

Hw4 書面報告

生機一

陳玟鉸

R09631011

AWS EMR

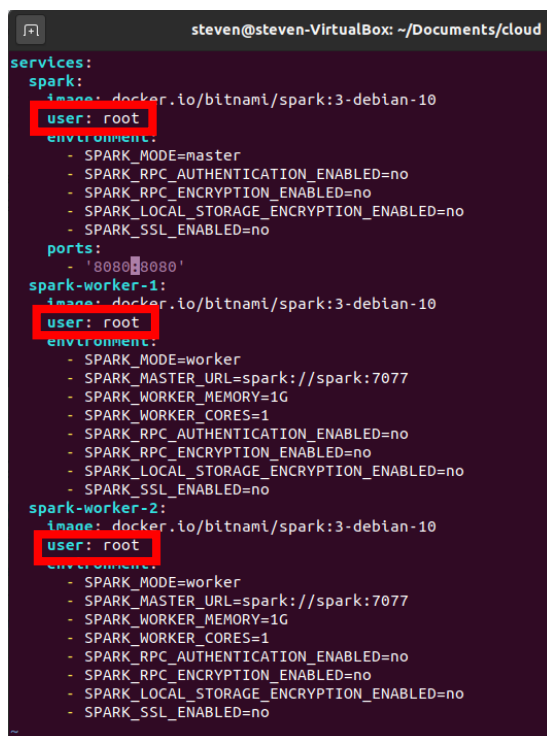
這次作業一開始選擇在 EMR 上操作，但當“叢集”與資料都上傳後，隔天因為 AWS 美東伺服器當機緣故，因此無法繼續完成作業，所以只嘗試在本機上重新做一次作業。附上這次作業的 GitHub URL：

https://github.com/WenAnChen/Cloud_computing/tree/main/Hw4

(一) 本機上 reproduce

在本機上操作時遇到不少 bug，首先確認 python 版本，但因為限定要 python2.7，因此再次安裝 python2。

1. 參照之前 Hw1 跟 Hw2，以 docker-compose 的方式啟動 spark 的 container



```
services:
  spark:
    image: docker.io/bitnami/spark:3-debian-10
    user: root
    environment:
      - SPARK_MODE=master
      - SPARK_RPC_AUTHENTICATION_ENABLED=no
      - SPARK_RPC_ENCRYPTION_ENABLED=no
      - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
      - SPARK_SSL_ENABLED=no
    ports:
      - '8080:8080'
  spark-worker-1:
    image: docker.io/bitnami/spark:3-debian-10
    user: root
    environment:
      - SPARK_MODE=worker
      - SPARK_MASTER_URL=spark://spark:7077
      - SPARK_WORKER_MEMORY=1G
      - SPARK_WORKER_CORES=1
      - SPARK_RPC_AUTHENTICATION_ENABLED=no
      - SPARK_RPC_ENCRYPTION_ENABLED=no
      - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
      - SPARK_SSL_ENABLED=no
  spark-worker-2:
    image: docker.io/bitnami/spark:3-debian-10
    user: root
    environment:
      - SPARK_MODE=worker
      - SPARK_MASTER_URL=spark://spark:7077
      - SPARK_WORKER_MEMORY=1G
      - SPARK_WORKER_CORES=1
      - SPARK_RPC_AUTHENTICATION_ENABLED=no
      - SPARK_RPC_ENCRYPTION_ENABLED=no
      - SPARK_LOCAL_STORAGE_ENCRYPTION_ENABLED=no
      - SPARK_SSL_ENABLED=no
```

左圖為 socker-compose.yml，在三個 image 的下方都增加 user: root，使得之後在 container 中使用者可被辨識。

```

steven@steven-VirtualBox:~/Documents/cloud$ sudo docker container ls
[sudo] password for steven:
CONTAINER ID        IMAGE               COMMAND                  CREATED      STATUS      PORTS                    NAMES
d5568cd412a6       bitnami/spark:3-debian-10  "/opt/bitnami/script..." 7 hours ago  Up 7 hours  0.0.0.0:8080->8080/tcp    cloud_spark_1
39f9cb99325a       bitnami/spark:3-debian-10  "/opt/bitnami/script..." 7 hours ago  Up 7 hours  cloud_spark-worker-1_1    cloud_spark-worker-1_1
4d1f657fd64e       bitnami/spark:3-debian-10  "/opt/bitnami/script..." 7 hours ago  Up 7 hours  cloud_spark-worker-2_1    cloud_spark-worker-2_1

```

在 docker-compose up 之後，能發現有三個 container 被喚醒，其中一個為 spark 架構中的 master(cloud_spark_1)，其他兩個為 worker(cloud_spark-worker-1_1/2-1)。

- 我選擇在 container cloud_spark-worker-1_1 中執行作業，並在這 container 中將這次要訓練的資料 data_banknote_authentication.txt 及程式 run-spark-ex.py 複製進去
- 為了讓程式中能使用 np，在下載 numpy 時，發現指令 `sudo apt install python-numpy` 無法正確讓 pyspark 使用 numpy。需要透過 `sudo pip install numpy` 才能使用，目前還沒去了解原因。
- 修改程式碼：
在 container 中開啟 pyspark，好處是能在每行程式後都執行一次，來檢視是否有 bug。



為了使 pyspark 能使用 python2.7，先將預設路徑改掉：

```
export pyspark_python=/usr/bin/python2.7
```

```
export pyspark_driver_python=/usr/bin/python2.7
```

```
export spark_yarn_user_env="pyspark_python=/usrbin/python2.7"
```

修改 run-spark-ex.py 中的程式碼：

- 增加 `from pyspark.mllib.regression import LabeledPoint`
- 修改 data 的位置
- 將 `sc = SparkContext(conf = conf)` 改成 `sc = pyspark.SparkContext()`
- 將 mapper 中的 `np.array(features)` 改成 `LabeledPoint(label, features)`

此時在執行下一行程式：`sc = getSparkContext()`時出錯，茶道原因為 spark 重複喚醒，因此需要先停止前一次的 spark，因此加入程式 `sc.stop()`

```
>>> sc = getSparkContext()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 3, in getSparkContext
  File "/opt/bitnami/spark/python/pyspark/context.py", line 133, in __init__
    SparkContext._ensure_initialized(self, gateway=gateway, conf=conf)
  File "/opt/bitnami/spark/python/pyspark/context.py", line 341, in _ensure_initialized
    callsite.function, callsite.file, callsite.linenum))
ValueError: Cannot run multiple SparkContexts at once; existing SparkContext(app=PySparkShell, master=local[*])
created by <module> at /opt/bitnami/spark/python/pyspark/shell.py:41
>>> sc = SparkContext().getOrCreate();
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/opt/bitnami/spark/python/pyspark/context.py", line 133, in __init__
    SparkContext._ensure_initialized(self, gateway=gateway, conf=conf)
  File "/opt/bitnami/spark/python/pyspark/context.py", line 341, in _ensure_initialized
    callsite.function, callsite.file, callsite.linenum))
ValueError: Cannot run multiple SparkContexts at once; existing SparkContext(app=PySparkShell, master=local[*])
created by <module> at /opt/bitnami/spark/python/pyspark/shell.py:41
>>> sc = SparkContext().getOrCreate(conf = conf);
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/opt/bitnami/spark/python/pyspark/context.py", line 133, in __init__
    SparkContext._ensure_initialized(self, gateway=gateway, conf=conf)
  File "/opt/bitnami/spark/python/pyspark/context.py", line 341, in _ensure_initialized
    callsite.function, callsite.file, callsite.linenum))
ValueError: Cannot run multiple SparkContexts at once; existing SparkContext(app=PySparkShell, master=local[*])
created by <module> at /opt/bitnami/spark/python/pyspark/shell.py:41
>>> sc.stop()
>>> sc = getSparkContext()
```

(5) 將 `labelsAndPreds` 中的 `point.item(0)`及 `take(range(1, point.size))`改成 `point.label` 與 `point.features`

最後在執行 `print("Training Error = " + str(trainErr))`得到結果輸出：

```
Training Error = 0.0443681245768
```

發現此結果每次都會略有不同，猜測是因為在資料做機器學習時每次的梯度下降會是不同的所導致。

(二) own SGD

運用 MXnet 中的 `gluon` 來做為模型，且將資料拆成訓練資料及測試資料，計算經過隨機梯度下降演算法後的結果。

```
import mxnet as mx
from mxnet import gluon
```

資料讀取後座正規化(0~1)

```
def load_data(self):
def transform(data, label):
return data.astype(np.float32) / 255., label.astype(np.float32)
train_data = DataLoader(MNIST(train=True, transform=transform),
self.batch_size, shuffle=True)
test_data = DataLoader(MNIST(train=False, transform=transform),
self.batch_size, shuffle=False)
return train_data, test_data
```

建立 Sequential() 序列，Sequential 是全部操作單元的容器，其中 net.name_scope() 能為 Sequential 中的操作單元自動新增名稱。

```
def model(self):
num_hidden = 64
net = gluon.nn.Sequential()
with net.name_scope():
net.add(gluon.nn.Dense(units=num_hidden, activation="relu"))
net.add(gluon.nn.Dense(units=num_hidden, activation="relu"))
net.add(gluon.nn.Dense(units=self.num_outputs))
net.collect_params().initialize(init=mx.init.Normal(sigma=.1), ctx=self.model_ctx)
return net
```

參數設定

```
epochs = 200
smoothing_constant = 0.01
num_examples = 1000
```

在訓練一次 epoch 之後，計算測試和訓練資料的準確率，不斷進行迴圈直到完成全部的 epochs 為止。

因為太晚開始寫作業，並因為 EMR 當機造成多花了許多時間在做第一題，因此還沒辦法完全解決第二題的 bug，還無法順利測資。

可預期的是：與深度學習架構的 MXnet 相比，Spark 在兩層的神經穩路上浪費許多效能，造成運算速度及準確度都會顯得較不足。