AWS Tutorial: AWS SageMaker

Time spent: 60 min

Amazon Overview

Amazon SageMaker is a fully managed machine learning service. With Amazon SageMaker, data scientists and developers can quickly and easily build and train machine learning models, and then directly deploy them into a production-ready hosted environment. It provides an integrated Jupyter authoring notebook instance for easy access to your data sources for exploration and analysis, so you don't have to manage servers. It also provides common machine learning algorithms that are optimized to run efficiently against extremely large data in a distributed environment. With native support for bring-your-own-algorithms and frameworks, Amazon SageMaker offers flexible distributed training options that adjust to your specific workflows. Deploy a model into a secure and scalable environment by launching it with a single click from the Amazon SageMaker console. Training and hosting are billed by minutes of usage, with no minimum fees and no upfront commitments.

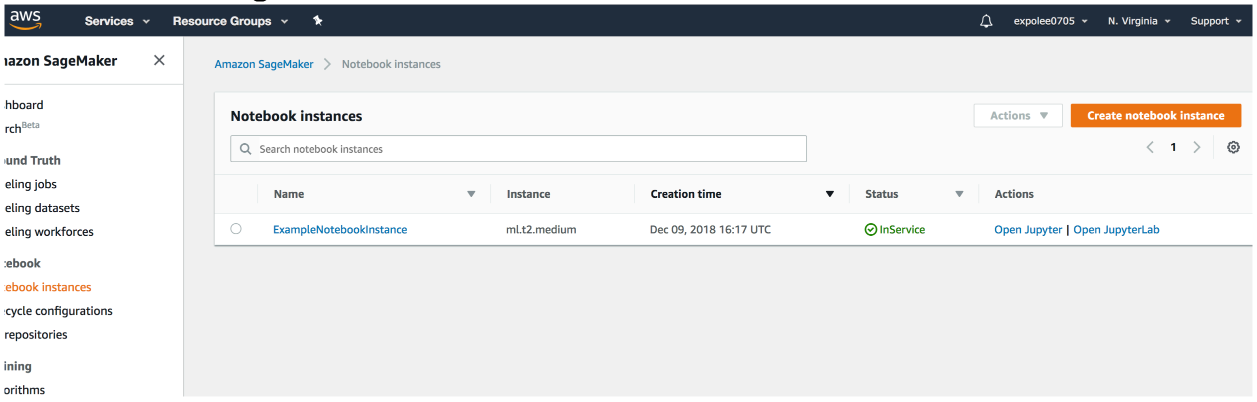
– From AWS document

What I learned

- What is Amazon SageMaker
- How to create the notebook to operate the instance
- How to do the training job by high-level Python library
- Related components: AWS SageMaker, S3, jupyer

What I built

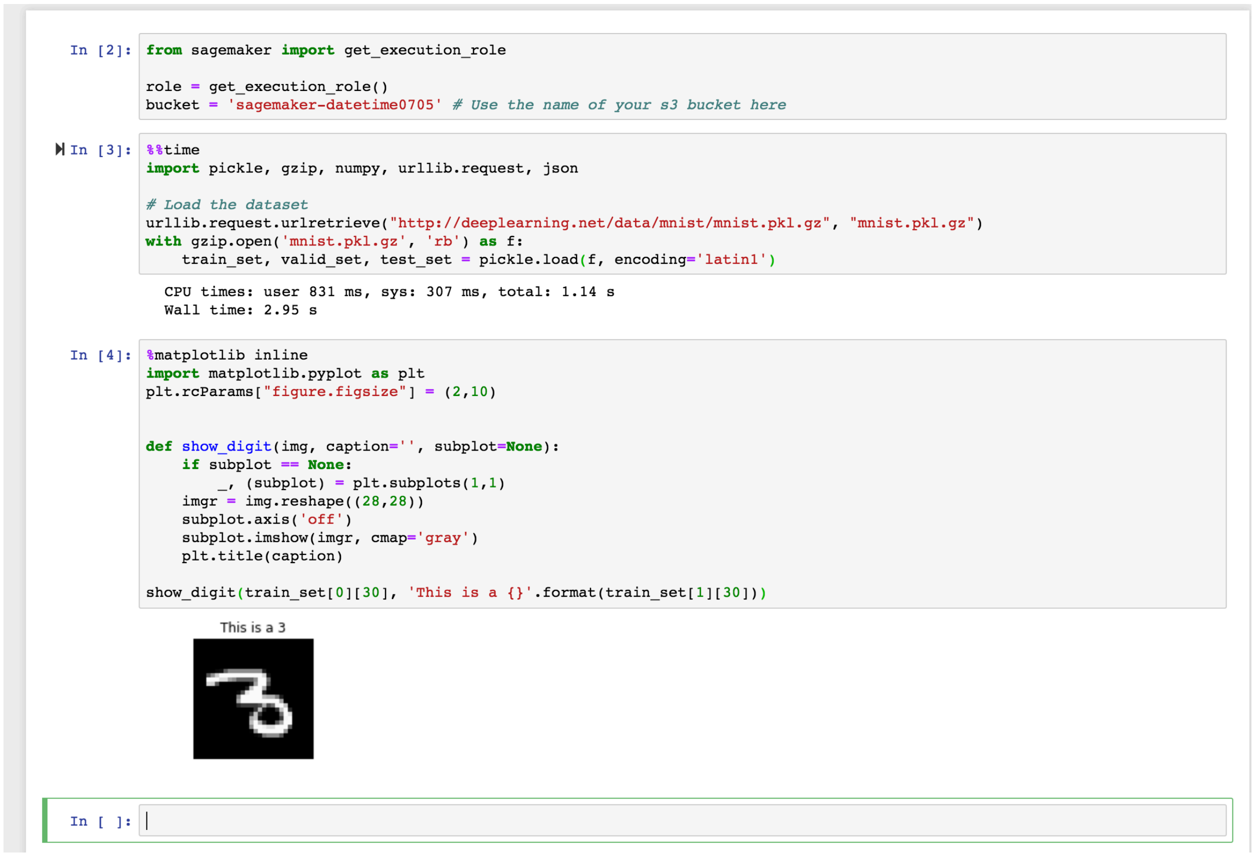
- Create the SageMaker instance



Notebook instances

Name	Instance	Creation time	Status	Actions
ExampleNotebookInstance	ml.t2.medium	Dec 09, 2018 16:17 UTC	InService	Open Jupyter | Open JupyterLab

- Create the notebook to operate the instance to train the data



```
In [2]: from sagemaker import get_execution_role

        role = get_execution_role()
        bucket = 'sagemaker-datetime0705' # Use the name of your s3 bucket here
```

```
In [3]: %%time
        import pickle, gzip, numpy, urllib.request, json

        # Load the dataset
        urllib.request.urlretrieve("http://deeplearning.net/data/mnist/mnist.pkl.gz", "mnist.pkl.gz")
        with gzip.open('mnist.pkl.gz', 'rb') as f:
            train_set, valid_set, test_set = pickle.load(f, encoding='latin1')

        CPU times: user 831 ms, sys: 307 ms, total: 1.14 s
        Wall time: 2.95 s
```

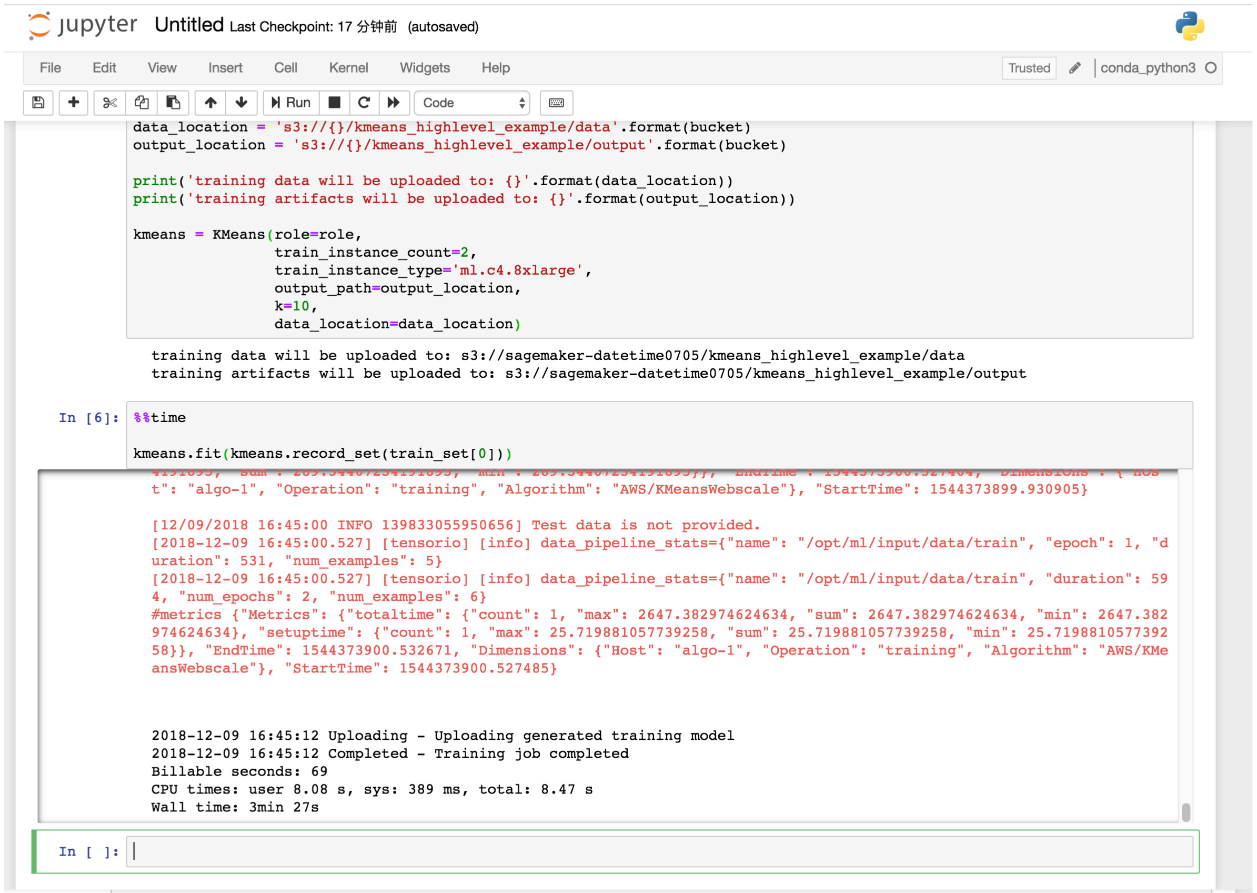
```
In [4]: %matplotlib inline
        import matplotlib.pyplot as plt
        plt.rcParams["figure.figsize"] = (2,10)


        def show_digit(img, caption='', subplot=None):
            if subplot == None:
                _, (subplot) = plt.subplots(1,1)
            imgr = img.reshape((28,28))
            subplot.axis('off')
            subplot.imshow(imgr, cmap='gray')
            plt.title(caption)

        show_digit(train_set[0][30], 'This is a {}'.format(train_set[1][30]))
```

```
In [ ]:
```

- Deploy and validate the model



```
data_location = 's3://{}/kmeans_highlevel_example/data'.format(bucket)
output_location = 's3://{}/kmeans_highlevel_example/output'.format(bucket)

print('training data will be uploaded to: {}'.format(data_location))
print('training artifacts will be uploaded to: {}'.format(output_location))

kmeans = KMeans(role=role,
                train_instance_count=2,
                train_instance_type='ml.c4.8xlarge',
                output_path=output_location,
                k=10,
                data_location=data_location)
```

```
training data will be uploaded to: s3://sagemaker-datetime0705/kmeans_highlevel_example/data
training artifacts will be uploaded to: s3://sagemaker-datetime0705/kmeans_highlevel_example/output
```

In [6]:

```
%%time

kmeans.fit(kmeans.record_set(train_set[0]))
```

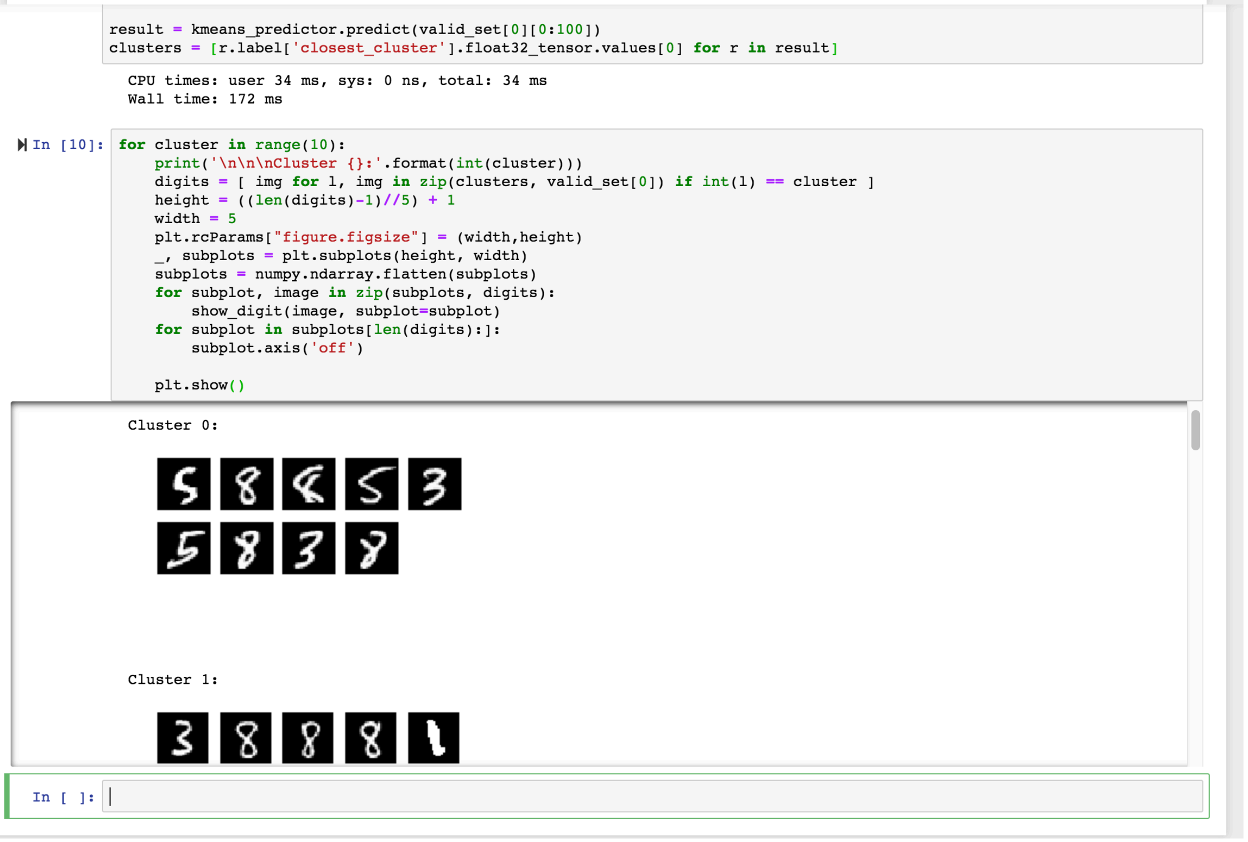
```
t": "algo-1", "Operation": "training", "Algorithm": "AWS/KMeansWebscale"}, "StartTime": 1544373899.930905}

[12/09/2018 16:45:00 INFO 139833055950656] Test data is not provided.
[2018-12-09 16:45:00.527] [tensorio] [info] data_pipeline_stats={"name": "/opt/ml/input/data/train", "epoch": 1, "d
uration": 531, "num_examples": 5}
[2018-12-09 16:45:00.527] [tensorio] [info] data_pipeline_stats={"name": "/opt/ml/input/data/train", "duration": 59
4, "num_epochs": 2, "num_examples": 6}
#metrics {"Metrics": {"totaltime": {"count": 1, "max": 2647.382974624634, "sum": 2647.382974624634, "min": 2647.382
974624634}, "setuptime": {"count": 1, "max": 25.719881057739258, "sum": 25.719881057739258, "min": 25.7198810577392
58}}, "EndTime": 1544373900.532671, "Dimensions": {"Host": "algo-1", "Operation": "training", "Algorithm": "AWS/KMe
ansWebscale"}, "StartTime": 1544373900.527485}



2018-12-09 16:45:12 Uploading - Uploading generated training model
2018-12-09 16:45:12 Completed - Training job completed
Billable seconds: 69
CPU times: user 8.08 s, sys: 389 ms, total: 8.47 s
Wall time: 3min 27s
```

In [ ]:



```
result = kmeans_predictor.predict(valid_set[0][0:100])
clusters = [r.label['closest_cluster'].float32_tensor.values[0] for r in result]
```

```
CPU times: user 34 ms, sys: 0 ns, total: 34 ms
Wall time: 172 ms
```

In [10]:

```
for cluster in range(10):
    print('\n\n\nCluster {}:'.format(int(cluster)))
    digits = [ img for l, img in zip(clusters, valid_set[0]) if int(l) == cluster ]
    height = ((len(digits)-1)//5) + 1
    width = 5
    plt.rcParams["figure.figsize"] = (width,height)
    _, subplots = plt.subplots(height, width)
    subplots = numpy.ndarray.flatten(subplots)
    for subplot, image in zip(subplots, digits):
        show_digit(image, subplot=subplot)
    for subplot in subplots[len(digits):]:
        subplot.axis('off')

    plt.show()
```

```
Cluster 0:
```

```
Cluster 1:
```

In [ ]: