**23.2 ذخیره و بارگذاری یک مدل TensorFlow**

**مسئله**

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

**راه‌حل**

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



سپس می توانیم 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 می کند، TensorFlow روش خود را برای ذخیره و بارگذاری مدل ها ارائه می دهد. فرمت saved\_model یک دایرکتوری ایجاد می کند 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 می کند) ذخیره می کند.



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

**همچنین ببینید:**

* [سریال سازی و ذخیره مدل های 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 می خواهید 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 می بینیم که چگونه می توان از همان رویکرد برای بارگذاری مجدد یک مدل در حافظه برای انجام پیش بینی استفاده کرد. model.pt که مدل را در آن ذخیره می کنیم 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 می کنیم و وضعیت مدل را با استفاده از تابع network.load\_state\_dict بارگذاری می کنیم.

**همچنین ببینید:**

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

**23.4 ارائه مدل های sikit-learn**

**مسئله**

شما می خواهید 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 می کند:



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 برنامه های تحت وب در پایتون استفاده کردیم. ما یک مسیر به نام /predict تعریف می کنیم که داده های JSON را در یک درخواست POST می گیرد و یک dictionary حاوی پیش بینی ها 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 مدل های تنسورفلو**

**مسئله**

شما می خواهید 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 مدل های 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 می کند که می توانیم 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 می دهد:



ما می توانیم 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)

**22.4 ارائه مدل های PyTorch در سِلدون[[2]](#footnote-2)**

**مسئله**

شما می خواهید یک مدل PyTorch آموزش دیده برای پیش بینی های 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 می دهد 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)