wk10_mini-project

November 1, 2015

1 Clustering Mini-project Wk 10

Aaron Palumbo

1.1 Dependencies

```
In [3]: from pyspark.mllib.clustering import KMeans, KMeansModel
        from numpy import array
        from math import sqrt

        import matplotlib.pyplot as plt

        import os

        %matplotlib inline
```

Next we need to load the data directly into our spark context (sc). We do this by reading the text file, however, we then need to parse the text file into the array we need. We do this by splitting each line with our map function.

```
In [1]: # Load and parse the data
        data = sc.textFile("file://" + os.getcwd() + "/birch1.txt")
        # parsedData = data.map(lambda line: array([float(x) for x in line.split(' ') if len(x) > 0]))
        parsedData = data.map(lambda line: array([float(x) for x in line.split()]))
```

Now we are ready to call the clustering algorithm from MLLib. We have a fairly limited set of tuning parameters. We will choose the same number of iterations we chose when running the Hadoop map reduce job for the wk 8 project, however this time we will allow the algorithm to do this 10 times (runs). This is because the kmeans algorithm is subject to local minimum. 10 runs with 10 random initializations will hopefully allow us to find a good solution.

```
In [4]: # Build the model (cluster the data)
        clusters = KMeans.train(parsedData, 100, maxIterations=10,
                                runs=10, initializationMode="random")
```

```
In [5]: clusters.centers[:20]
```

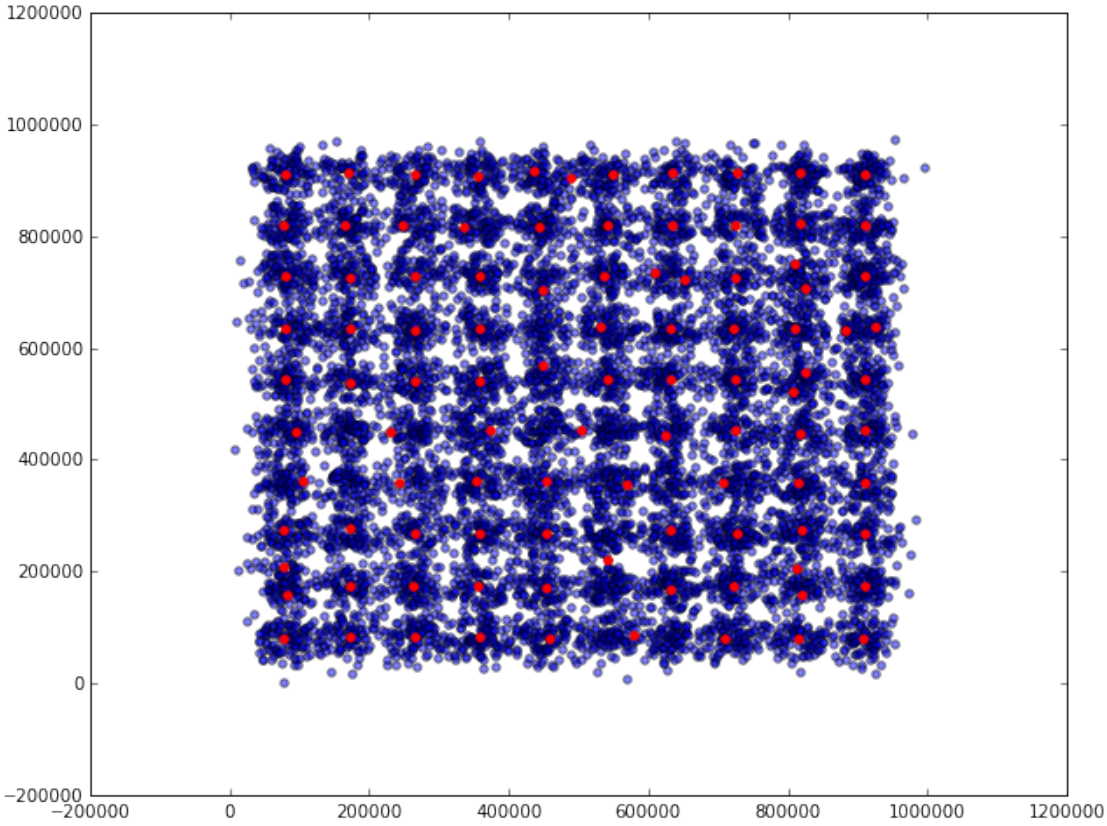
```
Out[5]: [array([ 449039.98017446,  703764.03092784]),
         array([ 815393.23416507,  358809.74088292]),
         array([ 708167.11726384,  357679.76465798]),
         array([ 653477.23529412,  720560.32263815]),
         array([ 173513.80055147,  537174.48253676]),
         array([ 357916.37400794,  173263.73214286]),
         array([ 570431.5620339,  353615.9159322]),
```

```
       array([  80120.60462777,  541755.20120724]),
       array([ 725966.4631068 ,  450228.84757282]),
       array([ 360838.31314073,  635110.67940354]),
       array([ 632537.47696477,  166909.25203252]),
       array([ 449859.5710207 ,  567052.72162741]),
       array([ 360012.42326981,  727755.03109328]),
       array([ 542601.04789272,  541959.79885057]),
       array([  81296.09127382,  910316.28686058]),
       array([ 375652.8623242 ,  451843.54182087]),
       array([ 265775.16682927,  631266.91707317]),
       array([ 166173.49460043,  819145.83153348]),
       array([ 635659.80816327,  911225.44897959]),
       array([ 820247.74776786,  272116.33816964])]
```

In [6]:
```
sampledData = parsedData.sample(False, 0.1, 81).collect()
[x, y] = array(sampledData).transpose()
[cx, cy] = array(clusters.centers).transpose()

fig, ax = plt.subplots(figsize=(10, 8))
ax.scatter(x, y, alpha=0.5)
ax.scatter(cx, cy, color="red")
```

Out[6]: <matplotlib.collections.PathCollection at 0x7fb10cafeb50>



In [7]:
```
clusters.k
```

```
Out[7]: 100
```

Obviously using the clustering algorithm from MLlib with pyspark was much easier than using Hadoop Kmeans clustering from the week 8 assignment. It also produced a much better answer in a lot less time. The time difference is impressive. The algorithm ran 10 times as many computations and was still noticeably faster.