

Submit a pyspark job through an ActionScript

Steps:

1. Create python script for spark-submit. In the sample below, we have a Random Forest model as a python script by accessing input files from `dtap(dtap://TenantStorage/data/sample_libsvm_data.txt)` and writing output data back to `dtap(dtap://TenantStorage/output_test)` location.

Important notes:

Add these three lines in your python script to create `SparkContext(sc)`, if it's not included already.

```
from pyspark import SparkContext, SparkConf
##Spark Config
conf = SparkConf().setAppName("sample_app")
sc = SparkContext(conf=conf)
```

Script name: `randomforest.py`. Can also be tested from CLI by running `'spark-submit randomforest.py'` from Jupyter node.

```
import sys

from pyspark.mllib.tree import RandomForest, RandomForestModel
from pyspark.mllib.util import MLUtils
from pyspark import SparkContext, SparkConf

## #Spark Config
conf = SparkConf().setAppName("sample_app")
sc = SparkContext(conf=conf)

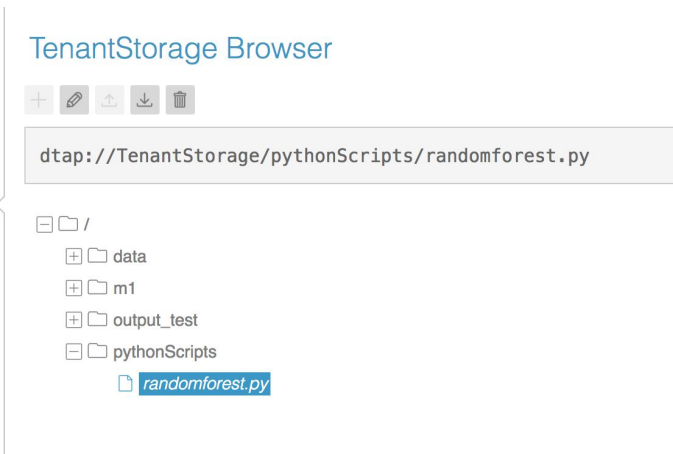
# Load and parse the data file into an RDD of LabeledPoint.
data = MLUtils.loadLibSVMFile(sc, 'dtap://TenantStorage/data/sample_libsvm_data.txt')
# Split the data into training and test sets (30% held out for testing)
(trainingData, testData) = data.randomSplit([0.7, 0.3])

# Train a RandomForest model.
# Empty categoricalFeaturesInfo indicates all features are continuous.
# Note: Use larger numTrees in practice.
# Setting featureSubsetStrategy="auto" lets the algorithm choose.
model = RandomForest.trainRegressor(trainingData, categoricalFeaturesInfo={},
                                   numTrees=3, featureSubsetStrategy="auto",
                                   impurity='variance', maxDepth=4, maxBins=32)

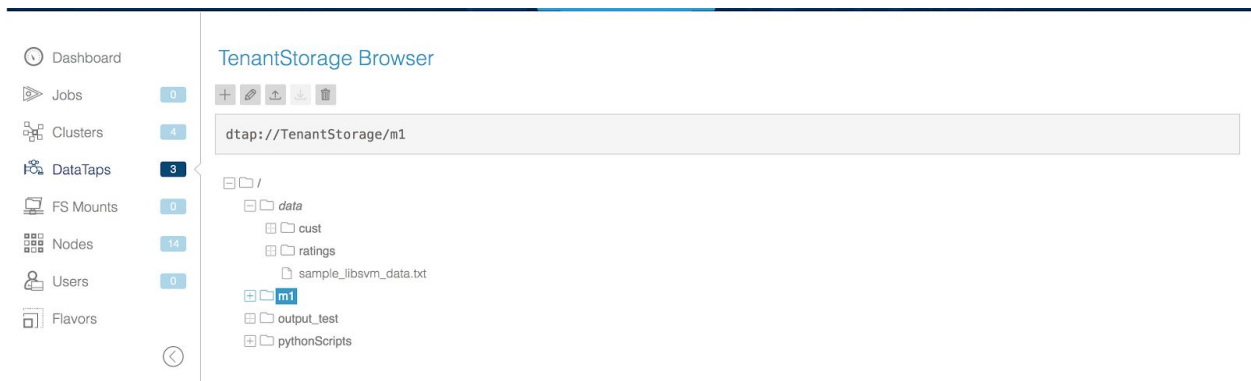
# Evaluate model on test instances and compute test error
predictions = model.predict(testData.map(lambda x: x.features))
labelsAndPredictions = testData.map(lambda lp: lp.label).zip(predictions)
testMSE = labelsAndPredictions.map(lambda vp : (vp[0] - vp[1]) * (vp[0] - vp[1])).sum() /\
float(testData.count())
print('Test Mean Squared Error = ' + str(testMSE))
print('Learned regression forest model:')
print(model.toDebugString())

# Save and load model
model.save(sc, "dtap://TenantStorage/output_test")
sameModel = RandomForestModel.load(sc, "dtap://TenantStorage/output_test")
print(sameModel)
```

2. If your script is in on your desktop, upload it to dtap location. In this example below, the script is uploaded to dtap://TenantStorage/pythonScripts/randomforest.py



3. Make Sure all the input and output directories defined in python exist in dtap storage. As you can see below, the input file (sample_libsvm_data.txt) used in python script output directory are in dtap.



4. Now let's create an ActionScript "[pyspark_dtap.sh](#)" to perform spark-submit on the python script we created.

The sample ActionScript below performs the following

- Copies python script to the node where you are performing spark-submit, under "/tmp" location.

- In this case, script is running on jupyter node, and hence the \$node == "jupyter" in the action script. By default, action scripts run on all nodes unless specified. you can replace it with appropriate node where you want to perform spark-submit.
- Makes the script executable.
- Run spark-submit on the python script.

```

pyspark_dtap.sh
1  #/bin/bash
2
3  export node=`bdvcli --get node.role_id`
4
5  if [[ $node == "jupyter" ]]; then
6
7      hadoop fs -get dtap://TenantStorage/pythonScripts/randomforest.py /tmp
8      chmod a+x /tmp/randomforest.py
9      /usr/lib/spark/spark-2.1.1-bin-hadoop2.7/bin/spark-submit /tmp/randomforest.py
10 fi
11

```

Role definitions are as shown below;

NAME	DISTRIBUTION	ROLE	INSTANCE IP	SERVICES
bluedata-98.bdiocal	Spark 2.1.1 multi with centos7x	rstudio	10.39.250.7	Rstudio server, Shiny server SSH: 10.39.250.7 -p 22
bluedata-96.bdiocal	Spark 2.1.1 multi with centos7x	jupyter	10.39.250.15	Jupyter Notebook SSH: 10.39.250.15 -p 22
bluedata-88.bdiocal	Spark 2.1.1 multi with centos7x	worker	10.39.250.5	Spark worker SSH: 10.39.250.5 -p 22
bluedata-87.bdiocal	Spark 2.1.1 multi with centos7x	controller	10.39.250.6	Spark master, Spark worker Spark master: 10.39.250.6:7077 SSH: 10.39.250.6 -p 22

4. To run an action script, create an action as shown below from BlueData cluster detail screen

Run ActionScript(s)

This feature assumes that the user has appropriate privileges to run scripts on the cluster.

Action Name

Script source

Script Clear Change

Arguments

Run as a root ☒

Cancel Submit

This submits a spark job to the cluster, you will see the script running with its status

Spark2.2.1 ready

Node(s) Info | **ActionScript(s)** | Job(s) Info | Services Status | Cluster History

Edit Stop Reboot Delete

ActionScript List

Run Delete

ACTION NAME >	USER >	START TIME ^	END TIME >	RUN TIME >	ACTION STATUS >	ACTIONS >
<input type="checkbox"/> pyspark-dtap	admin (As Root)	Wed Mar 21 2018 15:46:23	Wed Mar 21 2018 15:46:53	00:30	completed	

Rows 10 Showing 1 to 1 of 1 entries Previous 1 Next

If you click on the logs, under individual actions, You will see the log files as shown below. Here, this script only ran on Jupyter node.

Per-Node Action Results

bluedata-98.bdlocal's logs completed

bluedata-96.bdlocal's logs completed

18/03/21 22:46:34 INFO MemoryStore: Block broadcast_0_piece0 stored as bytes in memory (estimated size 22.9 KB, free 366.0 MB)
18/03/21 22:46:34 INFO BlockManagerInfo: Added broadcast_0_piece0 in memory on 10.39.250.15:44557 (size: 22.9 KB, free: 366.3 MB)
18/03/21 22:46:34 INFO SparkContext: Created broadcast 0 from textFile at NativeMethodAccessorImpl.java:0
18/03/21 22:46:34 INFO FileInputFormat: Total input paths to process : 1
18/03/21 22:46:34 INFO SparkContext: Starting job: loadLibSVMFile at /tmp/randomforest.py:13
18/03/21 22:46:34 INFO DAGScheduler: Get job 0 (loadLibSVMFile at /tmp/randomforest.py:13) with 2 output

bluedata-87.bdlocal's logs completed

bluedata-88.bdlocal's logs completed

Close