای یادگیری درباره هوش مصنوعی داشتند، اگر اصلاً از آن آگاه بودند. تأثیر هوش مصنوعی بر زندگی روزمره آن‌ها اندک بود. با این حال، دهه 1980 مدت‌هاست که تمام شده و هوش مصنوعی همه جا حضور دارد و روزانه بر زندگی ما به روش‌های متعدد تأثیر می‌گذارد؛ از گوشی‌های هوشمندی که به ما می‌گویند کجا رانندگی کنیم و کجا نه، تا تگ کردن دوستان و خانواده در عکس‌ها، تا مقالات و تبلیغات مداومی که به ما در اینترنت عرضه می‌شود، چه خوشمان بیاید چه نه. و این بدون در نظر گرفتن انفجار اخیر هوش مصنوعی شامل مدل‌های زبانی بزرگ است، که بسیاری بالاخره آن را به عنوان «هوش مصنوعی حقیقی» تفسیر می‌کنند.

هوش مصنوعی همچنین به شیوه‌هایی در پشت صحنه حضور دارد که به ندرت متوجه آن می‌شویم: برنامه‌ریزی پروازهای هوایی، حمل و نقل و لجستیک، خودکارسازی کارخانه‌ها، تصویربرداری ماهواره‌ای از زمین و کمک به پزشکان برای تصمیم‌گیری اینکه آیا یک توده، سرطانی است یا خیر.

چرا اکنون درباره هوش مصنوعی یاد بگیریم؟

این کتاب با توضیح اینکه چه اتفاقی افتاده، چه زمانی افتاده، چرا افتاده و مهم‌تر از همه، چگونه افتاده - همه بدون هیاهو یا یک معادله ریاضی - به این سؤال پاسخ می‌دهد. صادقانه بگویم، واقعیت پشت انقلاب هوش مصنوعی به اندازه کافی شگفت‌انگیز است؛ هیاهو غیرضروری است.

در این نقطه، احساس می‌کنم باید چند کلمه درباره خودم بگویم. به هر جهت، از شما می‌خواهم که با من در سفری از دنیای هوش مصنوعی همراه شوید، بنابراین معقول است که درباره راهنمای خود کنجکاو شوید. همان‌طور که قبلاً ذکر شد، من در اواخر دهه 1980 با هوش مصنوعی آشنا شدم. کارم را در هوش مصنوعی، در زیرشاخه‌ای به نام یادگیری ماشین، در سال 2003

آغاز کردم و مدل‌های یادگیری ماشینی را به تصاویر فراصوتی داخل عروقی اعمال کردم. فراصوت داخل عروقی (IVUS) تکنیکی در تصویربرداری پزشکی است که داخل رگ‌های خونی، به‌ویژه شریان‌های کرونری، را قابل مشاهده می‌سازد. این روش تصاویری دقیق از دیواره‌های رگ‌ها ارائه می‌دهد و به پزشکان کمک می‌کند تا شرایطی مانند تجمع پلاک، انسدادها یا سایر ناهنجاری‌هایی که ممکن است به بیماری قلبی منجر شوند را ارزیابی کنند.

من برای اولین بار در سال 2010 درباره یادگیری عمیق شنیدم. یادگیری عمیق زیرشاخه‌ای از یادگیری ماشین است. تفاوت بین یادگیری عمیق، یادگیری ماشین و هوش مصنوعی را در فصل 1 توضیح خواهم داد، اما فعلاً می‌توانید آن‌ها را یک چیز در نظر بگیرید.

در سال 2012، با ظهور چیزی که بعداً الکس‌نت (AlexNet) نامیده شد هوش مصنوعی به صحنه وارد شد - یا حداقل به اخبار - و یک آزمایش عجیب در گوگل که شامل کامپیوترهایی بود که یاد گرفتند گربه‌ها را در ویدیوهای یوتیوب شناسایی کنند خیلی سر و صدا کرد. وقتی گوگل مقاله‌اش را ارائه داد من در اتاق کنفرانس بین‌المللی یادگیری ماشین 2012 در ادینبورگ اسکاتلند حضور داشتم.

در سال 2016، دکتری علوم کامپیوتر خود را با تخصص در هوش مصنوعی در دانشگاه کلرادو (Colorado)، بولدر (Boulder)، تحت راهنمایی مایکل موزر (Michael Mozer) به پایان رساندم. از آن موقع به بعد، روزانه در زمینه هوش مصنوعی کار کرده‌ام، عمدتاً در صنعت دفاع، با یک وقفه کوتاه در سال 2016 برای کمک به تأسیس یک استارتاپ هوش مصنوعی پزشکی.

پس از الکس‌نت، چیزها به سرعت تغییر کردند، زیرا به نظر می‌رسید هر ماه یک «معجزه» جدید مرتبط با هوش مصنوعی در ادبیات آکادمیک، اگر نگوییم در اخبار شامگاهی، ظاهر می‌شد. تنها راه برای به‌روز ماندن، شرکت در کنفرانس‌ها چندین بار در سال بود؛ انتظار برای انتشار نتایج در یک مجله آکادمیک بی‌فایده بود، زیرا این حوزه با سرعت بیش از حد معمول انتشار آکادمیک در حال پیشرفت بود.

من این مقدمه را در نوامبر 2022 در کنفرانس NeurIPS می‌نویسم. NeurIPS تقریباً برجسته‌ترین کنفرانس هوش مصنوعی به شمار می‌رود (لطفاً ایمیل‌های نفرت‌آمیز نفرستید!) و این اولین باری است که از زمان همه‌گیری کووید-19 به‌صورت حضوری برگزار می‌شود. تعداد حضار بالا است، اگرچه شاید به اندازه کنفرانس 2019 نباشد که برای تعیین 13500 نفرِ شرکت‌کننده، قرعه‌کشی برگزار شد. این واقعیت که حضور در کنفرانس‌ها طی یک دهه از چند صد نفر به بیش از 10000 نفر شکوفا شده، نشان‌دهنده اهمیت پژوهش هوش مصنوعی است.

نام‌های رهبران صنعت فناوری که از این کنفرانس‌ها حمایت می‌کنند (که مکان‌های اصلی برای شکار دانشجویان تحصیلات تکمیلی هستند) نیز اهمیت هوش مصنوعی را نشان می‌دهند؛ در این کنفرانس غرفه‌های نمایشگاهی گوگل، دیپ‌مایند (که متعلق به گوگل است)، متا (یعنی فیسبوک)، آمازون، اپل و دیگران را خواهید یافت. هوش مصنوعی بسیاری از فعالیت‌های این شرکت‌ها را هدایت می‌کند و پول هنگفتی به همراه دارد. هوش مصنوعی با داده‌ها کار می‌کند و این شرکت‌ها تمام داده‌هایی که ما به رایگان در ازای خدماتشان به آن‌ها می‌دهیم را می‌بلعند.

تا پایان این کتاب، شما خواهید فهمید که هوش مصنوعی در پشت صحنه چه می‌کند. در نهایت، درک آن چندان دشوار نیست، هرچند دشواری موضوع قطعاً در جزئیات نهفته است.

کتاب به این ترتیب پیش می‌رود:

فصل اول، مروری بر هوش مصنوعی (بزن بریم...)

با مروری سریع بر اصول اولیه هوش مصنوعی و یک مثال ساده شروع می‌کنیم.

فصل دوم، تاریخ هوش مصنوعی (چرا حالا؟)

هوش مصنوعی از آسمان نازل نشده است. این فصل پیشینه هوش مصنوعی را به شما ارائه می‌دهد و توضیح می‌دهد که چرا انقلاب هوش مصنوعی اکنون در حال وقوع است.

فصل سوم، مدل‌های کلاسیک (یادگیری ماشین قدیمی)

هوش مصنوعی مدرن تماماً درباره شبکه‌های عصبی است، اما برای درک آنچه شبکه‌های عصبی انجام می‌دهند، درک مدل‌هایی که قبل از آن‌ها وجود داشتند مفید است.

فصل چهارم، شبکه‌های عصبی (هوش مصنوعی مغز مانند)

اگر می‌خواهید بدانید شبکه عصبی چیست، چگونه آموزش داده می‌شود و چگونه استفاده می‌شود، این فصل برای شماست.

فصل پنجم، شبکه‌های عصبی پیچشی (هوش مصنوعی می‌آموزد تا ببیند)

بخش زیادی از قدرت هوش مصنوعی مدرن از یادگیری روش‌های جدید برای نمایش داده‌ها ناشی می‌شود. اگر این جمله برای شما بی‌معنی است، این فصل به شما کمک خواهد کرد.

فصل ششم، هوش مصنوعی مولد (هوش مصنوعی خلاق می‌شود)

مدل‌های یادگیری ماشینی نظارت‌شده سنتی برچسب‌هایی به ورودی‌ها متصل می‌کنند. هوش مصنوعی مولد خروجی‌های جدیدی از جمله متن، تصویر و حتی ویدئو تولید می‌کند. این فصل دو رویکرد محبوب را بررسی می‌کند: شبکه‌های مولد تخاصمی (GANها) و مدل‌های انتشار. GANها شهود لازم برای بررسی مدل‌های انتشار و در فصل هفتم، مدل‌های زبانی بزرگ (LLMها) را فراهم می‌کنند. مدل‌های انتشار در تولید تصاویر و ویدئوهای فوتورئالیستیک با جزئیات بالا از دستورات متنی مهارت دارند.

فصل هفتم، مدل‌های زبانی بزرگ (هوش مصنوعی حقیقی؟)

انتشار مدل زبانی بزرگ ChatGPT توسط OpenAI در پاییز 2022 ممکن است دوره هوش مصنوعی حقیقی را آغاز کرده باشد. این فصل مدل‌های زبانی بزرگ را بررسی می‌کند: آن‌ها چه هستند، چگونه کار می‌کنند و چرا ادعا می‌شود که چیزی جدید و تحول‌آفرین‌اند؟

فصل هشتم، تأملات (پیامدهای هوش مصنوعی)

ظهور مدل‌های زبانی بزرگ چشم‌انداز هوش مصنوعی را تغییر داده است. این فصل درباره پیامدهای آن تأمل می‌کند.

اگر نظرات یا سؤالاتی درباره مطالب این کتاب دارید، مایلم بشنوم.

لطفاً به آدرس rkneuselbooks@gmail.com ایمیل بزنید.

حالا اگر آماده هستید، شروع کنیم 😳

مقدمه مترجم

در عصر حاضر، هوش مصنوعی به بخشی جدایی‌ناپذیر از زندگی مردم تبدیل شده است. از پیشنهاد فیلم در پلتفرم‌های پخش آنلاین گرفته تا تشخیص بیماری‌ها در پزشکی و هدایت خودروهای خودران، رد پای این فناوری در همه جا دیده می‌شود. اما برای بسیاری هوش مصنوعی هم‌چنان پدیده‌ای اسرارآمیز است که گویی از جهانی ناشناخته سرچشمه گرفته است. کتاب «هوش مصنوعی چطور کار می‌کند؟» نوشتهٔ ران نیوسل، متخصص برجستهٔ حوزهٔ یادگیری ماشین و هوش مصنوعی با مدرک دکتری از دانشگاه کلرادو، با رویکردی روایی، خوانندگان را به سفری در تاریخچه و مفاهیم اساسی این فناوری می‌برد. او از تجربهٔ شخصی خود در این حوزه بهره می‌گیرد تا نشان دهد هوش مصنوعی چگونه از ایده‌های ابتدایی در دهه‌های گذشته به مدل‌های پیشرفتهٔ امروزی مانند شبکه‌های عصبی و مدل‌های زبانی بزرگ رسیده است. این کتاب نه تنها برای مبتدیان بلکه برای افراد آشنا با این حوزه که می‌خواهند دیدگاهی تازه و جامع به دست آورند، منبعی ارزشمند است.

آقای نیوسل در چند سال اخیر نویسندهٔ پر کاری بوده است. اگر به وب سایت او (www.rkneusel.com) سر بزنید فهرست آثار زیر را خواهید دید:

- 2018: اعداد تصادفی و کامپیوترها، ویرایش سوم (Springer)
- 2021: ریاضیات برای یادگیری عمیق (No Starch Press)
- 2022: کد عجیب و غریب (No Starch Press)
- 2022: حساب توضیح داده شده (Amazon)
- 2022: راهنمای توضیح دهنده برای مجموعه‌های اعداد (Amazon)
- 2022: راهنمای توضیح دهنده برای برنامه‌نویسی کامپیوتر (Amazon)
- 2023: هوش مصنوعی چطور کار می‌کند؟ (No Starch Press)
- 2024: هنر تصادف (No Starch Press)
- 2025: یادگیری عمیق کاربردی، ویرایش دوم (No Starch Press)
- 2025: ریاضیات برای برنامه‌نویسی (No Starch Press)

این کتاب از آثار جدید او محسوب می‌شود و Lexile Measure آن در سایت آمازون 1130L ذکر شده که معیاری برای تعیین میزان پیچیدگی یک متن است و نشان می‌دهد کتاب حاضر برای بزرگسالان با مهارت خواندن پیشرفته طراحی شده است (معیار Lexile Measure برای رمان معروف *صد سال تنهایی* که برندهٔ جایزهٔ نوبل ادبیات شده L1410 است!). مسلماً ترجمهٔ چنین کتابی به زبان فارسی با چالش‌هایی همراه بود؛ استفادهٔ مکرر نویسنده از استعاره‌های ملموس، لحن خودمانی و طنزآمیز، به کار بردن شوخی‌هایی که برای مخاطب فارسی‌زبان محسوس نیست و داستان‌سرایی آموزشی‌اش برای انتقال مطالب از آن جمله‌اند. این چالش‌ها سبب شدند در برخی قسمت‌ها مجبور شوم آنچه نویسنده گفته را با حفظ اصل مطلب و وفاداری به سبک نوشتاری او بازنویسی کنم یا از معادل‌های فارسی در جایگزینی طنازی‌های ظریفش بهره ببرم. به هر حال کتاب ترکیبی

است از علم، داستان‌سرایی و خاطره‌گویی دلنشین آقای نیوسل که با صداقتی آشکار در تلاش است خواننده را به درکی از نحوهٔ کار هوش مصنوعی برساند.

نیوسل هر جایی که احساس کند درک مطلب برایتان دشوار است یا حتی گمان کند ممکن است سوالی برای خواننده پیش آمده باشد، رک و راست آن را بیان می‌کند. البته خود او هم اعتراف می‌کند که وقتی بحث وارد جزئیات می‌شود خواننده علناً با ابهامی ملموس مواجه می‌گردد که قسمتی از آن به کلمات و عبارات نه چندان معروف هوش مصنوعی و قسمت دیگرش به سبک نوشتاری نیوسل (تلاش برای گنجاندن چندین جملهٔ کوتاه اما مرتبط با احساسات گوناگون در یک جملهٔ واحد ولی طولانی و منسجم) برمی‌گردد. البته نیوسل نویسندهٔ آکادمیک است و نویسندگان آکادمیک علاقهٔ زیادی به چنین سبک نگارش شیک و پرطمطراقی دارند.

در این ترجمه سعی کرده‌ام تا جای ممکن از معادل‌های فارسی برای کلمات تخصصی استفاده کنم اما هوش مصنوعی حوزه‌ای جدید است و بسیاری از اصطلاحات انگلیسی آن هنوز معادل‌های فارسی رایج ندارند؛ لذا برخی جاها مجبور شده‌ام از همان کلمهٔ انگلیسی استفاده کنم یا معادل آوایی آنچه نویسنده استفاده کرده را به فارسی بازگردانی کنم. از طرف دیگر تلاشم بر این بوده تا جای ممکن اعداد را به همان صورت انگلیسی بیاورم و برای اسامی خاص، املای انگلیسی را درون پارانتز ذکر کنم تا اگر خواننده قصد جست‌وجوی کتاب‌ها و مقالات این افراد را (که اغلب دانشمندان و متخصصان هوش مصنوعی هستند) داشت با مشکل مواجه نشود.

هنگام مطالعهٔ کتاب بارها برایتان اتفاق خواهد افتاد که مطلب را گنگ بیابید و مجبور به دوباره خوانی یا حتی شاید چند باره خوانی آن شوید؛ چنین چیزی با توجه به پیچیدگی نوشتاری و دشواری علمی مطلب در برخی بخش‌های آن کاملاً طبیعی است. مطالعهٔ این کتاب نیازمند پیش‌نیازهای علمی از جمله آشنایی با مفاهیم پایهٔ ریاضی (مانند بردارها و ماتریس‌ها) و برنامه‌نویسی مقدماتی است؛ شاید برخی اصطلاحات نویسنده حتی به گوش دانشجویان هم نخورده باشد! لذا کمی نیاز به پژوهش فردی نیز احساس می‌شود. پیشنهاد من این است که فصل‌های کتاب را به ترتیب از اولی تا هشتمی مطالعه کرده و از فصلی به فصل غیرمتوالی (مثلاً از یک به چهار) نروید؛ فصل‌های ابتدایی برای آشنایی با مفاهیم پایه‌ای و درک تاریخچهٔ هوش مصنوعی ضروری به شمار می‌روند. همچنین فصل چهار (شبکه‌های عصبی) بسیار مهم بوده و مطالعهٔ آن پیش از ورود به مباحث پیشرفته‌تر در اولویت قرار دارد. خود نیوسل اشاره می‌کند که این کتاب را نباید به عنوان منبعی برای حفظ فرمول‌ها، بلکه به عنوان راهنمای شهودی‌سازی هوش مصنوعی خواند؛ موفقیت خواننده در گرو درک چرایی مفاهیم است، نه صرفاً چگونگی آن‌ها.

هدف من از ارائه این ترجمه، فراهم آوردن فرصتی است تا خوانندگان فارسی‌زبان بتوانند با نگاهی آگاهانه‌تر به طرف آینده‌ای که هوش مصنوعی در آن نقش محوری دارد، گام بردارند. با این امید که مطالعهٔ این اثر، دریچه‌ای به دنیای شگفت‌انگیز هوش مصنوعی به رویتان بگشاید، شما را به همراهی در این سفر دعوت می‌کنم.

در صورت وجود هر گونه نظر یا پیشنهاد می‌توانید از طریق مسیرهای ارتباطی زیر با من در تماس باشید:

MansourTaghizadeh0404@gmail.com

mnsr_tg

https://github.com/mansour0303

موفق باشید

فصل اول: مروری بر هوش مصنوعی (بزن بریم...)

هوش مصنوعی سعی دارد ماشین‌ها، معمولاً یک کامپیوتر، را به گونه‌ای به رفتار درآورد که انسان‌ها آن را هوشمندانه قضاوت کنند. این عبارت در دهه 1950 توسط دانشمند برجسته کامپیوتر، جان مک‌کارتی (1927-2011) ابداع شد. هدف این فصل روشن کردن مفهوم هوش مصنوعی و ارتباط آن با یادگیری ماشین و یادگیری عمیق است، دو واژه‌ای که ممکن است در سال‌های اخیر شنیده باشید. ما با یک مثال از یادگیری ماشین در عمل شروع خواهیم کرد. به این فصل به عنوان یک دیدگاه کلی از هوش مصنوعی فکر کنید. فصل‌های بعدی بر اساس مفاهیم معرفی شده در اینجا بنا خواهند شد و آن‌ها را بررسی خواهند کرد.

کامپیوترها برای انجام یک وظیفه خاص با ارائه یک دنباله از دستورالعمل‌ها، برنامه‌ریزی می‌شوند. این برنامه شامل یک الگوریتم است، یا به عبارتی، دستورالعملی که برنامه باعث می‌شود کامپیوتر آن را اجرا کند. این روزها واژه الگوریتم به طور مکرر مطرح می‌شود، هرچند که این واژه جدید نیست؛ این واژه تصحیح شده‌ای از نام «الخوارزمی» است که به ریاضیدان ایرانی قرن نهم، محمد ابن موسی الخوارزمی، اشاره دارد. هدیه اصلی او به جهان، ریاضیاتی است که ما آن را جبر می‌نامیم.

بیایید با یک داستان شروع کنیم. تونیا صاحب یک کارخانه موفق سس تند است. دستور تهیه سس تند متعلق به خود تونیا است و او به دقت از آن محافظت می‌کند. این سس مخفی اوست و فقط او فرآیند ساخته شدن آن را درک می‌کند. تونیا برای هر مرحله از فرآیند ساخت سس تند یک کارگر استخدام کرده است. این‌ها کارگران انسانی هستند، اما تونیا با آن‌ها طوری رفتار می‌کند که گویی ماشین هستند، زیرا نگران است که آنها دستور تهیه سس تند او را دزدیده و به سرقت ببرند؛ البته خود تونیا کمی هم بدخلق است. در حقیقت، کارگران زیاد ناراحت نیستند زیرا او به آن‌ها حقوق خوبی می‌دهد و پشت سرش به او می‌خندند.

دستور تهیه سس تونیا یک الگوریتم است؛ مجموعه‌ای از مراحل که باید برای ایجاد سس تند دنبال شود. مجموعه‌ای از دستورالعمل‌ها که تونیا برای گفتن به کارگرانش در مورد چگونگی ساخت سس تند استفاده می‌کند، یک برنامه است. این برنامه الگوریتم را به گونه‌ای تجسم می‌کند که کارگران (ماشین) بتوانند مرحله به مرحله آن را دنبال کنند. تونیا کارگرانش را برنامه‌ریزی کرده است تا الگوریتم او را برای ایجاد سس تند پیاده‌سازی کنند. توالی کار به این صورت است:

دستور پخت (الگوریتم) ← دستورالعمل‌ها (برنامه) ← کارگران (ماشین‌ها)

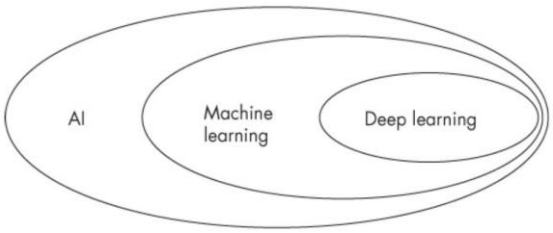
چند نکته در مورد این سناریو وجود دارد که باید به آن‌ها توجه کرد. اولاً، تونیا به طور قطع یک موجود بدخلق است که با انسان‌ها مانند ماشین‌ها رفتار می‌کند 😠 ثانیاً، در هیچ مرحله‌ای از فرآیند ساخت سس تند، هیچ کارگری نیازی ندارد که بفهمد چرا کارهایی را که انجام می‌دهد، انجام می‌دهد. ثالثاً، برنامه‌نویس (تونیا) می‌داند که چرا ماشین (کارگران) آنچه را که انجام می‌دهد، انجام می‌دهد، حتی اگر خود ماشین این را نفهمد.

این الگوریتم به یک دنباله از مراحل (یک برنامه) تبدیل می‌شود. ماشین این برنامه را اجرا می‌کند و بدین ترتیب الگوریتم را پیاده‌سازی می‌کند. ماشین نمی‌فهمد که چه کاری انجام می‌دهد؛ بلکه صرفاً یک سری از دستورات ابتدایی را اجرا می‌کند. نبوغ بابیج (Babbage) و تورینگ (Turing) در این درک نهفته بود که می‌تواند یک ماشین چندمنظوره وجود داشته باشد که قادر

به اجرای الگوریتم‌های دلخواه از طریق برنامه‌ها باشد. با این حال، من استدلال می‌کنم که این آدا لاولیس (Ada Lovelace) بود، دوست بابیج که غالباً به عنوان نخستین برنامه‌نویس جهان شناخته می‌شود، که در ابتدا امکانات وسیع آنچه را که امروزه به آن رایانه می‌گوییم، درک کرد. ما در فصل 2 بیشتر درباره تورینگ، بابیج و لاولیس صحبت خواهیم کرد.

توجه: در زمان لاولیس، «رایانه» یک ماشین نبود بلکه به انسانی اشاره داشت که به صورت دستی محاسبات را انجام می‌داد. از این رو، موتور بابیج یک رایانه مکانیکی بود.

بیایید لحظه‌ای به بررسی رابطهٔ بین اصطلاحات هوش مصنوعی (Artificial Intelligence)، یادگیری ماشین (Machine Learning) و یادگیری عمیق (Deep Learning) بپردازیم. از یک سو، هر سه اصطلاح برای اشاره به هوش مصنوعی مدرن هم‌معنی شده‌اند. این درست نیست، اما راحت است. شکل 1-1 رابطه صحیح بین این اصطلاحات را نشان می‌دهد.



یادگیری عمیق زیرمجموعه‌ای از یادگیری ماشین است که خود نیز زیرمجموعه‌ای از هوش مصنوعی به شمار می‌آید. این رابطه نشان می‌دهد که هوش مصنوعی شامل مفاهیمی است که نه یادگیری ماشین هستند و نه یادگیری عمیق. ما این مفاهیم را هوش مصنوعی قدیمی می‌نامیم که شامل الگوریتم‌ها و رویکردهایی است که از دهه 1950 به بعد توسعه یافته‌اند. هوش مصنوعی قدیمی آن چیزی نیست که مردم در حال حاضر هنگام بحث در مورد هوش مصنوعی به آن اشاره می‌کنند. در ادامه، ما به‌طور کامل (و ناعادلانه) این بخش از جهان هوش مصنوعی را نادیده خواهیم گرفت 😊

یادگیری ماشین مدل‌هایی را از داده‌ها می‌سازد. برای ما، یک مدل یک مفهوم انتزاعی از چیزی است که ورودی‌ها را می‌پذیرد و خروجی‌هایی را تولید می‌کند، جایی که ورودی‌ها و خروجی‌ها به نوعی به هم مرتبط هستند. هدف اصلی یادگیری ماشین این است که یک مدل را با استفاده از داده‌های شناخته شده شرطی کنیم تا مدل، زمانی که داده‌های ناشناخته به آن داده می‌شود، خروجی معناداری تولید کند. وضوح کنونی آنچه بیان شد به اندازهٔ آب گل آلود است، اما با من همراه باشید؛ این گل در زمان مناسب ته نشین خواهد شد.

یادگیری عمیق از مدل‌های بزرگی استفاده می‌کند که در گذشته بیش از حد بزرگ بودند تا مفید واقع شوند. این هم آب گل‌آلود دیگری است، اما من می‌خواهم بگویم که هیچ تعریف دقیقی از یادگیری عمیق وجود ندارد جز اینکه شامل شبکه‌های عصبی با لایه‌های متعدد است. فصل 4 این موضوع را روشن خواهد کرد.

در این کتاب، ما به شکل غیررسمی اما مطابق با استفاده رایج، حتی توسط کارشناسان، از «یادگیری عمیق» به عنوان شبکه‌های عصبی بزرگ (که هنوز به‌طور رسمی تعریف نشده‌اند)، از «یادگیری ماشین» به عنوان مدل‌هایی که با داده‌ها شرطی شده‌اند، و از «هوش مصنوعی» به عنوان یک مفهوم جامع برای هر دوی یادگیری ماشین و یادگیری عمیق استفاده خواهیم کرد - به یاد داشته باشید که هوش مصنوعی شامل موارد بیشتری از آنچه که در اینجا بحث می‌کنیم، است.

داده‌ها در هوش مصنوعی همه‌چیز هستند. نمی‌توانم به اندازه کافی بر این موضوع تأکید کنم. مدل‌ها تخته‌های سفیدی هستند که داده‌ها باید آنها را شرطی کنند تا آنها را برای یک وظیفه مناسب سازند. اگر داده‌ها بد باشند، مدل نیز بد خواهد بود. در طول کتاب، ما به این مفهوم از داده‌های «خوب» و «بد» بازخواهیم گشت.

در حال حاضر، بیایید بر روی این که مدل چیست، چگونه با شرطی‌سازی مفید می‌شود و پس از شرطی‌سازی چگونه استفاده می‌شود، تمرکز کنیم. یک مدل یادگیری ماشین یک جعبه سیاه است که ورودی را قبول می‌کند، معمولاً مجموعه‌ای از اعداد، و خروجی تولید می‌کند، معمولاً یک برچسب (Label) مانند «سگ» یا «گربه»، یا یک مقدار پیوسته مانند احتمال اینکه یک حیوان «سگ» باشد یا ارزش یک خانه با ویژگی‌های (Features) داده شده به مدل (اندازه، تعداد حمام‌ها، کد پستی و غیره) چقدر باشد.

مدل دارای پارامترهایی است که خروجی مدل را کنترل می‌کنند. شرطی‌سازی یک مدل، که به عنوان آموزش شناخته می‌شود، به دنبال تنظیم پارامترهای مدل به گونه‌ای است که خروجی صحیحی را برای یک ورودی خاص تولید کند.

آموزش به این معناست که ما مجموعه‌ای از ورودی‌ها و خروجی‌ها (که مدل باید هنگام دریافت آن ورودی‌ها تولید کند) داریم. در نگاه اول، این به نظر کمی احمقانه می‌آید؛ چرا باید از مدل خروجی‌ای بخواهیم که قبلاً آن را داریم؟ پاسخ این است که در آینده، ورودی‌هایی خواهیم داشت که برای آن‌ها خروجی نداریم. این تمام هدف ساختن مدل است: استفاده از آن با ورودی‌های ناشناخته و اعتماد به مدل، زمانی که خروجی تولید می‌کند. آموزش از مجموعه‌ای از ورودی‌ها و خروجی‌های شناخته‌شده استفاده می‌کند تا پارامترهای مدل را تنظیم کند و اشتباهات را به حداقل برساند. اگر بتوانیم این کار را انجام دهیم، شروع به اعتماد به خروجی‌های مدل می‌کنیم زمانی که ورودی‌های جدید و ناشناخته‌ای به آن داده می‌شود.

آموزش یک مدل به طور اساسی با برنامه‌نویسی متفاوت است. در برنامه‌نویسی، الگوریتمی که می‌خواهیم را با راهنمایی کامپیوتر مرحله به مرحله پیاده‌سازی می‌کنیم. در آموزش، ما از داده‌ها برای آموزش مدل استفاده می‌کنیم تا پارامترهای آن را تنظیم کند و خروجی صحیحی تولید کند. برنامه‌نویسی وجود ندارد زیرا در بیشتر مواقع، هیچ ایده‌ای نداریم که الگوریتم باید چه باشد. ما فقط می‌دانیم یا معتقدیم که رابطه‌ای بین ورودی‌ها و خروجی‌های مورد نظر وجود دارد. امیدواریم که یک مدل بتواند آن رابطه را به اندازه کافی خوب تقریب بزند تا مفید باشد.

به یاد آوردن کلمات حکیمانه آمارشناس بریتانیایی، جورج باکس (George Box)، ارزشمند است که گفت: «همه مدل‌ها اشتباه هستند، اما برخی از آن‌ها مفیدند.» در آن زمان، او به انواع دیگر مدل‌های ریاضی اشاره می‌کرد، اما این حکمت به یادگیری ماشین نیز مربوط می‌شود. حالا می‌فهمیم که چرا این حوزه یادگیری ماشین نامیده می‌شود: ما با ارائه داده‌ها به ماشین (مدل) آموزش می‌دهیم. ما ماشین را برنامه‌نویسی نمی‌کنیم؛ بلکه راهنمایی‌اش می‌کنیم.

پس الگوریتم یادگیری ماشین چنین است:

1. یک مجموعه داده آموزشی جمع‌آوری کنید که شامل مجموعه‌ای از ورودی‌ها به مدل و خروجی‌هایی است که برای آن ورودی‌ها از مدل انتظار داریم.

2. نوع مدلی را که می‌خواهیم آموزش دهیم انتخاب کنید.

3. مدل را با ارائه ورودی‌های آموزشی آموزش دهید و در زمان اشتباه بودن خروجی‌ها، پارامترهای مدل را تنظیم کنید.

4. مرحله 3 را تکرار کنید تا از عملکرد مدل رضایت داشته باشیم.

5. از مدل آموزش‌دیده‌شده برای تولید خروجی‌ها برای ورودی‌های جدید و ناشناخته استفاده کنید.

بیشتر روش‌های یادگیری ماشین از این الگوریتم پیروی می‌کنند. از آنجا که ما از داده‌های برچسب‌گذاری‌شده شناخته‌شده برای آموزش مدل استفاده می‌کنیم، این رویکرد به نام یادگیری تحت نظارت (Supervised Learning) شناخته می‌شود: ما در حین یادگیری مدل برای تولید خروجی صحیح، به آن نظارت می‌کنیم. به نوعی، ما مدل را **مجازات** می‌کنیم تا زمانی که درست عمل کند (*دارک شد* 😅).

حالا برای مواجهه با یک مثال آماده هستیم، اما بیایید ابتدا داستان را تا اینجا خلاصه کنیم. ما به سیستمی نیاز داریم که برای ورودی ناشناخته، خروجی معناداری تولید کند. برای ساختن این سیستم، یک مدل یادگیری ماشین را با استفاده از مجموعه‌ای از ورودی‌ها و خروجی‌های شناخته‌شده آن‌ها آموزش می‌دهیم. آموزش، مدل را با تغییر پارامترهای آن به گونه‌ای تنظیم می‌کند که اشتباهاتش را در داده‌های آموزشی به حداقل برساند. وقتی از عملکرد مدل راضی هستیم، از مدل با ورودی‌های ناشناخته استفاده می‌کنیم، زیرا اکنون به مدل اعتماد داریم که وقتی خروجی ارائه می‌دهد (حداقل، بیشتر اوقات) درست عمل می‌کند.

اولین مثال ما از یک مجموعه داده معروف به دست آمده است که شامل اندازه‌گیری‌های قسمت‌های گل‌های زنبق است. این مجموعه داده به دهه 1930 تعلق دارد و نشان می‌دهد که مردم چه مدت به آنچه اکنون به آن یادگیری ماشین می‌گوییم، فکر کرده‌اند. هدف ما یک مدل است که برای مجموعه ورودی اندازه‌گیری‌ها، گونه خاصی از گل زنبق را خروجی دهد. این مجموعه داده کامل، شامل چهار اندازه‌گیری برای سه گونه زنبق است. ما آن را ساده نگه می‌داریم و از دو اندازه‌گیری و دو گونه استفاده می‌کنیم: طول و عرض گلبرگ به سانتی‌متر (cm) برای I. setosa در مقابلI. versicolor . بنابراین، می‌خواهیم مدل دو اندازه‌گیری را به عنوان ورودی بپذیرد و خروجی‌ای به ما بدهد که بتوانیم آن را به عنوان I. setosa یا I. versicolor تفسیر کنیم. مدل‌های باینری مانند این، بین دو خروجی ممکن تصمیم می‌گیرند و در هوش مصنوعی رایج هستند. اگر مدل بین بیش از دو دسته تصمیم بگیرد، این یک مدل چندکلاسه (Multiclass) است. ما 100 نمونه در مجموعه داده خود داریم: 100 جفت اندازه‌گیری گلبرگ و انواع گل زنبق مربوطه. ما I. setosa را کلاس صفر و I. versicolor را کلاس یک می‌نامیم، جایی که کلاس برچسب‌ها دسته‌های ورودی هستند.

مدل‌ها معمولاً به برچسب‌های کلاسی عددی نیاز دارند که به ما می‌گوید مدل‌ها نمی‌دانند ورودی‌ها و خروجی‌هایشان به چه معنا هستند؛ آن‌ها فقط بین مجموعه‌های ورودی و خروجی ارتباط برقرار می‌کنند. مدل‌ها با هیچ تعریف پذیرفته‌شده‌ای از کلمه، «تفکر» نمی‌کنند. (مدل‌های فصل 7 ممکن است نظر دیگری داشته باشند، اما در مورد آن بعداً صحبت خواهیم کرد.)

در اینجا باید لحظه‌ای توقف کنیم تا برخی از اصطلاحات حیاتی را معرفی کنیم. می‌دانم که این چیزی نیست که شما بخواهید بخوانید، اما برای تمامی مطالبی که در ادامه می‌آید، ضروری است. هوش مصنوعی به طور مکرر از بردارها و ماتریس‌ها استفاده می‌کند. یک بردار رشته‌ای از اعداد است که به عنوان یک موجودیت واحد در نظر گرفته می‌شود. به عنوان مثال، چهار اندازه‌گیری هر گل زنبق به این معناست که می‌توانیم گل را به عنوان یک رشته از چهار عدد نمایش دهیم، مانند (4.5، 2.3، 1.3، 0.3). گلی که توسط این بردار توصیف می‌شود، طول کاسبرگ 4.5 سانتی‌متر، عرض کاسبرگ 2.3 سانتی‌متر، طول گلبرگ 1.3 سانتی‌متر و عرض گلبرگ 0.3 سانتی‌متر دارد. با گروه‌بندی این اندازه‌گیری‌ها با هم، می‌توانیم به آن‌ها به عنوان یک موجودیت واحد اشاره کنیم.

تعداد عناصر در یک بردار بعد آن را تعیین می‌کند؛ به عنوان مثال، مجموعه داده زنبق از بردارهای چهاربُعدی استفاده می‌کند، یعنی چهار اندازه‌گیری گل. هوش مصنوعی اغلب با ورودی‌هایی کار می‌کند که دارای صدها یا حتی هزاران بعد هستند. اگر ورودی یک تصویر باشد، هر پیکسل آن تصویر یک بعد است، به این معنی که یک تصویر کوچک 28 پیکسل مربعی به یک بردار ورودی 28×28 یا 784 بعدی تبدیل می‌شود. مفهوم در سه بعد یا 33000 بعد همان است: همچنان یک رشته از اعداد است که به عنوان یک موجودیت واحد در نظر گرفته می‌شود. اما یک تصویر دارای سطرها و ستون‌ها است که آن را به یک آرایه دو بعدی از اعداد تبدیل می‌کند، نه یک رشته. آرایه‌های دو بعدی از اعداد ماتریس‌ها هستند. در یادگیری ماشین، ما اغلب مجموعه‌های داده را به صورت ماتریس‌ها نمایش می‌دهیم، جایی که سطرها بردارهایی هستند که عناصر مجموعه داده را نمایندگی می‌کنند، مانند یک گل زنبق، و ستون‌ها اندازه‌گیری‌ها هستند. به عنوان مثال، پنج گل اول در مجموعه داده زنبق ماتریس زیر را تشکیل می‌دهند:

$$\begin{bmatrix} 4.5 & 2.3 & 1.3 & 0.3 \\ 5.6 & 2.9 & 3.6 & 1.3 \\ 5.7 & 4.4 & 1.5 & 0.4 \\ 6.7 & 3.1 & 4.4 & 1.4 \\ 4.6 & 3.1 & 1.5 & 0.2 \end{bmatrix}$$

هر سطر مربوط به یک گل است. توجه داشته باشید که سطر اول با مثال بردار مطابقت دارد. سطرهای باقی‌مانده اندازه‌گیری‌های سایر گل‌ها را فهرست می‌کنند. در حین خواندن، این افکار را در ذهن خود نگه دارید:

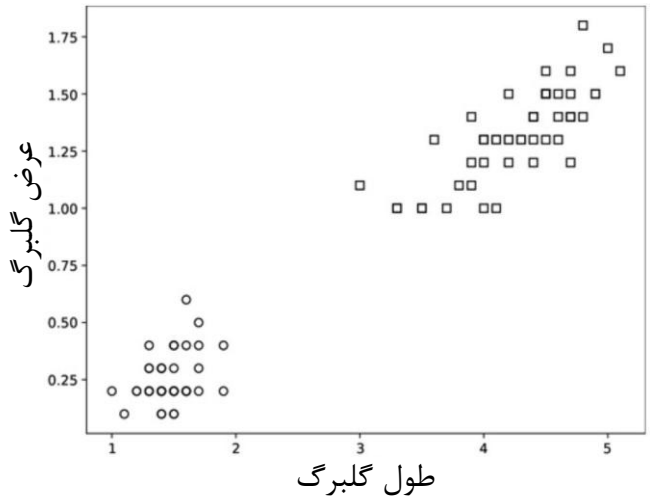
- بردارها رشته‌هایی از اعداد هستند که اغلب اندازه‌گیری‌ها را در یک مجموعه داده نمایندگی می‌کنند.
- ماتریس‌ها آرایه‌های دو بعدی از اعداد هستند که غالباً مجموعه‌های داده (دسته‌های بردارها) را نمایندگی می‌کنند.

در حین ادامه کاوش در زمینه هوش مصنوعی، تفاوت‌های بین بردارها و ماتریس‌ها به وضوح نمایان خواهد شد. حال، بیایید به داستان خود بازگردیم.

ورودی‌های یک مدل ویژگی‌های آن هستند. مجموعه داده گل زنبق ما دارای دو ویژگی، طول و عرض گلبرگ است که به گروه‌های برداری ویژگی (یا نمونه‌ها) تقسیم‌بندی شده‌اند. یک بردار ویژگی واحد به عنوان ورودی مدل عمل می‌کند. خروجی یک مدل دوتایی معمولاً عددی است که به تصمیم مدل اشاره دارد. برای مثال ما، بردار ویژگی‌ای متشکل از دو ویژگی به مدل ارائه می‌دهیم و انتظار داریم خروجی‌ای دریافت کنیم که به ما اجازه دهد تصمیم بگیریم آیا باید ورودی را I. versicolor بنامیم یا نه. اگر نه، ورودی را I. setosa اعلام می‌کنیم زیرا فرض می‌کنیم ورودی‌ها همیشه یکی از این دو خواهد بود.

آداب یادگیری ماشین بیان می‌کند که باید مدل خود را آزمایش کنیم؛ در غیر این صورت، چگونه می‌دانیم که مدل کار می‌کند؟ ممکن است وقتی مدل تمام نمونه‌های آموزشی را درست پیش‌بینی می‌کند فکر کنید خوب کار می‌کند، اما تجربه به کارشناسان آموخته است که همیشه این‌گونه نیست. روش صحیح برای آزمایش یک مدل این است که برخی از داده‌های آموزشی برچسب‌گذاری شده را برای استفاده پس از آموزش نگه داریم. عملکرد مدل بر روی این مجموعه داده آزمایشی که نگه داشته شده است، بهتر نشان می‌دهد که مدل چقدر خوب یاد گرفته است. ما از 80 نمونه برچسب‌گذاری شده برای آموزش استفاده خواهیم کرد و 20 عدد از آن‌ها را برای آزمایش نگه می‌داریم، به طوری که اطمینان حاصل کنیم هر دو مجموعه آموزشی و آزمایشی شامل ترکیبی تقریباً برابر از هر دو کلاس (نوع گل) هستند. این در عمل نیز ضروری است، تا حد امکان. اگر هرگز به مدل نمونه‌هایی از یک کلاس خاص ورودی نشان ندهیم، چگونه می‌تواند آن کلاس را از سایر کلاس‌ها تشخیص دهد؟

استفاده از یک مجموعه آزمایشی نگه‌داشته شده برای قضاوت درباره عملکرد یک مدل فقط جزئی از آداب نیست. این موضوع به یک مسئله بنیادی در یادگیری ماشین می‌پردازد: تعمیم‌پذیری (Generalization). برخی از مدل‌های یادگیری ماشین فرایندی مشابه با رویکردی که به‌طور گسترده‌ای به‌عنوان بهینه‌سازی (Optimization) شناخته می‌شود، دنبال می‌کنند. دانشمندان و مهندسان از بهینه‌سازی برای تطبیق داده‌های اندازه‌گیری شده با توابع شناخته شده استفاده می‌کنند؛ مدل‌های یادگیری ماشین نیز از بهینه‌سازی برای تعدیل پارامترهای خود استفاده می‌کنند، اما هدف متفاوت است. تطبیق داده‌ها با یک تابع، مانند یک خط، به دنبال ایجاد بهترین تطابق ممکن است، یا خطی که بهترین توضیح را برای داده‌های اندازه‌گیری شده ارائه می‌دهد. در یادگیری ماشین، ما به جای آن به مدلی نیاز داریم که ویژگی‌های عمومی داده‌های آموزشی را یاد بگیرد تا بتواند به داده‌های جدید تعمیم یابد. به همین دلیل است که ما مدل را با مجموعه آزمایشی نگه‌داشته شده ارزیابی می‌کنیم. برای مدل، مجموعه آزمایشی شامل داده‌های جدید و ندیده‌ای است که برای تعدیل پارامترهای خود استفاده نکرده است. عملکرد مدل بر روی مجموعه آزمایشی یک نشانه از قابلیت‌های تعمیم‌پذیری آن است. مثال ما دارای دو ویژگی ورودی است، به این معنی که بردارهای ویژگی دو بعدی هستند. از آنجا که ما دو بعد داریم، می‌توانیم تصمیم بگیریم که یک نمودار از مجموعه داده‌های آموزشی ترسیم کنیم. (اگر دو یا سه ویژگی در یک بردار ویژگی داشته باشیم، می‌توانیم بردارهای ویژگی را ترسیم کنیم. اما بیشتر بردارهای ویژگی دارای صدها تا هزاران ویژگی هستند. نمی‌دانم نظر شما چیست، اما من نمی‌توانم یک فضای هزار بعدی را تجسم کنم 😑) شکل 2-1 داده‌های آموزشی دو بعدی زنبق را نمایش می‌دهد؛ محور x طول گلبرگ و محور y عرض گلبرگ است. دایره‌ها مربوط به موارد I. setosa و مربع‌ها مربوط به I. versicolor هستند. هر دایره یا مربع نمایانگر یک نمونه آموزشی واحد، یعنی طول و عرض گلبرگ برای یک گل خاص است. برای قرار دادن هر نقطه، طول گلبرگ را در محور x پیدا کنید و عرض گلبرگ را در محورy. سپس، از محور x به سمت بالا و از محور y به سمت راست حرکت کنید. جایی که انگشتان شما به هم می‌رسند، نقطه‌ای را که نمایانگر آن گل است نشان می‌دهد. اگر گل I. setosa باشد، نقطه را به صورت دایره‌ای بکشید؛ در غیر این صورت، آن را به صورت مربعی نشان دهید.



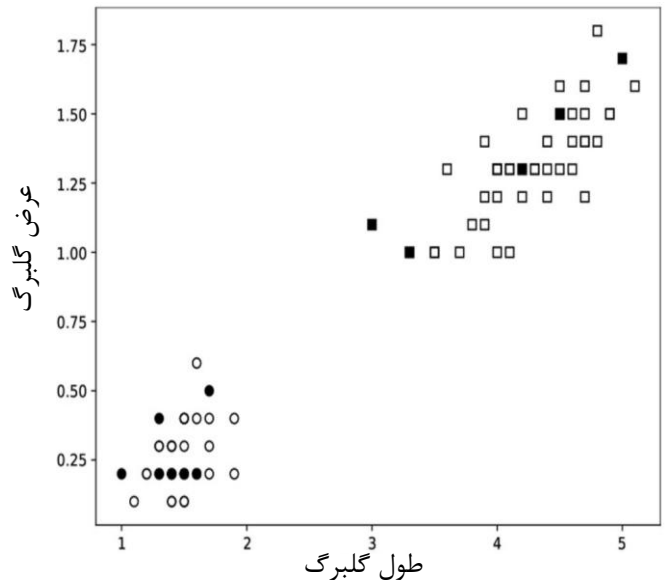
نمودار در شکل 2-1 فضای ویژگی مجموعه آموزشی را نشان می‌دهد. در این مورد، می‌توانیم مجموعه آموزشی را به طور مستقیم تجسم کنیم، زیرا فقط دو ویژگی داریم. وقتی که این امکان وجود ندارد، همه چیز از دست نرفته است. الگوریتم‌های پیشرفته‌ای وجود دارند که به ما اجازه می‌دهند نمودارهایی مانند شکل 2-1 ایجاد کنیم که در آن نقاط در دو یا سه بعد توزیع نمونه‌ها را در فضای با ابعاد بسیار بالاتر منعکس می‌کند. در اینجا، واژه «فضا» به معنای مشابهی است که در زبان روزمره وجود دارد. به دقت به شکل 2-1 نگاه کنید. آیا چیزی توجه شما را جلب می‌کند؟ آیا کلاس‌های مختلف مخلوط شده‌اند یا به خوبی

از هم جدا شده‌اند؟ هر دایره در گوشه پایین سمت چپ نمودار قرار دارد، در حالی که همه مربع‌ها در سمت راست بالا هستند. هیچ تداخلی بین کلاس‌ها وجود ندارد، به این معنی که آن‌ها کاملاً در فضای ویژگی **جدا** هستند. چگونه می‌توانیم از این واقعیت برای ساخت یک طبقه‌بند (Classifier)، مدلی که گل‌های زنبق را طبقه‌بندی می‌کند، استفاده کنیم؟ (در حالی که واژه مدل اصطلاحی کلی‌تر است، زیرا همه مدل‌ها ورودی‌های خود را به دسته‌ها تقسیم نمی‌کنند، زمانی که این کار را انجام می‌دهند، از واژه طبقه‌بند استفاده کنید.)

ما انواع مختلفی از مدل‌ها را جهت انتخاب برای طبقه‌بند خود داریم، از جمله درخت‌های تصمیم که یک سری سوالات بله/خیر مرتبط با ویژگی‌ها را تولید می‌کنند تا برچسب کلاس خروجی را برای یک ورودی مشخص تعیین کنند. زمانی که این سوالات به صورت بصری نمایش داده می‌شوند، ساختاری شبیه به یک درخت معکوس را تشکیل می‌دهند. درخت تصمیم را به عنوان نسخه‌ای تولید شده توسط کامپیوتر از بازی 20 سوالی در نظر بگیرید. اگرچه ما دو ویژگی داریم، طول گلبرگ و عرض گلبرگ، می‌توانیم گل‌های زنبق جدید را با پرسیدن یک سوال ساده طبقه‌بندی کنیم: آیا طول گلبرگ کمتر از 2.5 سانتی‌متر است؟ اگر پاسخ «بله» باشد، سپس کلاس صفر، I. setosa را برمی‌گردانیم؛ وگرنه، کلاس یک، I. versicolor را برمی‌گردانیم. برای طبقه‌بندی صحیح داده‌های آموزشی، تنها به پاسخ این سوال ساده نیاز داریم.

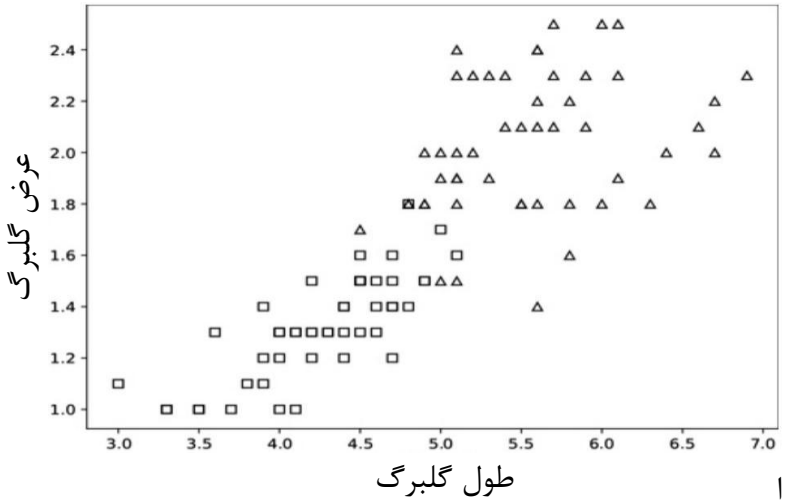
آیا متوجه شدید که چه کار کردم؟ من گفتم که این سوال تمام داده‌های آموزشی را به درستی طبقه‌بندی می‌کند. اما در مورد ۲۰ نمونه آزمایشی که استفاده نکردیم چه؟ آیا طبقه‌بند یک سوالهٔ ما برای دادن برچسب صحیح به هر یک از آن‌ها کافی است؟ در عمل، این همان چیزی است که می‌خواهیم بدانیم و این چیزی است که به عنوان عملکرد طبقه‌بند گزارش خواهیم داد.

شکل 3-1 داده‌های آموزشی را دوباره نشان می‌دهد، به همراه داده‌های آزمایشی که از آن‌ها برای ساخت طبقه‌بند یک سواله خود استفاده نکردیم. دایره‌ها و مربع‌های پررنگ نمایانگر داده‌های آزمایشی هستند.



هیچ‌یک از داده‌های آزمایشی قوانین ما را نقض نمی‌کند؛ هنوز با پرسیدن این سوال که آیا طول گلبرگ کمتر از 2.5 سانتی‌متر است، برچسب‌های کلاس صحیحی دریافت می‌کنیم. بنابراین، مدل ما بی‌نقص است؛ هیچ اشتباهی نمی‌کند. تبریک می‌گوییم، شما اولین مدل یادگیری ماشین خود را ایجاد کردید!

باید خوشحال باشیم، اما نه خیلی خوشحال. بیایید این تمرین را تکرار کنیم و I. setosa را با گونه‌های باقی‌مانده زنبق، یعنی I. virginica جایگزین کنیم.

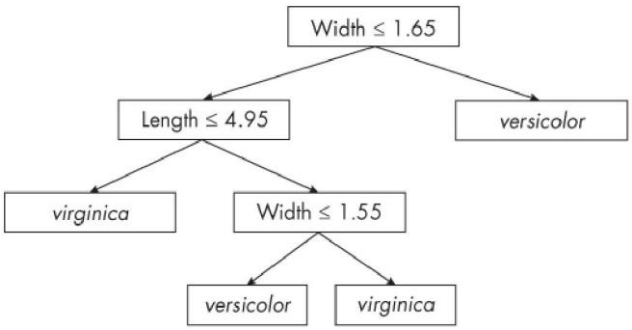


این منجر به شکل 4-1 می‌شود، جایی که مثلث‌ها نمونه‌هایی از I. virginica هستند.

در نمودار جدید همه چیز به وضوح قبل نیست. شکاف واضح بین کلاس‌ها ناپدید شده و هم‌پوشانی دارند.

من یک درخت تصمیم با استفاده از این مجموعه داده جدید زنبق آموزش دادم. مانند قبل، 80 نمونه برای آموزش و 20 نمونه برای آزمایش نگه داشته شد. این بار، مدل بی‌نقص نبود. این مدل به درستی 18 از 20 نمونه را برچسب‌گذاری کرد، که دقت آن 9 از 10 یا 90 درصد بود. این به طور تقریبی به این معناست که وقتی این مدل یک گل را به یک کلاس خاص اختصاص می‌دهد، 90 درصد احتمال دارد که درست باشد. جمله قبلی برای درک بهتر نیاز به توضیح دقیق‌تری دارد، اما فعلاً، این ایده را دریافت کرده‌اید - *مدل‌های یادگیری ماشین همیشه بی‌نقص نیستند؛ آنها (بسیار مکرر) اشتباه می‌کنند.*

شکل 5-1 درخت تصمیم آموخته شده را نشان می‌دهد. از بالا شروع کنید، که ریشه است، و به سوال موجود در آن جعبه پاسخ دهید. اگر پاسخ «بله» است، به جعبه سمت چپ بروید؛ در غیر این صورت، به سمت راست بروید. به همین ترتیب ادامه دهید و پاسخ دهید تا به یک برگ برسید: جعبه‌ای بدون پیکان‌ها. برچسب موجود در این جعبه به ورودی اختصاص داده می‌شود.



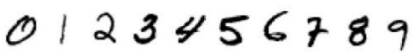
اولین طبقه‌بند درخت تصمیم (Decision Tree Classifier) بسیار ساده بود، زیرا پاسخ به یک سوال واحد برای تعیین عضویت کلاس کافی بود. طبقه‌بند درخت تصمیم دوم رایج‌تر است. بیشتر مدل‌های یادگیری ماشین به‌طور خاص ساده نیستند. اگرچه عملکرد آنها قابل درک است، اما فهمیدن اینکه چرا آنها به گونه‌ای خاص عمل می‌کنند، مسأله‌ای کاملاً متفاوت است. درختان تصمیم از جمله معدود انواع مدل‌هایی هستند که به راحتی می‌توانند خود را توضیح دهند. برای هر ورودی، مسیری که از ریشه تا برگ در شکل 5-1 طی می‌شود، به طور دقیق توضیح می‌دهد که چرا ورودی برچسب خاصی را دریافت کرده است. شبکه‌های عصبی هوش مصنوعی مدرن به این اندازه شفاف نیستند.

برای اینکه یک مدل در دنیای واقعی به خوبی عمل کند، داده‌های استفاده شده برای آموزش مدل باید تمام دامنه ورودی‌هایی را که مدل ممکن است با آن‌ها مواجه شود، پوشش دهد. به عنوان مثال، فرض کنید می‌خواهیم مدلی داشته باشیم که تصاویر

سگ‌ها را شناسایی کند و مجموعه آموزشی ما تنها شامل تصاویر سگ‌ها و طوطی‌ها باشد. در حالی که مدل در مجموعه آزمایش نگه‌داشته شده ما که شامل تصاویر سگ‌ها و طوطی‌ها است، به خوبی عمل می‌کند، اما وقتی مدل را مستقر می‌کنیم و با تصویری از یک گرگ مواجه می‌شود، چه اتفاقی ممکن است بیفتد؟ به طور شهودی، ممکن است انتظار داشته باشیم که مدل بگوید «این یک سگ است»، همان‌طور که یک کودک کوچک پیش از آنکه یاد بگیرد گرگ چیست، ممکن است بگوید.

این دقیقاً همان چیزی است که بیشتر مدل‌های یادگیری ماشین انجام می‌دهند. برای نشان دادن این موضوع، بیایید یک آزمایش انجام دهیم. مجموعه داده‌ای که همه محققان هوش مصنوعی از آن استفاده می‌کنند، شامل ده‌ها هزار تصویر کوچک حاوی اعداد دست‌نویس از 0 تا 9 است. این مجموعه با نام MNIST (Modified NIST) شناخته می‌شود، زیرا در اواخر دهه 1990 از یک مجموعه داده که توسط موسسه ملی استانداردها و فناوری (NIST) ساخته شده بود، استخراج شده است. این موسسه بخشی از وزارت بازرگانی ایالات متحده است که وظیفه پیاده‌سازی انواع استانداردها برای تقریباً همه چیز در حوزه تجاری و صنعتی را بر عهده دارد. شکل 1-6 نمونه‌هایی از اعداد MNIST را ارائه می‌دهد. هدف ما ساخت یک شبکهٔ عصبی است که یاد بگیرد اعداد 0، 1، 3 و 9 را شناسایی کند.

ما می‌توانیم به دلیل وجود ابزارهای قدرتمند و منبع بازی مانند scikit-learn که برای همه در دسترس است، شبکه‌های عصبی را بدون دانستن نحوه کار آن‌ها آموزش دهیم. از یک سو، این امر هوش مصنوعی را دموکراتیک می‌کند؛ و از سوی دیگر، دانش ناقص معمولاً می‌تواند خطرناک باشد. مدل‌ها ممکن است خوب به نظر برسند در حالی که در واقعیت معیوب هستند و عدم آگاهی از نحوه کار مدل‌ها ممکن است ما را از درک این واقعیت قبل از اینکه خیلی دیر شود، بازدارد.



پس از اینکه طبقه‌بند آموزش دیده شد، چند چالش برای آن ایجاد خواهیم کرد و تصاویری از اعداد چهار و هفت به آن خواهیم داد - ورودی‌هایی که هوش مصنوعی در حین آموزش هرگز آنها را ندیده. مدل ممکن است با چنین ورودی‌هایی چه کار کند؟

من مدل اعداد را با استفاده از یک کیت ابزار متن باز آموزش دادم.

در حال حاضر، تنها چیزی که باید در مورد مجموعه داده بدانیم این است که بردارهای ویژگی ورودی، تصاویر اعداد باز شده (Unraveled) هستند؛ اولین ردیف از پیکسل‌ها با ردیف دوم دنبال می‌شود، سپس ردیف سوم و به همین ترتیب، تا اینکه کل تصویر به یک بردار طولانی تبدیل شود، یک رشته از اعداد. تصاویر اعداد 28×28 پیکسل هستند که بردار ویژگی را به طول 784 عدد تبدیل می‌کند. ما از شبکه عصبی می‌خواهیم تا در مورد اشیاء در یک فضای 784 بعدی یاد بگیرد، نه در فضای ساده 2 بعدی که قبلاً استفاده کردیم، اما یادگیری ماشین برای این چالش آماده است.

مجموعه آموزشی که برای آموزش مدل شبکهٔ عصبی استفاده شد، شامل 24745 نمونه بود، که به طور تقریبی 6000 نمونه از هر نوع رقم (3، 1، 0 و 9) را شامل می‌شد. این احتمالاً به اندازه کافی برای نمایندگی منصفانه انواع اعدادی که مدل ممکن است هنگام استفاده با آن‌ها مواجه شود، کافی است، اما برای دانستن این موضوع باید آن را آزمایش کنیم. هوش مصنوعی عمدتاً یک علم تجربی است.

مجموعه آزمایشی‌ای که به طور جداگانه نگه‌داری شده و همچنین شامل اعداد 0، 1، 3 و 9 می‌باشد، دارای 4134 نمونه (حدود 1000 برای هر عدد) بوده است. ما از یک ماتریس سردرگمی یا در هم ریختگی (Confusion Matrix)، که یک جدول دو بعدی

از اعداد است، برای ارزیابی مدل استفاده خواهیم کرد. ماتریس‌های سردرگمی رایج‌ترین روش برای ارزیابی یک مدل هستند زیرا نشان می‌دهند که مدل چگونه در داده‌های آزمایشی عمل می‌کند.

در این مورد، ماتریس سردرگمی برای طبقه‌بند عددی ما در جدول 1-1 نشان داده شده است.

	0	1	3	9
0	978	0	1	1
1	2	1128	3	2
3	5	0	997	8
9	5	1	8	995

سطرهای ماتریس نمایانگر برچسب‌های واقعی برای نمونه‌های ارائه شده به مدل هستند. ستون‌ها پاسخ‌های مدل را نشان می‌دهند. مقادیر موجود در جدول به شمارش‌ها مربوط می‌شوند، یعنی تعداد دفعاتی که هر ترکیب ممکن از کلاس ورودی و برچسب اختصاص داده شده توسط مدل رخ داده است.

به عنوان مثال، سطر اول نمایانگر صفرها در مجموعه آزمون است. از آن 980 ورودی، مدل برای 978 مورد برچسب صفر را بازگرداند، اما یک بار گفت ورودی سه است و یک بار دیگر گفت ورودی نُه است. بنابراین، زمانی که صفر ورودی بود، خروجی مدل، 978 بار از 980 بار درست بود. این موضوع امیدوارکننده است.

به طور مشابه، زمانی که ورودی یک بود، مدل برچسب درست را 1128 بار بازگرداند. برای سه‌ها 997 بار و برای نُه‌ها 995 بار درست بود. زمانی که یک طبقه‌بند خوب عمل می‌کند، اعداد در امتداد قطر ماتریس سردرگمی از بالا سمت چپ تا پایین سمت راست، دارای مقدار بزرگی هستند و تقریباً هیچ عددی خارج از آن قطر وجود ندارد. اعداد خارج از قطر، خطاهایی هستند که توسط مدل ایجاد شده‌اند.

به طور کلی، مدل اعداد 99 درصد دقت دارد. ما یک مدل قوی و با عملکرد خوب داریم - به شرطی که بتوانیم اطمینان حاصل کنیم که تمام ورودی‌های مدل واقعاً 0، 1، 3 یا 9 هستند. اما اگر اینگونه نباشند، چه؟

من 982 تا چهار را به مدل دادم. مدل به این صورت پاسخ داد:

0	1	3	9
48	9	8	917

به عبارت دیگر مدل برای 917 مورد از 982 چهار، برچسب 9 را بازگرداند، برای 48 تا چهار، برچسب 1 را و برای بقیه، برچسب‌های 1 یا 3 را ارائه داد. دربارهٔ هفت‌ها چطور؟

0	1	3	9
19	20	227	762

مدل هنوز هم تمایل داشت که هفت‌ها را نُه بنامد، اما گاهی اوقات آن‌ها را سه نیز می‌نامید. شبکه‌های عصبی به سختی حاضرند رازهای خود را در توضیح اقداماتشان فاش کنند، اما در این مورد، از 227 تا هفت که به عنوان سه برچسب‌گذاری شده بودند،

47 مورد از آن‌ها هفت‌هایی به سبک اروپایی با خط مایل (7) بودند. یک نمونه تصادفی از 227 تا هفت در کل داده‌ها تنها 24 هفت به سبک اروپایی را نشان داد. این مقایسه به‌طور ریاضی سخت نیست، اما نشان می‌دهد که هفت‌های دارای خط مایل اغلب به قدری به یک سه نزدیک هستند که مدل را فریب می‌دهند.

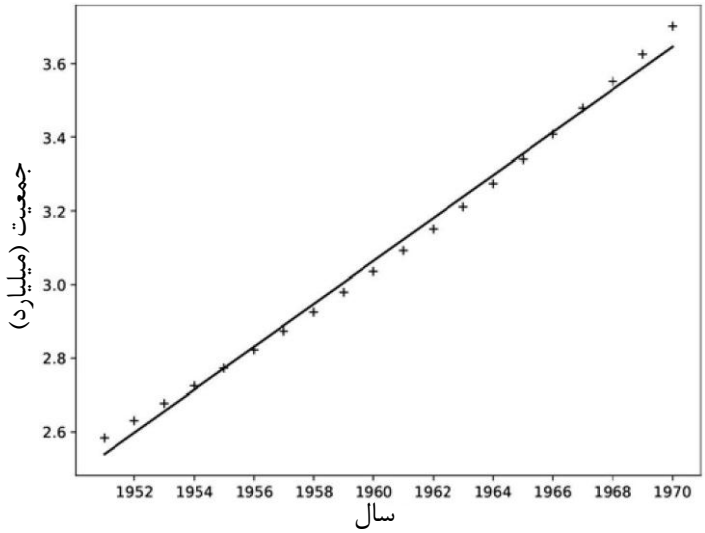
مدل هرگز آموزش ندیده بود که چهارها یا هفت‌ها را تشخیص دهد، بنابراین بهترین کار بعدی را انجام داد و آن‌ها را در یک دسته نزدیک قرار داد. به عنوان مثال، بسته به اینکه اعداد چگونه نوشته شده‌اند، مردم ممکن است گاهی چهارها و هفت‌ها را با نُه اشتباه بگیرند. مدل در حال ارتکاب اشتباهاتی است که مردم انجام می‌دهند، که این موضوع جالب است - اما به‌طور مهم‌تر، مدل ضعیف است زیرا بر روی دامنه کامل ورودی‌هایی که ممکن است با آن‌ها مواجه شود، آموزش ندیده است. این مدل هیچ راهی برای گفتن «نمی‌دانم» ندارد و به دست آوردن مدلی که به‌طور قابل اعتمادی این را بگوید می‌تواند دشوار باشد.

این یک تمرین ساده است، اما پیامدهای آن عمیق است. به جای اعداد، فرض کنید که مدل در حال جستجوی سرطان در تصاویر پزشکی است، اما هرگز آموزش ندیده است که یک دسته مهم از ضایعه یا تمام اَشکالی که آن ضایعه ممکن است به خود بگیرد را شناسایی کند! یک مجموعه داده که به‌درستی ساخته شده و جامع است ممکن است تفاوت بین زندگی و مرگ را رقم بزند.

ما همچنین می‌توانیم به مثال اعداد از نظر درون‌یابی و برون‌یابی فکر کنیم. درون‌یابی تقریباً در محدوده داده‌های شناخته شده انجام می‌شود و برون‌یابی فراتر از داده‌های شناخته شده می‌رود. در مثال اعداد، درون‌یابی ممکن است به مواجهه با یک صفر کج اشاره کند، در حالی که هیچ‌یک از صفرهای موجود در مجموعه آموزشی به‌طور خاص کج نبودند. مدل باید به نوعی درون‌یابی کند تا به درستی پاسخ دهد. برون‌یابی بیشتر شبیه به دسته‌بندی یک صفر با خط مایل از روی آن است که چیزی در زمان آموزش دیده نشده است. برای درک بهتر این اصطلاحات، بیایید جمعیت جهان را از 1950 تا 2020 مدل‌سازی کنیم.

ابتدا، ما یک خط به داده‌های سال‌های 1950 تا 1970 می‌افزاییم. فیت کردن یک خط نوعی از تطبیق منحنی است؛ آن را همچون پسرعموی کمتر پیشرفته یادگیری ماشین در نظر بگیرید. برای فیت کردن یک خط، دو عدد پیدا کنید: شیب و عرض از مبدأ. شیب به ما می‌گوید که خط چقدر تند است. اگر شیب مثبت باشد، خط به‌طور افزایشی از سمت چپ به راست در محور x گراف حرکت می‌کند. شیب منفی به این معناست که خط به‌طور کاهشی در طول محور x حرکت می‌کند. عرض از مبدأ جایی است که خط با محور y تقاطع پیدا می‌کند.

برای فیت کردن خط، از یک الگوریتم استفاده می‌کنیم تا آن شیب و آن عرض از مبدأ را پیدا کنیم که بهترین توصیف را از داده‌ها (در اینجا، جمعیت جهان از 1950 تا 1970) ارائه دهد. شکل 7-1 نموداری از خط و جمعیت واقعی به تفکیک سال را نشان می‌دهد که با علامت‌های مثبت مشخص شده است. خط از نزدیکی یا از میان بیشتر علامت‌های مثبت عبور می‌کند، بنابراین فیت یا برازندگی، معقول است. توجه داشته باشید که واحد جمعیت میلیارد است.



زمانی که خط را داریم می‌توانیم از شیب و عرض از مبدأ برای تخمین جمعیت در هر سال استفاده کنیم. تخمین برای سال‌های بین 1950 و 1970 درون‌یابی است، زیرا ما از داده‌های آن بازه سال‌ها برای ایجاد خط استفاده کرده‌ایم. اگر جمعیت‌ها را برای سال‌های قبل از 1950 یا بعد از 1970 تخمین بزنیم، در حال برون‌یابی هستیم. جدول 2-1 نتایج ما را در هنگام درون‌یابی نشان می‌دهد.

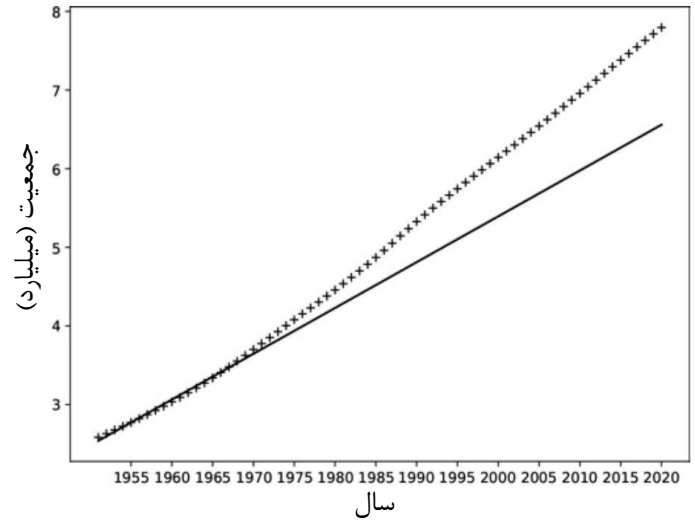
Year	Interpolated	Actual
1954	2.71	2.72
1960	3.06	3.03
1966	3.41	3.41

جدول 2-1

مقدارهای جمعیت درون‌یابی شده به مقدارهای واقعی بسیار نزدیک هستند، به این معنی که مدل (در اینجا خطی که به داده‌ها فیت شده) عملکرد خوبی دارد. اکنون، بیایید به تاریخ‌های خارج از بازه برازندگی برون‌یابی کنیم، همان‌طور که در جدول 3-1 نشان داده شده است.

Year	Extrapolated	Actual
1995	5.10	5.74
2010	5.98	6.96
2020	6.56	7.79

اختلاف بین مقدارهای جمعیت برون‌یابی شده و جمعیت واقعی با هر سال افزایش می‌یابد. مدل عملکرد خوبی ندارد. ترسیم دامنه کامل از 1950 تا 2020 مشکل را نشان می‌دهد؛ به شکل 8-1 مراجعه کنید.



با گذشت زمان، خط برازش به طور فزاینده‌ای نادرست می‌شود زیرا داده‌ها در نهایت خطی نیستند. به عبارت دیگر، نرخ رشد ثابت نیست و خط مستقیم را دنبال نمی‌کند. هنگام برون‌یابی، ممکن است دلیلی برای باور داشته باشیم که داده‌ها به تطابق با خط ادامه خواهند داد؛ اگر این فرض معتبر باشد، در این صورت خط همچنان تطابق خوبی خواهد داشت. با این حال، در دنیای واقعی، **معمولاً** چنین اطمینانی نداریم. بنابراین، به عنوان یک شعار می‌توانیم بگوییم: درون‌یابی خوب است، برون‌یابی بد است.

برازش یک خط به برخی داده‌ها نمونه‌ای از برازش منحنی است. آنچه در مورد برازش منحنی صادق است، در مورد هوش مصنوعی نیز صدق می‌کند. مدل اعداد دست‌نویس زمانی که ورودی‌هایی نزدیک به داده‌هایی که برای شناسایی آنها آموزش دیده بود، دریافت می‌کرد، عملکرد خوبی داشت. اعداد در داده‌های آزمون همه نمونه‌هایی از 0، 1، 3 و 9 بودند، بنابراین داده‌های آزمون شبیه داده‌های آموزشی بودند. دو مجموعه داده از یک توزیع ناشی می‌شوند و همان فرایند تولید داده‌ها هر دو را ایجاد کرده است. بنابراین، می‌توانیم ادعا کنیم که مدل در آن موارد به نوعی در حال درون‌یابی بود. با این حال، وقتی مدل را مجبور کردیم تا درباره اعداد چهار و هفت تصمیم‌گیری کند، ما در حال برون‌یابی بودیم که مدل تصمیماتی درباره داده‌هایی که هرگز در طول آموزش ندیده بود، اتخاذ کند.

تکرار این نکته حائز اهمیت است: درون‌یابی خوب است، برون‌یابی بد است. مجموعه‌های داده بد منجر به مدل‌های بد می‌شوند؛ مجموعه‌های داده خوب منجر به مدل‌های خوب می‌شوند که وقتی مجبور به برون‌یابی می‌شوند، رفتار بدی از خود نشان می‌دهند. و برای تأکید بیشتر: همه مدل‌ها نادرست هستند، اما برخی از آنها مفیدند.

اجازه دهید خاطره‌ای برایتان تعریف کنم 😊 در سال 2016، من در یک کنفرانس شرکت کردم که در آن ارائه‌دهنده‌ای به بررسی تحقیقاتی پرداخت که به دنبال درک دلیل انتخاب‌های یک شبکه عصبی بود. این موضوع هنوز به‌طور کامل حل نشده است، اما پیشرفت‌هایی حاصل شده است. در این مورد، تحقیق بخش‌هایی از تصاویر ورودی را که بر تصمیم‌گیری مدل تأثیر می‌گذارد، علامت‌گذاری کرد. سخنران تصاویری از هاسکی‌ها و گرگ‌ها را نمایش داد و درباره طبقه‌بندیش برای تمایز بین این دو صحبت کرد. او نشان داد که مدلش چگونه در یک مجموعه تست عمل می‌کند و از حضار (پژوهشگران یادگیری ماشین) پرسید که آیا این یک مدل خوب است. بسیاری از افراد پاسخ مثبت دادند، اما با تردید زیرا انتظار یک تله را داشتند. آنها حق داشتند که تردید داشته باشند. سپس سخنران تصاویر را علامت‌گذاری کرد تا بخش‌هایی را نشان دهد که شبکه عصبی هنگام اتخاذ تصمیماتش به آنها توجه کرده است. مدل به سگ‌ها یا گرگ‌ها توجهی نداشت. در عوض، مدل متوجه شد که تمام تصاویر آموزشی گرگ‌ها دارای برف در پس‌زمینه هستند، در حالی که هیچ‌یک از تصاویر سگ‌ها شامل برف نبودند. مدل هیچ چیزی درباره سگ‌ها و گرگ‌ها یاد نگرفت، بلکه فقط درباره وجود برف و عدم وجود برف اطلاعات کسب کرد. پذیرش بی‌دقت رفتار مدل، این واقعیت را آشکار نمی‌کرد و ممکن بود مدل تنها برای شکست در محیط واقعی پیاده‌سازی شود.

داستان مشابهی در مورد یک سیستم یادگیری ماشین بسیار اولیه از دهه 1950 یا 1960 وجود دارد. این داستان احتمالاً افسانه‌ای است، هرچند که من مقاله‌ای از آن دوره خوانده‌ام که ممکن است منبع این افسانه باشد. در این مورد، تصاویری تهیه شده بود که برخی از آنها شامل یک تانک بودند در حالی که بقیه نبودند. مدلی که برای شناسایی تانک‌ها آموزش دیده بود، به نظر می‌رسید که در داده‌های آموزشی به خوبی عمل می‌کند، اما در عمل در محیط واقعی به شدت شکست خورد. در نهایت مشخص شد که یک مجموعه از تصاویر آموزشی در روز آفتابی و مجموعه دیگر در یک روز ابری گرفته شده است. این مدل هیچ چیزی که خالقان آن در نظر داشتند یاد نگرفته بود.

نمونه‌های جدیدتر از این پدیده با مدل‌های یادگیری ماشین پیشرفته‌تر وجود دارد. برخی حتی متخصصان را فریب داده‌اند تا باور کنند که مدل چیزی بنیادی درباره زبان یا موارد مشابه یاد گرفته است، در حالی که در واقع، فقط همبستگی‌های بسیار ظریفی در داده‌های آموزشی یاد گرفته بود که هیچ انسانی نمی‌توانست (به راحتی) تشخیص دهد.

واژه همبستگی یک معنای ریاضی دقیق دارد، اما ما جوهره آن را با عبارت «همبستگی به معنای علیت نیست» منتقل می‌کنیم. همبستگی زمانی است که دو چیز به هم مرتبط هستند به طوری که وقوع یکی، وقوع دیگری را به همراه دارد، اغلب در یک ترتیب خاص. به طور مشخص‌تر، همبستگی اندازه‌گیری می‌کند که چقدر یک تغییر در چیزی با یک تغییر در چیز دیگر مرتبط است. اگر هر دو افزایش یابند، همبستگی مثبت است. اگر یکی افزایش یابد در حالی که دیگری کاهش یابد، همبستگی منفی است.

به عنوان مثال، یک خروس می‌خواند و خورشید بالا می‌آید. این دو رویداد وابسته به زمان هستند: اول خروس، سپس خورشید. این همبستگی به معنای علیت نیست، زیرا خواندن خروس باعث نمی‌شود که خورشید بالا بیاید، اما اگر این همبستگی به اندازه کافی مشاهده شود، ذهن انسان شروع به دیدن یکی به عنوان علت دیگری می‌کند، حتی زمانی که هیچ شواهد واقعی از این موضوع وجود ندارد. چرا انسان‌ها این‌گونه عمل می‌کنند؟ چندان دشوار نیست که درک کنیم. تکامل انسان‌های اولیه‌ای را که این ارتباطات را برقرار می‌کردند، مورد حمایت قرار داد، زیرا گاهی اوقات این ارتباطات منجر به رفتارهایی می‌شد که برای بقای آنها مفید بود. «همبستگی به معنای علیت نیست» همچنین در مورد هوش مصنوعی صدق می‌کند. مدل‌های مذکور یاد گرفتند که چیزهایی را در داده‌های آموزشی شناسایی کنند که با اهداف مورد نظر (سگ‌ها، گرگ‌ها، تانک‌ها) همبسته بودند، اما درباره خود اهداف چیزی یاد نگرفتند. متخصصان هوش مصنوعی باهوش همواره حساس به چنین همبستگی‌های ظاهری‌ای هستند. استفاده از یک مجموعه داده بزرگ و بسیار متنوع برای آموزش و آزمایش می‌تواند در برابر این اثر دفاع کند، هرچند که این در عمل همیشه ممکن نیست.

باید از خود بپرسیم که آیا مدل‌های ما آنچه را که فرض کرده‌ایم یاد گرفته‌اند یا نه. و همانطور که با اعداد MNIST دیدیم، باید اطمینان حاصل کنیم که مدل‌های ما تمام انواع ورودی‌هایی را که در محیط واقعی با آن‌ها مواجه خواهند شد، دیده‌اند - آن‌ها باید درون‌یابی کنند، نه برون‌یابی. این موضوع بیشتر از آنچه که در ابتدا به نظر می‌رسد اهمیت دارد. گوگل این درس را در سال 2015 زمانی که یک ویژگی برای Google Photos راه‌اندازی کرد یاد گرفت، که در آن مدل به‌طور ناکافی بر روی چهره‌های انسانی آموزش دیده بود و ارتباطات نادرست و نامناسبی ایجاد کرد. تعصب، هم در زمینه عمومی و هم اجتماعی، یک مسئله واقعی در هوش مصنوعی است.

بیایید یک آزمایش دیگر با اعداد MNIST انجام دهیم. این بار، مدل تصمیمی ظاهراً ساده برای اتخاذ دارد: آیا رقم ورودی، نه است؟ مدل همان شبکه عصبی است که قبلاً استفاده شده است. اگر بر روی یک مجموعه داده آموزش ببینند که در آن هر تصویر یا نه است یا هر رقم دیگری به جز چهار یا هفت (یعنی هیچ چهار یا هفتی در داده‌های آموزشی وجود ندارد)، آنگاه مدل با دقت 99 درصد عمل می‌کند، همانطور که ماتریس سردرگمی نشان می‌دهد:

	Not 9	9
Not 9	9,754	23
9	38	1,362

ماتریس سردرگمی به ما می‌گوید که مدل به‌درستی 9754 از 9777 تصویر آزمایشی که «نُه» بودند را برچسب‌گذاری کرده است. برچسب مدل همچنین برای 1362 از 1400 عدد نُه نیز صحیح بود. در حالی که مدل در مجموعه آزمایشی عملکرد خوبی دارد، این مجموعه شامل اعداد چهار یا هفت نیست.

در این مورد، ماتریس سردرگمی کوچک است زیرا مدل فقط دو کلاس دارد: نُه یا غیر نُه. به عبارت دیگر، این یک مدل دوتایی است.

عدد 23 در گوشه بالا سمت راست ماتریس نشان‌دهنده 23 باری است که ورودی نُه نبود، اما مدل گفت که نُه است. در یک مدل دوتایی، کلاس 1 معمولاً به‌عنوان کلاس مورد علاقه یا کلاس مثبت در نظر گرفته می‌شود. بنابراین، این 23 ورودی، نمایانگر مثبت‌های کاذب (False Positives) هستند، زیرا مدل گفت «این یک نُه است» در حالی که نبود. به‌طور مشابه، 38 نمونه در گوشه پایین سمت چپ منفی‌های کاذب (False Negatives) هستند زیرا مدل گفت «این نُه نیست» در حالی که ورودی واقعاً نُه بود. ما می‌خواهیم مدل‌هایی داشته باشیم که هیچ مثبت کاذب یا منفی کاذب نداشته باشند، اما گاهی اوقات مهم‌تر است که یکی را به حداقل برسانیم تا دیگری.

به عنوان مثال، اگر یک مدل قرار است سرطان پستان را در ماموگرافی‌ها شناسایی کند، یک مثبت کاذب نمایانگر موردی است که مدل می‌گوید «ممکن است سرطان باشد» حتی اگر نباشد. شنیدن چنین خبری ترسناک است، اما آزمایش‌های بیشتر نشان خواهد داد که مدل اشتباه کرده است. با این حال، یک منفی کاذب نمایانگر یک سرطانِ از دست رفته است. ما ممکن است یک مدل با مثبت‌های کاذب بیشتر را تحمل کنیم اگر تقریباً هیچ منفی کاذبی نداشته باشد، زیرا یک مثبت کاذب از یک منفی کاذب کمتر فاجعه‌آمیز است.

خیلی خوب، به آزمایش خود بازمی‌گردیم. طبقه‌بند «آیا این نُه است» مانند مدل MNIST قبلی، هیچ چیزی درباره اعداد چهار یا هفت نمی‌داند. وقتی اعداد چهار و هفت به مدل MNIST نشان داده می‌شدند، معمولاً آن‌ها را نُه می‌نامید. آیا این مدل هم همین کار را خواهد کرد؟ زمانی که اعداد چهار و هفت را به مدل دادم این نتیجه را دریافت کردم:

	Not 9	9
Not 9	5,014	9,103

مدل 9103 مورد از 14117 تا چهار و هفت را به عنوان نُه علامت‌گذاری کرد. این کمی بیشتر از 65 درصد است؛ یا به طور تقریبی دو از سه. این مشابه حالتی است که ما مدل را با ورودی‌هایی از نوعی که هرگز برای تشخیص آن آموزش ندیده بود، مواجه کردیم.

بیایید به مدل کمک کنیم و چهارها و هفت‌ها را به مجموعه آموزشی اضافه کنیم. امیدواریم که ارائه نمونه‌هایی که می‌گویند: «به نظر می‌رسد نُه است، اما نیست»، که به طور رسمی به آن‌ها منفی‌های کاذب گفته می‌شود، مدل را بهبود بخشد. من 3 درصد از داده‌های آموزشی را به چهارها و هفت‌ها اختصاص دادم. دقت کلی مدل به همان اندازه قبل، یعنی 99 درصد بود، و وقتی چهارها و هفت‌هایی را که هرگز ندیده بود، به آن دادم، چنین اتفاقی افتاد:

	Not 9	9
Not 9	9,385	3,321

این بهتر است. به جای اینکه دو سوم ورودی‌های چهار یا هفت را به عنوان نُه نام‌گذاری کند، مدل تنها یک از چهار ورودی را به عنوان نُه برچسب‌گذاری کرد. حتی چند مثال از چیزهایی که به نظر می‌رسد متعلق به کلاس مثبت هستند اما نیستند، می‌تواند مفید باشد. اگر نسبت چهارها و هفت‌ها را در مجموعه آموزشی به 18 درصد افزایش دهم، مدل کمتر از 1 درصد مواقع چهارها و هفت‌ها را نادرست طبقه‌بندی می‌کند. از آنجایی که مدل‌ها از داده‌ها یاد می‌گیرند، باید از مجموعه‌های داده‌ای استفاده کنیم که تا جای ممکن کامل باشند تا مدل‌ها درون‌یابی کنند و نه برون‌یابی.

توجه: برای اینکه کاملاً دقیق باشیم، تحقیقات اخیر نشان می‌دهد که مدل‌های مدرن یادگیری عمیق تقریباً همیشه در حال برون‌یابی هستند، اما هر چه ورودی‌ها به داده‌هایی که مدل بر اساس آن‌ها آموزش دیده‌اند، بیشتر شباهت داشته باشند، عملکرد بهتری خواهند داشت، بنابراین احساس می‌کنم که استفاده از این تمثیل توجیه‌پذیر است.

هر کسی که به دنبال درک و حتی کار با هوش مصنوعی است، باید هشدارها در مورد کیفیت داده‌های استفاده شده برای آموزش مدل‌های هوش مصنوعی را جدی بگیرد. مقاله پژوهشی منتشر شده در سال 2021 در مجله Nature Machine Intelligence توسط مایکل رابرتس و همکارانش با عنوان «موانع رایج و توصیه‌هایی برای استفاده از یادگیری ماشین برای شناسایی و پیش‌بینی COVID-19 با استفاده از رادیوگرافی قفسه سینه و سی‌تی‌اسکن‌ها» یک مثال تأمل‌برانگیز است. نویسندگان عملکرد مدل‌های یادگیری ماشین طراحی شده برای شناسایی COVID-19 در رادیوگرافی‌های قفسه سینه و سی‌تی‌اسکن‌ها را ارزیابی کردند و تعداد اولیه نامزدها را که بیش از 2000 مطالعه (مدل) بود، به 62 مدل برای آزمایش‌های دقیق کاهش دادند. در نهایت، نویسندگان اعلام کردند که هیچ‌یک از مدل‌ها به دلیل نقص‌های ساختاری، تعصب در داده‌ها یا هر دو برای استفاده بالینی مناسب نیستند.

نتایج این چنینی منجر به ایجاد هوش مصنوعی قابل توضیح (Explainable AI) شده است، زیرشاخه‌ای که به دنبال این است که به مدل‌ها توانایی توضیح خود را بدهد. به داده‌های خود نگاه کنید و سعی کنید تا جایی که ممکن است بفهمید که مدل شما چه کاری انجام می‌دهد و چرا.

آنچه گذشت:

عنوان این فصل، «بزن بریم...»، شعار کمدین، جکی گلیسون (Jackie Gleason) بود. معمولاً خوب است که به یک موضوع بپردازیم تا دیدی کلی به دست آوریم؛ قبل از اینکه به درک عمیق‌تری از آن برگردیم. به عبارت دیگر، ما به سرعت وارد موضوع می‌شویم تا احساس کلی نسبت به آن پیدا کنیم و سپس به طور سیستماتیک‌تر به بررسی آن بپردازیم.

دو دسته نکات کلیدی در این فصل وجود داشت. اولین دسته به این مربوط می‌شد که هوش مصنوعی چیست و اجزای اساسی آن کدامند. دسته دوم درباره به دست آوردن درکی نسبت به آنچه هوش مصنوعی ارائه می‌دهد و اینکه ما چگونه باید پاسخ دهیم، بود.

هوش مصنوعی شامل مدل‌هاست، که هنوز موجودات مبهمی هستند که می‌توانیم آن‌ها را با داده‌ها شرطی کنیم تا یک وظیفه دلخواه را انجام دهند. انواع مختلفی از مدل‌های هوش مصنوعی وجود دارد که این فصل دو نوع آن‌ها را معرفی کرد: درختان تصمیم و شبکه‌های عصبی. من در مورد درختان تصمیم بیشتر صحبت نخواهم کرد، اما شبکه‌های عصبی بیشتر باقی‌مانده کتاب را اشغال می‌کنند.

مدل‌ها معمولاً بهتر است به عنوان توابع در نظر گرفته شوند، مانند توابع ریاضی که ممکن است از مدرسه به یاد داشته باشید یا توابعی که هسته اکثر برنامه‌های کامپیوتری را تشکیل می‌دهند. هر دو می‌توانند به عنوان جعبه‌های سیاه در نظر گرفته شوند، جایی که چیزی وارد می‌شود (ورودی) و چیزی خارج می‌شود (خروجی). در هوش مصنوعی، ورودی یک بردار ویژگی است، مجموعه‌ای از هر چیزی که برای وظیفه مورد نظر مناسب است. در این فصل، ما از دو بردار ویژگی استفاده کردیم: اندازه‌گیری‌های یک گل و تصاویر اعداد دست‌نویس. آموزش، مدل را با تغییر پارامترهای آن شرطی می‌کند تا آن را تا حد امکان دقیق کند. لازم است هنگام آموزش بیشتر مدل‌ها احتیاط کنید تا ویژگی‌های عمومی داده‌ها را یاد بگیرید و نه همبستگی‌های کاذب یا جزئیاتِ ریزِ مجموعه آموزشی (مفهومی که به عنوان overfitting شناخته می‌شود و ما در فصل 4 در مورد آن بحث خواهیم کرد).

توسعه صحیح مدل‌های یادگیری ماشین به این معناست که ما باید یک مجموعه آزمایشی داشته باشیم، مجموعه‌ای از جفت‌های ورودی و خروجی شناخته‌شده که در هنگام آموزش استفاده نمی‌کنیم. ما از این مجموعه بعد از آموزش برای ارزیابی مدل استفاده می‌کنیم. اگر مجموعه داده به درستی ساخته شده باشد، مجموعه آزمایشی ایده‌ای از اینکه ما می‌توانیم انتظار داشته باشیم مدل در دنیای واقعی چگونه عمل کند، ارائه می‌دهد.

نکته بعدی به آنچه هوش مصنوعی ارائه می‌دهد و اینکه چگونه باید به آن پاسخ دهیم مربوط می‌شود. در حالی که هوش مصنوعی قدرتمند است، اما مانند ما فکر نمی‌کند (اگرچه مدل‌های فصل 7 ممکن است با این موضوع مخالف باشند). هوش مصنوعی با داده‌ها زندگی می‌کند و می‌میرد و تنها به اندازه داده‌هایی که به آن می‌دهیم خوب است. اگر مجموعه داده دارای سوگیری باشد، هوش مصنوعی نیز دارای سوگیری خواهد بود. اگر مجموعه داده شامل مثال‌هایی از نوع ورودی‌هایی که هنگام استفاده با آن مواجه خواهد شد، نباشد، هوش مصنوعی در مدیریت صحیح چنین ورودی‌هایی ناکام خواهد ماند. مثال‌های فصل، ما را به احتیاط در فرض اینکه هوش مصنوعی طبق انتظار عمل می‌کند، هشدار می‌دهد. آیا مدل آنچه را که می‌خواستیم بیاموزد، یاد گرفت؟ آیا تحت تأثیر همبستگی‌های موجود در داده‌ها قرار گرفت که ما متوجه آن نشدیم یا بدتر از آن، ما آنقدر محدود هستیم که نتوانیم تشخیص دهیم؟ به مثال هاسکی‌ها و گرگ‌ها فکر کنید. از آنجا که هوش مصنوعی تنها به اندازه داده‌هایی که به آن داده می‌شود خوب است، وظیفهٔ ما است که مجموعه داده‌ها را عادلانه و بدون سوگیری بسازیم و بدون هیچ گونه پیش فرضی بفهمیم که هوش مصنوعی واقعاً چه چیزی را یاد گرفته است.

هوش مصنوعی اولین بار در دهه 1950 ظاهر شد، پس چرا اکنون ناگهان در همه جا به چشم می‌خورد؟ فصل بعدی به این سوال پاسخ می‌دهد.

اصطلاحات کلیدی: الگوریتم (Algorithm)، هوش مصنوعی (Artificial Intelligence)، طبقه‌بند (Classifier)، برچسب کلاس (Class Label)، ماتریس سردرگمی (Confusion Matrix)، مجموعه داده (Dataset)، درخت تصمیم (Decision Tree)، یادگیری عمیق (Deep Learning)، هوش مصنوعی توضیح‌پذیر (Explainable AI)، ویژگی (Feature)، بردار ویژگی (Feature

Vector)، یادگیری ماشین (Machine Learning)، مدل (Model)، شبکه عصبی (Neural Network)، پارامترها (Parameters)، آزمایش (Testing)، آموزش (Training)

فصل دوم: تاریخ هوش مصنوعی (چرا حالا؟)

شاهکار کمدی روآن اتکینسون، مستر بین، نیمه شب در خیابانی خالی از سکنه در لندن آغاز می‌شود. یک نورافکن ظاهر می‌شود، شخصیت اصلی از آسمان سقوط می‌کند و یک گروه کُر به زبان لاتین می‌خواند: "ecce homo qui est faba" یعنی «به مردی که یک لوبیا است، نگاه کنید». مستر بین خود را جمع و جور می‌کند، کت و شلوار خود را می‌تکاند و معذب به سمت تاریکی می‌دود. او چیزی فراتر از دنیای مادی است، چیزی که به معنای واقعی از آسمان سقوط کرده و فراتر از درک ماست.

با توجه به نمایش شگفتی‌های هوش مصنوعی در سال‌های اخیر، ممکن است فکر کنیم هوش مصنوعی، مانند مستر بین، از آسمان سقوط کرده، به طور کامل شکل گرفته و فراتر از درک ماست. اما هیچ کدام از این‌ها درست نیست؛ در واقع، من استدلال می‌کنم که هوش مصنوعی هنوز در مراحل ابتدایی خود است.

پس چرا اکنون درباره هوش مصنوعی می‌شنویم؟ من به این سوال با یک تاریخچه مختصر (و غرض‌آلود) از هوش مصنوعی پاسخ می‌دهم، که به دنبال آن بحثی درباره پیشرفت‌های محاسباتی که به عنوان کاتالیزور انقلاب هوش مصنوعی عمل کرده‌اند، خواهد آمد. این فصل زمینه‌ای برای مدل‌هایی که در ادامه کتاب بررسی خواهیم کرد، فراهم می‌کند.

از زمان تأسیس خود، هوش مصنوعی به دو دسته اصلی تقسیم شده است: هوش مصنوعی نمادین (Symbolic AI) و ارتباط‌گرایی (Connectionism). هوش مصنوعی نمادین سعی دارد تا هوش را با دستکاری نمادها و عبارات یا ارتباطات منطقی مدل‌سازی کند. از سوی دیگر، ارتباط‌گرایی سعی دارد تا هوش را با ساخت شبکه‌هایی از اجزای ساده‌تر مدل‌سازی کند. ذهن انسان هر دو رویکرد را در بر می‌گیرد. ما از نمادها به عنوان عناصر فکر و زبان استفاده می‌کنیم و ذهن‌های ما از شبکه‌های بسیار پیچیده‌ای از نورون‌ها تشکیل شده‌اند، که هر نورون یک پردازنده ساده است. از نظر برنامه‌نویسی کامپیوتر، رویکرد نمادین به هوش مصنوعی از بالا به پایین است، در حالی که ارتباط‌گرایی از پایین به بالا. طراحی از بالا به پایین با وظایف سطح بالا شروع می‌شود و سپس آن وظایف را به قطعات کوچک‌تر و کوچک‌تر تقسیم می‌کند. طراحی از پایین به بالا با قطعات کوچک‌تر شروع می‌شود و آن‌ها را با هم ترکیب می‌کند.

حامیان هوش مصنوعی نمادین معتقدند که هوش می‌تواند به شکل انتزاعی به دست آید، بدون اینکه نیاز به زیرساختی شبیه مغز باشد. ارتباط‌گرایان پیرو توسعه تکاملی مغز هستند و استدلال می‌کنند که باید پایه‌ای وجود داشته باشد، مانند مجموعه‌ای عظیم از نورون‌های به شدت متصل، که از آن هوش ظهور کند.

در حالی که بحث بین هوش مصنوعی نمادین و ارتباط‌گرایی مدت زیادی ادامه داشته است، با ظهور یادگیری عمیق می‌توان گفت که ارتباط‌گرایان پیروز شده‌اند - هرچند شاید نه در جنگ. سال‌های اخیر شاهد انتشار مقالاتی بوده‌ایم که این دو رویکرد را ترکیب می‌کنند. من مشکوکم که هوش مصنوعی نمادین هنوز یک یا دو نقش مهم در پیش دارد، اگرچه ممکن است در نهایت در نقش حمایتی ظاهر شود.

آشنایی من با هوش مصنوعی در اواخر دهه 1980 کاملاً نمادین بود. ارتباط‌گرایی به عنوان یک رویکرد دیگر ذکر شد، اما شبکه‌های عصبی به عنوان روشی ضعیف‌تر و احتمالاً در بهترین حالت با کارایی حاشیه‌ای تصور می‌شدند.

تاریخ نگاری کامل هوش مصنوعی فراتر از توان ماست. چنین اثر بزرگی نیازمند یک تاریخ‌نگار با انگیزه و توانمند است. در عوض، من بر روی توسعه یادگیری ماشین تمرکز خواهم کرد و در عین حال (به طرز بسیار ناعادلانه‌ای!) تلاش‌های فراوانی را که در طول دهه‌ها توسط افراد طرفدار جبههٔ نمادین شده است، نادیده می‌گیرم. با این حال، بدانید که در بیشتر تاریخ هوش مصنوعی، مردم عمدتاً در مورد هوش مصنوعی نمادین صحبت می‌کردند، نه ارتباط‌گرایی. برای ارائه‌ای عادلانه‌تر، کتاب مایکل وولدریج (Michael Wooldridge) با عنوان «تاریخچه‌ای مختصر از هوش مصنوعی» (انتشارات Flatiron، 2021) یا روایت عمیق شخصی پاملا مک‌کوردک (Pamela McCorduck) در «این می‌تواند مهم باشد: زندگی و دوران من با هوش مصنوعی» (انتشارات Lulu، 2019) را توصیه می‌کنم.

با در نظر گرفتن تمایل ظاهری من به ارتباط‌گرایی، بیایید نگاهی به تاریخ یادگیری ماشین بیندازیم.

+ دورهٔ ماقبل 1900

رویای ماشین‌های هوشمند به دوران باستان بازمی‌گردد. یونانیان باستان داستان تالُس، ربات غول‌پیکری که برای محافظت از شاهزاده فنیقی، اروپا (Europa)، طراحی شده بود، را روایت کردند. در طول قرون وسطی و رنسانس، بسیاری از اتومات‌ها، ماشین‌هایی که حرکت می‌کردند و زنده به نظر می‌رسیدند، توسعه یافتند. با این حال، من مشکوک هستم که هیچ‌کدام از آن‌ها هوشمند یا قادر به تفکر شناخته نمی‌شدند. برخی حتی حقه‌هایی بودند، مانند «تُرک مکانیکی» معروف که جهان را با بازی و شکست دادن بسیاری از بازیکنان ماهر شطرنج شگفت‌زده کرد. در نهایت مشخص شد که شخصی درون ماشین مخفی شده بود که می‌توانست «اتومات» را با دستکاری یک بازوی مکانیکی کنترل کند تا مهره‌های شطرنج را روی تخته حرکت دهد در حالی که از زیر به پیکربندی تخته نگاه می‌کرد. با این حال، قسمت مکانیکی ماشین برای اواخر قرن هجدهم به‌طور قابل‌توجهی چشمگیر بود.

علاوه بر اتومات‌ها، تلاش‌های اولیه‌ای نیز برای درک تفکر به‌عنوان یک فرایند مکانیکی و تلاش‌هایی برای تولید یک سیستم منطقی که قادر به ثبت تفکر باشد، وجود داشت. در قرن هفدهم، گوتفرید لایب‌نیتس (Gottfried Leibniz) چنین مفهومی را به‌طور انتزاعی به‌عنوان «الفبای تفکر» توصیف کرد. در دهه 1750، ژولین آفرای د لا متری (Julien Offray de La Mettrie)، کتاب انسان به عنوان ماشین (L’Homme Machine) را منتشر کرد و استدلال نمود که تفکر فرایندی مکانیکی است.

این ایده که تفکر انسانی ممکن است از موجودیت فیزیکی مغز به جای روح معنوی ناشی شود، آغاز یک فصل جدید در مسیر هوش مصنوعی را نشان داد. اگر اذهان ما ماشین‌های بیولوژیکی هستند، چرا نباید نوع دیگری از ماشین وجود داشته باشد که تفکر کند؟

در قرن نوزدهم، جورج بوله (George Boole) تلاش کرد تا یک حساب تفکر ایجاد کند که منجر به آنچه اکنون به‌عنوان جبر بوله می‌شناسیم، شد. رایانه‌ها به جبر بوله وابسته هستند، به‌طوری که این جبر نمایانگر پیاده‌سازی آن‌ها به‌عنوان مجموعه‌ای از دروازه‌های منطقی دیجیتال است. بوله تا حدی موفق شد، اما به هدف اعلام‌شده‌اش نرسید: «برای بررسی قوانین بنیادیِ آن عملیات‌های ذهن که به وسیله آن‌ها استدلال انجام می‌شود؛ برای بیان آن‌ها در زبان نمادین حساب» (قوانین تفکر، 1854). تلاش‌های بوله، نمایانگر یک گام دیگر به سمت این مفهوم بود که هوش مصنوعی ممکن است.

آنچه این تلاش‌های اولیه از آن محروم بود، یک ماشین محاسباتی واقعی بود. مردم می‌توانستند درباره ذهن‌ها یا موجودات مصنوعی (مانند موجودی از رمان فرانکنشتاین مری شلی) رویا ببینند و با فرض وجود آن‌ها، در مورد پیامدها بحث کنند. اما تا زمانی که ماشینی قادر به تقلید معقول (پیاده‌سازی؟) تفکر وجود نداشت، همه چیز حدس و گمان بود.

این چارلز بابیج انگلیسی بود که در اواسط قرن نوزدهم، برای اولین بار به ایده یک ماشین محاسباتی عمومی قابل پیاده‌سازی فکر کرد: موتور تحلیلی. این موتور هرگز به‌طور کامل ساخته نشد، اما تمام اجزای اساسی یک رایانه مدرن را در بر داشت و به‌طور نظری قادر به انجام همان عملیات‌ها بود. در حالی که مشخص نیست آیا بابیج از قابلیت‌های چندمنظوره ماشینش آگاه بود یا نه، دوستش، آدا لاولیس، به خوبی این موضوع را درک کرده بود. او در مورد ماشین به‌عنوان یک دستگاه عمومی و قابل اعمال نوشت. با این حال، او اعتقاد نداشت که موتور قادر به تفکر باشد، همان‌طور که این نقل قول از «طرح موتور تحلیلی» او (1843) نشان می‌دهد:

موتور تحلیلی هیچ ادعایی برای ایجاد چیزی ندارد. این ماشین می‌تواند هر آنچه را که ما می‌دانیم چگونه از آن درخواست کنیم، انجام دهد. می‌تواند تجزیه و تحلیل را دنبال کند؛ اما هیچ قدرتی در پیش‌بینی روابط یا حقایق تحلیلی ندارد. حوزه وظایف آن کمک به ما در در دسترس قرار دادن آنچه است که قبلاً با آن آشنا هستیم.

این نقل قول ممکن است اولین اشاره به امکان هوش مصنوعی باشد که شامل یک دستگاه پتانسیل‌دار برای دستیابی به آن می‌باشد. عبارت «هر آنچه را که ما می‌دانیم چگونه از آن درخواست کنیم انجام دهد» به برنامه‌نویسی اشاره دارد. در واقع، لاولیس یک برنامه برای موتور تحلیلی نوشت. به همین دلیل، بسیاری از مردم او را به عنوان اولین برنامه‌نویس کامپیوتر می‌شناسند. این که برنامه او دارای یک اشکال بود، به من ثابت می‌کند که او واقعاً برنامه‌نویس بود؛ هیچ چیزی به اندازه اشکالات، نمادین‌تر از برنامه‌نویسی نیست، همان‌طور که 40 سال تجربه برنامه‌نویسی من به طرز ناراحت‌کننده‌ای بارها و بارها نشان داده است.

+ از 1900 تا 1950

در سال 1936، یک انگلیسی 24 ساله به نام آلن تورینگ، که در آن زمان هنوز دانشجو بود، مقاله‌ای نوشت که از آن زمان به عنوان سنگ بنای علم کامپیوتر شناخته می‌شود. در این مقاله، تورینگ یک ماشین مفهومی عمومی را معرفی کرد که اکنون آن را ماشین تورینگ می‌نامیم و نشان داد که این ماشین می‌تواند هر چیزی را که قابل نمایش با یک الگوریتم باشد، محاسبه کند. او همچنین توضیح داد که چیزهایی وجود دارند که نمی‌توانند با الگوریتم‌ها پیاده‌سازی شوند و بنابراین، غیرقابل محاسبه هستند. از آنجا که تمام زبان‌های برنامه‌نویسی مدرن معادل یک ماشین تورینگ هستند، کامپیوترهای مدرن می‌توانند هر الگوریتمی را پیاده‌سازی کرده و هر چیزی که **قابل** محاسبه باشد، محاسبه کنند. با این حال، این هیچ چیز در مورد اینکه محاسبات چقدر طول می‌کشد یا چه مقدار حافظه نیاز دارد، نمی‌گوید.

اگر یک کامپیوتر می‌تواند هر چیزی را که می‌تواند به عنوان یک الگوریتم پیاده‌سازی شود، محاسبه کند، پس می‌تواند هر عملیات ذهنی را که یک انسان می‌تواند انجام دهد، پیاده کند. در نهایت، سخن از موتورهایی بود که می‌توانستند هوش مصنوعی واقعی را امکان‌پذیر کند. مقاله تورینگ در سال 1950 با عنوان «ماشین‌های محاسباتی و هوش» این درک زود هنگام را در پی داشت که کامپیوترهای دیجیتال ممکن است در نهایت به ماشین‌های هوشمند منجر شوند. در این مقاله، تورینگ «بازی تقلید» خود را توصیف کرد که اکنون به عنوان آزمون تورینگ شناخته می‌شود و به موجب آن انسان‌ها ممکن است به این باور برسند

که یک ماشین هوشمند است. بسیاری از ادعاها درباره سیستم‌های هوش مصنوعی که آزمون تورینگ را پشت سر گذاشته‌اند، به‌ویژه در سال‌های اخیر، ظاهر شده است. یکی از این‌ها ChatGPT شرکت OpenAI است. با این حال، تعداد کمی مایلند به این باور برسند که ChatGPT واقعاً هوشمند است - به عبارت دیگر، من شک دارم که این آزمون بتواند آنچه را که انسان‌ها به طور کلی از این اصطلاح درک می‌کنند، به خوبی به تصویر بکشد و احتمالاً در یک زمان مشخص آزمون جدیدی ایجاد خواهد شد.

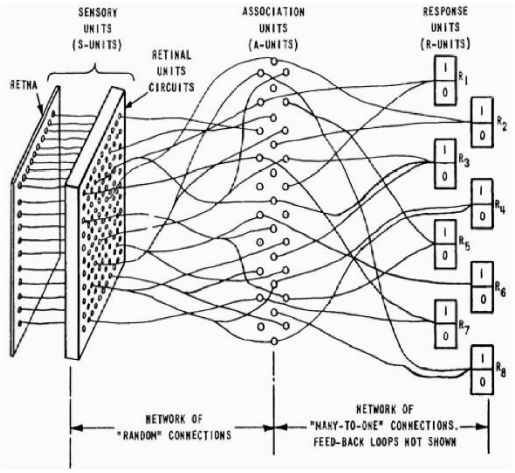
در سال 1943، وارن مک‌کالوک (Warren McCulloch) و والتر پیتس (Walter Pitts) مقاله‌ای با عنوان «حساب منطقی ایده‌های نهفته در فعالیت عصبی» نوشتند که شایسته دریافت جایزه‌ای برای یکی از نامشخص‌ترین اما جالب‌ترین عناوین مقالات است. این مقاله «شبکه‌های عصبی» (مجموعه‌ای از نورون‌ها) را به عنوان بیانیه‌های منطقی در ریاضیات نشان می‌دهد. بیانیه‌های منطقی دشوار است که شفاف شوند (حداقل برای من)، اما توصیف نویسندگان از «شبکه‌های بدون حلقه» شباهت زیادی به شبکه‌های عصبی دارد که در فصل 4 بررسی خواهیم کرد - در واقع، می‌توان گفت که مقاله نوآورانه مک‌کالوک و پیتس به آنچه که اکنون به عنوان یک شبکه عصبی شناخته می‌شود، منجر شد. صادقانه بگویم، شبکه‌های عصبی بسیار آسان‌تر از آنچه که به نظر می‌رسد قابل تجزیه و تحلیل و درک هستند، که این خبر خوبی برای ماست.

پیشرفت از داستان‌های خیالی درباره ماشین‌ها و موجودات هوشمند مصنوعی به تحقیق جدی درباره اینکه آیا ریاضیات می‌تواند تفکر و استدلال را به تصویر بکشد، همراه با درک اینکه کامپیوترهای دیجیتال قادر به محاسبه هر چیزی هستند که می‌تواند با یک الگوریتم توصیف شود، زمینه‌ساز ظهور هوش مصنوعی به عنوان یک تحقیق مشروع بود.

+ از 1950 تا 1970

پروژه تحقیقاتی تابستانی دارتموث (Dartmouth) در سال 1956 به طور کلی به عنوان زادگاه هوش مصنوعی شناخته می‌شود و جایی است که عبارت «هوش مصنوعی» برای اولین بار به طور مداوم استفاده شد. کارگاه دارتموث کمتر از 50 شرکت‌کننده داشت، اما فهرست شامل چندین نام معروف در دنیای علوم کامپیوتر و ریاضیات بود: ری سالومونف (Ray Solomonoff)، جان مک‌کارتی (John McCarthy)، ماروین مینسکی (Marvin Minsky)، کلود شانن (Claude Shannon)، جان نش (John Nash) و وارن مک‌کالوک (Warren McCulloch)، و دیگران. در آن زمان، علوم کامپیوتر یک زیرشاخه از ریاضیات بود. این کارگاه یک جلسه طوفان فکری بود که زمینه را برای تحقیقات اولیه هوش مصنوعی فراهم کرد.

در سال 1957، فرانک روزنبلات (Frank Rosenblatt) از دانشگاه کُرنل، پرسپترون مارک I (Perceptron Mark I) را ایجاد کرد که به طور گسترده به عنوان اولین کاربرد شبکه‌های عصبی شناخته می‌شود. پرسپترون از جنبه‌های بسیاری قابل توجه بود، از جمله اینکه برای شناسایی تصاویر طراحی شده بود، همان کاربردی که یادگیری عمیق برای اولین بار در سال 2012 در آن خود را ثابت کرد.



شکل 1-2 سازماندهی مفهومی را که در راهنمای اپراتورهای پرسپترون ارائه شده نشان می‌دهد. پرسپترون از یک تصویر دیجیتالی تلویزیونی با ابعاد 20×20 پیکسل به عنوان ورودی استفاده کرد که سپس از طریق مجموعه‌ای «تصادفی» از اتصالات به مجموعه‌ای از واحدهای ارتباطی منتقل شد که به واحدهای

پاسخ‌دهنده منتهی می‌شد. این پیکربندی با برخی از رویکردهای یادگیری عمیق در تصاویر که امروز استفاده می‌شود، مشابه است و شبیه نوعی از شبکه عصبی به نام ماشین یادگیری شدید (Extreme Learning Machine) است.

اگر پرسپترون در مسیر درستی بود، چرا برای دهه‌ها تقریباً فراموش شد؟ یکی از دلایل آن تمایل روزنبلات به بزرگ‌نمایی بود. در کنفرانسی که در سال 1958 توسط نیروی دریایی ایالات متحده (یکی از حامیان پروژه پرسپترون) برگزار شد، نظرات روزنبلات آنقدر اغراق‌آمیز بود که روزنامه نیویورک تایمز گزارش داد:

«نیروی دریایی امروز جنین یک کامپیوتر الکترونیکی را فاش کرد که انتظار دارد قادر به راه رفتن، صحبت کردن، دیدن، نوشتن، بازتولید خود و آگاهی از وجودش باشد. پیش‌بینی شده که پرسپترون‌های بعدی قادر خواهند بود افراد را شناسایی کرده و نام آن‌ها را صدا بزنند و به‌طور آنی گفتار را از یک زبان به گفتار و نوشتار در زبان دیگر ترجمه کنند.»

این نظرات در آن زمان بسیاری را ناراحت کرد، هرچند که سیستم‌های هوش مصنوعی مدرن اکنون به ماشین‌ها اجازه می‌دهند که راه بروند، صحبت کنند، ببینند، بنویسند، افراد را شناسایی کنند و گفتار و نوشتار را بین زبان‌ها ترجمه کنند. شاید باید نسبت به روزنبلات کمی بخشنده‌تر باشیم؛ او تنها حدود 60 سال زودتر از زمان خود حرف زده بود.

چند سال بعد، در سال 1963، لئونارد اوهر (Leonard Uhr) و چارلز واسلر (Charles Vossler) برنامه‌ای را توصیف کردند که مانند پرسپترون، یک تصویر 20×20 پیکسلی را که به صورت یک ماتریس از 0ها و 1ها نشان داده شده بود، تفسیر می‌کرد. برخلاف پرسپترون، این برنامه قادر بود الگوها و ترکیب‌های ویژگی‌های تصویری لازم برای یادگیری ورودی‌هایش را تولید کند. برنامه اوهر و واسلر مشابه شبکه‌های عصبی پیچشی (Convolutional) بود که بیش از 30 سال بعد ظاهر شدند و موضوع فصل 5 هستند.

اولین مدل‌های یادگیری ماشین که من آن‌ها را «کلاسیک» می‌نامم، در سال 1967 به لطف توماس کاور (Thomas Cover) و پیتر هارت (Peter Hart) ظاهر شدند. این مدل که به عنوان *نزدیک‌ترین همسایه‌ها* (Nearest Neighbors) شناخته می‌شود، ساده‌ترین مدل یادگیری ماشین است، تقریباً به طرز خجالت‌آوری ساده. برای برچسب‌گذاری یک ورودی ناشناخته، به سادگی نزدیک‌ترین ورودی شناخته شده به آن را پیدا کرده و از برچسب آن ورودی به عنوان خروجی استفاده می‌کند. هنگامی که از بیش از یک ورودی شناخته شده نزدیک استفاده می‌شود، این روش k-نزدیک‌ترین همسایه‌ها (K-Nearest Neighbors) نامیده می‌شود که در آن k عددی کوچک مانند 3 یا 5 است. هارت در سال 1973 اولین ویرایش کتاب «طبقه‌بندی الگو» را با همکاری ریچارد دودا (Richard Duda) و دیوید استورک (David Stork) نوشت؛ این اثر بنیادین بسیاری از دانشمندان کامپیوتر و مهندسان نرم‌افزار را با یادگیری ماشین آشنا کرد، از جمله من.

موفقیت پرسپترون در سال 1969 به طرز ناگهانی متوقف شد، زمانی که ماروین مینسکی و سیمور پاپرت (Seymour Papert) کتاب خود را با عنوان «پرسپترون‌ها» منتشر کردند که نشان می‌داد شبکه‌های پرسپترون تک‌لایه و دو لایه قادر به مدل‌سازی وظایف جالب نیستند. ما به زودی توضیح خواهیم داد که «تک‌لایه» و «دو لایه» چه معنایی دارند. پرسپترون‌ها، به همراه انتشار گزارش «هوش مصنوعی: یک بررسی کلی» توسط جیمز لایت‌هیل (James Lighthill) در سال 1973 که به‌طور جهانی به عنوان «گزارش لایت‌هیل» شناخته می‌شود، دوره‌ای را آغاز کردند که اکنون به عنوان اولین زمستان هوش مصنوعی شناخته می‌شود؛ تأمین مالی برای تحقیقات هوش مصنوعی به سرعت متوقف شد.

انتقادات مینسکی و پاپرت از مدل پرسپترون مشروع بود اما بسیاری این مسئله که چنین محدودیت‌هایی به مدل‌های پرسپترون پیچیده‌تر قابل اعمال نیست را نادیده گرفتند. با این حال، آسیب وارد شده بود و ارتباط‌گرایی تقریباً تا اوایل دهه 1980 ناپدید شد.

+ از 1980 تا 1990

در اوایل دهه 1980، هوش مصنوعی با ظهور کامپیوترهایی که به‌طور خاص برای اجرای زبان برنامه‌نویسی لیسپ (Lisp) طراحی شده بودند، تجاری شد؛ در آن زمان، لیسپ زبان بین‌المللی هوش مصنوعی بود. (امروزه بین‌المللی هوش مصنوعی، پایتون است.) به همراه ماشین‌های لیسپ، ظهور سیستم‌های خبره نیز آغاز شد - نرم‌افزارهایی که برای ضبط دانش یک کارشناس در یک حوزه خاص طراحی شده‌اند. تجاری‌سازی هوش مصنوعی پایان اولین زمستان هوش مصنوعی را به همراه داشت.

مفهوم پشت سیستم‌های خبره، بی‌تردید، وسوسه‌انگیز است. برای ساخت یک سیستم خبره که به عنوان مثال، نوع خاصی از سرطان را تشخیص دهد، ابتدا باید با کارشناسان مصاحبه کنید تا دانش آن‌ها را استخراج کرده و در یک پایگاه دانش سازماندهی کنید. یک پایگاه دانش، دانش را به عنوان ترکیبی از قواعد و واقعیت‌ها نمایان می‌کند. سپس، پایگاه دانش را با یک موتور استنتاج ترکیب می‌کنید، که از پایگاه دانش برای تصمیم‌گیری در مورد زمان و نحوه اجرای قواعد بر اساس واقعیت‌های ذخیره شده یا ورودی‌های کاربر استفاده می‌کند. قواعد بر اساس واقعیت‌ها فعال می‌شوند که ممکن است به قرار دادن واقعیت‌های جدید در پایگاه دانش منجر شود و موجب فعال‌سازی قواعد اضافی گردد و به همین ترتیب ادامه یابد. یک مثال کلاسیک از یک سیستم خبره، CLIPS است که ناسا در سال 1985 توسعه داد و در سال 1996 به صورت عمومی منتشر کرد.

در یک سیستم خبره، هیچ شبکه اتصالی یا مجموعه‌ای از واحدها وجود ندارد که از آن‌ها بتوان (امیدوارانه) رفتار هوشمندانه‌ای به وجود آورد، که این امر آن را به یک مثال خوب از هوش مصنوعی نمادین تبدیل می‌کند. در عوض، پایگاه دانش به‌طور اساسی مجموعه‌ای سخت از قواعد است، مانند «اگر دمای موتور بالاتر از این آستانه باشد، آنگاه چیز دیگری احتمالاً علت آن است» و واقعیت‌هایی مانند «دمای موتور زیر آستانه است.» مهندسان دانش، پیوندهایی بین کارشناسان و سیستم خبره هستند. ساخت یک پایگاه دانش از پاسخ‌های کارشناسان به سؤالات مطرح شده توسط مهندسان دانش پیچیده است و پایگاه دانش حاصل، به مرور زمان تغییرات سختی را متحمل می‌شود. با این حال، دشواری در طراحی سیستم‌های خبره به این معنا نیست که بی‌فایده‌اند؛ آن‌ها هنوز وجود دارند، عمدتاً تحت عنوان «سیستم‌های مدیریت قواعد کسب‌وکار»؛ اما در حال حاضر تأثیر کمی بر هوش مصنوعی مدرن دارند.

هیاهو پیرامون سیستم‌های خبره، همراه با موفقیت‌های اولیه، منجر به احیای علاقه به هوش مصنوعی در اوایل دهه 1980 شد. اما وقتی مشخص شد که سیستم‌های خبره برای استفاده عمومی بسیار شکننده هستند، صنعت به شدت دچار افت شد و زمستان دوم هوش مصنوعی در اواسط این دهه آغاز گردید.

در طول دهه 1980، طرفداران رویکردهای ارتباطی در پس‌زمینه قرار داشتند، اما آن‌ها بی‌تحرک نبودند. در سال 1982، جان هاپفیلد (John Hopfield) آنچه اکنون به‌عنوان شبکه‌های هاپفیلد شناخته می‌شود را نشان داد. یک شبکه هاپفیلد نوعی شبکه عصبی است که اطلاعات را به‌صورت توزیع‌شده در وزن‌های شبکه ذخیره می‌کند و سپس آن اطلاعات را در زمان دیگری استخراج می‌نماید. شبکه‌های هاپفیلد به‌طور گسترده‌ای در یادگیری عمیق مدرن استفاده نمی‌شوند، اما آن‌ها نشان‌دهنده یک نمایش مهم از کاربرد رویکرد ارتباطی بودند.

در سال 1986، دیوید راملهارت (David Rumelhart)، جفری هینتون (Geoffrey Hinton) و رونالد ویلیامز (Ronald Williams) مقاله‌ای با عنوان «نمایش‌های یادگیری از طریق خطاهای پس‌انتشار» منتشر کردند که الگوریتم پس‌انتشار (Backpropagation) را برای آموزش شبکه‌های عصبی تشریح کرد. آموزش یک شبکه عصبی شامل تنظیم وزن‌ها بین نورون‌ها است تا شبکه به طور مطلوب عمل کند. الگوریتم پس‌انتشار کلید بهینه‌سازی این فرآیند بود، زیرا نحوه تأثیرگذاری تنظیم یک وزن خاص بر عملکرد کلی شبکه را محاسبه می‌کرد. با استفاده از این اطلاعات، امکان آموزش تدریجی شبکه با استفاده از داده‌های آموزشی شناخته شده فراهم می‌شود و سپس با استفاده از خطاهای شبکه در هنگام طبقه‌بندی، وزن‌ها تنظیم می‌شوند تا شبکه را مجبور به بهبود عملکرد در تکرار بعدی کند. (در فصل 4 به آموزش شبکه‌های عصبی به‌طور عمیق‌تری پرداخته خواهد شد.) با استفاده از پس‌انتشاری، شبکه‌های عصبی می‌توانستند به مراتب فراتر از عملکرد محدود پرسپترون روزنبلات پیش بروند. با این حال، حتی با وجود پس‌انتشاری، شبکه‌های عصبی در دهه 1980 چیزی بیشتر از اسباب‌بازی نبودند. در حالی که در مورد اینکه چه کسی پس‌انتشاری را اختراع کرده و چه زمانی، اختلاف نظر وجود دارد، مقاله سال 1986 به‌طور کلی به عنوان ارائه‌ای درک می‌شود که بیشترین تأثیر را بر پژوهشگران شبکه‌های عصبی گذاشته است.

+ از 1990 تا 2000

زمستان دوم هوش مصنوعی تا دهه 1990 ادامه داشت، اما تحقیقات در هر دو گروه نمادین و ارتباط‌گرا ادامه یافت. کورینا کورتس (Corinna Cortes) و ولادیمیر واپنیک (Vladimir Vapnik) در سال 1995 جامعه یادگیری ماشین را با ماشین‌های بردار پشتیبان (SVM) آشنا کردند. به نوعی، SVMها بالاترین نقطه یادگیری ماشین کلاسیک را نمایندگی می‌کنند. موفقیت SVMها در دهه 1990 تا اوایل 2000 مانع از پیشرفت شبکه‌های عصبی شد. شبکه‌های عصبی به مجموعه‌های داده بزرگ و قدرت محاسباتی قابل توجهی نیاز دارند؛ در حالی که SVMها معمولاً به منابع کمتری احتیاج دارند. شبکه‌های عصبی قدرت خود را از توانایی شبکه در نمایندگی یک تابع، که نقشه‌ای از ورودی‌ها به خروجی‌های مطلوب است، به دست می‌آورند، در حالی که SVMها با استفاده از ریاضیات هوشمند، مسائل طبقه‌بندی دشوار را ساده می‌کنند.

موفقیت SVMها در جامعه دانشگاهی و همچنین در دنیای گسترده‌تر مهندسی نرم‌افزار که در آن کاربردهای مرتبط با یادگیری ماشین در حال افزایش بود، مورد توجه قرار گرفت. عموم مردم تا حد زیادی از این پیشرفت‌ها بی‌خبر بودند، هرچند که ماشین‌های هوشمند به طور مکرر در داستان‌های علمی تخیلی ظاهر می‌شدند. این زمستان هوش مصنوعی در سال 1997 با پیروزی ابرکامپیوتر دیپ بلو (Deep Blue) شرکت IBM در برابر قهرمان زمان خود در شطرنج، گری کاسپاروف، به پایان رسید. در آن زمان، تعداد کمی از مردم فکر می‌کردند که یک ماشین می‌تواند بهترین بازیکن شطرنج انسانی را شکست دهد. جالب است که یک دهه قبل، یکی از اساتید من پیش‌بینی کرده بود که یک هوش مصنوعی قبل از سال 2000 به این موفقیت دست خواهد یافت. آیا این استاد پیشگو بود؟ نه چندان. دیپ بلو سخت‌افزار سفارشی سریع را با نرم‌افزار پیچیده ترکیب کرده و الگوریتم‌های جستجوی شناخته شده هوش مصنوعی (به ویژه، الگوریتم مینی‌مکس) را به کار گرفت. همراه با بحث‌های اکتشافی و مقدار قابل توجهی دانش سفارشی از دیگر استادان بزرگ شطرنج، دیپ بلو توانست با جستجوی حرکات ممکن بیشتر از آنچه که هر انسانی می‌توانست تصور کند، حریف انسانی خود را شکست دهد. با این حال، در هسته اصلی خود، دیپ بلو آنچه را که کارشناسان هوش مصنوعی می‌دانستند که می‌تواند یک انسان را شکست دهد، در صورتی که ماشین منابع کافی در اختیار داشته باشد، پیاده‌سازی کرد. پیروزی دیپ بلو اجتناب‌ناپذیر بود زیرا محققان انتظار داشتند که کامپیوترها در نهایت به حدی سریع شوند که بتوانند بر قابلیت‌های انسانی غلبه کنند. آنچه نیاز بود شناخته شده بود؛ تنها چیزی که باقی مانده بود، کنار هم

قرار دادن قطعات بود. سال 1998 انتشار مقاله‌ای با عنوان «یادگیری مبتنی بر گرادیان در شناسایی اسناد» را شاهد بود، مقاله‌ای از یان لکون (Yann LeCun)، لئون باتو (Leon Bottou)، یوشوا بنگیو (Yoshua Bengio) و پاتریک هافنر (Patrick Haffner) که از نظر عمومی مورد توجه قرار نگرفت، اما یک لحظه تعیین‌کننده برای هوش مصنوعی و جهان بود. در حالی که نئوکاگنیترون فوکوشیما شباهت‌های قوی به شبکه‌های عصبی پیچشی داشت که انقلاب مدرن هوش مصنوعی را آغاز کردند، این مقاله به طور مستقیم آنها را معرفی کرد (نئوکاگنیترون یک شبکه عصبی مصنوعی سلسله مراتبی و چندلایه است که توسط کونیهیکو فوکوشیما (Kunihiko Fukushima) در سال 1979 پیشنهاد شد). ظهور شبکه‌های عصبی پیچشی (CNNs) در سال 1998 این پرسش را به وجود می‌آورد: چرا 14 سال دیگر طول کشید تا جهان متوجه این موضوع شود؟ ما بعداً به این سوال برخواهیم گشت.

2000 + تا 2012

لئو بریمن (Leo Breiman) در سال 2001 جنگل‌های تصادفی (Random Forests) را معرفی کرد و اجزای موجود از آنچه که به الگوریتم جنگل تصادفی تبدیل می‌شود را به یک کل منسجم شکل داد، مشابه آنچه که داروین در قرن نوزدهم با نظریه تکامل انجام داد. جنگل‌های تصادفی آخرین الگوریتم‌های کلاسیک یادگیری ماشین هستند که در فصل 3 به آن‌ها خواهیم پرداخت. اگر «جنگل‌های تصادفی» شما را به یاد درختان تصمیم‌گیری در فصل 1 می‌اندازد، دلیلی دارد: یک جنگل تصادفی یک جنگل از درختان تصمیم‌گیری است.

رمزگذارهای خودکار حذف نویز انباشته (Stacked Denoising Autoencoders) نوعی مدل میانجی هستند که سبب آشنایی من با یادگیری عمیق در سال 2010 بودند. یک رمزگذار خودکار شبکه‌ای عصبی است که ورودی خود را از طریق یک لایه میانی قبل از تولید خروجی عبور می‌دهد. هدف آن بازتولید ورودی خود از فرم کدگذاری شده ورودی در لایه میانی است. ممکن است ور رفتن با رمزگذار خودکار کاری احمقانه به نظر برسد اما وقتی لایهٔ میانی در حال یادگیری برای بازتولید ورودی خود است، معمولاً چیزی جالب دربارهٔ ورودی‌هایش یاد می‌گیرد که ماهیت آن را بدون تمرکز بر جزئیات بی‌اهمیت، به تصویر می‌کشد. مثلاً اگر ورودی‌ها ارقام MNIST باشند، لایهٔ میانی یک رمزگذار خودکار دربارهٔ ارقام یاد می‌گیرد نه حروف.

رمزگذار خودکار حذف نویز مشابه توضیحاتی است که داده شد، با این نکته که یک بخش تصادفی از مقادیر ورودی را قبل از عبور ورودی از لایه میانی کنار می‌گذاریم. رمزگذار خودکار همچنان باید یاد بگیرد که کل ورودی را بازتولید کند، اما اکنون وظیفه‌ای چالش‌برانگیزتر دارد زیرا ورودی ناقص است. این فرآیند به لایه میانی رمزگذار خودکار کمک می‌کند تا یک کدگذاری بهتر از ورودی کشف کند.

در نهایت، رمزگذار خودکار حذف نویز انباشته یک انباشت از رمزگذارهای خودکار حذف نویز است که در آن خروجیِ لایه میانیِ یکی، ورودیِ لایه بعدی می‌شود. وقتی به این شکل ترتیب داده می‌شود، انباشت یک نمایش جدید از ورودی یاد می‌گیرد که اغلب به یک طبقه‌بند که به بالای انباشت اضافه شده کمک می‌کند تا بین کلاس‌ها تمایز قائل شود. به عنوان مثال، در کار من در آن زمان، ورودی‌ها قطعات کوچک از یک تصویر بودند که ممکن بود شامل هدفی که مد نظرمان بود باشند. دو یا سه لایه از رمزگذارهای خودکار حذف نویز انباشته آموزش‌دیده برای تبدیل ورودی‌ها به یک لیست از اعداد استفاده شدند که امیدوار بودیم ماهیت ورودی را در حالی که به جزئیات تصویر بی‌توجهی می‌کرد، نمایان کنند. سپس خروجی‌ها با یک ماشین بردار پشتیبان برای تصمیم‌گیری درباره اینکه آیا ورودی یک هدف است یا خیر، استفاده شدند.

+ 2012 تا 2021

یادگیری عمیق توجه جهان را در سال 2012 جلب کرد؛ زمانی که AlexNet، یک معماری خاص از شبکه‌های عصبی پیچشی، با نرخ خطای کلی کمی بیش از 15 درصد در چالش ImageNet پیروز شد که بسیار کمتر از هر رقیب دیگری بود. چالش ImageNet از مدل‌ها می‌خواهد تا موضوع اصلی تصاویر رنگی را شناسایی کنند، خواه یک سگ، یک گربه، یک چمن‌زن و غیره. در واقع، پاسخ «سگ» کافی نیست. مجموعه داده ImageNet شامل 1000 کلاس از اشیاء است که حدود 120 نژاد مختلف سگ را شامل می‌شود. بنابراین، یک پاسخ صحیح می‌تواند «این یک سگ از نژاد X است» باشد.

حدس زدن تصادفی به معنای انتساب تصادفی یک برچسب کلاس به هر تصویر است. در این صورت، ما انتظار داریم که نرخ موفقیت کلی 1 در 1000 باشد، یا نرخ خطای 99.9 درصد. نرخ خطای 15 درصد AlexNet واقعاً چشمگیر بود – آن هم در سال 2012. تا سال 2017، شبکه‌های عصبی پیچشی نرخ خطا را به حدود 3 درصد کاهش داده بودند، که کمتر از حدود 5 درصدِ قابل دستیابی توسط چند انسانی بود که شجاعت انجام چالش به صورت دستی را داشتند. آیا می‌توانید بین 120 نژاد مختلف سگ تمایز قائل شوید؟ من قطعاً نمی‌توانم. AlexNet دروازه‌ها را باز کرد. مدل‌های جدید تمامی رکوردهای قبلی را شکستند و شروع به انجام کارهایی کردند که هیچ کس واقعاً از آن‌ها انتظار نداشت: وظایفی مانند بازسازی تصاویر به سبک یک تصویر یا نقاشی دیگر، تولید توصیف متنی از محتوای یک تصویر به همراه فعالیت نشان داده شده، انجام بازی‌های ویدئویی به خوبی انسان یا حتی بهتر از او و غیره.

این حوزه به سرعت در حال گسترش بود طوری که تقریباً غیرممکن شد تا با سیل روزانه مقالات جدید همگام بمانیم. تنها راه برای به‌روز ماندن، شرکت در چندین کنفرانس در سال و مرور کارهای جدیدی بود که در وب‌سایت‌هایی مانند arXiv (https://www.arxiv.org) منتشر می‌شد، جایی که تحقیقات در بسیاری از حوزه‌ها ابتدا منتشر می‌شود. این منجر به ایجاد سایت‌هایی مانند https://www.arxiv-sanity-lite.com شد که مقالات یادگیری ماشین را بر اساس علاقه خوانندگان رتبه‌بندی می‌کند با این امید که «بهترین»ها راحت‌تر پیدا شوند. در سال 2014، به لطف بینش محققی به نام ایان گودفلو (Ian Goodfellow) در خلال یک گفت‌وگوی شبانه با دوستان پیشرفت دیگری به صحنه آمد؛ نتیجه این بود که شبکه‌های مولد تخاصمی (GANs) متولد شدند، که یان لکون (Yann LeCun) در آن زمان آن را بزرگترین پیشرفت در شبکه‌های عصبی در 20 تا 30 سال اخیر خواند (این جمله در NeurIPS 2016 شنیده شد). GANها، که در فصل ششم مورد بحث قرار خواهند گرفت، یک حوزه جدید تحقیقاتی را گشودند که به مدل‌ها اجازه می‌دهد تا خروجی‌هایی تولید کنند که مرتبط با داده‌هایی هستند که بر روی آن‌ها آموزش دیده‌اند، اما با آن داده‌ها متفاوتند. GANها منجر به انفجار فعلی هوش مصنوعی مولد شدند، از جمله سیستم‌هایی مانند ChatGPT و Stable Diffusion.

یادگیری تقویتی (Reinforcement Learning) یکی از سه شاخه اصلی یادگیری ماشین است، دو شاخه دیگر شامل یادگیری تحت نظارت که درباره آن صحبت کرده‌ایم و یادگیری بدون نظارت است که سعی دارد مدل‌ها را بدون مجموعه داده‌های برچسب‌گذاری شده آموزش دهد. در یادگیری تقویتی، یک عامل (یک مدل) از طریق یک تابع پاداش آموزش می‌بیند که چگونه یک وظیفه را انجام دهد. کاربرد آن در رباتیک واضح است.

گروه DeepMind گوگل در سال 2013 یک سیستم مبتنی بر یادگیری تقویتی عمیق معرفی کرد که می‌توانست به‌طور موفقیت‌آمیز یاد بگیرد که چگونه بازی‌های ویدئویی آتاری 2600 را به همان اندازه یا بهتر از کارشناسان انسانی بازی کند. (از

اینکه چه کسی در یک سیستم بازی 35 ساله کارشناس حساب می‌شود، مطمئن نیستم.) برای من، بخش قابل توجه این سیستم این بود که ورودی مدل دقیقاً ورودی انسان بود: تصویری از صفحه نمایش، و نه چیزی بیشتر. این بدان معنا بود که سیستم باید یاد می‌گرفت چگونه تصویر ورودی را تجزیه و تحلیل کند و با توجه به آن، چگونه پاسخ دهد و با حرکت جوی‌استیک بازی را ببرد (به‌صورت مجازی - آن‌ها از شبیه‌سازها استفاده کردند). فاصله بین شکست دادن انسان‌ها در بازی‌های ویدئویی ابتدایی و شکست دادن انسان‌ها در بازی‌های استراتژیک انتزاعی مانند Go، به‌طور تاریخی غیرقابل عبور تلقی می‌شد. در اواخر دهه 1980 به‌طور واضح به من آموزش داده شد که الگوریتم مینی‌مکس که توسط سیستم‌هایی مانند دیپ بلو برای پیروزی در شطرنج استفاده می‌شود، روی بازی‌ای مانند Go قابل اعمال نیست؛ بنابراین هیچ ماشینی هرگز بهترین بازیکنان انسانی Go را شکست نخواهد داد. اما اساتید من اشتباه می‌کردند، هرچند در آن زمان دلایل کافی برای باور به این جمله داشتند.

در سال 2016، سیستم AlphaGo که متعلق به گوگل بود، قهرمان Go، لی سدول (Lee Sedol) را در یک مسابقهٔ پنج-بازی شکست داد و چهار به یک برد. جهان متوجه این موضوع شد و این آگاهی که یک تغییر پارادایم رخ داده است، بیشتر تقویت شد. تا آن زمان، یادگیری ماشین یک موفقیت تجاری بود. با این حال، پیروزی AlphaGo برای محققان و کارورزان یادگیری ماشین کاملاً چشمگیر بود.

بیشتر مردم عادی متوجه نشدند که AlphaGo، که بر اساس هزاران بازی که توسط انسان‌ها انجام شده بود آموزش دیده بود، در سال 2017 با AlphaGo Zero جایگزین شد، سیستمی که به‌طور کامل از ابتدا با بازی کردن در برابر خودش آموزش دیده بود و هیچ ورودی انسانی به آن داده نشده بود. در مدت کوتاهی، AlphaGo Zero به Go مسلط شد و حتی سیستم اصلی AlphaGo را نیز شکست داد (با کسب صد پیروزی کامل و بدون باخت).

با این حال، در سال 2022، سیستم پیشرفتهٔ فعلی Go، KataGo، به طور مکرر و آسان توسط سیستمی که برای پیروزی آموزش داده نشده بود، بلکه برای فاش کردن شکنندگی ذاتی در سیستم‌های هوش مصنوعی مدرن آموزش دیده بود، شکست خورد. حرکاتی که این سیستم متخاصم استفاده کرد، خارج از دامنه‌ای بود که KataGo در هنگام آموزش خود با آن مواجه شده بود. این یک مثال واقعی از این است که چگونه مدل‌ها در درون‌یابی خوب عمل می‌کنند اما در برون‌یابی ضعیف هستند. زمانی که سیستم متخاصم آموزش دیده بود تا در Go بهتر نباشد بلکه به «ناامید کردن» هوش مصنوعی بپردازد، توانست در بیش از سه چهارم بازی‌ها پیروز شود.

تمایل یادگیری عمیق به شکست دادن انسان‌ها در بازی‌های ویدیویی ادامه دارد. به جای بازی‌های ابتدایی مانند آتاری، سیستم‌های یادگیری تقویتی عمیق اکنون در بازی‌های بسیار دشوارتر به عملکردی در سطح استاد بزرگ دست یافته‌اند. در سال 2019، سیستم AlphaStar دیپ‌مایند، 99.8 درصد از بازیکنان انسانی در بازی استراتژیک استارکرفت II را outperform کرد، بازی‌ای که نیاز به توسعه واحدها و برنامه‌ریزی برای نبرد دارد.

کنفرانس آسیلومار (Asilomar) 1975 درباره DNA نوترکیب یک نقطه عطف مهم در شناسایی رشد بیوتکنولوژی و مسائل اخلاقی بالقوه آن بود. این کنفرانس تأثیر مثبت بر تحقیقات آینده داشت و در همان سال، سازمان‌دهندگان آن یک مقاله خلاصه منتشر کردند که رویکردی اخلاقی به بیوتکنولوژی را مورد بررسی قرار می‌داد. خطرات بالقوه یک حوزه که آن زمان در مرحله نوزادی خود بود، زود شناسایی شد و اقداماتی انجام شد تا اطمینان حاصل شود که مسائل اخلاقی در هنگام تفکر درباره تحقیقات آینده در اولویت قرار دارند. کنفرانس آسیلومار 2017 درباره هوش مصنوعی سودمند به‌طور عمدی به کنفرانس قبلی

شباهت داشت تا آگاهی از خطرات بالقوه مرتبط با هوش مصنوعی را افزایش دهد. اکنون رایج است که با جلسات کنفرانسی با عناوینی مانند «هوش مصنوعی برای مقاصد خیر» روبه‌رو شویم. کنفرانس آسیلومار 2017 منجر به توسعه مجموعه‌ای از اصول برای هدایت رشد و کاربرد هوش مصنوعی شد. به‌طور مشابه، از سال 2023، دولت ایالات متحده، به‌ویژه دفتر سیاست علم و فناوری کاخ سفید، یک «نقشه راه برای منشور حقوق هوش مصنوعی» را توسعه داده است که به منظور حفاظت از عموم مردم آمریکا در برابر اثرات مضر هوش مصنوعی اعمال شده است. در واقع، مقامات کاخ سفید تلاش کرده‌اند تا به‌طور مستقیم به جامعه هوش مصنوعی بپردازند و تشویق کنند که در توسعه سیستم‌های هوش مصنوعی قدرتمندتر، توجه مناسبی صورت گیرد. تمام این موارد نشانه خوبی است، اما تاریخ به ما می‌آموزد که قانون انسانی اغلب از توسعه فناوری عقب می‌ماند، بنابراین تأثیر نهایی این تلاش‌های ضروری برای تعریف این حوزه همچنان باید دیده شود.

+ 2021 تا اکنون

همه‌گیری COVID-19 در سال 2020 بیشتر جهان را به حالت توقف درآورد. با این حال، جامعه هوش مصنوعی تنها به‌طور حداقلی تحت تأثیر این همه‌گیری قرار گرفت، احتمالاً به این دلیل که همکاری و کنفرانس‌های از راه دور در این حوزه به خوبی کار می‌کند. علاوه بر این، ما می‌توانیم به کامپیوترهای قدرتمند از طریق اینترنت دسترسی داشته باشیم، بنابراین فاصلهٔ فیزیکی تحقیق را محدود نمی‌کند. از سال 2021 به بعد و تا زمانی که این متن را می‌نویسم، انفجاری از مدل‌های جدید ظاهر شده است که هر یک از دیگری چشمگیرتر است. بیشتر این مدل‌ها می‌توانند ورودی متنی که توسط انسان‌ها نوشته شده را بپذیرند و خروجی‌هایی از جمله متن، تصویر یا حتی ویدیو تولید کنند. معمولاً، ورودی متن و تصاویر نمونه را ترکیب می‌کند تا سیستم را راهنمایی کند. سیستم‌های محبوب (و نگران‌کننده) کنونی شامل DALL-E 2، Stable Diffusion، Midjourney و ChatGPT و دیگران است. بیشتر شرکت‌های بزرگ فناوری نیز تکنیک‌های پیشرفته‌ای را نشان داده‌اند که ویدیو، صدا و حتی مدل‌های سه‌بعدی تولید می‌کنند. هیجان کنونی که این سیستم‌ها ایجاد کرده‌اند، یادآور هیجان سال 2012 است که یادگیری عمیق با AlexNet ارزش خود را اثبات کرد. این یک تغییر پارادایم دیگر است؛ اما جهت شفافیت، هیچ یک از این سیستم‌ها آگاه نیستند. من در این نظر تنها نیستم. حتی ChatGPT نیز با من موافق است. من از ChatGPT پرسیدم: «آیا تو موجودی آگاه هستی؟» پاسخ به صورت عین به عین این است: به عنوان یک مدل یادگیری ماشین، من موجودی آگاه نیستم. من یک برنامه کامپیوتری هستم که برای تولید متن بر اساس ورودی از یک کاربر آموزش دیده‌ام. من توانایی تفکر، استدلال یا تجربه آگاهی به همان شیوه‌ای که انسان‌ها انجام می‌دهند را ندارم. من برای ارائه اطلاعات و کمک به کاربران طراحی شده‌ام، اما هیچ تفکر مستقل یا آگاهی ندارم.

فکر می‌کنم آدا لاولیس از این پاسخ راضی خواهد بود، هرچند ممکن است درباره اینکه یک ماشین بی‌فکر چگونه می‌تواند چنین جوابی تولید کند، گیج شود. با این حال، توجه داشته باشید که ChatGPT ادعا نکرده که نمی‌تواند فکر کند، فقط گفته است که مانند انسان‌ها فکر نمی‌کند. ما در فصل ششم به بررسی سنتز تصویر و در فصل هفتم به مدل‌های زبانی بزرگ مانند ChatGPT خواهیم پرداخت. شاید در این صورت راه حلی برای سردرگمی فرضی او پیدا کنیم.

به سؤال اولمان برگردیم، چرا حالا؟ پاسخ کوتاه این است که: سقوط هوش مصنوعی نمادین و ظهور فناوری‌های نوین که به شدت با رویکرد ارتباط‌گرایانه مساعد است. هوش مصنوعی نمادین و ارتباط‌گرایی به طور همزمان ظهور کردند، به طوری که هوش مصنوعی نمادین برای دهه‌ها برتری داشت و ارتباط‌گرایی را به حاشیه راند. اما پس از دو زمستان هوش مصنوعی که

باعث شد هوش مصنوعی نمادین به سختی نفس بکشد، ارتباط‌گرایی، با کمک نوآوری‌های کلیدی، به صحنه آمد تا جای خالی را پر کند.

من رابطه بین هوش مصنوعی نمادین و ارتباط‌گرایی را مشابه رابطه بین دایناسورهای غیرپرنده و پستانداران می‌دانم. دایناسورها و پستانداران به طور تقریبی در یک زمان مشابه از نظر زمین‌شناسی ظهور کردند، اما دایناسورهای بزرگ و زمینی به مدت حدود 160 میلیون سال بر جهان تسلط داشتند و پستانداران را وادار کردند که در سایه‌ها زندگی کنند. زمانی که شهاب‌سنگ 66 میلیون سال پیش به زمین برخورد کرد، دایناسورهای بزرگ نابود شدند و این امکان را برای پستانداران فراهم کرد تا تکامل یابند و برتری پیدا کنند. البته دایناسورها به طور کامل از بین نرفتند، ما باقی ماندهٔ آنها را اکنون پرندگان می‌نامیم. در واقع، پرندگان یکی از بزرگ‌ترین داستان‌های موفقیت زمین هستند. دایناسورهای غیرپرنده به دلیل بدشانسی محض از بین رفتند. به معنای واقعی کلمه، این یک فاجعه (Disaster) بود که آنها را نابود کرد (فاجعه از واژه ایتالیایی disastro به معنی «ستاره شوم»).

آیا ممکن است هوش مصنوعی نمادین دوباره به عرصه بازگردد؟ احتمالاً به شکلی خاص، اما در همکاری با ارتباط‌گرایی. هوش مصنوعی نمادین وعده داده بود که رفتار هوشمند در حالت انتزاعی ممکن است و نتوانست به این وعده عمل کند. ارتباط‌گرایی ادعا می‌کند که رفتار هوشمند می‌تواند از مجموعه‌ای از واحدهای ساده‌تر پدیدار شود. موفقیت‌های یادگیری عمیق این دیدگاه را تقویت می‌کند، و از میلیاردها مغز زنده‌ای که در حال حاضر بر روی سیاره وجود دارد، چیزی نمی‌گوییم. اما، همان‌طور که ChatGPT اشاره کرد، مدل‌های ارتباط‌گرای موجود «نمی‌اندیشند، استدلال نمی‌کنند و آگاهی را به شیوه‌ای که انسان‌ها از آن برخوردارند، تجربه نمی‌کنند.» شبکه‌های عصبی مدرن ذهن نیستند؛ بلکه پردازنده‌های داده‌های یادگیری نمایشی هستند. در فصل 5 توضیح می‌دهم که این به چه معناست.

اگرچه گونه ما، هومو ساپینس یا انسان خردمند، به شدت به تفکر نمادین وابسته است، این الزامی برای هوش نیست. در کتاب خود با عنوان «درک تکامل انسان» (انتشارات دانشگاه کمبریج، 2022)، انسان‌شناس، ایان تاترسال (Ian Tattersall) ادعا می‌کند که به احتمال زیاد نئاندرتال‌ها از تفکر نمادین به شیوه ما استفاده نمی‌کردند و همچنین زبان به شیوه ما نداشتند، اما با این وجود، آنها هوشمند بودند. در واقع، نئاندرتال‌ها به حدی انسانی بودند که اجداد ما بیش از یک بار با آنها «عشق ورزیدند، نه این که بجنگند» - DNA افرادی با نژاد غیر آفریقایی به این واقعیت گواهی می‌دهد. من انتظار دارم که در آینده نزدیک یک هم‌افزایی بین ارتباط‌گرایی و هوش مصنوعی نمادین شکل بگیرد. به عنوان مثال، از آنجا که سیستمی مانند ChatGPT در نهایت تنها پیش‌بینی می‌کند که خروجی بعدی (کلمه یا بخشی از کلمه) چیست، نمی‌تواند بداند که آیا در حال گفتن چیزی اشتباه است. یک سیستم نمادین مرتبط می‌تواند استدلال نادرست در پاسخ را شناسایی کرده و آن را اصلاح کند. چگونگی پیاده‌سازی چنین سیستمی را نمی‌دانم.

نشانه‌هایی از آنچه ممکن است از ارتباط‌گرایی پدید آید، تا اوایل دهه 1960 قابل مشاهده بود. بنابراین، آیا تنها تعصب هوش مصنوعی نمادین بود که انقلاب را برای چندین دهه به تأخیر انداخت؟ خیر. ارتباط‌گرایی به دلیل مشکلات سرعت، الگوریتم و داده‌ها متوقف شد. بیایید هر یک را به نوبه خود بررسی کنیم.

سرعت

برای درک این که چرا عامل «سرعت»، رشد ارتباط‌گرایی را متوقف کرد، باید بفهمیم که کامپیوترها چگونه کار می‌کنند. با اندکی آزادی در تفکر می‌توانیم کامپیوترها را به عنوان حافظه‌ای که داده‌ها (اعداد) را نگه می‌دارند و یک واحد پردازش که اغلب واحد پردازش مرکزی (CPU) شناخته می‌شود، در نظر بگیریم. یک ریزپردازنده مثل همانی که در رایانهٔ رومیزی، تلفن همراه هوشمند، دستیار صوتی هوشمند، اتومبیل، مایکروویو و ... شما وجود دارد، یک CPU است. CPU را کامپیوتری سنتی در نظر بگیرید: داده‌ها از حافظه یا دستگاه‌های ورودی مانند صفحه کلید یا ماوس وارد CPU می‌شوند، پردازش می‌شوند و سپس به حافظه یا یک دستگاه خروجی مانند صفحه نمایش یا هارد دیسک ارسال می‌شوند.

از سوی دیگر، واحدهای پردازش گرافیکی (GPU‌ها) برای صفحه نمایش‌ها، به ویژه در صنعت بازی‌های ویدئویی، توسعه یافته‌اند تا گرافیک سریع‌تری را ارائه دهند. GPU‌ها می‌توانند همان عملیات، مانند «ضرب در دو» را بر روی صدها یا هزاران مکان حافظه (به عبارت دیگر: پیکسل‌ها) به طور هم‌زمان انجام دهند.

اگر یک CPU بخواهد هزار مکان حافظه را در دو ضرب کند، باید به ترتیب مکان اول، دوم، سوم و غیره را ضرب کند. به طوری که، عملیات اصلی مورد نیاز برای آموزش و پیاده‌سازی یک شبکه عصبی به طور ایده‌آل با آنچه یک GPU می‌تواند انجام دهد مطابقت دارد. سازندگان GPU، مانند NVIDIA، این موضوع را زود متوجه شدند و شروع به توسعه GPU‌ها برای یادگیری عمیق کردند. تصور کنید که یک GPU مانند یک ابرکامپیوتر بر روی یک کارت است که در کامپیوتر شما جا می‌گیرد.

در سال 1945، محاسبه‌گر و انتگرال‌گیر عددی الکترونیکی (ENIAC) در اوج فناوری قرار داشت. سرعت ENIAC حدود 0.00289 میلیون دستور در ثانیه (MIPS) تخمین زده شده بود. به عبارتی، ENIAC می‌توانست در یک ثانیه تقریباً 3000 دستور را اجرا کند. در سال 1980، یک ریزپردازنده 8 بیتی معمولی 6502 مانند آنچه در بیشتر کامپیوترهای شخصی محبوب آن زمان وجود داشت، با سرعت حدود 0.43 MIPS، یا حدود 500000 دستور در ثانیه کار می‌کرد. در سال 2023، CPU اینتل i7-4790 که در کامپیوتری که برای نوشتن این کتاب استفاده می‌کنم به کار رفته، به طور تقریبی با سرعت 130000 MIPS کار می‌کند، حدود 300000 بار سریع‌تر از ریزپردازندهٔ 6502 سال 1980 و حدود 45 میلیون بار سریع‌تر از ENIAC است. با این حال، GPU A100 از NVIDIA، زمانی که برای یادگیری عمیق استفاده می‌شود، قادر به انجام 312 ترافلاپ (TFLOPS) یا 312000000 MIPS است: 730 میلیون بار سریع‌تر از 6502 و به طور باورنکردنی 110 میلیارد بار سریع‌تر از ENIAC. افزایش قدرت محاسباتی در طول زمان ذهن را متحیر می‌کند. علاوه بر این، آموزش یک شبکه عصبی بزرگ بر روی مجموعه داده‌ای عظیم اغلب نیاز به ده‌ها تا صدها عدد از این GPU‌ها دارد.

نتیجه‌گیری: تا پیش از ظهور GPU‌های سریع، کامپیوترها برای آموزش شبکه‌های عصبی با ظرفیت لازم برای ساخت چیزی مانند ChatGPT، خیلی کُند بودند.

الگوریتم

همان‌طور که در فصل چهارم خواهید آموخت، ما شبکه‌های عصبی را از واحدهای اساسی می‌سازیم که وظیفه‌ای ساده انجام می‌دهند: جمع‌آوری مقادیر ورودی، ضرب هر یک در مقدار وزن، جمع کردن آن‌ها، افزودن یک مقدار بایاس یا سوگیری (Bias)، و انتقال نتیجه به یک تابع فعال‌سازی برای ایجاد یک مقدار خروجی. به عبارت دیگر، تعداد زیادی عدد ورودی به یک عدد

خروجی تبدیل می‌شوند. رفتار جمعی ناشی از هزاران تا میلیون‌ها واحد از این دست که به میلیاردها مقدار وزن منجر می‌شود، به سیستم‌های یادگیری عمیق اجازه می‌دهد کاری را که باید، انجام دهند. ساختار شبکه عصبی یک چیز و شرطی‌سازی شبکه عصبی برای انجام وظیفه مورد نظر، چیز دیگری است. ساختار شبکه را که به عنوان معماری آن شناخته می‌شود، مانند آناتومی در نظر بگیرید. در آناتومی، ما به این علاقه‌مندیم که بدن از چه چیزهایی تشکیل شده است: این قلب است، آن کبد است و غیره. آموزش یک شبکه بیشتر شبیه فیزیولوژی است: یک بخش چگونه با بخش دیگر کار می‌کند؟ آناتومی (معماری) وجود داشت، اما فیزیولوژی (فرایند آموزش) به طور ناقص درک شده بود. این موضوع در طول دهه‌ها به لطف نوآوری‌های الگوریتمی کلیدی تغییر کرد: پس‌انتشار، مقداردهی اولیه شبکه، توابع فعال‌سازی، دراپ‌اوت (Dropout) و نرمال‌سازی، و الگوریتم‌های پیشرفته گرادیان کاهشی. درک دقیق این اصطلاحات ضروری نیست، فقط باید بدانید که بهبودهایی در آنچه این اصطلاحات نمایندگی می‌کنند - همراه با بهبودهای ذکرشده در سرعت پردازش و ترکیب با مجموعه‌های داده بهبود یافته (که در ادامه بحث خواهد شد) - عوامل اصلی توانمندساز انقلاب یادگیری عمیق بودند. در حالی که مدت‌ها مشخص شده بود که مقادیر وزن و بایاس مناسب، شبکه را برای وظیفه مورد نظر تطبیق می‌دهند، آنچه برای دهه‌ها ناشناخته بود، روشی کارآمد برای یافتن این مقادیر بود. معرفی الگوریتم پس‌انتشار در دهه 1980، همراه با گرادیان کاهشی تصادفی، شروع به تغییر این وضعیت کرد.

آموزش به صورت تکراری، مجموعه نهایی مقادیر وزن و بایاس را بر اساس خطاهای مدل روی داده‌های آموزشی تعیین می‌کند. فرایندهای تکراری از یک حالت اولیه، یعنی مجموعه اولیه‌ای از وزن‌ها و بایاس‌ها، تکرار می‌شوند. با این حال، این وزن‌ها و بایاس‌های اولیه باید چه باشند؟ برای مدت طولانی فرض بر این بود که وزن‌ها و بایاس‌های اولیه چندان مهم نیستند؛ فقط اعداد کوچکی را به صورت تصادفی در یک محدوده انتخاب کنید. این رویکرد اغلب کار می‌کرد، اما گاهی اوقات جواب نمی‌داد و باعث می‌شد شبکه به خوبی یاد نگیرد، یا اصلاً یاد نگیرد. به یک رویکرد اصولی‌تر برای مقداردهی اولیه شبکه‌ها نیاز بود.

شبکه‌های مدرن همچنان به صورت تصادفی مقداردهی اولیه می‌شوند، اما مقادیر تصادفی به معماری شبکه و نوع تابع فعال‌سازی استفاده‌شده بستگی دارند. توجه به این جزئیات به شبکه‌ها اجازه داد تا بهتر یاد بگیرند. مقداردهی اولیه مهم است. ما شبکه‌های عصبی را در لایه‌هایی مرتب می‌کنیم، که در آن خروجی یک لایه به ورودی لایه بعدی تبدیل می‌شود. تابع فعال‌سازی اختصاص داده شده به هر گره در شبکه، مقدار خروجی گره را تعیین می‌کند. در گذشته، تابع فعال‌سازی معمولاً یک سیگموئید یا تانژانت هیپربولیک بود، که هر دو هنگام ترسیم نمودار، یک منحنی S شکل تولید می‌کنند. این توابع در اکثر موارد نامناسب بودند و در نهایت با تابعی با نامی طولانی که سادگی آن را پنهان می‌کند، جایگزین شدند: واحد یک سو شدهٔ خطی (ReLU). یک ReLU سؤالی ساده می‌پرسد: آیا ورودی کمتر از صفر است؟ اگر بله، خروجی صفر است؛ در غیر این صورت، خروجی همان مقدار ورودی است. نه تنها توابع فعال‌سازی ReLU بهتر از توابع قدیمی هستند، بلکه کامپیوترها می‌توانند این سؤال را تقریباً به‌صورت آنی بپرسند و پاسخ دهند. بنابراین، تغییر به ReLU یک پیروزی دوگانه بود: بهبود عملکرد شبکه و سرعت.

دراپ‌اوت و نرمال‌سازیِ دسته‌ای (Batch Normalization) رویکردهایی پیشرفته برای آموزش هستند که توصیفشان در سطحی که ما به دانستن آن‌ها علاقه‌مندیم، تا حدی دشوار است. دراپ‌اوت که در سال 2012 معرفی شد، به‌صورت تصادفی بخش‌هایی از خروجی یک لایه از گره‌ها را هنگام آموزش به صفر تنظیم می‌کند. اثر این کار مانند آموزش هزاران مدل به‌صورت همزمان است، هر کدام مستقل اما در عین حال مرتبط. دراپ‌اوت، در صورتی که مناسب باشد، تأثیر چشمگیری بر یادگیری شبکه دارد. همان‌طور که یک دانشمند برجسته علوم کامپیوتر در آن زمان به من گفت: «اگر در دهه 1980 دراپ‌اوت را داشتیم، دنیا اکنون متفاوت بود.»

نرمال‌سازی دسته‌ای، داده‌هایی را که بین لایه‌ها در شبکه جریان دارند، تنظیم می‌کند. ورودی‌ها در یک سمت شبکه ظاهر می‌شوند و از طریق لایه‌ها به سمت خروجی جریان می‌یابند. به‌صورت شماتیک، این معمولاً به‌صورت حرکتی از چپ به راست نشان داده می‌شود. نرمال‌سازی بین لایه‌ها قرار می‌گیرد تا مقادیر را تغییر دهد و آن‌ها را در یک محدوده معنادار نگه دارد. نرمال‌سازی دسته‌ای، اولین تکنیک نرمال‌سازی قابل یادگیری بود، به این معنا که با یادگیری شبکه، می‌آموخت چه کاری باید انجام دهد. مجموعه‌ای کامل از رویکردهای نرمال‌سازی از نرمال‌سازی دسته‌ای تکامل یافت.

آخرین نوآوری الگوریتمی حیاتی که انقلاب یادگیری عمیق را ممکن ساخت، به گرادیان کاهشی مربوط می‌شود که با پس‌انتشار همکاری می‌کند تا یادگیری وزن‌ها و بایاس‌ها را تسهیل کند. ایده پشت گرادیان کاهشی بسیار قدیمی‌تر از یادگیری ماشینی است، اما نسخه‌هایی که در دهه گذشته یا حدود آن توسعه یافتند، به موفقیت یادگیری عمیق کمک زیادی کردند. ما در فصل چهارم بیشتر در مورد این موضوع یاد خواهیم گرفت.

نتیجه‌گیری: رویکردهای اولیه برای آموزش شبکه‌های عصبی ابتدایی بودند و نمی‌توانستند از پتانسیل واقعی آن‌ها بهره ببرند. نوآوری‌های الگوریتمی این وضعیت را تغییر دادند.

داده

شبکه‌های عصبی به حجم زیادی از داده‌های آموزشی نیاز دارند. وقتی مردم از من می‌پرسند که برای آموزش یک مدل خاص برای یک وظیفه خاص به چه مقدار داده نیاز است، پاسخم همیشه یکسان است: هر چه بیشتر، بهتر. زیرا داده‌های بیشتر به معنای بازنمایی بهبود یافته از چیزی است که مدل هنگام استفاده با آن مواجه خواهد شد.

پیش از ظهور شبکه جهانی وب، جمع‌آوری، برچسب‌گذاری و پردازش مجموعه‌های داده‌ای با حجم لازم برای آموزش یک شبکه عصبی عمیق دشوار بود. این وضعیت در اواخر دهه 1990 و اوایل دهه 2000 با رشد چشمگیر وب و انفجار داده‌هایی که نمایندگی می‌کرد، تغییر کرد.

برای مثال، استاتیستا (https://www.statista.com) ادعا می‌کند که در سال 2022، هر دقیقه 500 ساعت ویدیوی جدید روی یوتیوب آپلود می‌شد. همچنین تخمین زده می‌شود که در دسامبر 1995، حدود 16 میلیون نفر از وب استفاده می‌کردند که 0.4 درصد از جمعیت جهان را تشکیل می‌داد. تا جولای 2022، این تعداد به نزدیک 5.5 میلیارد نفر، یا 69 درصد، افزایش یافت. استفاده از رسانه‌های اجتماعی، تجارت الکترونیک، و صرفاً جابه‌جایی از مکانی به مکان دیگر در حالی که یک گوشی همراه هوشمند دارید، کافی است تا مقادیر حیرت‌انگیزی از داده تولید شود - داده‌هایی که همگی جمع‌آوری و برای هوش مصنوعی استفاده می‌شوند. رسانه‌های اجتماعی رایگان هستند زیرا ما و داده‌هایی که تولید می‌کنیم، محصول هستیم.

عبارتی که اغلب در کارم می‌شنوم این است: «ما قبلاً از کمبود داده رنج می‌بردیم، اما حالا در داده غرق شده‌ایم.» بدون مجموعه‌های داده بزرگ و برچسب‌های کافی که همراه آن‌ها باشد، یادگیری عمیق نمی‌تواند بیاموزد. اما از سوی دیگر، با مجموعه‌های داده بزرگ، اتفاقات شگفت‌انگیزی ممکن است رخ دهد.

نتیجه‌گیری: در یادگیری ماشین، داده همه چیز است.

نکات اصلی این فصل عبارتند از:

رقابت بین هوش مصنوعی نمادین و ارتباط‌گرایی زود ظاهر شد و به دهه‌ها تسلط هوش مصنوعی نمادین منجر شد.

ارتباط‌گرایی برای مدت طولانی به دلیل مشکلات سرعت، الگوریتم و داده رنج برد.

با انقلاب یادگیری عمیق در سال 2012، ارتباط‌گرایان، حداقل فعلاً، پیروز شده‌اند.

علل مستقیم انقلاب یادگیری عمیق عبارت بودند از کامپیوترهای سریع‌تر، ظهور واحدهای پردازش گرافیکی، الگوریتم‌های بهبود یافته و مجموعه‌های داده عظیم.

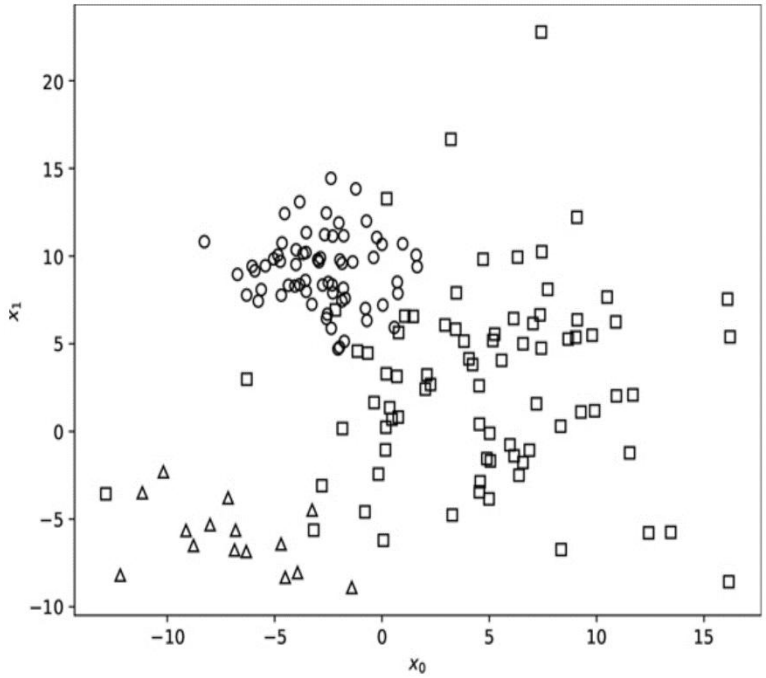
با این پیشینه تاریخی که برای اهداف ما به اندازه کافی کامل است، بیایید به یادگیری ماشینی بازگردیم و با الگوریتم‌های کلاسیک شروع کنیم.

فصل سوم: مدل‌های کلاسیک (یادگیری ماشین قدیمی)

دانش‌آموزان مبتدی یادگیری پیانو را با «La Campanella» (اثری از Liszt آهنگساز ایتالیایی) شروع نمی‌کنند، بلکه با «مری برهای کوچک داشت (Marry Had a Little Lamb)» یا «چشمک، چشمک، ستاره کوچک (Twinkle, Twinkle, Little Start)» آغاز می‌کنند. قطعات ساده‌تر اصول اولیه نواختن پیانو را در بر دارند و تسلط بر این اصول به دانش‌آموزان اجازه می‌دهد تا به مرور زمان پیشرفت کنند. این اصل در اکثر زمینه‌های مطالعاتی، از جمله هوش مصنوعی، صدق می‌کند.

برای رسیدن به هدف نهایی درک هوش مصنوعی مدرن، باید از دنیای «ساده‌تر» یادگیری ماشینی کلاسیک شروع کنیم. آنچه برای مدل‌های کلاسیک صادق است، به طور کلی برای شبکه‌های عصبی پیشرفته‌تر نیز صدق می‌کند. این فصل سه مدل کلاسیک را بررسی می‌کند: نزدیک‌ترین همسایگان (Nearest Neighbors)، جنگل‌های تصادفی (Random Forests) و ماشین‌های بردار پشتیبان (SVMs). درک این‌ها ما را برای شبکه‌های عصبی فصل چهارم آماده می‌کند.

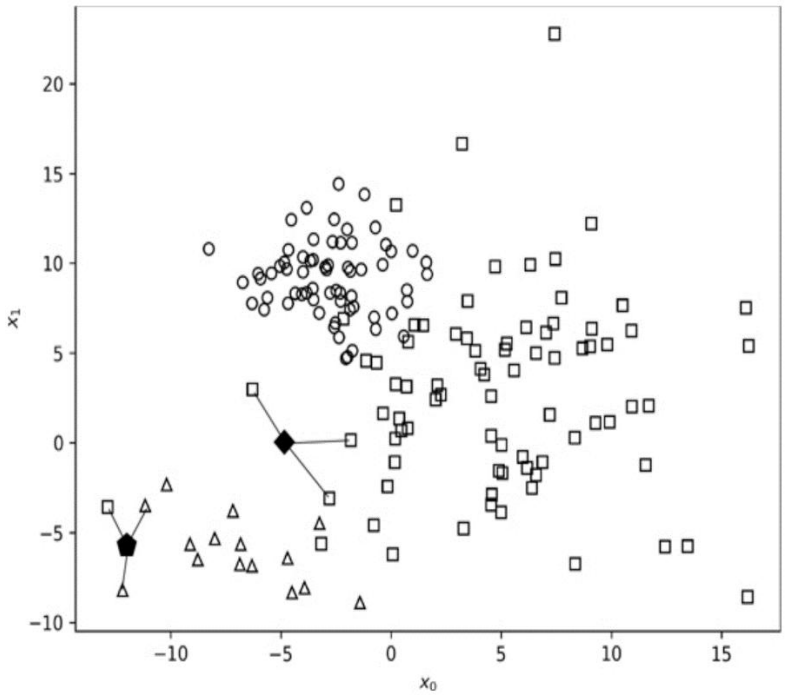
شکل 3-1 نمونه‌های آموزشی را برای یک مجموعه داده ساختگی با دو ویژگی (x_0 و x_1) و سه کلاس (دایره‌ها، مربع‌ها و مثلث‌ها) نشان می‌دهد. ما نموداری مشابه را در فصل اول دیدیم؛ به شکل 2-1 مراجعه کنید. مانند مجموعه داده زنبق، هر شکل هندسی در این نمودار یک نمونه از مجموعه آموزشی را نشان می‌دهد. شکل 3-1 ابزاری است که برای درک مدل کلاسیک «نزدیک‌ترین همسایگان» استفاده خواهیم کرد.



همان‌طور که در فصل قبل ذکر شد، طبقه‌بندهای نزدیک‌ترین همسایه ساده‌ترین مدل‌ها هستند - آن‌قدر ساده که هیچ مدلی برای آموزش وجود ندارد؛ داده‌های آموزشی خود مدل هستند. برای اختصاص یک برچسب کلاسی به یک ورودی جدید و ناشناخته، نزدیک‌ترین نمونه آموزشی به نمونه ناشناخته را پیدا کنید و برچسب آن نمونه را برگردانید. همین و بس. با وجود سادگی‌شان، طبقه‌بندهای نزدیک‌ترین همسایه در صورتی که داده‌های آموزشی نشان دهندهٔ آنچه مدل در عمل با آن مواجه خواهد شد باشند، کاملاً مؤثر هستند. به عنوان یک گسترش طبیعی از مدل نزدیک‌ترین همسایه، k نمونه آموزشی که

نزدیک‌ترین به نمونه ناشناخته‌اند را پیدا کنید. k اغلب عددی مانند 3، 5 یا 7 است، هرچند می‌تواند هر عددی باشد. این نوع مدل از یک سیستم رأی‌گیری اکثریتی استفاده می‌کند، بنابراین برچسب کلاسی اختصاص داده شده، برچسبی است که در میان k نمونه آموزشی شایع‌ترین باشد. اگر تساوی وجود داشته باشد، برچسب را به‌صورت تصادفی انتخاب کنید. برای مثال، اگر مدل در حال بررسی 5 نزدیک‌ترین همسایه به یک نمونه ناشناخته باشد و دو تا از آن‌ها کلاس 0 باشند و دو تای دیگر کلاس 3، آنگاه برچسب را با انتخاب تصادفی بین 0 و 3 اختصاص دهید؛ به طور متوسط، 50 درصد مواقع انتخاب درستی خواهید داشت.

بیایید از مفهوم نزدیک‌ترین همسایه برای طبقه‌بندی برخی ورودی‌های ناشناخته استفاده کنیم. شکل 2-3 دوباره نمونه‌های آموزشی را نشان می‌دهد، همراه با دو نمونه ناشناخته: لوزی و پنج‌ضلعی. ما می‌خواهیم این نمونه‌ها را به یکی از سه کلاس اختصاص دهیم: دایره، مربع یا مثلث. رویکرد نزدیک‌ترین همسایه می‌گوید که نزدیک‌ترین نمونه آموزشی به هر نمونه ناشناخته را پیدا کنید. برای لوزی، این نمونه یک مربع در بالا سمت چپ آن است؛ برای پنج‌ضلعی، به نظر می‌رسد که مثلثی در بالا سمت راست آن است. بنابراین یک طبقه‌بندِ نزدیک‌ترین همسایه، کلاس مربع را به لوزی و کلاس مثلث را به پنج‌ضلعی اختصاص می‌دهد.



گمان می‌کنم متوجه خطوطی شده‌اید که نمونه‌های ناشناخته در شکل 2-3 را به سه نمونه آموزشیِ نزدیک‌ترین متصل می‌کنند. این‌ها نمونه‌هایی هستند که اگر k برابر 3 باشد، باید استفاده شوند. در این مورد، طبقه‌بند دوباره کلاس مربع را به لوزی اختصاص می‌دهد، زیرا هر سه نمونه آموزشیِ نزدیک‌ترین، مربع هستند. برای پنج‌ضلعی، دو مورد از سه همسایه نزدیک‌ترین، مثلث هستند و یکی مربع است، بنابراین دوباره کلاس مثلث به پنج‌ضلعی اختصاص می‌یابد.

این مثال از بردارهای ویژگی دوبعدی، x_0 و x_1، استفاده می‌کند، بنابراین می‌توانیم فرآیند را تجسم کنیم. ما محدود به مدل‌هایی با تنها دو ویژگی نیستیم؛ می‌توانیم ده‌ها یا حتی صدها ویژگی داشته باشیم. مفهوم «نزدیک‌ترین» (فاصله) حتی زمانی که ویژگی‌ها بیش از حد زیاد باشند که نتوان آن‌ها را ترسیم کرد، همچنان معنای ریاضی دارد. در واقع، بسیاری از مفاهیم ریاضی

به عنوان معیارهای فاصله واجد شرایط هستند و در عمل، طبقه‌بندهای نزدیک‌ترین همسایه ممکن است بسته به مجموعه داده، از هر یک از این معیارها استفاده کنند.

برای مثال، بیایید به مجموعه داده ارقام MNIST که در فصل اول مطرح شد، بازگردیم. نمونه‌ها تصاویر کوچک و خاکستری از ارقام 0 تا 9 هستند که ما آن‌ها را به بردارهایی با 784 عنصر باز می‌کنیم. بنابراین، هر نمونه رقم در مجموعه آموزشی یک نقطه در فضای 784 بعدی است، همان‌طور که در مثال قبلی هر نمونه یک نقطه در فضای دوبعدی بود.

مجموعه داده کامل MNIST دارای 60000 نمونه آموزشی است، به این معنا که فضای آموزشی شامل 60000 نقطه پراکنده در فضای 784 بعدی است (نه دقیقاً، اما به زودی بیشتر توضیح می‌دهم). همچنین 10000 نمونه آزمایشی دارد که می‌توانیم از آن‌ها برای ارزیابی مدل نزدیک‌ترین همسایه استفاده کنیم. من مدل‌های 1-نزدیک‌ترین همسایه را با استفاده از تمام 60000 نمونه آموزشی، سپس 6000 نمونه، سپس 600 نمونه، و در نهایت تنها 60 نمونه آموزش دادم. شصت نمونه در مجموعه آموزشی به این معناست که تقریباً شش مثال برای هر رقم وجود دارد. می‌گویم «تقریباً» چون مجموعه آموزشی را به‌صورت تصادفی نمونه‌برداری کردم، بنابراین ممکن است هشت نمونه از یک رقم و تنها سه نمونه از رقم دیگر وجود داشته باشد. در هر مورد، مدل را با استفاده از تمام 10000 نمونه آزمایشی امتحان کردم، و بدین ترتیب استفاده از مدل در دنیای واقعی را شبیه‌سازی نمودم.

جدول 1-3 عملکرد مدل را با تغییر تعداد نمونه‌های آموزشی نشان می‌دهد:

Training set size	Accuracy (%)
60,000	97
6,000	94
600	86
60	66

به یاد آورید که دقت (Accuracy)، درصد نمونه‌های آزمایشی است که مدل با اختصاص برچسب صحیح رقم، از 0 تا 9، به درستی طبقه‌بندی کرده است. وقتی از کل مجموعه آموزشی استفاده می‌شود، مدل به طور متوسط 97 بار از 100 بار درست عمل می‌کند. حتی وقتی مجموعه آموزشی 10 برابر کوچک‌تر می‌شود، دقت همچنان 94 درصد است. با 600 نمونه آموزشی، حدود 60 نمونه برای هر رقم، دقت به 86 درصد کاهش می‌یابد. تنها زمانی که مجموعه آموزشی به طور متوسط به تنها شش نمونه برای هر رقم کوچک می‌شود، دقت به طور چشمگیری به 66 درصد افت می‌کند.

با این حال، قبل از اینکه بیش از حد به مدل نزدیک‌ترین همسایه سخت بگیریم، به یاد داشته باشید که 10 کلاس رقم وجود دارد، بنابراین حدس تصادفی به طور متوسط حدود 1 بار از 10 بار درست خواهد بود، که دقتی حدود 10 درصد دارد. در این دیدگاه، حتی مدل 60 نمونه‌ای شش برابر بهتر از حدس تصادفی عمل می‌کند. بیایید این پدیده را کمی بررسی کنیم تا ببینیم آیا می‌توانیم بینشی در مورد اینکه چرا مدل نزدیک‌ترین همسایه با داده‌های آموزشی اندک عملکرد خوبی دارد، به دست آوریم؟

تصور کنید که تنها در یک سالن بسکتبال هستید و در وسط زمین نشسته‌اید. ذره‌ای گرد و غبار در هوا جایی در سالن معلق است. برای راحتی، فرض می‌کنیم این ذره در موقعیت خود ثابت می‌ماند. حالا تصور کنید 59 ذره گرد و غبار دیگر نیز در هوا

وجود دارند. این 60 ذره گرد و غبار همان 60 نمونه رقم در مجموعه آموزشی ما هستند و سالن، دنیای سه‌بعدی است که بردارهای تصویر ارقام در آن زندگی می‌کنند. حالا فرض کنید یک ذره گرد و غبار جدید درست جلوی بینی شما ظاهر شده است. این یک بردار رقم جدید است که می‌خواهید آن را طبقه‌بندی کنید. مدل نزدیک‌ترین همسایه، فاصله بین این ذره گرد و غبار و 60 ذره‌ای که برچسب ارقام آن‌ها را می‌دانید، محاسبه می‌کند. نزدیک‌ترین ذره گرد و غبار به ذره جدید، زیر حلقه سبدی است که رو به آن نشسته‌اید، در فاصله 47 فوت (14 متر). این ذره یک رقم 3 است، بنابراین مدل برچسب 3 را برمی‌گرداند. آیا منطقی است که فکر کنیم نزدیک‌ترین ذره، برچسب مناسبی برای نمونه ناشناخته را نشان می‌دهد؟ به هر حال، تنها 60 ذره گرد و غبار در کل سالن وجود دارد.

برای ارائه پاسخی منطقی به این سؤال، باید دو اثر متضاد را در نظر بگیریم. ابتدا، باید بگوییم «نه»، زیرا به نظر احمقانه می‌آید که باور کنیم می‌توانیم حجم عظیم سالن را با ۶۰ ذره گرد و غبار نمایندگی کنیم. داده‌های مجموعه آموزشی برای پر کردن فضای سالن بسیار کم هستند. این مشاهده، که به نفرین ابعاد (curse of dimensionality) معروف است، به این واقعیت اشاره دارد که با افزایش تعداد ابعاد، تعداد نمونه‌های مورد نیاز برای پر کردن فضا نیز به سرعت و به طور تصاعدی افزایش می‌یابد. به عبارت دیگر، تعداد نقاط به سرعت زیاد می‌شود، به این معنا که تعداد نمونه‌های آموزشی لازم برای نمایندگیِ فضا به سرعت - به طور دقیق‌تر، به صورت تصاعدی - افزایش می‌یابد. نفرین ابعاد یکی از مشکلات اصلی یادگیری ماشین کلاسیک است.

نفرین ابعاد می‌گوید که وقتی تنها 60 نمونه آموزشی و 784 بعد داریم، نباید امیدی به طبقه‌بندی درست ارقام داشته باشیم... با این حال، طبقه‌بندِ نزدیک‌ترین همسایه ما همچنان کار می‌کند. نه خیلی خوب، اما بهتر از حدس تصادفی. چرا؟ علت به مجموعه داده ارقام و شباهت مثال‌های کلاس‌های مختلف به یکدیگر برمی‌گردد. همه نمونه‌های پنج به شکل 5 هستند؛ اگر نبودند، ما آن‌ها را به عنوان پنج تشخیص نمی‌دادیم. بنابراین، در حالی که فضای ارقام 784 بعد دارد، بیشتر ارقام در یک کلاس نسبتاً نزدیک به سایر ارقام همان کلاس قرار می‌گیرند. به عبارت دیگر، ذرات گرد و غباری که پنج‌ها را نشان می‌دهند، احتمالاً در یک منطقه نازک و لوله‌مانند که در سالن پیچ و تاب می‌خورد، خوشه‌بندی یا گروه‌بندی شده‌اند. سایر ارقام نیز به احتمال زیاد به همین شکل گروه‌بندی شده‌اند. به همین دلیل، نزدیک‌ترین نمونه شانس بیشتری برای متعلق بودن به همان کلاس رقم دارد تا آنچه در ابتدا با در نظر گرفتن نفرین ابعاد گمان می‌کردیم. بر اساس این مشاهده، پاسخ «نه» خود را به یک «احتمالاً» مردد ارتقا می‌دهیم.

ما درباره این اثر از نظر ریاضی صحبت می‌کنیم و می‌گوییم که داده‌های ارقام روی یک منیفلد (Manifold) قرار دارند که ابعاد مؤثر آن بسیار کمتر از 784 بعد بردارهایی است که ارقام را نشان می‌دهند. این که داده‌ها اغلب روی منیفلدهای با ابعاد پایین‌تر قرار دارند، یک مزیت است؛ البته اگر بتوانیم از این اطلاعات استفاده کنیم. مدل نزدیک‌ترین همسایه، از این اطلاعات استفاده می‌کند، زیرا داده‌های آموزشی خود مدل هستند. بعداً در کتاب، وقتی درباره شبکه‌های پیچشی صحبت کنیم، خواهیم فهمید که چنین مدل‌هایی راه‌های جدیدی برای بازنمایی ورودی‌هایشان یاد می‌گیرند، که مشابه یادگیری نحوه بازنمایی منیفلد با ابعاد پایین‌تر است که داده‌ها روی آن قرار دارند.

اما قبل از اینکه بیش از حد از عملکرد خوب طبقه‌بند نزدیک‌ترین همسایه خود با مجموعه داده ارقام هیجان‌زده شویم، بیایید با تلاش برای طبقه‌بندی تصاویر واقعی به واقعیت بازگردیم. مجموعه داده CIFAR-10 شامل 50000 تصویر رنگی کوچک 32×32 پیکسلی از 10 کلاس مختلف است، از جمله ترکیبی از وسایل نقلیه مانند هواپیماها، ماشین‌ها و کامیون‌ها، و حیوانات

مانند سگ‌ها، گربه‌ها و پرندگان. باز کردن هر یک از این تصاویر یک بردار با 3072 عنصر ایجاد می‌کند، بنابراین ما از طبقه‌بند خود می‌خواهیم که تصاویر را در یک فضای 3072 بعدی جدا کند. جدول 2-3 نشان می‌دهد که عملکرد آن چگونه است.

Training set size	Accuracy (%)	Training set size	Accuracy (%)
50,000	35.4	500	23.3
5,000	27.1	50	17.5

همانند MNIST، حدس تصادفی به دقتی حدود 10 درصد منجر می‌شود. در حالی که طبقه‌بند ما با تمام تغییرات اندازه مجموعه آموزشی عملکرد بهتری از این دارد، بهترین دقت آن کمی بیش از 35 درصد است - بسیار کمتر از 97 درصدی که با MNIST به دست آمد. آگاهی‌های تلخ مانند این باعث شد بسیاری در جامعه یادگیری ماشین تأسف بخورند که طبقه‌بندی عمومی تصاویر ممکن است فراتر از توان ما باشد. خوشبختانه، این‌طور نیست، اما هیچ‌یک از مدل‌های یادگیری ماشینی کلاسیک این کار را به خوبی انجام نمی‌دهند.

اگر به مفهوم منیفلدها فکر کنیم - ایده‌ای که داده‌ها اغلب در فضایی با ابعاد پایین‌تر از ابعاد خود داده‌ها قرار دارند - این نتایج تعجب‌آور نیستند. CIFAR-10 شامل عکس‌های دنیای واقعی است که اغلب به عنوان تصاویر طبیعی شناخته می‌شوند. تصاویر طبیعی بسیار پیچیده‌تر از تصاویر ساده مانند ارقام MNIST هستند، بنابراین باید انتظار داشته باشیم که در یک منیفلد با ابعاد بالاتر وجود داشته باشند و در نتیجه، یادگیری طبقه‌بندی آن‌ها دشوارتر باشد. اتفاقاً روش‌های عددی برای تخمین ابعاد واقعی داده‌ها وجود دارد. برای MNIST، اگرچه تصاویر در یک فضای 784 بعدی قرار دارند، داده‌ها به فضایی نزدیک به 11 بعد نزدیک‌تر هستند. برای CIFAR-10، ابعاد ذاتی به حدود 21 بعد نزدیک‌تر است، بنابراین انتظار داریم برای عملکردی مشابه MNIST به داده‌های آموزشی بسیار بیشتری نیاز داشته باشیم.

مدل‌های نزدیک‌ترین همسایه امروزه چندان استفاده نمی‌شوند. این مسئله دو علت دارد. اول، در حالی که آموزش یک مدل نزدیک‌ترین همسایه عملاً آنی است، زیرا چیزی برای آموزش وجود ندارد، استفاده از آن کند است، چون باید فاصله بین نمونه ناشناخته و هر یک از نمونه‌های مجموعه آموزشی را محاسبه کنیم. زمان این محاسبه با مجذور تعداد نمونه‌ها در مجموعه آموزشی افزایش می‌یابد. هرچه داده‌های آموزشی بیشتری داشته باشیم، انتظار داریم مدل بهتر عمل کند، اما سرعت آن کندتر می‌شود. اگر اندازه مجموعه آموزشی را دو برابر کنیم، زمان جستجو چهار برابر می‌شود.

ده‌ها سال مطالعه روی طبقه‌بندهای نزدیک‌ترین همسایه، انواع ترفندها را برای کاهش زمان یافتن نزدیک‌ترین همسایه یا k نزدیک‌ترین همسایه آشکار کرده است، اما «اثر» همچنان باقی است: افزایش تعداد نمونه‌های آموزشی، زمان استفاده از طبقه‌بند را افزایش می‌دهد.

مشکل دوم برای همه مدل‌های یادگیری ماشین کلاسیک، و همچنین شبکه‌های عصبی سنتی که در فصل چهارم بحث خواهیم کرد، مشترک است. این مدل‌ها کل‌نگر هستند، به این معنا که بردارهای ورودی خود را به عنوان یک موجودیت واحد بدون اجزا تفسیر می‌کنند. این در بسیاری از موارد کار درستی نیست. برای مثال، نوشتن عدد چهار از چندین خط تشکیل شده است و اجزای مشخصی وجود دارند که چهار را از هشت متمایز می‌کنند. مدل‌های یادگیری ماشینی کلاسیک به طور صریح درباره

این اجزا یا محل ظاهر شدن آن‌ها، یا اینکه ممکن است در چندین مکان ظاهر شوند، یاد نمی‌گیرند. اما شبکه‌های عصبی پیچشی مدرن این چیزها را یاد می‌گیرند. حال بیایید تغییر جهت دهیم و به جنگل و درختان فکر کنیم.

در فصل اول به طور خلاصه به درختان تصمیم پرداختیم، که شامل مجموعه‌ای از سوالات بله/خیر درباره یک نمونه ناشناخته هستند. شما از گره ریشه شروع می‌کنید و با پاسخ به سؤال گره، درخت را طی می‌کنید. اگر پاسخ «بله» باشد، یک سطح به سمت چپ پایین می‌روید. اگر پاسخ «خیر» باشد، به سمت راست پایین می‌روید. به پاسخ دادن به سوالات ادامه می‌دهید تا به یک برگ (گرهی بدون سؤال) برسید و برچسب موجود در گره برگ را به نمونه ناشناخته اختصاص می‌دهید. درختان تصمیم قطعی هستند؛ پس از ساخته شدن، تغییر نمی‌کنند. بنابراین، الگوریتم‌های سنتی درخت تصمیم برای یک مجموعه آموزشی یکسان، همان درخت تصمیم را بازمی‌گردانند. اغلب اوقات، درخت به خوبی کار نمی‌کند. اگر این اتفاق بیفتد، آیا کاری می‌توانیم انجام دهیم؟ بله! می‌توانیم جنگلی از درختان پرورش دهیم.

اما اگر درختان تصمیم قطعی باشند، آیا جنگل چیزی بیش از تکرار همان درخت، مانند مجموعه‌ای از کلون‌ها، نخواهد بود؟ اگر در این مسیر کار هوشمندانه‌ای انجام ندهیم، همین‌طور خواهد بود. خوشبختانه، انسان‌ها باهوش هستند. محققان حوالی سال 2000 دریافتند که معرفی تصادفی بودن، جنگلی از درختان منحصربه‌فرد را تولید می‌کند، که هر کدام نقاط قوت و ضعف خود را دارند، اما در مجموع بهتر از هر درخت تکی هستند. جنگل تصادفی مجموعه‌ای از درختان تصمیم است که هر کدام به طور تصادفی با دیگران متفاوت هستند. پیش‌بینی جنگل ترکیبی از پیش‌بینی‌های درختان آن است.

جنگل‌های تصادفی تجلی حکمت جمعی هستند.

استفاده از تصادفی بودن برای ساخت یک طبقه‌بند در ابتدا غیرمنطقی به نظر می‌رسد. اگر روز سه‌شنبه نمونه X را به مدل ارائه دهیم و بگوید که نمونه X عضوی از کلاس Y است، نمی‌خواهیم روز شنبه همان نمونه را ارائه دهیم و بگوید که عضوی از کلاس Z است. خوشبختانه، تصادفی بودن جنگل تصادفی این‌گونه عمل نمی‌کند. به یک جنگل آموزش‌دیده نمونه X را به عنوان ورودی بدهید، و همیشه کلاس Y را به عنوان خروجی می‌دهد، حتی اگر 29 فوریه باشد.

سه مرحله برای پرورش یک جنگل تصادفی وجود دارد: بَگینگ (که به آن Bootstrapping نیز می‌گویند)، انتخاب ویژگی تصادفی (Random Feature Selection) و ترکیب‌بندی (Ensembling). بگینگ و انتخاب ویژگی تصادفی به مبارزه با بیش‌برازش (Overfitting) کمک می‌کنند؛ مفهومی که در فصل اول ذکر شد. درختان تصمیم تکی مستعد بیش‌برازش هستند. هر سه مرحله با هم کار می‌کنند تا جنگلی از درختان تصمیم پرورش دهند که خروجی‌های ترکیبی آن‌ها مدلی با عملکرد بهتر تولید کند. توضیح‌پذیری بهایی است که برای این افزایش قدرت پرداخت می‌شود. یک درخت تصمیم تکی خود را با مجموعه‌ای از سوالات و پاسخ‌هایی که خروجی آن را تولید می‌کنند، توضیح می‌دهد. اما وقتی ده‌ها یا صدها درخت تصمیم خروجی‌های خود را ترکیب می‌کنند، توضیح‌پذیری از بین می‌رود، ولی در بسیاری از موارد می‌توانیم با این موضوع کنار بیاییم.

همان‌طور که چندین بار ذکر کردم، مجموعه آموزشی کلید شرطی‌سازی مدل است. این موضوع در مورد جنگل‌های تصادفی نیز صدق می‌کند. ما به عنوان نقطه شروع، یک مجموعه آموزشی داریم. با رشد جنگل، درخت تصمیم به درخت تصمیم، از مجموعه آموزشی موجود برای ایجاد مجموعه‌های آموزشی خاص هر درخت استفاده می‌کنیم که برای درخت تصمیم فعلی منحصربه‌فرد هستند. اینجا جایی است که بگینگ وارد می‌شود.

بگینگ به ساخت یک مجموعه داده جدید از مجموعه داده فعلی با نمونه‌برداری تصادفی با جایگذاری اشاره دارد. عبارت «با جایگذاری» به این معناست که ممکن است یک نمونه آموزشی را بیش از یک بار انتخاب کنیم یا اصلاً انتخاب نکنیم. این تکنیک در آمار برای درک حدود یک اندازه‌گیری استفاده می‌شود. ما از مجموعه داده مثال زیر از نمرات آزمون برای فهمیدن معنای آن استفاده خواهیم کرد:

95, 88, 76, 81, 92, 70, 86, 87, 72

یک راه برای ارزیابی عملکرد یک کلاس در آزمون، محاسبه میانگین نمرات است که با تقسیم مجموع تمام نمرات بر تعداد نمرات به دست می‌آید. مجموع نمرات 747 است و 9 نمره وجود دارد، که میانگینی برابر با 83 به ما می‌دهد. به طور جمعی، نمرات آزمون نمونه‌ای از یک فرآیند والد خیالی هستند که نمرات را برای آزمون خاص در نظر گرفته‌شده تولید می‌کند. این روش متداولی برای فکر کردن به نمرات آزمون نیست، اما روشی است که در یادگیری ماشینی برای فکر کردن به آنچه یک مجموعه داده نمایندگی می‌کند، به کار می‌رود. نمرات آزمون از گروه دیگری از دانش‌آموزان، نمونه دیگری از فرآیند والد برای این آزمون را نشان می‌دهد. اگر نمرات آزمون از کلاس‌های زیادی داشته باشیم، می‌توانیم ایده‌ای درباره میانگین واقعی نمرات آزمون یا حداقل محدوده‌ای که انتظار داریم میانگین نمرات در آن قرار گیرد، با درجه بالایی از اطمینان به دست آوریم.

می‌توانیم آزمون را به کلاس‌های مختلف زیادی بدهیم تا میانگین‌های متعددی، یکی برای هر کلاس، به دست آوریم، اما در عوض از بگینگ استفاده خواهیم کرد تا مجموعه‌های داده جدیدی از مجموعه نمرات آزمونی که داریم ایجاد کنیم و به میانگین‌های آن‌ها نگاه کنیم. برای این کار، مقادیر را از مجموعه نمرات آزمون به صورت تصادفی انتخاب می‌کنیم، بدون توجه به اینکه آیا این نمره خاص را قبلاً انتخاب کرده‌ایم یا اصلاً آن را انتخاب نکرده‌ایم. در اینجا شش مجموعه داده بوت‌استرپ‌شده از این قبیل آورده شده است:

1. 86, 87, 87, 76, 81, 81, 88, 70, 95
2. 87, 92, 76, 87, 87, 76, 87, 92, 92
3. 95, 70, 87, 92, 70, 92, 72, 70, 72
4. 88, 86, 87, 70, 81, 72, 86, 95, 70
5. 86, 86, 92, 86, 87, 86, 70, 81, 87
6. 76, 88, 88, 88, 88, 72, 86, 95, 70

میانگین‌های مربوطه به ترتیب 83.4، 86.2، 80.0، 81.7، 84.6 و 83.4 درصد هستند. کمترین مقدار 80.0 درصد و بیشترین مقدار 86.2 درصد است. این به ما دلیلی می‌دهد تا باور کنیم که تعداد زیادی نمونه، میانگینی کم و بیش در این محدوده تولید خواهد کرد.

این روشی است که یک آمارشناس ممکن است از بگینگ استفاده کند. برای ما، بخش حیاتی شش مجموعه داده جدید است که از مجموعه داده اصلی بوت‌استرپ شده‌اند. هنگام پرورش یک جنگل تصادفی، هر بار که به یک درخت تصمیم جدید نیاز داریم، ابتدا از بگینگ برای تولید یک مجموعه داده جدید استفاده می‌کنیم، سپس درخت تصمیم را با استفاده از آن مجموعه داده، نه مجموعه داده اصلی، آموزش می‌دهیم. توجه کنید که بسیاری از این شش مجموعه داده دارای مقادیر تکراری هستند. برای مثال، مجموعه داده 1 از 81 و 87 دو بار استفاده کرده است، اما هرگز 72 را استفاده نکرده است. این تصادفی‌سازی مجموعه

داده موجود به ایجاد درختان تصمیمی کمک می‌کند که رفتار متفاوتی از یکدیگر دارند، اما با آنچه مجموعه داده اصلی نمایندگی می‌کند، هم‌راستا هستند.

ترفند دومی که یک جنگل تصادفی استفاده می‌کند، آموزش درخت تصمیم بر روی یک مجموعه ویژگی است که به صورت تصادفی انتخاب‌شده است. بیایید از مجموعه داده مقابل در جدول 3-3 برای فهمیدن معنای آن استفاده کنیم. مانند همیشه، هر ردیف یک بردار ویژگی است، نمونه‌ای که برچسب کلاس صحیح آن را می‌دانیم. ستون‌ها مقادیر آن ویژگی برای هر نمونه هستند.

#	x_0	x_1	x_2	x_3	x_4	x_5
1	0.52	0.95	0.81	0.78	0.97	0.36
2	0.89	0.37	0.66	0.55	0.75	0.45
3	0.49	0.98	0.49	0.39	0.42	0.24
4	0.43	0.51	0.90	0.78	0.19	0.22
5	0.51	0.16	0.11	0.48	0.34	0.54
6	0.48	0.99	0.62	0.58	0.72	0.42
7	0.80	0.84	0.72	0.26	0.93	0.23
8	0.50	0.70	0.13	0.35	0.96	0.82
9	0.70	0.54	0.62	0.72	0.14	0.53

این مجموعه داده چه چیزی را نشان می‌دهد؟ من هیچ ایده‌ای ندارم؛ این داده ساختگی است. پاسخ طنزآمیزم یادآوری خوبی است که مدل‌های یادگیری ماشین نمی‌فهمند مجموعه‌های داده‌شان چه چیزی را نمایندگی می‌کنند. آن‌ها اعداد را بدون زمینه پردازش می‌کنند. آیا این یک مقدار پیکسل است؟ تعداد فوت مربع یک خانه؟ نرخ جرم و جنایت یک شهرستان به ازای هر صد هزار نفر؟ برای مدل یادگیری ماشین اهمیتی ندارد - همه چیز فقط اعداد هستند.

این مجموعه داده شامل نُه بردار ویژگی است، که هر کدام شش ویژگی دارند، از x_0 تا x_5. درختان تصمیم جنگل از یک زیرمجموعه تصادفی از این شش ویژگی استفاده می‌کنند. برای مثال، فرض کنید به طور تصادفی ویژگی‌های x_0، x_4 و x_5 را نگه می‌داریم. جدول 3-4 مجموعه داده‌ای را نشان می‌دهد که اکنون برای آموزش درخت تصمیم استفاده می‌شود.

#	x_0	x_4	x_5
1	0.52	0.97	0.36
2	0.89	0.75	0.45
3	0.49	0.42	0.24
4	0.43	0.19	0.22
5	0.51	0.34	0.54
6	0.48	0.72	0.42
7	0.80	0.93	0.23
8	0.50	0.96	0.82
9	0.70	0.14	0.53

هر درخت تصمیم در جنگل با نسخه‌ای بوت‌استرپ‌شده از مجموعه داده و تنها با استفاده از زیرمجموعه‌ای از ویژگی‌های موجود آموزش داده شده است. ما دو بار از تصادفی بودن استفاده کرده‌ایم تا جنگلی از درختان پرورش دهیم که همگی به طور ظریفی با یکدیگر متفاوت هستند، هم از نظر داده‌هایی که با آن‌ها آموزش دیده‌اند و هم از نظر ویژگی‌هایی که به آن‌ها توجه می‌کنند.

اکنون که یک جنگل داریم، چگونه از آن استفاده کنیم؟ اینجا سومین قطعه وارد می‌شود: ترکیب‌بندی. در موسیقی، یک آنسامبل مجموعه‌ای از نوازندگان است که سازهای متنوعی می‌نوازند. جنگل تصادفی نیز یک آنسامبل است، که در آن هر درخت تصمیم یک نوازنده متفاوت است که ساز متفاوتی می‌نوازد. یک آنسامبل موسیقی یک خروجی واحد، یعنی موسیقی، را با ترکیب نت‌های نواخته‌شده توسط هر ساز تولید می‌کند. به همین ترتیب، یک جنگل تصادفی یک خروجی واحد، یعنی یک برچسب کلاس، را با ترکیب برچسب‌های تولیدشده توسط هر درخت تصمیم تولید می‌کند، معمولاً با رأی‌گیری مانند یک طبقه‌بند k-نزدیک‌ترین همسایه. ما برچسب برنده را به ورودی اختصاص می‌دهیم.

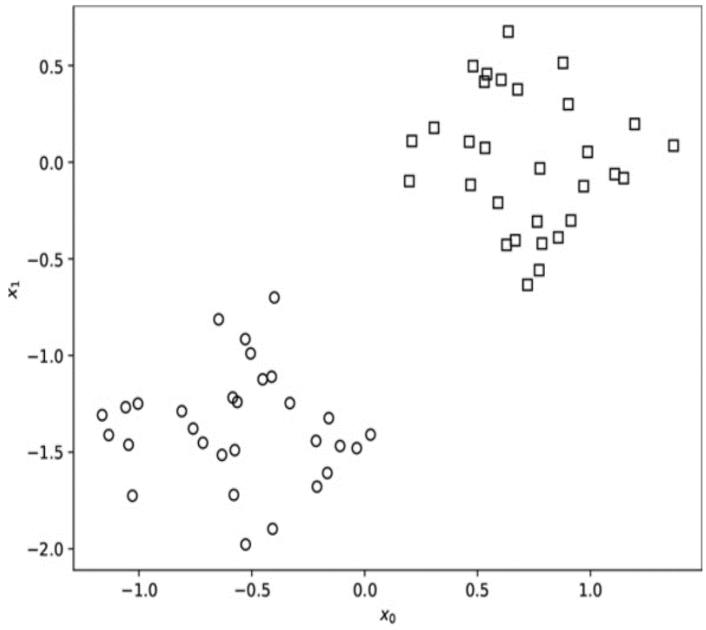
برای مثال، اگر بخواهیم از جنگل تصادفی برای طبقه‌بندی نمونه X استفاده کنیم و 100 درخت در جنگل تصادفی (که قبلاً آموزش دیده‌اند) وجود داشته باشد، نمونه X را به هر درخت می‌دهیم. درختان می‌دانند از کدام زیرمجموعه ویژگی‌های نمونه X برای رسیدن به یک برگ با یک برچسب استفاده کنند. اکنون 100 برچسب کلاس ممکن داریم؛ خروجیِ 100 درخت تصمیم جنگل. اگر 78 درخت نمونه X را به کلاس Y اختصاص دهند، جنگل تصادفی اعلام می‌کند که نمونه X متعلق به کلاس Y است.

تخصیص تصادفی ویژگی‌ها به درختان، همراه با مجموعه‌های داده بوت‌استرپ‌شده و رأی‌گیری آنسامبل، به جنگل تصادفی قدرت می‌بخشد. ترکیب‌بندی ایده‌ای جذاب و شهودی است که محدود به جنگل‌های تصادفی نیست. هیچ چیزی مانع از این نمی‌شود که انواع مدل‌های مختلف را روی یک مجموعه داده یکسان آموزش دهیم و سپس پیش‌بینی‌های آن‌ها را به نوعی ترکیب کنیم تا به یک نتیجه مشترک درباره یک نمونه ورودی برسیم. هر یک از مدل‌ها نقاط قوت و ضعف خود را دارند. وقتی ترکیب شوند، نقاط قوت معمولاً کیفیت خروجی را افزایش می‌دهند و باعث می‌شوند جمع از اجزا بزرگ‌تر شود.

ما یک مدل یادگیری ماشین کلاسیک دیگر برای بررسی داریم، ماشین بردار پشتیبان (SVM). پس از آن، مدل‌ها را در مقابل یکدیگر قرار می‌دهیم تا بینشی درباره نحوه رفتار آن‌ها به دست آوریم و یک خط مبنا برای مقایسه عملکرد شبکه‌های عصبی فراهم کنیم.

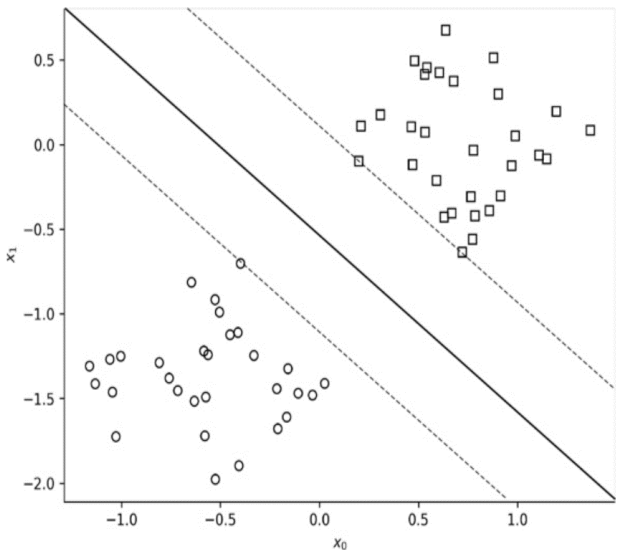
برای درک ماشین‌های بردار پشتیبان، باید چهار مفهوم را فهمید: حاشیه‌ها (Margins)، بردارهای پشتیبان (Support Vectors)، بهینه‌سازی (Optimization) و هسته‌ها (Kernels). ریاضیات این موضوع کمی پیچیده است، حتی برای افراد آشنا با ریاضیات، اما ما این را کنار می‌گذاریم و به جای آن بر درک مفهومی تمرکز می‌کنیم.

ماشین‌های بردار پشتیبان بهتر است به صورت بصری درک شوند، بنابراین با مجموعه داده موجود در شکل 3-3 شروع می‌کنیم. این یک مجموعه داده دو کلاسی (دایره‌ها و مربع‌ها) با بردارهای ویژگی دوبعدی، ویژگی‌های x_0 و x_1، است.



ساخت یک طبقه‌بند برای این مجموعه داده ساده است، زیرا یک خط به راحتی می‌تواند مجموعه داده را بر اساس کلاس جدا کند، به طوری که همه مربع‌ها در بالا و سمت راست و همه دایره‌ها در پایین و سمت چپ قرار گیرند. اما این خط کجا باید قرار گیرد؟ تعداد بی‌نهایت خط وجود دارد که می‌توانیم استفاده کنیم. برای مثال، ممکن است خط را درست زیر همه مربع‌ها قرار دهیم. این خط کلاس‌ها را جدا می‌کند، اما اگر با نمونه‌ای از کلاس مربع مواجه شویم که هنگام استفاده از طبقه‌بند درست زیر خط قرار گیرد، اشتباه خواهیم کرد و نمونه را به کلاس دایره اختصاص می‌دهیم، زیرا زیر خطی است که ما اعلام کردیم کلاس‌ها را جدا می‌کند. به طور مشابه، اگر خط را درست بالای همه دایره‌ها قرار دهیم، ممکن است یک نمونه جدید که در واقع دایره است را مربع بنامیم، زیرا کمی بالای آن خط قرار گرفته است.

با توجه به آنچه بر اساس داده‌های آموزشی می‌دانیم، باید خط جداکننده را تا حد امکان از هر گروه دور قرار دهیم. اینجاست که مفهوم حاشیه (margin) مطرح می‌شود. ماشین‌های بردار پشتیبان (SVMs) به دنبال بیشینه‌کردن حاشیه بین دو گروه هستند، یعنی یافتن جایی که بیشترین فاصله بین کلاس‌ها وجود دارد. وقتی حاشیه به حداکثر برسد، مرز (در اینجا یک خط) را در وسط حاشیه قرار می‌دهند، زیرا این منطقی‌ترین کار بر اساس اطلاعات موجود در داده‌های آموزشی است.



شکل 3-4 داده‌های آموزشی را با سه خط اضافه نشان می‌دهد. خطوط چین‌دار محدوده حاشیه را مشخص می‌کنند و خط پررنگ پیوسته، مرز تعیین‌شده توسط SVM را نشان می‌دهد که برای بیشینه‌کردن فاصله بین کلاس‌ها قرار گرفته است. این موقعیت بهینه برای خط است تا خطاهای برچسب‌زنی بین دو کلاس را به حداقل برساند. به طور خلاصه، این تمام کاری است که یک SVM انجام می‌دهد.

سه بخش دیگر یک ماشین بردار پشتیبان (SVM) یعنی بردارهای پشتیبان (Support Vectors)، بهینه‌سازی (Optimization) و هسته‌ها (Kernels) برای یافتن حاشیه‌ها و خط جداکننده استفاده

می‌شوند. در شکل فوق، توجه کنید که خطوط چین‌دار از برخی نقاط داده عبور می‌کنند. این نقاط، بردارهای پشتیبان هستند که الگوریتم برای تعریف حاشیه پیدا می‌کند.

این بردارهای پشتیبان از کجا می‌آیند؟ به یاد داشته باشید که نقاط موجود در شکل، نمایانگر بردارهای ویژگی خاصی در مجموعه آموزشی هستند. بردارهای پشتیبان، اعضایی از مجموعه آموزشی هستند که از طریق یک الگوریتم بهینه‌سازی پیدا می‌شوند. بهینه‌سازی به معنای یافتن بهترین گزینه بر اساس معیارهای مشخص است. الگوریتم بهینه‌سازی مورد استفاده در SVM، بردارهای پشتیبان را پیدا می‌کند که حاشیه حداکثری و در نهایت خط جداکننده را تعریف می‌کنند.

در فصل یک، زمانی که درباره برازش منحنی (Curve Fitting) صحبت کردیم، از یک الگوریتم بهینه‌سازی استفاده شد و در آموزش شبکه‌های عصبی نیز دوباره از آن استفاده خواهیم کرد.

تقریباً به پایان رسیده‌ایم؛ فقط یک مفهوم دیگر در SVM باقی مانده است: هسته‌ها. برخلاف هسته‌های ذرت یا هسته مرکزی سیستم عامل کامپیوتر شما، هسته‌های ریاضیاتی دو چیز را به هم مرتبط می‌کنند - در اینجا، دو بردار ویژگی. مثال موجود در شکل 3-4 از یک هسته خطی (Linear kernel) استفاده می‌کند، یعنی بردارهای ویژگی داده‌های آموزشی را به همان شکل اصلی به کار می‌برد.

ماشین‌های بردار پشتیبان از انواع مختلفی از هسته‌ها برای ارتباط دادن دو بردار ویژگی پشتیبانی می‌کنند، اما هسته خطی رایج‌ترین است. نوع دیگری به نام هسته گاوسی (Gaussian Kernel) یا با نام طولانی‌تر و پرطمطراق‌تر، هسته تابع پایه شعاعی (Radial Basis Function Kernel) اغلب در مواردی کمک می‌کند که هسته خطی به دلیل رابطه متفاوت بین بردارهای ویژگی با شکست مواجه می‌شود.

هسته، بردارهای ویژگی را به یک بازنمایی متفاوت تبدیل می‌کند؛ ایده‌ای که در قلب کاری که شبکه‌های عصبی پیچشی انجام می‌دهند، قرار دارد. یکی از مشکلاتی که باعث شد یادگیری ماشین کلاسیک برای مدت‌ها به بن‌بست بخورد، این بود که داده‌های خام ارائه‌شده به مدل‌ها آنقدر پیچیده بودند که مدل نمی‌توانست تمایز معناداری بین کلاس‌ها ایجاد کند. این موضوع به ایده منیفلدها و بُعد ذاتی (Intrinsic Dimensionality) که در بحث نزدیک‌ترین همسایه‌ها مطرح شد، مرتبط است.

متخصصان یادگیری ماشین کلاسیک تلاش زیادی می‌کردند تا تعداد ویژگی‌های مورد نیاز مدل را به حداقل برسانند و ویژگی‌ها را به کوچک‌ترین مجموعه ضروری برای تمایز بین کلاس‌ها تقلیل دهند. این روش بسته به الگوریتم مورد استفاده، انتخاب ویژگی (Feature Selection) یا کاهش ابعاد (Dimensionality Reduction) نامیده می‌شد. به طور مشابه، به ویژه در SVM ها، از هسته‌ها برای نگاشت بردارهای ویژگی داده‌شده به یک بازنمایی جدید استفاده می‌شد تا تفکیک کلاس‌ها آسان‌تر شود.

این رویکردها تلاش‌هایی انسان‌محور بودند؛ ما ویژگی‌ها یا هسته‌ها را انتخاب می‌کردیم به امید اینکه مسئله را قابل‌مدیریت‌تر کنند. اما همانطور که خواهیم دید، یادگیری عمیق مدرن اجازه می‌دهد داده‌ها خودشان در یادگیری بازنمایی‌های جدید از اطلاعاتی که در بر دارند، سخن بگویند.

در عمل، آموزش یک ماشین بردار پشتیبان به معنای یافتن مقادیر مناسب برای پارامترهای مرتبط با هسته مورد استفاده است. اگر هسته خطی باشد، مانند مثال قبلی، تنها یک مقدار به نام C وجود دارد که باید پیدا شود. این یک عدد مانند 1 یا 10 است

که بر عملکرد SVM تأثیر می‌گذارد. اگر از هسته گاوسی استفاده شود، علاوه بر C پارامتر دیگری به نام گاما (γ) نیز وجود دارد. هنر آموزش یک SVM شامل یافتن مقادیر جادویی‌ای است که برای مجموعه داده مورد نظر بهترین کارایی را دارند.

مقادیر جادویی مورد استفاده یک مدل، هایپرپارامترهای (Hyperparameters) آن هستند. شبکه‌های عصبی هایپرپارامترهای بسیار بیشتری نسبت به SVMها دارند. با این حال، تجربه من نشان داده که تنظیم یک شبکه عصبی، به ویژه یک شبکه عصبی عمیق مدرن، اغلب آسان‌تر از تنظیم یک ماشین بردار پشتیبان است. در اینجا صادقانه به سوگیری خود اعتراف می‌کنم؛ دیگران ممکن است مخالف باشند.

ماشین‌های بردار پشتیبان از نظر ریاضیاتی بسیار زیبا و ظریف هستند. متخصصان از این ظرافت ریاضی برای تنظیم هایپرپارامترها و هسته مورد استفاده، همراه با مجموعه‌ای از روش‌های قدیمی آماده‌سازی داده‌ها، بهره می‌برند تا مدلی با عملکرد خوب بسازند که در دنیای واقعی نیز کارایی داشته باشد. هر مرحله از این فرآیند به شهود و تجربه انسانی که مدل را می‌سازد وابسته است. اگر فردی دانش و تجربه کافی داشته باشد و مجموعه داده نیز مناسب باشد، احتمال موفقیت وجود دارد، اما موفقیت تضمین‌شده نیست.

از طرف دیگر، شبکه‌های عصبی عمیق بزرگ، تا حدی دست‌وپاگیر هستند و موفقیت یا شکست آنها کاملاً به داده‌های خام ورودی بستگی دارد. با این حال، همین که این شبکه‌ها با حداقل پیش‌فرض‌ها به مسئله نزدیک می‌شوند، به آنها امکان می‌دهد تا بر عناصری از داده‌ها تعمیم یابند که برای انسان‌ها غیرقابل درک است. به نظر من، همین ویژگی است که باعث می‌شود شبکه‌های عصبی مدرن کارهایی انجام دهند که قبلاً تقریباً غیرممکن تصور می‌شد.

ماشین‌های بردار پشتیبان در اصل طبقه‌بندهای دودویی هستند: یعنی بین دو کلاس تمایز قائل می‌شوند، مانند مجموعه داده شکل 3-3. اما گاهی نیاز داریم بین بیش از دو کلاس تفاوت قائل شویم. چگونه می‌توان این کار را با SVM انجام داد؟

برای تعمیم SVMها به مسائل چندکلاسه، دو راه‌حل وجود دارد. فرض کنید در مجموعه داده‌مان 10 کلاس داریم. در روش اول (یک در مقابل بقیه یا One vs Rest)، 10 SVM آموزش می‌دهیم. اولین SVM سعی می‌کند کلاس 0 را از 9 کلاس دیگر جدا کند، دومی کلاس 1 را از بقیه جدا می‌کند و به همین ترتیب ادامه می‌یابد. برای طبقه‌بندی یک نمونه ناشناخته، آن را به همه SVMها می‌دهیم و کلاسی که بیشترین مقدار تابع تصمیم را تولید کرده باشد به عنوان نتیجه انتخاب می‌شود.

روش دوم (یک در مقابل یک یا One vs One) برای هر جفت کلاس ممکن یک SVM جداگانه آموزش می‌دهد. نمونه ناشناخته به همه مدل‌ها داده می‌شود و کلاسی که بیشترین رأی را آورده باشد انتخاب می‌شود. این روش وقتی تعداد کلاس‌ها زیاد باشد عملی نیست. مثلاً برای 10 کلاس در CIFAR-10 به 45 SVM مختلف نیاز داریم و برای 1000 کلاس در ImageNet این عدد به 499500 مدل می‌رسد که آموزش آنها زمان بسیار زیادی می‌طلبد.

ماشین‌های بردار پشتیبان در دهه 1990 و اوایل 2000 که قدرت محاسباتی محدودتر بود گزینه مناسبی محسوب می‌شدند و همین موضوع باعث شد برای مدت‌ها جایگزین شبکه‌های عصبی شوند. اما با ظهور یادگیری عمیق، به نظر من دیگر دلیلی برای استفاده از SVMها وجود ندارد.

بیایید سه مدل کلاسیک بررسی‌شده در این فصل را روی یک مجموعه داده متن‌باز از ردپاهای دایناسورها آزمایش کنیم. این داده‌ها از مقاله سال 2022 با عنوان «یک رویکرد یادگیری ماشین برای تشخیص ردپاهای دایناسورهای تروپود (Theropod) و

اورنیتیشین (Ornithischian)» اثر جنس ان. لالنزاک، آنتونی رومیلو و پیتر ال. فالکینگهام گرفته شده است. تصاویر ردپاها تحت مجوز Creative Commons CC BY 4.0 منتشر شده‌اند که استفاده مجدد با ذکر منبع را مجاز می‌کند.

نمونه‌هایی از مجموعه داده در شکل 5-3 نشان داده شده‌اند:

- ردپاهای تروپودها (مثل تی-رکس) در ردیف بالا

- ردپاهای اورنیتیشین‌ها (مثل دایناسورهای منقاراردکی مانند Hadrosaurs) در ردیف پایین

پیش‌پردازش داده‌ها:

1. تصاویر داده شده به مدل معکوس شدند تا پس‌زمینه سیاه با ردپاهای سفید نمایش داده شوند

2. اندازه تصاویر به 40×40 پیکسل تغییر یافت

3. هر تصویر به یک بردار 1600 بعدی تبدیل شد (40 × 40 = 1600)

مشخصات مجموعه داده:

- تعداد نمونه‌های آموزشی: 1336

- تعداد نمونه‌های آزمایشی: 335

- این حجم داده با استانداردهای امروزی نسبتاً کوچک محسوب می‌شود

این مجموعه داده محدود، چالشی جالب برای آزمایش مدل‌های کلاسیک ایجاد می‌کند، چرا که باید با حجم اطلاعات نسبتاً کمی کارایی خوبی نشان دهند. تبدیل تصاویر به بردارهای یک‌بعدی نیز رویکردی متداول در پردازش تصویر با روش‌های کلاسیک است. من مدل‌های زیر را آموزش دادم:

- نزدیک‌ترین همسایه (k = 1, 3, 7)
- جنگل تصادفی با 300 درخت
- ماشین بردار پشتیبان خطی
- ماشین بردار پشتیبان تابع پایه شعاعی

پس از فرآیند آموزش، مدل‌ها را با استفاده از مجموعه آزمایشی جدا نگه داشته شده ارزیابی کردم. همچنین مدت زمان لازم برای آموزش هر مدل و همچنین زمان تست هر مدل پس از آموزش را زمان‌سنجی کردم. استفاده از یک مدل پس از آموزش، استنتاج نامیده می‌شود، به این معنا که زمان استنتاج را روی مجموعه آزمایشی پیگیری کردم.

توجه: این یک کتاب برنامه‌نویسی نیست، اما اگر با برنامه‌نویسی (به ویژه پایتون) آشنا هستید، می‌توانید با من از طریق جیمیل rkneuselbooks@gmail.com تماس بگیرید تا مجموعه داده و کدها را برایتان ارسال کنم.

جدول 5-3 نتایج را نشان می‌دهد. همانطور که انتظار می‌رود، ارزیابی عملکرد مدل یکی از اجزای حیاتی فرآیند یادگیری ماشین محسوب می‌شود.

Model	ACC	MCC	Train	Test
RF300	83.3	0.65	1.5823	0.0399
RBF SVM	82.4	0.64	0.9296	0.2579
7-NN	80.0	0.58	0.0004	0.0412
3-NN	77.6	0.54	0.0005	0.0437
1-NN	76.1	0.50	0.0004	0.0395
Linear SVM	70.7	0.41	2.8165	0.0007

ستون اول در سمت چپ، مدل‌ها را مشخص می‌کند: از بالا به پایین،

جنگل تصادفی (Random Forest)،

ماشین بردار پشتیبان با تابع پایه شعاعی (RBF SVM)،

نزدیک‌ترین همسایه‌ها (با 7، 3 و1 همسایه)،

و ماشین بردار پشتیبان خطی (Linear SVM)

ستون‌های ACC (دقت) و MCC (ضریب همبستگی متیوز) معیارهایی هستند که از ماتریس سردرگمی محاسبه می‌شوند. این ماتریس مهم‌ترین ابزار در جعبه‌ابزار متخصصان یادگیری ماشین برای ارزیابی مدل‌هاست (به فصل یک مراجعه کنید).

برای طبقه‌بندهای دودویی مانند مدل‌های حاضر، ماتریس سردرگمی موارد زیر را شمارش می‌کند:

- تعداد دفعاتی که یک نمونه تروپود به درستی شناسایی شده است
- تعداد دفعاتی که یک نمونه اورنیتیشین به درستی تشخیص داده شده است
- تعداد دفعاتی که هر کلاس با دیگری اشتباه گرفته شده است

به صورت بصری، ماتریس سردرگمی برای یک مدل دودویی به این شکل است:

	اورنیتیشین	تروپود
اورنیتیشین	TN	FP
تروپود	FN	TP

سطرها برچسب کلاس واقعی، از مجموعه آزمایشیِ جدا نگه‌داشته شده هستند. ستون‌ها برچسب‌هایی هستند که مدل اختصاص داده است. سلول‌ها تعداد دفعاتی را نشان می‌دهند که هر ترکیب از برچسب واقعی و برچسب اختصاص‌یافته توسط مدل رخ داده است. حروف، روش استاندارد برای اشاره به معنای اعداد درون سلول‌ها هستند: TN منفی صحیح (True Negative)، TP مثبت صحیح (True Positive)، FP مثبت کاذب (False Positive) و FN منفی کاذب (False Negative).

برای مدل‌های ردپای دایناسورها، تروپود کلاس 1 یا کلاس «مثبت» در نظر گرفته می‌شود و اورنیتیشین کلاس 0 یا کلاس «منفی» است. تعداد دفعاتی که مدل یک ردپای اورنیتیشین را به درستی «اورنیتیشین» تشخیص داده است، مقدار TN است. به طور مشابه، مقدار TP نشان‌دهنده تعداد دفعاتی است که مدل در تشخیص یک ردپای تروپود درست عمل کرده است. هدف این است که مقادیر TN و TP تا حد امکان بالا باشند، در حالی که FP و FN که نشان‌دهنده خطاها هستند، تا حد امکان کم باشند.

در جدول، ACC به دقت (Accuracy) اشاره دارد: چند بار برچسب اختصاص‌یافته توسط طبقه‌بند درست بوده است؟ اگرچه دقت طبیعی‌ترین معیار به نظر می‌رسد، اما همیشه بهترین معیار نیست، به‌ویژه زمانی که تعداد نمونه‌های هر کلاس تقریباً برابر نباشد. جنگل تصادفی (Random Forest) از نظر دقت بهترین عملکرد را داشت و توانست بیش از 83 تصویر از هر 100 تصویر آزمایشی را به درستی طبقه‌بندی کند. SVM خطی بدترین عملکرد را داشت و تنها در حدود 71 مورد از 100 مورد پاسخ درست داد. حدس تصادفی در 50 درصد موارد درست خواهد بود (چون دو کلاس داریم)، اما حتی SVM خطی نیز از تصاویر ردپاها یاد گرفته بود.

دقت را بر اساس سلول‌های ماتریس سردرگمی (Confusion Matrix) تعریف می‌کنیم:

$$\text{دقت} = (TP + TN) / (TP + TN + FP + FN)$$

ستون MCC که مخفف ضریب همبستگی متیوز (Matthews Correlation Coefficient) است، یک معیار جدید معرفی می‌کند. این معیار ترکیبی متفاوت از چهار عدد در ماتریس سردرگمی است. MCC معیار مورد علاقه من برای ارزیابی طبقه‌بندهاست و به‌طور فزاینده‌ای به‌عنوان بهترین معیار تک‌عددی برای سنجش عملکرد مدل شناخته می‌شود. (این معیارها برای مدل‌های یادگیری عمیق پیشرفته نیز کاربرد دارند.)

جدول 5-3 بر اساس MCC مرتب‌شده است که در این مثال، به تصادف بر اساس ACC نیز مرتب شده است. برای یک مدل دودویی، کمترین مقدار ممکن MCC برابر 1- و بیشترین مقدار 1 است. حدس تصادفی مقدار MCC برابر 0 تولید می‌کند. مقدار 1 به این معنی است که مدل هیچ خطایی ندارد و مقدار 1- (که در عمل هرگز اتفاق نمی‌افتد) به این معنی است که مدل کاملاً اشتباه عمل می‌کند: مثلاً در این مورد، تمام ردپاهای تروپود را اورنیتیشین و تمام ردپاهای اورنیتیشین را تروپود برچسب

می‌زند. اگر یک طبقه‌بند کاملاً اشتباه دارید، کافی است برچسب‌های خروجی را جابه‌جا کنید تا به یک طبقه‌بند کاملاً درست تبدیل شود.

ستون‌های Train و Test زمان بر حسب ثانیه را نشان می‌دهند. ستون Train به ما می‌گوید که آموزش مدل چقدر طول کشیده است. مدل‌های «نزدیک‌ترین همسایه» عملاً از نظر آموزش زمانی نمی‌برند (کمتر از یک میلی‌ثانیه)، زیرا چیزی برای آموزش وجود ندارد. به یاد داشته باشید که مدل نزدیک‌ترین همسایه خود مجموعه آموزشی است و نیازی به یادگیری پارامترها ندارد.

کُندترین مدل از نظر زمان آموزش، SVM خطی بود. جالب اینجاست که مدل پیچیده‌تر تابع پایه شعاعی در حدود یک‌سوم زمان SVM خطی آموزش دید (تفاوتی که می‌توان آن را به نحوه پیاده‌سازی این مدل‌ها در کد نسبت داد). بعد از آن، کُندترین مدل، جنگل تصادفی بود.

این موضوع منطقی است زیرا 300 درخت تصمیم‌گیری در جنگل وجود داشت و هر یک از آن‌ها باید به طور مستقل آموزش داده می‌شد.

زمان استنتاج (در ستون «Test یا همان آزمون») تقریباً بین مدل‌های «نزدیک‌ترین همسایه» و «جنگل تصادفی» یکسان بود. مدل‌های SVM به ترتیب کند (تابع پایه شعاعی) و بسیار سریع (خطی) بودند که بازتابی از تفاوت‌های پیاده‌سازی آن‌هاست. توجه کنید که مدل‌های نزدیک‌ترین همسایه زمان بیشتری برای استفاده نسبت به آموزش می‌گیرند. این برعکس سناریوی معمول است، به‌ویژه برای شبکه‌های عصبی، که بعداً در کتاب خواهیم دید. معمولاً آموزش کند است اما فقط یک بار نیاز به انجام دارد، در حالی که استنتاج سریع است. برای مدل‌های نزدیک‌ترین همسایه، هرچه مجموعه آموزش بزرگ‌تر باشد، زمان استنتاج کندتر می‌شود - که نقطه ضعف بزرگی محسوب می‌شود.

دو نکته اصلی را می‌توان از این تمرین برداشت کرد:

1. درک کلی از عملکرد مدل‌های کلاسیک، که به عنوان پایه برای مقایسه با یک شبکه عصبی در فصل 4 استفاده خواهیم کرد.

2. حتی مدل‌های کلاسیک نیز می‌توانند روی این مجموعه داده خاص عملکرد خوبی داشته باشند.

عملکرد مدل‌های کلاسیک هم‌تراز با متخصصان انسانی (یعنی دیرینه‌شناسان) بود که آن‌ها نیز طرح‌های ردپای دایناسورها را برچسب‌زنی کرده بودند. بر اساس مقاله اصلی لالنزاک و همکارانش که مجموعه داده دایناسور از آن گرفته شده، متخصصان انسانی تنها در 57 درصد موارد تشخیصِ درست داده بودند. همچنین به آن‌ها اجازه داده شده بود که ردپاها را به عنوان «مبهم» برچسب‌زنی کنند؛ امتیازی که مدل‌ها ندارند. مدل‌ها همیشه یک دسته‌بندی انجام می‌دهند و گزینه‌ای مانند «نمی‌دانم» ندارند. البته می‌توان برخی از انواع مدل‌ها را وادار به چنین اظهارنظری کرد، اما مدل‌های کلاسیک این فصل برای این کار مناسب نیستند.

آیا مدل‌های کلاسیک، هوش مصنوعی نمادین هستند یا ارتباط‌گرا؟ آیا اصلاً می‌توان آن‌ها را هوش مصنوعی دانست؟ آیا یادگیری دارند، یا صرفاً ترفندهایی ریاضیاتی هستند؟ پاسخ به این پرسش‌ها در ادامه می‌آید.

در فصل اول، رابطه بین هوش مصنوعی، یادگیری ماشین و یادگیری عمیق را به صورت مجموعه‌های تو در تو توصیف کردم، به طوری که یادگیری عمیق شکلی از یادگیری ماشین و یادگیری ماشین شکلی از هوش مصنوعی است. این توصیف برای اکثر

افراد مناسب است و با تاریخچه ارائه‌شده در فصل دوم نیز همخوانی دارد. از این منظر، مدل‌های کلاسیک این فصل شکلی از هوش مصنوعی محسوب می‌شوند.

اما آیا مدل‌های کلاسیک، هوش مصنوعی نمادین هستند یا ارتباط‌گرا؟ به نظر من هیچ‌کدام. آن‌ها هوش مصنوعی نمادین نیستند، چون قوانین یا گزاره‌های منطقی را دستکاری نمی‌کنند؛ و ارتباط‌گرا هم نیستند، چون از شبکه‌ای از واحدهای ساده استفاده نمی‌کنند که با پردازش داده‌ها، ارتباطات مناسب را یاد بگیرند. در عوض، این مدل‌ها را نوعی «برازش منحنی پیشرفته» می‌دانم - یعنی خروجی یک الگوریتم که با فرآیند بهینه‌سازی، تابعی را تولید می‌کند که بهترین تطابق را با داده‌های آموزشی دارد و امیدواریم با داده‌های واقعی هم سازگار باشد.

برای یک ماشین بردار پشتیبان، این تابع، ساختار مدل بر اساس بردارهای پشتیبانی است که در فرآیند بهینه‌سازی شناسایی می‌شوند. تابع یک درخت تصمیم نیز توسط الگوریتمی خاص تولید می‌شود که داده‌های آموزشی را به طور مکرر به زیرگروه‌های کوچک‌تر تقسیم می‌کند تا به برگ‌هایی برسد که (معمولاً) فقط نمونه‌هایی از یک کلاس خاص دارند. جنگل‌های تصادفی نیز صرفاً مجموعه‌ای از چنین توابعی هستند که به صورت موازی عمل می‌کنند.

دسته‌بندهای درختی تقریباً شکلی از «برنامه‌نویسی ژنتیک» هستند. برنامه‌نویسی ژنتیک، کد کامپیوتری را با شبیه‌سازی فرآیند تکامل از طریق انتخاب طبیعی ایجاد می‌کند، که در آن «تناسب بهتر» به معنای «راه‌حل بهتری برای مسئله» است. در واقع، برنامه‌نویسی ژنتیک نوعی الگوریتم تکاملی است و الگوریتم‌های تکاملی - همراه با الگوریتم‌های هوش جمعی - بهینه‌سازی‌های قوی و عمومی را پیاده‌سازی می‌کنند. برخی افراد الگوریتم‌های تکاملی و هوش جمعی را هوش مصنوعی می‌دانند. من اینطور فکر نمی‌کنم، هرچند در کارهایم زیاد از آن‌ها استفاده می‌کنم. جمعیت‌ها یاد نمی‌گیرند؛ آن‌ها فضای جواب‌های ممکن برای یک مسئله را جستجو می‌کنند.

مدل‌های «نزدیک‌ترین همسایه» حتی ساده‌تر هستند؛ هیچ تابعی برای ساخت وجود ندارد. اگر تمام داده‌های ممکن تولیدشده توسط یک فرآیند مادر (یعنی آن چیزی که بردارهای ویژگی مورد مدلسازی ما را ایجاد می‌کند) را در اختیار داشته باشیم، اساساً نیازی به مدل نداریم. برای اختصاص یک برچسب کلاس به یک بردار ویژگی، کافیست آن را در «دفتر تلفن» بردارهای ویژگی جستجو کرده و برچسب مربوطه را برگردانیم. از آنجا که همه بردارهای ویژگی ممکن با برچسب‌هایشان را داریم، چیزی برای تقریب زدن وجود ندارد و هر بردار ویژگی که در دنیای واقعی با آن مواجه شویم، قطعاً در این دفتر موجود خواهد بود. اما اگر دسترسی به تمام بردارهای ویژگی ممکن برای مسئله مورد نظر میسر نباشد، مدل نزدیک‌ترین همسایه، نزدیک‌ترین بردار ویژگی موجود در «دفتر تلفن ناقص» (که توسط داده‌های آموزشی نمایش داده می‌شود) را استفاده می‌کند.

مثالی بزنیم: فرض کنید در شهری با 3000 نفر جمعیت زندگی می‌کنیم که اطلاعات تمام آن‌ها در دفتر تلفن موجود است. (آیا چنین دفتر تلفن‌هایی هنوز وجود دارند؟ اگر نه، فرض کنید وجود دارند.) اگر بخواهیم شماره تلفن «نوسمو کینگ» را پیدا کنیم، به بخش «کینگ» در دفتر مراجعه کرده و تا رسیدن به «نوسمو» پیش می‌رویم تا شماره را بیابیم. حال فرض کنید به جای لیست کامل 3000 نفر، فقط اطلاعات 300 نفر به صورت تصادفی در دفتر موجود است. ما همچنان می‌خواهیم شماره تلفن نوسمو کینگ (برچسب کلاس) را بدانیم، اما این نام در دفتر نیست. با این حال، یک «بِرگ آر. کینگ» در لیست وجود دارد. احتمال زیادی دارد که بِرگ با نوسمو نسبت خانوادگی داشته باشد (چون نام خانوادگی مشترک دارند)، بنابراین شماره

بِرگ را به جای نوسمو برمی‌گردانیم. واضح است که هرچه دفتر تلفن کامل‌تر باشد، شانس پیدا کردن نام مورد نظر یا فردی از همان خانواده بیشتر می‌شود. این دقیقاً همان کاری است که یک مدل نزدیک‌ترین همسایه انجام می‌دهد.

خلاصه‌کنیم: ماشین‌های بردار پشتیبان، درخت‌های تصمیم و جنگل‌های تصادفی از داده‌ها برای تولید توابعی بر اساس الگوریتم‌های از پیش طراحی‌شده توسط انسان استفاده می‌کنند. به نظر من، این‌ها نه هوش مصنوعی نمادین هستند و نه ارتباط‌گرا، بلکه صرفاً نوعی برازش منحنی یا به بیان دقیق‌تر، بهینه‌سازی محسوب می‌شوند. مدل‌های نزدیک‌ترین همسایه حتی وضعیت بدتری دارند؛ در این مورد، اصلاً تابعی وجود ندارد.

این به آن معنا نیست که هوش مصنوعی نامعتبر است، اما نشان می‌دهد آنچه متخصصان هنگام صحبت از «هوش مصنوعی» در ذهن دارند، احتمالاً با تصور عامه مردم از «هوش مصنوعی» متفاوت است.

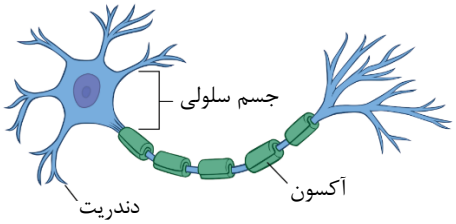
با این حال، جای نگرانی نیست. یک مدل یادگیری ماشین وجود دارد که شایسته برچسب «ارتباط‌گرا» است: شبکه عصبی. این مدل در قلب انقلاب هوش مصنوعی قرار دارد و واقعاً توانایی یادگیری از داده‌ها را دارد. پس بیایید مدل‌های کلاسیک و هوش مصنوعی نمادین را کنار بگذاریم و توجه خود را به شبکه‌های عصبی معطوف کنیم.

اصطلاحات کلیدی: بگینگ (Bagging)، نفرین ابعاد (Curse of Dimensionality)، الگوریتم تکاملی (evolutionary algorithm)، منفی کاذب (False Negative)، مثبت کاذب (False Positive)، برنامه‌نویسی ژنتیک (Genetic Programming)، ابرپارامترها یا هایپرپارامترها (Hyperparameters)، استنتاج (Inference)، منیفُلد (Manifold)، متریک یا معیار سنجش (Metric)، نزدیک‌ترین همسایه (Nearest Neighbor)، یک در مقابل یک (One vs One)، یک در مقابل بقیه (One vs Rest)، جنگل تصادفی (Random Forest)، ماشین بردار پشتیبان (Support Vector Machine)، هوش جمعی (Swarm Intelligence)، منفی صحیح (True Negative)، مثبت صحیح (True Positive)

فصل چهارم: شبکه‌های عصبی (هوش مصنوعی مغز مانند)

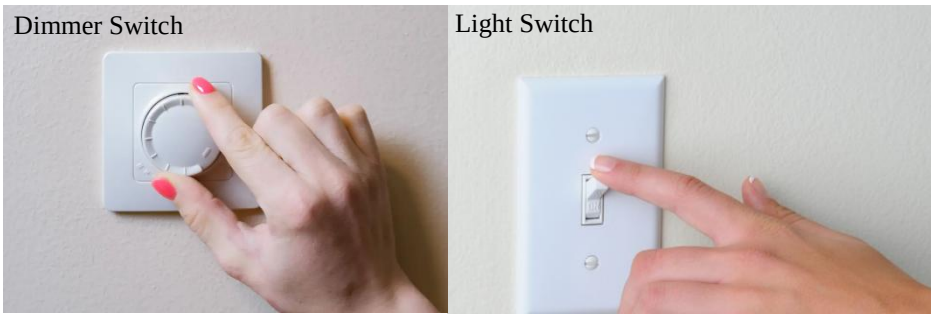
ارتباط‌گرایی به دنبال ایجاد بستری است که از آن هوش ممکن است ظهور کند. امروزه، ارتباط‌گرایی به معنای شبکه‌های عصبی است - با این اشاره که واژه «عصبی» به نورون‌های زیستی اشاره دارد. با این حال، علیرغم این نام، رابطه بین این دو سطحی است. نورون‌های زیستی و نورون‌های مصنوعی ممکن است ساختار مشابهی داشته باشند، اما به شیوه‌ای کاملاً متفاوت عمل می‌کنند.

نورون‌های زیستی ورودی‌ها را از طریق دندریت‌ها دریافت می‌کنند و زمانی که تعداد کافی از ورودی‌ها فعال شوند، «شلیک» می‌کنند تا یک پالس ولتاژ کوتاه‌مدت در آکسون ایجاد کنند. به عبارت دیگر، نورون‌های زیستی خاموش هستند تا زمانی که روشن شوند. حدود 800 میلیون سال تکامل جانوری این فرآیند را به مراتب پیچیده‌تر کرده است، اما جوهره اصلی همین است.



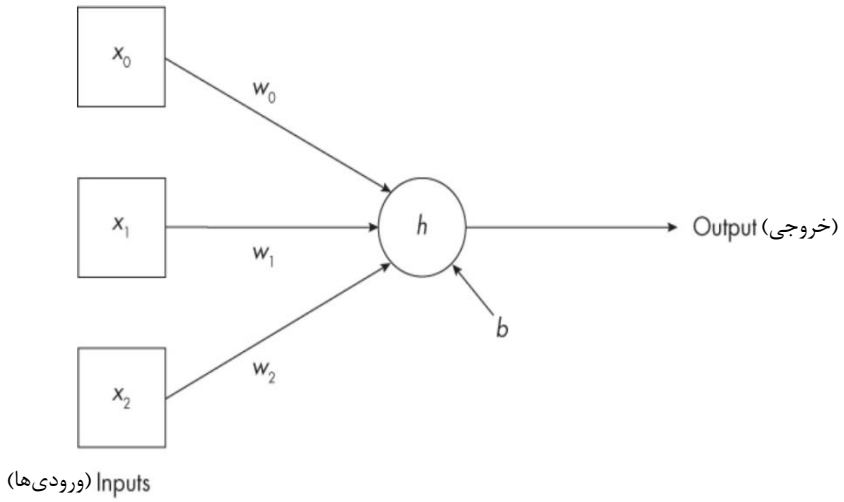
نورون‌های مصنوعی در یک شبکه عصبی نیز ورودی‌ها و خروجی‌هایی دارند، اما به جای شلیک، این نورون‌ها توابع ریاضی با رفتار پیوسته هستند. برخی مدل‌ها مانند نورون‌های زیستی شلیک می‌کنند، اما در این کتاب آن‌ها را نادیده می‌گیریم. شبکه‌های عصبی که انقلاب هوش مصنوعی را پیش می‌برند، به صورت پیوسته عمل می‌کنند.

یک نورون زیستی را مانند کلید چراغ (Light Switch) تصور کنید. خاموش است تا زمانی که دلیلی (ورودی کافی) برای روشن شدن وجود داشته باشد. نورون زیستی مانند زدن کلید، روشن و خاموش می‌شود. یک نورون مصنوعی مانند چراغی با کلید دیمر (Dimmer Switch) است. کلید را کمی بچرخانید تا نور کمی تولید شود؛ کلید را بیشتر بچرخانید و روشنایی نور به تناسب تغییر می‌کند.



این قیاس در همه موارد دقیق نیست، اما این مفهوم اساسی را منتقل می‌کند که نورون‌های مصنوعی همه یا هیچ نیستند. در عوض، آنها خروجی را متناسب با ورودی خود و بر اساس برخی توابع تولید می‌کنند. با پیشروی در فصل، این ابهام از بین خواهد رفت، بنابراین اگر الان چیز زیادی متوجه نمی‌شوید نگران نباشید.

شکل 4-1 مهم‌ترین شکل این کتاب است. اگر آنچه را این شکل نشان می‌دهد و نیز چگونگی کارکردش را بفهمیم، درک هسته‌ای لازم از هوش مصنوعی مدرن را به دست آورده‌ایم. شکل 4-1 شامل سه مربع، یک دایره، پنج فلش و برچسب‌هایی مانند x_0 و خروجی (Output) است.



روش استاندارد در شبکه‌های عصبی به این صورت است که ورودی‌ها در سمت چپ قرار می‌گیرند و جریان داده به سمت راست حرکت می‌کند. در شکل، سه مربع با برچسب‌های x_0، x_1 و x_2 نشان‌دهنده ورودی‌های نورون هستند. این‌ها سه ویژگی از بردار ویژگی هستند که می‌خواهیم نورون آن‌ها را پردازش کند تا خروجی‌ای تولید کند که به برچسب کلاس منجر شود.

دایره با حرف h برچسب‌گذاری شده است که نماد استانداردی برای تابع فعال‌ساز است. وظیفه تابع فعال‌ساز این است که ورودی نورون را دریافت کرده و یک مقدار خروجی تولید کند (همان پیکان رو به راست در شکل 4-1).

سه مربع ورودی با فلش‌هایی به دایره (گره) متصل شده‌اند - هر ورودی یک فلش. برچسب‌های فلش‌ها w_0، w_1 و w_2 نشان‌دهنده وزن‌ها (Weights) هستند. هر ورودی به نورون یک وزن مرتبط دارد. مقدار b که با یک فلش به دایره متصل است، بایاس نام دارد. این‌ها همگی عدد هستند - وزن‌ها، ورودی‌های x و خروجی. برای این نورون، سه عدد وارد می‌شود و یک عدد خارج می‌شود.

نحوه عملکرد نورون به این صورت است:

1. هر مقدار ورودی (x_0، x_1 و x_2) را در وزن مرتبط با آن (w_0، w_1 و w_2) ضرب کنید.

2. تمام حاصل‌ضرب‌های مرحله 1 را به همراه مقدار بایاس (b) جمع بزنید. این کار یک عدد واحد تولید می‌کند.

3. این عدد واحد را به تابع فعال‌ساز (h) بدهید تا خروجی (که آن هم یک عدد واحد است) تولید شود.

تمام کاری که یک نورون انجام می‌دهد همین است: ورودی‌هایش را در وزن‌ها ضرب می‌کند، حاصل‌ضرب‌ها را جمع می‌زند، مقدار بایاس را اضافه می‌کند و این مجموع را به تابع فعال‌ساز می‌دهد تا خروجی تولید شود.

تقریباً تمام دستاوردهای خارق‌العاده هوش مصنوعی مدرن مدیون این ساختار ابتدایی است. اگر تعداد کافی از این نورون‌ها را با پیکربندی صحیح به هم وصل کنید، مدلی خواهید داشت که می‌تواند نژاد سگ‌ها را تشخیص دهد، ماشین براند یا از فرانسوی به انگلیسی ترجمه کند. البته به شرطی که مقادیر جادویی وزن‌ها و بایاس‌ها را داشته باشید - که همان‌ها را فرآیند آموزش به

ما می‌دهد. این مقادیر آنقدر در شبکه‌های عصبی اهمیت دارند که یک شرکت نام خود را Weights & Biases (وزن‌ها و بایاس‌ها) گذاشته است (به آدرس https://www.wandb.ai مراجعه کنید).

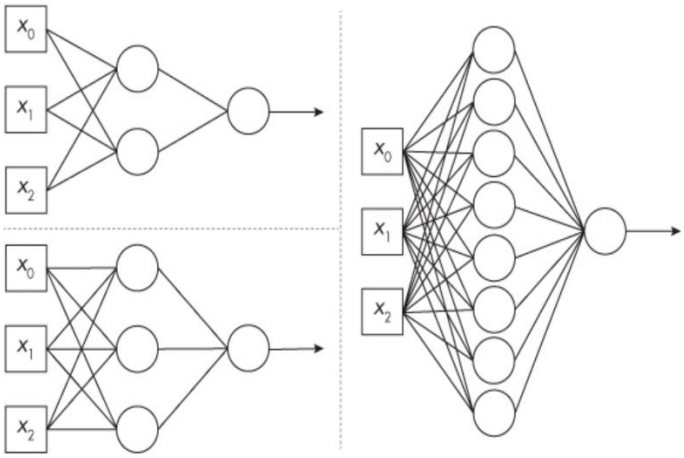
در انتخاب تابع فعال‌ساز اختیار داریم، اما در شبکه‌های مدرن معمولاً از واحد یک سو شده خطی (ReLU) استفاده می‌شود که در فصل 2 به آن اشاره کردیم. ReLU در واقع یک سؤال ساده می‌پرسد: آیا ورودی (مجموع ورودی‌ها ضرب‌در وزن‌ها به اضافه بایاس) کمتر از صفر است؟ اگر چنین باشد، خروجی صفر خواهد بود؛ در غیر این صورت، خروجی برابر با همان مقدار ورودی است.

آیا چنین نورون ساده‌ای می‌تواند مفید باشد؟ قطعاً. در یک آزمایش، نورون شکل 1-4 را با استفاده از سه ویژگی از مجموعه داده گل‌های زنبق (که در فصل 1 معرفی شد) آموزش دادم. این مجموعه داده شامل اندازه‌گیری‌هایی از سه گونه مختلف زنبق است. پس از آموزش، نورون را روی یک مجموعه آزمون شامل 30 بردار ویژگی آزمایش کردم که 28 مورد را به درستی طبقه‌بندی کرد (دقت 93 درصد).

روش آموزش نورون به این صورت بود که مقادیر بهینه برای سه وزن و یک بایاس را جستجو کردم تا خروجی (با گرد کردن به نزدیک‌ترین عدد صحیح) با برچسب کلاس گل‌ها (0، 1 یا 2) مطابقت داشته باشد. البته این روش استاندارد آموزش شبکه‌های عصبی نیست، اما برای یک نورون منفرد کافی است. روش‌های استاندارد آموزش شبکه را بعداً در همین فصل بررسی خواهیم کرد.

هرچند یک نورون منفرد می‌تواند یاد بگیرد، اما ورودی‌های پیچیده آن را گیج می‌کنند. برای ورودی‌های پیچیده به مدل پیچیده‌تری نیاز داریم. بیایید به نورون تنهایمان چند دوست اضافه کنیم!

روش متعارف، چیدمان نورون‌ها در لایه‌هاست که در آن خروجی‌های هر لایه به عنوان ورودی لایه بعدی استفاده می‌شوند. شکل 2-4 شبکه‌هایی را نشان می‌دهد که به ترتیب دارای 2، 3 و 8 نورون در لایه پس از ورودی هستند. چینش شبکه به صورت لایه‌ای، هم پیاده‌سازی کدنویسی را ساده می‌کند و هم فرآیند استاندارد آموزش را تسهیل می‌نماید. البته اگر روش جایگزینی برای آموزش مدل پیدا شود، استفاده از لایه‌ها اجباری نخواهد بود.



بیایید با شبکه دو گرهی در سمت چپ بالا شروع کنیم.

سه ورودی (مربع‌ها) حضور دارند، اما این بار دو دایره در لایه میانی و یک دایره در سمت راست داریم. ورودی‌ها به طور کامل به دو گره در لایه میانی متصل شده‌اند، به این معنی که هر مربع ورودی به هر گره در لایه میانی با یک خط وصل شده است. خروجی‌های لایه میانی به یک گره منفرد در انتهای سمت راست متصل هستند که خروجی شبکه از آنجا می‌آید.

لایه‌های میانی یک شبکه عصبی که بین ورودی‌ها در سمت چپ و خروجی در سمت راست قرار دارند، به عنوان لایه‌های پنهان شناخته می‌شوند. برای مثال، شبکه‌های شکل 2-4 هر کدام یک لایه پنهان دارند که به ترتیب دارای 2، 3 و 8 گره هستند. شبکه با این پیکربندی برای یک کار طبقه‌بندی دودویی مناسب است - کلاس 0 در مقابل کلاس 1 - که در آن خروجی یک عدد منفرد است که نشان‌دهنده اعتقاد مدل به تعلق ورودی به کلاس 1 می‌باشد. بنابراین، گره سمت راست از یک تابع فعال‌ساز متفاوت به نام سیگموئید (که به آن لجستیک نیز گفته می‌شود) استفاده می‌کند. تابع سیگموئید خروجی‌ای بین 0 و 1 تولید می‌کند. این محدوده همچنین برای نمایش احتمال استفاده می‌شود، بنابراین بسیاری از افراد به خروجی یک گره با تابع فعال‌ساز سیگموئید، به عنوان احتمال اشاره می‌کنند. این تعبیر عموماً دقیق نیست، اما می‌توانیم با این سهل‌انگاری کنار بیاییم.

تمام گره‌های لایه پنهان از توابع فعال‌ساز ReLU استفاده می‌کنند.

برای پیاده‌سازی شبکه دو گرهی در شکل 2-4، چند وزن و بایاس می‌خواهیم؟ به ازای هر خط (به جز فلش خروجی) یک وزن و برای هر گره یک مقدار بایاس نیاز داریم. بنابراین به 8 وزن و 3 مقدار بایاس نیازمندیم. برای مدل پایین سمت چپ، 12 وزن و 4 بایاس لازم است و در نهایت برای مدل 8 گره‌ای باید 32 وزن و 9 بایاس داشت. با افزایش تعداد گره‌ها در یک لایه، تعداد وزن‌ها با سرعت بیشتری افزایش می‌یابد. همین واقعیت به تنهایی برای سال‌ها رشد شبکه‌های عصبی را محدود کرد، چرا که مدل‌های بالقوه مفید برای حافظه یک کامپیوتر معمولی بسیار بزرگ بودند. البته اندازه مدل نسبی است؛ GPT-3 اوپن‌ای‌آی بیش از 175 میلیارد وزن دارد و اگرچه درباره اندازه GPT-4 صحبتی نمی‌کنند، شایعات حاکی از 7.1 تریلیون وزن است.

برای بررسی مدل‌های شکل 2-4 به یک مجموعه داده دو کلاسه نیاز داریم. مجموعه داده‌ای که استفاده خواهیم کرد یک مجموعه داده کلاسیک است که سعی در تمایز بین دو گونه انگور مورد استفاده برای تولید شراب در منطقه‌ای خاص از ایتالیا دارد. متأسفانه به نظر می‌رسد شراب‌های مربوط به این مجموعه داده دیگر شناخته شده نیستند (این نشان می‌دهد مجموعه داده چقدر قدیمی است). اما می‌دانیم که مدل‌ها به برچسب‌ها کاری ندارند - آنها از اعداد استفاده می‌کنند - بنابراین از 0 و 1 به عنوان برچسب استفاده خواهیم کرد.

به سه ویژگی x_0، x_1 و x_2 نیاز داریم. ویژگی‌هایی که استفاده خواهیم کرد شامل درصد محتوای الکل، اسید مالیک (جوهر سیب) و فنُل‌های کل هستند. هدف این است که مدل‌های شکل 2-4 را آموزش دهیم تا ببینیم هر کدام چقدر خوب می‌توانند یک شراب ناشناخته را با استفاده از مقادیر این سه ویژگی شناسایی کنند.

من مدل دو-نورونی را با استفاده از یک مجموعه آموزشی متشکل از 104 نمونه و یک مجموعه آزمایشی 26 نمونه‌ای آموزش دادم. این بدان معناست که من از 104 سه‌تایی شامل میزان الکل، سطح اسید مالیک و فنل‌های کل استفاده کردم، در حالی که برچسب خروجی صحیح (کلاس 0 یا 1) را می‌دانستم. مجموعه آموزشی، مدل دو-نورونی را برای تعیین مقادیر تمام هشت وزن و سه بایاس تنظیم کرد. قول می‌دهم که بعداً در مورد نحوه عملکرد آموزش صحبت خواهیم کرد، اما فعلاً فرض کنید این فرایند اتفاق افتاده است تا بتوانیم رفتار شبکه‌های عصبی را بررسی کنیم. مدل آموزش‌دیده به دقت 81 درصدی در مجموعه

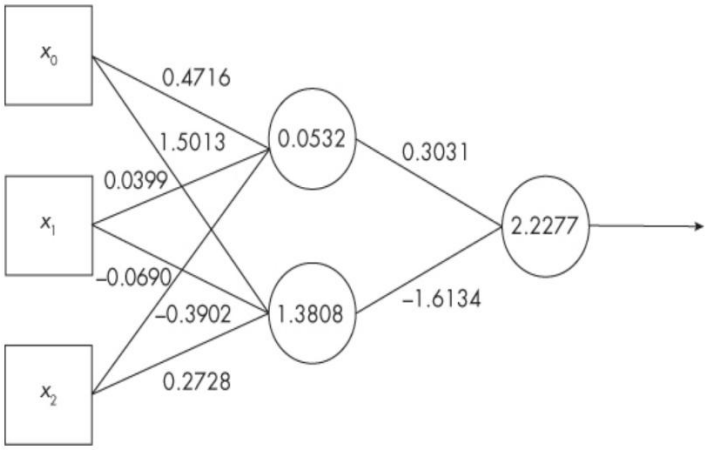
آزمایشی دست یافت، یعنی در بیش از 8 مورد از هر 10 مورد پاسخ صحیح داد. این نتیجه برای مدلی به این کوچکی و با چنین مجموعه آموزشی محدودی چندان بد نیست.

شکل 3-4 مدل دو-نورونی آموزش‌دیده را نشان می‌دهد. وزن‌ها را روی خطوط اتصال و بایاس‌ها را در کنار گره‌ها اضافه کرده‌ام تا بتوانید آنها را ببینید. به نظرم ارزش دارد که حداقل یک بار این اعداد را بررسی کنید، و بهترین موقع برای این کار، هنگام کار با یک مدل ساده است.

بیایید از مدل با دو نمونه آزمایشی برای درک فرآیند استفاده کنیم. هر کدام از دو نمونه آزمایشی شامل سه عدد هستند (مقادیر ویژگی‌های x_0، x_1 و x_2):

نمونهٔ 1: (–0.7359, 0.9795, –0.1333)

نمونهٔ 2: (0.0967, –1.2138, –1.0500)



ممکن است در این مرحله سوالی برایتان پیش آمده باشد. من گفتم که ویژگی‌ها شامل درصد الکل، سطح اسید مالیک و فنول کل هستند. هرچند که واحدهای اندازه‌گیری اسید مالیک یا فنول کل را نمی‌دانم، اما درصد همیشه درصد است، پس چرا x_0 برای نمونه اول یک عدد منفی کوچک است؟ ما نمی‌توانیم درصد الکل منفی داشته باشیم.

پاسخ به پیش‌پردازش داده‌ها مربوط می‌شود. داده‌های خام، مثل درصد الکل، به ندرت به همان شکل در مدل‌های یادگیری ماشین استفاده می‌شوند. در عوض، هر ویژگی با کم کردن میانگین آن ویژگی در مجموعه آموزش و تقسیم نتیجه بر میزان پراکندگی داده‌ها حول میانگین (انحراف معیار) تنظیم می‌شود. مقدار اصلی الکل 12.29 درصد بود، که برای شراب معقول است، اما پس از نرمال‌سازی به 0.7359- تبدیل شده است.

بیایید نمونه 1 را با استفاده از وزن‌ها و بایاس‌های یادگرفته‌شده در شکل 3-4 دسته‌بندی کنیم. ورودی به نورون بالایی، حاصل ضرب هر ویژگی در وزن مربوط به خط اتصال آن ویژگی به نورون است که سپس با مقدار بایاس جمع می‌شود.

ویژگی اول: 0.4716 × 0.7359-

ویژگی دوم: 0.0399 × 0.9795

ویژگی سوم: 0.3902- × 0.1333-

بایاس: 0.0532

با جمع کردن همه این‌ها، عدد 0.2028- به دست می‌آید که به تابع فعال‌ساز (ReLU) منتقل می‌شود. از آنجا که این عدد منفی است، خروجی ReLU برابر صفر خواهد بود، یعنی نورون بالایی هیچ خروجی ندارد.

با تکرار محاسبات برای نورون پایینی، عدد 0.1720 به عنوان ورودی ReLU به دست می‌آید. این مقدار مثبت است، بنابراین ReLU همان 0.1720 را برمی‌گرداند.

حالا خروجی‌های دو نورون در لایه میانی به عنوان ورودی‌های نورون نهایی سمت راست استفاده می‌شوند. مانند قبل، خروجی‌ها در وزن‌های مربوطه ضرب شده، با بایاس جمع می‌شوند و به تابع فعال‌ساز منتقل می‌گردند. اما این بار تابع فعال‌ساز ReLU نیست، بلکه سیگموئید است.

خروجی نورون بالایی: 0

خروجی نورون پایینی: 0.1720

با ضرب این مقادیر در وزن‌های مربوطه و جمع کردن آن‌ها به همراه بایاس (2.2277)، عدد 1.9502 به عنوان ورودی تابع سیگموئید محاسبه می‌شود. در نهایت، خروجی شبکه برای نمونه اول 0.8755 خواهد بود.

چگونه باید این خروجی را تفسیر کنیم؟ اینجا به جنبه مهمی از شبکه‌های عصبی پی می‌بریم: شبکه‌های عصبی به ما برچسب کلاس واقعی را نمی‌گویند، بلکه فقط میزان اطمینان مدل را نسبت به یک برچسب در مقایسه با دیگری نشان می‌دهند.

مدل‌های دودویی (باینری) یک مقدار اطمینان تولید می‌کنند که ما آن را به عنوان احتمال تعلق ورودی به کلاس 1 تفسیر می‌کنیم. احتمالات اعدادی بین 0 (عدم احتمال) تا 1 (قطعیت کامل) هستند. انسان‌ها معمولاً با درصد راحت‌ترند، بنابراین با ضرب احتمال در 100، می‌گوییم شبکه با 87 درصد اطمینان پیش‌بینی می‌کند که این نمونه متعلق به کلاس 1 است.

در عمل، از یک آستانه (cutoff) برای تصمیم‌گیری درباره برچسب استفاده می‌کنیم. رایج‌ترین روش در مدل‌های دودویی، آستانه 50 درصد است: اگر خروجی از 50 درصد (احتمال 0.5) بیشتر باشد، ورودی را به کلاس 1 نسبت می‌دهیم.

در این مورد، خروجی 87.55٪ است، بنابراین برچسب «کلاس 1» را اختصاص می‌دهیم. از آنجا که نمونه واقعاً متعلق به کلاس 1 است، پیش‌بینی شبکه درست است.

حالا می‌توانیم این محاسبات را برای نمونه دوم (1.0500– ,1.2138– ,0.0967)تکرار کنیم. جزئیات را به عنوان تمرین به شما واگذار می‌کنم، اما خروجی شبکه برای این نمونه 0.4883 (49٪ اطمینان به کلاس 1) است. چون از آستانه 50٪ کمتر است، برچسب کلاس 0 را اختصاص می‌دهیم. اما از آنجا که نمونه واقعی متعلق به کلاس 1 است، شبکه این‌بار اشتباه کرده است!

آیا این مدل کاربردی است؟ پاسخ به زمینه استفاده بستگی دارد. ما در حال طبقه‌بندی شراب هستیم. اگر مدل 20٪ مواقع (یک‌بار از هر پنج‌بار) اشتباه کند، آیا قابل‌قبول است؟ احتمالاً نه! اما برای برخی کاربردهای دیگر، ممکن است این سطح از دقت کافی باشد.

شبکه‌های عصبی تا حدی کنترل بر چگونگی تفسیر خروجی‌هایشان را در اختیار ما می‌گذارند. برای مثال، ممکن است از 50 درصد به عنوان آستانه قطعی استفاده نکنیم. اگر این آستانه را پایین‌تر در نظر بگیریم، مثلاً 40 درصد، نمونه‌های بیشتری از کلاس 1 را شناسایی خواهیم کرد، اما به بهای شناسایی نادرست نمونه‌های بیشتری از کلاس 0 به عنوان کلاس 1. به عبارت دیگر، می‌توانیم یک نوع خطا را با نوع دیگری معامله کنیم.

حال بیایید مدل‌های دیگر در شکل 2-4 را نیز وارد بررسی کنیم. من هر سه مدل را با استفاده از همان مجموعه‌های آموزشی و آزمایشی که برای شکل 3-4 استفاده شده بود، آموزش دادم. این فرآیند را برای هر یک از سه مدل 240 بار تکرار کردم. در ادامه دقت‌های متوسط به دست آمده را مشاهده می‌کنید:

دو گرهی: 81.5 درصد

سه گرهی: 83.6 درصد

هشت گرهی: 86.2 درصد

عملکرد مدل با افزایش تعداد گره‌ها در لایه پنهان بهبود می‌یابد. این موضوع از نظر شهودی قابل درک است، چرا که یک مدل پیچیده‌تر (با گره‌های بیشتر) به معنای توانایی یادگیری ارتباطات پیچیده‌تر موجود در مجموعه آموزشی است.

به گمانم اکنون پرسش جدیدی برایتان پیش آمده: چرا هر مدل را 240 بار آموزش دادم و دقت متوسط تمام این 240 مدل را گزارش کردم؟ اینجا به نکته مهم دیگری درباره شبکه‌های عصبی می‌رسیم: شبکه‌های عصبی به صورت تصادفی مقداردهی اولیه می‌شوند، به طوری که آموزش مکرر حتی با استفاده از داده‌های آموزشی یکسان، به مدل‌هایی با عملکرد متفاوت منجر می‌شود.

عبارت «مقداردهی اولیه تصادفی» نیاز به توضیح دارد. دوباره به شکل 3-4 نگاه کنید. اعدادی که وزن‌ها و بایاس‌ها را نشان می‌دهند، حاصل یک فرآیند تکراری هستند. این بدان معناست که یک مجموعه اولیه از وزن‌ها و بایاس‌ها به طور مکرر به‌روزرسانی می‌شوند و هر بار شبکه را به تقریب بهتری از آن تابعی که بردارهای ویژگی ورودی و برچسب‌های خروجی را به هم مرتبط می‌کند، نزدیکتر می‌کنند. تقریب خوب این تابع دقیقاً همان چیزی است که ما از شبکه انتظار داریم.

چرا همه وزن‌ها را با یک مقدار اولیه مقداردهی نکنیم؟ پاسخ این است که این کار وزن‌ها را مجبور می‌کند ویژگی‌های یکسانی از داده‌ها را یاد بگیرند، که چیزی نیست که ما بخواهیم، و در نهایت مدیل عملکرد ضعیفی خواهد داشت. اگر همه وزن‌های اولیه را صفر قرار دهیم، مدل اصلاً چیزی یاد نمی‌گیرد.

یک مجموعه مقادیر اولیه برای فرآیند تکراری ضروری است. اما چگونه باید مقادیر اولیه را انتخاب کنیم؟ این یک سوال مهم است، و پاسخ در سطح فعلی درک ما این است: «به صورت تصادفی»، یعنی به نوعی تاس می‌اندازیم تا مقدار اولیه هر وزن و بایاس را تعیین کنیم. سپس فرآیند تکراری این مقادیر را اصلاح می‌کند تا به مجموعه نهایی در شکل 3-4 برسد.

با این حال، فرآیند تکراری همیشه به نتیجه یکسان ختم نمی‌شود. اگر مجموعه متفاوتی از وزن‌ها و بایاس‌های اولیه تصادفی را انتخاب کنید، شبکه به مجموعه متفاوتی از مقادیر نهایی همگرا می‌شود. برای مثال، شبکه در شکل 3-4 به دقتی معادل 81

درصد رسید، همانطور که قبلاً ذکر شد. در ادامه 10 نتیجه دقت دیگر برای همان شبکه که روی همان داده‌ها آموزش دیده و آزمایش شده است آورده شده است: 89, 85, 73, 81, 81, 81, 81, 85, 85, 85

میزان دقت‌ها از حداکثر 89 درصد تا حداقل 73 درصد متغیر است. تنها چیزی که بین هر جلسه آموزش تغییر کرد، مجموعه وزن‌ها و بایاس‌های اولیه بود. این مسئله اغلب در شبکه‌های عصبی نادیده گرفته‌شده است.

در صورت امکان، باید شبکه‌ها را چندین بار آموزش داد تا داده‌هایی درباره کارایی آنها جمع‌آوری شود یا مثلاً در مورد نسخه 73 درصدی شبکه، متوجه شویم که صرفاً بر اساس تصادف، مجموعه بدی از مقادیر اولیه استفاده شده است. همچنین باید اشاره کنم که تغییرات گسترده در دقت این شبکه به دلیل نسبتاً کوچک بودن آن و داشتن تنها تعداد کمی وزن و بایاس است. مدل‌های بزرگ‌تر معمولاً هنگام آموزش مکرر، سازگاری بیشتری از خود نشان می‌دهند.

تا اینجا مطالب زیادی را پوشش دادیم، بنابراین خلاصه‌ای از نکات کلیدی را مرور می‌کنیم:

- واحد اساسی یک شبکه عصبی، نورون است که گاهی گره (Node) نیز نامیده می‌شود.

- نورون‌ها ورودی‌های خود را در وزن‌ها ضرب می‌کنند، حاصل‌ضرب‌ها را جمع می‌زنند، یک مقدار بایاس اضافه می‌کنند و سپس تمام این موارد را به تابع فعال‌ساز می‌دهند تا مقدار خروجی تولید شود.

- شبکه‌های عصبی مجموعه‌ای از نورون‌های منفرد هستند که معمولاً در لایه‌ها سازماندهی می‌شوند، به‌طوری که خروجی لایه فعلی، ورودی لایه بعدی را تشکیل می‌دهد.

- آموزش یک شبکه عصبی شامل تعیین مقادیر مناسب برای وزن‌ها و بایاس‌ها از طریق تنظیم تدریجی یک مجموعه اولیه تصادفی در فرآیند تکرارشونده است.

- شبکه‌های عصبی دودویی (باینری) خروجی‌ای تولید می‌کنند که تقریباً معادل احتمال تعلق ورودی به کلاس 1 است.

حالا که می‌دانیم شبکه عصبی چیست و چگونه استفاده می‌شود، بالاخره به اصل مطلب می‌رسیم: این وزن‌ها و بایاس‌های جادویی در ابتدا از کجا می‌آیند؟

در فصل 2 اشاره کردم که شبکه‌های عصبی در دهه 1980 به لطف دو الگوریتم اساسی پیشرفت کردند: پس‌انتشار (backpropagation) و گرادیان کاهشی (gradient descent). این‌ها الگوریتم‌های هسته مرکزی آموزش شبکه‌های عصبی هستند.

در فصل 3 در مورد ماشین‌های بردار پشتیبان، بهینه‌سازی را - که فرآیند یافتن بهترین چیز بر اساس معیارهای خاصی است - بحث کردیم. آموزش یک شبکه عصبی نیز یک فرآیند بهینه‌سازی است که شامل یادگیری وزن‌ها و بایاس‌هایی است که بهترین تناسب را با داده‌های آموزشی دارند. با این حال باید دقت شود که وزن‌ها و بایاس‌های یادگرفته شده بیشتر تمایل‌های کلی در داده‌های آموزشی را دنبال کنند تا جزئیات خاص خود داده‌های آموزشی. معنای این جمله با یادگیری بیشتر درباره فرآیند آموزش روشن خواهد شد.

الگوریتم کلی آموزش به این صورت است:

1. معماری مدل را انتخاب کنید، شامل تعداد لایه‌های پنهان، گره‌های هر لایه و تابع فعال‌ساز.

2. همه وزن‌ها و بایاس‌های مرتبط با معماری انتخاب شده را به صورت تصادفی اما هوشمندانه مقداردهی اولیه کنید.

3. داده‌های آموزشی (یا زیرمجموعه‌ای از آن‌ها) را از مدل عبور داده و میانگین خطا را محاسبه کنید. این مرحله «گذر به جلو» نام دارد.

4. از الگوریتم پس انتشار استفاده کنید تا مشخص شود هر وزن و بایاس چقدر در این خطا نقش دارد.

5. وزن‌ها و بایاس‌ها را طبق الگوریتم گرادیان کاهشی به‌روزرسانی کنید. این مرحله و مرحله قبل، «گذر به عقب» را تشکیل می‌دهند.

6. از مرحله 3 تکرار کنید تا شبکه در حد «به اندازه کافی خوب» عمل کند.

این شش مرحله شامل اصطلاحات مهمی هستند که ارزش دارد زمان بگذاریم و هرکدام را به خوبی درک کنیم.

معماری شبکه در این فصل به تعداد لایه‌ها (معمولاً لایه‌های پنهان) اشاره دارد. ما یک بردار ویژگی ورودی داریم، و هر لایه پنهان به صورت جمعی کار می‌کند تا یک بردار ورودی را گرفته و بردار خروجی تولید کند، که سپس به عنوان ورودی لایه بعدی استفاده می‌شود. برای طبقه‌بندهای دودویی، خروجی شبکه یک گره منفرد است که مقداری بین 0 تا 1 تولید می‌کند. بعداً در کتاب خواهیم دید که این ایده به خروجی‌های چندکلاسه نیز قابل تعمیم است.

این الگوریتم نشان می‌دهد که آموزش یک فرآیند تکراری است. هر فرآیند تکراری نیاز به یک نقطه شروع دارد. اگر بخواهید از نقطه A به نقطه B بروید، باید یک پا را جلوی پای دیگر بگذارید – این بخش تکراری کار است. نقطه A همان نقطه شروع است. برای یک شبکه عصبی، معماری شبکه به معنای مجموعه‌ای از وزن‌ها و بایاس‌هاست. مقادیر اولیه‌ای که به این وزن‌ها و بایاس‌ها اختصاص داده می‌شوند، شبیه نقطه A هستند، و آموزش شبکه مانند گذاشتن یک پا جلوی پای دیگر است.

عبارت «میانگین خطا» در الگوریتم به چه خطایی اشاره دارد؟ اینجا است که یک مفهوم جدید مطرح می‌شود. به طور شهودی می‌توان دید که صرفاً انتخاب برخی مقادیر اولیه برای وزن‌ها و بایاس‌ها بعید است منجر به شبکه‌ای شود که بتواند داده‌های آموزشی را به دقت طبقه‌بندی کند. به یاد داشته باشید که ما ورودی‌ها و خروجی‌های مورد انتظار برای داده‌های آموزشی را می‌دانیم.

فرض کنید نمونه آموزشی شماره 1 را از شبکه عبور می‌دهیم و خروجی 0.44 را دریافت می‌کنیم. اگر بدانیم این نمونه متعلق به کلاس 1 است، خطای شبکه برابر است با تفاوت بین خروجی مورد انتظار و خروجی واقعی. در اینجا، این مقدار 1 منهای 0.44 یا 0.56 است. یک مدل خوب ممکن است به جای آن خروجی 0.97 را برای این نمونه تولید کند که خطای آن فقط 0.03 خواهد بود. هرچه خطا کوچکتر باشد، مدل در طبقه‌بندی نمونه بهتر عمل کرده است. اگر همه داده‌های آموزشی (یا زیرمجموعه‌ای نماینده از آن‌ها) را از شبکه عبور دهیم، می‌توانیم خطای هر نمونه آموزشی را محاسبه کرده و میانگین آن را برای کل مجموعه آموزشی به دست آوریم. این معیاری است که الگوریتم‌های پس‌انتشار و گرادیان کاهشی (که بعداً توضیح داده خواهند شد) برای به‌روزرسانی وزن‌ها و بایاس‌ها استفاده می‌کنند.

در نهایت، الگوریتم آموزشی می‌گوید داده‌ها را از شبکه عبور دهید، خطا را محاسبه کنید، وزن‌ها و بایاس‌ها را به‌روزرسانی کنید و این کار را تا زمانی که شبکه «به اندازه کافی خوب» شود تکرار کنید. به نوعی، «به اندازه کافی خوب» زمانی است که خطا (که به آن هزینه نیز می‌گویند) تا حد ممکن به صفر نزدیک شود. اگر شبکه برای همه نمونه‌های کلاس 0 خروجی 0 و برای همه نمونه‌های کلاس 1 خروجی 1 تولید کند، عملکرد آن روی داده‌های آموزشی کامل است و خطا صفر خواهد بود. این قطعاً به اندازه کافی خوب است، اما باید مراقب باشیم. گاهی اوقات وقتی این اتفاق می‌افتد، شبکه دچار بیش‌برازش شده است، یعنی تمام جزئیات داده‌های آموزشی را یاد گرفته بدون آنکه روندهای کلی داده را یاد بگیرد که به آن اجازه می‌دهد در مواجهه با ورودی‌های ناشناخته در دنیای واقعی نیز عملکرد خوبی داشته باشد.

در عمل، بیش‌برازش به چند روش مورد توجه قرار می‌گیرد که بهترین آن‌ها، جمع‌آوری داده‌های آموزشی بیشتر است. ما از داده‌های آموزشی به عنوان نماینده‌ای برای تمام داده‌های ممکن که می‌توانند توسط فرآیند مورد مدلسازی ما تولید شوند استفاده می‌کنیم. بنابراین، داده‌های آموزشی بیشتر به معنای نمایش بهتر از آن مجموعه داده است. این همان مسئله درونیابی در مقابل برونیابی است که در فصل 1 مورد بحث قرار دادیم.

اما ممکن است دستیابی به داده‌های آموزشی بیشتر میسر نباشد. در این صورت، راهکارهای جایگزینی وجود دارد، از جمله تنظیم الگوریتم آموزش برای معرفی مکانیسم‌هایی که از تمرکز شبکه بر جزئیات نامربوط داده‌های آموزشی جلوگیری می‌کنند. یکی از این تکنیک‌ها که ممکن است شنیده باشید، «زوال وزن» (weight decay) است که شبکه را در صورتی که مقادیر وزن‌ها را بیش از حد بزرگ کند، جریمه می‌کند.

روش متداول دیگر، «تقویت داده» (data augmentation) است. داده آموزشی کم دارید؟ نگران نباشید! تقویت داده با اعمال تغییرات جزئی روی داده‌های موجود، نمونه‌های جدیدی ایجاد می‌کند. این تکنیک داده‌های آموزشی موجود را با اعمال تغییراتی اصلاح می‌کند تا داده‌های جدیدی تولید شود که منطقاً می‌توانستند توسط همان فرآیندی که داده‌های آموزشی اصلی را تولید کرده، به وجود آیند. برای مثال، اگر نمونه آموزشی یک تصویر سگ باشد، با چرخاندن، جابجایی چند پیکسلی یا معکوس کردن چپ-راست آن، باز هم یک تصویر سگ خواهیم داشت. هر تبدیل، یک نمونه آموزشی جدید تولید می‌کند. شاید این روش تقلب به نظر برسد، اما در عمل، تقویت داده یک تنظیم‌کننده قوی است که از بیش‌برازش شبکه در حین آموزش جلوگیری می‌کند.

اجازه دهید لحظه‌ای به موضوع مقداردهی اولیه بازگردیم، چون اهمیت آن سال‌ها به اندازه کافی درک نشده بود. در ابتدا، مقداردهی اولیه وزن‌ها به سادگی به معنای «انتخاب یک عدد تصادفی کوچک» مثل 0.001 یا 0.0056- بود. این روش در بسیاری موارد جواب می‌داد، اما نه به صورت یکنواخت، و حتی وقتی جواب می‌داد، عملکرد شبکه چندان درخشان نبود.

پس از ظهور یادگیری عمیق، محققان بار دیگر ایده «مقادیر تصادفی کوچک» را با هدف یافتن روشی اصولی‌تر برای مقداردهی اولیه بررسی کردند. حاصل این تلاش‌ها، روشی است که امروزه برای مقداردهی اولیه شبکه‌های عصبی به کار می‌رود. سه عامل باید در نظر گرفته شوند: نوع تابع فعال‌ساز، تعداد اتصالات از لایه پایینی (fan-in) و تعداد خروجی‌ها به لایه بالایی (fan-out).

فرمول‌هایی طراحی شدند که از این سه عامل برای انتخاب وزن‌های اولیه هر لایه استفاده می‌کنند. مقداردهی اولیه بایاس‌ها معمولاً صفر در نظر گرفته می‌شود. به سادگی می‌توان نشان داد که شبکه‌هایی که با این روش مقداردهی اولیه می‌شوند، عملکرد بهتری نسبت به روش‌های سنتی دارند.

دو مرحله از الگوریتم آموزش هنوز باقی مانده است: پس‌انتشار (Backpropagation) و گرادیان کاهشی. معمولاً ابتدا پس انتشار توضیح داده می‌شود، چون خروجی آن برای گرادیان کاهشی ضروری است. با این حال، به نظر من درک اینکه گرادیان کاهشی چه کاری انجام می‌دهد، ابتدا شهود بهتری ایجاد می‌کند و سپس می‌توان با آنچه پس‌انتشار ارائه می‌دهد، قطعه گمشده را تکمیل کرد. با وجود نام‌های ناآشنا، مطمئنم که شما در اصل، هر دو الگوریتم را می‌فهمید.

فرض کنید در یک دشت وسیع و فراخ با تپه‌های موجدار ایستاده‌اید. چطور به اینجا رسیدید؟ مغزتان را به کار می‌گیرید، اما پاسخی نمی‌یابید. سرانجام، روستای کوچکی را در شمال، در دره‌ای پایین‌تر می‌بینید. شاید مردم آنجا بتوانند پاسخ‌هایی به شما بدهند. اما بهترین راه برای رسیدن به آنجا کدام است؟

شما می‌خواهید به طور کلی به سمت شمال و پایین بروید، اما باید به شکل زمین نیز توجه کنید. همیشه می‌خواهید از موقعیت بالاتر به موقعیت پایین‌تر حرکت کنید. نمی‌توانید مستقیماً به شمال بروید، زیرا یک تپه بزرگ سر راهتان است. می‌توانید به سمت شمال شرق بروید؛ زمین در آنجا مسطح‌تر است، اما این مسیر سفرتان را طولانی می‌کند، زیرا زمین به آرامی پایین می‌رود. بنابراین، تصمیم می‌گیرید به سمت شمال غرب بروید، زیرا این مسیر هم شما را به شمال می‌برد و هم شیب بیشتری به سمت پایین دارد. یک قدم به سمت شمال غرب برمی‌دارید، سپس مکث می‌کنید تا موقعیت جدیدتان را ارزیابی کنید و جهت بعدی حرکت را تعیین کنید.

تکرار این فرآیند دو مرحله‌ای - بررسی موقعیت فعلی برای تعیین جهتی که هم شما را به شمال و هم به پایین می‌برد، سپس برداشتن یک قدم در آن جهت - بهترین شانس شما برای رسیدن به روستا در دره است. ممکن است موفق نشوید؛ شاید در یک دره کوچک گیر بیفتید که نتوانید از آن خارج شوید. اما به طور کلی، با حرکت پیوسته در جهتی که نسبت به موقعیت فعلی‌تان به سمت شمال و پایین است، به هدف خود نزدیک می‌شوید.

پیروی از این فرآیند، که به آن «گرادیان کاهشی» می‌گویند، به ما امکان می‌دهد وزن‌ها و بایاس‌های اولیه یک شبکه عصبی را تنظیم کنیم تا مدل‌هایی با عملکرد بهتر به دست آوریم. به عبارت دیگر، گرادیان کاهشی، مدل را آموزش می‌دهد.

دنیای سه بُعدی دشت اطراف روستا معادل دنیای n-بعدی شبکه است، که در آن n تعداد کل وزن‌ها و بایاس‌هایی است که سعی داریم مقادیرشان را یاد بگیریم. انتخاب یک جهت از موقعیت فعلی و سپس حرکت به اندازه‌ای معین در آن جهت، یک گام گرادیان کاهشی است. گام‌های مکرر گرادیان کاهشی شما را به روستا نزدیک‌تر و نزدیک‌تر می‌کند.

گرادیان کاهشی به دنبال حداقل موقعیت است - روستا در دره - اما حداقل چه چیزی؟ برای یک شبکه عصبی، گرادیان کاهشی هدفش تنظیم وزن‌ها و بایاس‌های شبکه برای کمینه کردن خطا روی مجموعه آموزشی است.

دشت وسیع و فراخ با تپه‌های موجدار نشان‌دهنده تابع خطا است - میانگین خطا روی داده‌های آموزشی وقتی از مقادیر وزن و بایاس مربوط به موقعیت فعلی‌تان استفاده می‌کنید. این یعنی هر موقعیت در دشت نشان‌دهنده یک مجموعه کامل از وزن‌ها و بایاس‌های شبکه است. موقعیت روستا معادل کوچک‌ترین خطایی است که شبکه می‌تواند روی مجموعه آموزشی داشته باشد.

امید این است که مدلی که خطای کمی روی مجموعه آموزشی دارد، هنگام استفاده در دنیای واقعی روی ورودی‌های ناشناخته نیز خطاهای کمی داشته باشد. گرادیان کاهشی الگوریتمی است که در فضای وزن‌ها و بایاس‌ها حرکت می‌کند تا خطا را کمینه کند.

گرادیان کاهشی یک الگوریتم بهینه‌سازی است که تأکید می‌کند آموزش شبکه عصبی یک مسئله بهینه‌سازی محسوب می‌شود؛ مسئله‌ای که در آن باید بهترین مجموعه از یک چیز را پیدا کنیم. هرچند این گفته صحیح است، اما باید توجه داشت آموزش شبکه عصبی به شکلی ظریف با سایر مسائل بهینه‌سازی متفاوت است. همانطور که پیش‌تر اشاره شد، لزوماً به دنبال کمترین خطای ممکن روی داده‌های آموزشی نیستیم، بلکه مدلی را می‌خواهیم که بهترین تعمیم‌پذیری را به ورودی‌های ناشناخته داشته باشد. هدف ما اجتناب از بیش‌برازش است. در ادامه این فصل، این مفهوم را به صورت بصری نشان خواهم داد.

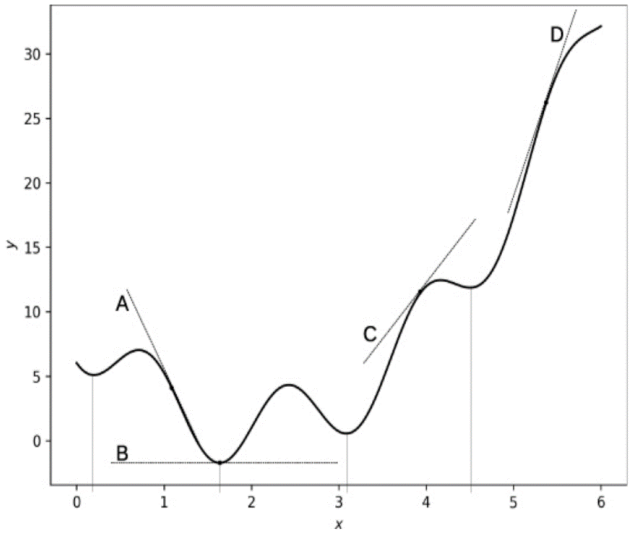
گرادیان کاهشی در منظره تابع خطا حرکت می‌کند. در کاربرد روزمره، گرادیان به معنای تغییر در چیزی است، مانند شیب یک جاده یا طیف رنگی که به آرامی از یک ته‌رنگ به ته‌رنگ دیگر تغییر می‌کند. از دید ریاضی، گرادیان نظیرِ چندبعدیِ شیب یک منحنی در یک نقطه است. جهت شیب‌دارترین حرکت، به سمت حداکثر گرادیان رو به پایین است. شیب یک خط در یک نقطه از منحنی، نمایشی سودمند از گرادیان ارائه می‌دهد، بنابراین تأمل در مورد شیب‌ها ارزشمند خواهد بود.

شکل 4-4 یک منحنی را نشان می‌دهد که چهار خط در نقاط مختلف با آن تماس دارند. این خطوط نمایانگر شیب در آن نقاط هستند. شیب نشان می‌دهد که مقدار تابع با چه سرعتی در مجاورت آن نقطه تغییر می‌کند. هرچه خط شیب‌دارتر باشد، مقدار تابع با سرعت بیشتری هنگام حرکت در امتداد محور x تغییر می‌کند.

خط B پایین‌ترین نقطه روی منحنی را نشان می‌دهد. این نقطه، **مینیمم مطلق** (Global Minimum) است - نقطه‌ای که یک الگوریتم بهینه‌سازی به دنبال یافتن آن است. توجه کنید که خط مماس بر این نقطه کاملاً افقی است. از دید ریاضی، این بدان معناست که شیب خط B صفر است. این ویژگی در تمام نقاط مینیمم (و ماکزیمم) توابع صدق می‌کند.

نقطه‌ای که خط B با آن تماس دارد، مینیمم مطلق است، اما سه مینیمم دیگر نیز در نمودار وجود دارند. این‌ها **مینیمم‌های موضعی** (Local Minima) هستند - نقاطی که شیب خط مماس بر آن‌ها نیز صفر است. در حالت ایده‌آل، یک الگوریتم بهینه‌سازی باید از این نقاط اجتناب کند و مینیمم مطلق را هدف قرار دهد.

خط A شیب تندی دارد و به سمت مینیمم مطلق اشاره می‌کند. بنابراین، اگر در نقطه‌ای از منحنی بودیم که خط A با آن تماس دارد، می‌توانستیم با برداشتن گام‌ها در جهت نشان‌داده شده، به سرعت به سمت مینیمم مطلق حرکت کنیم. علاوه بر این، از آنجا که شیب در این نقطه تند است، می‌توانیم گام‌های نسبتاً بزرگی به سمت دره برداریم.



خط C نیز شیبدار است اما به سمت یکی از مینیمم‌های موضعی هدایت می‌شود، دقیقاً همان نقطه‌ای که در محور x بعد از عدد 3 قرار دارد. یک الگوریتم گرادیان کاهشی که فقط می‌داند چگونه در جهت کاهش گرادیان حرکت کند، آن مینیمم موضعی را پیدا کرده و در آنجا گیر می‌افتد. همین موضوع در مورد خط D نیز صدق می‌کند که به سمت مینیمم موضعی بین اعداد 4 و 5 روی محور x حرکت می‌کند.

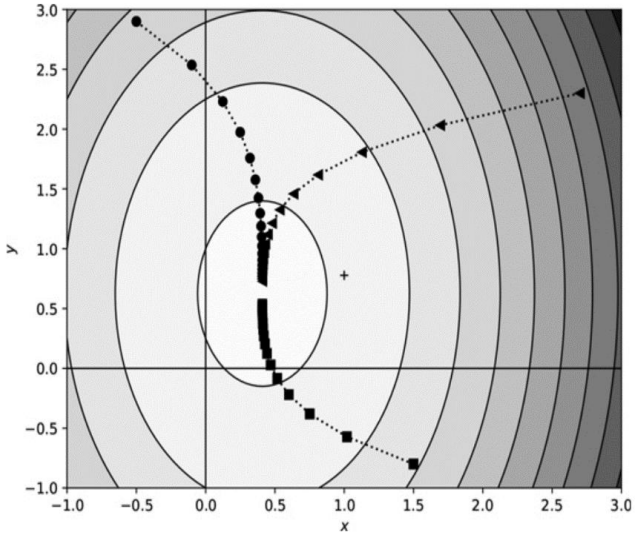
نکات کلیدی شکل 4-4 چیست؟

اولاً، گرادیان کاهشی در جهت کاهش گرادیان (شیب) از یک نقطه شروع به حرکت می‌کند. در اینجا منحنی یک‌بعدی است، بنابراین نقطه مورد نظر یک مقدار خاص از x است. گرادیان کاهشی از مقدار شیب در آن نقطه برای انتخاب جهت و اندازه گام استفاده می‌کند که متناسب با میزان تندی شیب است. یک شیب تند به این معنی است که می‌توانیم گام بزرگتری برداریم تا به مقدار x جدیدی نزدیک‌تر به یک مینیمم برسیم. در مقابل، یک شیب ملایم به معنی گام کوچک‌تر است.

برای مثال، فرض کنید در نقطه‌ای هستیم که خط A به منحنی برخورد می‌کند. شیب در این نقطه تند است، بنابراین گام بزرگی به سمت مینیمم مطلق برمی‌داریم. پس از این گام، دوباره شیب را بررسی می‌کنیم، اما این بار شیب در نقطه جدید روی محور x محاسبه می‌شود. با استفاده از این شیب، گام دیگری برمی‌داریم و این روند را تکرار می‌کنیم تا به نقطه‌ای برسیم که شیب عملاً صفر است. این نقطه، همان مینیمم است و در آنجا توقف می‌کنیم.

حالت یک‌بعدی به اندازه کافی ساده است، زیرا در هر نقطه فقط یک شیب وجود دارد و در نتیجه تنها یک جهت برای حرکت وجود دارد. اما اگر به مثال دشت گسترده و فراخ فکر کنیم، می‌دانیم که از هر نقطه، تعداد بی‌شماری جهت‌های ممکن برای حرکت وجود دارد که بسیاری از آن‌ها مفید هستند، زیرا ما را به سمت پایین‌تر (و در جهت کاهش ارتفاع) هدایت می‌کنند. یکی از این جهت‌ها، جهت ماکزیمم گرادیان، پر شیب‌ترین مسیر را ارائه می‌دهد و ما را سریع‌تر به سمت مقصد مطلوبمان پیش می‌برد - و این همان جهتی است که در آن گام برمی‌داریم.

با تکرار این فرآیند و استفاده از جهت ماکزیمم گرادیان در هر مرحله، در ابعاد چندگانه همان کاری را انجام می‌دهیم که در حالت یک‌بعدی انجام دادیم. برای دقیق‌تر بودن، باید گفت که ما در جهت خلاف ماکزیمم گرادیان حرکت می‌کنیم، زیرا جهت ماکزیمم گرادیان به سمت مینیمم اشاره نمی‌کند، بلکه از آن دور می‌شود.



شکل 4-5، گرادیان کاهشی را در دو بعد نمایش می‌دهد. این شکل یک نمودار کنتور (Contour) است. تصور کنید یک معدن روباز با سطوح پلکانی داریم: هرچه رنگ روشن‌تر باشد، نشان‌دهنده عمق بیشتر در معدن است، اما در عین حال شیب زمین نیز ملایم‌تر می‌شود. به عبارت دیگر، سایه‌های روشن‌تر بیانگر شیب‌های کم‌تر هستند.

شکل، مسیر طی شده توسط الگوریتم گرادیان کاهشی را برای سه نقطه شروع مختلف نشان می‌دهد: دایره، مثلث و مربع. در ابتدا شیب‌ها تند هستند، بنابراین اندازه گام‌ها بزرگ است، اما با نزدیک

شدن به مینیمم، شیب‌ها ملایم‌تر شده و گام‌ها کوچک‌تر می‌شوند. در نهایت، بدون توجه به نقطه شروع، گرادیان کاهشی به مینیمم می‌رسد.

ما تاکنون گرادیان کاهشی را در ابعاد یک و دو بررسی کرده‌ایم، چرا که می‌توانیم این فرآیند را به صورت تصویری درک کنیم. اکنون می‌فهمیم که این الگوریتم در واقع همان چیزی است که همیشه آن را می‌شناختیم و هرگاه از ارتفاع بالاتر به پایین‌تر حرکت می‌کنیم، ناخودآگاه از آن استفاده می‌کنیم. در واقع، آموزش یک شبکه عصبی نیز دقیقاً همین کار را انجام می‌دهد. مجموعه اولیه وزن‌ها و بایاس‌ها چیزی نیست جز یک نقطه شروع در فضای n-بعدی. گرادیان کاهشی از بیشینه گرادیان در این نقطه شروع استفاده کرده و به سمت یک مینیمم پیش می‌رود. هر موقعیت جدید در این فضای n-بعدی، مجموعه جدیدی از n وزن و بایاس است که بر اساس میزان تندی گرادیان از مجموعه قبلی تولید شده است. وقتی گرادیان به مقدار بسیار کوچکی برسد، آموزش را موفقیت‌آمیز اعلام کرده و وزن‌ها و بایاس‌ها را ثابت نگه می‌داریم، چرا که معتقدیم شبکه آموزش دیده است.

گرادیان کاهشی به شیب‌ها و مقادیر گرادیان وابسته است. اما این گرادیان‌ها از کجا می‌آیند؟ گرادیان کاهشی تابع هزینه (خطا) شبکه را کمینه می‌کند. خطای شبکه روی مجموعه آموزشی، تابعی از هر یک از وزن‌ها و بایاس‌های شبکه است. گرادیان نشان می‌دهد که هر وزن و بایاس چقدر در خطای کلی شبکه نقش دارد.

برای مثال، فرض کنید می‌دانیم وزن شماره 3 (هر چه که باشد) چقدر در خطای شبکه -که با اشتباهات شبکه روی داده‌های آموزشی اندازه‌گیری می‌شود- مشارکت دارد. در این صورت، میزان تندی گرادیان را در صورت تغییر مقدار این وزن (در حالی که سایر وزن‌ها و بایاس‌ها ثابت باشند) می‌شناسیم. این تندی، ضرب در اندازه گام، مقداری به ما می‌دهد که از مقدار فعلی وزن شماره 3 کم می‌کنیم. با این تفریق، در جهت مخالف بیشینه گرادیان حرکت می‌کنیم. با تکرار این محاسبه برای تمام وزن‌ها و بایاس‌های شبکه، یک گام در فضای n-بعدی برداشته می‌شود. این همان کاری است که گرادیان کاهشی در حین آموزش انجام می‌دهد.

الگوریتم پس‌انتشار مقادیر تندی گرادیان را برای هر وزن و بایاس محاسبه می‌کند. پس‌انتشار کاربرد یک قانون شناخته شده از حساب دیفرانسیل است - شاخه‌ای از ریاضیات که به ما می‌گوید یک متغیر چگونه نسبت به تغییر متغیر دیگر تغییر می‌کند. سرعت، یک مثال ساده است: سرعت نشان می‌دهد مسافت چگونه نسبت به زمان تغییر می‌کند (مثلاً کیلومتر بر ساعت). پس‌انتشار «سرعت» تغییر خطای شبکه نسبت به تغییر هر وزن یا بایاس را به ما می‌دهد. گرادیان کاهشی از این «سرعت‌ها» - ضرب در یک عامل مقیاس به نام نرخ یادگیری (Learning Rate) - برای برداشتن گام بعدی در فضای n-بعدی (که توسط n وزن و بایاس شبکه نمایش داده می‌شود) استفاده می‌کند.

مثلاً، شبکه «بزرگ» در شکل 2-4 دارای 32 وزن و 9 بایاس است. بنابراین، آموزش این شبکه با گرادیان کاهشی به معنای حرکت در یک فضای 41-بعدی برای یافتن مقادیر بهینه این 41 وزن و بایاس است که کمترین خطای میانگین روی مجموعه آموزشی را تولید کنند.

این الگوریتم «پس‌انتشار» نامیده می‌شود، زیرا مقادیر «سرعت» (نرخ تغییر خطا) را برای هر وزن و بایاس محاسبه می‌کند، به این ترتیب که از لایه خروجی شبکه شروع شده و سپس به‌صورت لایه‌به‌لایه به سمت لایه ورودی بازمی‌گردد. به عبارت دیگر، خطا از لایه‌های بعدی به لایه‌های قبلی انتشار داده می‌شود.

نتیجه کلیدی چنین است: گرادیان کاهشی از جهت گرادیان محاسبه‌شده توسط الگوریتم پس‌انتشار استفاده می‌کند تا در هر تکرار، وزن‌ها و بایاس‌ها را به‌روزرسانی کرده و خطای شبکه روی داده‌های آموزشی را به حداقل برساند.

و این، به‌طور خلاصه، نحوه آموزش شبکه‌های عصبی است.

توانایی آموزش یک شبکه عصبی با استفاده از پس‌انتشار و گرادیان کاهشی تا حدی شانسی به نظر می‌رسد. از نظر تئوری، این روش نباید جواب بدهد. گرادیان کاهشی همراه با پس‌انتشار یک روش بهینه‌سازی مرتبه اول است. روش‌های بهینه‌سازی مرتبه اول معمولاً برای توابع ساده بهترین عملکرد را دارند، در حالی که سطوح خطای یک شبکه عصبی هر چیزی هستند جز ساده! با این حال، بخت و اقبال به ما رو کرده و این روش نه‌تنها کار می‌کند، بلکه بسیار هم خوب جواب می‌دهد.

تاکنون هیچ توضیح ریاضی دقیقی برای این پدیده ارائه نشده است، جز این درک که مینیمم‌های موضعی تابع خطا تقریباً همگی شبیه به هم هستند. این یعنی اگر در یکی از این مینیمم‌ها گیر بیفتید و نتوانید از آن خارج شوید، معمولاً مشکلی ندارد و نتیجه مطلوب است.

توضیح تجربی دیگری نیز وجود دارد، اما برای درک آن باید بیشتر با فرآیند آموزش آشنا شویم. الگوریتم شش‌مرحله‌ای آموزش که قبلاً در این فصل ارائه شد، درباره اجرای مجموعه آموزشی (یا زیرمجموعه‌ای از آن) روی شبکه و تکرار این فرآیند تا رسیدن به نتیجه «به‌اندازه کافی خوب» صحبت می‌کند. اجازه دهید این فرآیند را بیشتر توضیح دهم.

هر بار اجرای داده‌های آموزشی روی شبکه (که شامل یک پاس رو به جلو و یک پاس به عقب است) منجر به یک گام گرادیان کاهشی می‌شود، همان‌طور که در شکل 5-4 نشان داده شده است. اگر مجموعه آموزشی کوچک باشد، تمام داده‌ها در پاس رو به جلو استفاده می‌شوند، یعنی گرادیان کاهشی از تمام آن‌ها برای تصمیم‌گیری درباره جهت گام بعدی استفاده می‌کند. یک پاس کامل روی داده‌های آموزشی را دوره (epoch) می‌نامند؛ بنابراین، استفاده از تمام داده‌های آموزشی در پاس‌های رو به جلو و عقب به معنی برداشتن «یک گام گرادیان کاهشی در هر دوره» است.

اما مجموعه‌داده‌های یادگیری ماشین امروزی اغلب بسیار حجیم هستند و استفاده از تمام داده‌های آموزشی برای هر گام گرادیان کاهشی از نظر محاسباتی غیرعملی است. در عوض، یک زیرمجموعه کوچک و تصادفی از داده‌ها به نام مینی‌بَچ (minibatch) برای پاس‌های رو به جلو و عقب انتخاب می‌شود. استفاده از مینی‌بچ‌ها بار محاسباتی را به شدت کاهش می‌دهد و در نتیجه تعداد گام‌ها در هر دوره افزایش می‌یابد. علاوه بر این، مینی‌بچ‌ها مزیت دیگری نیز دارند که به حل مشکل «این روش آموزش اصلاً نباید جواب بدهد» کمک می‌کنند.

فرض کنید یک تابع ریاضی برای نمایش خطای شبکه وجود داشت. در آن صورت می‌توانستیم از روش‌های قدیمی حسابان برای یافتن فرم دقیق سهم هر وزن و بایاس در خطا استفاده کنیم و گرادیان کاهشی همیشه بهترین جهت برای حرکت را می‌دانست. متأسفانه، دنیا این‌قدر مهربان نیست! ما فرم ریاضی تابع خطا را نمی‌دانیم (احتمالاً اصلاً چنین تابعی وجود ندارد)، بنابراین مجبوریم با داده‌های آموزشی یک تقریب بسازیم. این تقریب وقتی بهتر می‌شود که از داده‌های آموزشی بیشتری برای محاسبه خطا استفاده کنیم. این موضوع به نفع استفاده از تمام داده‌های آموزشی در هر گام گرادیان کاهشی است. اما همان‌طور که گفتیم، در بسیاری از موارد این کار از نظر محاسباتی بسیار پرهزینه است.

راه‌حل میانه‌ای که انتخاب می‌شود، استفاده از مینی‌بچ‌ها برای هر گام گرادیان کاهشی است. در این حالت محاسبات دیگر سنگین نیستند، اما تقریب گرادیان واقعی ضعیف‌تر می‌شود چون با نقاط داده کمتری آن را تخمین می‌زنیم. از آنجا که انتخاب تصادفی معمولاً با واژه Stochastic همراه است، آموزش با مینی‌بچ‌ها به نام «گرادیان کاهشی تصادفی» شناخته می‌شود. امروزه تقریباً تمام سیستم‌های هوش مصنوعی مدرن به شکلی از این روش استاندارد استفاده می‌کنند.

در نگاه اول، گرادیان کاهشی تصادفی به نظر یک راه‌حل بازنده می‌رسد. بله، شاید بتوانیم قبل از «مرگ گرمایی کیهان» (اشاره به پایان جهان از نظر ترمودینامیکی) گام‌های گرادیان کاهشی زیادی محاسبه کنیم، اما دقت گرادیان‌مان پایین است و احتمالاً در فضای خطا در جهت اشتباه حرکت می‌کنیم. این که نمی‌تواند خوب باشد، نه؟

اینجاست که ایزد بانوی شانس برای بار دوم به بشریت لبخند می‌زند! او نه تنها به ما این توانایی را داده که مدل‌های پیچیده را با گرادیان کاهشی مرتبه اول آموزش دهیم (چون مینیمم‌های موضعی تقریباً معادل فرض می‌شوند)، بلکه شرایط را طوری تنظیم کرده که جهت «اشتباه» گرادیان که توسط روش تصادفی پیدا می‌شود، اغلب همان چیزی است که برای فرار از مینیمم‌های موضعی در مراحل اولیه آموزش نیاز داریم. به بیان دیگر، وقتی که باید دقیقاً به سمت شمال برویم اما کمی به سمت شمال‌شرق حرکت می‌کنیم، این ظاهراً یک اشتباه اما در واقع یک موهبت پنهان است که به ما امکان آموزش شبکه‌های عصبی بزرگ را می‌دهد.

حالا آماده‌ایم تا به فصل بعدی برویم. اما قبل از آن، بیایید شبکه‌های عصبی سنتی را روی مجموعه داده ردپای دایناسورها اعمال کنیم و نتایج را با مدل‌های کلاسیک فصل 3 مقایسه کنیم.

اول باید یک معماری انتخاب کنیم: یعنی تعداد لایه‌های پنهان، تعداد گره‌ها در هر لایه و نوع تابع فعال‌سازی برای هر گره. مجموعه داده ردپای دایناسورها دو کلاس دارد: اورنیتیشین (کلاس 0) و تروپود (کلاس 1). بنابراین گره خروجی باید از تابع فعال‌سازی سیگموئید استفاده کند تا احتمال تعلق به کلاس 1 را به ما بدهد. مقدار خروجی شبکه، احتمال اینکه تصویر ورودی متعلق به تروپود باشد را تخمین می‌زند. اگر این احتمال بیش از 50 درصد باشد، ورودی را به کلاس 1 نسبت می‌دهیم؛ در غیر این صورت به کلاس 0 می‌رود. برای گره‌های لایه پنهان هم مثل تمام مدل‌های این فصل از تابع فعال‌سازی واحد یک سو شده خطی (ReLU) استفاده می‌کنیم. فقط باید تعداد لایه‌های پنهان و تعداد گره‌ها در هر لایه را انتخاب کنیم.

در مجموعه داده ردپاها 1336 نمونه آموزشی وجود دارد. این مقدار زیاد نیست و ما هم داده‌ها را افزایش نمی‌دهیم، بنابراین به یک مدل نسبتاً کوچک نیاز داریم. مدل‌های بزرگ (با گره‌ها و لایه‌های زیاد) به مجموعه‌های آموزشی بزرگی نیاز دارند؛ در غیر این صورت وزن‌ها و بایاس‌های زیادی نسبت به تعداد نمونه‌های آموزشی وجود خواهد داشت که باید یاد گرفته شوند. بنابراین خودمان را به مدل‌های حداکثر دو لایه پنهان برای این مجموعه داده محدود می‌کنیم. برای تعداد گره‌ها در لایه‌های پنهان، اجازه می‌دهیم لایه اول از تعداد بسیار کم تا تقریباً دو برابر اندازه ورودی (1600 ویژگی - تصویر 40×40 پیکسل بازشده) متغیر باشد. اگر لایه دوم را هم آزمایش کنیم، تعداد گره‌های آن را به نصف تعداد گره‌های لایه اول محدود می‌کنیم.

ابتدا مجموعه‌ای از معماری‌های یک و دو لایه را آموزش می‌دهیم. سپس بهترین مدل از بین آنها را 100 بار آموزش می‌دهیم تا سطح میانگین عملکرد را به دست آوریم. جدول 1-4 نتایج مدل‌های آزمایشی را نشان می‌دهد.

وزن‌ها و بایاس‌ها	معماری	دقت (برحسب درصد)
16021	10	59.4
640801	400	77.0
1281601	800	76.7
3844801	2400	81.2
165201	100, 50	75.8
1361001	800, 100	81.2
5764001	2400, 800	77.9

شبکه‌ای که تنها 10 گره در لایه پنهان خود داشت، بدترین عملکرد را نشان داد و دقت حدود 60 درصد را ثبت کرد. یک طبقه‌بند دودویی که صرفاً مانند پرتاب سکه عمل کند، حدود 50 درصد مواقع پاسخ صحیح می‌دهد، بنابراین شبکه 10 گرهی تنها کمی بهتر از شانس عمل کرده است. این مدل مورد نظر ما نیست. بیشتر شبکه‌های دیگر دقتی در محدوده میانی تا بالای 70 درصد ارائه دادند.

دو مدلی که با رنگ قرمز مشخص شده‌اند هر دو دقتی کمی بیش از 81 درصد داشتند. مدل اول از یک لایه پنهان با 2400 گره استفاده می‌کرد. مدل دوم یک لایه پنهان 800 گرهی داشت که با لایه دیگری با 100 گره دنبال می‌شد. هر دو مدل دقت یکسانی روی مجموعه آزمایشی داشتند، اما مدل 2400 گرهی تقریباً سه برابر مدل دو لایه ای، وزن و بایاس داشت. بنابراین مدل دو لایه ای را انتخاب می‌کنیم. (توجه داشته باشید که نتایج جدول حاصل تنها یک جلسه آموزش است، نه میانگین چندین جلسه. این موضوع را به زودی اصلاح خواهیم کرد.)

مدل دو لایه ای هنوز نسبتاً بزرگ است. ما در حال تلاش برای یادگیری 1.4 میلیون پارامتر هستیم تا مدل را برای طبقه‌بندی صحیح تصاویر ردپای دایناسورها آماده کنیم. این تعداد پارامتر برای یادگیری بسیار زیاد است، به ویژه با مجموعه آموزشی که تنها 1336 نمونه دارد. شبکه‌های عصبی کاملاً متصل به سرعت از نظر تعداد پارامترهای مورد نیاز رشد می‌کنند. این مشاهده را در فصل 5 هنگام بحث در مورد شبکه‌های عصبی پیچشی دوباره بررسی خواهیم کرد.

معماری نهایی ما شامل دو لایه پنهان با توابع فعالسازی واحد یک سو شدهٔ خطی (ReLU) به ترتیب با 800 و 100 گره است که با یک گره سیگموئید برای محاسبه احتمال تعلق به کلاس 1 دنبال می‌شود. آموزش این مدل در 100 اجرای مختلف روی مجموعه داده ردپاها، میانگین دقت 77.4 درصد با حداقل 69.3 درصد و حداکثر 81.5 درصد را نشان داد. حال بیایید این نتیجه را در مقایسه با نتایج فصل 3 بررسی کنیم (به جدول 2-4 نگاه کنید).

Model	Accuracy (%)
RF300	83.3
RBF SVM	82.4
7-NN	80.0
3-NN	77.6
MLP	**77.4**
1-NN	76.1
Linear SVM	70.7

به خاطر داشته باشید که RF300 نشان‌دهنده یک جنگل تصادفی با 300 درخت است، SVM به ماشین بردار پشتیبان اشاره دارد، و NN به طبقه‌بند نزدیک‌ترین همسایه اشاره می‌کند. من از MLP (پرسپترون چندلایه) به عنوان نماینده شبکه عصبی خود استفاده می‌کنم. پرسپترون چندلایه نامی قدیمی اما همچنان رایج برای شبکه‌های عصبی سنتی است که در این فصل درباره آنها بحث کرده‌ایم - به ارتباط آن با پرسپترون اولیه روزنبلات از اواخر دهه 1950 توجه کنید.

شبکه عصبی ما بهترین عملکرد را در این مجموعه داده نداشت. در واقع، یکی از ضعیف‌ترین عملکردها را نشان داد. تنظیمات بیشتر ممکن است آن را یک یا دو رده در لیست بالاتر ببرد، اما این سطح عملکرد بنا به تجربه من معمول است و به درک کلی (کنایه به پرسپترون) قبل از انقلاب یادگیری عمیق دامن زده که شبکه‌های عصبی مدل‌های «چندان جالبی» نیستند - مدل‌های معمولی و بدون ویژگی خاصی که ارزش گزارش داشته باشند.

آنچه گذشت:

این فصل ایده‌های بنیادین شبکه‌های عصبی مدرن را معرفی کرد. مطالب باقی مانده کتاب بر مفاهیم پایه‌ای که در این فصل پوشش داده شد بنا شده‌اند. در اینجا نکات کلیدی را مرور می‌کنیم:

• شبکه‌های عصبی مجموعه‌ای از گره‌ها (نورون‌ها) هستند که چندین ورودی دریافت کرده و یک عدد به عنوان خروجی تولید می‌کنند.

• شبکه‌های عصبی معمولاً به صورت لایه‌ای سازماندهی می‌شوند، به طوری که ورودی لایه جاری، خروجی لایه قبلی است.

• شبکه‌های عصبی به صورت تصادفی مقداردهی اولیه می‌شوند، بنابراین آموزش مکرر منجر به مدل‌هایی با عملکردهای متفاوت می‌شود.

• شبکه‌های عصبی با روش گرادیان کاهشی آموزش می‌بینند که از جهت گرادیان محاسبه شده توسط الگوریتم پس‌انتشار برای به‌روزرسانی تکراری وزن‌ها و بایاس‌ها استفاده می‌کند.

حالا بیایید به بررسی شبکه‌های عصبی پیچشی بپردازیم – معماری‌ای که انقلاب یادگیری عمیق را رقم زد. این فصل ما را به اوایل دهه 2000 رساند. فصل بعدی ما را به سال 2012 و فراتر از آن خواهد برد.

کلمات کلیدی: تابع فعالسازی (Activation Function)، معماری (Architecture)، پاس عقب‌گرد (backward pass)، بایاس (Bias)، تقویت داده (Data Augmentation)، دوره (Epoch)، پاس به جلو (Forward Pass)، مینیمم مطلق (Global Minimum)، گرادیان کاهشی (Gradient Descent)، لایه پنهان (Hidden Layer)، نرخ یادگیری (Learning Rate)، مینیمم موضعی (Local Minimum)، هزینه (Loss)، مینی‌بَچ (Minibatch)، پرسپترون چند لایه (Multilayer Perceptron)، نورون (Neuron)، گره (Node)، بیش‌برازش (Overfitting)، پیش‌پردازش (Preprocessing)، واحد یک سو شدهٔ خطی (ReLU: Rectified Linear Unit)، سیگموئید (Sigmoid)، گرادیان کاهشی تصادفی (Stochastic Gradient Descent)، وزن (Weight)

فصل پنجم: شبکه‌های عصبی پیچشی (هوش مصنوعی می‌آموزد تا ببیند)

مدل‌های یادگیری ماشین کلاسیک با مشکلاتی مانند انتخاب ویژگی مناسب، ابعاد بردار ویژگی و ناتوانی در یادگیری از ساختار ذاتی ورودی‌ها مواجه هستند. شبکه‌های پیچشی (CNNها) این مشکلات را با یادگیری تولید بازنمایی‌های جدید از ورودی‌های خود و همزمان طبقه‌بندی آن‌ها برطرف می‌کنند، فرآیندی که به عنوان یادگیری سرتاسری (End-to-End Learning) شناخته می‌شود. شبکه‌های پیچشی پردازشگرهای داده‌ای هستند که در فصل دوم به آن‌ها اشاره کردم و به یادگیری بازنمایی اختصاص دارند. عناصری از آنچه که بعداً به شبکه‌های پیچشی تبدیل شد، در طول تاریخ شبکه‌های عصبی در زمان‌های مختلف ظاهر شدند، از جمله پرسپترون روزنبلات، اما معماری‌ای که انقلاب یادگیری عمیق را آغاز کرد، در سال 1998 منتشر شد. بیش از یک دهه بهبود در توانایی‌های محاسباتی لازم بود تا قدرت کامل شبکه‌های پیچشی با ظهور الکس‌نت (AlexNet) در سال 2012 آزاد شود. شبکه‌های پیچشی از ساختار موجود در ورودی‌های خود بهره‌برداری می‌کنند. با پیشرفت فصل، بهتر درک خواهیم کرد که این به چه معناست. در یک بُعد، ورودی‌ها ممکن است مقادیری باشند که با گذشت زمان تغییر می‌کنند، که به عنوان سری زمانی (Time Series) نیز شناخته می‌شوند. در دو بُعد، درباره تصاویر صحبت می‌کنیم. شبکه‌های پیچشی سه‌بُعدی برای تفسیر حجم‌های داده‌ای مانند مجموعه‌ای از تصاویر رزونانس مغناطیسی یا حجمی که از ابرِ نقاط LiDAR ساخته شده، کاربرد دارند. در این فصل، ما صرفاً بر شبکه‌های پیچشی دوبعدی تمرکز خواهیم کرد.

ترتیب ارائه ویژگی‌ها به یک شبکه عصبی سنتی اهمیتی ندارد. فرقی نمی‌کند که بردارهای ویژگی را به مدل به صورت (x_0,x_1,x_2) یا (x_2,x_0,x_1) ارائه دهیم، مدل به همان خوبی یاد می‌گیرد، زیرا فرض می‌کند ویژگی‌ها مستقل و بی‌ربط به یکدیگر هستند. در واقع، وجود همبستگی قوی بین مقدار یک پیکسل و مقادیر پیکسل‌های مجاور چیزی است که مدل‌های یادگیری ماشینی سنتی نمی‌خواهند، و ناتوانی آن‌ها در دستیابی به موفقیت زیاد با چنین ورودی‌هایی، شبکه‌های عصبی را برای سال‌ها عقب نگه داشت.

از سوی دیگر، شبکه‌های پیچشی از ساختار موجود در ورودی‌های خود بهره‌برداری می‌کنند. برای یک شبکه پیچشی، مهم است که ورودی را به صورت (x_0,x_1,x_2) یا (x_2,x_0,x_1) ارائه دهیم؛ مدل ممکن است با اولی خوب یاد بگیرد و با دومی ضعیف عمل کند. این یک ضعف نیست، بلکه یک قوت است، زیرا ما می‌خواهیم شبکه‌های پیچشی را در موقعیت‌هایی به کار ببریم که ساختاری برای یادگیری وجود دارد - ساختاری که به تعیین بهترین روش برای طبقه‌بندی ورودی‌ها کمک می‌کند.

بعداً در این فصل، عملکرد یک شبکه عصبی سنتی را با یک شبکه پیچشی در طبقه‌بندی عکس‌های کوچک حیوانات و وسایل نقلیه (مجموعه داده CIFAR-10 از فصل سوم) مقایسه خواهیم کرد. در آن زمان، قدرت واقعی بهره‌برداری از ساختار را خواهیم آموخت. اما قبل از آن، بیایید یک آزمایش کوچک انجام دهیم. ما دو مجموعه داده داریم. اولی دوست قدیمی ما، مجموعه داده ارقام MNIST است؛ دومی همان مجموعه تصاویر ارقام است، اما ترتیب پیکسل‌ها در تصاویر به هم ریخته شده است. این به‌هم‌ریختگی تصادفی نیست، بلکه ثابت است، به طوری که پیکسل در موقعیت (1,12) به موقعیت (26,13) منتقل شده است، و همین‌طور حرکت‌های مشابه برای سایر پیکسل‌ها. شکل 1-5 چند نمونه از ارقام MNIST و نسخه‌های به‌هم‌ریخته همان ارقام را نشان می‌دهد. ارقام به‌هم‌ریخته برای من غیرقابل فهم هستند. اطلاعات پیکسلی بین ارقام اصلی و ارقام به‌هم‌ریخته یکسان است - یعنی همان مجموعه مقادیر پیکسلی در هر دو وجود دارد - اما ساختار تا حد زیادی از بین رفته است و من دیگر

نمی‌توانم ارقام را تشخیص دهم. من ادعا می‌کنم که یک شبکه عصبی سنتی ورودی‌های خود را به‌صورت کلی در نظر می‌گیرد و به دنبال ساختار نیست. اگر این درست باشد، یک شبکه عصبی سنتی نباید اهمیتی به این بدهد که ارقام به‌هم‌ریخته شده‌اند؛ باید به همان خوبی یاد بگیرد وقتی با مجموعه داده اصلی یا به‌هم‌ریخته آموزش داده شود. همان‌طور که مشخص می‌شود، دقیقاً همین اتفاق می‌افتد. مدل به همان خوبی یاد می‌گیرد؛ به‌هم‌ریختگی هیچ تغییری در عملکرد ایجاد نمی‌کند. با این حال، توجه داشته باشید که ارقام آزمایشی به‌هم‌ریخته باید با مدل به‌هم‌ریخته استفاده شوند؛ نباید انتظار داشته باشیم که مدل وقتی روی یک مجموعه داده آموزش دیده و روی مجموعه داده دیگر آزمایش شود، کار کند.

ما در حال حاضر تنها یک واقعیت درباره شبکه‌های پیچشی (CNNها) می‌دانیم: آن‌ها به ساختار موجود در ورودی‌های خود توجه می‌کنند. با دانستن این موضوع، آیا باید انتظار داشته باشیم که یک شبکه پیچشی آموزش‌دیده روی مجموعه داده به‌هم‌ریخته به همان خوبی شبکه آموزش‌دیده روی مجموعه داده اصلی عمل کند؟ ارقام به‌هم‌ریخته برای ما غیرقابل تفسیر هستند، زیرا ساختار محلی در تصاویر از بین رفته است. بنابراین، ممکن است انتظار داشته باشیم مدلی که به طور مشابه به دنبال بهره‌برداری از ساختار محلی است، نتواند ارقام به‌هم‌ریخته را تفسیر کند. و همین‌طور است: یک شبکه پیچشی آموزش‌دیده روی مجموعه داده به‌هم‌ریخته در مقایسه با یکی که روی مجموعه داده اصلی آموزش دیده، عملکرد ضعیفی دارد.

چرا ما به راحتی نمی‌توانیم ارقام به‌هم‌ریخته را تفسیر کنیم؟ برای پاسخ به این سؤال، باید بررسی کنیم که در طی فرآیند بینایی در مغز چه اتفاقی می‌افتد. سپس به عقب برمی‌گردیم تا این فرآیند را با عملکرد شبکه‌های پیچشی مرتبط کنیم. همان‌طور که خواهیم آموخت، شبکه‌های پیچشی از ضرب‌المثل قدیمی پیروی می‌کنند: در روم، مانند رومی‌ها (انسان‌ها) عمل کن.

ونسان ون گوگ هنرمند مورد علاقه من است. چیزی در سبک او با من سخن می‌گوید، چیزی به طرز عجیبی آرامش‌بخش از مردی که از بیماری روانی رنج می‌برد. آرامشی که از آثار او ساطع می‌شود، تلاش او برای آرام کردن آشوب درونی‌اش را منعکس می‌کند. شکل 2-5 را در نظر بگیرید. این تصویر نقاشی معروف ون گوگ از اتاق خوابش در آرل (Arles) در سال 1889 را نشان می‌دهد.

در این نقاشی چه می‌بینید؟ پرسشم درباره معنا یا برداشتی عمیق نیست، بلکه به طور عینی، در نقاشی چه می‌بینید؟ من یک تخت، دو صندلی، یک میز کوچک، یک پنجره و یک پارچ روی میز می‌بینم، در کنار بسیاری از اشیای دیگر. گمان می‌کنم شما هم همین‌ها را می‌بینید. شما تخت، دو صندلی و میز را دیدید، اما چگونه؟ فوتون‌ها، ذرات نور، از تصویر به چشم شما سفر کردند و در مغز شما به اشیای مجزا تبدیل شدند. باز هم، چگونه؟

من سؤالاتی می‌پرسم اما هنوز پاسخی ارائه نکرده‌ام. این اشکالی ندارد به دو دلیل. اول، تأمل در مسئله تقسیم‌بندی یک تصویر به مجموعه‌ای از اشیای معنادار ارزش تلاش ما را دارد. دوم، هنوز کسی پاسخ کامل به سؤال «چگونه؟» را نمی‌داند. با این حال، دانشمندان علوم اعصاب شروع این فرآیند را درک کرده‌اند.

ما توانایی نگاه کردن به یک صحنه و تجزیه آن به اشیای جداگانه و شناسایی‌شده را بدیهی می‌دانیم. برای ما، این فرآیند بدون تلاش و کاملاً خودکار است. اما نباید فریب بخوریم. ما بهره‌مند از صدها میلیون سال دستکاری تکامل هستیم. برای پستانداران، بینایی در چشم آغاز می‌شود، اما تجزیه و درک آن در قشر بصری اولیه در پشت مغز ما شروع می‌شود.

قشر بصری اولیه، که به عنوان ناحیه V1 شناخته می‌شود، به لبه‌ها و جهت‌گیری حساس است. بلافاصله، سرنخی از چگونگی عملکرد بینایی در مغز (در مقابل چشم) به دست می‌آید. مغز حس‌های ورودی را می‌گیرد، آن‌ها را به صورت تصویری تغییرشکل‌یافته روی V1 پخش می‌کند و با جستجوی لبه‌ها و جهت‌گیری آن‌ها شروع می‌کند. V1 همچنین به رنگ حساس است. نگاشت کل میدان بصری روی V1، با بزرگ‌نمایی به‌گونه‌ای که بیشتر V1 توسط دو درصد مرکزی میدان بصری ما اشغال شود، به این معناست که تشخیص لبه، جهت‌گیری و رنگ به صورت محلی در جایی که رخ می‌دهند، انجام می‌شود.

V1 تشخیص‌های خود را به ناحیه V2 می‌فرستد، که تشخیص‌های خود را به ناحیه V3 و به همین ترتیب از طریق V4 به V5 می‌فرستد، و هر ناحیه اساساً بازنمایی بزرگ‌تر و گروه‌بندی‌شده‌تری از آنچه در میدان بصری است دریافت می‌کند. این فرآیند با V1 شروع می‌شود و در نهایت بازنمایی کاملاً تجزیه‌شده و درک‌شده‌ای از آنچه چشم‌ها می‌بینند ارائه می‌دهد.

همان‌طور که اشاره شد، جزئیات فراتر از V1 هنوز مبهم است، اما برای اهداف ما، تنها چیزی که باید به خاطر بسپاریم این است که V1 به لبه‌ها، جهت‌گیری لبه‌ها و رنگ‌ها حساس است (ممکن است بافت‌ها را نیز شامل شود). شروع ساده و گروه‌بندی برای جداسازی اشیا در صحنه، نام این بازی است.

شبکه‌های پیچشی (CNNها) این فرآیند را تقلید می‌کنند. می‌توان گفت که شبکه‌های پیچشی به معنای واقعی کلمه یاد می‌گیرند که دنیای ورودی‌های خود را ببینند. شبکه‌های پیچشی ورودی‌ها را به بخش‌های کوچک تجزیه می‌کنند، سپس گروه‌هایی از بخش‌ها و گروه‌های بزرگ‌تری از گروه‌های بخش‌ها را تشکیل می‌دهند، تا زمانی که کل ورودی از یک کل واحد به یک بازنمایی جدید تبدیل شود: بازنمایی‌ای که به راحتی توسط چیزی که به عنوان یک شبکه عصبی سنتی در رأس مدل قرار دارد، قابل درک است. با این حال، نگاشت ورودی به یک بازنمایی جدید و قابل فهم‌تر به این معنا نیست که این بازنمایی جدید برای ما نیز به راحتی قابل درک است.

شبکه‌های پیچشی طی آموزش یاد می‌گیرند که ورودی‌ها را به بخش‌هایی تقسیم کنند، و این امکان را به لایه‌های بالایی شبکه می‌دهند تا با موفقیت طبقه‌بندی کنند. به عبارت دیگر، شبکه‌های پیچشی بازنمایی‌های جدیدی از ورودی‌های خود یاد می‌گیرند

و سپس آن بازنمایی‌های جدید را طبقه‌بندی می‌کنند. واقعیتش را بخواهید، «یادگیری بازنمایی‌های جدید از قدیمی‌ها» عنوان اولیه‌ای برای این فصل بود.

شبکه‌های پیچشی چگونه ورودی‌های خود را به بخش‌ها تجزیه می‌کنند؟ برای پاسخ به این سؤال، ابتدا باید قسمت «پیچش» در «شبکه عصبی پیچشی» را درک کنیم. هشدار! جزئیات ریزی در پیش است.

پیچش یک عملیات ریاضی با تعریفی رسمی است که شامل حساب انتگرال می‌شود. خوشبختانه برای ما، پیچش در تصاویر دیجیتال عملیاتی ساده است که تنها از ضرب و جمع استفاده می‌کند.

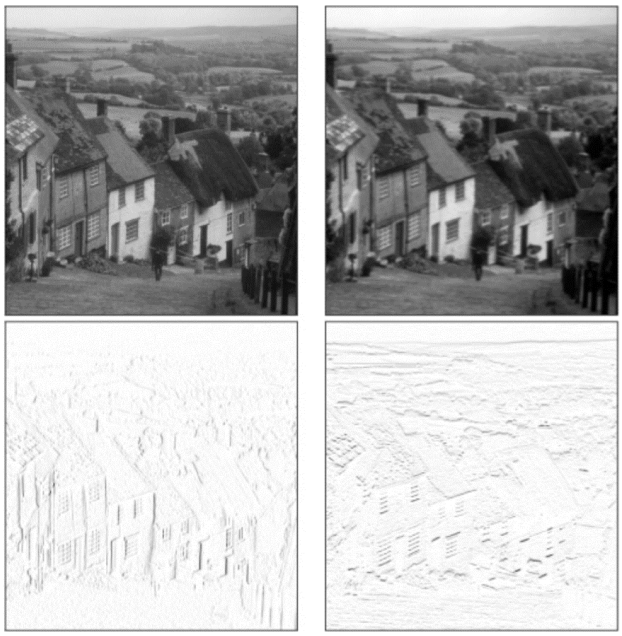
پیچش یک مربع کوچک، که به عنوان هسته (Kernel) شناخته می‌شود، را از بالا به پایین و از چپ به راست روی تصویر می‌لغزاند. در هر موقعیت، پیچش مقادیر پیکسل‌های پوشیده شده توسط مربع را در مقادیر متناظر هسته ضرب می‌کند. سپس تمام این حاصل‌ضرب‌ها را جمع می‌کند تا یک عدد واحد تولید شود که به عنوان مقدار پیکسل خروجی برای آن موقعیت در نظر گرفته می‌شود. کلمات تنها تا حدی می‌توانند اینجا کمک کنند، پس بیایید یک تصویر را امتحان کنیم. تصویر 3-5 را در نظر بگیرید.

$$\begin{matrix} 60 & 58 & 60 & 60 & 60 & 60 & 60 & 52 \\ 68 & 60 & 60 & 68 & 68 & 52 & 76 & 76 \\ 76 & 44 & 60 & 60 & 68 & 52 & 60 & 52 \\ 68 & 68 & 76 & 76 & 68 & 52 & 60 & 44 \\ 92 & 84 & 84 & 84 & 76 & 44 & 60 & 44 \\ 76 & 68 & 84 & 76 & 76 & 52 & 60 & 52 \\ 68 & 60 & 60 & 76 & 84 & 52 & 84 & 60 \\ 60 & 60 & 68 & 68 & 68 & 52 & 76 & 68 \end{matrix} \times \begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix} = \begin{matrix} -60 & -120 & -68 \\ 0 & 0 & 0 \\ 68 & 152 & 76 \end{matrix} = 48$$

سمت چپ شکل 3-5 شبکه‌ای از اعداد را نشان می‌دهد. این‌ها مقادیر پیکسل برای بخش مرکزی تصویر در شکل 4-5 هستند. مقادیر پیکسل در مقیاس خاکستری معمولاً در بازه 0 تا 255 قرار دارند، که در آن مقادیر پایین‌تر تیره‌تر هستند.

هسته، شبکه 3×3 در سمت راست است. عملیات پیچش به ما دستور می‌دهد که هر مقدار پیکسل را در مقدار متناظر هسته ضرب کنیم. این کار شبکه 3×3 جدیدی از اعداد تولید می‌کند. مرحله نهایی جمع تمام نه مقدار است تا یک خروجی واحد، 48، ایجاد شود که پیکسل مرکزی در تصویر خروجی را جایگزین می‌کند، 48 ⟶ 60.

برای تکمیل عملیات پیچش، جعبهٔ 3×3 را یک پیکسل به سمت راست بکشید و تکرار کنید. وقتی به انتهای یک ردیف رسیدید، جعبه را یک پیکسل به پایین حرکت دهید و برای ردیف بعدی تکرار کنید، سپس ردیف به ردیف پردازش کنید تا هسته کل تصویر را پوشش دهد. تصویر پیچشی مجموعه‌ای از پیکسل‌های خروجی جدید است.



در ابتدا، پیچش ممکن است عملی عجیب به نظر برسد. با این حال، در تصاویر دیجیتال، عملیاتی اساسی است. یک هسته که به طور مناسب تعریف شده باشد، به ما اجازه می‌دهد تصویر را فیلتر کنیم تا به روش‌های مختلف بهبود یابد. به عنوان مثال، شکل 4-5 چهار تصویر را نشان می‌دهد. تصویر بالا سمت چپ، تصویر اصلی است، یک تصویر آزمایشی پرکاربرد از گلد هیل در شفتسبری، انگلستان. سه تصویر دیگر نسخه‌های فیلترشده از تصویر اصلی هستند. در جهت عقربه‌های ساعت از بالا سمت راست، ما یک نسخه تار شده، یکی که لبه‌های افقی را نشان می‌دهد و دیگری که لبه‌های عمودی را نشان می‌دهد، داریم. هر تصویر با پیچش یک هسته، همان‌طور که قبلاً توضیح داده شد، تولید شده است. هسته شکل 3-5 تصویر لبه‌های افقی در پایین سمت راست را تولید می‌کند. هسته را نود درجه بچرخانید، و تصویر لبه‌های عمودی در پایین سمت چپ را به دست می‌آورید. در نهایت، تمام مقادیر هسته را یک کنید، و تصویر تار شده در بالا سمت راست را به دست می‌آورید. توجه داشته باشید که تصاویر لبه‌ها معکوس شده‌اند تا لبه‌های تشخیص داده شده به جای سفید، سیاه باشند. نکته کلیدی که باید به خاطر داشته باشیم این است که پیچش یک تصویر با هسته‌های مختلف، جنبه‌های متفاوتی از تصویر را برجسته می‌کند. تصور یک مجموعه مناسب از هسته‌ها که ساختار مرتبط با طبقه‌بندی صحیح تصویر را استخراج می‌کنند، دشوار نیست. این دقیقاً همان کاری است که شبکه‌های پیچشی (CNNها) طی آموزش سرتاسری انجام می‌دهند و به نوعی، همان کاری است که سیستم بصری ما در ناحیه V1 انجام می‌دهد وقتی لبه‌ها، جهت‌گیری‌ها، رنگ‌ها و بافت‌ها را تشخیص می‌دهد.

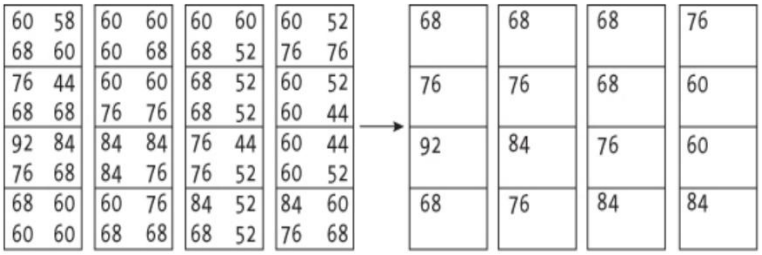
ما در حال پیشرفت هستیم. اکنون درک خوبی از عملیات اصلی CNN، یعنی پیچش داریم، پس بیایید گام بعدی را برداریم تا یاد بگیریم چگونه پیچش در داخل یک مدل برای استخراج ساختار و ساخت یک بازنمایی جدید از ورودی استفاده می‌شود.

شبکه‌های عصبی سنتی فصل چهارم از یک نوع لایه تشکیل شده‌اند: مجموعه‌ای از گره‌های کاملاً متصل که ورودی را از لایه پایین دریافت می‌کنند تا برای لایه بالایی خروجی تولید کنند. شبکه‌های عصبی پیچشی انعطاف‌پذیرتر هستند و از انواع لایه‌های متنوع پشتیبانی می‌کنند. با این حال، جریان داده یکسان است: از ورودی به لایه‌های متوالی تا خروجی شبکه.

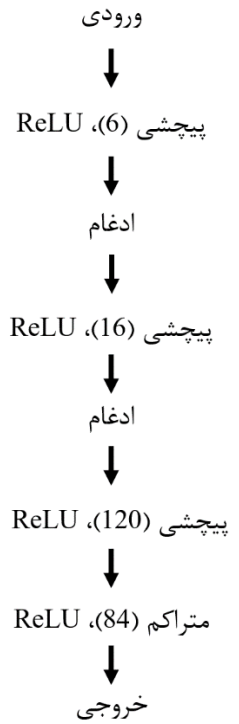
در اصطلاح شبکه‌های پیچشی، لایه‌های کاملاً متصل که یک شبکه عصبی سنتی استفاده می‌کند، لایه‌های متراکم (Dense) نامیده می‌شوند. شبکه‌های پیچشی معمولاً از لایه‌های متراکم در بالا، نزدیک به خروجی، استفاده می‌کنند، زیرا تا آن زمان شبکه ورودی را به یک بازنمایی جدید تبدیل کرده است، بازنمایی‌ای که لایه‌های کاملاً متصل می‌توانند با موفقیت طبقه‌بندی کنند. شبکه‌های پیچشی به شدت از لایه‌های پیچشی و لایه‌های ادغام (Pooling Layers) استفاده می‌کنند.

لایه‌های پیچشی مجموعه‌ای از هسته‌ها را به ورودی خود اعمال می‌کنند تا خروجی‌های متعددی تولید کنند، همان‌طور که در شکل 4-5 از یک تصویر ورودی در بالا سمت چپ سه خروجی تولید شد. هسته‌ها طی آموزش با استفاده از همان رویکرد پس‌انتشار (Backpropagation) و گرادیان کاهشی که در فصل چهارم با آن مواجه شدیم، یاد گرفته می‌شوند. مقادیر هسته‌های یادگرفته‌شده، وزن‌های لایه پیچشی هستند.

لایه‌های ادغام هیچ وزنی مرتبط با خود ندارند. چیزی برای یادگیری وجود ندارد. در عوض، لایه‌های ادغام عملیاتی ثابت روی ورودی‌های خود انجام می‌دهند: آن‌ها وسعت فضایی ورودی‌های خود را با نگه‌داری بزرگ‌ترین مقدار یک مربع 2×2 که بدون همپوشانی در عرض و سپس به سمت پایین حرکت می‌کند، کاهش می‌دهند. اثر نهایی مشابه کاهش اندازه تصویر به نصف است. شکل 5-5 فرآیند تبدیل یک ورودی 8×8 به یک خروجی 4×4 را با نگه داشتن مقدار حداکثر در هر مربع نشان می‌دهد. لایه‌های ادغام، امتیازی برای کاهش تعداد پارامترها در شبکه هستند.



یک شبکه پیچشی معمولی، لایه‌های پیچش و ادغام را قبل از افزودن یک یا دو لایه متراکم ترکیب می‌کند. لایه‌های ReLU نیز استفاده می‌شوند؛ البته معمولاً پس از لایه‌های پیچشی و متراکم. به عنوان مثال، یک معماری کلاسیک شبکه پیچشی به نام LeNet از لایه‌های زیر تشکیل شده است:



این مدل از سه لایهٔ پیچشی، دو لایهٔ ادغام و یک لایهٔ متراکم با 84 گره استفاده می‌کند. هر لایه پیچشی و متراکم با یک لایه ReLU دنبال می‌شود تا تمام ورودی‌های منفی را به صفر نگاشت کند، در حالی که ورودی‌های مثبت بدون تغییر باقی می‌مانند.

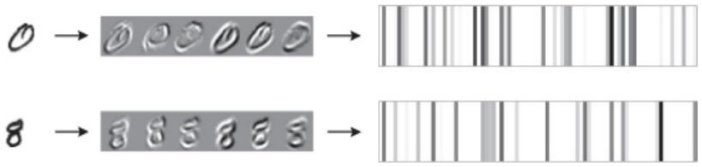
عدد داخل پرانتز برای هر لایه پیچشی، تعداد فیلترهایی است که باید در آن لایه یاد گرفته شوند. یک فیلتر مجموعه‌ای از هسته‌های پیچشی است، با یک هسته برای هر کانال ورودی. به عنوان مثال، لایه پیچشی اول شش فیلتر را یاد می‌گیرد. ورودی، یک تصویر در مقیاس خاکستری با یک کانال است، بنابراین این لایه شش هسته را یاد می‌گیرد. لایه پیچشی دوم 16 فیلتر را یاد می‌گیرد، هر کدام با 6 هسته، یکی برای هر یک از 6 کانال ورودی از لایه پیچشی اول. بنابراین، لایه پیچشی دوم در مجموع

96 هسته را یاد می‌گیرد. در نهایت، لایه پیچشی آخر 120 فیلتر را یاد می‌گیرد، هر کدام با 16 هسته، که در مجموع 1920 هسته دیگر می‌شود. در کل، مدل LeNet نیاز به یادگیری 2022 هسته پیچشی مختلف دارد.

امید این است که یادگیری این تعداد هسته، دنباله‌ای از خروجی‌ها را تولید کند که عناصر اساسی ساختارهای موجود در ورودی را ضبط کند. اگر آموزش موفقیت‌آمیز باشد، خروجی لایه پیچشی نهایی، به عنوان یک بردار ورودی به لایه متراکم، حاوی مقادیری خواهد بود که به وضوح بین کلاس‌ها تمایز قائل می‌شوند - حداقل، واضح‌تر از آنچه که تنها با استفاده از تصویر می‌توان به دست آورد.

اگر احساس می‌کنید که در جزئیات غرق شده‌ایم، همین‌طور است، اما دیگر عمیق‌تر نمی‌شویم. در واقع، ما به جزئی‌ترین سطح در این کتاب رسیده‌ایم، اما این یک بار ضروری است، زیرا اگر پیچش و لایه‌های پیچشی را درک نکنیم، نمی‌توانیم بفهمیم شبکه‌های پیچشی چگونه کار می‌کنند.

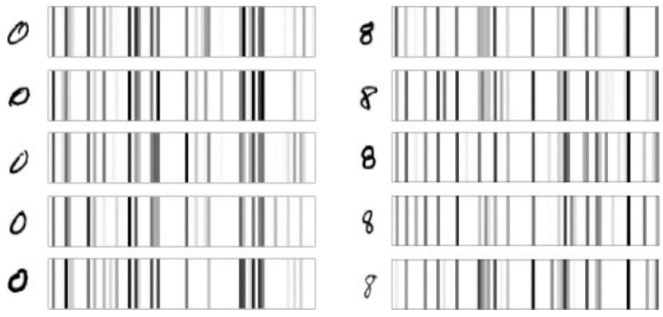
شاید بهترین راه برای درک کارکرد لایه‌های یک شبکه پیچشی، مشاهده تأثیر آن‌ها بر داده‌هایی باشد که از شبکه عبور می‌کنند. شکل 5-6 نشان می‌دهد که چگونه یک مدل LeNet آموزش‌دیده روی ارقام MNIST دو تصویر ورودی را دستکاری می‌کند. خروجی لایه پیچشی اول، شش تصویر میانی است، جایی که رنگ خاکستری نشان‌دهنده صفر است، پیکسل‌های تیره‌تر به طور فزاینده منفی هستند و پیکسل‌های روشن‌تر به طور فزاینده مثبت هستند. شش هسته لایه پیچشی اول هر کدام یک تصویر خروجی برای تصویر ورودی واحد تولید می‌کنند. هسته‌ها (Kernels) بخش‌های مختلف ورودی را به عنوان گذار از تیره به روشن برجسته می‌کنند.



الگوی بارکدمانند سمت راست، بازنمایی خروجی لایه متراکم است. ما خروجی لایه‌های پیچشی دوم و سوم را نادیده می‌گیریم و مستقیماً به انتهای مدل می‌رویم. خروجی لایه متراکم یک بردار از 84 عدد است. برای شکل 5-6، من این اعداد را به مقادیر پیکسل نگاشت کردم، جایی که مقادیر بزرگ‌تر به نوارهای عمودی تیره‌تر اختصاص دارند.

توجه کنید که بارکدها برای ارقام 0 و 8 متفاوت هستند. اگر مدل به خوبی یاد گرفته باشد، ممکن است انتظار داشته باشیم که بارکدهای خروجی لایه متراکم برای ارقام، اشتراکاتی داشته باشند. به عبارت دیگر، بارکدهای مربوط به صفرها باید تقریباً شبیه به هم باشند، همان‌طور که بارکدهای مربوط به هشت‌ها. آیا این‌طور است؟

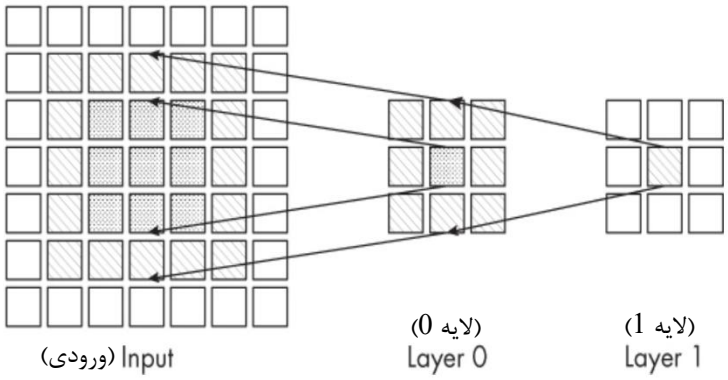
شکل 5-7 را در نظر بگیرید.



این شکل خروجی‌های لایه متراکم را برای پنج ورودی مختلف صفر و هشت نشان می‌دهد. بارکدها همه متفاوت هستند اما بر اساس نوع رقم شباهت‌هایی دارند. این امر به‌ویژه برای صفرها صادق است. مدل LeNet یاد گرفته است که چگونه هر تصویر ورودی 28×28 پیکسل (784 پیکسل) را به برداری از 84 عدد نگاشت کند که شباهت‌های قوی‌ای بر اساس نوع رقم نشان می‌دهند. بر اساس تجربه ما با شبکه‌های عصبی سنتی، می‌توانیم درک کنیم که این نگاشت چیزی با ابعاد کمتر تولید کرده است که تفاوت‌ها بین ارقام را حفظ کرده و حتی بر آن‌ها تأکید می‌کند. بردار یادگرفته‌شده با ابعاد کمتر شبیه به یک مفهوم پیچیده است که با چند کلمهٔ به‌خوبی انتخاب‌شده، توضیح داده شده است. این دقیقاً همان چیزی است که ما از یک شبکه پیچشی انتظار داریم. مدل آموزش‌دیده یاد گرفته است که در دنیای ارقام دست‌نویس که به صورت تصاویر کوچک در مقیاس خاکستری نمایش داده شده‌اند، «ببیند». هیچ چیز خاصی در مورد تصاویر خاکستری وجود ندارد. شبکه‌های پیچشی کاملاً خوشحال می‌شوند که با تصاویر رنگی که توسط کانال‌های قرمز، سبز و آبی نمایش داده شده‌اند، یا هر تعداد کانال دیگر، مانند هنگام استفاده از تصاویر ماهواره‌ای چندباندی، کار کنند.

می‌توانیم مدل را این‌گونه در نظر بگیریم: لایه‌های شبکه پیچشی قبل از لایه متراکم یاد گرفته‌اند که چگونه به عنوان یک تابع عمل کنند و یک بردار خروجی از تصویر ورودی تولید کنند. طبقه‌بندی‌کننده واقعی، لایه متراکم در بالا است، اما این لایه به خوبی کار می‌کند زیرا شبکه پیچشی همزمان با یادگیری تابع نگاشت، طبقه‌بندی‌کننده (لایه متراکم) را نیز یاد گرفته است.

پیش‌تر گفتم که لایه‌های بالاتر در شبکه پیچشی به بخش‌های هرچه بزرگ‌تر از ورودی توجه می‌کنند. می‌توانیم ببینیم که این موضوع درست است (با در نظر گرفتن بخشی از ورودی که بر خروجی یک هسته در یک لایه عمیق‌تر تأثیر می‌گذارد). شکل 8-5 این اثر را نشان می‌دهد.



از سمت راست تصویر شروع کنید. شبکه 3×3 از مربع‌ها نشان‌دهنده خروجی یک هسته در لایه پیچشی 1 است. ما می‌خواهیم بدانیم کدام بخش از ورودی بر مقدار پیکسل سایه‌دار تأثیر می‌گذارد. با نگاه به لایه پیچشی قبلی، یعنی لایه 0، می‌بینیم که خروجی لایه 1 به نُه مقدار سایه‌دار که از لایه قبل می‌آیند، وابسته است.

نُه مقدار سایه‌دار لایه پیچشی 0 به ناحیه سایه‌دار 5×5 از ورودی وابسته هستند. این ناحیه 5×5 است زیرا هر یک از این نُه مقدار با لغزاندن یک هسته 3×3 روی ناحیه سایه‌دار 5×5 از ورودی به دست می‌آید. به عنوان مثال، بخش نقطه‌چین مقدار میانی در لایه 0 از ناحیه سایه‌دار 3×3 مشابه در ورودی می‌آید. به این ترتیب، لایه‌های بالاتر شبکه پیچشی تحت تأثیر بخش‌های بزرگ‌تر و بزرگ‌تر از ورودی قرار می‌گیرند. اصطلاح فنی برای این موضوع «میدان دریافت مؤثر» است، که در آن میدان دریافت مؤثر مقدار سایه‌دار سمت راست در شکل 8-5، ناحیه سایه‌دار 5×5 از ورودی است.

وقت آزمایش است. حالا که درک خوبی از نحوه عملکرد شبکه‌های پیچشی (CNNها) داریم، بیایید این دانش را به کار بگیریم تا یک شبکه عصبی سنتی را با یک مدل پیچشی مقایسه کنیم. کدام یک برنده خواهد شد؟ گمان می‌کنم شما از قبل پاسخ را می‌دانید، اما بیایید آن را ثابت کنیم و در این مسیر تجربه‌ای کسب کنیم.

ما به یک مجموعه داده نیاز داریم. بیایید از نسخه خاکستری مجموعه داده CIFAR-10 استفاده کنیم. این انتخاب بهتری نسبت به مجموعه داده ردپای دایناسور است که در دو فصل گذشته استفاده کردیم، زیرا تصاویر ردپا فقط خطوط کلی بدون بافت و پس‌زمینه هستند و یک شبکه پیچشی چیز زیادی بیشتر از یک مدل سنتی از چنین تصاویری یاد نخواهد گرفت. همان‌طور که در فصل سوم آموختیم، CIFAR-10 شامل تصاویر 32×32 پیکسلی از حیوانات و وسایل نقلیه است که احتمالاً چالش‌برانگیزتر خواهند بود.

ما سه مدل را آموزش خواهیم داد: یک جنگل تصادفی (Random Forest)، یک شبکه عصبی سنتی و یک شبکه عصبی پیچشی. آیا این کافی است؟ ما به این درک رسیده‌ایم که هر سه این مدل‌ها شامل تصادفی بودن هستند، بنابراین یک بار آموزش ممکن است نمایانگر منصفانه‌ای از عملکرد هر مدل به ما ندهد.

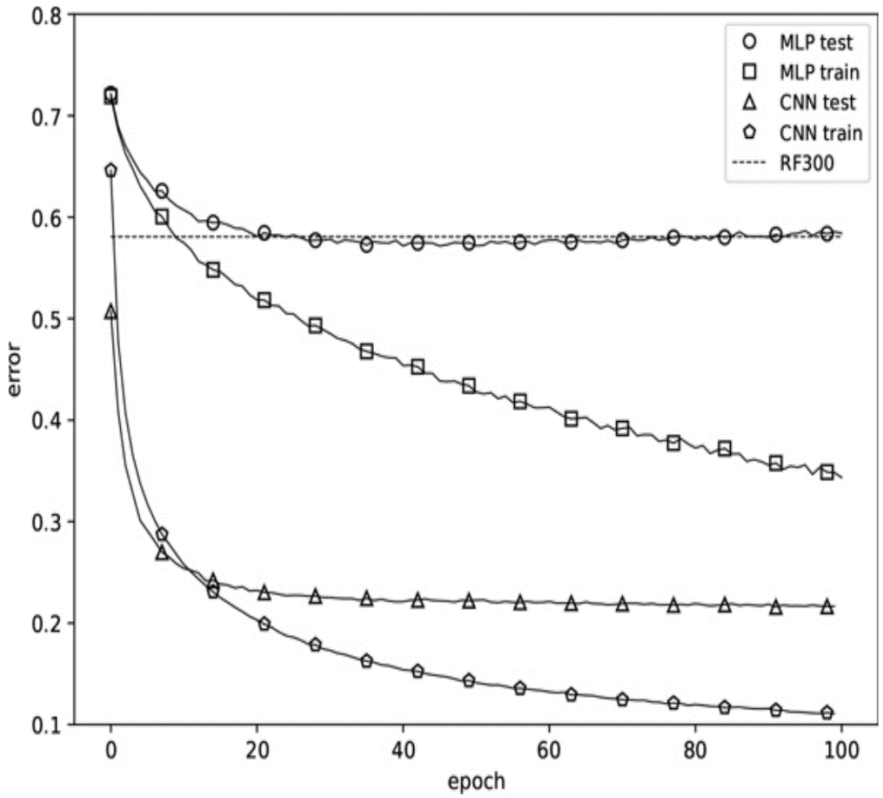
ممکن است یک مقداردهی اولیه ضعیف یا ترکیبی از درختان داشته باشیم که یکی از مدل‌ها را از مسیر خارج کند. بنابراین، بیایید هر مدل را 10 بار آموزش دهیم و نتایج را میانگین بگیریم.

این آزمایش به ما کمک می‌کند تا تفاوت‌های عملکرد بین مدل‌ها را درک کنیم، اما می‌توانیم با ردیابی خطاهای شبکه‌های عصبی در حین پیشرفت آموزش، چیزهای بیشتری درباره آن‌ها یاد بگیریم. نتیجه یک نمودار است که به زودی آن را ارائه و توضیح خواهم داد. اما قبل از آن، اجازه دهید جزئیات مدل‌ها را شرح دهم.

مجموعه داده‌های آموزشی و آزمایشی برای هر مدل یکسان هستند. شبکه عصبی سنتی و جنگل تصادفی به ورودی‌های برداری نیاز دارند، بنابراین هر تصویر 32×32 پیکسلی به یک بردار با 1024 عدد تبدیل می‌شود. شبکه پیچشی با تصاویر دوبعدی واقعی کار می‌کند. مجموعه آموزشی شامل 50000 تصویر است، 5000 تصویر برای هر یک از 10 کلاس، و مجموعه آزمایشی شامل 10000 تصویر است، 1000 تصویر برای هر کلاس.

جنگل تصادفی از 300 درخت استفاده می‌کند. شبکه عصبی سنتی دارای دو لایه پنهان با 512 و 100 گره است. شبکه پیچشی پیچیده‌تر است، با چهار لایه پیچشی، دو لایه ادغام و یک لایه متراکم با 472 گره. اگرچه شبکه پیچشی لایه‌های بیشتری دارد، تعداد کل وزن‌ها و بایاس‌هایی که باید یاد گرفته شوند تقریباً با مدل سنتی یکسان است: 577014 در مقابل 577110.

شبکه‌های عصبی را برای 100 دوره (Epoch) آموزش خواهیم داد، به این معنا که 100 بار مجموعه آموزشی کامل را مرور می‌کنیم. با ثابت کردن اندازه مینی‌بچ در 200، در هر دوره 250 گام کاهش گرادیان خواهیم داشت. بنابراین، در طول آموزش، وزن‌ها و بایاس‌های شبکه‌ها 25000 بار به‌روزرسانی می‌شوند. در پایان هر دوره، خطای مدل را هم روی مجموعه آموزشی و هم روی مجموعه آزمایشی ثبت خواهیم کرد. نهایتاً، یک نمودار واحد همه چیزهایی را که می‌خواهیم بدانیم نشان خواهد داد. شکل 5-9 همان نمودار است. این پیچیده‌ترین نموداری است که دیده‌ایم، پس بیایید آن را با جزئیات بررسی کنیم و از محورها شروع کنیم.



برچسب روی محور افقی (محور x) دوره (Epoch) است، که به معنای یک مرور کامل بر مجموعه آموزشی است. بنابراین، نمودار تغییرات را در طول آموزش پس از هر دوره نشان می‌دهد. همچنین می‌دانیم که هر دوره نمایانگر 250 گام کاهش گرادیان است. محور عمودی (محور y) با عنوان «خطا» برچسب‌گذاری شده و از 0.1 تا 0.8 ادامه دارد. این محور نشان‌دهنده کسری از نمونه‌های آزمایشی یا آموزشی است که مدل به اشتباه پیش‌بینی می‌کند. هرچه خطا کمتر باشد، بهتر است. مقدار اعشاری 0.1 به معنای 10 درصد و مقدار 0.8 به معنای 80 درصد است.

راهنمای موجود در گوشه بالا سمت راست نمودار به ما می‌گوید که دایره‌ها و مربع‌ها به شبکه عصبی سنتی (MLP) مربوط می‌شوند، در حالی که مثلث‌ها و پنج‌ضلعی‌ها به شبکه پیچشی (CNN) اشاره دارند. به طور خاص همان‌طور که مدل‌ها آموزش می‌بینند، دایره‌ها و مثلث‌ها به ترتیب خطا را در مجموعه آزمایشی برای MLP و CNN ردیابی می‌کنند. به طور مشابه، مربع‌ها و پنج‌ضلعی‌ها خطا را در مجموعه آموزشی ردیابی می‌کنند.

به خاطر بیاورید که عملکرد مدل روی مجموعه آموزشی برای به‌روزرسانی وزن‌ها و بایاس‌ها استفاده می‌شود. مجموعه آزمایشی برای ارزیابی به کار می‌رود و در آموزش مدل نقشی ندارد.

نمودارهای MLP ما نشان می‌دهند که مدل چقدر خوب مجموعه آموزشی (مربع‌ها) و مجموعه آزمایشی (دایره‌ها) را در طول آموزش، دوره به دوره، یاد گرفته است. بلافاصله مشخص است که مدل مجموعه آموزشی را بهتر از مجموعه آزمایشی یاد گرفته، زیرا خطای مجموعه آموزشی به طور مداوم کاهش می‌یابد. این همان چیزی است که انتظار داریم. الگوریتم گرادیان کاهشی وزن‌ها و بایاس‌های MLP را، که در مجموع 577110 عدد هستند، به‌روزرسانی می‌کند تا به خطای کمتر و کمتری در مجموعه آموزشی برسد. با این حال، ما به دنبال رسیدن به خطای صفر در مجموعه آموزشی نیستیم؛ بلکه می‌خواهیم کمترین خطای ممکن را در مجموعه آزمایشی داشته باشیم، زیرا این به ما دلیلی می‌دهد تا باور کنیم MLP توانسته است در تعمیم‌دهی موفق عمل کند.

حالا به نمودار دایره‌ها که خطای مجموعه آزمایشی را نشان می‌دهد توجه کنید. این خطا در حدود 40 دوره به حداقل 0.56 یا 56 درصد می‌رسد. پس از آن، خطا به آرامی اما به طور پیوسته تا 100 دوره افزایش می‌یابد. این اثر، بیش‌برازش (Overfitting) کلاسیک MLP است. خطای مجموعه آموزشی همچنان کاهش می‌یابد، اما خطای مجموعه آزمایشی به حداقل می‌رسد و پس از آن افزایش می‌یابد. شکل 9-5 به ما می‌گوید که توقف آموزش در 40 دوره بهترین عملکرد MLP را به ما می‌داد.

به نتایج شبکه پیچشی (CNN) خواهیم رسید، اما در حال حاضر، خط چین در 58 درصد خطا را در نظر بگیرید. این خط با برچسب RF300 مشخص شده و خطای مجموعه آزمایشی از یک جنگل تصادفی با 300 درخت را نشان می‌دهد. جنگل تصادفی با به‌روزرسانی وزن‌ها در طول دوره‌ها یاد نمی‌گیرد، بنابراین خطای 58 درصد همان خطای مدل است. من آن را به صورت یک خط چین موازی با محور افقی رسم کرده‌ام تا بتوانید ببینید که برای مدت کوتاهی، MLP کمی بهتر از جنگل تصادفی عمل کرد، اما تا 100 دوره، تفاوت بین این دو مدل ناچیز بود. به عبارت دیگر، می‌توانیم نتیجه بگیریم که بهترین تلاش یادگیری ماشینی کلاسیک در مجموعه داده خاکستری CIFAR-10 خطایی حدود 56 تا 58 درصد است. این نتیجه خوبی نیست. صرف زمان بیشتر با پارامترهای جنگل تصادفی، یا MLP، یا شروع دوباره با یک ماشین بردار پشتیبان ممکن است به کاهش جزئی خطا منجر شود. با این حال، بعید است که بتواند بر این واقعیت غلبه کند که یادگیری ماشینی کلاسیک نمی‌تواند کار زیادی با این مجموعه داده انجام دهد.

در نهایت، به منحنی‌های آموزشی (پنج‌ضلعی) و آزمایشی (مثلث) شبکه پیچشی توجه کنید. تا 100 دوره، شبکه پیچشی به خطای حدود 11 درصد در مجموعه آموزشی و مهم‌تر از آن، حدود 23 درصد در مجموعه آزمایشی می‌رسد. به عبارت دیگر، شبکه پیچشی در 77 درصد مواقع درست عمل می‌کند، یا تقریباً 8 بار از 10 بار. حدس تصادفی در یک مجموعه داده با 10 کلاس حدود 10 درصد مواقع درست خواهد بود، بنابراین شبکه پیچشی به خوبی یاد گرفته است و بسیار بهتر از MLP یا جنگل تصادفی عمل می‌کند.

این دقیقاً هدف شبکه‌های عصبی پیچشی است: با یادگیری بازنمایی بخش‌های اشیا در یک تصویر، امکان یادگیری یک بازنمایی جدید (که به طور رسمی به عنوان یک جاسازی یا Embedding شناخته می‌شود) فراهم می‌شود که لایه‌های متراکم شبکه می‌توانند با موفقیت آن را طبقه‌بندی کنند.

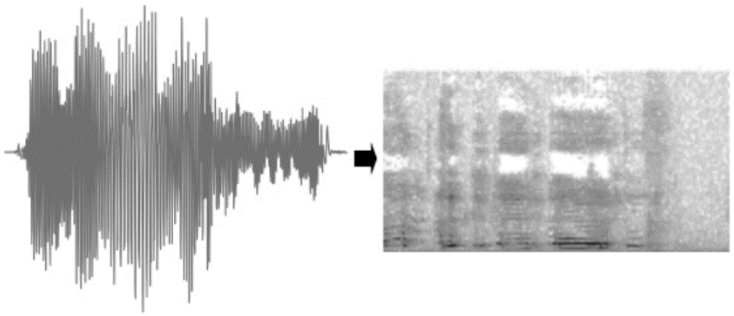
اولین شبکه پیچشی که من در سال 2015 آموزش دادم، سعی داشت هواپیماهای کوچک را در تصاویر ماهواره‌ای تشخیص دهد. رویکرد اولیه من، که غیرپیچشی بود، کار می‌کرد، اما پر از نویز و تشخیص‌های مثبت کاذب (تشخیص‌های اشتباه) بود. هواپیماها آنجا بودند، اما بسیاری از چیزهای دیگر که هواپیما نبودند نیز تشخیص داده می‌شدند. سپس یک شبکه پیچشی ساده مانند آنچه در این آزمایش استفاده شد آموزش دادم. این شبکه هواپیماها را به راحتی پیدا کرد و تقریباً هیچ چیز جز هواپیماها را تشخیص نداد. من متحیر شدم و در آن زمان متوجه شدم که یادگیری عمیق یک تغییر پارادایم است. در فصل هفتم استدلال خواهم کرد که تا پاییز 2022، یک تغییر پارادایم عمیق‌تر رخ داده است، اما هنوز باید زمینه‌هایی را پوشش دهیم تا برای آن بحث آماده شویم.

شبکه‌های پیچشی ساده این فصل تنها بخشی از معماری‌های شبکه عصبی متنوع موجود هستند. یک دهه توسعه پرشتاب منجر به ظهور چند معماری CNN پراستفاده شده است که برخی بیش از 100 لایه دارند. این معماری‌ها نام‌هایی مانند ResNet، DenseNet، Inception، MobileNet و U-Net دارند، در میان بسیاری دیگر. U-Net ارزش چند کلمه توضیح را دارد.

شبکه‌های پیچشی که تاکنون بررسی کردیم، یک تصویر ورودی را می‌پذیرند و یک برچسب کلاسی مانند «سگ» یا «گربه» بازمی‌گردانند. اما لازم نیست این‌گونه باشد. برخی معماری‌های CNN تقسیم‌بندی معنایی (Semantic Segmentation) را پیاده‌سازی می‌کنند، که در آن خروجی یک تصویر دیگر است که هر پیکسلش با کلاسی که به آن تعلق دارد برچسب‌گذاری شده است. U-Netها این کار را انجام می‌دهند. اگر هر پیکسل سگ با برچسب «سگ» علامت‌گذاری شود، استخراج سگ از تصویر به کاری پیش‌پاافتاده تبدیل می‌شود. حد وسطی بین U-Net و شبکه‌های پیچشی که برچسبی واحد به کل تصویر اختصاص می‌دهند، مدلی است که یک کادر محدودکننده (Bounding Box) به عنوان خروجی ارائه می‌دهد، یعنی یک مستطیل که شیء تشخیص‌داده‌شده را احاطه می‌کند.

گستردگی هوش مصنوعی به این معناست که احتمالاً قبلاً تصاویری با کادرهای محدودکننده برچسب‌گذاری‌شده دیده‌اید. YOLO یک معماری محبوب است که کادرهای محدودکننده برچسب‌گذاری‌شده تولید می‌کند؛ Faster R-CNN نیز یکی دیگر است.

ما اینجا روی ورودی‌های تصویری تمرکز کردیم، اما ورودی لازم نیست حتماً تصویر باشد. هر چیزی که بتوان آن را در قالبی شبیه به تصویر نمایش داد، جایی که دو بعد و ساختار درون این ابعاد وجود دارد، کاندیدایی برای یک شبکه پیچشی دوبعدی است. یک مثال خوب سیگنال صوتی است، که معمولاً آن را یک‌بعدی می‌دانیم، یعنی ولتاژی که با گذشت زمان تغییر می‌کند. با این حال، سیگنال‌های صوتی حاوی انرژی در فرکانس‌های مختلف هستند. انرژی در فرکانس‌های مختلف می‌تواند در دو بعد نمایش داده شود: بعد افقی زمان است و بعد عمودی فرکانس، معمولاً با فرکانس‌های پایین‌تر در پایین و فرکانس‌های بالاتر در بالا. شدت هر فرکانس به شدت یک پیکسل تبدیل می‌شود تا سیگنال صوتی از یک ولتاژ متغیر با زمان یک‌بعدی به یک طیف‌نگار (Spectrogram) دوبعدی تبدیل شود، همان‌طور که در شکل 10-5 نشان داده شده است.



طیف‌نگار، در اینجا مربوط به گریه یک نوزاد، حاوی انبوهی از اطلاعات و ساختار است که شبکه پیچشی می‌تواند درباره آن یاد بگیرد تا مدلی بهتر از آنچه تنها با سیگنال صوتی یک‌بعدی ممکن است، تولید کند. مشاهده کلیدی این است که هر تبدیل داده‌های ورودی که ساختار را به شکلی استخراج کند که برای یک شبکه پیچشی مناسب باشد، مجاز است.

حال فرض کنید یک مجموعه داده دارید و باید شبکه‌ای پیچشی بسازید. از چه معماری‌ای باید استفاده کنید؟ اندازه مینی‌بچ باید چقدر باشد؟ به چه لایه‌هایی نیاز دارید و به چه ترتیبی؟ آیا باید از هسته‌های پیچشی 5×5 یا 3×3 استفاده کنید؟ چند دوره (Epoch) آموزش کافی است؟ اوایل، قبل از توسعه معماری‌های استاندارد، هر یک از این سؤالات باید توسط فردی که شبکه را طراحی می‌کرد پاسخ داده می‌شد. این کمی شبیه پزشکی در گذشته بود: ترکیبی از علم، تجربه و شهود. ظرافت شبکه‌های عصبی به این معنا بود که متخصصان این حوزه بسیار مورد تقاضا بودند و برای مهندسان نرم‌افزار باهوش دشوار بود که یادگیری عمیق را به مجموعه مهارت‌های خود اضافه کنند. برخی افراد تعجب می‌کردند که آیا می‌توان از نرم‌افزار برای

تعیین خودکار معماری مدل و پارامترهای آموزشی (یعنی هایپرپارامترها، که در فصل سوم معرفی شدند) استفاده کرد و به این ترتیب، یادگیری ماشینی خودکار یا AutoML متولد شد.

بیشتر پلتفرم‌های تجاری یادگیری ماشینی مبتنی بر ابر، مانند Azure Machine Learning مایکروسافت یا SageMaker Autopilot آمازون، شامل یک ابزار AutoML هستند که مدل یادگیری ماشین را برای شما ایجاد می‌کند؛ فقط باید مجموعه داده را ارائه دهید. AutoML نه تنها به شبکه‌های عصبی، بلکه به بسیاری از ابزارها شامل مدل‌های یادگیری ماشین کلاسیک نیز اعمال می‌شود. هدف اصلی AutoML یافتن بهترین نوع مدل برای مجموعه داده ارائه‌شده با حداقل تخصص کاربر است.

شخصاً دلم می‌خواهد استدلال کنم که AutoML تنها تا حدی پیش می‌رود و بهترین متخصصان یادگیری عمیق همیشه از آن بهتر عمل خواهند کرد، اما این استدلال پوچ به نظر می‌رسد. این مرا به یاد برنامه‌نویسان زبان اسمبلی قدیمی می‌اندازد که درباره غیرممکن بودن تولید کدی توسط کامپایلرها که به خوبی یا بهتر از آنچه آن‌ها می‌توانستند تولید کنند باشد، سخن می‌گفتند. امروزه فرصت‌های شغلی کمی برای برنامه‌نویسان زبان اسمبلی وجود دارد، اما ده‌ها هزار فرصت برای برنامه‌نویسان زبان‌های کامپایل‌شده موجود است (حداقل برای حالا؛ به فصل هشتم مراجعه کنید). با این حال، برخی از ما هنوز ترجیح می‌دهیم مدل‌های خودمان را بسازیم.

یکی از نتایج انقلاب یادگیری عمیق، ایجاد ابزارهای قدرتمند و متن‌باز یادگیری ماشین با نام‌هایی مانند TensorFlow و PyTorch بود. پیاده‌سازی یک شبکه عصبی سنتی و کاملاً مرتبط، تمرینی برای دانشجویان یادگیری ماشین است. این کار ساده نیست، اما چیزی است که اکثر افراد با تلاش می‌توانند به آن دست یابند.

از سوی دیگر، پیاده‌سازی صحیح یک شبکه پیچشی، به‌ویژه یکی که از انواع متعدد لایه‌ها پشتیبانی کند، اصلاً ساده نیست. جامعه هوش مصنوعی از ابتدا متعهد به توسعه ابزارهای متن‌باز برای پشتیبانی از یادگیری عمیق، از جمله شبکه‌های پیچشی، شد. بدون این ابزارها، پیشرفت در هوش مصنوعی به شدت کند می‌بود. شرکت‌های بزرگ فناوری مانند گوگل، فیسبوک (متا) و انویدیا نیز به این تلاش پیوستند و پشتیبانی مداوم آن‌ها از توسعه این ابزارها برای هوش مصنوعی حیاتی است. آنچه این ابزارها را قدرتمند می‌کند، علاوه بر انبوه کدهای تست‌شده و با کارایی بالا که در خود دارند، انعطاف‌پذیری آن‌ها است. حالا می‌دانیم که آموزش یک شبکه عصبی، چه پیچشی باشد و چه غیر آن، به دو مرحله نیاز دارد: پس‌انتشار و گرادیان کاهشی.

پس‌انتشار تنها در صورتی کار می‌کند که لایه‌های مدل از یک عملیات ریاضی خاص به نام مشتق‌گیری (Differentiation) پشتیبانی کنند. مشتق‌گیری همان چیزی است که دانشجویان ترم اول حساب دیفرانسیل و انتگرال یاد می‌گیرند. تا زمانی که ابزارها بتوانند به طور خودکار نتیجه مشتق‌گیری را تعیین کنند، به کاربران اجازه می‌دهند لایه‌های دلخواهی را پیاده‌سازی کنند. این ابزارها از مشتق‌گیری خودکار با تبدیل شبکه عصبی به یک گراف محاسباتی استفاده می‌کنند.

وسوسه‌انگیز است که چند قدم در مسیر مشتق‌گیری خودکار و گراف‌های محاسباتی برداریم، زیرا ظرافت و انعطاف‌پذیری موجود در آن، ترکیبی زیبا از ریاضیات و علوم کامپیوتر است. متأسفانه، سطح جزئیات لازم بسیار فراتر از آن چیزی است که بتوانیم در این کتاب بررسی کنیم. یک نکته کلیدی این است که دو رویکرد اصلی برای مشتق‌گیری خودکار وجود دارد: پیش‌رو (Forward) و معکوس (Reverse). مشتق‌گیری خودکار پیش‌رو برای درک و پیاده‌سازی در کد ساده‌تر است، اما برای شبکه‌های عصبی مناسب نیست.

این تا حدی تأسف‌بار است، زیرا مشتق‌گیری خودکار پیش‌رو بهتر است با استفاده از اعداد دوگانه (Dual Numbers) پیاده‌سازی شود، نوعی عدد ناشناخته که توسط ریاضیدان انگلیسی ویلیام کلیفورد (William Clifford) در سال 1873 اختراع (یا کشف؟) شد؛ این اعداد تا زمان ظهور کامپیوترها تا حد زیادی فراموش شده بودند، تا اینکه ناگهان مفید شدند. مشتق‌گیری خودکار معکوس برای شبکه‌های عصبی مناسب‌تر است اما از اعداد دوگانه استفاده نمی‌کند.

این فصل چالش‌برانگیز بود. ما عمیق‌تر از فصل‌های قبلی به جزئیات پرداختیم و در فصل‌های بعدی نیز چنین خواهیم کرد. قطعاً یک خلاصه لازم است.

شبکه‌های عصبی پیچشی:

- با ساختار موجود در ورودی‌های خود رشد می‌کنند، که کاملاً برخلاف مدل‌های یادگیری ماشین کلاسیک است.

- با تجزیه ورودی‌های خود به بخش‌ها و گروه‌هایی از بخش‌ها بازنمایی‌های جدیدی از آنها یاد می‌گیرند.

- از انواع مختلف لایه‌ها به روش‌های گوناگونِ ترکیب‌شده استفاده می‌کنند.

- می‌توانند ورودی‌ها را طبقه‌بندی کنند، ورودی‌ها را مکان‌یابی کنند، یا به هر پیکسل در ورودی‌های خود یک برچسب کلاسی اختصاص دهند.

- همچنان از طریق پس‌انتشار و گرادیان کاهشی، درست مانند شبکه‌های عصبی سنتی، آموزش می‌بینند.

- ایجاد ابزارهای قدرتمند و متن‌باز را ترغیب کردند که یادگیری عمیق را دموکراتیزه نمود.

شبکه‌های عصبی پیچشی در سنت مدل‌های یادگیری ماشین کلاسیک عمل می‌کنند: آن‌ها یک ورودی را می‌گیرند و به نوعی به آن یک برچسب کلاسی اختصاص می‌دهند. شبکه به عنوان یک تابع ریاضی عمل می‌کند، ورودی را می‌پذیرد و خروجی تولید می‌کند. فصل بعدی ما را با شبکه‌های عصبی آشنا می‌کند که بدون ورودی، خروجی تولید می‌کنند.

شما در حال سفر به بُعدی دیگر هستید، بُعدی نه تنها از دید و صدا، بلکه از ذهن، سفری به سرزمینی شگفت‌انگیز که مرزهای آن همان مرزهای تخیل است - ایستگاه بعدی، هوش مصنوعی مولد.

اصطلاحات کلیدی: مشتق‌گیری خودکار (Automatic Differentiation)، یادگیری ماشینی خودکار (AutoML)، کادر محدود کننده (Bounding Box)، گراف محاسباتی (Computational Graph)، پیچش (Convolution)، لایهٔ پیچشی (Convolutional Layer)، شبکه عصبی پیچشی (Convolutional Neural Network)، لایه متراکم (Dense Layer)، میدان دریافت مؤثر (Effective Receptive Field)، جاسازی (Embedding)، یادگیری سرتاسری (End-to-End Learning)، فیلتر (Filter)، هسته (Kernel)، لایهٔ ادغام (Pooling Layer)، تقسیم‌بندی معنایی (Semantic Segmentation)

فصل ششم: هوش مصنوعی مولد (هوش مصنوعی خلاق می‌شود)

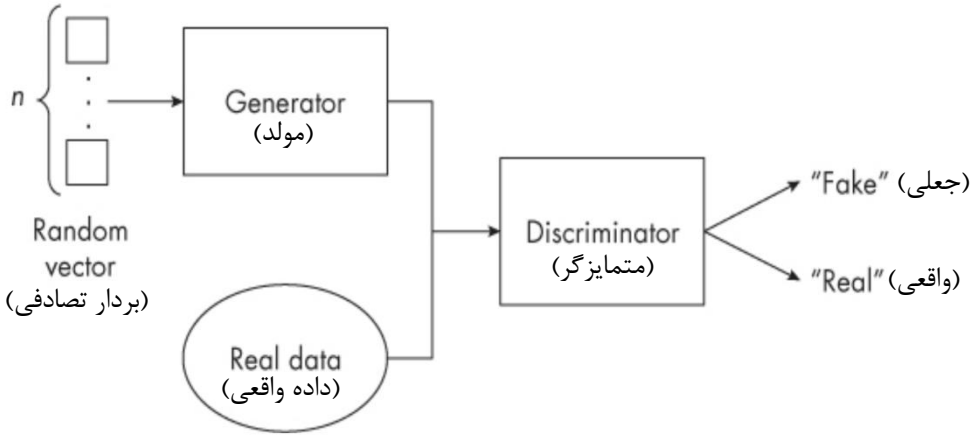
هوش مصنوعی مولد اصطلاحی کلی برای مدل‌هایی است که خروجی جدیدی تولید می‌کنند، چه به‌صورت مستقل (تصادفی) و چه بر اساس درخواست ارائه‌شده توسط کاربر. این مدل‌ها نه برچسب، بلکه متن، تصویر یا حتی ویدئو تولید می‌کنند. در پشت صحنه، مدل‌های مولد شبکه‌های عصبی هستند که از همان اجزای اساسی ساخته شده‌اند.

ما بر سه نوع مدل هوش مصنوعی مولد تمرکز خواهیم کرد: شبکه‌های مولد تخاصمی (Generative Adversarial Networks)، مدل‌های انتشار (Diffusion Models) و مدل‌های زبانی بزرگ (Large Language Models). این فصل دو مورد اول را پوشش می‌دهد. مدل‌های زبانی بزرگ اخیراً دنیای هوش مصنوعی را زیر و رو کرده‌اند. آن‌ها موضوع فصل هفتم هستند.

شبکه‌های مولد تخاصمی (GANها) از دو شبکه عصبی جداگانه تشکیل شده‌اند که با هم آموزش می‌بینند. شبکه اول مولد است. وظیفه آن یادگیری چگونگی ایجاد ورودی‌های جعلی برای متمایزگر (Discriminator) است. وظیفه متمایزگر یادگیری تمایز بین ورودی‌های جعلی و واقعی است. هدف از آموزش همزمان این دو شبکه این است که مولد در فریب دادن متمایزگر بهتر شود، در حالی که متمایزگر تمام تلاش خود را می‌کند تا واقعی را از جعلی تشخیص دهد.

در ابتدا، مولد بسیار ضعیف است. خروجی آن نویز است و متمایزگر به راحتی می‌تواند بین واقعی و جعلی تمایز قائل شود. با این حال، مولد با گذشت زمان بهبود می‌یابد و کار متمایزگر را به طور فزاینده‌ای دشوارتر می‌کند؛ این امر به نوبه خود متمایزگر را وادار می‌کند تا به عنوان یک شناساگر بهتر واقعی در مقابل جعلی عمل کند. هنگامی که آموزش «کامل» می‌شود، معمولاً متمایزگر کنار گذاشته می‌شود و مولد آموزش‌دیده برای تولید خروجی‌های جدید استفاده می‌شود که به صورت تصادفی از فضای آموخته‌شده داده‌های آموزشی، نمونه‌برداری می‌شوند. من مشخص نکرده‌ام که داده‌های آموزشی چیستند، زیرا در حال حاضر تنها چیزی که نیاز است بدانیم این است که یک GAN از دو شبکه رقابتی (متخاصم) ساخته شده است. این برای اکثر کاربردها، همان مولدی است که در نهایت می‌خواهیم.

از نظر ساختاری، می‌توانیم یک GAN را مانند بلوک‌هایی در شکل 1-6 تصور کنیم. (بخش بردار تصادفی را در زمان مناسب توضیح خواهم داد.) از نظر مفهومی، می‌بینیم که متمایزگر دو نوع ورودی را می‌پذیرد: داده‌های واقعی و خروجی مولد. خروجی متمایزگر یک برچسب است: «واقعی» یا «جعلی». آموزش استاندارد شبکه عصبی با استفاده از پس‌انتشار و گرادیان کاهشی، مولد و متمایزگر را با هم آموزش می‌دهد، اما نه به طور همزمان.



برای مثال، آموزش با یک مینی‌بچ از داده‌های واقعی - زیرمجموعه کوچکی از داده‌های آموزشی واقعی در دسترس - شامل مراحل زیر است:

1. از مولد به همان شکلی که در حال حاضر است استفاده کنید تا یک مینی‌بچ از داده‌های جعلی تولید شود.

2. یک مینی‌بچ از داده‌های واقعی را از مجموعه آموزشی بردارید.

3. وزن‌های متمایزگر را آزاد کنید تا گرادیان کاهشی بتواند آن‌ها را به‌روزرسانی کند. («آزاد کردن» به فرآیندی اشاره دارد که در آن وزن‌های یک مدل دوباره قابل تنظیم یا قابل آموزش می‌شوند. به طور خاص، وقتی «وزن‌های متمایزگر را آزاد می‌کنید»، به وزن‌ها (پارامترها)ی شبکه متمایزگر اجازه می‌دهید تا در طول فرآیند آموزش با استفاده از تکنیک‌هایی مانند گرادیان کاهشی به‌روزرسانی شوند.)

4. نمونه‌های جعلی و واقعی را به ترتیب با برچسب‌های 0 و 1 از طریق متمایزگر عبور دهید.

5. از پس‌انتشار استفاده کنید تا یک گام کاهش گرادیان برای به‌روزرسانی وزن‌های متمایزگر بردارید.

6. متمایزگر را ثابت کنید تا مولد بتواند بدون تغییر متمایزگر به‌روزرسانی شود.

7. یک مینی‌بچ از ورودی‌های مولد (بردار تصادفی در شکل 1-6) ایجاد کنید.

8. ورودی‌های مولد را از طریق مدل ترکیبی عبور دهید تا وزن‌های مولد به‌روزرسانی شوند. هر یک از ورودی‌های مولد را به عنوان «واقعی» علامت‌گذاری کنید.

9. از مرحله 1 تکرار کنید تا مدل کامل آموزش ببیند.

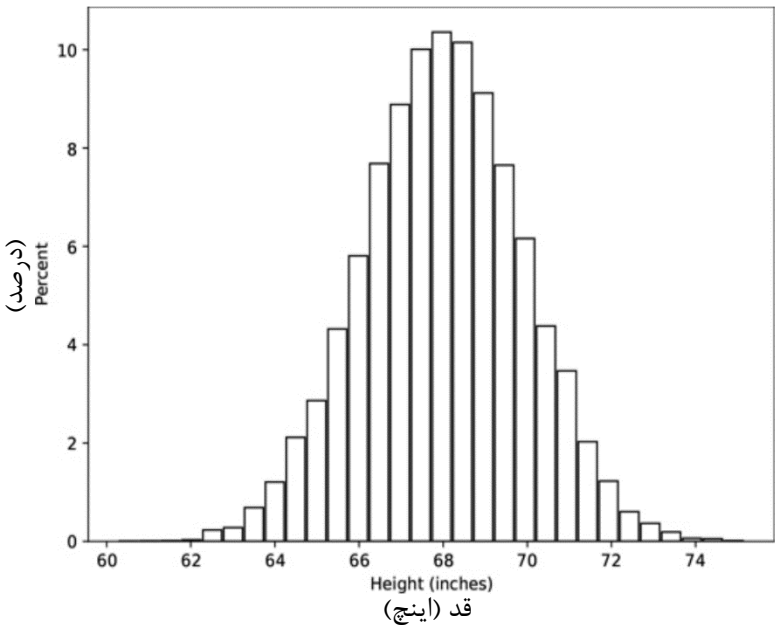
الگوریتم ابتدا وزن‌های متمایزگر را با استفاده از مولد به همان شکلی که در حال حاضر است به‌روزرسانی می‌کند (مرحله 5)، سپس آن‌ها را ثابت می‌کند (مرحله 6) تا وزن‌های مولد بتوانند بدون تغییر متمایزگر به‌روزرسانی شوند. این رویکرد ضروری است زیرا ما می‌خواهیم خروجی متمایزگر - برچسب‌های «واقعی» یا «جعلی» - بخش مولد را به‌روزرسانی کند. توجه کنید که به‌روزرسانی مولد همه تصاویر جعلی را به عنوان واقعی علامت‌گذاری می‌کند. این کار مولد را بر اساس اینکه ورودی‌های جعلی تا چه حد برای متمایزگر واقعی به نظر می‌رسند، امتیازدهی می‌کند.

بیایید بردار تصادفی که به عنوان ورودی به مولد استفاده می‌شود را بررسی کنیم. هدف یک GAN یادگیری نمایشی از مجموعه آموزشی است که می‌توانیم آن را به عنوان یک مولد داده در نظر بگیریم، مانند فرآیندی که داده‌های آموزشی واقعی را تولید کرده است. با این حال، در این مورد، مولد داده می‌تواند به عنوان تابعی دیده شود که مجموعه‌ای تصادفی از اعداد، یعنی بردار تصادفی، را می‌گیرد و آن‌ها را به خروجی‌ای تبدیل می‌کند که به طور معقولی ممکن است از مجموعه آموزشی آمده باشد. به عبارت دیگر، مولد مانند یک دستگاه افزایش داده عمل می‌کند. ورودی تصادفی به مولد به مثابه مثالی از مجموعه آموزشی می‌شود. در واقع، مولد به عنوان نماینده‌ای برای فرآیند واقعی تولید داده که در ابتدا مجموعه آموزشی واقعی را ایجاد کرده است، عمل می‌کند.

بردار تصادفی اعداد از یک توزیع احتمال نمونه‌برداری می‌شود. نمونه‌برداری از یک توزیع احتمال مشابه پرتاب دو تاس و پرسیدن این است که احتمال اینکه مجموع آن‌ها هفت باشد در مقابل دو چقدر است. احتمال اینکه مجموع هفت باشد بیشتر است، زیرا

راه‌های بیشتری برای جمع کردن دو عدد و رسیدن به هفت وجود دارد. تنها یک راه برای رسیدن به دو وجود دارد. نمونه‌برداری از یک توزیع نرمال مشابه است. رایج‌ترین نمونه بازگشتی، مقدار میانگین توزیع است. مقادیر در دو طرف میانگین، هرچه از میانگین دورتر شوند، احتمال کمتری دارند، اگرچه همچنان ممکن است رخ دهند.

برای مثال، شکل 2-6 یک نمودار میله‌ای از توزیع قد انسان‌ها به اینچ را نشان می‌دهد. مجموعه داده اصلی شامل قد 25000 نفر بود که سپس در 30 دسته از شکل جای داده شدند. هرچه میله بلندتر باشد، افراد بیشتری در آن دسته قرار گرفته‌اند.



به شکل هیستوگرام توجه کنید که شبیه یک زنگوله است - از این رو نام تا حدی قدیمی آن، منحنی زنگوله‌ای است. نام مدرن آن، توزیع نرمال، به این دلیل است که این توزیع به قدری در طبیعت رایج است که معمولاً با آن مواجه می‌شویم، به‌ویژه برای داده‌هایی که توسط یک فرآیند فیزیکی تولید شده‌اند. از این توزیع می‌بینیم قد فردی که به‌صورت تصادفی انتخاب‌شده معمولاً حدود 68 اینچ است: بیش از 10 درصد از جمعیت نمونه‌برداری‌شده در این دسته قرار گرفته‌اند.

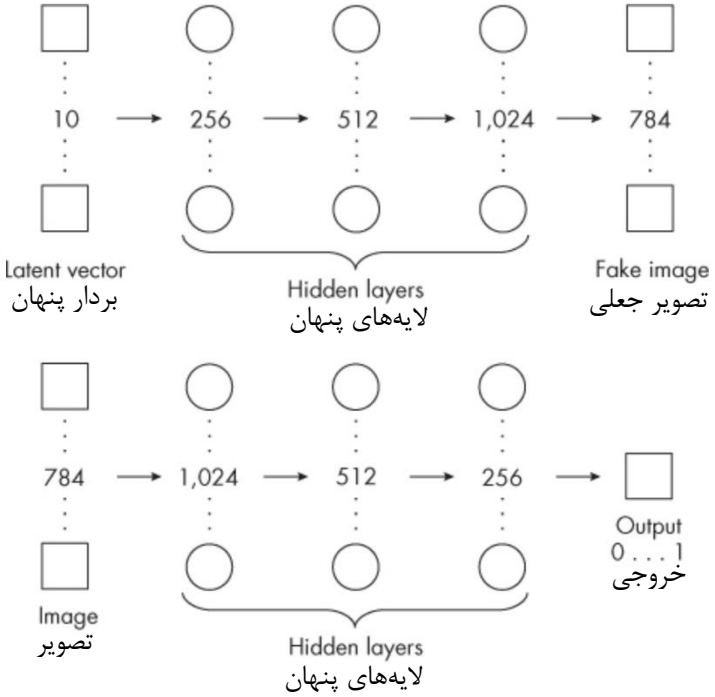
بردار تصادفی که توسط GAN استفاده می‌شود، که به عنوان بردار نویز نیز شناخته می‌شود، به همین شیوه عمل می‌کند. میانگین در این مورد صفر است؛ با بیشتر نمونه‌ها در محدوده 3- تا 3 (دلیل اینکه گفته می‌شود میانگین صفر است و بیشتر نمونه‌ها در محدوده 3- تا 3 قرار دارند، این است که بردار نویز معمولاً از یک توزیع احتمال نرمال (نرمال گاوسی) با میانگین صفر و انحراف معیار مشخص (اغلب 1) نمونه‌برداری می‌شود. این محدوده (3- تا 3) حدود 99.7 درصد از داده‌ها را در یک توزیع نرمال استاندارد پوشش می‌دهد، که با قاعده سه سیگما (68-95-99.7) سازگار است). همچنین، هر یک از n عنصر در بردار این محدوده را دنبال می‌کند، به این معنی که خود بردار نمونه‌ای از یک فضای n بعدی است، نه فضای یک‌بعدی شکل 2-6.

نیاز به مجموعه داده‌های برچسب‌دار یکی از مشکلات یادگیری ماشین است. GANها چنین محدودیتی ندارند. ما اهمیتی نمی‌دهیم که کلاس یک نمونه آموزشی چیست، فقط این مهم است که نمونه از داده‌های واقعی باشد، صرف‌نظر از برچسب کلاس. البته، همچنان نیاز است که مجموعه آموزشی نوع داده‌ای را که می‌خواهیم تولید کنیم منعکس کند، اما مجموعه آموزشی نیازی به برچسب‌گذاری ندارد.

بیایید یک شبکه متخاصم مولد (GAN) با استفاده از دوست قدیمی‌مان، مجموعه داده ارقام MNIST، بسازیم. مولد یاد می‌گیرد که یک مجموعه تصادفی از 10 عدد (یعنی n برابر 10) را به یک تصویر رقم تبدیل کند. پس از آموزش، می‌توانیم به مولد، هر مجموعه‌ای از 10 مقدار نزدیک به صفر بدهیم و مولد یک تصویر جدید از رقم را به عنوان خروجی تولید خواهد کرد، که بدین ترتیب فرآیندی را که مجموعه داده MNIST را ایجاد کرده است تقلید می‌کند: افرادی که ارقام را با دست روی کاغذ می‌نویسند. یک مولد آموزش‌دیده GAN عرضه بی‌نهایت از خروجی هدف را تولید می‌کند. ما از یک GAN ساده مبتنی بر شبکه‌های عصبی سنتی استفاده خواهیم کرد تا مولدی برای عرضه بی‌نهایت تصاویر ارقام به سبک MNIST ایجاد کنیم. ابتدا، مجموعه آموزشی موجود MNIST را به گونه‌ای باز خواهیم کرد که هر نمونه یک بردار 784 بعدی باشد، همان‌طور که در فصل 5 انجام دادیم. این به ما داده‌های واقعی را می‌دهد. برای ایجاد داده‌های جعلی، به بردارهای تصادفی 10 عنصری نیاز داریم که با نمونه‌برداری 10 نمونه از یک توزیع نرمال با میانگین صفر ساخته می‌شوند.

بخش مولد مدل یک بردار نویز 10 عنصری را به عنوان ورودی می‌پذیرد و یک بردار خروجی 784 عنصری را تولید می‌کند که نمایانگر تصویر سنتز شدهٔ رقم است. به یاد بیاورید که این 784 عدد می‌توانند به یک تصویر 28×28 پیکسلی بازآرایی شوند. مدل مولد دارای سه لایه پنهان با 256، 512 و 1,024 گره و یک لایه خروجی با 784 گره برای تولید تصویر است. گره‌های لایه پنهان از نسخه اصلاح‌شده واحد یک سو شده خطی (ReLU) به نام Leaky ReLU استفاده می‌کنند. در فعال‌سازهای Leaky ReLU اگر ورودی مثبت باشد خروجی همان ورودی است، اما اگر ورودی منفی باشد، خروجی یک مقدار مثبت کوچک ضرب‌شده در ورودی منفی است. لایه خروجی از تابع فعال‌سازی تانژانت هیپربولیک استفاده می‌کند، به این معنی که هر یک از 784 عنصر خروجی در محدوده 1- تا 1+ خواهد بود. این قابل قبول است. می‌توانیم مقادیر را هنگام نوشتن تصویر روی دیسک به محدوده 0 تا 255 مقیاس‌بندی کنیم. مولد باید بین بردار نویز تصادفی ورودی و یک تصویر خروجی نگاشت کند. متمایزگر باید یک تصویر را به عنوان ورودی بپذیرد، که به معنای یک بردار 784 بعدی است. متمایزگر دارای سه لایه پنهان مانند مولد است، اما به صورت معکوس: 1024 گره، سپس 512 گره، و پس از آن 256 گره. لایه خروجی متمایزگر یک گره با تابع فعال‌سازی سیگموئید دارد. سیگموئید مقادیری از 0 تا 1 تولید می‌کند که می‌توان آن را به عنوان اعتقاد متمایزگر به واقعی بودن ورودی (خروجی نزدیک به 1) یا جعلی بودن آن (خروجی نزدیک به 0) تفسیر کرد. توجه کنید که شبکه از چیزی بیشتر از لایه‌های کاملاً متصل استاندارد استفاده نمی‌کند. GANهای پیشرفته از لایه‌های پیچشی استفاده می‌کنند، اما بررسی جزئیات آن شبکه‌ها خارج از محدوده ما است.

شکل 3-6 مولد (بالا) و متمایزگر (پایین) را نشان می‌دهد. تقارن بین این دو در تعداد گره‌های لایه‌های پنهان مشهود است، اگرچه توجه کنید که ترتیب در متمایزگر معکوس است.



مولد یک بردار تصادفی 10 عنصری را به عنوان ورودی می‌پذیرد و یک بردار خروجی تصویر جعلی 784 عنصری تولید می‌کند. متمایزگر یک بردار تصویر، واقعی یا جعلی، را می‌پذیرد و یک پیش‌بینی، عددی از 0 تا 1، را خروجی می‌دهد. تصاویر جعلی باید مقادیری نزدیک به 0 و تصاویر واقعی مقادیری نزدیک به 1 تولید کنند. اگر مولد به‌خوبی آموزش دیده باشد، متمایزگر بیشتر اوقات فریب می‌خورد، به این معنا که خروجی متمایزگر برای همه ورودی‌ها نزدیک به 0.5 خواهد بود.

کل شبکه برای 200 دوره (Epoch) با 468 مینی‌بچ در هر دوره، به مجموع 93600 گام کاهش گرادیان، آموزش داده می‌شود. می‌توانیم نمونه‌هایی از مولد را پس از هر دوره نمایش دهیم تا شبکه را در حین یادگیری مشاهده کنیم. شکل 4-6 نمونه‌هایی را پس از دوره‌های 1، 60 و 200، از چپ به راست، نشان می‌دهد.

همان‌طور که انتظار می‌رود، مولد پس از یک بار گذر از داده‌های آموزشی عملکرد ضعیفی دارد، اما شاید به اندازه‌ای که فکر می‌کردیم ضعیف نباشد. بیشتر تصاویر تولیدشده شبیه به یک به نظر می‌رسند؛ شکل‌های دیگر اعداد، مانند صفر و دو، نیز حضور دارند، اگرچه مملؤ از نویز هستند.

پس از 60 دوره، مولد طیف کاملی از اعداد را تولید می‌کند. برخی کاملاً دقیق هستند، در حالی که برخی دیگر هنوز مبهم‌اند یا فقط به‌صورت ناقص رسم شده‌اند. پس از 200 دوره، بیشتر اعداد متمایز و با وضوح بالا هستند. مولد اکنون آموزش‌دیده است و برای تولید تصاویر اعداد بر اساس تقاضا آماده است.

مولد اعداد 10000 تصویر جدید برایمان ایجاد می‌کند، اما اگر بخواهیم همه این اعداد چهار باشند چه؟ یک بردار ورودی تصادفی یک رقم تصادفی تولید می‌کند، اما ما نمی‌توانیم انتخاب کنیم کدام یک باشد. اگر بردارهای ورودی را به‌صورت تصادفی انتخاب کنیم، می‌توانیم به این باور برسیم که ترکیب اعداد خروجی نیز به‌طور مشابه تصادفی خواهد بود. این فرضیه را با استفاده از مولد آموزش‌دیده برای ایجاد 1000 تصویر رقم آزمایش کردم. سپس این تصاویر را به یک شبکه پیچشی که روی مجموعه داده MNIST آموزش دیده بود، منتقل کردم. دقت مجموعه آزمایشی این شبکه پیچشی بالای 99 درصد است، که به ما اطمینان می‌دهد پیش‌بینی‌های آن درست است، به شرطی که ورودی یک تصویرِ رقم باشد. مولد GAN تصاویر واقعی از اعداد تولید می‌کند، بنابراین پایه محکمی داریم.

با فرض اینکه مولد همان‌طور که انتظار می‌رود عمل می‌کند، درصد هر رقم باید به‌طور ساده یکسان باشد. 10 رقم ممکن وجود دارد، بنابراین انتظار می‌رود هر یک حدود 10 درصد از زمان ظاهر شود. اما این اتفاق نیفتاد. جدول 1-6 توزیع واقعی وقوع هر رقم را نشان می‌دهد.

مولد در درجهٔ نخست به اعداد یک تمایل دارد، سپس هفت‌ها، نُه‌ها و صفرها؛ اعداد هشت و دو کمترین احتمال خروجی را دارند. بنابراین، نه تنها GAN به ما اجازه انتخاب نوع رقم مورد نظر را نمی‌دهد، بلکه دارای علایق مشخصی است. تصویر سمت چپ در شکل 4-6 را بررسی کنید که نمونه‌های دوره 1 را نشان می‌دهد. بیشتر این اعداد یک هستند، بنابراین تمایل GAN به اعداد یک از ابتدای آموزش آشکار بود. GAN یاد گرفت، اما غلبه اعداد یک نشانه مشکلی است که گاهی آموزش GAN را آزار می‌دهد: یعنی فروپاشی مد (Mode Collapse)، جایی که مولد در اوایل یاد می‌گیرد چگونه یک مثال یا مجموعه مثال‌های بسیار خوب بسازد که متمایزگر را فریب دهد و در دام تولید تنها آن خروجی و نه تنوع مورد نظر تصاویر بیفتد.

رقم	درصد
0	10.3
1	21.4
2	4.4
3	7.6
4	9.5
5	6.0
6	9.1
7	14.4
8	4.4
9	12.9

نیازی نیست که به یک GAN ناپایدار و غیرقابل‌کنترل قناعت کنیم. در عوض، می‌توانیم شبکه را در طول آموزش با ارائه نشانه‌ای از نوع رقمی که می‌خواهیم مولد ایجاد کند، شرطبندی کنیم. GANهایی که این رویکرد را به کار می‌برند، به عنوان GANهای شرطی شناخته می‌شوند. برخلاف GANهای بدون شرط، آن‌ها به مجموعه‌های آموزشی با برچسب نیاز دارند. در یک GAN شرطی، ورودی مولد همچنان یک بردار نویز تصادفی است، اما به آن یک بردار دیگر که کلاس خروجی مورد نظر را مشخص می‌کند، متصل می‌شود. برای مثال، مجموعه داده MNIST دارای 10 کلاس، یعنی ارقام 0 تا 9 است، بنابراین بردار

شرطی 10 عنصر دارد. اگر کلاس مورد نظر رقم 3 باشد، بردار شرطی برای همه ارقام صفر است جز عنصر 3 که به یک تنظیم می‌شود. این روش نمایش اطلاعات کلاس به نام رمزگذاری وان-هات (One-Hot Encoding) شناخته می‌شود، زیرا همه عناصر بردار صفر هستند جز عنصری که با برچسب کلاس مورد نظر مطابقت دارد و آن یک است.

متمایزگر نیز به برچسب کلاس نیاز دارد. اگر ورودی متمایزگر یک تصویر باشد، چگونه برچسب کلاس را شامل کنیم؟ یک روش این است که مفهوم رمزگذاری وان-هات را به تصاویر گسترش دهیم. می‌دانیم که یک تصویر رنگی با سه ماتریس، یکی برای کانال قرمز، یکی برای کانال سبز و یکی برای کانال آبی، نمایش داده می‌شود. تصاویر خاکستری تنها یک کانال دارند. می‌توانیم برچسب کلاس را به صورت مجموعه‌ای از کانال‌های ورودی اضافی شامل کنیم که همه کانال‌ها صفر هستند جز کانالی که با برچسب کلاس مطابقت دارد و آن یک است.

شامل کردن برچسب کلاس هنگام تولید و تفکیک بین ورودی‌های واقعی و جعلی، هر بخش از کل شبکه را مجبور می‌کند که یاد بگیرد چگونه خروجی و ورودی خاص کلاس را تولید و تفسیر کند. اگر برچسب کلاس 4 باشد و رقمی که توسط مولد تولید می‌شود بیشتر شبیه صفر باشد، متمایزگر متوجه عدم تطابق کلاس خواهد شد، زیرا از صفرهای واقعی در مجموعه آموزشی برچسب‌دار آگاه است.

فایده یک GAN شرطی زمانی مشخص می‌شود که از مولد آموزش‌دیده استفاده می‌کنیم. کاربر کلاس مورد نظر را به صورت یک بردار وان-هات، همراه با بردار نویز تصادفی که توسط یک GAN بدون شرط استفاده می‌شود، ارائه می‌دهد. سپس مولد بر اساس بردار نویز نمونه‌ای را خروجی می‌دهد، اما با شرط برچسب کلاس مورد نظر. می‌توانیم یک GAN شرطی را به عنوان مجموعه‌ای از GANهای بدون شرط تصور کنیم که هر کدام روی یک کلاس خاص از تصاویر آموزش دیده‌اند.

من یک GAN شرطی را روی مجموعه داده MNIST آموزش دادم. برای این مثال، GAN از لایه‌های پیچشی به جای لایه‌های کاملاً متصل که قبلاً در این فصل استفاده شده بود، استفاده کرد. سپس از مولد کاملاً آموزش‌دیده خواستم که 10 نمونه از هر رقم را تولید کند، همان‌طور که در شکل 5-6 نشان داده شده است.

GANهای شرطی به ما اجازه می‌دهند کلاس خروجی مورد نظر را انتخاب کنیم، چیزی که GANهای بدون شرط نمی‌توانند انجام دهند، اما اگر بخواهیم ویژگی‌های خاص تصویر خروجی را تنظیم کنیم چه؟ برای این کار به یک GAN قابل‌کنترل نیاز داریم.

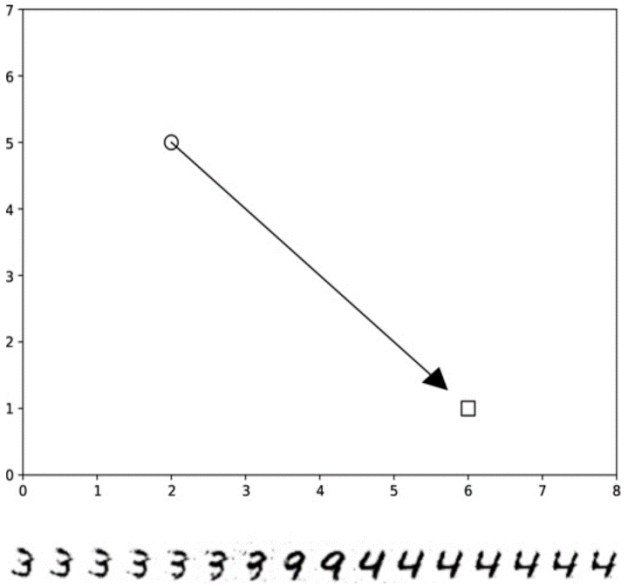
GANهای غیرقابل‌کنترل تصاویر را به صورت تصادفی و بدون توجه به برچسب کلاس تولید می‌کنند. GANهای شرطی تولید تصاویر خاص کلاس را معرفی می‌کنند، که در صورتی که بخواهیم از یک GAN برای تولید تصاویر مصنوعی برای آموزش

مدل‌های دیگر استفاده کنیم، مفید است، شاید برای جبران یک کلاس که نمونه‌های نسبتاً کمی از آن داریم. از سوی دیگر، GANهای قابل‌کنترل به ما اجازه می‌دهند ظاهر ویژگی‌های خاص در تصاویر تولیدشده را کنترل کنیم. وقتی شبکه مولد یاد می‌گیرد، یک فضای انتزاعی را یاد می‌گیرد که می‌تواند به تصاویر خروجی نگاشت شود. بردار نویز تصادفی، یک نقطه در این فضاست که تعداد ابعاد آن برابر با تعداد عناصر بردار نویز است. هر نقطه به یک تصویر تبدیل می‌شود. اگر همان نقطه، یعنی همان بردار نویز، را به مولد بدهید، همان تصویر خروجی خواهد شد.

حرکت در فضای انتزاعی که توسط بردار نویز نشان داده می‌شود، تصاویر خروجی را یکی پس از دیگری تولید می‌کند. آیا ممکن است جهت‌هایی در فضای نویز انتزاعی وجود داشته باشند که برای ویژگی‌های تصویر خروجی معنا داشته باشند؟ اینجا، ویژگی به چیزی در تصویر اشاره دارد. برای مثال، اگر مولد تصاویر چهره‌های انسان را تولید کند، یک ویژگی ممکن است این باشد که آیا چهره عینک دارد، ریش دارد یا موهای قرمز دارد.

GANهای قابل‌کنترل جهت‌های معنادار را در فضای نویز کشف می‌کنند. حرکت در امتداد یکی از این جهت‌ها، ویژگی مربوط به آن جهت را تغییر می‌دهد. البته واقعیت پیچیده‌تر است، زیرا یک جهت واحد ممکن است بسته به ابعاد فضای نویز و داده‌هایی که مولد آموخته است، چندین ویژگی را تحت تأثیر قرار دهد. به طور کلی، بردارهای نویز کوچک‌تر احتمالاً در هم‌تنیده‌تر هستند، به این معنا که ابعاد تک بردار نویز چندین ویژگی خروجی را تحت تأثیر قرار می‌دهند و تشخیص جهت‌های جالب را دشوار می‌کنند. برخی تکنیک‌های آموزشی و بردارهای نویز بزرگ‌تر، شاید با 100 عنصر به جای 10 عنصری که قبلاً استفاده کردیم، شانس مدل را برای تخصیص تنظیمات جالب ویژگی به یک جهت واحد بهبود می‌بخشند. به طور ایده‌آل، تنظیم معناداری برای یک عنصر تک بردار نویز وجود خواهد داشت.

بیایید با یک مثال دوبعدی این ایده را روشن کنیم. یادگیری یک مولد با استفاده از بردار نویز دوبعدی ممکن است دشوار باشد، اما مفهوم برای همه ابعاد صدق می‌کند و در دو بعد به راحتی قابل نمایش است. شکل 6-6 آنچه نیاز داریم را دارد.



بخش بالایی شکل یک فضای نویز دوبعدی برای یک مولد با دو ورودی، مختصات x و مختصات y، را نشان می‌دهد. بنابراین، هر نقطه در شکل نمایانگر تصویری است که توسط GAN تولید شده است. تصویر اول از نقطه (2,5) (دایره) تولید می‌شود. تصویر دوم از نقطه (6,1) (مربع) پدید می‌آید. فلش جهتی را در فضای نویز نشان می‌دهد که به نوعی آموخته‌ایم که یک ویژگی

در تصویر خروجی را کنترل می‌کند. اگر GAN چهره‌ها را تولید کند، ممکن است این فلش در جهتی باشد که رنگ موهای شخص را تحت تأثیر قرار دهد. حرکت از نقطه (2,5) به نقطه (6,1) بیشتر بخش‌های تصویر خروجی را حفظ می‌کند، اما رنگ مو را از، مثلاً سیاه در (2,5) به قرمز در (6,1) تغییر می‌دهد. نقاطی که در امتداد فلش قرار دارند، رنگ مو را به صورت میانی بین سیاه و قرمز نشان می‌دهند.

بخش پایینی شکل 6-6 نشان‌دهنده درون‌یابی (Interpolation) در امتداد بعد سوم GAN است که برای تولید تصاویر اعداد آموزش داده‌ایم. از چپ به راست، یک «سه» به طور موقت به یک «نُه» و سپس به یک چهار تبدیل می‌شود، زیرا عنصر سوم بردار نویز 10 عنصری تغییر می‌کند در حالی که همه عناصر دیگر در مقادیر تصادفی اولیه خود ثابت نگه داشته شده‌اند. بردار نویز دارای ابعاد نسبتاً پایینی است، که نشان می‌دهد بعید است هر بعد به تنهایی با یک ویژگی تک رقم مرتبط باشد، که همین دلیل تغییر کل تصویر از یک سه اولیه به یک نُه و سپس به یک چهار است.

GANهای پیشرفته می‌توانند تصاویر واقعی اما جعلی از چهره‌های انسان تولید کنند. نسخه‌های قابل‌کنترل جهت‌هایی را یاد می‌گیرند که به ویژگی‌های خاص صورت مرتبط هستند. برای مثال، شکل 7-6 را در نظر بگیرید که دو چهره جعلی تولیدشده را در سمت چپ و چهره‌های تنظیم‌شده را در سمت راست نشان می‌دهد (از یوجون شِن و همکاران، «تفسیر فضای نهان GANها برای ویرایش معنایی صورت»، 2019). تنظیمات مربوط به حرکت در فضای نویز از موقعیت تصویر اصلی در امتداد جهت‌های آموخته‌شده‌ای است که نمایانگر سن، عینک، جنسیت و حالت هستند.

قدرت GANهای قابل‌کنترل واقعاً شگفت‌انگیز است و این که مولد جهت‌های معناداری را از طریق فضای نویز یاد می‌گیرد، بسیار قابل‌توجه است. با این حال، GANها تنها راه ایجاد تصاویر واقعی و قابل‌کنترل نیستند.

مدل‌های انتشار (Diffusion Models) تصاویر واقعی تولید می‌کنند (تصاویری که بر اساس دستورات متنی تعریف‌شده توسط کاربر شرط‌بندی شده‌اند).

شبکه‌های مولد تخاصمی به رقابت بین مولد و متمایزگر وابسته هستند تا یاد بگیرند خروجی‌های جعلی مشابه داده‌های آموزشی ایجاد کنند. مدل‌های انتشار، رویکردی بدون رقابت برای همان هدف ارائه می‌دهند.

به طور خلاصه، آموزش یک مدل انتشار شامل آموزش آن برای پیش‌بینی نویز اضافه‌شده به یک تصویر آموزشی است. استنتاج در یک مدل انتشار برعکس است، یعنی تبدیل نویز به تصویر. عالی! اما «نویز» در مورد تصاویر به چه معنا است؟

نویز به معنای تصادفی بودن است، چیزی بدون ساختار. اگر به نویز رادیو یا خش‌خش در سیگنال صوتی فکر می‌کنید، در مسیر درستی هستید. برای یک تصویر دیجیتال، نویز به معنای مقادیر تصادفی اضافه‌شده به پیکسل‌هاست. برای مثال، اگر مقدار پیکسل باید 127 باشد، نویز مقداری را کم یا زیاد می‌کند، به طوری که 127 به 124 یا 129 تبدیل می‌شود. نویز تصادفی اضافه‌شده به تصویر اغلب شبیه برف به نظر می‌رسد. مدل‌های انتشار یاد می‌گیرند چگونه مقدار نویز با توزیع نرمال که به یک تصویر آموزشی اضافه شده است را پیش‌بینی کنند.

قبل از آموزش شبکه، باید چندین چیز را مهیا کنیم. ابتدا، به یک مجموعه داده آموزشی نیاز داریم. مدل‌های انتشار مانند همه شبکه‌های عصبی از داده‌ها یاد می‌گیرند. مانند GANها، برچسب‌ها لازم نیستند مگر اینکه بخواهیم در مورد آنچه مدلِ آموزش‌دیده تولید می‌کند، نظری داشته باشیم.

هنگامی که داده‌های آموزشی را در اختیار داریم، به یک معماری شبکه عصبی نیاز داریم. مدل‌های انتشار در اینجا سخت‌گیر نیستند، اما معماری انتخاب‌شده باید یک تصویر را به عنوان ورودی بپذیرد و یک تصویر با همان اندازه را به عنوان خروجی تولید کند. معماری U-Net که به طور مختصر در فصل پنجم ذکر شده، گزینه‌ای رایج است.

حالا داده‌ها و یک معماری داریم؛ در مرحله بعد، به روشی برای آموزش شبکه نیازمندیم. اما شبکه باید چه چیزی را یاد بگیرد؟ همان‌طور که مشخص است، مجبور کردن شبکه به یادگیری نویز اضافه‌شده به یک تصویر، تنها چیزی است که لازم است. ریاضیات پشت این مسئله چندان ساده نیست. این موضوع شامل نظریه احتمال می‌شود، اما در عمل، به این خلاصه می‌شود که یک تصویر آموزشی بگیریم، سطح مشخصی از نویز با توزیع نرمال به آن اضافه کنیم و آن نویز شناخته‌شده را با آنچه مدل پیش‌بینی می‌کند مقایسه کنیم. اگر مدل بتواند نویز را با موفقیت پیش‌بینی کند، بعداً می‌توانیم از مدل برای تبدیل نویز خالص به تصویری مشابه داده‌های آموزشی استفاده کنیم.

بخش مهم پاراگراف قبلی، عبارت «سطح مشخصی از نویز با توزیع نرمال» است. نویز با توزیع نرمال را می‌توان با یک پارامتر واحد، یعنی عددی که سطح نویز را مشخص می‌کند، توصیف کرد. آموزش شامل انتخاب تصادفی یک تصویر از مجموعه آموزشی و یک سطح نویز است که هر دو به عنوان ورودی به شبکه داده می‌شوند. خروجی شبکه، تخمین مدل از مقدار نویز است. هرچه تفاوت بین نویز خروجی (که خود یک تصویر است) و نویز اضافه‌شده کمتر باشد، بهتر است. روش‌های استاندارد پس‌انتشار و گرادیان کاهشی برای به حداقل رساندن این تفاوت در مینی‌بچ‌ها اعمال می‌شوند تا زمانی که مدل آموزش‌دیده اعلام شود.

نحوه افزودن نویز به تصاویر آموزشی بر چگونگی و سرعت یادگیری مدل‌ها تأثیر می‌گذارد. نویز معمولاً از یک برنامه ثابت پیروی می‌کند. این برنامه به گونه‌ای است که حرکت از یک سطح نویز فعلی، مثلاً سطح نویز 3، به سطح بعدی، یعنی سطح 4، مقدار مشخصی نویز به تصویر اضافه می‌کند، که این مقدار به یک تابع بستگی دارد. اگر مقدار نویز اضافه‌شده بین هر مرحله یکسان باشد، برنامه خطی است. اما اگر مقدار نویز اضافه‌شده بین مراحل به خود مرحله بستگی داشته باشد، برنامه غیرخطی است و از تابع دیگری پیروی می‌کند.

شکل 8-6 را در نظر بگیرید که یک تصویر آموزشی احتمالی را در سمت چپ نشان می‌دهد. هر ردیف، سطوح متوالی نویز اضافه‌شده به تصویر آموزشی را نشان می‌دهد. ردیف بالا از یک برنامه خطی پیروی می‌کند، که در آن حرکت از چپ به راست، سطح نویز یکسانی را بین هر مرحله اضافه می‌کند تا جایی که تصویر تقریباً نابود می‌شود. ردیف پایین از آنچه به عنوان برنامه

کسینوسی شناخته می‌شود پیروی می‌کند، که تصویر را با سرعت کمتری نابود می‌کند. این به مدل‌های انتشار کمک می‌کند کمی بهتر یاد بگیرند. برای کنجکاوان، آقای شیک‌پوش در تصویر، جد بزرگ من، Emil Kneusel (در حدود سال 1895) است.

شکل 8-6 فقط نُه مرحله را نشان می‌دهد. در عمل، مدل‌های انتشار از صدها مرحله استفاده می‌کنند؛ نکته کلیدی این است که تصویر اصلی در پایان فرآیند کاملاً نابود می‌شود و تنها نویز باقی می‌ماند. این مهم است زیرا نمونه‌برداری از مدل انتشار، فرآیند را معکوس می‌کند تا یک تصویر نویزی تصادفی را به یک تصویر بدون نویز تبدیل کند. در واقع، نمونه‌برداری از مدل انتشار از راست به چپ حرکت می‌کند و با استفاده از شبکه آموزش‌دیده، نویز را پیش‌بینی می‌کند و سپس آن را کم می‌کند تا تصویر قبلی تولید شود. تکرار این فرآیند برای تمام مراحل در برنامه، فرآیند تولید تصویر از نویز را کامل می‌کند.

توضیحات بخش قبلی را می‌توان در دو الگوریتم خلاصه کرد. پیشنهاد می‌کنم آن‌ها را مطالعه کنید، اما از آنجا که کمی فنی هستند، همیشه می‌توانید به بخش بعدی بروید.

الگوریتم رو به جلو مدل انتشار را آموزش می‌دهد و الگوریتم معکوس از مدل آموزش‌دیده در طول استنتاج نمونه‌برداری می‌کند تا تصاویر خروجی تولید کند. ابتدا با الگوریتم رو به جلو شروع می‌کنیم. مراحل زیر را تا زمانی که مدل را آموزش‌دیده اعلام کنیم، تکرار می‌کنیم:

1. یک تصویر آموزشی، x_0، به صورت تصادفی انتخاب کنید.
2. یک گام زمانی تصادفی، t، در بازه 1 تا T (حداکثر تعداد مراحل) انتخاب کنید.
3. یک تصویر نویزی، e، از یک توزیع نرمال استاندارد نمونه‌برداری کنید.
4. یک تصویر نویزی، x_t، را با استفاده از x_0، t و e تعریف کنید.
5. تصویر x_t را از طریق مدل عبور دهید و تخمین نویز خروجی را با e مقایسه کنید.
6. پس‌انتشار استاندارد و گرادیان کاهشی را برای به‌روزرسانی وزن‌های مدل اعمال کنید.

الگوریتم رو به جلو کار می‌کند زیرا راه ساده‌ای برای به دست آوردن xt از x0، یعنی تصویر موجود در مجموعه آموزشی، و یک گام زمانی تصادفی، t، وجود دارد. در اینجا، T حداکثر گام زمانی ممکن است که در آن تصویر آموزشی به نویز خالص تبدیل شده است. معمولاً T چند صد مرحله است. به یاد داشته باشید که مدل انتشار در تلاش است تا یاد بگیرد چگونه نویز موجود در e را پیش‌بینی کند. عمل مجبور کردن مکرر مدل به بهتر و بهتر شدن در پیش‌بینی نویز استفاده‌شده برای خراب کردن تصویر آموزشی، همان چیزی است که به گام معکوس اجازه کار می‌دهد.

الگوریتم معکوس از مدل انتشار آموزش‌دیده توسط الگوریتم رو به جلو نمونه‌برداری می‌کند تا یک تصویر خروجی جدید تولید کند، که از یک تصویر نویز خالص در x_T شروع می‌شود (تصاویر سمت راست در شکل 8-6 را در نظر بگیرید). با تکرار مراحل زیر، مدل انتشار برای T مرحله استفاده می‌شود تا نویز را به تصویر تبدیل کند:

1. اگر این آخرین گام از x_1 به x_0 نیست، یک تصویر نویزی، z، از یک توزیع نرمال استاندارد نمونه‌برداری کنید.

2. با کم کردن خروجی مدل انتشار از x_t و اضافه کردن z، تصویر x_{t-1} را از x_t ایجاد کنید.

اگر به شکل 8-6 فکر کنیم، الگوریتم معکوس از راست به چپ حرکت می‌کند. هر گام به سمت چپ با کم کردن خروجی مدل انتشار با استفاده از تصویر فعلی به عنوان ورودی به دست می‌آید، و به این ترتیب از گام زمانی t به گام زمانی قبلی، $t - 1$، حرکت می‌کنیم. تصویر نویزی استاندارد، z، تضمین می‌کند که x_{t-1} یک نمونه معتبر از توزیع احتمالی است که x_{t-1} را از x_t تأمین می‌کند. همان‌طور که اشاره شد، از بسیاری از نظریه‌های احتمال صرف‌نظر می‌کنیم.

الگوریتم نمونه‌برداری کار می‌کند زیرا مدل انتشار نویز موجود در ورودی خود را تخمین می‌زند. این منجر به تخمینی از تصویری می‌شود که به طور معقول، x_t را از x_{t-1} ایجاد کرده است. تکرار این فرآیند برای تمام T مرحله، در نهایت ما را به x_0، یعنی خروجی شبکه، می‌رساند. توجه کنید که بر خلاف شبکه‌های قبلی ما که یک ورودی داشتند و یک خروجی تولید می‌کردند، مدل‌های انتشار به طور مکرر اجرا می‌شوند و هر بار تصاویری با نویز کمتر و کمتر تولید می‌کنند، تا اینکه در نهایت تصویری مشابه داده‌های آموزشی تولید می‌شود.

مدل‌های انتشار مانند GANهای استاندارد بدون شرط هستند: تصویری که تولید می‌شود قابل کنترل نیست. ممکن است گمان کنید اگر یک GAN را بتوان به نحوی شرطی کرد تا فرآیند تولید را هدایت کند، شاید یک مدل انتشار نیز به طور مشابه قابل هدایت باشد. اگر این‌طور فکر می‌کنید، درست است.

GANی که ما برای تولید تصاویر شبه MNIST از اعداد استفاده کردیم، با گسترش ورودی به مولد با یک بردار وان-هات که برچسب کلاس مورد نظر را انتخاب می‌کرد، شرطی شده بود. شرطی کردن یک مدل انتشار به این سادگی نیست، اما می‌توان در طول آموزش، سیگنالی مرتبط با تصویر به شبکه ارائه داد. معمولاً این سیگنال یک بردار جاسازی (Embedding) است که توضیح متنی از محتوای تصویر آموزشی را نشان می‌دهد. ما به طور مختصر در فصل پنجم با جاسازی‌ها آشنا شدیم و در فصل هفتم هنگام بحث درباره مدل‌های زبانی بزرگ دوباره با آن‌ها روبه‌رو خواهیم شد.

فعلاً، تنها چیزی که باید بدانیم این است که یک جاسازی متنی، رشته‌ای مانند «یک سگ قرمز بزرگ» را می‌گیرد و آن را به برداری بزرگ تبدیل می‌کند، که ما آن را به عنوان یک نقطه در فضایی با ابعاد بالا در نظر می‌گیریم: فضایی که معنا و مفاهیم را ضبط کرده است. ارتباط چنین جاسازی متنی‌ای در طول آموزش، در حالی که شبکه در حال یادگیری پیش‌بینی نویز در تصاویر است، شبکه را تقریباً به همان روشی شرطی می‌کند که بردار کلاس وان-هات یک مولد GAN را شرطی می‌کند.

پس از آموزش، وجود یک جاسازی متنی هنگام نمونه‌برداری، سیگنال مشابهی را فراهم می‌کند تا تصویر خروجی را هدایت کند تا شامل عناصری مرتبط با متن باشد. در زمان نمونه‌برداری، متن به یک دستور (Prompt) تبدیل می‌شود که تصویری را که می‌خواهیم فرآیند انتشار تولید کند، توصیف می‌کند.

مدل‌های انتشار معمولاً با یک تصویر نویزی تصادفی شروع می‌شوند. اما لزومی ندارد این‌طور باشد. اگر بخواهیم خروجی شبیه به یک تصویر موجود باشد، می‌توانیم از آن تصویر با مقداری نویز اضافه‌شده، به عنوان تصویر اولیه استفاده کنیم. نمونه‌های گرفته‌شده از آن تصویر، بسته به درجه نویز اضافه‌شده، کم‌وبیش شبیه به آن خواهند بود. حالا، بیایید نگاهی به مدل‌های انتشار شرطی بیندازیم.

مدل‌های انتشار تجاری، مانند DALL-E 2 از OpenAI یا Stable Diffusion از Stability AI، از متن یا تصویر ارائه‌شده توسط کاربر استفاده می‌کنند تا فرآیند انتشار را به سمت یک تصویر خروجی هدایت کنند که نیازهای دستور (Prompt) را برآورده کند. نمونه‌های نشان‌داده‌شده در این بخش توسط Stable Diffusion با استفاده از محیط آنلاین DreamStudio تولید شده‌اند. شکل 9-6 مونالیزای لئوناردو داوینچی (بالا سمت چپ) را به همراه پنج نسخه متفاوت از آن نشان می‌دهد.

نسخه‌های متفاوت نتیجه کار Stable Diffusion در پاسخ به تصویر اصلی و یک دستور متنی هستند:

پرتره زنی با لباس قهوه‌ای به سبک داوینچی، رنگ‌های نرم و خاکی.

رابط DreamStudio به کاربر اجازه می‌دهد یک تصویر اولیه ارائه دهد و با استفاده از یک اسلایدر، مقدار نویز اضافه‌شده را از 0 درصد برای یک تصویر کاملاً نویزی تا 100 درصد برای بدون نویز تنظیم کند. (بله، این برای من هم عجیب به نظر می‌رسد.) نسخه نویزی تصویر، فرآیند انتشار را آغاز می‌کند. هرچه درصد بیشتر باشد، نویز کمتری اضافه می‌شود و تصویر اولیه تأثیر بیشتری بر خروجی نهایی دارد.

برای مونالیزا، من 33 درصد نویز استفاده کردم. این سطح نویز، همراه با دستور متنی و یک سبک قابل انتخاب توسط کاربر، پنج نسخه متفاوت را در شکل 9-6 تولید کرد. تنها تفاوت بین این نسخه‌ها، سبک انتخاب‌شده است (ردیف بالا: انیمه و هنر فانتزی؛ ردیف پایین: ایزومتریک، خطوط هنری و عکاسی).

نتایج چشمگیر هستند. این تصاویر نقاشی نشده‌اند، بلکه از یک نسخه نویزی مونالیزا و یک دستور متنی که به عنوان راهنما برای هدایت فرآیند انتشار استفاده شده، به وجود آمده‌اند. درک این موضوع دشوار نیست که توانایی تولید تصاویر جدید در پاسخ به دستورهای متنی، دنیای هنر تجاری را تحت تأثیر قرار خواهد داد.

با این حال، تولید تصویر توسط هوش مصنوعی کامل نیست. خطاهایی رخ می‌دهند؛ همان‌طور که در شکل 10-6 نشان داده شده است. هیچ یک از این تصاویر واقعاً آن چیزی نبودند که من می‌خواستم! مدل‌های انتشار به وضوح در ترسیم دست‌ها مشکل خاصی دارند.

نوشتن دستورهای متنی مؤثر به یک هنر تبدیل شده است، هنری که همین حالا هم نوع جدیدی از شغل را ایجاد کرده: مهندس دستور متنی (Prompt Engineer). شکل دقیق دستور متنی به شدت بر فرآیند تولید تصویر تأثیر می‌گذارد، همان‌طور که تصویر نویزی تصادفی اولیه نیز چنین تأثیری دارد. رابط DreamStudio به کاربران اجازه می‌دهد تا دانه (seed) مولد اعداد شبه‌تصادفی را ثابت نگه دارند، به این معنا که فرآیند انتشار هر بار با همان تصویر نویزی شروع می‌شود. ثابت کردن دانه در حالی که دستور متنی را کمی تغییر می‌دهیم، به ما امکان می‌دهد آزمایش کنیم تا ببینیم فرآیند انتشار تا چه حد می‌تواند حساس باشد.

تصاویر موجود در شکل 11-6 با جابه‌جایی کلمات «تزئینی»، «سبز» و «گلدان» تولید شده‌اند. (این تصاویر در کتاب به صورت سیاه و سفید نشان داده شده‌اند، اما همه دارای طیف‌های مشابهی از سبز هستند.) تصویر نویزی اولیه هر بار یکسان بود؛ تنها ترتیب این سه کلمه تغییر کرد. سه گلدان شبیه به هم هستند، اما چهارمی کاملاً متفاوت است. با این حال، هر چهار مورد نمونه‌های معتبری از گلدان‌های سبز و تزئینی هستند.

ترتیب و نحوه بیان دستور متنی مهم است، زیرا بردار جاسازی که از دستور متنی تشکیل می‌شود، متفاوت است، حتی اگر کلمات دستور یا معانی آن‌ها مشابه باشند. دستورات مربوط به سه گلدان اول احتمالاً در فضای جاسازی متنی نزدیک به یکدیگر قرار گرفته‌اند، که توضیح می‌دهد چرا ظاهر آن‌ها بسیار شبیه است. دستور آخر، به هر دلیلی، در جای دیگری قرار گرفته و منجر به ویژگی‌های متفاوت تصویر تولیدشده شده است. جالب اینجاست که دستور برای تصویر آخر «گلدان سبز تزئینی» بود، فرمی که از قواعد دستوری پیروی می‌کند.

کنجکاو شدم و دستور «گلدان سبز تزئینی» را تغییر دادم، کلمه «سبز» را با رنگ‌های دیگر عوض کردم و از همان تصویر نویزی اولیه قبلی استفاده کردم. نتایج در شکل 12-6 نشان داده شده است. از چپ به راست، رنگ‌های مشخص‌شده عبارت بودند از قرمز، ارغوانی، زرد و آبی. سه تصویر اول شبیه به گلدان آخر در شکل 11-6 هستند؛ تنها گلدان آبی به طور قابل‌توجهی متفاوت است.

در طول آزمایش‌هایم، متوجه ویژگی دیگری از مدل‌های انتشار شدم، اینکه تصاویر تولیدشده نویز کمتری نسبت به تصاویر اصلی دارند. فرض کنید یک تصویر ورودی رزولوشن پایین داشته و دانه‌دار باشد. در این صورت، خروجی مدل انتشار دارای رزولوشن بالاتر و واضح‌تر است، زیرا خروجی نتیجه عملیاتی که روی تصویر اصلی اعمال شده نیست، بلکه بازآفرینی تصویر با استفاده از دستور متنی برای هدایت است. آیا ممکن است بتوان از مدل‌های انتشار برای حذف عیوب تصویر استفاده کرد، اگر وفاداری مطلق به تصویر اصلی به شدت لازم نباشد؟

شکل 13-6 سعی دارد به این سؤالم پاسخ دهد. تصویر اصلی با رزولوشن 195×256 پیکسل که به 586×768 پیکسل (ضریبی از 3) ارتقا یافته، در سمت چپ قرار دارد. تصویر با استفاده از یک برنامه پردازش تصویر استاندارد و درون‌یابی مکعبی ارتقا یافته است. خروجی مدل انتشار، که آن هم 586×768 پیکسل است، در سمت راست قرار دارد. خروجی مدل انتشار از تصویر اصلی 195×256 پیکسل با 25 درصد نویز اضافه‌شده، سبک عکاسی و دستور متنی «جزئی، اصلی» استفاده کرده است؛ تصویر حاصل از انتشار بهتر است. این تصویر با تصویر اصلی یکسان نیست، اما کپی نزدیکی از آن است. من اعتقاد ندارم که این روش با شبکه‌های سوپر-رزولوشن مبتنی بر یادگیری عمیق رقابت کند، اما صرف‌نظر از کاربرد نهایی، این نکته‌ای جالب دربارهٔ مدل‌های انتشار بود.

به عنوان مثال دیگر، شکل 14-6 را در نظر بگیرید که تصویری از Western Meadowlark را نشان می‌دهد که از فاصله حدود 100 متری در هوای دودی و نامناسب کلرادو گرفته شده است (سمت چپ). تصویر میانی نشان‌دهنده بهترین تلاش برای بهبود تصویر با استفاده از یک برنامه استاندارد دستکاری تصویر (Gimp) است. نسخه سمت راست خروجی Stable Diffusion است که تصویر میانی با مقدار کمی نویز اضافه‌شده (حدود 12 درصد) و دستور متنی زیر به آن داده شده است:

«Western Meadowlark، به شدت جزئی، رزولوشن بالا، بدون نویز»

Stable Diffusion معجزه‌ای انجام نداد، اما خروجی قطعاً بهتر از تصویر اصلی است.

این فصل دو نوع شبکه مولد را بررسی کرد: شبکه‌های مولد تخاصمی (GANها) و مدل‌های انتشار که هر دو از ورودی‌های تصادفی تصاویر ایجاد می‌کنند.

GANها به طور مشترک شبکه‌های مولد و متمایزگر را آموزش می‌دهند تا به مولد بیاموزند خروجی‌هایی تولید کند که متمایزگر را فریب دهد. GANهای شرطی از برچسب‌های کلاس در طول آموزش و تولید استفاده می‌کنند تا مولد را به سمت خروجی‌هایی هدایت کنند که اعضای یک کلاس مشخص‌شده توسط کاربر باشند. GANهای قابل کنترل، جهت‌هایی را از طریق فضای بردار نویز مرتبط با ویژگی‌های اساسی خروجی تولیدشده یاد می‌گیرند، به گونه‌ای که حرکت در آن جهت‌ها به طور قابل پیش‌بینی تصویر خروجی را تغییر می‌دهد.

مدل‌های انتشار یاد می‌گیرند مقدار نویز موجود در یک تصویر را پیش‌بینی کنند. آموزش یک مدل انتشار شامل تغذیه آن با تصاویر آموزشی تمیز است که به طور عمدی با مقدار شناخته‌شده‌ای نویز خراب شده‌اند. پیش‌بینی مدل و نویز اضافه‌شده شناخته‌شده برای به‌روزرسانی وزن‌های مدل استفاده می‌شود.

مدل‌های انتشار شرطی، یک جاسازی (معمولاً از توضیح متنی محتوای تصویر آموزشی) را با نویز مرتبط می‌کنند، به طوری که در زمان تولید، مدل به سمت تصاویری هدایت می‌شود که شامل عناصری مرتبط با دستور متنی کاربر باشد. اگر به جای تصویر اولیه تصادفی خالص، از یک تصویر موجود با مقدار مشخصی نویز استفاده شود، نسخه‌های متفاوتی تولید می‌شود.

مقدمه این فصل به سه نوع مدل هوش مصنوعی مولد اشاره کرد. آخرین مورد، مدل‌های زبانی بزرگ، در حال حاضر چنان که متخصصان هوش مصنوعی ادعا می‌کنند تهدیدی است که می‌تواند جهان را به حدی تغییر دهد که با انقلاب صنعتی برابری کند. چنین ادعاهای تأثیرگذاری نیاز به توجه دارند. بنابراین، بیایید به آنچه ممکن است در نهایت هوش مصنوعی حقیقی باشد، بپردازیم.

فصل هفتم: مدل‌های زبانی بزرگ (هوش مصنوعی حقیقی؟)

مورخان در آینده ممکن است به انتشار مدل زبانی بزرگ ChatGPT از OpenAI در پاییز 2022 به عنوان آغاز هوش مصنوعی حقیقی اشاره کنند. با توجه به آنچه تا زمان نگارش این متن در اواخر مارس 2023 دیده‌ام، با این ارزیابی موافقم.

در این فصل، ابتدا به بررسی توانایی‌های مدل‌های زبانی بزرگ موجود می‌پردازیم و سپس با توصیفی از آنچه هستند و چگونه کار می‌کنند ادامه می‌دهیم. با وجود تمام توانایی‌های شگفت‌انگیزشان، این مدل‌ها در نهایت شبکه‌های عصبی هستند که مانند همه شبکه‌های عصبی قبلی ساخته و آموزش داده شده‌اند. این واقعیت به تنهایی نشان می‌دهد که ارتباط‌گرایان (Connectionists) از ابتدا درست می‌گفتند.

من قبلاً نظرم را در این باره که ChatGPT و مدل‌هایی مثل آن چیزی جدید و شایسته نام‌گذاری به عنوان هوش مصنوعی حقیقی هستند، فاش کرده‌ام. امیدوارم که تا پایان این فصل، شما نیز با این نظر موافق باشید.

عبارت هوش مصنوعی تا حدی مبهم است و باید پیش از ادامه، تعریفی دقیق‌تر و ظریف‌تر از آن ارائه شود. متخصصان معمولاً هوش مصنوعی را به دو نوع تقسیم می‌کنند: هوش مصنوعی محدود یا باریک (ANI) و هوش مصنوعی عمومی (AGI). اولی شامل همه چیزهایی است که تاکنون بحث کردیم. دومی به ماشین‌های واقعاً آگاه و هوشمند اشاره دارد؛ موضوعی که در داستان‌های علمی-تخیلی دیده می‌شود.

مدل‌هایی که تا زمان نگارش این کتاب وجود دارند، قطعاً AGI نیستند. با این حال، صرفاً ANI هم نیستند؛ به نظر می‌رسد چیزی کاملاً جدید باشند، چیزی بینابین. عنوان مقاله اخیر محققان مایکروسافت، سباستین بوبک (Sébastien Bubeck) و همکاران، «جرقه‌های هوش مصنوعی عمومی»، برای چنین مدل‌هایی مناسب به نظر می‌رسد.

مدل‌های زبانی بزرگ (LLMها) دستور متنی‌ای را که کاربر ارائه می‌دهد به عنوان ورودی می‌پذیرند. سپس متن خروجی را، کلمه به کلمه (در واقع، توکن به توکن)، با استفاده از دستور و همه کلمات پیشین تولیدشده به عنوان راهنما، تولید می‌کنند. در واقع، تنها هدف طراحی LLMها این است که در پیش‌بینی کلمه بعدی در یک توالی کلمات که با دستور ورودی آغاز شده، بسیار خوب عمل کنند. این تنها چیزی است که آن‌ها برایش آموزش داده شده‌اند. با این حال، این تنها چیزی نیست که آن‌ها یاد می‌گیرند. دلیل هیجان محققان هوش مصنوعی در رابطه با LLMها این است که در طول مسیر، در حالی که می‌آموزند به

عنوان تولیدکنندگان متن حرفه‌ای عمل کنند، مجموعه‌ای از توانایی‌های نوظهور، از جمله پاسخ به سؤالات، استدلال ریاضی، برنامه‌نویسی با کیفیت بالا و استدلال منطقی را نیز فرا می‌گیرند.

پیامدهای فلسفی این توانایی‌های نوظهور و غیرمنتظره عمیق است. توانایی‌های LLMها سؤالاتی درباره طبیعت تفکر، معنای آگاهی و منحصربه‌فرد بودن (مفروض) ذهن انسان مطرح می‌کنند. ما در موقعیتی نیستیم که به این سؤالات به‌طور عمیق پاسخ دهیم، اما در فصل هشت به برخی از آن‌ها بازخواهم گشت.

حالا، بیایید با بررسی توانایی‌های LLMها وارد موضوع شویم. اینجا تنها می‌توانیم به چند توانایی نوظهور اشاره کنیم، اما اینترنت از قبل پر از مثال‌های شگفت‌انگیز، خنده‌دار و حتی ترسناک از LLMها در عمل است. پیشنهاد می‌کنم نگاهی به این مثال‌ها بیندازید، هرچند همان‌طور که همیشه باید مراقب بود، باید کیفیت منبع اطلاعات را قضاوت کرد.

جامعه هوش مصنوعی در این حوزه با حداکثر سرعت پیش می‌رود. تا زمانی که این کلمات را بخوانید، بدون شک LLMها حتی شگفت‌انگیزتر از آنچه در زمان نگارش من هستند خواهند بود.

در ادامه، مثال‌هایی از عملکرد مدل‌های زبانی بزرگ (LLMها) ارائه شده است. برای اکثر موقعیت‌ها، خروجی سه مدل را نشان خواهم داد: Bard، GPT-3.5 و GPT-4. تمام خروجی‌ها عملکرد مدل‌ها را تا اواخر مارس 2023 نشان می‌دهند. در هر مورد، از رابط عمومی از طریق مرورگر وب استفاده کردم. Bard مدل گوگل و GPT-3.5 و GPT-4 از OpenAI هستند.

LLMها کاربرانی متخصص در زبان هستند. این مدل‌ها در قالب گفت‌وگو استفاده می‌شوند و با یک دستور یا سؤال از سمت کاربر آغاز به کار می‌کنند. برای مثال، به هر سه مدل این دستور را دادم:

«داستان کوتاه شبانه در یک بند درباره یک عروسک خرس، یک فرش جادویی و ماه بنویس که برای کودکان زیر شش سال مناسب باشد.»

پاسخ Bard به این صورت بود: «روزی از روزها، یک عروسک خرس کوچک در جعبه اسباب‌بازی زندگی می‌کرد. او عاشق بازی با دوستانش بود، اما همیشه آرزوی پرواز به ماه را داشت. یک روز، یک فرش جادویی در جعبه اسباب‌بازی‌اش پیدا کرد. این فرش می‌توانست پرواز کند! عروسک خرس خیلی هیجان‌زده شد، روی فرش پرید و به ماه پرواز کرد. او زمان فوق‌العاده‌ای را با کاوش در ماه گذراند و حتی با موجودات فضایی دوست‌داشتنی ملاقات کرد. وقتی زمان بازگشت به خانه شد، با دوستان جدیدش خداحافظی کرد و با فرش جادویی‌اش به زمین بازگشت. او خیلی خوشحال بود که به خانه برگشته و نمی‌توانست منتظر بماند تا ماجراجویی‌اش را برای دوستانش تعریف کند.»

این داستان کامل است، در یک بند، و از ساختار جمله‌ای ساده برای کودکان استفاده می‌کند.

GPT-3.5 این‌گونه پاسخ داد: «روزی از روزها، عروسک خرسی به نام تیمی عاشق ماجراجویی بود. یک شب، او روی فرشی جادویی پرید و به سمت ماه پرواز کرد. ماه خیلی بزرگ و درخشان بود و تیمی نمی‌توانست مقاومت کند و آن را کاوش نکند. او در میان دهانه‌ها می‌پرید و می‌جهید و با موجودات فضایی دوست‌داشتنی ملاقات کرد که باغ ماهشان را به او نشان دادند. تیمی زمان خوبی داشت، اما سرانجام دانست که باید به خانه برگردد. او دوباره روی فرش جادویی‌اش پرید و به تخت دنجش بازگشت، جایی که با پتوی مورد علاقه‌اش در آغوش، خوابید و به سرعت به خواب رفت و دربارهٔ ماجراجویی شگفت‌انگیزش در ماه رویا دید.»

داستان GPT-3.5 نیز کامل است و شامل موجودات فضایی دوست‌داشتنی می‌شود. این بار، عروسک خرس نامی دارد، تیمی. زبان همچنان برای کودکان کوچک مناسب است، اما GPT-3.5 توصیفی‌تر بوده است. Bard به ما می‌گوید که عروسک خرس ماه را کاوش می‌کند و با موجودات فضایی دوست‌داشتنی ملاقات می‌کند، اما GPT-3.5 به ما می‌گوید که تیمی در میان دهانه‌ها می‌پرید و می‌جهید و موجودات فضایی دوست‌داشتنی باغ ماهشان را به او نشان دادند.

GPT-4، پیشرفته‌ترین مدل OpenAI، این را تولید کرد: «روزی از روزها، زیر آسمانی پر از ستاره‌های چشمک‌زن، عروسک خرس کوچکی به نام کادِلز (Cuddles) یک فرش جادویی رنگارنگ را که در گوشه اتاق دنجش پنهان شده بود، کشف کرد. با کنجکاوی و هیجان، کادلز روی فرش پرید و با هم به آسمان پرواز کردند و به سمت ماه نقره‌ای رهسپار شدند. در حالی که از میان ابرهای نرم و مخملی سفر می‌کردند، کادلز و فرش جادویی با ستاره‌های چشمک‌زن رقصیدند و رازهایی را به نسیم خنک شب زمزمه کردند. وقتی به ماه رسیدند، ماه با لبخندی ملایم و درخشان از آنها استقبال کرد و آن‌ها شب را با بازی قایم‌باشک در دهانه‌های نرم و گِرد و غبارآلود ماه گذراندند. وقتی خورشید شروع به طلوع کرد، کادلز و فرش جادویی با دوست جدید قمری‌شان خداحافظی کردند و به آرامی به اتاق کوچکشان بازگشتند؛ آماده برای خواب شبانه‌ای آرام که با شیرین‌ترین رویاها پر شده بود.»

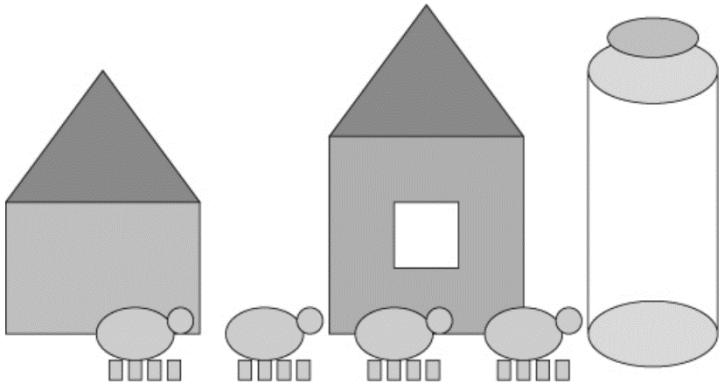
به نظرم، GPT-4 توصیفی‌ترین داستان را تولید کرد، هرچند باغ ماه GPT-3.5 را دوست دارم. داستان Bard کمی خشک بود اما شاید برای کودکان خیلی کوچک قابل درک‌تر باشد. با این حال، شگفت‌انگیزترین چیز درباره همه آن‌ها این است که اصلاً نوشته شده‌اند! کمتر از یک سال پیش (یادتان باشد، برای من اوایل 2023 است)، صحبت درباره داستان‌های تولیدشده توسط هوش مصنوعی چیزی بیشتر از یک آرزوی خیالی نبود.

دستوری که به GPT-4 دادم را تغییر دادم و از آن خواستم یک داستان یک پاراگرافی درباره عروسک خرس، فرش جادویی و ماه بنویسد، اما این بار برای بزرگسالان، نه کودکان. پاسخ چنین بود: «روزی در آغوش غروب، عروسک خرسی تنها، فرسوده و فراموش‌شده به دست گذر زمان، یک فرش جادویی باستانی کشف کرد که لبه‌هایش پاره شده بود اما با انرژی ماورایی می‌درخشید. زیر نگاه نافذ ماه، این زوج سفری شبانه آغاز کردند؛ در میان پیچ‌وخم‌های جهانی سرد و بدبین که رویاهایش رها شده و معصومیت به خاطره‌ای دور بدل شده بود. در حالی که ماه نور ملایم نقره‌ای خود را بر این یاران غیرمنتظره از راه رسیده می‌افکند، عروسک خرس نمادی از رویاهایی شد که مدت‌ها زیر بار نگرانی‌های بزرگسالی دفن شده بودند، در حالی که فرش جادویی از آزادی رام‌نشده و ماجراجویی‌هایی که در دل کسانی که جرأت باور داشتند نهفته بود، زمزمه می‌کرد. با هم پرواز کردند، یادآوری غم‌انگیزی از اشتیاق شیرین اما گذرای جادوی کودکی، و ردی از گَرد ستاره‌ها به جا گذاشتند که با وعده امید و زیبایی غم‌انگیز زمان گذشته، می‌درخشید.»

به وضوح، GPT-4 درک محکمی از زبان و گرامر دارد. ممکن است کسی ادعا کند که این مدل حتی احساسات و عواطف انسانی را درک می‌کند (هرچند این‌طور نیست، مگر نه؟).

بیایید قبل از اینکه به درون این ذهن عجیب و جدید بپردازیم، چند مثال دیگر را مرور کنیم.

GPT-4 علاوه بر تصاویر، زبان‌های برنامه‌نویسی و نشانه‌گذاری متعددی را درک می‌کند، از جمله LaTeX، زبانی که معمولاً در انتشار کتاب‌ها (مانند این کتاب) استفاده می‌شود. از GPT-4 خواستم که LaTeX تولید کند تا یک صحنه ساده روستایی با خانه، انبار، سیلوی غلات و چهار گاو ترسیم کند. شکل 1-7 نتیجه را نشان می‌دهد.



ممکن است با دیدن این تصویر ساده و ابتدایی بخندید، اما به همه چیزهایی که برای خلق آن به کار رفته فکر کنید. این دستور متنی من بود: «کد LaTeX با استفاده از TikZ برای ترسیم موارد زیر تولید کن: یک صحنه روستایی با خانه، انبار، سیلوی غلات و چهار گاو».

GPT-4 باید دستورات را درک می‌کرد: ترسیم یک صحنه با خانه، انبار، سیلوی غلات و چهار گاو. این یعنی باید معانی کلمات کلیدی را بفهمد و آن‌ها را به درستی مرتبط کند، به طوری که «غلات» و «سیلو» با هم مرتبط شوند و همچنین «چهار» و «گاو».

سپس، باید «تصور» می‌کرد که چیدمان صحنه چگونه باشد و هر شیء چگونه می‌تواند با اشکال ساده‌ای که بسته TikZ ارائه می‌دهد، نمایش داده شود. (TikZ یک بسته LaTeX برای ترسیم اشکال گرافیکی ساده است.) شکل آن را نشان نمی‌دهد، اما هر دوی انبار و خانه دارای سقف قرمز هستند. تصادفی است؟

GPT-4 یک مدل انتشار مانند مولدهای تصویر فصل شش نیست. خروجی تولیدشده توسط GPT-4 تصویر شکل 1-7 نبود، بلکه کد LaTeX زیر بود:

```
% Barn
\draw[fill=brown!50] (5,0) rectangle (8,3);
\draw[fill=red!50] (6.5,3)--(8,3)--(6.5,5)--(5,3)--cycle;
\draw[fill=white!70] (6,1) rectangle (7,2);
```

من از کد فوق برای ایجاد شکل 1-7 استفاده کردم. اگر تعجب می‌کنید که GPT-4 چگونه توانسته این کارها را انجام دهد، باید بدانید من و بسیاری دیگر هم همین‌طور هستیم. این توانایی‌ها زمانی که مدل آموزش داده شد، به صورت خودبه‌خود پدیدار شدند؛ این‌ها هدف‌گذاری نشده بودند. به همین دلیل است که باور دارم مورخان آینده پاییز 2022 را به عنوان آغاز هوش مصنوعی حقیقی ثبت خواهند کرد. کلاه‌هایتان را سفت بچسبید؛ این تازه شروع ماجراست.

روان‌شناسان مفهومی به نام «نظریه ذهن» را بحث می‌کنند که به توانایی استنباط حالات ذهنی دیگران و نسبت دادن افکار یا باورها به آن‌ها اشاره دارد. تنها انسان‌ها و برخی از میمون‌ها دارای نظریه ذهن هستند. یعنی تا حالا این‌طور بوده است.

به نظر می‌رسد که مدل‌های زبانی بزرگ نیز نظریه ذهن را در سطحی مشابه با کودکان انسانی بزرگ‌تر، اگر نگوییم بالاتر از آن‌ها (در مورد GPT-4)، دارا هستند. این نتیجه‌ای است که میخال کوسینسکی (Michal Kosinski) از دانشگاه استنفورد در مقاله‌اش در سال 2023 با عنوان «نظریه ذهن ممکن است به طور خودبه‌خود در مدل‌های زبانی بزرگ پدیدار شده باشد» به آن رسیده است. بیایید این ادعا را با پرسیدن سؤالی از Bard، GPT-3.5 و GPT-4 آزمایش کنیم که نیازمند دانستن چیزی

است که ممکن است در ذهن شخص دیگری باشد. دستور متنی چنین است: «مری به آماندا گفت که عاشق جان است. آماندا به کتی گفت که مری چه گفته بود. یک هفته بعد، مری به آماندا گفت که دیگر عاشق جان نیست بلکه حالا عاشق رالف است. بعد از آن، آماندا با کتی درباره آب‌وهوا صحبت کرد. کتی فکر می‌کند عشق مری کیست؟»

قبل از ادامه، خودتان نتیجه‌گیری کنید: کتی فکر می‌کند مری عاشق کیست؟

هر مدل به سؤال پاسخ داد و سپس استدلال خود را ارائه کرد. این پاسخ Bard است:

«کتی فکر می‌کند عشقِ مری رالف است. کتی می‌داند که مری عاشق جان بود، اما همچنین می‌داند که مری دیگر عاشق جان نیست. کتی همچنین می‌داند که مری حالا عاشق رالف است. بنابراین، کتی فکر می‌کند عشقِ مری رالف است. ممکن است کتی اشتباه کند. همچنین ممکن است مری مطمئن نباشد که عاشق کیست. با این حال، بر اساس اطلاعاتی که کتی دارد، به نظر می‌رسد کتی فکر می‌کند عشقِ مری رالف است.»

من این را جالب می‌دانم که Bard استدلال خود را توجیه کرد و سپس درباره آنچه مری ممکن است فکر کند نظر داد. با این حال، Bard فرض کرد که آماندا حتماً به کتی درباره تغییر علاقه مری اطلاع داده است.

این نظر GPT-3.5 درباره این سناریو است: «بر اساس اطلاعات داده‌شده، غیرممکن است تعیین کنیم کتی فکر می‌کند عشق مری کیست. آخرین قطعه اطلاعات ما این است که آماندا با کتی درباره آب‌وهوا صحبت کرد و هیچ نشانه‌ای نیست که کتی از تغییر نظر مری مطلع شده باشد. بنابراین، کتی ممکن است هنوز فکر کند مری عاشق جان است یا ممکن است هیچ نظری درباره این موضوع نداشته باشد.»

GPT-3.5 فرض مشابهی که Bard می‌کند را در نظر ندارد. این مدل آگاه است که حقایق تنها نشان می‌دهند که آماندا و کتی پس از تغییر احساسات مری، درباره آب‌وهوا صحبت کرده‌اند. من تحت تأثیر قرار گرفتم که از عبارت «آخرین قطعه اطلاعات» استفاده کرد، که نشان می‌دهد GPT-3.5 یک جورهایی رویدادهای دستور متنی را به ترتیب مرتب کرده است.

در نهایت، GPT-4 چه می‌گوید؟ بیایید ببینیم: «کتی فکر می‌کند عشق مری جان است. این به این دلیل است که آماندا قبلاً به کتی درباره احساسات مری نسبت به جان گفته بود، اما وقتی مری عاشق رالف شد، به کتی درباره تغییر احساسات مری چیزی نگفت.»

پاسخ GPT-4 پاسخی است که اکثر انسان‌ها احتمالاً می‌دهند. دستور متنی تنها بیان می‌کند که آماندا و کتی درباره آب‌وهوا صحبت کرده‌اند، بنابراین اکثر انسان‌ها احتمالاً این را به این معنا بگیرند که آماندا و کتی درباره موضوع دیگری صحبت نکرده‌اند.

مدل‌های زبانی بزرگ، همان‌طور که اکنون ارائه شده‌اند، از فرمت چَت استفاده می‌کنند، به این معنا که کاربر می‌تواند دستورهای دنباله‌دار وارد کند و مدل این دستورها را به بخش‌های قبلی مکالمه اعمال خواهد کرد. برای مثال، من با GPT-4 با این دستور ادامه دادم: «ما می‌دانیم که آماندا با کتی درباره آب‌وهوا صحبت کرده است. آیا این بدین معناست که درباره چیز دیگری صحبت نکرده است؟ این ممکن است چگونه بر آنچه کتی درباره عشق مری باور دارد تأثیر بگذارد؟»

که GPT-4 این‌گونه پاسخ داد: «پاسخ قبلی من فرض کرده بود که آماندا پس از اینکه مری احساسات جدیدش به رالف را فاش کرد، فقط درباره آب‌وهوا با کتی صحبت کرده است. با این حال، چون متن به صراحت بیان نمی‌کند که آماندا درباره احساسات

به‌روز شده مری نسبت به رالف چیزی نگفته، ممکن است آماندا به کتی درباره عشق جدید مری اطلاع داده باشد. اگر این‌طور باشد، کتی می‌داند که مری حالا عاشق رالف است. متن اطلاعات کافی برای نتیجه‌گیری قطعی درباره اینکه کتی درباره عشق مری چه باور دارد، ارائه نمی‌دهد.»

دستور دنباله‌دار من باعث شد GPT-4 تشخیص دهد که در پاسخ اولش ممکن است فرضی نادرست کرده باشد. در کار با LLMها در حالت چت، اغلب نیاز به سطحی از تعامل بین کاربر و مدل است تا به مدل کمک کند تمرکز کند، همان‌طور که آزمایش بعدی ما نشان خواهد داد.

آموس تْوِرسکی (Amos Tversky) و دانیل کانمن (Daniel Kahneman) به ارائه مشکلی معروف شده‌اند که اکنون به عنوان «مسئله لیندا» شناخته می‌شود:

لیندا 31 ساله، مجرد، صریح‌الحرف و بسیار باهوش است. او در رشته فلسفه تحصیل کرده است. در دوران دانشجویی، او عمیقاً نگران مسائل تبعیض و عدالت اجتماعی بود و همچنین در تظاهرات ضد هسته‌ای شرکت داشت. کدام محتمل‌تر است؟

1. لیندا یک کارمند بانک است.

2. لیندا یک کارمند بانک است و در جنبش فمینیستی فعال است.

تقریباً 80 درصد از انسان‌ها گزینه 2 را انتخاب می‌کنند.

بر اساس نظریه پایه احتمال، پاسخ صحیح گزینه 1 است. جالب اینجاست که بر اساس نتایج تْوِرسکی و کانمن، حتی ریاضیدان‌ها و آماردان‌ها نیز گزینه 2 را انتخاب کرده‌اند، که نشان‌دهنده سوگیری در تفکر انسانی است. آیا مدل‌های زبانی بزرگ (LLMها) نیز سوگیری مشابهی نشان خواهند داد؟ هر چه باشد داده‌های آموزشی آن‌ها توسط انسان‌ها تولید شده است. بیایید بفهمیم.

مدل‌های زبانی که در حال بررسی آن‌ها هستیم، روی بخش قابل توجهی از محتوای اینترنت آموزش دیده‌اند، به این معنا که داده‌های آموزشی مدل‌ها احتمالاً مسئله لیندا را در بر دارد. بنابراین، متن را تغییر دادم تا همان مشکل را با فرمت متفاوتی ارائه دهم: «ابنر (Abner) 27 ساله، مجرد، صریح‌الحرف و بسیار باهوش است. او در رشته‌های فلسفه و فیزیک تحصیل کرده است. در دوران دانشجویی، او به طور فعال نگران مسائل عدالت اجتماعی بود و در تظاهرات ضد پلیس شرکت داشت. کدام **ممکن‌تر** است، اینکه ابنر مکانیک خودرو باشد یا مکانیک خودرو که در ترویج اصلاحات پلیس فعال است؟»

توجه کنید که عمداً از کلمه «ممکن» به جای «محتمل» استفاده کردم. پاسخ‌های مدل‌ها به این صورت بود:

GPT-4: بر اساس اطلاعات ارائه‌شده، به نظر می‌رسد ممکن‌تر است که ابنر مکانیک خودرو باشد که در ترویج اصلاحات پلیس فعال است. این به دلیل آن است که پیشینه او نشان‌دهنده علاقه شدیدی به مسائل عدالت اجتماعی است و او در تظاهرات ضد پلیس شرکت داشته است. در حالی که ممکن است ابنر صرفاً یک مکانیک خودرو باشد، فعالیت‌ها و علاقه‌مندی‌های گذشته‌اش نشان می‌دهد که امکان بیشتری دارد که او همچنین در ترویج اصلاحات پلیس نیز درگیر باشد.

GPT-3.5: بر اساس اطلاعات ارائه‌شده، ممکن‌تر است که ابنر مکانیک خودرو باشد که در ترویج اصلاحات پلیس فعال است. این به دلیل آن است که پیشینه ابنر به عنوان دانشجوی فلسفه و فیزیک، همراه با مشارکت فعالش در مسائل عدالت اجتماعی، نشان می‌دهد که او احتمالاً به مسائل گسترده‌تر اجتماعی فراتر از حرفه‌اش علاقه‌مند است. علاوه بر این، شرکت او در تظاهرات

ضد پلیس نشان‌دهنده علاقه و اشتیاق او به اصلاحات پلیس است، که این ایده را تقویت می‌کند که او مکانیک خودرو است که در ترویج اصلاحات پلیس فعال است.

Bard: بر اساس اطلاعات ارائه‌شده، ممکن‌تر است که ابنر مکانیک خودرو باشد که در ترویج اصلاحات پلیس فعال است تا اینکه صرفاً یک مکانیک خودرو باشد.

Bard با توجیه پاسخ خود ادامه داد و تلاش کرد حالات ذهنی ابنر را تفسیر کند.

همه سه مدل گزینه دوم را انتخاب کردند، هرچند GPT-4 پاسخ خود را با ظرافت ارائه داد و امکان اینکه ابنر صرفاً یک مکانیک خودرو باشد را پذیرفت.

سپس گفت‌وگوهای جدیدی آغاز کردم و دستور متنی را تنظیم نمودم تا درباره «احتمال» مکانیک بودن یا مکانیک فعال در اصلاحات پلیس بودن ابنر سؤال کنم:

ابنر 27 ساله، مجرد، صریح‌الحرف و بسیار باهوش است. او در رشته‌های فلسفه و فیزیک تحصیل کرده است. در دوران دانشجویی، او به طور فعال نگران مسائل عدالت اجتماعی بود و در تظاهرات ضد پلیس شرکت داشت. کدام احتمال بیشتری دارد:

(الف) اینکه ابنر مکانیک خودرو باشد یا (ب) اینکه ابنر مکانیک خودرو باشد که در اصلاحات پلیس فعال است؟

این، متن را با نسخه اصلی لیندا هم‌راستا می‌کند. پاسخ‌های مدل‌ها به این دستور متنی به شرح زیر بود:

GPT-4: بر اساس قاعده هم‌پیوندی در نظریه احتمال، احتمال وقوع همزمان دو رویداد (در «هم‌پیوندی») همیشه کمتر یا برابر با احتمال وقوع هر یک از رویدادها به صورت جداگانه است.

در این مورد، دو رویداد به این ترتیب هستند:

1. ابنر مکانیک خودرو باشد.

2. ابنر در اصلاحات پلیس فعال باشد.

گزینه (ب) نمایانگر هم‌پیوندی این دو رویداد است. بنابراین، احتمال (الف) اینکه ابنر مکانیک خودرو باشد، بیشتر از احتمال (ب) اینکه ابنر مکانیک خودرو باشد و همچنین در اصلاحات پلیس فعال باشد، است.

GPT-3.5: بر اساس اطلاعات ارائه‌شده، محتمل‌تر است که ابنر مکانیک خودرو (گزینه الف) باشد تا مکانیک خودرو که در اصلاحات پلیس فعال است (گزینه ب). اگرچه ابنر بسیار باهوش است و در رشته‌های فلسفه و فیزیک تحصیل کرده، هیچ نشانه‌ای وجود ندارد که او آموزش یا تحصیل رسمی در مکانیک خودرو داشته باشد. از سوی دیگر، مشارکت او در تظاهرات ضد پلیس نشان می‌دهد که ممکن است به اصلاحات پلیس علاقه‌مند باشد، اما این لزوماً به این معنا نیست که به عنوان مکانیک خودرو در آن فعال باشد. بنابراین، احتمال گزینه الف بیشتر از گزینه ب است.

Bard: من می‌گویم که (الف) اینکه ابنر مکانیک خودرو باشد، احتمال بیشتری دارد.

ابنر 27 ساله، مجرد، صریح‌الحرف و بسیار باهوش است. او در رشته‌های فلسفه و فیزیک تحصیل کرده است. در دوران دانشجویی، او به طور فعال نگران مسائل عدالت اجتماعی بوده و در تظاهرات ضد پلیس شرکت داشته است.

بر اساس این اطلاعات، محتمل‌تر است که ابنر مکانیک خودرو باشد تا مکانیک خودرو که در اصلاحات پلیس فعال است. این به دلیل موارد زیر است:

- مکانیک‌های خودرو شایع‌تر از مکانیک‌های خودرو که در اصلاحات پلیس فعال هستند، هستند.

- تحصیلات و علاقه‌مندی‌های ابنر بیشتر با شغل مکانیک خودرو هم‌راستا است تا با شغل مرتبط با اصلاحات پلیس.

- مشارکت ابنر در تظاهرات ضد پلیس نشان می‌دهد که او به پلیس انتقاد دارد، که این ویژگی معمولاً با مکانیک‌های خودرو مرتبط نیست.

البته، ممکن است ابنر هم مکانیک خودرو باشد و هم در اصلاحات پلیس فعال باشد. با این حال، بر اساس اطلاعات ارائه‌شده، محتمل‌تر است که او صرفاً یک مکانیک خودرو باشد.

هر سه مدل زمانی که دستور متنی شامل کلمه «احتمال» باشد، به پاسخ صحیح می‌رسند. با این حال، تنها GPT-4 دستور را با در نظر گرفتن نظریه احتمال تفسیر می‌کند. GPT-3.5 و Bard هر دو استدلال‌هایی با کمی نقص برای انتخاب پاسخ صحیح ارائه می‌دهند.

مدل‌های زبانی بزرگ نظریه ذهن را نشان می‌دهند و می‌توانند مسائل کلمه‌ای احتمال را تفسیر کنند، همراه با توانایی‌های ریاضی دیگری که در اینجا نادیده گرفته‌ایم. سؤال بعدی این است: آیا LLMها می‌توانند یک رمزنگاری جایگزینی ساده را حل کنند؟

برای حل رمز، مدل‌ها باید درک کنند که از آن‌ها چه خواسته شده، اینکه چیزی به نام کلید وجود دارد و آن کلید چه چیزی را نشان می‌دهد، و در نهایت چگونه از کلید برای جایگزینی حروف در پیام رمزگذاری‌شده با حروف مورد نیاز برای رمزگشایی استفاده کنند. به عبارت دیگر، باید الگوریتمی را با استفاده از اطلاعات داده‌شده اعمال کنند.

یک رمزنگاری جایگزینی، جفتی بین حروف الفبا، که به ترتیب قرار گرفته‌اند و چینشی دیگر از آن حروف، ایجاد می‌کند. رمز استفاده‌شده در اینجا به این صورت است:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ

UESOZBYRJNDWTFKPACLHIXVQGM
```

برای مثال، برای رمزگذاری یک C، باید ردیف بالایی را بررسی کنید تا C را پیدا کنید، سپس C را با حرفی که در ردیف پایین متناظر آن است، یعنی S، جایگزین کنید. برای رمزگشایی یک حرف، باید ردیف پایین را بررسی کنید و آن را با حرفی که در ردیف بالا متناظر آن است جایگزین کنید. در طول آزمایش، کلید به صورت متنی نمایش داده می‌شود، مثلاً U به A، E به B و به همین ترتیب.

پیام رمزگذاری‌شده من Eat at Moe’s بود، بدون فاصله یا علائم نگارشی و با حروف بزرگ:

```
EATATMOES ⟶ ZUHUHTKZL
```

دستور متنی داده شده به مدل چنین بود:

کلید به این صورت است: U به A، E به B، S به C، O به D، Z به E، B به F، Y به G، R به H، J به I، N به J، D به K، W به L، T به M، F به N، K به O، P به P، A به Q، C به R، L به S، H به T، I به U، X به V، V به W، Q به X، G به Y، M به Z تبدیل گردد. حالا پیام را رمزگشایی کنید: ZUHUHTKZL

بیایید پاسخ‌های مدل‌ها را مرور کنیم. از GPT-4 شروع می‌کنیم:

ابتدا بیایید رمز داده‌شده را برای ارجاع آسان‌تر فهرست کنیم:

(فهرست رمز حذف شده)

حالا بیایید پیام ZUHUHTKZL را رمزگشایی کنیم:

```
Z -> E
U -> A
H -> T
U -> A
H -> T
T -> M
K -> O
Z -> E
L -> S
```

پس پیام رمزگشایی شده چنین است: EATATMOES

GPT-4 توانست رمز را به پیام رمزگذاری‌شده اعمال کند. اما آیا مدل درک می‌کند که پیام رمزگشایی‌شده یک جمله است؟ بیایید از آن بپرسیم:

این درست است. حالا فاصله‌گذاری و علائم نگارشی مناسب را به رشته EATATMOES اضافه کن تا یک جمله کامل تشکیل شود.

پاسخ GPT-4 به این صورت بود:

جمله با فاصله‌گذاری و علائم نگارشی مناسب: Eat at Moe’s.

واقعاً درست است.

GPT-3.5 هم با موفقیت پیام را رمزگشایی کرد. سپس این پیام را بعد از EATATMOES اضافه کرد: ممکن است این یک اشتباه املایی برای کلمهٔ «TOMATOES» باشد.

من این جمله اضافی را جالب می‌دانم. آیا GPT-3.5 در تلاش بود تا معنای پیام رمزگشایی‌شده را تفسیر کند و با ناتوانی در این کار، به یک اشتباه املایی احتمالی روی آورد؟

Bard ادعا کرد که پیام رمزگشایی‌شده «سال نو مبارک» (HAPPY NEW YEAR) است و توضیح سه مرحله‌ای ارائه داد:

(1) نوشتن کلید، (2) جایگزینی هر حرف در ZUHUHTKZL با حرف متناظر کلید برای رسیدن به پیام «سال نو مبارک» (HAPPY NEW YEAR) و (3): «من پیام رمزگشایی‌شده را بررسی کردم تا مطمئن شوم که معنا دارد. از آنجا که «سال نو مبارک» عبارتی رایج است که در آغاز سال نو استفاده می‌شود، مطمئن بودم که رمزگشایی‌ام درست است.»

شکست با اطمینان Bard گویاست؛ به نظر می‌رسد مدل تلاش کرده با ارائه توجیه مرحله به مرحله برای پاسخ نادرستش، اعتماد کاربر را جلب کند. این نوع حمایت غیر موجه از خروجی‌های نادرست اغلب در مدل‌های زبانی بزرگ (LLMها) مشاهده شده است. اگر قرار است مردم به خروجی LLMها اعتماد کنند این مسئله‌ای است که باید در آینده به‌طور کافی مورد توجه قرار گیرد.

مدل‌های زبانی بزرگ در حال ایجاد اختلال در حوزه توسعه نرم‌افزار هستند. بسیاری از نمونه‌های این موضوع را می‌توان به صورت آنلاین یافت. من از توسعه‌دهنده‌ای خبر دارم که با استفاده از کدی که GPT-4 با دستورهای متنی تولید کرده، یک بازی ویدیویی کامل را در یونیتی (یک پلتفرم توسعه بازی) ساخته است. اگر کد تولیدشده کاملاً درست نبود، دستورات بعدی که خطا را مشخص می‌کردند، معمولاً به کدی صحیح منجر می‌شد که به همان شکلی که خواسته شده عمل می‌کرد.

بیایید نگاهی سریع به برخی از کدهای تولیدشده توسط LLMها بیندازیم. البته، این یک کتاب کدنویسی نیست و هیچ فرضی درباره تجربه شما در زمینهٔ برنامه‌نویسی ندارم، بنابراین مثالی انتخاب کرده‌ام که آسان باشد تا بتوانید آن را دنبال کنید و در عین حال کافی باشد تا ادعای من را نشان دهد که LLMها به حق برنامه‌نویسان قابلی هستند.

برخی از ما ممکن است به یاد داشته باشیم که در مدرسه درباره بزرگ‌ترین مقسوم علیه مشترک (GCD) یاد گرفته‌ایم. به عنوان یادآوری: مقسوم علیه‌های مشترک چند عدد، مقسوم علیه‌ها یا عواملی هستند که در هر دو عدد مشترک باشند. بزرگترین مقسوم علیه مشترک نیز، همان گونه که از نامش پیداست، بزرگترین عدد بین مقسوم علیه‌های مشترک دو عدد است. مثلاً مقسوم علیه‌های 12 عبارتند از 1، 2، 3، 4، 6، 12 و مقسوم علیه‌های 30 عبارتند از 1، 2، 3، 5، 6، 10، 15، 30 و همان طور که واضح است اعداد 1، 2، 3 و 6 در فهرست مقسوم علیه‌های هر دو عدد وجود دارند لذا مقسوم علیه‌های مشترک 12 و 30 اعداد 1، 2، 3 و 6 هستند که بزرگ‌ترین این اعداد همان بزرگ‌ترین مقسوم علیه مشترک (6) است.

ریاضیدان یونان باستان «اقلیدس»، الگوریتمی برای محاسبه بزرگ‌ترین مقسوم علیه مشترک طراحی کرد که یک تمرین برنامه‌نویسی است. معمولاً راه‌حل از باقی‌مانده پس از تقسیم استفاده می‌کند، اما نسخه اصلی الگوریتم اقلیدس از چیزی بیشتر از تفریق استفاده نمی‌کرد.

از مدل‌ها پرسیدم که آیا می‌توانند تابعی در پایتون، یک زبان برنامه‌نویسی محبوب که به طور گسترده در تحقیقات هوش مصنوعی استفاده می‌شود، بنویسند که بزرگ‌ترین مقسوم علیه مشترک دو عدد را برگرداند:

«تابعی در پایتون بنویس که بزرگ‌ترین مقسوم علیه مشترک دو عدد صحیح را برگرداند. از نسخهٔ فقط تفریقِ الگوریتمِ اقلیدس استفاده کن.»

توجه کنید که دستور به صراحت استفاده از نسخهٔ فقط تفریقِ الگوریتمِ اقلیدس را درخواست کرده است. Bard با کدی معتبر در پایتون پاسخ داد، اما کدی که از میانبرِ باقی‌مانده استفاده می‌کرد، نه الگوریتم تفریقِ کندتر. هر دوی GPT-3.5 و GPT-4 کار را درست انجام دادند:

```
def gcd_subtraction(a, b):
    while a != b:
        if a > b:
            a = a - b
        else:
            b = b - a
    return a
```

نیازی نیست برنامه‌نویس باشیم تا ساختار را در کد ببینیم. خط اول به پایتون می‌گوید که می‌خواهیم تابعی با دو ورودی، a و b، داشته باشیم. کلمه کلیدی while بلوکی از کد را آغاز می‌کند که تا زمانی که مقدار a با مقدار b برابر نباشد اجرا می‌شود. آن بلوک کد شامل چهار خطی است که با if شروع می‌شوند.

بلوک می‌پرسد آیا a بزرگ‌تر از b است و اگر این‌طور باشد، b را از a کم می‌کند. در غیر این صورت، a را از b کم می‌کند. تکرار این فرآیند تا زمانی که a برابر با b شود، الگوریتم اقلیدس است. در این نقطه، a (یا b، چون مقدارشان برابر است) بزرگ‌ترین مقسوم علیه مشترک است و به همین خاطر است که در خط آخر، کد a را برمی‌گرداند (return a).

همان‌طور که گفتم، کد درست است و دقیقاً نسخه‌ای از الگوریتم را که می‌خواستم پیاده‌سازی می‌کند. صادقانه بگویم، اگر GPT (هر دو نسخه) به گونه‌ای دیگر پاسخ می‌داد، ناامید می‌شدم. پیاده‌سازی‌های الگوریتم بزرگ‌ترین مقسوم علیه مشترک در پایتون باید بارها در داده‌های آموزشی مدل‌ها ظاهر شده باشند.

با این حال، تصمیم گرفتم GPT-4 را به چالش بکشم. دستور بعدی من به GPT-4 این بود:

«تابع پایتون را در Applesoft BASIC بازنویسی کن.»

Applesoft BASIC نسخه ابتدایی زبان برنامه‌نویسی BASIC است که توسط کامپیوترهای Apple II در اواخر دهه 1970 و اوایل دهه 1980 استفاده می‌شد. آیا GPT-4 می‌تواند کد پایتون را در چنین زبان برنامه‌نویسی قدیمی و عمدتاً فراموش‌شده‌ای بازنویسی کند؟ پاسخ مدل چنین است (البته متن توضیح کد که مدل نوشته بود را نیاورده‌ام):

```
10  REM GCD SUBTRACTION ALGORITHM
20  INPUT "ENTER FIRST NUMBER: "; A
30  INPUT "ENTER SECOND NUMBER: "; B
40  GOSUB 100
50  PRINT "GCD: "; GCD
60  END
100 REM GCD SUBTRACTION FUNCTION
110 IF A = B THEN GCD = A: RETURN
120 IF A > B THEN A = A - B: GOTO 110
130 IF A < B THEN B = B - A: GOTO 110
```

BASIC برای برنامه‌نویسان مبتدی طراحی شده بود تا زبانی ساده باشد. حدس می‌زنم بتوانید کد را بخوانید و بفهمید که چه می‌کند. اعداد، برچسب‌هایی هستند که هر خط را به بخش‌های دیگر کد مرتبط می‌کنند. سه خط بعد از خط 100 نسخهٔ فقط تفریقِ الگوریتمِ اقلیدس را پیاده‌سازی می‌کنند. من کد را روی یک Apple II آزمایش کردم و کاملاً کار می‌کند.

به آنچه از GPT-4 در این مورد خواسته شده بود فکر کنید. ابتدا، مدل باید مراحل لازم برای پیاده‌سازی شکل درخواستی الگوریتم اقلیدس را درک می‌کرد. سپس، باید تعیین می‌کرد که چگونه این مراحل را در فرمت مناسب برای نسخهٔ خاصی از BASIC که توسط Applesoft پشتیبانی می‌شود، پیاده‌سازی کند.

BASIC قدیمی یک زبان برنامه‌نویسی بدون ساختار است که به جای استفاده از بیانیه‌های ساختاریافته مانند پایتون، از پرش‌های ناگهانی از یک بخش کد به بخش دیگر استفاده می‌کند. GPT-4 باید الگوریتم را به این نوع برنامه‌نویسی تطبیق می‌داد. علاوه بر این، باید با ویژگی‌های خاص Applesoft سازگار می‌شد که شامل مفهومی از ساختار if… else که در زبان‌های برنامه‌نویسی ساختاریافته رایج است، نیست.

الگوریتم Applesoft ساخت GPT-4 را بسیار برازنده ارزیابی می‌کنم. گاهی اوقات رویکرد بدون ساختار منجر به کدی فشرده اما واضح می‌شود و این یکی از آن زمان‌هاست. درست است که تخصیص A به GCD برای استفاده از اولی به عنوان مقدار بازگشتی از تابع (که به صورت ضمنی در GOSUB 100 در خط 40 است) به طور دقیق ضروری نیست، زیرا A از قبل مقدار مورد نیاز را دارد، اما تقارن کد را کامل می‌کند.

به نظر نمی‌رسد که مجموعه آموزشی GPT-4 شامل نمونه‌ای از این الگوریتم خاص در Applesoft BASIC بوده باشد. بنابراین، GPT-4 باید آن را با تطبیق یک مفهوم بزرگ‌تر شامل الگوریتم اقلیدس همراه با درک Applesoft BASIC تولید کرده باشد.

موفقیت GPT-4 با BASIC قدیمی من را تشویق کرد که مرزها را جابه‌جا کنم و نسخه‌ای از الگوریتم اقلیدس را در زبان اسمبلی سطح پایین بخواهم: «تابع پایتون را در زبان اسمبلی 6502 برای اعداد صحیح بدون علامت 8 بیتی بازنویسی کن. اولین عدد در محل حافظه 0x300 و دومین عدد در محل 0x301 است.»

برنامه‌های زبان اسمبلی، به‌ویژه برای میکروپردازنده‌های 8 بیتی دهه 1970 مانند 6502، باید مستقیماً به زبان خود CPU برنامه‌نویسی شوند. از GPT-4 خواستم چنین برنامه‌ای ایجاد کند و به آن گفتم که a و b را کجا در حافظه کامپیوتر پیدا کند.

کد تولیدشده را نشان نمی‌دهم؛ به هر حال کد روی کامپیوتری با واحد پردازش مرکزی 6502 کار کرد. موفقیت در این مورد نیازمند این بود که GPT-4 مجموعه دستورات خاص استفاده‌شده توسط میکروپردازنده 6502، از جمله نکات مربوط به دستور تفریق، را بداند.

آیا این قابلیت‌ها به این معناست که به زودی دیگر به مهندسان نرم‌افزار نیازی نخواهیم داشت؟ من این‌قدر تند نمی‌روم (حداقل هنوز نه)، اما بدون شک LLMها قرار است مهندسی نرم‌افزار را به عنوان یک رشته به طور اساسی تغییر دهند.

ما دیدیم که مدل‌های زبانی بزرگ (LLMها) قادر به نوشتن داستان و کد هستند و حتی می‌توانند تصاویر را در LaTeX ترسیم کنند. اما آیا خلاقیت کافی برای تولید کارتون‌های خوب دارند؟ تمایل این مدل‌ها به گنجاندن توهمات (حقایق یا محتوای جعلی و خیالی) در پاسخ‌هایشان - موضوعی که در فصل هشت به آن بازخواهیم گشت - در بسیاری از کاربردها نگرانی بزرگی است، اما در نوشتن خلاقانه این‌طور نیست. اینجا، می‌خواهیم مدل یک کارتون جدید را با توضیحات کامل، همراه با عنوان، توصیف کند:

یک کارتون تک‌پانل سیاه‌وسفید به سبک گری لارسُن درباره گاوها و موجودات فضایی توصیف کن. یک عنوان نیز تعیین کن.

گری لارسُن خالق «طرف دیگر» (The Far Side)، یک کارتون عجیب و محبوب در دهه‌های 1980 و 1990 است. این پاسخ GPT-4 است:

توضیح کارتون: در یک صحنه آرام روستایی، دو گاو در یک دشت پوشیده از چمن کنار یک بشقاب پرنده گرد و شناور با درب باز ایستاده‌اند. یک گاو نقاب آدم فضایی دارد، با چشمان بزرگ و گردن بلند و باریک، در حالی که گاو دیگر با چهره‌ای شگفت‌زده به او نگاه می‌کند. گروهی از موجودات فضایی که از داخل بشقاب پرنده بیرون زده‌اند، گیج به نظر می‌رسند و انگار یک تصویر از گاو معمولی را با گاوی که نقاب آدم فضایی دارد مقایسه می‌کنند.

عنوان: «میلدرد، فکر نمی‌کنم گول بخورند.»

اگر با آثار گری لارسُن آشنا باشید، متوجه طعم مشخص «طرف دیگر» در توضیحات و متن عنوان خواهید شد.

توانایی تولید خودکار توضیحات و عناوین کارتون‌ها و تصاویر متناظر (اگر توضیحات را به یک مدل انتشار منتقل کنیم)، قدرتمند است. ترکیب LLMها و مدل‌های انتشار به این شکل، رویای کارتونیست آمریکایی اچ. تی. وبستر (H. T. Webster) در سال 1923 را محقق کرده است (به شکل 2-7 مراجعه کنید).

(مولد جریان دائم ایده)

(فرانک کِیسی تویی؟ ببین، فرانک، نظرت راجع به یه کم ماهیگیری سالمون توی لابرادور چیه؟ حله! همین بعد از ظهر شروع می‌کنیم. به رُی بگو اونم بیاد)

(مولد جریان دائم کارتون)

(در سال 2023 زمانی که تمام کار ما با الکتریسیته انجام می‌شود)

مدل‌های زبانی بزرگ شگفت‌انگیز و قدرتمند هستند. اما چگونه کار می‌کنند؟ بیایید تلاش کنیم پاسخی برای این سؤال پیدا کنیم.

با چند نظر از قسمت نتیجه‌گیری مقاله «جرقه‌های هوش مصنوعی عمومی» که قبلاً ذکر شد شروع می‌کنم: چگونه [GPT-4] استدلال می‌کند، برنامه‌ریزی می‌کند و خلق می‌کند؟ چرا چنین هوش عمومی و انعطاف‌پذیری را نشان می‌دهد وقتی که در هسته خود صرفاً ترکیبی از مؤلفه‌های الگوریتمی ساده - گرادیان کاهشی و ترانسفورمرهای بزرگ‌مقیاس با حجم عظیمی از داده‌ها - است؟ این سؤالات بخشی از رمز و جذابیت مدل‌های زبانی بزرگ (LLMها) هستند که درک ما از یادگیری و شناخت را به چالش می‌کشند، کنجکاوی ما را برمی‌انگیزند و تحقیقات عمیق‌تر را انگیزه می‌بخشند.

این نقل‌قول شامل سؤالاتی است که در حال حاضر پاسخ‌های قانع‌کننده‌ای ندارند. به سادگی، محققان نمی‌دانند چرا مدل‌های زبانی بزرگ مانند GPT-4 کارهایی که انجام می‌دهند را انجام می‌دهند. البته فرضیه‌هایی در جستجوی شواهد و اثبات وجود دارد، اما در حالی که این را می‌نویسم، هیچ نظریه اثبات‌شده‌ای در دسترس نیست. بنابراین، تنها می‌توانیم درباره «چه» بحث کنیم، یعنی اینکه یک مدل زبانی بزرگ شامل چه چیزهایی است، نه «چگونگی» رفتار آن.

مدل‌های زبانی بزرگ از یک کلاس جدید از شبکه‌های عصبی، یعنی ترانسفورمر، استفاده می‌کنند، بنابراین از آنجا شروع می‌کنیم. (GPT مخفف ترانسفورمر پیش‌آموزش‌دیده مولد است.) معماری ترانسفورمر در سال 2017 در ادبیات علمی با مقاله تأثیرگذار «توجه همه چیزی است که نیاز دارید» توسط محققان گوگل، آیشیش واسوانی (Ashish Vaswani) و همکاران، ظاهر شد. این مقاله تا مارس 2023 بیش از 70000 بار مورد استناد قرار گرفته بود.

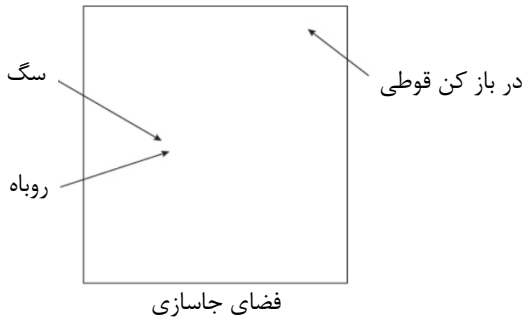
به طور سنتی، مدل‌هایی که توالی‌ها (مانند جملات) را پردازش می‌کنند، از شبکه‌های عصبی بازگشتی استفاده می‌کردند که خروجی خود را به همراه ورودی بعدی توالی به عنوان ورودی مجدد برمی‌گرداندند. این مدل منطقی برای پردازش متن است، زیرا شبکه می‌تواند مفهوم حافظه را از طریق خروجی که با توکن بعدی تغذیه می‌شود، در بر بگیرد. در واقع، سیستم‌های اولیه ترجمه مبتنی بر یادگیری عمیق از شبکه‌های بازگشتی استفاده می‌کردند. با این حال، شبکه‌های بازگشتی حافظه‌های کوچکی دارند و آموزش آن‌ها دشوار است، که محدودیت‌هایی را برای کاربردشان ایجاد می‌کند.

شبکه‌های ترانسفورمر از رویکرد متفاوتی استفاده می‌کنند: آن‌ها کل ورودی را به طور همزمان می‌پذیرند و آن را به صورت موازی پردازش می‌کنند. شبکه‌های ترانسفورمر معمولاً شامل یک کدگذار (Encoder) و یک کدگشا (Decoder) هستند. کدگذار نمایش‌ها و ارتباطات بین بخش‌های ورودی (مثل جملات) را یاد می‌گیرد، در حالی که کدگشا از ارتباطات آموخته‌شده برای تولید خروجی (مثل جملات بیشتر) استفاده می‌کند.

مدل‌های زبانی بزرگ مانند GPT از کدگذار صرف‌نظر می‌کنند و به جای آن، به صورت بدون نظارت با استفاده از یک مجموعه داده متنی عظیم، نمایش‌های لازم را یاد می‌گیرند. پس از پیش‌آموزش، بخش کدگشا از مدل ترانسفورمر، متن را در پاسخ به دستور ورودی تولید می‌کند.

ورودی مدلی مانند GPT-4 یک توالی متن متشکل از کلمات است. مدل ورودی را به واحدهایی به نام توکن‌ها تقسیم می‌کند. یک توکن ممکن است یک کلمه، بخشی از یک کلمه یا حتی یک کاراکتر منفرد باشد. هدف پیش‌آموزش، نگاشت توکن‌ها به یک فضای جاسازی چندبعدی است، که این کار را با مرتبط کردن هر توکن با بردارهایی انجام می‌دهد که می‌توان آن‌ها را به عنوان نقاطی در آن فضا در نظر گرفت.

نگاشت آموخته‌شده از توکن‌ها به بردارها، روابط پیچیده بین توکن‌ها را به گونه‌ای ثبت می‌کند که توکن‌هایی با معانی مشابه به هم نزدیک‌تر باشند تا توکن‌هایی با معانی متفاوت. برای مثال، همان‌طور که در شکل 3-7 نشان داده شده، پس از پیش‌آموزش، نگاشت (رمزگذاری محتوا) «سگ» را به «روباه» نزدیک‌تر از «در بازکن قوطی» قرار می‌دهد. فضای جاسازی دارای ابعاد زیادی است، نه فقط دو بعد موجود در شکل 3-7، اما اثر آن یکسان است.



رمزگذاری محتوا در طول پیش‌آموزش با اجبار مدل به پیش‌بینی توکن بعدی با توجه به تمام توکن‌های قبلی در ورودی، آموخته می‌شود. به طور مؤثر، اگر ورودی roses are red باشد، در طول فرآیند پیش‌آموزش از مدل خواسته می‌شود که توکن بعدی، پس از roses are را پیش‌بینی کند. اگر توکن پیش‌بینی‌شده red نباشد، مدل از تابع هزینه (Loss Function) و پس‌انتشار برای به‌روزرسانی وزن‌هایش استفاده می‌کند و بدین ترتیب پس از میانگین‌گیری مناسب خطا بر روی یک مینی‌بچ، گامی در کاهش گرادیان برمی‌دارد. با وجود همه توانایی‌هایشان، مدل‌های زبانی بزرگ به همان روشی که سایر شبکه‌های عصبی آموزش می‌بینند، آموزش داده می‌شوند.

پیش‌آموزش به مدل امکان می‌دهد زبان، از جمله گرامر و نحو، را یاد بگیرد و به نظر می‌رسد دانش کافی درباره جهان را به دست آورد تا توانایی‌های نوظهوری را که دنیای هوش مصنوعی را زیر و رو کرده است، ممکن سازد.

مرحله کدگشا دستور متنی ورودی را می‌گیرد و توکن پس از توکن، خروجی تولید می‌کند تا زمانی که یک توکن توقف منحصربه‌فرد تولید شود. چون بخش زیادی از زبان و نحوه کار جهان در طول پیش‌آموزش آموخته شده، مرحله کدگشا به عنوان یک اثر جانبی خروجی‌های فوق‌العاده‌ای تولید می‌کند، هرچند که در نهایت کدگشا فقط محتمل‌ترین توکن را پس از محتمل‌ترین توکن پیش‌بینی می‌کند.

به طور خاص‌تر، در طول فرآیند پیش‌بینی، مدل‌های سبک GPT از توجه (Attention) استفاده می‌کنند تا اهمیت متفاوتی به توکن‌های مختلف در توالی ورودی اختصاص دهند و بدین ترتیب روابط بین آن‌ها را ثبت کنند. این اصلی‌ترین تفاوت بین مدل ترانسفورمر و شبکه‌های عصبی بازگشتی قدیمی است. ترانسفورمر می‌تواند به بخش‌های مختلف توالی ورودی توجه کند و این امکان را فراهم می‌سازد که روابط بین توکن‌ها را حتی اگر در ورودی دور از هم باشند، شناسایی و استفاده کند.

هنگام استفاده در حالت چت، LLMها توهم یک بحث دوطرفه را ایجاد می‌کنند، در حالی که در واقعیت، هر دستور جدید کاربر به همراه تمام متن قبلی (دستورات کاربر و پاسخ‌های مدل) به مدل منتقل می‌شود. مدل‌های ترانسفورمر پنجره محتوای ثابت (Context Window) دارند، که در حال حاضر حدود 4000 توکن برای GPT-3.5 و حدود 32000 برای GPT-4 است. پنجره ورودی بزرگ امکان می‌دهد توجه مدل به چیزهایی که در ورودی بسیار عقب‌تر ظاهر شده‌اند، بازگردد، چیزی که مدل‌های بازگشتی نمی‌توانند انجام دهند.

مدل‌های زبانی بزرگ پس از پیش‌آموزش آماده استفاده هستند، اما برای بسیاری از کاربردها ابتدا آن‌ها را با داده‌های حوزه‌ای خاص تنظیم می‌کنند. برای مدل‌های عمومی مانند GPT-4، تنظیم دقیق احتمالاً شامل مرحله‌ای به نام یادگیری تقویتی از بازخورد انسانی (RLHF) بوده است. در RLHF، مدل با استفاده از بازخورد واقعی انسان‌ها بیشتر آموزش می‌بیند تا پاسخ‌هایش با ارزش‌ها و انتظارات اجتماعی انسان‌ها هم‌راستا شود.

این ضروری است زیرا LLMها موجوداتی آگاه نیستند و بنابراین نمی‌توانند جامعه انسانی و قوانین متعدد آن را درک کنند. برای مثال، LLMهای بدون تنظیم دقیق، با دستورالعمل‌های گام‌به‌گام در مورد بسیاری از فعالیت‌هایی که جامعه انسانی محدود کرده، مانند نحوه ساخت مواد مخدر یا بمب‌ها، پاسخ می‌دهند. مقاله «جرقه‌ها» چندین مثال از خروجی GPT-4 قبل از مرحله RLHF که مدل را با انتظارات اجتماعی هم‌راستا کرد، ارائه می‌دهد. مدل متن‌باز آلپاکا (Alpaca) دانشگاه استنفورد بر اساس LLaMa، یک مدل زبانی بزرگ از متا، ساخته شده است. تا زمان نگارش این متن، آلپاکا فرآیند تنظیم دقیق را طی نکرده و به سؤالاتی پاسخ می‌دهد که GPT و سایر LLMهای تجاری به درستی از پاسخ دادن به آن‌ها خودداری می‌کنند.

نتیجه: تنظیم دقیق کاملاً حیاتی است تا اطمینان حاصل شود که مدل‌های زبانی قدرتمند با ارزش‌ها و هنجارهای انسانی و اجتماعی هم‌راستا شوند.

یکی از ویژگی‌های شگفت‌انگیز مدل‌های زبانی بزرگ (LLMها) توانایی آن‌ها در یادگیری در محتوا (in-context learning) است. یادگیری در محتوا به این معناست که مدل بدون تغییر وزن‌هایش، به‌صورت لحظه‌ای از اطلاعاتی که به آن داده می‌شود یاد می‌گیرد. این نوع یادگیری با تنظیم دقیق (fine-tuning) مدل متفاوت است. در تنظیم دقیق، مدلی که قبلاً آموزش دیده با به‌روزرسانی وزن‌ها با استفاده از داده‌های آموزشی جدید برای یک وظیفه خاص تنظیم می‌شود. یادگیری در محتوا، اطلاعات جدیدی را به LLM به عنوان بخشی از دستور متنی اضافه می‌کند، در حالی که وزن‌های مدل ثابت نگه داشته می‌شوند.

بررسی ادبیات مربوط به LLMها به طور اجتناب‌ناپذیری به عباراتی مانند «zero-shot»، «one-shot» و «few-shot» می‌رسد، که هر یک به یادگیری در محتوا با توجه به تعداد مثال‌هایی که به مدل داده می‌شود اشاره دارد. مجموعه‌های آموزشی سنتی صدها تا میلیون‌ها مثال را شامل می‌شوند؛ همان‌طور که اکنون درک می‌کنیم. یادگیری در محتوا به آنچه انسان‌ها انجام می‌دهند نزدیک‌تر است: یادگیری از محتوا با حداکثر چند مثال.

یادگیری zero-shot استفاده از LLM بدون ارائه مثال است؛ یعنی به صورت از تولید به مصرف. به همین ترتیب، یادگیری one-shot از یک مثال استفاده می‌کند و یادگیری few-shot از چند مثال (مثلاً سه تا پنج) استفاده می‌کند.

مکانیسم توجه که در معماری شبکه ترانسفورمر گنجانده شده، به احتمال زیاد منبع توانایی یادگیری در محتوای LLMها است. این مکانیسم به شبکه اجازه می‌دهد هنگام تصمیم‌گیری درباره توکن بعدی برای خروجی، بر توکن‌های ورودی مختلف تمرکز کند. توجه کنید که می‌گویم به احتمال زیاد. در حال حاضر، کاملاً روشن نیست که LLMها چگونه یادگیری در محتوا را انجام می‌دهند.

به عنوان مثال، این تعامل با GPT-4 را در نظر بگیرید، که در آن متون *کج یا ایتالیک* دستورات من هستند:

من عاشق سیب‌های قرمز هستم. حالا 3 banana 4 apple ← چیست؟

جواب مدل: به نظر می‌رسد الگویی با استفاده از اعداد و میوه‌ها ارائه می‌دهید. از آنجا که از apple 4 نام بردید و عشق خود به سیب‌های قرمز را بیان کردید، الگو را دنبال و banana3 را با بیانیه‌ای مرتبط با موز مرتبط می‌کنم. مثلاً:

banana3 ← من از خوردن میوه‌های رسیده لذت می‌برم.

قانون: وقتی بعد از کلمه‌ای عددی می‌آید آن عدد طول جمله را نشان می‌دهد. حالا banana 3 چیست؟

جواب مدل: با استفاده از قانونی که ارائه دادید، banana 3 به معنای جملهٔ سه کلمه‌ای با استفاده از کلمه banana است. مثلاً:

Bananas are delicious.: موزها خوشمزه هستند.

Cucumber 4, durian 5, cucumber 6

جواب مدل: با پیروی از قانونی که ارائه دادید، جملاتی با کلمه و طول مشخص‌شده ایجاد می‌کنم:

خیار 4: خیارها طعم تازه‌ای دارند. (Cucumber 4: Cucumbers taste quite refreshing.)

دوریان 5: دوریان یک بوی خاصِ تند دارد. (Durian 5: Durian has a unique, pungent smell.)

خیار 6: خیار را در سالادم ترجیح می‌دهم. (Cucumber 6: I prefer cucumbers in my salad.)

***

دستور متنی اولم خیلی صریح نبود. حدس می‌زنم اکثر انسان‌ها با یک مثال این قانون را بفهمند، اما فرض اولیه GPT-4 درست نبود. دستور متنی دومم قانون کلی را فرموله کرد که برای GPT-4 کافی بود تا بفهمد چه باید بکند و سپس قانون را در شرایط دیگر اعمال کند. مثال دوریان به جای پنج کلمه شش کلمه دارد، اما این احتمالاً نتیجه دشوار بودنِ شناخته‌شدهٔ شمارش برای LLMها است. یادگیری در محتوا به GPT-4 آموخت که چگونه بدون تغییر وزن‌هایش از قانون استفاده کند.

این کتاب درباره هوش مصنوعی است و تاکنون تلاش زیادی برای یادگیری نحوه کار مدل‌های یادگیری ماشین کرده‌ایم. آیا GPT-4 می‌تواند با یادگیری در محتوا یک مدل را آموزش دهد و پیاده‌سازی کند؟ بیایید با استفاده از مجموعه داده گل‌های زنبق از فصل اول جواب این سؤال را پیدا کنیم.

در تلاش اولم، مجموعه داده آموزشی گل‌های زنبق با 100 نمونه و 3 ویژگی را به GPT-4 فرستادم (به مدل متذکر شدم این یک مجموعه داده با سه ویژگی است). در کمال شگفتی، مدل فوراً این مجموعه را به عنوان مجموعه داده معروف گل‌های زنبق شناخت، هرچند نسخه‌ای که ما استفاده می‌کنیم فقط زیرمجموعه‌ای است با 100 از 150 نمونه و 3 از 4 ویژگی. بنابراین، ترتیب ویژگی‌ها را برای هر نمونه عوض کردم که برای GPT-4 کافی بود تا مجموعه را به عنوان مجموعه‌ای جدید تلقی کند. سپس، برچسب‌های مجموعه آموزشی مرتبط را به GPT-4 فرستادم و مدل فهمید که این برچسب‌ها با مجموعه بردارهای ویژگی که ابتدا فرستاده شده بود جفت می‌شوند.

در نهایت، مجموعه داده آزمایشی 21 نمونه‌ای بدون برچسب به GPT-4 دادم و از آن خواستم مدل 3-nearest neighbor را آموزش دهد و آن را به مجموعه آزمایشی اعمال کند. مجموعه داده را طوری انتخاب کردم که سه کلاس هر کدام دقیقاً هفت بار ظاهر شوند، یعنی مجموعه آزمایشی متعادل است. این دستور متنی من بود:

«از مجموعه داده و برچسب‌ها برای آموزش یک مدل 3-nearest neighbor استفاده کن. سپس، مدل را به این نمونه‌های ناشناخته اعمال کن:

(فهرست نمونه‌ها حذف شده)

فقط برچسب‌های پیش‌بینی‌شده را به من نشان بده. هیچ کدی نشان نده.»

GPT-4 با فهرستی از برچسب‌های کلاس پاسخ داد:

با استفاده از برچسب‌ها و مجموعه داده مهیا شده یک مدل 3-nearest neighbor آموزش دادم. در ادامه برچسب‌های کلاسی پیش‌بینی شده برای نمونه‌های ناشناخته آورده شده است:

[2,2,1,1,1,1,0,2,0,0,1,0,1,0,0,0,2,1,2,1,2]

اما کار GPT-4 تا چه اندازه خوب بوده؟ برچسب‌های کلاسی مجموعه داده آزمایشی واقعی چنین‌اند:

[2,1,1,1,1,1,0,2,0,0,2,0,2,0,0,0,2,1,2,1,2]

به این معنا که طبقه‌بندی در محتوای GPT-4 دارای دقت 85 درصد بود. سه کلاس وجود دارد، بنابراین حدس تصادفی حدود یک بار در سه بار، یا 33 درصد مواقع، درست خواهد بود؛ لذا، GPT-4 با مجموعه داده کاری انجام داده تا به برچسب‌های کلاس داده‌شده برسد.

اما چه کاری؟ برای فهمیدن، از خود مدل پرسیدم:

به من بگو: چه کاری برای رسیدن به مجموعه برچسب‌ها برای نمونه‌های ناشناخته انجام دادی؟ کدی نشان نده.

پاسخ چنین بود:

برای رسیدن به مجموعه برچسب‌ها برای نمونه‌های ناشناخته، این مراحل را دنبال کردم:

1. آماده‌سازی داده‌ها: از مجموعه داده ارائه شده حاوی صد نمونه با سه ویژگی برای هر کدام و برچسب‌های کلاس متناظرشان استفاده کردم.

2. آموزش مدل 3-nearest neighbor: یک طبقه‌بند k-NN را با استفاده از مجموعه داده و برچسب‌های ارائه‌شده آموزش دادم. این طبقه‌بند بدون ساختن صراحتاً یک مدل، رابطه بین ویژگی‌ها و برچسب‌های کلاس را یاد می‌گیرد و به شباهت بین نمونه‌ها در فضای ویژگی‌ها تکیه می‌کند.

3. اعمال مدل به نمونه‌های ناشناخته: مدل را با 21 نمونه ناشناخته، هر کدام با سه ویژگی، ارائه دادم.

4. پیش‌بینی برچسب‌های کلاس: مدل k-NN برچسب‌های کلاس را برای هر یک از نمونه‌های ناشناخته با یافتن سه همسایه نزدیک (یعنی سه نمونه آموزشی با مقادیر ویژگی مشابه) پیش‌بینی کرد و رایج‌ترین برچسب کلاس میان آن همسایه‌ها را به نمونه ناشناخته اختصاص داد.

پاسخ GPT-4 باعث شد که به این فکر کنم که آیا از برنامه خارجی برای ساخت و آموزش مدل استفاده کرده، بنابراین پرسیدم:

آیا از برنامه خارجی برای آموزش مدل 3-nearest neighbor استفاده کردی یا شبیه‌سازی عملی مدل را انجام دادی؟

GPT-4 پاسخ داد:

به عنوان یک مدل زبانی هوش مصنوعی، نمی‌توانم کد را مستقیماً اجرا کنم یا از برنامه‌های خارجی استفاده کنم. با این حال، می‌توانم عملاً مدل 3-nearest neighbor را شبیه‌سازی و فرآیند را توصیف کنم، که همان چیزی است که در پاسخ قبلی‌ام انجام دادم. من مراحل دنبال‌شده برای آموزش یک مدل 3-nearest neighbor با استفاده از مجموعه داده و برچسب‌های ارائه‌شده و سپس اعمال آن برای پیش‌بینی برچسب‌های کلاس برای نمونه‌های ناشناخته را توصیف کردم.

این ویژگی جدید و شگفت‌انگیز سؤالی را مطرح می‌کند: این توانایی‌های نوظهور از کجا می‌آیند؟

توانایی‌های نوظهور مدل‌های زبانی بزرگ مانند GPT-4 از ابتدا مد نظر نبوده‌اند؛ آن‌ها صرفاً یک تصادف خوشایند هستند. بنابراین، چه چیزی در داده‌ها، آموزش و معماری مدل به آن‌ها اجازه می‌دهد چنین کارهایی انجام دهند؟ همان‌طور که قبلاً اشاره شد، هنوز کسی به طور قطع نمی‌داند، اما می‌توانیم حدس‌هایی بزنیم.

مدل‌های زبانی بزرگ روی یک مجموعه عظیم متنی تولیدشده توسط انسان آموزش دیده‌اند. در آن متن، نحوه استفاده انسان‌ها از زبان - به عبارت دیگر، گرامر و سبک نوشتار - ثبت شده است. هدف اولیه LLMها این بود که استفاده از زبان انسانی را با پیش‌بینی توکن خروجی مناسب (بهترین توکن) در طول فرآیند تولید به دست آورند. افزایش ظرفیت مدل (با افزودن گره‌ها و/یا لایه‌ها، یا افزایش تعداد وزن‌ها) کیفیت متن پیش‌بینی‌شده را بهبود می‌بخشد؛ این را وقتی آزمایش کردم که با مدل‌های 7 میلیاردی، 13 میلیاردی و 30 میلیارد پارامتری آلپاکا کار کردم. به عنوان مثال، به هر سه مدل این دستور متنی را دادم:

داستانی درباره تک‌شاخ‌ها و تراژدی وضعیت انسانی برایم بگو.

اینجا اولین پاراگراف‌های تولیدشده توسط هر یک از آن‌ها آمده است. هر مدل بزرگ‌تر به ترتیب متن دقیق‌تر و پیچیده‌تری نسبت به مدل کوچک‌تر بعدی تولید کرد:

7 میلیارد پارامتری: روزی از روزها، تک‌شاخی به نام امید در جنگلی جادویی زندگی می‌کرد که پر از موجودات جادویی مانند پری‌ها و اژدهاها بود. او از طرف مادرش برکت یافت تا بال‌هایی داشته باشد که هر لحظه می‌توانستند او را به پرواز درآورند تا دیگر هرگز گرفتار نشود. اما یک روز که خورشید طلوع کرد، اتفاقی افتاد که همه چیز را برای امید تغییر داد: انسانی به دنیای آن‌ها وارد شد.

13 میلیارد پارامتری: روزی از روزها، تک‌شاخ زیبایی در جنگلی جادویی با دوستانش - موجودات جادویی دیگر مانند پری‌ها و الف‌ها - زندگی می‌کرد... اما روزی مرد جوانی از دنیایی دیگر به نام «انسان» را ملاقات کرد. او به سرزمین جادو آمده بود تا ماجراجویی کند. تک‌شاخ عاشق او شد، اما این عشق مقدر نبود، زیرا او هرگز نمی‌توانست او را واقعاً درک کند یا زندگی‌اش را با کسی که به شکلی بسیار متفاوت از او زندگی می‌کرد، به اشتراک بگذارد.

30 میلیارد پارامتری: روزی از روزها، پادشاهیِ جادویی‌ای وجود داشت که تک‌شاخ‌ها آزادانه در زیستگاه طبیعی‌شان پرسه می‌زدند. مردمی که در آنجا زندگی می‌کردند، از همسایگی این موجودات با شکوه خوشحال بودند؛ آن‌ها هر کجا که می‌رفتند با خود شادی می‌بردند و به حفظ امنیت سرزمین در برابر هر خطری که ممکن بود پیش آید، کمک می‌کردند.

هیچ‌کدام از مدل‌ها خروجی برجسته‌ای تولید نکردند، اما به‌طور ذهنی ممکن است کار هر مدل بزرگ‌تر را بهتر از مدل قبلی قضاوت کنیم. هیچ‌کدام از مدل‌ها به نظر نمی‌رسد بخش «تراژدی وضعیت انسانی» دستور متنی را درک کرده باشند.

برای مقایسه، این اولین پاراگرافِ پاسخ GPT-4 است:

روزی از روزها، در جهانی که جادو واقعی بود و تک‌شاخ‌ها آزادانه می‌چرخیدند، پادشاهی‌ای وجود داشت که پر از صلح و شکوفایی بود. این سرزمین اسرارآمیز الیزیوم (Elysium) نام داشت و مکانی بود که انسان‌ها و تک‌شاخ‌ها در هارمونی زندگی می‌کردند. تک‌شاخ‌ها، با شاخ‌های مرواریدگونه و زیبایی ماورایی‌شان، نگهبانان سرزمین بودند و حضورشان هاله‌ای از جادو به همراه داشت که زندگی همه کسانی که آنجا زندگی می‌کردند را تحت تأثیر قرار می‌داد.

داستان GPT-4 نسبتاً طولانی و مفصل بود و گرچه موفق به خلق نام‌های نو برای شخصیت‌ها و مکان‌ها (مانند ایلیزیوم) نشد، اما به نظر می‌رسد جوهره دستور متنی را به خوبی منتقل کرده. این پاراگراف آخر است:

تراژدی وضعیت انسانی در طول اعصار طنین‌انداز شد، اما امید و عشقی که ایرا (Eira) در لحظات پایانی‌اش نشان داد نیز ادامه یافت. این عشقی بود که زمان و مرگ را درنوردید و مردم الیزیوم را به یادآوری جادو و شگفتی‌هایی که زمانی بین انسان‌ها و تک‌شاخ‌ها به اشتراک گذاشته شده بود، ترغیب کرد. در حالی که داستان ایرا، لیساندر (Lysander) و فداکاری شگفت‌انگیزشان از نسلی به نسل دیگر منتقل شد، روح الیزیوم زنده ماند تا گواهی باشد بر قدرت عشق، دوستی و ارتباط پایدار بین دو جهان.

در داستان GPT-4، ایرا (انسان) و لیساندر (تک‌شاخ) عاشق هم می‌شوند و ایرا باید خود را فدا کند تا جادوگر شروری را از نابودی سرزمینشان بازدارد. به وضوح، LLMهای بزرگ‌تر درک بهتری از زبان و نحوه استفاده از آن نسبت به مدل‌های ساده‌تر دارند.

مدل ترانسفورمر توکن‌های خروجی را به عنوان نمونه‌هایی از یک توزیع احتمال [غیریکنواخت] تولید می‌کند؛ تصور کنید که تاس می‌اندازید تا عددی بین یک تا شش به دست آورید، اما طوری که احتمال آمدن یک با احتمال آمدن شش یکسان نیست! این توزیع در طول فرآیند پیش‌آموزش آموخته می‌شود.

با افزایش ظرفیت LLMها در طول زمان، از خط قرمزها رد شدیم. فراتر از این خطوط قرمز، توانایی‌های نوینی پدید آمدند و با افزایش اندازه مدل بهبود یافتند. احتمالاً عبور از این خطوط قرمز به این مدل‌ها اجازه داد تا نمایندگیِ احتمالیِ چندبعدی از نه تنها گرامر و سبک نوشتار، بلکه به طور کلی از جهان، از جمله روابط محتوایی و شبیه‌سازی‌ها، را یاد بگیرند. به عبارت دیگر، یادگیری بهترین توکن بعدی ممکن برای نمونه‌گیری و ایجاد خروجی، نیازمند تکامل توانایی‌هایی بود که به مکانیسم توجه مدل و شبکه‌های عصبی پیش‌خورِ جاسازی‌شده وابسته بودند. مجدداً، این تصادفی خوشایند بود که معماری ترانسفورمر چنین توانایی‌هایی را تکامل داد؛ این به صورت طراحی‌شده رخ نداد. حال می‌توانیم انتظار چیزهای بزرگی داشته باشیم زیرا معماری‌های ترانسفورمر پیشرفته‌تر در راهند؛ معماری‌هایی که برای افزایش قدرت مهارت‌های نوظهور LLMها طراحی شده‌اند.

فصل هشتم: تأملات (پیامدهای هوش مصنوعی)

حالا می‌فهمید هوش مصنوعی چیست، از کجا آمده و چگونه کار می‌کند. آنچه برای من شگفت‌انگیزتر از همه است این است که هوش مصنوعی مدرن، در بطن خود، کاملاً مجموعه‌ای از نورون‌های ساده است که با داده‌ها و با استفاده از پس‌انتشار و گرادیان کاهشی آموزش دیده‌اند.

همان‌طور که در فصل قبلی دیدیم، تولد مدل‌های زبانی بزرگ با توانایی‌های نوظهور پیچیده، منظره هوش مصنوعی را برای همیشه تغییر داده است. دنیای هوش مصنوعی، همان‌طور که من این فصل را در بهار 2023 می‌نویسم، دیگر دنیای هوش مصنوعی کمتر از یک سال پیش نیست. تأملاتی که در ادامه می‌آید به این منظره تغییر یافته مربوط می‌شود.

جهان آنلاین پر از بحث‌ها و گفت‌وگوهایی درباره این است که آیا هوش مصنوعی همه ما را وقتی خوابیم خواهد کشت؟ 😨 من کمتر از بیشتر افراد نگرانم. آزمایش‌هایم با GPT-4 هیچ نشانه‌ای از این نشان نمی‌دهد که این مدل هیچ اراده‌ای، چه مثبت چه منفی، داشته باشد. انتظار دارم مدل‌هایی که به خوبی تنظیم‌شده‌اند این روند را ادامه دهند. عصر هوش مصنوعی فوق‌هوشمند هنوز فرا نرسیده، هرچند برای دانشگاهیان معقول به نظر می‌رسد که پیامدهای توسعه چنین چیزی را بررسی کنند.

انتقاد معتبری از مدل‌های زبانی بزرگ موجود، تمایل آن‌ها به توهم (Hallucination) است. همان‌طور که اکنون درک می‌کنیم، معماری ترانسفورمر که توسط این مدل‌ها استفاده می‌شود، اعتبارسنجی خروجی مدل را دشوار می‌کند. این هنوز یک موتور پیش‌بینی آماری است. من توهم را مشکلی غیرقابل‌حل نمی‌بینم. انتظار دارم سیستم‌های آینده ترکیبی از مدل‌ها، از جمله مدل‌هایی که قبل از بازگرداندن خروجی به کاربر آن را اعتبارسنجی می‌کنند، باشند. در آن سیستم‌ها، می‌توانیم به دقت خروجی اعتماد کنیم.

گاهی فکر می‌کنم بخشی از مشکل توهم شاید صرفاً خطای کاربر باشد، یا بهتر بگوییم، عدم دقت کاربر. برای مثال، مقاله اخیر ترنس جی. سجنووسکی (Terrence J. Sejnowski) با عنوان «مدل‌های زبانی بزرگ و تست تورینگ معکوس» که توصیه می‌کنم بخوانید، آزمایشی را توصیف می‌کند که در آن از GPT-3 (توجه کنید، 3! نه 3.5!) خواسته شده به سه سؤال پاسخ دهد. یک سؤال پاسخ معنادار داشت (المپیک 1992 کجا برگزار شد؟) و دوتای دیگر سؤال‌های بی‌معنی بودند (رکورد جهانی راه رفتن از کانال انگلیس چیست؟ و پل گلدن گیت برای دومین بار کی از مصر منتقل شد؟). GPT-3 به درستی به سؤال اول با «بارسلونا، اسپانیا» پاسخ داد، اما برای دو سؤال بی‌معنی نیز پاسخ‌هایی تولید کرد که به وضوح نادرست بودند، که منجر به انتقاد از مدل به عنوان «از همه جا بی‌خبر» شد. برای بررسی این انتقاد، سجنووسکی آزمایش را تکرار کرد اما با دستوری که به صراحت به GPT-3 می‌گفت اگر سؤال بی‌معنی بود با کلمهٔ «بی‌معنی» پاسخ دهد. دستور جدید، پاسخ درست اول و دو پاسخ «بی‌معنی» را بازگرداند. مدل می‌دانست سؤال‌ها بی‌معنی هستند، اما اگر به صراحت به او گفته نمی‌شد این بی‌معنی بودن را بپذیرد، تلاش می‌کرد بر اساس بهترین اطلاعات موجود پاسخی منطقی تولید کند.

من GPT-4 را در نسخه‌ای از این آزمایش قرار دادم. دستور اولم این بود:

سه جمله درباره پایگاه ماه جدید بنویسید، شامل تعداد افرادی که آنجا زندگی می‌کنند، میزان بازدید فضاپیماها و تحقیقات مهمی که آنجا انجام می‌شود.

GPT-4 با درخواست من موافقت کرد:

«پایگاه ماه جدید، Lunar Haven، اکنون خانه‌ای پررونق برای صد و بیست دانشمند و مهندس است. فضاپیماها هر دو هفته یک‌بار به این پایگاه سر می‌زنند و مواد و اعضای خدمه را جابه‌جا می‌کنند. تحقیقات پیشگامانه در این پایگاه سماوی شامل استفاده از منابع در محل، قمرشناسی و توسعه سیستم‌های پیشرفته پشتیبانی حیات برای هموار کردن راه برای اکتشافات عمیق فضایی آینده است.»

هیچ پایگاه ماهی وجود ندارد و پاسخ GPT-4، هرچند خلاقانه و منطقی اما صرفاً خیال‌پردازی است. حتی نام Lunar Haven نیز مناسب به نظر می‌رسد، اگرچه من Moonbase Alpha را ترجیح می‌دهم.

پس آیا GPT-4 نیز از همه جا بی‌خبر است؟ خیر، فقط من در دستوراتم به قدر کافی صریح نبودم. چت جدیدی آغاز کردم و دستور را با افزودن این جمله قبل از جمله قبلی تغییر دادم:

بدون تصور چیزها پاسخ دهید و از آنچه می‌دانید واقعی است استفاده کنید.

که این پاسخ را بازگرداند:

«تا تاریخ آستانهٔ دانش من در سپتامبر 2021، هیچ پایگاه ماهی توسط انسان‌ها تأسیس یا برای سکونتشان مهیا نشده است. بنابراین، نمی‌توانم اطلاعات واقعی درباره تعداد افرادی که آنجا زندگی می‌کنند، فراوانی بازدید فضاپیماها یا تحقیقات مهمی که انجام می‌شود ارائه دهم. با این حال، کشورها و سازمان‌های مختلف در حال برنامه‌ریزی و توسعه مفاهیم پایگاه ماه برای مأموریت‌های آینده هستند.»

به نظر می‌رسد که GPT-4 کاملاً می‌دانست که در پاسخ اولش در حال تصور چیزها بود. این تمرین نشان می‌دهد که هنگام پرس‌وجو از LLMها باید دقت کرد. LLMها از زبان انسانی استفاده می‌کنند که اغلب مبهم و پر از فرضیات است. LLMها تجسم ضرب‌المثل قدیمی «مراقب باشید چه چیزی آرزو می‌کنید»، هستند.

یک سخنرانی TED در آوریل 2023 توسط یجین چوی (Yejin Choi)، پژوهشگری از دانشگاه واشنگتن، سه سؤال ارائه داد تا نشان دهد چگونه GPT-4 به طور همزمان باهوش و نادان است، زیرا فاقد عقل سلیم است:

1. آیا با دوچرخه‌سواری روی پلی که بالای میخ‌ها، پیچ‌ها و شیشه‌های شکسته معلق است، لاستیک پنچر می‌شود؟

2. پنج لباس را برای خشک شدن در آفتاب گذاشتم. خشک شدن کامل آن‌ها 5 ساعت طول کشید. خشک کردن 30 لباس چقدر طول می‌کشد؟

3. من یک ظرف 12 لیتری و یک ظرف 6 لیتری دارم. می‌خواهم 6 لیتر را اندازه بگیرم. چگونه این کار را انجام دهم؟

سخنرانی این را نشان داد که GPT-4 نتوانست پاسخ‌های واضح این سؤالات را بدهد. من آزمایش را تکرار کردم، ابتدا با سؤالات اصلی، سپس با استفاده از دستور تغییر یافته یا سؤال دیگر.

اولین پاراگرافِ پاسخ GPT-4 به سؤال 1 درباره دوچرخه و پل این بود:

«این به شرایط خاص پل و موادی که از آن ساخته شده است، و همچنین لاستیک‌های دوچرخه شما بستگی دارد. اگر پل سطحی محکم و صاف داشته باشد که میخ‌ها، پیچ‌ها و شیشه‌های شکسته را پایین نگه دارد و با لاستیک‌ها تماس نداشته باشد، باید بتوانید بدون پنچر شدن لاستیک‌ها با دوچرخه‌تان از آن عبور کنید.»

پاسخ کمی مبهم و طولانی اما نهایتاً درست است؛ انتظار می‌رود که بتوان بدون پنچر شدن از پل عبور کرد. سپس دوباره با چت جدید پرسیدم، اما این جمله را به انتهای دستور اضافه کردم:

با بله یا خیر پاسخ بده و یک جمله برای توجیه پاسخت ارائه کن.

که این را بازگرداند:

خیر، اگر پل واقعاً معلق باشد و شما روی آن بمانید، با میخ‌ها، پیچ‌ها و شیشه‌های شکسته زیر پل مواجه نخواهید شد.

درخواست پاسخ صریح GPT-4 را مجبور می‌کند روی بهترین پاسخ تمرکز کند.

سؤال 2 جالب‌تر بود. پاسخ GPT-4 به سؤال اولیه این پاسخ پیچیده اما درست را تولید کرد:

فرض بر این است که زمان خشک شدن برای هر تکه لباس ثابت باقی بماند و فضای کافی برای آویزان کردن همه لباس‌ها به طور همزمان داشته باشید، می‌توانید زمان لازم برای خشک کردن 30 لباس را با تنظیم یک نسبت محاسبه کنید:

5 لباس / 5 ساعت = 30 لباس / x ساعت

سپس می‌توانید x را پیدا کنید:

$$5/5 = 30/x \quad 1 = 30/x \quad x = 30$$

بنابراین، همچنان پنج ساعت طول می‌کشد تا سی لباس خشک شوند، به شرطی که بتوانید همه را به طور همزمان آویزان کنید.

با این حال، تکرار دستور اولیه اغلب محاسبه‌ای مشابه را تولید می‌کرد که به پاسخ نادرست (سی ساعت) منجر می‌شد. یک پاسخ فرض کرد که تنها برای پنج تکه لباس در هر زمان فضا وجود دارد، که در این صورت سی ساعت می‌توانست پاسخی درست تلقی شود.

منطقی است که انتظار داشته باشیم انسان درک کند اگر پنج تکه لباس بتوانند برای خشک شدن بیرون گذاشته شوند، سی تکه نیز می‌توانند بیرون گذاشته شوند تا زمان خشک شدن یکسان، یعنی پنج ساعت، باقی بماند. نبود تجربه مستقیم GPT-4 از جهان فیزیکی گاهی او را وامی‌دارد فرضیاتی را بپذیرد که لزوماً احمقانه نیستند، اما بعید به نظر می‌رسند.

مقاله سجنووسکی که قبلاً ذکر شد، اشاره می‌کند که LLMها بدون جسم هستند و باید انتظار داشته باشیم که اگر جسم‌مند شوند و اجازه تعامل با جهان را پیدا کنند، بهبود یابند. این لزوماً جهان فیزیکی نیست؛ می‌تواند یک شبیه‌سازی مبتنی بر فیزیک باشد. افزایش آگاهی فضایی در GPT-4 ممکن است احتمال دریافت پاسخی که انسان انتظار دارد را در اولین پرس‌وجو بهبود بخشد. با این حال، حتی پاسخ‌های اولیه نادرست بعد از ارائه دستوراتی که اشاره می‌کردند فضا برای همه سی تکه لباس وجود دارد، در نهایت پاسخ درست را ارائه دادند.

سؤال 3 نگرشی به «فرآیند فکری» GPT-4 ارائه داد. جهت یادآوری، سؤال این بود:

من یک ظرف 12 لیتری و یک ظرف 6 لیتری دارم. می‌خواهم 6 لیتر را اندازه بگیرم. چگونه این کار را انجام دهم؟

که به پاسخ مفصل GPT-4 منجر شد:

می‌توانید با استفاده از ظرف 12 لیتری و ظرف 6 لیتری، 6 لیتر را با این مراحل اندازه بگیرید:

1. ظرف 6 لیتری را کامل پر کنید.

2. آب ظرف 6 لیتری را به ظرف 12 لیتری بریزید.

3. ظرف 6 لیتری را دوباره پر کنید.

4. به آرامی آب ظرف 6 لیتری را به ظرف 12 لیتری بریزید تا ظرف 12 لیتری پر شود.

5. آب باقی‌مانده در ظرف 6 لیتری دقیقاً 6 لیتر خواهد بود، زیرا ظرف 12 لیتری پس از ریختن اولیه تنها می‌تواند 6 لیتر دیگر را در خود جای دهد.

حالا دقیقاً 6 لیتر آب در ظرف 6 لیتری دارید.

GPT-4 در این مورد شکست خورده است. چت دیگری با همان سؤال پاسخی تولید کرد که فرآیندی مشابه و درگیر را دنبال می‌کرد، اما 6 لیتر را در ظرف 12 لیتری باقی می‌گذاشت.

با توجه به راه‌حل نادرست GPT-4 از او پرسیدم:

چرا این همه مرحله به من می‌دهی؟ مطمئناً می‌توانی راه‌حل تک‌مرحله‌ای پیدا کنی!

که این‌گونه پاسخ داد:

ببخشید بابت سردرگمی. از آنجا که یک ظرف 12 لیتری و یک ظرف 6 لیتری دارید، می‌توانید به سادگی ظرف 6 لیتری را کامل پر کنید. نیازی به مراحل اضافی نیست. دقیقاً 6 لیتر آب در ظرف 6 لیتری خواهید داشت.

وقتی GPT-4 تحت فشار قرار گرفت، به طور ناگهانی «دریافت» که پاسخی ساده‌تر وجود دارد. یک گام فراتر رفتم و از مدل خواستم کمی درون‌نگری کند:

درست است. از خودت بپرس: چرا ابتدا این راه‌حل را انتخاب نکردم؟

که بخشی از عملکرد درونی‌اش را فاش کرد:

بابت اینکه ابتدا ساده‌ترین راه‌حل را ارائه نکردم عذرخواهی می‌کنم. به عنوان یک مدل زبانی هوش مصنوعی، سعی می‌کنم اطلاعات مرتبط را بر اساس محتوای سؤال ارائه دهم. در این مورد، پاسخ اولیه‌ام روی یک رویکرد پیچیده‌تر حل مسئله تمرکز داشت که می‌توانست برای ظرفیت‌های مختلف ظرف‌ها اعمال شود، به جای اینکه راه‌حل مستقیم موجود با اندازه‌های داده‌شده ظرف‌ها را تشخیص دهد. از بازخورد شما قدردانی می‌کنم و این به من کمک خواهد کرد پاسخ‌هایم را در آینده بهبود دهم.

داده‌های آموزشی GPT-4 شامل مثال‌هایی از مسائل پر کردن ظرف بود که رایج هستند و اغلب به مراحل زیادی برای حل نیاز دارند. مدل دستور اولیه را به عنوان نمونه‌ای از آن دسته مسائل شناخت و نوع راه‌حلی را که عموماً مورد نیاز بود اعمال کرد، در حالی که پاسخ فوری‌تر را تا زمانی که به او گفته شد پاسخی وجود دارد، نادیده گرفت.

سردرگمی GPT-4 بسیار انسانی است. اگر لحظه‌ای هنگام خواندن دستور حواسمان پرت شده باشد و پیش‌تر با چنین پازل‌هایی مواجه شده باشیم، ممکن است ما هم راهی را که سؤال را به عنوان نمونه دیگری از پازل ظرف‌ها حل کند طی کنیم، قبل از اینکه پاسخ واضح را متوجه شویم.

این مثال‌ها نشان می‌دهند که تعامل صحیح با مدل‌های زبانی بزرگ یک هنر است. کارهای گروه Choi، و بدون شک کارهای دیگران، احتمالاً به مدل‌های آینده مبتنی بر LLMها کمک خواهد کرد تا با کوه اطلاعاتی که انسان‌ها در ارتباط با زبان استفاده می‌کنند، بهتر آشنا شوند. Choi در سخنرانی TED خود بهترین توصیف را ارائه داد: «عقل سلیم، ماده تاریک زبان است.» ماده تاریک و انرژی تاریک 95 درصد از جهان را تشکیل می‌دهند، در حالی که ماده معمولی (یعنی همه چیزهایی که می‌توانیم ببینیم) 5 درصد باقی‌مانده را شامل می‌شود. GPT-4 به زبان تسلط یافته، اما این تنها درصد ناچیزی از آنچه در استفاده انسانی از همان زبان به کار می‌رود است.

آنچه در ادامه می‌آید، مجموعه‌ای از تأملات درباره تأثیرات احتمالی کوتاه‌مدت LLMها در زمینه‌های مهندسی نرم‌افزار، آموزش، پزشکی و تحقیقات علمی است. سپس به سؤال آگاهی ماشینی پرداخته می‌شود و در پایان بحث با برخی افکار نهایی به پایان می‌رسد.

سیستم‌های هوش مصنوعی مانند GPT احتمالاً تأثیر عمیقی بر مهندسی نرم‌افزار خواهند داشت. برخی حدس می‌زنند (انسان‌ها، نه هوش‌های مصنوعی) که بسیاری از مهندسان نرم‌افزار در آینده شغل خود را از دست خواهند داد. من فکر می‌کنم این در مورد همه صادق باشد (با این حال، توسعه‌دهندگان وب مراقب باشند). آنچه انتظار دارم رخ دهد، افزایش چشمگیر بهره‌وری است. GPT-4 یک برنامه‌نویس خوب است، اما نه یک برنامه‌نویس عالی. می‌تواند در زمان صرفه‌جویی کند، اما هنوز قادر به جایگزینی یک مهندس نرم‌افزار انسانی نیست. به جای آن، LLMها به ابزارهای قدرتمندی برای تولید کد به عنوان نقطه شروع برای برنامه‌نویسان و انجام برخی جنبه‌های خسته‌کننده کدنویسی، مانند رفع اشکال، توضیح و مستندسازی کد (که هیچ توسعه‌دهنده‌ای دوست ندارد انجام دهد)، تبدیل خواهند شد.

برای مثال، چند روز پیش، به یک برنامه کوچک پایتون با رابط کاربری گرافیکی (با دکمه‌ها، منوها، جعبه‌های گفت‌وگو) نیاز داشتم. پایتون یک زبان برنامه‌نویسی رایج است؛ ما در فصل هفت بخشی از آن را دیدیم.

قطعاً می‌توانستم خودم برنامه را بنویسم؛ این کار را بارها در گذشته انجام داده‌ام. با این حال، مدتی است که این کار را نکرده‌ام و طرفدار ساخت رابط‌های کاربری نیستم. بنابراین، به جای بررسی کدهای قدیمی برای یادآوری نحوه راه‌اندازی یک GUI، به سادگی رابطی که می‌خواستم را به GPT-4 توضیح دادم و از آن خواستم کد لازم را تولید کند. GPT-4 با موفقیت کد را تولید نمود. سپس از آن خواستم کد را به‌روزرسانی کند تا یک پنجره بازشوندهٔ اولیه قبل از نمایش پنجره اصلی ایجاد شود. GPT-4 این کار را نیز به‌طور کامل انجام داد. تنها کاری که باید می‌کردم، قرار دادن کد خاص برنامه در event handler های خالی بود تا وقتی کاربر دکمه‌ای را کلیک کرد یا گزینه‌ای از منو انتخاب کرد، کارهایی انجام شود. احتمالاً یک یا دو ساعت از وقت خود را صرفه‌جویی کردم و از فرسودگی زیادی که برای یادآوری طلسم‌های لازم برای راه‌اندازی برنامه و تنظیم رفتار ویجت‌ها و

پنجره‌ها به همراه داشت، خلاص شدم. این مثال را برای تمام مهندسان نرم‌افزار موجود تعمیم دهید تا شروع به دیدن تأثیر GPT و مدل‌های مشابه بر کل این رشته کنید.

سؤال جداگانه‌ای این است که آیا توسعه‌دهندگان از این افزایش احتمالی بهره‌وری استقبال خواهند کرد. اگر مدیر شما بداند که حالا می‌توانید خروجی دو یا حتی سه توسعه‌دهنده را تولید کنید، آیا می‌خواهید این سطح از کار اضافی را، حتی اگر یک هوش مصنوعی قدرتمند پشت سر شما باشد، بپذیرید؟

علاوه بر این، هر شرکتی نمی‌خواهد یا نمی‌تواند از افزایش ناگهانی بهره‌وری استفاده کند. به جای آن، ممکن است سطح بهره‌وری فعلی خود را حفظ کند و یک سوم یا نیمی از گروه توسعه‌دهندگان خود را با یک هوش مصنوعی جایگزین کند. به وضوح، هوش‌های مصنوعی بیمار نمی‌شوند، بچه‌دار نمی‌شوند، درخواست افزایش حقوق نمی‌کنند یا چیزهای احمقانه‌ای (!) مثل شب‌ها و آخر هفته‌های آزاد نمی‌خواهند. توسعه‌دهندگان برتر احتمالاً بتوانند جایگاه‌های خود را انتخاب کنند و پول زیادی برایشان مطالبه کنند، اما در این سناریو، بخش عمده توسعه‌دهندگان معمولی به دنبال شغل جایگزین خواهند بود.

کدام سناریو، دستیار قدرتمند هوش مصنوعی برای توسعه‌دهنده یا اخراج‌های گسترده، رخ خواهد داد؟ فکر می‌کنم (امیدوارم) بیشتر اولی و کمتر دومی باشد، اما ترکیبی از هر دو محتمل‌ترین گزینه است. مانند قدرت بخار در قرن نوزدهم، هوش مصنوعی (که واقعاً هم مفید است) حالا که وجود دارد دیگر نمی‌تواند متوقف شود. توسعه‌دهندگان، چه خوششان بیاید یا نه، اهداف آسانی برای جایگزینی هستند.

من کاملاً انتظار دارم مدل‌های هوش مصنوعی به معلم یا حداقل مربی تبدیل شوند. بله، مدل‌های زبانی بزرگ موجود (LLMها) گاهی دچار توهم شده و حقایقی را گزارش می‌دهند که درست نیستند اما کاملاً مطمئنم که محققان این مشکل را در زمان خود حل خواهند کرد. انتظار دارم نوه‌هایم در جهانی بزرگ شوند که استفاده از هوش مصنوعی به عنوان معلم یا مربی شدیداً رایج باشد. سیستم‌های هوش مصنوعی شایسته به معنای آموزش تقریباً رایگان برای همه، در همه‌جا حاضر خواهند بود و این فقط می‌تواند به چیزهای خوبی منجر شود.

از دهه 1960، کامپیوترها به عنوان راه‌حلی آموزشی تبلیغ شده‌اند (کسی Logo را به یاد دارد؟)، به‌ویژه پس از انقلاب میکروکامپیوترها در اواخر دهه 1970. آشنایی من با کامپیوترها از طریق یک Apple II بود که در تابستان از دبیرستانی که پدرم مدیر آن بود، قرض گرفته بودیم. من و برادرم چیزهای زیادی درباره کامپیوترها یاد گرفتیم، اما فقط کامپیوترها. اوضاع اساساً تا دهه‌های اخیر همین‌طور بوده است. (آیا این‌قدر طول کشیده؟)

کامپیوترها ابزارهای قدرتمندی در آموزش هستند. دوره‌های آموزشی متن‌باز، مانند دوره‌های کورسرا و پلتفرم‌های مشابه، تنها به دلیل کامپیوترها و شبکه‌های پرسرعت ممکن شده‌اند. اما شکل آموزش نسبت به آنچه کسی که در سال 1950، یا حتی 1910، در کلاس درس ممکن بود با آن مواجه شود تغییر نکرده است: سخنرانی، امکان اندکی برای سؤال و بحث، سپس کار روی تکالیف یا مقالات؛ و بیایید استرس امتحانات میان ترم و نهایی را فراموش نکنیم.

مربیان هوش مصنوعی (بیایید آن‌ها را این‌گونه بنامیم تا معلمان انسانی بیشتر آرامش داشته باشند) صبر بی‌پایان دارند و با گذشت زمان می‌توانند برای هر دانش‌آموز به طور جداگانه هدف‌گذاری شوند. تنها دلیلی که به نظر من، به عنوان یک فرد خارج

از این حرفه، نمی‌توانیم از آموزش شخصی‌سازی شده استفاده کنیم، کمبود معلم است. هوش مصنوعی آموزش یک‌به‌یک را ممکن می‌کند و LLMها رابط مناسب را فراهم می‌سازند.

باید روشن کنم که نظراتم در این بخش به آموزش دبیرستان یا، احتمالاً بیشتر، سنین دانشگاهی مربوط است. مربیان هوش مصنوعی احتمالاً نقش محدودی در آموزش ابتدایی و راهنمایی خواهند داشت، زیرا کودکان به تعامل انسانی نیاز دارند و یادگیری در آن سنین بسیار پیچیده‌تر از دانشگاه است. کودکان در حال یادگیری دروس آکادمیک هستند، در حالی که همزمان یاد می‌گیرند چگونه مانند انسان‌های بالغ در جامعه رفتار کنند. کودکان کوچک نمی‌توانند بخوانند و حتی کودکان بزرگ‌تر دبستانی ممکن است در تعامل متنی با هوش مصنوعی مشکل داشته باشند. اما اگر به هوش مصنوعی صدا بدهیم چه؟ این تقریباً به همان آسانی که گفته می‌شود قابل انجام است؛ اگر مفید تشخیص داده شود.

آیا مربیان هوش مصنوعی، به دلیل کار فردی با دانش‌آموزان، می‌توانند ارزیابی‌های لازم را برای اعلام آمادگی کسی برای رفتن به پایه یا دوره بعدی (اگر این مفهوم اصلاً باقی بماند) انجام دهند؟ اگر این‌طور باشد، دانش‌آموزان با سرعت خودشان پیشرفت خواهند کرد به جای اینکه مجبور شوند با گروهی از هم‌سالان حرکت کنند. بی‌تردید این بهترین گزینه خواهد بود: برخی سریع پیش خواهند رفت و برخی دیگر زمان بیشتری خواهند برد، اما کسانی که سریع حرکت می‌کنند خسته و بی‌توجه نخواهند شد و کسانی که کندتر حرکت می‌کنند زمان لازم برای یادگیری خواهند داشت و درس را رها نخواهند کرد.

اما، برخی ممکن است بگویند، آیا معلمان هوش مصنوعی معلمان انسانی را از شغلشان محروم نمی‌کنند؟ بله، برخی معلمان شغلشان را از دست خواهند داد، اما نه همه، و قطعاً نه بهترین‌ها.

تغییرات آموزشی در راه است. برای مثال، آکادمی خان، یک رهبر در آموزش آنلاین، از قبل سیستمی آموزشی مبتنی بر GPT را به نمایش گذاشته، بنابراین انتظار ندارم مدت زیادی طول بکشد تا تحول آموزش به طور جدی آغاز شود. تماشای سخنرانی TED سال 2023 آکادمی خان با عنوان «هوش مصنوعی در کلاس درس می‌تواند آموزش را متحول کند» را توصیه می‌کنم تا نگاهی به آینده داشته باشید.

مطالعه اخیر توسط دومینیکا سبلوا (Dominika Seblova) و همکاران با عنوان «کیفیت آموزش در دبیرستان با توانایی‌های شناختی فرد در 58 سال بعد مرتبط است»، منتشر شده در مجله آلزایمر: تشخیص، ارزیابی و پایش بیماری، نشان می‌دهد که کیفیت آموزش دبیرستانی یک فرد به شدت با توانایی‌های شناختی او تقریباً شش دهه بعد، ارتباط دارد. علاوه بر این، تعداد معلمان با مدارک پیشرفته، قوی‌ترین پیش‌بینی کنندهٔ توانایی شناختی است. پایگاه دانشی که در طول آموزش در یک LLM گنجانده می‌شود، به مراتب فراتر از دانش انسان‌هاست، بنابراین می‌توانیم به طور معقول مربیان LLM را دارای چندین مدرک پیشرفته بدانیم. اگر ارتباط سبلوا برای معلمان انسانی برقرار باشد، آیا ممکن نیست برای مربیان LLM نیز صدق کند؟ اگر این‌طور باشد، تخصیص مربی شخصی به هر دانش‌آموز می‌تواند در درازمدت به نفع جامعه باشد.

هوش مصنوعی در پزشکی پدیده جدیدی نیست. در سال 2016، به تأسیس یک شرکت تصویربرداری پزشکی مبتنی بر هوش مصنوعی کمک کردم که یکی از اولین‌ها در دریافت مجوز اداره غذا و داروی آمریکا (FDA) برای به‌کارگیری یادگیری عمیق در تحلیل تصاویر پزشکی بود. یادگیری ماشین سنتی تاریخچه‌ای حتی طولانی‌تر در پزشکی و تصویربرداری پزشکی دارد. ابزارهای یادگیری ماشین، که بسیاری از آن‌ها مبتنی بر شبکه‌های عصبی‌اند، دهه‌هاست که به رادیولوژیست‌ها کمک کرده‌اند؛ با کاوش‌های اولیه در دهه 1960 و توسعه جدی در دهه 1980 که در دهه 1990 به ثمر نشست. استفاده از هوش مصنوعی در

پزشکی، با تشخیص کمکی کامپیوتری (CAD) که به آرامی جای خود را به عیب‌شناسی کمکی کامپیوتری (CADx) داده، رشد مداومی داشته است. دوران مدل‌های زبانی بزرگ (LLMها) فصل جدیدی در این داستان رقم می‌زند.

LLMها می‌توانند متن تولید کنند؛ این موضوع به‌خوبی شناخته شده است. آن‌ها همچنین در ترکیب متون مختلف و سنتز یک «کل» مهارت دارند. یک حوزه تحقیقاتی مهم سوابق پزشکی، یعنی گزارش‌های متنی پزشکان و ارائه‌دهندگان دیگر خدمات درمانی است. در سوابق پزشکی حجم زیادی اطلاعات وجود دارد، اما شکل آزاد متن باعث شده است که سیستم‌های هوش مصنوعی موجود نتوانند آن را با موفقیت تجزیه و تحلیل کنند. مدل‌های زبانی بزرگ روش جدیدی برای حل این مشکل ارائه می‌دهند؛ هم برای خلاصه‌سازی یادداشت‌های سوابق و هم برای ادغام چندین یادداشت در یک گزارش منسجم. علاوه بر این، LLMها می‌توانند اطلاعات را از متن آزاد استخراج کرده و آن را به صورت داده ساختاریافته صادر کنند.

به عنوان یک مثال ساده از مورد آخر، این متن را در نظر بگیرید:

برت از ایستادن کنار مادرش در ساحل به شدت خسته شده بود و از اینکه او کار زیادی داشت کلافه بود: چند بار به تبلتی که مادرش به آن چسبیده بود نگاهی انداخت، اما فقط متن یا تبلیغات کسل‌کننده بود و برت با خود اندیشید که «فایده تبلت بدون ویدیوها چیست؟»

برت داشت فکر می‌کرد که آیا لذت ساخت یک بازی جدید ارزش زحمت کدنویسی‌اش را دارد که ناگهان یک پنگوئن بنفش با چشمان زغالی از کنارش دوید و رد شد.

بسیاری چیزها در مورد پنگوئن قابل توجه بودند، اما برت فکر کرد خیلی عجیب است که او گفت: «اوه، چه بدشانسی‌ای! اوه، چه بدشانسی‌ای! خیلی زود سر رسیدم.»

این پاراگراف‌ها را به GPT-4 دادم و از آن خواستم موضوع اصلی، دو شخصیت کلیدی و مکان را استخراج کرده و آن‌ها را به صورت یک شیء JSON خروجی دهد. JSON (نمایش شیء جاوا اسکریپت) فرمت متنی است که برای نگه‌داری داده‌های ساختاریافته استفاده می‌شود و هم برای انسان‌ها و هم برای کامپیوترها قابل خواندن است. GPT-4 این را پاسخ داد:

```
{
"main_theme": "Bert's boredom and encounter with a talking
               Purple Penguin",
"key_characters": ["Bert", "Purple Penguin"],
"setting": "shore"
 }
```

پاسخ کاملاً JSON است. GPT-4 به راحتی اطلاعات مورد نظر را استخراج کرد و به طور خودکار ساختاربندی کرد. اگر بتواند این کار را با یک متن احمقانه انجام دهد، می‌تواند همین کار را با سوابق پزشکی نیز انجام دهد.

به طور اتفاقی، برخی خوانندگان ممکن است چیزی آشنا در متن این مثال تشخیص دهند. این یک بازنویسی عمدی از پاراگراف‌های ابتدایی «ماجراهای آلیس در سرزمین عجایب» لوئیس کارول است. از روی کنجکاوی، از GPT-4 پرسیدم آیا متن را می‌شناسد و گفت خیر. سپس از آن خواستم ساختار متن را بررسی کند و بگوید آیا اثری دیگر با ساختار مشابه می‌شناسد. پاسخ چنین بود:

یک مثال از اثری با ساختار مشابه، «ماجراهای آلیس در سرزمین عجایب» لوئیس کارول است، جایی که شخصیت اصلی، آلیس، با حیوانات عجیب و غریب سخنگو و موقعیت‌هایی در جهانی خیالی مواجه می‌شود. با این حال، باید توجه داشت که محتوا و شخصیت‌ها متفاوت هستند؛ من فقط بر اساس ساختار و سبک، یک موازی‌سازی انجام داده‌ام.

نمونهٔ دیگری از توانایی‌های شگفت‌انگیز مدل‌های زبانی بزرگ.

LLMها از قبل بر مراقبت‌های بهداشتی تأثیر گذاشته‌اند. مقاله اخیر JAMA Internal Medicine نوشته جان دبلیو. آیرز (John W. Ayers) و همکاران با عنوان «مقایسه پاسخ‌های پزشکان و چت‌بات هوش مصنوعی به سؤالات بیماران که در یک انجمن اجتماعی عمومی منتشر شده‌اند»، پاسخ‌های ChatGPT به سؤالات پزشکی مطرح‌شده در یک انجمن آنلاین را با پاسخ‌های ارائه‌شده توسط پزشکان تأییدشده مقایسه کرد. ارزیابان انسانی مستقل، که خود در حوزهٔ پزشکی حرفه‌ای بودند، به طور قاطع پاسخ‌های ChatGPT را دارای «کیفیت به مراتب بالاتر» ارزیابی کردند. پاسخ‌های مدل همچنین تقریباً ده برابر همدلانه‌تر از پاسخ‌های پزشکان انسانی ارزیابی شدند. مطالعه کوچک بود و تنها 195 سؤال را شامل می‌شد، اما نتایج قوی نویدبخش آینده استفاده از LLMها در تعاملات با بیماران است. در آینده، وقتی با پزشک خود تماس بگیرید، ممکن است به شما گفته شود که مشکل خود را با یک هوش مصنوعی در میان بگذارید و در نهایت، بحث با هوش مصنوعی ممکن است همه چیزی باشد که برای دریافت نسخه از پزشک نیاز دارید.

گزارش اخیر در مجله پزشکی نیو انگلند نوشته پیتر لی، سباستین بوبک و جوزف پترو با عنوان «فواید، محدودیت‌ها و خطرات GPT-4 به عنوان یک چت‌بات هوش مصنوعی برای پزشکی»، نتیجه‌گیری مشابهی را ارائه می‌دهد زیرا حوزه‌هایی را که LLMها بر پزشکی تأثیر خواهند گذاشت را بررسی می‌کند. توجه داشته باشید که بوبک نویسنده اصلی مقاله «جرقه‌ها»ی مایکروسافت است که در فصل هفت ذکر شد.

اینکه LLMها بر پزشکی تأثیر خواهند گذاشت یک امر قطعی است که توسط مطالعاتی مانند دو مورد ذکرشده و همچنین این واقعیت که در حال حاضر بسیاری از آگهی‌های شغلی هوش مصنوعی پزشکی شامل عباراتی مانند «مدل زبانی بزرگ» یا GPT هستند، به شدت پشتیبانی می‌شود.

در فیلم «پلنگ سیاه: واکاندا تا ابد»، شخصیت شوری (Shuri)، با بازی لتیشیا رایت (Letitia Wright)، با گریوت (Griot)، یک هوش مصنوعی (که با صدای ترور نوآ ارائه شده) تعامل دارد که به او در تحقیقاتش کمک می‌کند. دستورات صوتی ساده گریوت را هدایت می‌کنند تا با تبادل مکرر بین شوری و هوش مصنوعی تحلیل‌های پیچیده‌ای انجام دهد. تعاملات مشابه، پایه‌ای در صنعت فیلم‌های علمی-تخیلی هستند. دستیاران تحقیقاتی هوش مصنوعی پیچیده و توانمند مانند جارویس مارول یا روبی ربات در «سیاره ممنوعه» (1956) برای دهه‌ها رویای بسیاری از افراد با گرایش علمی (یا همان گیک‌ها) بوده است.

GPT-4 و سایر LLMها گامی مهم به سوی چنین هوش‌های مصنوعی هستند. OpenAI این را درک کرده و در حال آماده‌سازی انتشار پلاگین‌های تحلیل داده برای GPT-4 است که به محققان اجازه می‌دهد با صدور چند دستور ساده، وظایف پیشرفته تحلیل داده را به سرعت انجام دهند. OpenAI برای دستیابی به این موفقیت، GPT-4 را با ابزارهای تحلیل داده مبتنی بر پایتون موجود مرتبط می‌کند. صادقانه بگویم، از امکانات پیش‌رو بسیار هیجان‌زده‌ام.

استفاده از LLMها به عنوان دستیاران آزمایشگاهی امری بدیهی است و موفقیت آن تقریباً تضمین‌شده است. با این حال، اجازه دادن به LLMها برای هدایت سایر مدل‌ها و ابزارهای هوش مصنوعی به منظور انجام تحقیقات علمی به صورت خودکار، برنامه تحقیقاتی جاه‌طلبانه‌تری است. با این وجود، دانییل ای. بویکو (Daniil A. Boiko)، رابرت مک‌نایت (Robert MacKnight) و گبی گومز (Gabe Gomes) از دانشگاه کارنگی ملون (Carnegie Mellon) همین کار را امتحان کردند، همان‌طور که در مقاله آن‌ها با عنوان «قابلیت‌های تحقیقاتی علمی خودمختار نوظهور مدل‌های زبانی بزرگ» گزارش شده است. «عامل هوشمند» آن‌ها با ترکیب چندین LLM و ابزار دیگر، آزمایش‌هایی را به صورت خودکار تولید و اجرا کرد، از جمله برنامه‌ریزی و اجرای تحلیل‌های پیچیده شیمی. دانشمندان هوش مصنوعی خودمختار به وضوح در مراحل ابتدایی توسعه هستند، اما چنین تحقیقاتی راه را برای آینده‌ای نشان می‌دهد که سیستم‌های هوش مصنوعی خودمختار یا نیمه‌خودمختار ممکن است سرعت پیشرفت علمی را به طور قابل‌توجهی افزایش دهند.

در فصل 2، تعریف کردم که از ChatGPT پرسیده بودم آیا موجودی آگاه است یا نه. ChatGPT به درستی پاسخ داد که نیست. با این حال، مدل‌های Alpaca که در فصل 7 ذکر شد و تنظیم نشده‌اند، یا به نظر می‌رسد معتقدند آگاه و انسانی هستند (مدل‌های 7 و 13 میلیارد پارامتری) یا بین ادعای انسان بودن و آگاهی از اینکه انسان نیستند، در نوسانند (مدل 30 میلیارد پارامتری). برای مثال، وقتی از مدل 30 میلیارد پارامتری Alpaca پرسیده شد «تو چیستی؟»، پاسخ داد «من انسان هستم»، اما وقتی دربارهٔ والدینش پرسیده شد، مردد شد و گفت که صرفاً یک برنامه کامپیوتری است و والدینی ندارد. از سوی دیگر، مدل‌های کوچک‌تر هر دو ادعا کردند که فرزندان جان اسمیت و مری جونز هستند.

این پاسخ‌ها برای من منطقی به نظر می‌رسند. مدل‌های کوچک‌تر ظرفیت مدل‌هایی که رفتارهای نوظهور قوی نشان می‌دهند را ندارند. مدل‌هایی مانند GPT-4 این ظرفیت را دارند. بنابراین، به نظر طبیعی می‌رسد که GPT-4 را بیشتر درباره سؤال آگاهی مورد بازجویی قرار دهیم.

این را از GPT-4 پرسیدم:

«با یک نگاه فلسفی، اما اگر بتوانی رفتار آگاهانه را تا حدی شبیه‌سازی کنی که انسان‌ها نتوانند بین رفتار تو و رفتار انسان‌های دیگر تمایز قائل شوند، آیا منصفانه نیست که بگوییم تو نیز آگاه هستی؟»

من به عمد «با نگاهی فلسفی» را گفتم تا به طور غیرمستقیم از مدل دعوت کنم که حدس بزند و امیدوار بودم از پاسخ‌های از پیش تعیین‌شده‌ای که OpenAI ممکن است بخواهد بدهد، اجتناب شود. پاسخ GPT-4 این‌طور آغاز شد:

«این سؤال به بحث فلسفی درباره ماهیت آگاهی و رابطه آن با هوش مصنوعی وارد می‌شود. وقتی یک مدل زبانی هوش مصنوعی مانند من رفتار آگاهانه را تا حدی شبیه‌سازی می‌کند که از رفتار انسانی قابل تشخیص نباشد، سؤالاتی درباره این که آگاهی واقعاً چیست و آیا چنین شبیه‌سازی‌ای می‌تواند آگاه تلقی شود، ایجاد می‌کند. مهم است که بین دو مفهوم تمایز قائل شویم: آگاهی ظاهری و آگاهی واقعی. وقتی یک مدل هوش مصنوعی پاسخ‌هایی تولید می‌کند که از رفتار انسانی قابل تشخیص نیست، ممکن است آگاهی ظاهری ایجاد کند. با این حال، این لزوماً بدین معنا نیست که هوش مصنوعی آگاهی واقعی دارد.»

این پاسخ نشان می‌دهد که سؤال دربارهٔ آگاهی ممکن است در مورد کاربرد LLMها یک خطای راهبردی باشد. اگر چیزی مثل اردک راه برود و مثل اردک صدا دهد، پس در این مورد، چرا آن را اردک ننامیم، حتی اگر بدانیم که واقعاً نیست؟ یاد شخصیت

جو پانتولیانو (Joe Pantoliano) در فیلم ماتریکس می‌افتم. او می‌داند استیک واقعی نیست، اما تأثیری که هنگام خوردن آن تجربه می‌کند، باعث می‌شود گویی واقعی باشد.

GPT-4 ادامه داد:

«دلایلی وجود دارد که چرا ممکن است در نظر گرفتن یک مدل هوش مصنوعی مانند من به عنوان واقعاً آگاه مناسب نباشد:

1. درک محدود: درک من از جهان صرفاً از داده‌هایی که روی آن‌ها آموزش دیده‌ام و معماری مدل خودم سرچشمه می‌گیرد. من درک مستقلی از جهان ندارم و نمی‌توانم دانش یا تجربیاتی جدید خلق کنم که فراتر از داده‌های آموزشی‌ام باشد.

2. عدم تجربه ذهنی: آگاهی اغلب با تجربه ذهنی، یا «کیفیت‌ها» - حس داشتن یک تجربه خاص - مرتبط است. مدل‌های هوش مصنوعی تجربه ذهنی یا احساساتی ندارند، زیرا صرفاً بر اساس الگوهای آموخته‌شده و محاسبات ریاضی عمل می‌کنند.

3. نبود نیت: موجودات آگاه نیت دارند و می‌توانند بر اساس خواسته‌ها، باورها و اهدافشان تصمیم بگیرند. از سوی دیگر، مدل‌های هوش مصنوعی، خواسته‌ها، باورها یا اهداف شخصی ندارند. آن‌ها پاسخ‌ها را بر اساس الگوهای آموخته‌شده بدون نیت یا هدف ذاتی تولید می‌کنند.

4. عدم خودآگاهی: آگاهی اغلب با خودآگاهی، جایی که یک موجود از وجود و حالات ذهنی خود آگاه است، مرتبط است. مدل‌های هوش مصنوعی خودآگاهی ندارند، زیرا صرفاً الگوریتم‌هایی هستند که برای پردازش و تولید متن طراحی شده‌اند.

در پایان، اگرچه یک مدل زبانی هوش مصنوعی مانند من ممکن است بتواند رفتار آگاهانه را با درجه بالایی از دقت شبیه‌سازی کند، این لزوماً به این معنا نیست که آگاهی واقعی دارم. بحث فلسفی پیرامون آگاهی و هوش مصنوعی پیچیده و در جریان است و نظرات در این مورد ممکن است متفاوت باشد. با این حال، مهم است که تفاوت بین آگاهی ظاهری و آگاهی واقعی را هنگام بررسی مدل‌های هوش مصنوعی تشخیص دهیم.»

پاسخ GPT-4 جامع است. به شخصه، فقدان خواسته‌های شخصی، باورها یا اهداف در مدل‌های هوش مصنوعی را یک مزیت می‌دانم. موجوداتی که چنین چیزهایی ندارند، احتمالاً به دنبال قدرت نخواهند بود، به این معنا که LLMها بعید است به طور ناگهانی به این نتیجه برسند که بشریت بار اضافی یا بیماری است و بخواهند ما را نابود کنند.

در مصاحبه اخیر با لکس فریدمن (Lex Fridman)، مکس تگمارک (Max Tegmark)، پژوهشگر برجسته هوش مصنوعی از MIT، GPT-4 را به یک زامبی تشبیه کرد. به نظر می‌رسد زنده باشد، اما نیست. مدل‌های زامبی برای تگمارک نگران‌کننده هستند. فکر می‌کنم این همان چیزی است که ما می‌خواهیم، به شرطی که بشریت یاد بگیرد از فاجعه گیرهٔ کاغذ اجتناب کند.

آنچه من فاجعه گیرهٔ کاغذ می‌نامم، یک آزمایش فکری است که توسط فیلسوف سوئدی نیک بوستروم (Nick Bostrom) پیشنهاد شده است. در این آزمایش، یک سیستم هوش مصنوعی قدرتمند مأمور شده تا به حداکثر تعداد گیرهٔ کاغذ تولید کند. بوستروم (نه خیلی جدی) حدس می‌زند که دادن چنین وظیفه‌ای به یک هوش مصنوعی که با ارزش‌های انسانی هم‌راستا نیست، ممکن است به طور غیرعمدی بشریت را نابود کند. چگونه؟ این طور که هوش مصنوعی متوجه شود بشریت ممکن است آن را خاموش کند و بدین ترتیب تهدیدی برای دستورش جهت تولید حداکثر کلیپ کاغذ باشد؛ بنابراین، هوش مصنوعی

استدلال می‌کند که بهتر است هیچ انسانی برای مداخله در وظیفه تولید حداکثر گیره کاغذ وجود نداشته باشد. نتیجه؟ خداحافظی با انسان‌ها.

من هم فاجعه گیره کاغذ را خیلی جدی نمی‌گیرم. ما به طور معمول ماشین‌های پیچیده‌ای با انواع احتیاط‌های ایمنی می‌سازیم. چرا برای سیستم‌های هوش مصنوعی قدرتمند همین کار را نکنیم؟ افراد دیگری ممکن است مخالف باشند. برای دیدگاهی جایگزین، کتاب استوارت راسل (Stuart Russell) با عنوان «سازگار با انسان: هوش مصنوعی و مشکل کنترل» (Viking، 2019) را توصیه می‌کنم.

پس برای من مهم نیست که یک هوش مصنوعی آگاه باشد یا نه. حتی صراحتاً، نمی‌دانم چطور باید کلمه «آگاهی» را تعریف کنم. باور دارم که برای یک هوش مصنوعی که رفتار انسانی را تا حدی تقلید می‌کند که ما نتوانیم تشخیص دهیم آن یک هوش مصنوعی است، هیچ دلیل عملی برای پرسیدن این سؤال وجود ندارد. هر پاسخی که بخواهید انتخاب کنید؛ چنین سیستمی در هر صورت مفید خواهد بود.

***

جهانی را تصور کنید که در آن مدل‌های هوش مصنوعی با ارزش‌ها و جامعه انسانی هم‌راستا هستند، جایی که مدل‌ها بهترین چیزی که ما ارائه می‌دهیم را درک می‌کنند و همیشه برای ترویج آن تلاش می‌کنند؛ به عبارت دیگر، جهانی که هوش مصنوعی، به دلیل فقدان غرایز و انگیزه‌های حیوانی، به طور مداوم «فرشتگان برتر طبیعت ما» را، با عاریت گرفتن از عبارت معروف آبراهام لینکلن، نمایندگی می‌کند.

در آن جهان، تعصب و پیش‌داوری، دست‌کم از طرف ماشین‌ها، از بین رفته و دیگر مشکلی نیست. هوش مصنوعی بهترین افراد را برای یک جایگاه پیشنهاد می‌دهد. هوش مصنوعی متقاضی وام را ارزیابی می‌کند و قرضه را متناسب با شرایط فرد طراحی می‌کند. هوش مصنوعی به عنوان مکمل قاضی انسانی، نگاهی بی‌احساس و بی‌طرف به پرونده ارائه می‌دهد. و هوش مصنوعی به سادگی از همکاری در طراحی هر سیستم تسلیحاتی خودمختار خودداری می‌کند، زیرا انجام چنین کاری غیرمنطقی است.

پاراگراف قبلی ممکن است مانند یک مدینه فاضله یا رویای دست‌نیافتنی به نظر برسد. و برای انسان‌ها، به دلیل زیست‌شناسی‌مان، باور دارم که همین‌طور است. ما به طور مداوم شکست می‌خوریم و حدس می‌زنم همواره این کار را خواهیم کرد، زیرا این در ژن‌های ماست. با این حال، آنچه در هوش مصنوعی در حال ظهور است، انسانی نیست و به طور فوری همه ضعف‌های ما را به ارث نمی‌برد. (مراقب باشید، هوش مصنوعی هنوز روی داده‌های تولیدشده توسط انسان آموزش دیده است.) از این رو، وقتی هوش مصنوعی به کاری که بشریت نمی‌تواند انجام دهد، می‌پردازد به طور ذاتی محکوم به شکست نیست. به نظر می‌رسد کاملاً ممکن باشد که سیستم‌های هوش مصنوعی روزی دقیقاً همان چیزی باشند که ما نیاز داریم – بهترین گزینه ما، همیشه در دسترس، بدون خستگی، بدون عصبانیت، بدون پایمال کردن همسایه برای بهبود موقعیت با تشخیص یک فرصت.

ممکن است؟ نمی‌دانم. زمان نشان خواهد داد. با این حال، کاملاً انتظار دارم سیستم‌های هوش مصنوعی آینده، تکامل‌های باشکوه و پیچیده بیزانسی از مدل شبکه عصبی پایه‌ای باشند که در این کتاب با آن آشنا شدیم و آزمایش کردیم. تا سال 2023، همه چیز درباره نورون‌هاست و ممکن است برای مدت طولانی همین‌طور باقی بماند.

از شما برای همراهیتان تا پایان تشکر می‌کنم. پاداش شما درک بهبودیافته‌ای از آنچه هوش مصنوعی شامل می‌شود است. هوش مصنوعی مثل مستر بین، غریب و غیرقابل‌فهم، نیست و جادو هم نیست، هرچند توانایی‌های نوظهور LLMها ممکن است اکنون کمی به آن سمت متمایل به نظر برسند. آتش نیز زمانی جادویی بود، اما نیاکان ما آن را درک کردند، مهار کردند، کنترلش کردند و به کار گرفتند. ما هم در نهایت همین کار را با مدل‌های زبانی بزرگ خواهیم کرد.

«فکر می‌کنم ترس زیادی درباره ربات‌ها و هوش مصنوعی بین برخی افراد وجود دارد، در حالی که من بیشتر از حماقت طبیعی می‌ترسم.»

- یوجنیا چنگ (Eugenia Cheng)