Cloud Computing and Cyber Security

(NTU, Fall 2020)

Homework 4: Spark - Logistic Regression

李承翰 P08942A01

1. **Use Online resource to build Spark env on docker**

   - **Download Spark Cluster**

   >> git clone https://github.com/mvillarrealb/docker-spark-cluster.git

   >> cd docker-spark-cluster

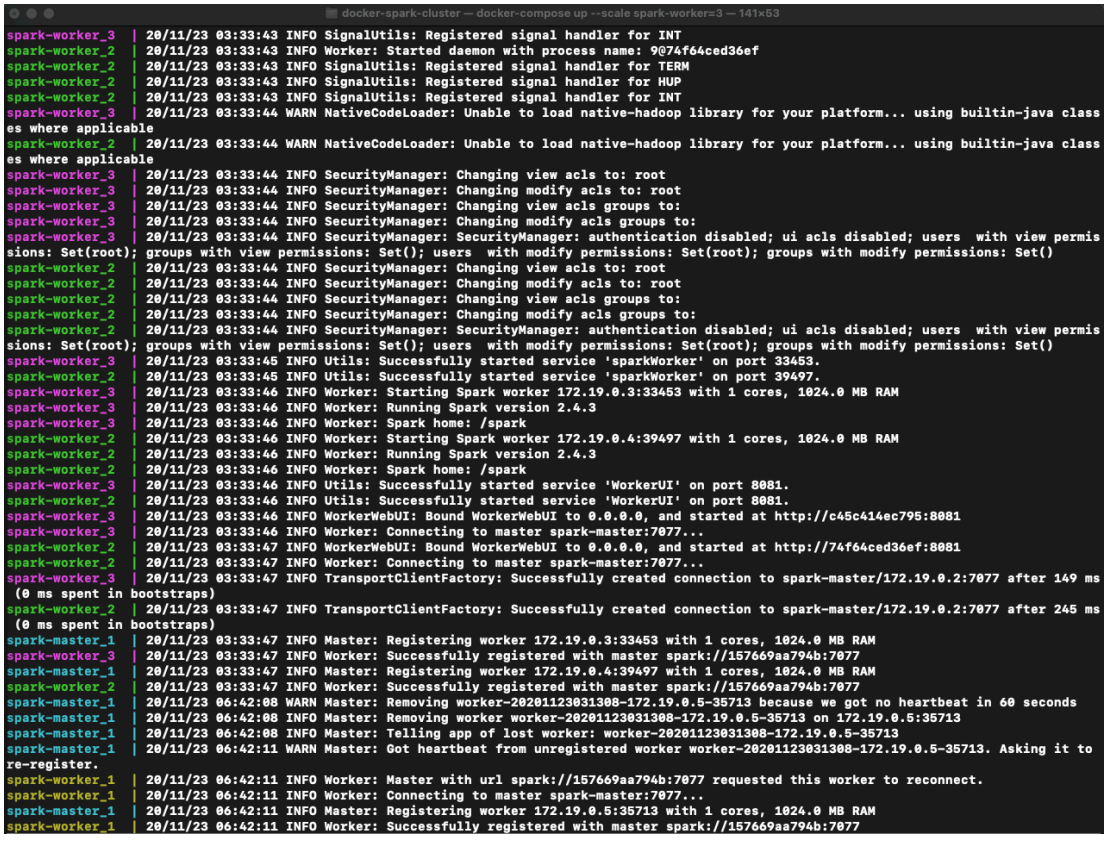
   >> chmod +x build-images.sh

   >> ./build-images.sh

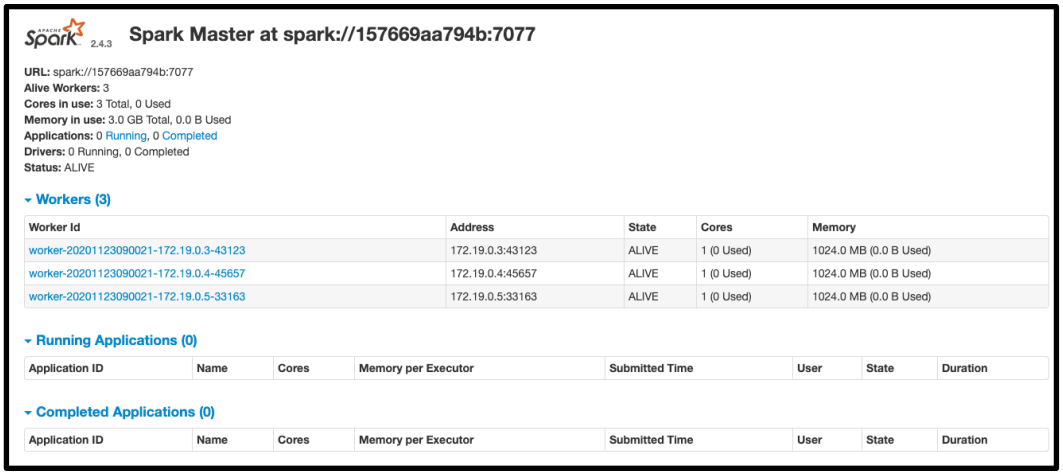
   - **The Docker compose will create the following containers:**

Container	IP address
spark-master	10.5.0.2
spark-worker-1	10.5.0.3
spark-worker-2	10.5.0.4
spark-worker-3	10.5.0.5

>> docker-compose up --scale spark-worker=3



- Validate your cluster accessing the spark UI on each worker & master URL



- Connect docker image network

>> docker exec -it docker-spark-cluster_spark-master_1 bash

2. **Reproduce the results of Logistic Regression in Apache Spark**

Put the python file into ./opt/spark-apps

```
root@157669aa794b:/# ls /opt/spark-apps/
logReg.py  logRegSGD.py
```

Put the data into ./opt/spark-data/

```
root@157669aa794b:/# ls /opt/spark-data/
data_banknote_authentication.txt
```

Using different library because the sample code run at python2 and spark version is the old version so that cant run at my docker spark cluster.

Reproduce the results of sample Logistic Regression

```python
from pyspark.mllib.classification import LogisticRegressionWithSGD
from pyspark.mllib.regression import LabeledPoint
from pyspark import SparkConf, SparkContext

def getSparkContext():
    """
    Gets the Spark Context
    """
    conf = (SparkConf()
         .setMaster("local") # run on local
         .setAppName("Logistic Regression") # Name of App
         .set("spark.executor.memory", "1g")) # Set 1 gig of memory
    sc = SparkContext(conf = conf)
    return sc

def mapper(line):
    """
    Mapper that converts an input line to a feature vector
    """
    feats = line.strip().split(",")
    # labels must be at the beginning for LRSGD, it's in the end in our data, so
    # putting it in the right place
    label = feats[len(feats) - 1]
    feats = feats[: len(feats) - 1]
    #feats.insert(0,label)
    #features = [ float(feature) for feature in feats ] # need floats
    return LabeledPoint(label, feats)

# Load and parse the data
sc = getSparkContext()
data = sc.textFile("./opt/spark-data/data_banknote_authentication.txt")
parsedData = data.map(mapper)

# Train model
iterations = int(10000)
model = LogisticRegressionWithSGD.train(parsedData,iterations)

# Predict the first elem will be actual data and the second
# item will be the prediction of the model
labelsAndPreds = parsedData.map(lambda point: (int(point.label),
        model.predict(point.features)))

# Evaluating the model on training data
trainErr = labelsAndPreds.filter(lambda vp: vp[0] != vp[1]).count() / float(parsedData.count())

# Print some stuff
print("Training Error = " + str(trainErr))
print("Final weights: " + str(model.weights))
print("Final intercept: " + str(model.intercept))
```

logReg.py

```
root@157669aa794b:/# python opt/spark-apps/logReg.py
20/11/25 03:57:26 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using bu
iltin-java classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
20/11/25 03:57:31 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeSystemBLAS
20/11/25 03:57:31 WARN BLAS: Failed to load implementation from: com.github.fommil.netlib.NativeRefBLAS
Training Error = 0.0444606413994169l
Final weights: [-1.44881734422,-0.758856452927,-0.7788540453,-0.401591778475]
Final intercept: 0.0
root@157669aa794b:/#
```

Result:

Training Error = **0.044460641**

Final weights: [-1.44881734422,-0.758856452927,-0.7788540453,-0.401591778475]

3. Write your own SGD (stochastic gradient descent or simple gradient descent) function of logistic regression. And compare the results.

```
def getSparkContext():
    """
    Gets the Spark Context
    """
    conf = (SparkConf()
         .setMaster("local") # run on local
         .setAppName("Logistic Regression") # Name of App
         .set("spark.executor.memory", "1g")) # Set 1 gig of memory
    sc = SparkContext(conf = conf)
    return sc

def mapper(line):
    """
    Mapper that converts an input line to a feature vector
    """
    feats = line.strip().split(",")
    # labels must be at the beginning for LRSGD, it's in the end in our data, so
    # putting it in the right place
    label = feats[len(feats) - 1]
    feats = feats[: len(feats) - 1]
    feats.insert(0,label)
    features = [ float(feature) for feature in feats ] # need floats
    return np.array(features)
    #return LabeledPoint(label, feats)

def gradient(row, w):
    Y = row[0]
    X = row[1:]
    return ((1.0 / (1.0 + np.exp(-Y * X.dot(w))) - 1.0) * Y * X.T).sum()

def add(x, y):
    x += y
    return x

# Load and parse the data
sc = getSparkContext()
data = sc.textFile("./opt/spark-data/data_banknote_authentication.txt")
parsedData = data.map(mapper).cache()

w = 2 * np.random.ranf(size=4) -1
for i in range(100):
    w -= parsedData.map(lambda r: gradient(r,w)).reduce(add)

labelsAndPreds = parsedData.map(lambda point: (int(point[0]), 1.0 / (1.0 + np.exp(-1 * w.dot(point[1:])))))
# Evaluating the model on training data
trainErr = labelsAndPreds.filter(lambda vp: vp[0] != vp[1]).count() / float(parsedData.count())

# Print some stuff
print("Training Error = " + str(trainErr))
```

logRegSGD.py

```
Training Error = 0.094752186588921 29
root@157669aa794b:/#
```

Result:

`Training Error = 0.09475218658892129`

Write own Logistic regression of training error is larger than the sample SGD code.

Github:

https://github.com/lee102699/Spark-Cluster.git

Reference:

1. Spark Cluster with Docker & docker-compose
   https://github.com/mvillarrealb/docker-spark-cluster
2. Apache Spark - Logistic_regression.py
   https://github.com/apache/spark/blob/master/examples/src/main/python/mllib/logistic_regression.py
3. Apache Spark - Linear_regression_with_sgd_example.py
   https://github.com/apache/spark/blob/master/examples/src/main/python/mllib/linear_regression_with_sgd_example.py