How to pass params to a ML Pipeline.fit method?

I am trying to build a clustering mechanism using

- Google Dataproc + Spark
- Google Bigquery
- Create a job using Spark ML KMeans+pipeline

As follows:

1. Create user level based feature table in bigquery
   Example: How the feature table looks like

```
userid |x1  |x2  |x3 |x4 |x5 |x6 |x7 |x8   |x9   |x10
```

```
00013  |0.01 | 0  |0  |0  |0  |0  |0  |0.06 |0.09 | 0.001
```

2. Spin up a default setting cluster, am using gcloud command line interface to create the cluster and run jobs as shown here
3. Using the starter code provided, I read the BQ table, convert RDD into a Dataframe and pass to KMeans model/pipeline:

```
#!/usr/bin/python
"""BigQuery I/O PySpark example."""
import json
import pprint
import subprocess
import pyspark
import numpy as np
from pyspark.ml.clustering import KMeans
from pyspark import SparkContext
from pyspark.ml import Pipeline
from pyspark.sql import SQLContext
from pyspark.mllib.linalg import Vectors, _convert_to_vector
from pyspark.sql.types import Row
from pyspark.mllib.common import callMLlibFunc, callJavaFunc, _py2java, _java2py
sc = pyspark.SparkContext()

# Use the Google Cloud Storage bucket for temporary BigQuery export data used by the
InputFormat.
# This assumes the Google Cloud Storage connector for Hadoop is configured.

bucket = sc._jsc.hadoopConfiguration().get('fs.gs.system.bucket')
project = sc._jsc.hadoopConfiguration().get('fs.gs.project.id')
input_directory ='gs://{}/hadoop/tmp/bigquery/pyspark_input'.format(bucket)
 conf = {# Input Parameters
 'mapred.bq.project.id': project,
 'mapred.bq.gcs.bucket': bucket,
 'mapred.bq.temp.gcs.path': input_directory,
 'mapred.bq.input.project.id': 'my-project',
 'mapred.bq.input.dataset.id': 'tempData',
 'mapred.bq.input.table.id': 'userFeatureInBQ'}

# Load data in from BigQuery.
table_data = sc.newAPIHadoopRDD(
 'com.google.cloud.hadoop.io.bigquery.JsonTextBigQueryInputFormat',
 'org.apache.hadoop.io.LongWritable',
 'com.google.gson.JsonObject',conf=conf)

# Tranform the userid-Feature table into feature_data RDD
 feature_data = (
 table_data
  .map(lambda (_, record): json.loads(record))
  .map(lambda   x:(x['x0'],x['x1'],x['x2'],x['x3'],x['x4'],
                 x['x5'],x['x6'],x['x7'],x['x8'],
```

```
                    x['x9'],x['x10'])))

# Function to convert each line in RDD into an array, return the vector
  def parseVector(values):
     array = np.array([float(v) for v in values])
     return _convert_to_vector(array)

# Convert the RDD into a row wise RDD
  data = feature_data.map(parseVector)
  row_rdd = data.map(lambda x: Row(x))

sqlContext = SQLContext(sc)

# cache the RDD to improve performance
row_rdd.cache()

# Create a Dataframe
df = sqlContext.createDataFrame(row_rdd, ["features"])

# cache the Dataframe
df.cache()
```

Here is the Schema and head() which I print to the console:

```
|-- features: vector (nullable = true)
[Row(features=DenseVector([0.01,0,0,0,0,0,0,0.06,0.09,0.001]))]
```

4. Run the clustering KMeans algorithm in following manner
    - Run the model multiple times
    - With different parameters (Namely, change the #clusters and init_mode)
    - Calculate error or Cost metric
    - Choose best model-parameter combination
    - Create pipeline with KMeans as an estimator
    - Pass multiple parameters using paramMap

```
#Define the paramMap & model
paramMap = ({'k':3,'initMode':'kmeans||'},{'k':3,'initMode':'random'},
  {'k':4,'initMode':'kmeans||'},{'k':4,'initMode':'random'},
  {'k':5,'initMode':'kmeans||'},{'k':5,'initMode':'random'},
  {'k':6,'initMode':'kmeans||'},{'k':6,'initMode':'random'},
  {'k':7,'initMode':'kmeans||'},{'k':7,'initMode':'random'},
```

```
    {'k':8,'initMode':'kmeans||'},{'k':8,'initMode':'random'},
    {'k':9,'initMode':'kmeans||'},{'k':9,'initMode':'random'},
    {'k':10,'initMode':'kmeans||'},{'k':10,'initMode':'random'})

  km = KMeans()

  # Create a Pipeline with estimator stage
  pipeline = Pipeline(stages=[km])

  # Call & fit the pipeline with the paramMap
  models = pipeline.fit(df, paramMap)`
  print models
```

I get the following output with a warning

```
 7:03:24 WARN org.apache.spark.mllib.clustering.KMeans: The input data was not directly cached, which may hurt performance if its parent RDDs
are also uncached.
[PipelineModel_443dbf939b7bd3bf7bfc, PipelineModel_4b64bb761f4efe51da50, PipelineModel_4f858411ac19beacc1a4,
PipelineModel_4f58b894f1d14d79b936, PipelineModel_4b8194f7a5e6be6eaf33, PipelineModel_4fc5b6370bff1b4d7dba,
PipelineModel_43e0a196f16cfd3dae57, PipelineModel_47318a54000b6826b20e, PipelineModel_411bbe1c32db6bf0a92b,
PipelineModel_421ea1364d8c4c9968c8, PipelineModel_4acf9cdbfda184b00328, PipelineModel_42d1a0c61c5e45cdb3cd,
PipelineModel_4f0db3c394bcc2bb9352, PipelineModel_441697f2748328de251c, PipelineModel_4a64ae517d270a1e0d5a,
PipelineModel_4372bc8db92b184c05b0]
```

```
 #Print the cluster centers:
 for model in models:
     print vars(model)
     print model.stages[0].clusterCenters()
     print model.extractParamMap()
```

Output: `[array([7.64676638e-07, 3.58531391e-01, 1.68879698e-03, 0.00000000e+00, 1.53477043e-02, 1.25822915e-02, 0.00000000e+00,
6.93060772e-07, 1.41766847e-03, 1.60941306e-02], array([2.36494105e-06, 1.87719732e-02, 3.73829379e-03, 0.00000000e+00, 4.20724542e-02,
2.28675684e-02, 0.00000000e+00, 5.45002249e-06, 1.17331153e-02, 1.24364600e-02])`

Here it the list of questions and need help with:

- I get a list with only 2 cluster centers as arrays for all models,
  - It seems the KMeans models is defaulting to k=2 when I try to access the pipeline? Why would this happen?
  - The last loop is supposed to access the pipelineModel and the 0th stage and run the clusterCenter() method? Is this the right method?

    - Why do I get the error that data is uncached?
- I could not find how to compute the WSSSE or any equivalent method like .computeCost()(for mllib) when using a pipeline? How can I compare the different models based on different parameters?
- I tried the following code to run the .computeCost method as defined in the source code here:
    - This defeats the purpose of running KMeans model and model selection in parallel using pipeline, however I have tried the following code:

```
#computeError
def computeCost(model, rdd):`
"""Return the K-means cost (sum of squared distances of
 points to their nearest center) for this model on the given data."""
    cost = callMLlibFunc("computeCostKmeansModel",
                         rdd.map(_convert_to_vector),
                [_convert_to_vector(c) for c in model.clusterCenters()])
    return cost

cost= np.zeros(len(paramMap))

for i in range(len(paramMap)):
    cost[i] = cost[i] + computeCost(model[i].stages[0], feature_data)
print cost
```

This prints out the following at the end of the loop:

```
[ 634035.00294687  634035.00294687  634035.00294687  634035.00294687
  634035.00294687  634035.00294687  634035.00294687  634035.00294687
  634035.00294687  634035.00294687  634035.00294687  634035.00294687
  634035.00294687  634035.00294687  634035.00294687  634035.00294687]
```

- The cost/error calculated is the same for each model? Again cannot access the pipelineModel with the correct parameters.

Any help/ guidance is much appreciated! Thanks!

python apache-spark pyspark apache-spark-mllib apache-spark-ml

edited Feb 7 at 15:48

zero323

65.9k 16 77 136

asked Feb 7 at 13:20

Tushar Kanade

22 5

1 Suggestion, try to show only the gist of your doubts, not all the code. – Alberto Bonsanto Feb 7 at 15:54

Sorry for being so long, did not know how to put the entire idea across as it has parts connected across multiple things. Will keep it short the next time, thanks! – Tushar Kanade Feb 8 at 10:03

1 Answer

Your param is not properly defined. It should map from the specific parameters to the values, not from arbitrary names. You get `k` equal 2 because parameters you pass are not utilized and every model uses exactly the same default parameters.

Lets start with example data:

```
import numpy as np
from pyspark.mllib.linalg import Vector

df = (sc.textFile("data/mllib/kmeans_data.txt")
  .map(lambda s: Vectors.dense(np.fromstring(s, dtype=np.float64, sep=" ")))
  .zipWithIndex()
  .toDF(["features", "id"]))
```

and a `Pipeline`:

```
from pyspark.ml.clustering import KMeans
from pyspark.ml import Pipeline

km = KMeans()

pipeline = Pipeline(stages=[km])
```

As mentioned above parameter map should use specific parameters as the keys. For example:

```
params = [
    {km.k: 2, km.initMode: "k-means||"},
    {km.k: 3, km.initMode: "k-means||"},
    {km.k: 4, km.initMode: "k-means||"}
]

models = pipeline.fit(df, params=params)

assert [len(m.stages[0].clusterCenters()) for m in models] == [2, 3, 4]
```

Notes:

- correct `initMode` for K-means|| is `k-means||` not `kmeans||` .
- using parameter map in a Pipeline doesn't mean that model are trained in parallel. Spark parallelizes training process over data not over params. It is nothing more than a convenience method.
- you get the warning about not cached data because actual input to K-Means is not a `DataFrame` but transformed RDD.

edited Feb 7 at 15:53

answered Feb 7 at 15:46

zero323
65.9k 16 77 136

Awesome, thanks for the edit and sorry for being really long on the question. I can now get the params passed and retrieve the cluster centers. I still need help on, how to get the error metric/computeCost method of pipeline method? – Tushar Kanade Feb 8 at 10:00

Thanks, I can use the computeCost(df) method on the each of the models and find out the cost. This completes the process for now. Thank you very much! – Tushar Kanade Feb 8 at 11:59

I double checked, the computeCost method is not yet available on the KMeansModel in ML, on github it says since V 2.0.0. I still have to call the function ComputeCost, pass it the rdd and cluster centers and get the cost, any alternative over here? – Tushar Kanade Feb 9 at 5:24

Thanks @zero323, i am also getting same warning even after i cached my Dataframe. How can i resolve this? – Kaushal Mar 28 at 14:26

@Kaushal AFAIK you don't. Caching input DF should be enough even if there is some overhead. – zero323 Mar 28 at 14:37