Solution for CIS 670 Data Science Assignment 5

1. *Suppose that the data mining task is to cluster the following nine points (with (x,y) representing location) into three clusters:* $A_1(3,9)$, $A_2(2,5)$, $A_3(9,4)$, $B_1(4,8)$, $B_2(8,5)$, $B_3(7,4)$, $C_1(2,2)$, $C_2(5,10)$, $C_3(6,8)$. *Suppose initially we assign* A_1, B_1 *and* C_1 *as the center of each cluster, respectively. Please add a Map-reduce function for the K-means algorithm. Show the results for the first two iterations and explain how Map-reduce can help.*

Answer:

Map Reduce:

Let K_1, K_2, K_3 be the three centroids for the current iteration, let $d(P, K)$ be the distance between points P and K.

Map: We map each points with coordinates i, j, $P(i, j)$ to l if P is closest to centroid K_l. A sample map function could be

```
map(P) {
        emit(index_of_cloest_cetroid(K1, K2, K3, P), P)
}
```

Reduce:

```
reduce(tuples){
        return [tuples.centroid_index, sum(tuples.x)/tuples.count,
sum(tuples.y)/tuples.count];
}
```

Execution:

Iteration 1:

Cluster 1: A_1, C_2
Centroid 1: (4, 9.5)

Cluster 2: B_1, C_3, B_3, B_2, A_3
Centroid 2: (6.8, 5.8)

Cluster 3: C_1, A_2
Centroid 3: (2, 3.5)

Iteration 2:

Cluster 1: A_1, B_1, C_2
Centroid 1: (4, 9)

Cluster 2: C_3, B_3, B_2, A_3
Centroid 2: (7.5, 5.25)

Cluster 3: C_1, A_2
Centroid 1: (2, 3.5)

Benefit:

The map reduce can help in the sense that all these operations can run in parallel.

2. A database has six transactions. Let min sup = 50%.

TID	items_sold
T001	A, B, C, D, E, F
T002	B, H, E, C, F, T
T003	C, U, O, E, W, D
T004	W, A, B, C, F, X
T005	W, X, C, D, F, Y
T006	B, C, D, E, O, Z

Please add a Map-reduce function for the Apriori algorithm to generate all frequent itemsets. Show the results for each step and explain how Map-reduce can help.

Answer:

Map: For each iteration, generate frequent items.

```
function(doc) {
    var iteration;
    var frequent_itemset;
    if(0 == iteration)
```

```
                for_each(trans = doc.transactions)
                        for_each(item = trans.items)
                                frequent_itemset.add(item);
        else{
                // the function below take each two frequent_itemset, generate the
    union of them and add in the set.
                frequent_itemset = set( Cartesian_union(frequent_itemset) );
        }
        for_each(trans in doc.transactions){
                for_each(item in frequent_itemset){
                        if(trans.contains(item) ){
                                emit(item, 1);
                        }
                }
        }
}
```

Reduce: Count all the emitted items, compare with the support threshold.

```
reduce(keys, values, reducer){
        var count = _count;
        if(count / doc.transactions.length > doc.support)
                return (c, value)
}
```

L_1 (frequency >= 3):

B, 4
C, 6
D, 4
E, 4
F, 4
W, 3

Candidate C_2

BC, BD, BE, BF, BW, CD, CE, CF, CW, DE, DF, DW, EF, EW, FW.

L_2

BC 4
BE 3
BF 3
CD 4
CE 4
CF 4
CW 3
DE 3

C_3:

BCE, BCF, CDE

L_3

BCE 3
BCF 3
CDE 3

C_4

None.

Benefit:
The map reduce function can help process the map reduce function in parallel.