به نام خدا

موضوع :

ترجمه متن

نام استاد :

دکتر عصایی معمم

نام و نام خانوادگی :

شایان جعفرپور

فصل ۶: جریان کار جهانی یادگیری ماشین

در زمینه‌ی این کار، استفاده از عقل سلیم انسانی. برای رسیدن به این هدف، باید نشان دهید که برخی از حالات شکست مدل شما چیست (به‌عنوان مثال، نشان دهید که نمونه‌های نادرست طبقه‌بندی شده چگونه به نظر می‌رسند، به‌ویژه آن دسته که برای آنها طبقه‌بندی نادرست تعجب‌آور به نظر می‌رسد).

آنها ممکن است انتظار عملکرد در سطح انسانی داشته باشند، به‌ویژه برای فرآیندهایی که قبلاً توسط افراد انجام می‌شدند. اکثر مدل‌های یادگیری ماشین، زیرا به‌طور (ناقص) برای تقریب برچسب‌های تولید شده توسط انسان آموزش دیده‌اند، تقریباً به آن سطح نمی‌رسند. شما باید انتظارات عملکرد مدل را به وضوح منتقل کنید. از استفاده از بیانات انتزاعی مانند «مدل ۹۸٪ دقت دارد» (که اکثر افراد به طور ذهنی آن را به ۱۰۰٪ گرد می‌کنند) پرهیز کنید و ترجیحاً درباره نرخ‌های منفی کاذب و مثبت کاذب صحبت کنید. به‌عنوان مثال می‌توانید بگویید: «با این تنظیمات، مدل تشخیص تقلب دارای نرخ منفی کاذب ۵٪ و نرخ مثبت کاذب ۲.۵٪ خواهد بود. به‌طور میانگین، ۲۰۰ تراکنش معتبر در روز به‌عنوان تقلبی شناسایی و برای بررسی دستی ارسال می‌شوند، و به‌طور میانگین ۱۴ تراکنش تقلبی نادیده گرفته خواهد شد. به‌طور میانگین ۲۶۶ تراکنش تقلبی به‌درستی شناسایی خواهد شد.» به‌وضوح معیارهای عملکرد مدل را با اهداف تجاری مرتبط کنید.

شما همچنین باید اطمینان حاصل کنید که با ذینفعان، انتخاب پارامترهای کلیدی راه‌اندازی—به‌عنوان مثال، آستانه احتمال که در آن تراکنش باید شناسایی شود—را مورد بحث قرار دهید (آستانه‌های مختلف نرخ‌های متفاوتی از مثبت کاذب و منفی کاذب تولید خواهند کرد). چنین تصمیمات شامل تعادل‌هایی است که تنها با درک عمیق از زمینه تجاری قابل مدیریت است.

**6.3.2 ارسال مدل استخراج**

یک پروژه یادگیری ماشین وقتی به یک نوت‌بوک Colab برسد که بتواند یک مدل آموزش‌دیده را ذخیره کند، تمام نمی‌شود. شما به ندرت دقیقاً همان شیء مدل پایتون را که هنگام آموزش آن را دستکاری کردید، به تولید می‌رسانید.

اول، ممکن است بخواهید مدل خود را به چیزی غیر از پایتون صادر کنید:

* محیط تولید شما ممکن است اصلاً از پایتون پشتیبانی نکند — برای مثال، اگر یک برنامه موبایلی یا یک سیستم تعبیه‌شده باشد.
* اگر بقیه برنامه به پایتون نوشته نشده باشد ) می‌تواند جاوا اسکریپت، C++ و غیره باشد )، استفاده از پایتون برای سرویس‌دهی به یک مدل ممکن است سربار قابل توجهی ایجاد کند. دوم، از آنجا که مدل تولید شما فقط برای خروجی گرفتن پیش‌بینی‌ها استفاده می‌شود (مرحله‌ای به نام استنتاج)، به جای آموزش، شما فضای لازم برای انجام بهینه‌سازی‌های مختلفی که می‌تواند مدل را سریع‌تر کند و جایگاه حافظه آن را کاهش دهد، دارید.

بیایید نگاهی سریع به گزینه‌های مختلف استقرار مدل که در اختیار دارید، بیندازیم.

**استقرار یک مدل به عنوان یک REST API**

این شاید رایج‌ترین راه برای تبدیل یک مدل به محصول باشد: نصب TensorFlow بر روی یک سرور یا نمونه ابری، و پرسش پیش‌بینی‌های مدل از طریق یک REST API. شما می‌توانید برنامه سرویس‌دهی خود را با استفاده از چیزی مانند) Flask یا هر کتابخانه توسعه وب پایتون دیگر) بسازید، یا از کتابخانه خود TensorFlow برای ارسال مدل‌ها به عنوان API‌ها استفاده کنید که به آن TensorFlow Serving گفته می‌شود ([www.tensorflow.org/tfx/guide/serving](http://www.tensorflow.org/tfx/guide/serving))

با استفاده از TensorFlow Serving، می‌توانید یک مدل Keras را در عرض چند دقیقه مستقر کنید.

شما باید از این تنظیمات استقرار استفاده کنید وقتی که:

* برنامه‌ای که از پیش‌بینی‌های مدل استفاده می‌کند به اینترنت دسترسی قابل اعتماد دارد (بدیهی است). به عنوان مثال، اگر برنامه شما یک برنامه موبایلی باشد، ارائه پیش‌بینی‌ها از طریق یک API راه دور به این معنی است که برنامه در حالت هواپیما یا در محیط کم‌ارتباط قابل استفاده نخواهد بود.
* برنامه الزامات تأخیر سختی ندارد: درخواست، استنتاج، و پاسخ معمولاً حدود 500 میلی‌ثانیه طول می‌کشد.
* داده‌های ورودی ارسال‌شده برای استنتاج بسیار حساس نیستند: داده‌ها باید به شکل رمزنگاری‌شده روی سرور موجود باشند، زیرا مدل باید آنها را ببیند) ولی توجه داشته باشید که باید از رمزنگاری SSL برای درخواست و پاسخ HTTP استفاده کنید(

به عنوان مثال، پروژه موتور جستجوی تصویر، سیستم توصیه موسیقی، پروژه تشخیص تقلب کارت اعتباری، و پروژه تصاویر ماهواره‌ای همگی گزینه‌های مناسبی برای ارائه از طریق یک REST API هستند.

یک سؤال مهم هنگام استقرار یک مدل به عنوان یک REST API این است که آیا می‌خواهید کد را خودتان میزبانی کنید یا از یک سرویس ابری کاملاً مدیریت‌شده استفاده کنید. به عنوان مثال، Cloud AI Platform، یک محصول گوگل، به شما اجازه می‌دهد که مدل TensorFlow خود را به سادگی در Google Cloud Storage (GCS) بارگذاری کنید، و یک نقطه پایانی API برای پرسش از آن به شما می‌دهد. این سرویس بسیاری از جزئیات عملی مانند دسته‌بندی پیش‌بینی‌ها، تعادل بار، و مقیاس‌پذیری را مدیریت می‌کند.

**استقرار یک مدل بر روی دستگاه**

گاهی اوقات، ممکن است نیاز داشته باشید که مدل شما در همان دستگاهی زندگی کند که برنامه‌ای که از آن استفاده می‌کند را اجرا می‌کند—شاید یک گوشی هوشمند، یک پردازنده ARM تعبیه‌شده روی یک ربات، یا یک میکروکنترلر روی یک دستگاه کوچک. ممکن است دیده باشید که یک دوربین قادر به تشخیص خودکار افراد و چهره‌ها در صحنه‌هایی که به آن اشاره می‌کنید است: احتمالاً یک مدل یادگیری عمیق کوچک بوده که مستقیماً روی دوربین اجرا می‌شود.

شما باید از این تنظیمات استفاده کنید وقتی

مدل شما دارای محدودیت‌های تأخیر سخت است یا نیاز به اجرا در محیط کم‌اتصال دارد. اگر در حال ساخت یک برنامه واقعیت افزوده فراگیر هستید، پرسش از یک سرور راه دور گزینه قابل قبولی نیست.

 مدل شما می‌تواند به اندازه کافی کوچک شود تا بتواند تحت محدودیت‌های حافظه و توان دستگاه هدف اجرا شود. شما می‌توانید از کیت ابزار بهینه‌سازی مدل TensorFlow برای کمک به این کار استفاده کنید([www.tensorflow.org/model\_optimization](http://www.tensorflow.org/model_optimization))

* دستیابی به بالاترین دقت ممکن برای وظیفه شما حیاتی نیست. همیشه بین کارایی زمان اجرا و دقت، یک تجارت وجود دارد، بنابراین محدودیت‌های حافظه و قدرت اغلب نیاز دارند که شما مدلی را ارسال کنید که به خوبی بهترین مدلی که می‌توانید روی یک GPU بزرگ اجرا کنید، نباشد.
* داده‌های ورودی به شدت حساس هستند و بنابراین نباید روی یک سرور از راه دور رمزگشایی شوند.

مدل تشخیص هرزنامه ما باید به عنوان بخشی از برنامه چت، بر روی گوشی هوشمند کاربر نهایی اجرا شود، زیرا پیام‌ها رمزگذاری شده‌اند و از این رو نمی‌توان توسط یک مدل میزبانی شده از راه دور خوانده شوند. به همین ترتیب، مدل تشخیص کوکی بد، محدودیت‌های تأخیر دارد و باید در کارخانه اجرا شود. خوشبختانه، در این مورد، ما هیچ محدودیت قدرت یا فضایی نداریم، بنابراین می‌توانیم مدل را بر روی یک GPU اجرا کنیم.

برای استقرار مدل Keras روی یک گوشی هوشمند یا دستگاه تعبیه‌شده، راه‌حل شما TensorFlow Lite است ([www.tensorflow.org/lite](http://www.tensorflow.org/lite)). این یک چارچوب برای اجرای کارآمد استنتاج یادگیری عمیق بر روی دستگاه است که بر روی گوشی‌های هوشمند Android و iOS، همچنین کامپیوترهای ARM64، Raspberry Pi یا میکروکنترلرهای خاص کار می‌کند. این شامل یک مبدل است که می‌تواند مدل Keras شما را به قالب TensorFlow Lite تبدیل کند.

**پیاده‌سازی یک مدل در مرورگر**

"یادگیری عمیق اغلب در برنامه‌های مبتنی بر مرورگر یا دسکتاپ بر اساس جاوااسکریپت استفاده می‌شود. در حالی که معمولاً ممکن است برنامه با استفاده از یک مدل راه دور از طریق یک REST API پرس‌وجو کند، ممکن است مزایای کلیدی در اجرای مدل به طور مستقیم در مرورگر، بر روی کامپیوتر کاربر وجود داشته باشد(در صورت موجود بودن استفاده از منابع GPU)

از این تنظیمات زمانی استفاده کنید که

 می‌خواهید محاسبات را به کاربر نهایی منتقل کنید، که می‌تواند به طور قابل توجهی هزینه‌های سرور را کاهش دهد.

 داده‌های ورودی باید روی کامپیوتر یا گوشی کاربر نهایی باقی بمانند. به عنوان مثال، در پروژه تشخیص هرزنامه ما، نسخه وب و نسخه دسکتاپ برنامه چت (که به صورت یک برنامه چند پلتفرمی در جاوا اسکریپت نوشته شده است) باید از یک مدل محلی استفاده کنند.

 برنامه شما محدودیت‌های تاخیری شدیدی دارد. در حالی که یک مدل که روی لپ‌تاپ یا گوشی هوشمند کاربر نهایی اجرا می‌شود احتمالاً کندتر از مدلی است که روی یک GPU بزرگ روی سرور شما اجرا می‌شود، اما شما ۱۰۰ میلی‌ثانیه اضافی از سفر رفت و برگشت شبکه ندارید.

 نیاز دارید که برنامه شما بدون اتصال به اینترنت نیز کار کند، پس از آن که مدل دانلود و کش شده است.

فقط در صورتی باید این گزینه را انتخاب کنید که مدل شما به اندازه کافی کوچک باشد که CPU، GPU یا RAM لپ‌تاپ یا گوشی هوشمند کاربر شما را اشغال نکند. علاوه بر این، از آنجا که کل مدل به دستگاه کاربر دانلود خواهد شد، باید مطمئن شوید که هیچ چیز درباره مدل نیازی به محرمانه بودن ندارد. توجه داشته باشید که با داشتن یک مدل یادگیری عمیق آموزش دیده، معمولاً ممکن است مقداری اطلاعات درباره داده‌های آموزشی بازیابی شود: بهتر است مدل آموزش دیده شما عمومی نشود اگر بر روی داده‌های حساس آموزش دیده شده است. برای استقرار مدل در جاوا اسکریپت، اکوسیستم TensorFlow شامل TensorFlow.js است([www.tensorflow.org/js](http://www.tensorflow.org/js)). ، یک کتابخانه جاوا اسکریپت برای یادگیری عمیق که تقریباً تمامی API های Keras (در ابتدا با نام کاری WebKeras توسعه یافته)و همچنین بسیاری از API های سطح پایین TensorFlow را پیاده‌سازی می‌کند. شما می‌توانید به راحتی یک مدل ذخیره‌شده Keras را به TensorFlow.js وارد کنید تا به عنوان بخشی از برنامه جاوا اسکریپت مبتنی بر مرورگر یا برنامه دسکتاپ Electron خود از آن استفاده کنید.

**بهینه‌سازی مدل استنتاج**

بهینه‌سازی مدل شما برای استنتاج اهمیت ویژه‌ای دارد، به ویژه در زمانی که در یک محیط با محدودیت‌های شدید توان و حافظه (مانند گوشی‌های هوشمند و دستگاه‌های تعبیه‌شده) یا برای برنامه‌هایی با نیاز به تأخیر پایین مستقر می‌شود. شما باید همیشه سعی کنید مدل خود را قبل از وارد کردن به TensorFlow.js یا صادر کردن آن به TensorFlow Lite بهینه‌سازی کنید.

دو تکنیک محبوب بهینه‌سازی که می‌توانید اعمال کنید عبارتند از:

 **هرس کردن وزن (Weight Pruning)**: هر ضریب در یک تانسور وزن به طور مساوی به پیش‌بینی‌ها کمک نمی‌کند. می‌توان تعداد پارامترهای لایه‌های مدل خود را با نگه داشتن تنها مهم‌ترین آن‌ها به طور قابل توجهی کاهش داد. این کاهش حافظه و محاسبات مدل شما را کاهش می‌دهد، در حالی که تنها هزینه کوچکی در متریک‌های عملکرد دارد. با تصمیم‌گیری در مورد میزان هرس کردن وزنی که می‌خواهید اعمال کنید، می‌توانید کنترل معامله بین اندازه و دقت را در دست داشته باشید.

 **کمیت‌بندی وزن (Weight Quantization)**: مدل‌های یادگیری عمیق با وزن‌های نقطه‌ شناور با دقت تکی (float32) آموزش داده می‌شوند. با این حال، می‌توان وزن‌ها را به اعداد صحیح ۸-بیتی (int8) کمیت‌بندی کرد تا یک مدل تنها برای استنتاج ایجاد شود که یک چهارم اندازه دارد اما نزدیک به دقت مدل اصلی باقی می‌ماند.

اکوسیستم TensorFlow شامل یک جعبه ابزار هرس وزن و کمیت‌بندی وزن است (www .tensorflow.org/model\_optimization)که به طور عمیق با API کیراس (Keras) یکپارچه شده است.

**6.3.3 مدل خود را در دنیای واقعی نظارت کنید**

شما یک مدل استنتاج را صادر کرده‌اید، آن را در برنامه خود یکپارچه کرده‌اید، و یک اجرای خشک روی داده‌های تولید انجام داده‌اید - مدل دقیقا به همان صورتی که انتظار می‌رفت رفتار کرده است. شما تست‌های واحد و همچنین کد ثبت و نظارت بر وضعیت را نوشته‌اید - عالی است. حالا زمان آن رسیده که دکمه قرمز بزرگ را فشار دهید و به تولید بپردازید.

حتی این نیز پایان کار نیست. پس از استقرار یک مدل، باید به نظارت بر رفتار آن، عملکرد آن بر روی داده‌های جدید، تعامل آن با بقیه برنامه و تاثیر نهایی آن بر روی متریک‌های کسب‌وکار ادامه دهید.

* آیا تعامل کاربران در رادیوی آنلاین شما پس از استقرار سیستم پیشنهادی موسیقی جدید افزایش یا کاهش یافته است؟ آیا نرخ کلیک تبلیغات پس از تغییر به مدل پیش‌بینی نرخ کلیک جدید افزایش یافته است؟ در نظر بگیرید که از آزمون A/B تصادفی برای ایزوله کردن تاثیر مدل خود از دیگر تغییرات استفاده کنید: یک زیرمجموعه از موارد باید از طریق مدل جدید پردازش شود، در حالی که یک زیرمجموعه کنترل دیگر باید به فرایند قدیمی پایبند بماند. هنگامی که به اندازه کافی موارد پردازش شده باشد، تفاوت در نتایج بین دو گروه به احتمال زیاد به مدل نسبت داده می‌شود.
* در صورت امکان، یک بررسی دستی منظم از پیش‌بینی‌های مدل بر روی داده‌های تولیدی انجام دهید. به طور کلی امکان استفاده مجدد از زیرساخت‌های همانند داده‌نگاری وجود دارد: برخی از داده‌های تولیدی را برای داده‌نگاری دستی ارسال کنید و پیش‌بینی‌های مدل را با داده‌های جدید مقایسه کنید. به عنوان مثال، حتما این کار را برای موتور جستجوی تصویر و سیستم پرچم‌گذاری کوکی‌های بد انجام دهید.
* وقتی بررسی‌های دستی امکان‌پذیر نیست، به گزینه‌های ارزیابی جایگزین مانند نظرسنجی‌های کاربری (مثلاً در مورد سیستم پرچم‌گذاری اسپم و محتوای توهین‌آمیز) فکر کنید.

**6.3.4 مدل خود را نگه دارید**

در نهایت، هیچ مدلی برای همیشه دوام ندارد. شما در مورد تغییر مفهوم یاد گرفته‌اید: با گذشت زمان، ویژگی‌های داده‌های تولید شما تغییر خواهد کرد و به تدریج عملکرد و ارتباط مدل شما را کاهش خواهد داد. طول عمر سیستم پیشنهاددهنده موسیقی شما به هفته‌ها خواهد رسید. برای سیستم‌های تشخیص تقلب کارت اعتباری، به روزها خواهد رسید. در بهترین حالت چند سال برای موتور جستجوی تصویر.

به محض اینکه مدل شما راه‌اندازی شد، باید آماده شوید تا نسل بعدی را آموزش دهید که آن را جایگزین کند. در این راستا:

* به تغییرات در داده‌های تولید توجه کنید. آیا ویژگی‌های جدید در دسترس هستند؟ آیا باید مجموعه برچسب‌ها را گسترش دهید یا به‌طور دیگر ویرایش کنید؟
* به جمع‌آوری و داده‌نگاری ادامه دهید و به مرور زمان خط لوله داده‌نگاری خود را بهبود بخشید. به ویژه، باید توجه ویژه‌ای به جمع‌آوری نمونه‌هایی داشته باشید که به نظر می‌رسد برای مدل فعلی شما دشوار باشد تا این گونه نمونه‌ها بتوانند به بهبود عملکرد کمک کنند.

این مراحل فرایند کار جهانی یادگیری ماشین را تکمیل می‌کند - چیزهای زیادی برای به خاطر سپردن وجود دارد. برای تبدیل شدن به یک متخصص زمان و تجربه لازم است، اما نگران نباشید، شما اکنون بسیار عاقل‌تر از آنچه چند فصل پیش بودید هستید. اکنون با تصویر بزرگ آشنا هستید - طیف کاملی از آنچه پروژه‌های یادگیری ماشین شامل می‌شوند. در حالی که بیشتر این کتاب بر توسعه مدل تمرکز دارد، شما اکنون می‌دانید که این تنها یک بخش از کل فرایند است. همیشه تصویر بزرگ را در ذهن داشته باشید!

**خلاصه**

وقتی که یک پروژه یادگیری ماشین جدید را بر عهده می‌گیرید، ابتدا مشکل مورد نظر را تعریف کنید:

* زمینه گسترده‌تری را که قصد دارید در آن فعالیت کنید، درک کنید - هدف نهایی چیست و محدودیت‌ها چه هستند؟
* یک مجموعه داده جمع‌آوری و داده‌نگاری کنید؛ مطمئن شوید که داده‌های خود را به عمق می‌فهمید.
* انتخاب کنید که چگونه موفقیت را برای مشکل خود اندازه‌گیری خواهید کرد - چه متریکی را روی داده‌های اعتبارسنجی خود نظارت خواهید کرد؟

وقتی که مشکل را فهمیدید و یک مجموعه داده مناسب دارید، یک مدل توسعه دهید:

* داده‌های خود را آماده کنید.
* پروتکل ارزیابی خود را انتخاب کنید: اعتبارسنجی بخش نگه‌داشته شده؟ اعتبارسنجی K-fold؟ کدام قسمت از داده‌ها را برای اعتبارسنجی استفاده خواهید کرد؟
* قدرت آماری را به دست آورید: یک خط پایه ساده را شکست دهید.
* مقیاس خود را بزرگ کنید: یک مدل توسعه دهید که بتواند بیش‌برازش کند.
* مدل خود را تنظیم کرده و ابرپارامترهای آن را بر اساس عملکرد روی داده‌های اعتبارسنجی تنظیم کنید. تحقیقات یادگیری ماشین زیادی تمایل دارند تنها بر روی این مرحله تمرکز کنند، اما همیشه تصویر بزرگ را در ذهن داشته باشید.
* وقتی مدل شما آماده شد و عملکرد خوبی روی داده‌های آزمایشی نشان داد، زمان آن رسیده که آن را مستقر کنید:
* ابتدا مطمئن شوید که انتظارات مناسبی با ذی‌نفعان تنظیم کرده‌اید.
* یک مدل نهایی برای استنتاج بهینه‌سازی کرده و آن را به محیط استقرار انتخابی ارسال کنید - سرور وب، موبایل، مرورگر، دستگاه‌های تعبیه‌شده و غیره.
* عملکرد مدل خود را در تولید نظارت کنید و به جمع‌آوری داده‌ها ادامه دهید تا بتوانید نسل بعدی مدل را توسعه دهید.

فصل 7 :

کار با Keras:

یک بررسی عمیق

این فصل پوشش می‌دهد:

* ایجاد مدل‌های Keras با استفاده از کلاس Sequential، API کاربردی، و زیر کلاس‌بندی مدل
* استفاده از حلقه‌های آموزش و ارزیابی داخلی Keras
* استفاده از callback‌های Keras برای سفارشی‌سازی آموزش
* استفاده از TensorBoard برای نظارت بر معیارهای آموزش و ارزیابی
* نوشتن حلقه‌های آموزش و ارزیابی از ابتدا

شما اکنون تجربه ای با Keras دارید—شما با مدل Sequenyial و لایه های Dense و API های داخلی برای اموزش و ارزیابی و استنتاج اشنا هستید—predict() و evaluate() , fit() , compile().شما حتی در فصل ۳ یاد گرفتید که چگونه از کلاس Layer به ارث ببرید تا لایه‌های سفارشی ایجاد کنید و چگونه از TensorFlow GradientTape استفاده کنید تا یک حلقه آموزشی گام به گام را پیاده‌سازی کنید.

در فصل‌های آینده، ما به مباحث بینایی کامپیوتر، پیش‌بینی سری‌های زمانی، پردازش زبان طبیعی و یادگیری عمیق مولد خواهیم پرداخت. این برنامه‌های پیچیده به معماری بیشتر از یک مدل Sequential و حلقه‌ی پیش‌فرض fit() نیاز خواهند داشت. بنابراین ابتدا شما را به یک کارشناس Keras تبدیل می‌کنیم! در این فصل، شما یک مرور کامل از راه‌های کلیدی کار با APIهای Keras دریافت خواهید کرد: هر چیزی که برای رسیدگی به موارد استفاده پیشرفته یادگیری عمیق که با آنها روبرو خواهید شد، نیاز خواهید داشت.

**7.1 طیفی از جریان‌های کاری**

طراحی Keras API با اصول افشای تدریجی پیچیدگی هدایت شده است: شروع کار آسان باشد، اما امکان رسیدگی به موارد استفاده با پیچیدگی بالا را نیز فراهم کند، تنها با نیاز به یادگیری تدریجی در هر مرحله. موارد استفاده ساده باید آسان و قابل دسترسی باشند و جریان‌های کاری پیشرفته دلخواه نیز ممکن باشند: مهم نیست که چقدر نادر و پیچیده باشد، باید مسیر روشنی به آن وجود داشته باشد. مسیری که بر پایه چیزهایی که از جریان‌های کاری ساده‌تر آموخته‌اید ساخته شده باشد. این بدان معناست که شما می‌توانید از مبتدی تا کارشناس رشد کنید و همچنان از همان ابزارها استفاده کنید—فقط به روش‌های مختلف.

به همین دلیل، راه "درست" واحدی برای استفاده از Keras وجود ندارد. بلکه Keras مجموعه‌ای از جریان‌های کاری را ارائه می‌دهد، از بسیار ساده تا بسیار انعطاف‌پذیر. روش‌های مختلفی برای ساخت مدل‌های Keras و روش‌های مختلفی برای آموزش آنها وجود دارد که به نیازهای مختلف پاسخ می‌دهد. از آنجایی که همه این جریان‌های کاری بر اساس APIهای مشترکی مانند Layer و Model هستند، اجزای هر جریان کاری می‌توانند در هر جریان کاری دیگری استفاده شوند—همه می‌توانند با یکدیگر ارتباط برقرار کنند.

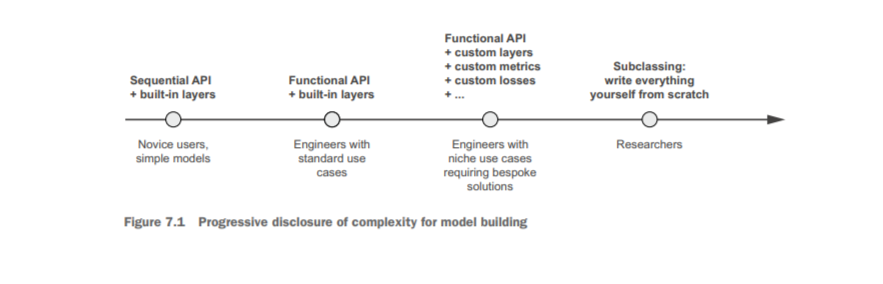
**7.2 روش‌های مختلفی برای ساخت مدل‌های Keras**

سه API برای ساخت مدل‌ها در Keras وجود دارد (نگاه کنید به شکل ۷.۱):

* مدل Sequential، ساده‌ترین API—این مدل اساساً یک لیست Python است. به همین دلیل، به دسته‌های ساده‌ای از لایه‌ها محدود است.
* API تابعی، که بر روی معماری مدل‌های گراف‌مانند تمرکز دارد. این API نقطه میانی خوبی بین قابلیت استفاده و انعطاف‌پذیری است و به همین دلیل، رایج‌ترین API برای ساخت مدل‌ها است.

 زیرکلاس مدل، یک گزینه سطح پایین که در آن همه چیز را خودتان از ابتدا می‌نویسید. این روش ایده‌آل است اگر می‌خواهید کنترل کامل بر هر جزئیات داشته باشید. با این حال، به بسیاری از ویژگی‌های داخلی Keras دسترسی نخواهید داشت و بیشتر در معرض اشتباهات خواهید بود.

دسترسی نخواهید داشت و بیشتر در معرض اشتباهات خواهید بود.

****

**7.2.1 مدل Sequential**

ساده‌ترین راه برای ساخت یک مدل Keras، استفاده از مدل Sequential است، که شما قبلاً با آن آشنا هستید.

فهرست 7.1 کلاس Sequential

from tensorflow import keras

from tensorflow.keras import layers

model = keras.Sequential([

layers.Dense(64, activation="relu"),

layers.Dense(10, activation="softmax")

])

توجه داشته باشید که ساخت مدل به صورت افزایشی از طریق متد add() نیز امکان‌پذیر است، که مشابه با متد append() در لیست‌های Python است.

فهرست 7.2 ساخت افزایشی یک مدل Sequential

model = keras.Sequential()

model.add(layers.Dense(64, activation="relu"))

model.add(layers.Dense(10, activation="softmax"))

شما در فصل ۴ دیدید که لایه‌ها فقط زمانی ساخته می‌شوند (یعنی وزن‌های خود را ایجاد می‌کنند) که برای اولین بار فراخوانی شوند. دلیل این امر این است که شکل وزن‌های لایه‌ها به شکل ورودی آنها بستگی دارد: تا زمانی که شکل ورودی مشخص نشود، نمی‌توانند ایجاد شوند. به همین دلیل، مدل Sequential قبلی هیچ وزنی ندارد (فهرست 7.3) تا زمانی که شما واقعاً آن را روی داده‌هایی فراخوانی کنید یا متد build() آن را با یک شکل ورودی فراخوانی کنید (فهرست 7.4).

فهرست 7.3 مدل‌هایی که هنوز ساخته نشده‌اند وزنی ندارند

>>> model.weights \*در اینجا مدل هنوز ساخته نشده

ValueError: Weights for model sequential\_1 have not yet been created.

فهرست 7.4 فراخوانی یک مدل برای اولین بار برای ساخت آن

>>> model.build(input\_shape=(None, 3)) \* مدل را می‌سازد—حال مدل انتظار نمونه‌هایی با شکل (۳،) را خواهد داشت. None در شکل ورودی نشان می‌دهد که اندازه دسته می‌تواند هر چیزی باشد.

>>> model.weights \* اکنون می‌توانید وزن‌های مدل را بازیابی کنید.

بعد از ساخت مدل، می‌توانید محتوای آن را از طریق متد summary() نمایش دهید که برای دیباگ کردن بسیار مفید است.

فهرست 7.5 متد summary()

>>> model.summary()

Model: "sequential\_1"

Layer (type) Output Shape Param #

my\_first\_layer (Dense) (None, 64) 256

my\_last\_layer (Dense) (None, 10) 650

Total params: 906

Trainable params: 906

Non-trainable params: 0

هنگام ساخت یک مدل Sequential به صورت مرحله‌ای، مفید است که بتوانید پس از اضافه کردن هر لایه، خلاصه‌ای از شکل ظاهری مدل فعلی را چاپ کنید. اما تا زمانی که مدل ساخته نشود، نمی‌توانید یک خلاصه چاپ کنید! در واقع راهی وجود دارد که مدل Sequential خود را به صورت آنلاین ساخته شود: فقط شکل ورودی‌های مدل را از قبل اعلام کنید. می‌توانید این کار را از طریق کلاس Input انجام دهید.

فهرست 7.7 مشخص کردن شکل ورودی مدل خود از قبل

model = keras.Sequential()

model.add(keras.Input(shape=(3,)))

model.add(layers.Dense(64, activation="relu"))

\*از کلاس Input برای اعلام شکل ورودی‌ها استفاده کنید. توجه داشته باشید که آرگومان shape باید شکل هر نمونه را نشان دهد، نه شکل یک دسته (batch) از نمونه‌ها.