Spark MLLIB Clustering

- MLLIB package: Kmeans

Assign points to ‘closest’ cluster mean,
Update cluster mean,
Iterate until assignments converge

Needs to iterate over data,
and calculate distance to cluster centers

- MLLIB package:Kmeans

```
from pyspark.mllib.cluster import Kmeans,
Kmeansmodel
```

- MLLIB package:Kmeans

```
my_kmmodel = KMeans.train(my_data,
        k=3,
        maxIterations=10,
        runs=2,
        initializationMode='k-means||')
```

number of clusters

use k-means over small, sample to initialize (other option is ‘random’)

- Generating Random data:

```
from pyspark.mllib.random import RandomRDDs
```

normal distribution

```
c1  v=RandomRDDs.normalVectorRDD(
      sc,20,2,
      numPartitions=2,
      seed=1L).
      map(lambda v:np.add([1,5],v))
```

20 rows, 2 columns

center points around [1,5]

- Generating Random data:

Ask for stats of the RandomRDD

```
print c1_v.stats()
```

```
Out[]: (count: 20, mean: [ 1.15426378  4.90223615],
stdev: [ 1.10801385  1.40049822],
max: [ 3.01083638  8.46783831],
min: [-1.08338413  2.83934928])
```

You get basic stats by column

- Generate 2 more classes and concatenate:

```
c2_v=RandomRDDs  ….   np.add([5,1],v))

c3_v=RandomRDDs  …    np.add([4,6],v))

c12        =c1_v.union(c2_v)
my_data=c12.union(c3_v)
```

- Kmeans model functions:

```
#Sum Square Error of points to their cluster's center
my_kmmodel.computeCost(my_data)

# get cluster centers
my_kmmodel.clusterCenters

Out: [array([ 5.0476959 ,  1.27729209]), array([ 3.99839705,
6.28073879]), array([ 0.95767935,  4.69770646])]
```

Note: with big data you sometimes only keep the cluster centers for further analysis

Spark MLLIB Frequent Item (Associated) Sets

- MLLIB package: FPGrowth

Search transactions for ‘frequent pattern’ (FP) items sets

Grows a tree based on frequency order of increasing set size

Note: currently experimental in MLLIB for Python (your cloudera vm version might not have it)

- MLLIB package:FPGrowth

The weather dataset, but coded as all categoricals

```
rawdata=[
    ['sunny',   'H','B','FALSE', 'No'],
    ['sunny',   'H','B','TRUE',  'No'],
    ...
    ...
```

Make it an RDD:

```
wrdd     = sc.parallelize(rawdata)
```

- MLLIB package: FPGrowth

```
from pyspark.mllib.fpm import FPGrowth
```

```
fp_model    = FPGrowth.train(wrdd, 0.5)
```

minimum support of item-set

- FPGrowth results:

```
freqset  =sorted(fp_model.freqItemsets().collect())

for fs in freqset:
   print fs

Out[]:FreqItemset(items=[u'B'], freq=8)
FreqItemset(items=[u'FALSE'], freq=8)
…
FreqItemset(items=[u'Yes', u'L'], freq=7)
```

Single items get more support, but ‘Yes’, ‘L’ (yes play and low temp) are found 7 times