**فصل 23- ذخیره و بارگذاری مدل‌های آموزش‌دیده**

**23.0 مقدمه**

در 20 فصل گذشته و حدود 200 دستور العمل، نحوه برداشت داده‌های خام و استفاده از یادگیری ماشین برای ایجاد مدل‌های پیش‌بینی با عملکرد خوب را پوشش داده‌ایم. با این حال، برای اینکه همه‌ی کار ما ارزشمند باشد، در نهایت باید کاری را با مدل خود انجام دهیم، مانند ادغام آن با یک برنامه نرم‌افزاری موجود. برای دستیابی به این هدف، باید بتوانیم هم مدل‌های خود را پس از آموزش ذخیره کنیم و هم در صورت نیاز در یک برنامه، آنها را بارگذاری کنیم. تمرکز این فصل نیز روی همین دو مورد است.

**23.1 ذخیره و بارگذاری یک مدل Sicit-Learn**

**مسئله**

شما یک مدل آموزش‌دیده‌ای دارید که می‌خواهید آن را ذخیره کنید و در جای دیگری بارگذاری کنید.

**راه‌حل**

مدل را به عنوان فایل pickle ذخیره کنید:





هنگامی‌که مدل ذخیره شد، می‌توانیم از scikit-learn در برنامه مقصد خود (به عنوان مثال یک برنامه وب) برای بارگذاری مدل استفاده کنیم:



و از آن برای پیش‌بینی استفاده کنیم:





**بحث**

اولین قدم در استفاده از یک مدل در تولید یک نرم‌افزار، ذخیره آن مدل به عنوان فایلی است که می‌تواند توسط یک برنامه یا در تعامل با برنامه‌های دیگری بارگذاری و استفاده شود. ما می‌توانیم این کار را با ذخیره مدل به عنوان یک فایل pickle، که یک فرمت داده‌ای خاص پایتون است انجام دهیم. به طور خاص، برای ذخیره یک مدل، از joblib استفاده می‌کنیم که کتابخانه‌ای است برای مواردی که آرایه‌های NumPy بزرگ داریم. این یک اتفاق رایج برای مدل‌های آموزش‌دیده در scikit-learn است.

هنگام ذخیره مدل‌های scikit-learn، توجه داشته باشید که مدل‌های ذخیره شده ممکن است با نسخه‌های متفاوت از scikit-learn سازگار نباشند. بنابراین، گنجاندن نسخه scikit-learn مورد استفاده در نام مدل می‌تواند مفید باشد:





**23.2 ذخیره و بارگذاری یک مدل TensorFlow**

**مسئله**

شما یک مدل TensorFlow آموزش دیده دارید و می‌خواهید آن را ذخیره کنید و در جای دیگری بارگذاری کنید.

**راه‌حل**

مدل را با استفاده از فرمت TensorFlow saved\_model ذخیره کنید:



سپس می‌توانیم مدل را در برنامه دیگری یا برای آموزش اضافی بارگذاری کنیم:



**بحث**

اگرچه در طول این کتاب به طور قابل توجهی از TensorFlow استفاده نکردیم، اما دانستن نحوه ذخیره و بارگذاری مدل‌های TensorFlow مفید است. برخلاف scikit-learn که از فرمت ترشی Python-native pickle استفاده می‌کند، TensorFlow روش خود را برای ذخیره و بارگذاری مدل‌ها ارائه می‌دهد. فرمت saved\_model یک دایرکتوری ایجاد می‌کند که مدل و تمام اطلاعات لازم برای بارگیری مجدد آن و همچنین پیش بینی کردن را در فرمت پروتکل بافر[[1]](#footnote-1) (که از پسوند فایل pb. استفاده می‌کند) ذخیره می‌کند.



در حالی که ما به طور عمیق وارد این قالب نمی شویم، این روش استاندارد ذخیره، بارگیری و ارائه مدل‌های آموزش دیده در TensorFlow است.

**همچنین ببینید:**

* [سریال سازی و ذخیره مدل‌های Keras](https://www.tensorflow.org/guide/keras/serialization_and_saving)
* [قالب مدل ذخیره شده TensorFlow](https://www.tensorflow.org/guide/saved_model#the_savedmodel_format_on_disk)

**23.3 ذخیره و بارگذاری یک مدل PyTorch**

**مسئله**

شما یک مدل PyTorch آموزش دیده دارید و می‌خواهید آن را ذخیره کنید و در جای دیگری بارگذاری کنید.

**راه‌حل**

از توابع torch.save و torch.load استفاده کنید:



**بحث**

اگرچه ما از فرمول مشابهی در فصل 21 برای بررسی پیشرفت آموزش خود استفاده کردیم، در اینجا می‌بینیم که چگونه می‌توان از همان رویکرد برای بارگذاری مجدد یک مدل در حافظه برای انجام پیش بینی استفاده کرد. model.pt که مدل را در آن ذخیره می‌کنیم در واقع فقط یک فرهنگ لغت (Dictionary) است که شامل پارامترهای مدل است. ما حالت مدل را در کلیدی به نام model\_state\_dict در این فرهنگ لغت ذخیره کردیم. برای بارگذاری مجدد مدل، شبکه خود را مجدداً مقداردهی اولیه می‌کنیم و وضعیت مدل را با استفاده از تابع network.load\_state\_dict بارگذاری می‌کنیم.

**همچنین ببینید:**

* [آموزش PyTorch: ذخیره و بارگذاری مدل ها](https://pytorch.org/tutorials/beginner/saving_loading_models.html)

**23.4 ارائه مدل‌های sikit-learn**

**مسئله**

شما می‌خواهید با استفاده از یک وب سرور، مدل **sikit-learn** آموزش دیده خود را ارائه دهید.

**راه‌حل**

یک برنامه با استفاده از Python Flask بسازید که مدل آموزش داده شده در این فصل را بارگیری می‌کند:



مطمئن شوید که Flask را نصب کرده اید:



و سپس برنامه را اجرا کنید:



اکنون، می‌توانیم پیش‌بینی‌هایی را در برنامه انجام دهیم و با ارسال داده با استفاده از curl، نتیجه را ببینیم:



**بحث**

در این مثال، ما با استفاده از Flask، یک کتابخانه منبع باز محبوب برای ساخت برنامه‌های تحت وب در پایتون استفاده کردیم. ما یک مسیر به نام /predict تعریف می‌کنیم که داده‌های JSON را در یک درخواست POST می‌گیرد و یک dictionary حاوی پیش بینی‌ها را برمی گرداند. اگرچه این سرور به صورت production نیست (به هشدار Flask در مورد استفاده از سرور توسعه مراجعه کنید)، ما به راحتی می‌توانیم این کد را با یک برنامه تحت وب آماده‌تر برای production گسترش دهیم.

**23.5 ارائه مدل‌های تنسورفلو**

**مسئله**

شما می‌خواهید مدل TensorFlow آموزش دیده خود را با استفاده از یک وب سرور ارائه دهید.

**راه‌حل**

از فریمورک منبع باز TensorFlow Serving و Docker استفاده کنید:



**بحث**

TensorFlow Serving یک راه حل ارائه منبع باز است که برای مدل‌های TensorFlow بهینه شده است. با استفاده از این مورد، به سادگی با ارائه مسیر مدل، یک سرور HTTP و gRPC با ویژگی‌های مفید اضافی برای توسعه‌دهندگان دریافت می‌کنیم.

دستور docker run یک کانتینر را با استفاده از image عمومی tensorflow/serving اجرا می‌کند و مسیر saved\_model دایرکتوری فعلی ما ($(pwd)/saved\_model) را در /models/saved\_model/1 در داخل container ما در برنامه‌ی Docker نصب می‌کند. این عملکرد به طور خودکار مدلی را که قبلاً در این فصل ذخیره کرده بودیم در یک Container در Docker در حال اجرا بارگیری می‌کند که می‌توانیم کوئری‌های پیش بینی را به آن ارسال کنیم.

اگر در مرورگر وب خود به http://localhost:8501/v1/models/saved\_model بروید، باید نتیجه JSON را در اینجا ببینید:



مسیر /metadata در http://localhost:8501/v1/models/saved\_model/metadata اطلاعات بیشتری در مورد مدل ارائه می‌دهد:



ما می‌توانیم با استفاده از curl و دادن متغیرها به آن، با استفاده از برنامه REST پیش بینی انجام دهیم (این شبکه عصبی 10 ویژگی دارد):



**همچنین ببینید:**

* [مستندات TensorFlow: مدل‌های Serving](https://www.tensorflow.org/tfx/guide/serving)

**23.6 ارائه مدل‌های PyTorch در سِلدون[[2]](#footnote-2)**

**مسئله**

شما می‌خواهید یک مدل PyTorch آموزش دیده برای پیش بینی‌های زمان واقعی ارائه دهید.

**راه‌حل**

مدل را با استفاده از Seldon Core Python wrapper ارائه دهید:



و آن را با Docker اجرا کنید:



**بحث**

در حالی که روش‌های مختلفی برای ارائه یک مدل PyTorch وجود دارد، در اینجا ما Seldon Core Python wrapper را انتخاب می‌کنیم. Seldon Core یک چارچوب محبوب برای ارائه مدل‌ها در prodution است و دارای تعدادی ویژگی مفید وجود دارند که استفاده از آن را آسان‌تر و مقیاس‌پذیرتر از یک برنامه Flask می‌کند. این مورد به ما امکان می‌دهد یک کلاس ساده بنویسیم (در کد بالا از MyModel استفاده کردیم)، در حالی که کتابخانه Python به تمام اجزای سرور و آدرس‌های مقصد اهمیت می‌دهد. سپس می‌توانیم سرویس را با استفاده از دستور seldon-core-microservice اجرا کنیم، که سرور REST، سرور gRPC را راه‌اندازی می‌کند و حتی یک آدرس مقصد متریک را نشان می‌دهد. برای انجام پیش‌بینی در این سرویس، می‌توانیم سرویس را با آدرس مقصد زیر در پورت 9000 فراخوانی کنیم:



شما باید خروجی زیر را ببینید:



**همچنین ببینید:**

* [پکیج Seldon Core Python](https://docs.seldon.io/projects/seldon-core/en/latest/python/python_module.html)
* [مستندات TorchServe](https://pytorch.org/serve/)

1. - protocol buffer [↑](#footnote-ref-1)
2. - Seldon [↑](#footnote-ref-2)