DADS 6002 / CI 7301 Machine Learning with Spark Lab

Python Setup

- Install Python 2.7

```
# sudo su -
# wget https://repo.continuum.io/archive/Anaconda2-4.2.0-Linux-x86_64.sh
# bash Anaconda2-4.2.0-Linux-x86_64.sh
```

(After reading the license agreement, answer yes then confirm the location that Anaconda2 will be installed, i.e. /root/anaconda2.)

Python Setup

```
#  cd /root
```

- Add an environment variable for running Python 2.7 by appending the following line to the file .bashrc under /root (using nano editor)

```
export  PYSPARK_PYTHON=/root/anaconda2/bin/python
```

- Login as the root user again so the new setup will take effect.

```
# sudo su -
```

ALS Example

```
from pyspark import SparkContext
from pyspark.mllib.recommendation import ALS
from numpy import array

sc = SparkContext(appName=‘als’)
# Load and parse the data
data = sc.textFile(“/user/cloudera/test.data")
ratings = data.map(lambda line: line.split(',')).map(lambda r: (int(r[0]),
int(r[1]), float(r[2])))

# Build the recommendation model using Alternating Least Squares
rank = 10
numIterations = 20
model = ALS.train(ratings, rank, numIterations)
```

ALS Example

```
# Evaluate the model on training data
testdata = ratings.map(lambda p: (p[0], p[1]))
pd = model.predictAll(testdata).map(lambda r: ((r[0], r[1]), r[2]))
predictions = pd.sortByKey().map(lambda r: r[1] )
rd = ratings.map(lambda r: ((r[0], r[1]), r[2]))
actualRates = rd.sortByKey().map(lambda r: r[1] )

ratesAndPreds = actualRates.zip(predictions)

MSE = ratesAndPreds.map(lambda r: (r[0] - r[1])**2).reduce(lambda x,
y: x + y)/ratesAndPreds.count()
print("Mean Squared Error = " + str(MSE))
```

ALS

- Download als.py from MS Teams into the shared folder and copy it to the working directory.
- Download test.data from MS Teams into the shared folder and copy it to the working directory.
- Copy test.data to hdfs

```
# hadoop fs –put test.data /user/cloudera
```

- Submit a spark job

```
# spark-submit als.py > out
# more out
```

Logistic Regression Example

```
from pyspark import SparkContext
from pyspark.mllib.classification import LogisticRegressionWithSGD
from pyspark.mllib.regression import LabeledPoint
from numpy import array

# Load and parse the data
def parsePoint(line):
    values = [float(x) for x in line.split(' ')]
    return LabeledPoint(values[0], values[1:])

sc = SparkContext(appName=‘logistics’)
data = sc.textFile(“/user/cloudera/sample_svm_data.txt")
parsedData = data.map(parsePoint)
```

Logistics Regression Example

```
# Build the model
model = LogisticRegressionWithSGD.train(parsedData)

# Evaluating the model on training data
labelsAndPreds = parsedData.map(lambda p: (p.label,
model.predict(p.features)))
trainErr = labelsAndPreds.filter(lambda (v, p): v != p).count() /
float(parsedData.count())
print("Training Error = " + str(trainErr))
```

Logistics Regression

- Download logistics.py from MS Teams into the shared folder and copy it to the working directory.
- Download sample_svm_data.txt from MS Teams into the shared folder and copy it to the working directory.
- Copy sample_svm_data.txt to hdfs

```
# hadoop fs –put sample_svm_data.txt /user/cloudera
```

- Submit a spark job

```
# spark-submit logistics.py > out
# more out
```

K-Means Example

```
from pyspark import SparkContext
from pyspark.mllib.clustering import KMeans
from numpy import array
from math import sqrt

sc = SparkContext(appName=‘kmeans’)
# Load and parse the data
data = sc.textFile(“/user/cloudera/kmeans_data.txt")
parsedData = data.map(lambda line: array([float(x) for x in line.split('
')]))

# Build the model (cluster the data)
clusters = KMeans.train(parsedData, 2, maxIterations=10,
        runs=10, initializationMode="random")
```

K-Means Example

```
# Evaluate clustering by computing Within Set Sum of Squared Errors
def error(point):
    center = clusters.centers[clusters.predict(point)]
    return sqrt(sum([x**2 for x in (point - center)]))

WSSSE = parsedData.map(lambda point: error(point)).reduce(lambda
x, y: x + y)
print("Within Set Sum of Squared Error = " + str(WSSSE))
```

K-Means

- Download kmeans.py from MS Teams into the shared folder and copy it to the working directory.
- Download kmeans_data.txt from MS Teams into the shared folder and copy it to the working directory.
- Copy kmeans_data.txt to hdfs

```
# hadoop fs –put kmeans_data.txt  /user/cloudera
```

- Submit a spark job

```
# spark-submit kmeans.py > out
# more out
```