Lab 3b. Bring your own Script (PyTorch)

- Prerequisites
- Overview
- Background
- Training Data
- Training script
- Run training job in Amazon SageMaker
- Calling fit
- Host
- Evaluate
- Clean up
- Conclusion

Prerequisites

The objective of this lab is to give you step by step instructions on how to bring your custom script to Amazon Sagemaker in your AWS account

- The following steps are an explanation on the cells you will be executing by pressing **Shift+Enter** in an Amazon SageMaker Notebook instance.
- Follow the instructions to launch Amazon SageMaker Studio
- Please ensure that you have git cloned the repository in your SageMaker Studio.

Overview

Amazon SageMaker provides both (1) built-in algorithms and (2) an easy path to train your own custom models. Although the built-in algorithms cover many domains (computer vision, natural language processing etc.) and are easy to use (just provide your data), sometimes training a custom model is the preferred approach. This notebook will focus on training a custom model using PyTorch.

Background

SageMaker Immersion Day

- Prerequisites
- Lab 1. Feature Engineering
  - Option 1: Amazon SageMaker Data Wrangler and Feature Store
  - Option 2: Numpy and Pandas
  - Option 3: Amazon SageMaker Processing
- Lab 2. Train, Tune and Deploy XGBoost
- Lab 3. Bring your own model
  - Lab 3a. Bring your own Script (TensorFlow)
  - **Lab 3b. Bring your own Script (PyTorch)**
  - Lab3c. Bring your own Container
- Lab 4. Autopilot, Debugger and Model Monitor
- Lab 5. Bias and Explainability
- Lab 6. SageMaker Pipelines
- Lab 7. Real Time ML inference on Streaming Data
- Lab 8. Build ML Model with No Code Using Sagemaker Canvas
- Lab 9. Amazon SageMaker

Content preferences

Language

Script mode is a training script format that lets you execute any PyTorch training script in SageMaker with minimal modification. The SageMaker Python SDK handles transferring your script to a SageMaker training instance. On the training instance, SageMaker's native PyTorch support sets up training-related environment variables and executes your training script. In this tutorial, we use the SageMaker Python SDK to launch a training job and deploy the trained model.

In this example, we use a Python script to train a classification model on MNIST dataset. MNIST is a widely used dataset for handwritten digit classification. It consists of 70,000 labeled 28x28 pixel grayscale images of hand-written digits. The dataset is split into 60,000 training images and 10,000 test images. There are 10 classes (one for each of the 10 digits). This tutorial will show how to train and test an MNIST model on SageMaker using PyTorch.

For more information about the PyTorch in SageMaker, please visit sagemaker-pytorch-containers and sagemaker-python-sdk github repositories.

> The notebook contains detailed explanation of each step. This guide helps by providing steps to execute and expected outcomes of each step.

Launch the notebook instance

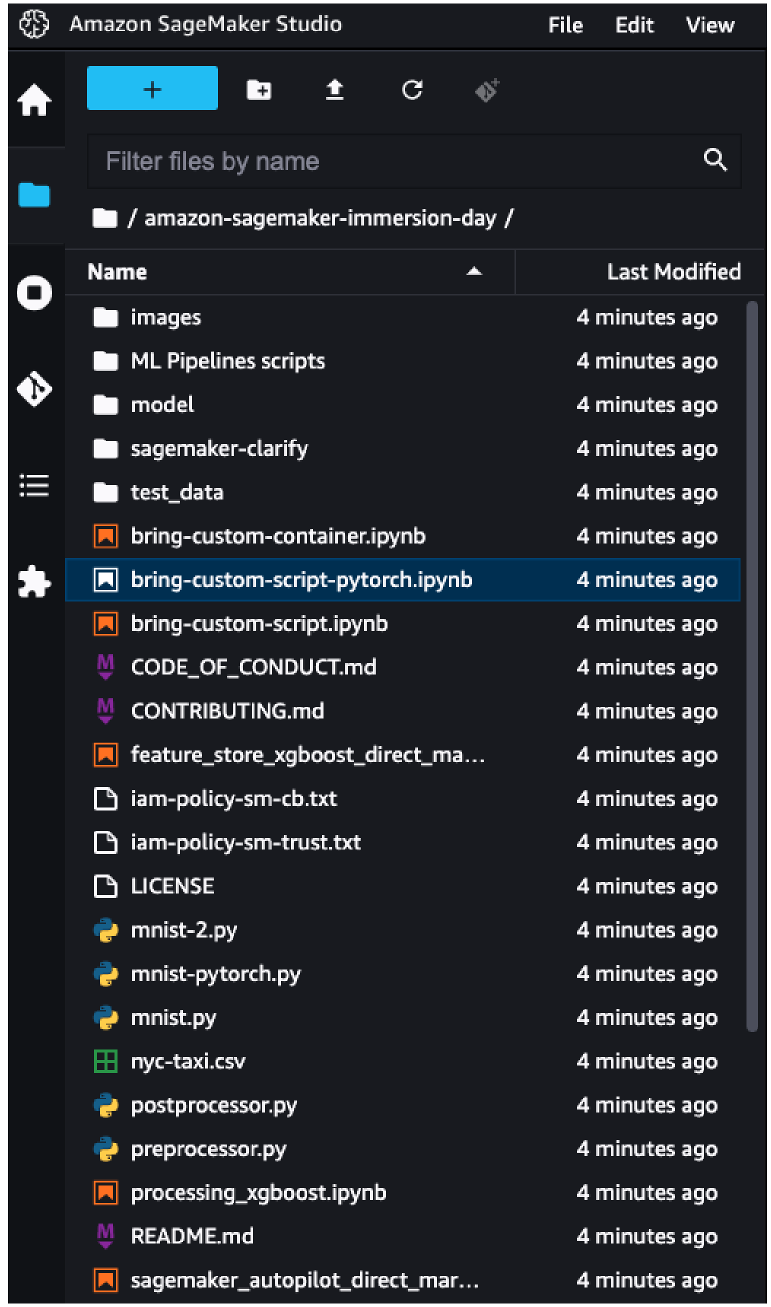
In your SageMaker Studio, in the "File Browser" pane on the left hand side, click on the file "amazon-sagemaker-immersion-day/bring-custom-script-pytorch.ipynb"

SageMaker Immersion Day

- Prerequisites
- Lab 1. Feature Engineering
  - Option 1: Amazon SageMaker Data Wrangler and Feature Store
  - Option 2: Numpy and Pandas
  - Option 3: Amazon SageMaker Processing
- Lab 2. Train, Tune and Deploy XGBoost
- Lab 3. Bring your own model
  - Lab 3a. Bring your own Script (TensorFlow)
  - **Lab 3b. Bring your own Script (PyTorch)**
  - Lab3c. Bring your own Container
- Lab 4. Autopilot, Debugger and Model Monitor
- Lab 5. Bias and Explainability
- Lab 6. SageMaker Pipelines
- Lab 7. Real Time ML inference on Streaming Data
- Lab 8. Build ML Model with No Code Using Sagemaker Canvas
- Lab 9. Amazon SageMaker

Content preferences

Language



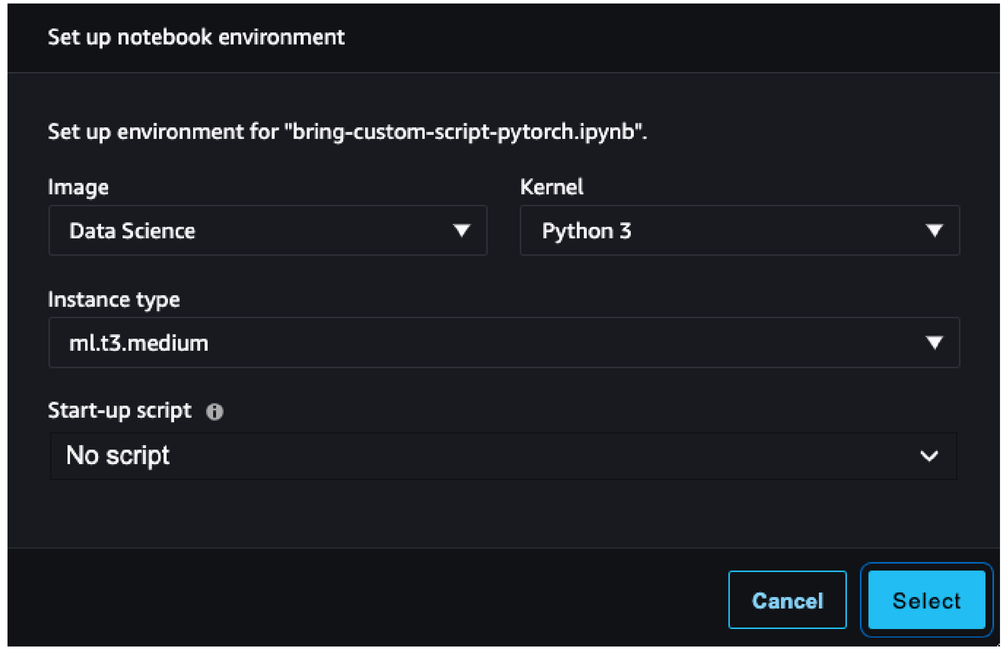
You will be prompted to choose a kernel. Choose **Python 3** Kernel **Data Science** Image.

SageMaker Immersion Day

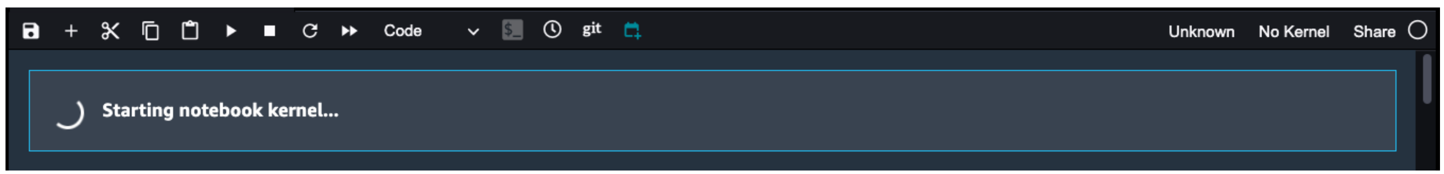
- Prerequisites
- Lab 1. Feature Engineering
  - Option 1: Amazon SageMaker Data Wrangler and Feature Store
  - Option 2: Numpy and Pandas
  - Option 3: Amazon SageMaker Processing
- Lab 2. Train, Tune and Deploy XGBoost
- Lab 3. Bring your own model
  - Lab 3a. Bring your own Script (TensorFlow)
  - **Lab 3b. Bring your own Script (PyTorch)**
  - Lab3c. Bring your own Container
- Lab 4. Autopilot, Debugger and Model Monitor
- Lab 5. Bias and Explainability
- Lab 6. SageMaker Pipelines
- Lab 7. Real Time ML inference on Streaming Data
- Lab 8. Build ML Model with No Code Using Sagemaker Canvas
- Lab 9. Amazon SageMaker

Content preferences

Language



The notebook kernel will take a while to start:



After that, will be able to verify that the right kernel is selected on the top right of the screen:

Set up

Execute the contents of cell 01 (click on the cell and then key Shift+Enter to execute) to install torchvision library. This can take some time to complete execution

```
[2]: !yes | pip uninstall torchvision
     !pip install -qU torchvision
```

SageMaker Immersion Day

- Prerequisites
- Lab 1. Feature Engineering
  - Option 1: Amazon SageMaker Data Wrangler and Feature Store
  - Option 2: Numpy and Pandas
  - Option 3: Amazon SageMaker Processing
- Lab 2. Train, Tune and Deploy
  - Lab 3a. Bring your own Script (TensorFlow)
  - **Lab 3b. Bring your own Script (PyTorch)**
  - Lab3c. Bring your own Container
- Lab 4. Autopilot, Debugger and Model Monitor
- Lab 5. Bias and Explainability
- Lab 6. SageMaker Pipelines
- Lab 7. Real Time ML inference on Streaming Data
- Lab 8. Build ML Model with No Code Using Sagemaker Canvas
- Lab 9. Amazon SageMaker

Content preferences

Language

Execute the contents of cell 02. Please note, there will be no output and it will execute immediately

```
[3]: import sagemaker

     sagemaker_session = sagemaker.Session()

     bucket = sagemaker_session.default_bucket()
     prefix = "sagemaker/pytorch-mnist"

     role = sagemaker.get_execution_role()
```

Training Data

Execute the contents of cell 03 to get data and apply couple of transformations on it

```
[4]: from torchvision.datasets import MNIST
     from torchvision import transforms

     MNIST.mirrors = ["https://sagemaker-sample-files.s3.amazonaws.com/datasets/image/MNIST/"]

     MNIST(
         "lab03-pytorch-data",
         download=True,
         transform=transforms.Compose(
             [transforms.ToTensor(), transforms.Normalize((0.1307,), (0.3081,))]
         ),
     )
     Downloading https://sagemaker-sample-files.s3.amazonaws.com/datasets/image/MNIST/train-images-idx3-ubyte.gz
```

```
     Extracting lab03-pytorch-data/MNIST/raw/train-images-idx3-ubyte.gz to lab03-pytorch-data/MNIST/raw

     Downloading https://sagemaker-sample-files.s3.amazonaws.com/datasets/image/MNIST/train-labels-idx1-ubyte.gz
     Downloading https://sagemaker-sample-files.s3.amazonaws.com/datasets/image/MNIST/train-labels-idx1-ubyte.gz to lab03-pytorch-data/MNIST/raw/train-labels-idx1-ubyte.gz

     100% 28881/28881 [00:00<00:00, 1960092.78it/s]

     Extracting lab03-pytorch-data/MNIST/raw/train-labels-idx1-ubyte.gz to lab03-pytorch-data/MNIST/raw

     Downloading https://sagemaker-sample-files.s3.amazonaws.com/datasets/image/MNIST/t10k-images-idx3-ubyte.gz
     Downloading https://sagemaker-sample-files.s3.amazonaws.com/datasets/image/MNIST/t10k-images-idx3-ubyte.gz to lab03-pytorch-data/MNIST/raw/t10k-images-idx3-ubyte.gz

     100% 1648877/1648877 [00:00<00:00, 2909577.61it/s]
```

Upload to Amazon S3

Execute contents of cell 04 to upload data to Amazon S3

```
[5]: inputs = sagemaker_session.upload_data(path="lab03-pytorch-data", bucket=bucket, key_prefix=prefix)
     print("input spec (in this case, just an S3 path): {}".format(inputs))
```

Training Script

Execute the contents of cell 05. It will take about a second to execute and then it will output the contents of the files `mnist-pytorch.py`. These files are in the root of the folder "amazon-sagemaker-immersion-day/"

```
[6]: !pygmentize mnist-pytorch.py

import argparse
import json
import logging
import os
import sys

#import sagemaker_containers
import torch
import torch.distributed as dist
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torch.utils.data
import torch.utils.data.distributed
from torchvision import datasets, transforms

logger = logging.getLogger(__name__)
logger.setLevel(logging.DEBUG)
logger.addHandler(logging.StreamHandler(sys.stdout))
```


Run training job in Amazon SageMaker

Execute the contents of cell 06. It will take about a second to execute. The `sagemaker.pytorch.PyTorch` estimator handles locating the script mode container, uploading your script to a S3 location and creating a SageMaker training job. Let's call out a couple important parameters here:

- `py_version` is set to `'py38'` to indicate that we want to use Python version 3.8 for executing the training code. You can use this parameter to specify different version as neededYou can use this parameter to specify different version as needed
- `hyperparameters` is used to pass hyperparameters to the training script and are accessed using the `argsparse.ArgumentParser` instance. The hyperparameter backend specified in script the backend to use for distributed training. Since we are performing distributed training using CPUs `gloo` is passed as argument. You can also specify `nccl` as backend for example when distributing across GPUs instances. For additional details on backend options supported in PyTorch refer to documentation here.

```
[8]: from sagemaker.pytorch import PyTorch

estimator = PyTorch(
    entry_point="mnist-pytorch.py",
    role=role,
    py_version="py38",
    framework_version="1.11.0",
    instance_count=2,
    instance_type="ml.c5.2xlarge",
    hyperparameters={"epochs": 1, "backend": "gloo"},
)
```

Calling fit

To start a training job, we call `estimator.fit({"training": inputs})`. Execute the contents of cell 07. It will take several minutes to execute.

SageMaker Immersion Day

- Prerequisites
- Lab 1. Feature Engineering
  - Option 1: Amazon SageMaker Data Wrangler and Feature Store
  - Option 2: Numpy and Pandas
  - Option 3: Amazon SageMaker Processing
- Lab 2. Train, Tune and Deploy XGBoost
- Lab 3. Bring your own model
  - Lab 3a. Bring your own Script (TensorFlow)
  - **Lab 3b. Bring your own Script (PyTorch)**
  - Lab3c. Bring your own Container
- Lab 4. Autopilot, Debugger and Model Monitor
- Lab 5. Bias and Explainability
- Lab 6. SageMaker Pipelines
- Lab 7. Real Time ML inference on Streaming Data
- Lab 8. Build ML Model with No Code Using Sagemaker Canvas
- Lab 9. Amazon SageMaker

Content preferences

Language

```
[*]: estimator.fit({"training": inputs})

INFO:sagemaker.image_uris:image_uri is not presented, retrieving image_uri based on instance_type, framework etc.
INFO:sagemaker:Creating training-job with name: pytorch-training-2023-03-20-00-02-32-142
2023-03-20 00:02:37 Starting - Starting the training job..
```

At the end, it will show an output similar to the following:

```
2023-03-19 20:05:13,977 sagemaker-training-toolkit INFO     Done waiting for a return code. Received 0 from exiting process.
2023-03-19 20:05:13,977 sagemaker-training-toolkit INFO     Reporting training SUCCESS

2023-03-20 00:05:32 Uploading - Uploading generated training model
2023-03-20 00:05:32 Completed - Training job completed
Training seconds: 226
Billable seconds: 226
```

Host

Create endpoint

Execute the contents of cell 08 to deploy the trained model. It will take several minutes to execute.

```
[10]: predictor = estimator.deploy(initial_instance_count=1, instance_type="ml.m4.xlarge")

INFO:sagemaker:Creating model with name: pytorch-training-2023-03-20-00-38-25-414
INFO:sagemaker:Creating endpoint-config with name pytorch-training-2023-03-20-00-38-25-414
INFO:sagemaker:Creating endpoint with name pytorch-training-2023-03-20-00-38-25-414
----------!
```

Evaluate

Execute the contents of the cell 09 to import test images for evaluating the endpoint.

```
[12]: import gzip
import numpy as np
import random
import os

data_dir = "lab03-pytorch-data/MNIST/raw"
with gzip.open(os.path.join(data_dir, "t10k-images-idx3-ubyte.gz"), "rb") as f:
    images = np.frombuffer(f.read(), np.uint8, offset=16).reshape(-1, 28, 28).astype(np.float32)

mask = random.sample(range(len(images)), 16)  # randomly select some of the test images
mask = np.array(mask, dtype=int)
data = images[mask]
```

SageMaker
Immersion Day

- Prerequisites
- Lab 1. Feature Engineering
  - Option 1: Amazon SageMaker Data Wrangler and Feature Store
  - Option 2: Numpy and Pandas
  - Option 3: Amazon SageMaker Processing
- Lab 2. Train, Tune and Deploy XGBoost
- Lab 3. Bring your own model
  - Lab 3a. Bring your own Script (TensorFlow)
  - **Lab 3b. Bring your own Script (PyTorch)**
  - Lab3c. Bring your own Container
- Lab 4. Autopilot, Debugger and Model Monitor
- Lab 5. Bias and Explainability
- Lab 6. SageMaker Pipelines
- Lab 7. Real Time ML inference on Streaming Data
- Lab 8. Build ML Model with No Code Using Sagemaker Canvas
- Lab 9. Amazon SageMaker

Content preferences

Language

Now, run the predictions by executing cell 10 which takes about a second to execute

```
[14]: response = predictor.predict(np.expand_dims(data, axis=1))
print("Raw prediction result:")
print(response)
print()

labeled_predictions = list(zip(range(10), response[0]))
print("Labeled predictions: ")
print(labeled_predictions)
print()

labeled_predictions.sort(key=lambda label_and_prob: 1.0 - label_and_prob[1])
print("Most likely answer: {}".format(labeled_predictions[0]))
```

```
Raw prediction result:
[[ -871.36743164  -971.89575195  -913.92596436  -654.67346191
    -97.44143677  -498.67459106  -707.69451904  -301.13406372
   -463.44345093     0.        ]
 [-1198.8449707  -1175.25219727  -936.64807129  -718.03765869
   -766.48907471 -1127.92248535 -1482.80273438     0.
   -904.03717041  -280.87255859]
 [ -915.80487061     0.          -370.35418701  -575.47436523
   -504.13796997  -668.82928467  -604.71899414  -464.13146973
   -392.08587646  -507.47665405]
 [ -425.44076538  -467.09265137  -282.55163574  -510.90609741
   -115.01202393  -289.24203491     0.          -542.6862793
   -380.04800415  -338.26568604]
```

Clean up

After analyzing the results, you can terminate the endpoints by executing cells 11. Optionally, you can use AWS console to verify that the endpoints are deleted.

```
[24]: sagemaker_session.delete_endpoint(endpoint_name=predictor.endpoint_name)
```

Conclusion

In this tutorial, we use the SageMaker Python SDK to launch a training job and deploy the trained model. On the training instance, SageMaker's native PyTorch support sets up training-related environment variables and executes your training script in file `mnist-pytorch.py`

Previous Next