ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΠΡΟΧΩΡΗΜΕΝΑ ΘΕΜΑΤΑ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

Ακ. έτος 2019-2020

Εξαμηνιαία Εργασία

ΚΟΝΤΟΓΙΑΝΝΗΣ ΑΝΔΡΕΑΣ
03115187

Θέμα 3ο: Machine Learning - Ομαδοποίηση δεδομένων με εκτέλεση του KMeans αλγόριθμου

Χρησιμοποιώντας τα δεδομένα της εκφώνησης, σκοπό της μελέτης αποτελούσε η εύρεση των 5 κορυφαίων σημείων επιβίβασης με την χρήση του KMeans αλγορίθμου. Επείδη τα δεδομένα ήταν αρκετά μεγάλα, κατασκευάσαμε τον KMeans με Map-Reduce method, αφού προηγουμένως είχαμε κάνει όλα τα απαραίτητα, που μας παρείχαν οι οδηγίες του μαθημάτος, για την εγκατάσταση του hdfs και του spark. Χρησιμοποιήσαμε το Module της python2, pyspark, μία τελευταίας τεχνολογίας δωρεάν βιβλιοθήκη, που παρέχει όλες τις απαραίτητες λειτουργίες του big data, συμπεριλαμβανομένων και map-reduce υπηρεσιών.

Ας ξεκινήσουμε από τα βασικά. Τι είναι ο KMeans; Ο KMeans είναι ένας κλασικός αλγόριθμος μηχανικής μάθησης, και μάλιστα unsupervised learning, που επιτελεί την συσταδοποιήση των δεδομένων σε Κ συστάδες (clusters), με το Κ να αποτελεί υπερπαράμετρο που δίνεται εξωτερικά. Αποτελείται από δυό στάδια: το E step και το M step. Στο E step (Expectation) βρίσκει κάθε σημείο του δείγματος σε ποια συστάδα ανήκει σύμφωνα με μία μετρική απόστασης (εδώ Haversine) από το prototype της καθεμιάς. Στο M step (Maximization) προσδιορίζει ξανά τα prototypes των συστάδων, ως τον μέσο όρο των σημείων που τις αποτελούν. Ο αλγόριθμός τερματίζει ύστερα από σύγκλιση είτε μετά από κάποιες επαναλήψεις οι οποίες έχουν ληφθεί εξωτερικά ως υπερπαράμετρος. Σημειώνουμε πως στην αρχή έχουμε αρχικοποιήσει τα prototypes σε κάποιες τιμές πχ. στις πρώτες Κ εγγραφές του συνόλου δεδομένων μας.

Αν είχαμε να κάνουμε με ένα αρκετά μικρότερο dataset, τότε το task μας θα ήταν απλά να προγραμματίσουμε τον παραπάνω κλασικό αλγόριθμο και να κάνουμε fit τα δεδομένα μας.

Σημείωση: Ο κώδικας για τον κλασικό KMeans χωρίς map-reduce παρέχεται extra μέσα στο .zip στον ξεχωριστό φάκελο classic_kmeans.

Συνεχίζουμε με το map-reduce πρόβλημα. Αρχικά, βλέπουμε πως είναι το jps στον master και στο slave, για να βεβαιωθούμε ότι έχουμε στήσει σωστά το hdfs και το spark:

```
user@master:~$ jps
816 Jps
32738 NameNode
500 SecondaryNameNode
655 Master
```

```
user@slave:~$ jps
4457 Jps
25497 Worker
25358 DataNode
```

Στο σημείο αυτό, μπορούμε να σχεδιάσουμε την map-reduce υλοποίηση. Παρακάτω, παρατίθεται ο ψευδοκώδικας για τον map-reduce KMeans. Θα ξεκινήσουμε με το σπλιτάρισμα των γραμμών, την προσάρτηση των επιθυμητών συντεταγμένων και το φιλτράρισμά τους για την αποφυγή invalid συντεταγμένων.

```
map(key, line):
    spl = split(line, ",")  # cut ","
    emit ( float(spl[3]), float(spl[4]) )  # return (x,y)

map(x,y):
    if x <> 0 and y <> 0 :
        emit (x,y)     # return valid (x,y)
```

Στην συνέχεια, παραθέτουμε τον ψευδοκώδικα για τον KMeans, ξεκινώντας από το E step και καταλήγοντας στο M step του, που έχουμε περιγράψει παραπάνω.

```
KMeans_e_step(metric, (x,y), centroids):
    # Returns the index of closest centroid to point (x,y) using metric distance
    r := 0
    i := 0
    min := +Inf
    for all centroid in centroids:
        distance := metric( (x,y), centroid)
        if distance < min:
            min := distance
            r := i
        i := i + 1
```

```
# KMeans #
centroids := first k records of dataset
MAX_ITER := 3
it := 0
for all it: iter < MAX_ITER
{
    # E Step #
    map(x,y):   # find closest cluster (r) and returns (r, ( (x,y), 1) in order to count cluster size
        emit ( KMeans_e_step(haversine, (x, y), centroids),  ((x,y), 1) )

    # M Step #
    reduce(r, list_of_xy_and_1):
        emit ( r, ( sum( list_of_xy(x)(y)), sum(list_of_1) )

    map(r, (sum_xy, sum_r)):   # update centroids
        emit ( r, ( sum_xy(x) / sum_r, sum_xy(y) / um_r) )

    store centroids

    it := it + 1
}
```

Σημειώνουμε για το M step ότι στο reduce βρίσκουμε για κάθε centroid το άθροισμα πάνω σε όλα τα x, το άθροισμα πάνω σε όλα τα y και το άθροισμα πάνω στους άσσους, ενώ στο map βρίσκουμε το μέσο όρο των δυο πρώτων αθροισμάτων με την διαίρεση τους με το τρίτο.

Τέλος, τα αποτελέσματα όταν τρέξαμε τον αλγόριθμο ήταν τα παρακάτω:

```
KMeans results
(0, (-74.0191832401856, 40.71055443942774))
(1, (-73.94113734517788, 40.76328399383515))
(2, (-73.99772401348547, 40.731550302890184))
(3, (-74.00521692085603, 40.73256456305599))
(4, (-73.98429681172031, 40.75122648858301))
```

Αξίζει να παραθέσουμε και τα logs, τόσο του master όσο και του slave:

master logs

```
user@master:~/spark-2.4.4-bin-hadoop2.7/logs$ ls
spark-user-org.apache.spark.deploy.master.Master-1-master.out    spark-user-org.apache.spark.deploy.master.Master-1-master.out.4
spark-user-org.apache.spark.deploy.master.Master-1-master.out.1  spark-user-org.apache.spark.deploy.master.Master-1-master.out.5
spark-user-org.apache.spark.deploy.master.Master-1-master.out.2  spark-user-org.apache.spark.deploy.worker.Worker-1-master.out
spark-user-org.apache.spark.deploy.master.Master-1-master.out.3
```

slave logs

```
user@slave:~/spark-2.4.4-bin-hadoop2.7/logs$ ls
spark-user-org.apache.spark.deploy.master.Master-1-slave.out    spark-user-org.apache.spark.deploy.worker.Worker-1-slave.out.3
spark-user-org.apache.spark.deploy.worker.Worker-1-slave.out    spark-user-org.apache.spark.deploy.worker.Worker-1-slave.out.4
spark-user-org.apache.spark.deploy.worker.Worker-1-slave.out.1  spark-user-org.apache.spark.deploy.worker.Worker-1-slave.out.5
spark-user-org.apache.spark.deploy.worker.Worker-1-slave.out.2
```

dataset link: http://83.212.76.198:50070/explorer.html#/