COMP9318 Assignment 1 Solution

Ran Bai z5187292

2020 March

**Question 1:**

(1)

The table as follow,

|  |  |  |  |
| --- | --- | --- | --- |
| **Location** | **Time** | **Item** | **SUM(Quantify)** |
| Sydney | 2005 | PS2 | 1400 |
| Sydney | 2005 | ALL | 1400 |
| Sydney | 2006 | PS2 | 1500 |
| Sydney | 2006 | Wii | 500 |
| Sydney | 2006 | ALL | 2000 |
| Sydney | ALL | PS2 | 2900 |
| Sydney | ALL | Wii | 500 |
| Sydney | ALL | ALL | 3400 |
| Melbourne | 2005 | XBox 360 | 1700 |
| Melbourne | 2005 | ALL | 1700 |
| Melbourne | ALL | XBox 360 | 1700 |
| Melbourne | ALL | ALL | 1700 |
| ALL | 2005 | PS2 | 1400 |
| ALL | 2005 | XBox 360 | 1700 |
| ALL | 2005 | ALL | 3100 |
| ALL | 2006 | PS2 | 1500 |
| ALL | 2006 | Wii | 500 |
| ALL | 2006 | ALL | 2000 |
| ALL | ALL | PS2 | 2900 |
| ALL | ALL | Wii | 500 |
| ALL | ALL | XBox 360 | 1700 |
| ALL | ALL | ALL | 5100 |

(2)

Equivalent SQL statement as below:

SELECT Location, Time, Item, SUM(Quantity)

FROM R

GROUP BY Location, Time, Item

UNION ALL

SELECT Location, Time, ALL, SUM(Quantity)

FROM R

GROUP BY Location, Time

UNION ALL

SELECT Location, ALL, Item, SUM(Quantity)

FROM R

GROUP BY Location, Item

UNION ALL

SELECT ALL, Time, Item, SUM(Quantity)

FROM R

GROUP BY Time, Item

UNION ALL

SELECT Location, ALL, ALL, SUM(Quantity)

FROM R

GROUP BY Location

UNION ALL

SELECT ALL, Time, ALL, SUM(Quantity)

FROM R

GROUP BY Time

UNION ALL

SELECT ALL, ALL, Item, SUM(Quantity)

FROM R

GROUP BY Item

UNION ALL

SELECT ALL, ALL, ALL, SUM(Quantity)

FROM R

(3)

The iceberg cube is

|  |  |  |  |
| --- | --- | --- | --- |
| Location | Time | Item | SUM(Quantity) |
| Sydney | 2006 | ALL | 2000 |
| Sydney | ALL | PS2 | 2900 |
| Sydney | ALL | ALL | 3400 |
| ALL | 2005 | ALL | 3100 |
| ALL | 2006 | ALL | 2000 |
| ALL | ALL | PS2 | 2900 |
| ALL | ALL | ALL | 5100 |

(4)

Use the function

f(Location, Time, Item) = 12 \* Location + 4 \* Time + Item

Step 1:

Mappings tables

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Location | Value | Time | Value | Item | Value |
| ALL | 0 | ALL | 0 | ALL | 0 |
| Sydney | 1 | 2005 | 1 | PS2 | 1 |
| Melbourne | 2 | 2006 | 2 | Xbox 360 | 2 |
|  |  |  |  | Wii | 3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Location** | **Time** | **Item** | **SUM(Quantify)** | **offset** |
| 1 | 1 | 1 | 1400 | 17 |
| 1 | 1 | 0 | 1400 | 16 |
| 1 | 2 | 1 | 1500 | 21 |
| 1 | 2 | 3 | 500 | 23 |
| 1 | 2 | 0 | 2000 | 20 |
| 1 | 0 | 1 | 2900 | 13 |
| 1 | 0 | 3 | 500 | 15 |
| 1 | 0 | 0 | 3400 | 12 |
| 2 | 1 | 2 | 1700 | 30 |
| 2 | 1 | 0 | 1700 | 28 |
| 2 | 0 | 2 | 1700 | 26 |
| 2 | 0 | 0 | 1700 | 24 |
| 0 | 1 | 1 | 1400 | 5 |
| 0 | 1 | 2 | 1700 | 6 |
| 0 | 1 | 0 | 3100 | 4 |
| 0 | 2 | 1 | 1500 | 9 |
| 0 | 2 | 3 | 500 | 11 |
| 0 | 2 | 0 | 2000 | 8 |
| 0 | 0 | 1 | 2900 | 1 |
| 0 | 0 | 3 | 500 | 3 |
| 0 | 0 | 2 | 1700 | 2 |
| 0 | 0 | 0 | 5100 | 0 |

Step 2:

So the final result is:

|  |  |
| --- | --- |
| ArrayIndex | Value |
| 17 | 1400 |
| 16 | 1400 |
| 21 | 1500 |
| 23 | 500 |
| 20 | 2000 |
| 13 | 2900 |
| 15 | 500 |
| 12 | 3400 |
| 30 | 1700 |
| 28 | 1700 |
| 26 | 1700 |
| 24 | 1700 |
| 5 | 1400 |
| 6 | 1700 |
| 4 | 3100 |
| 9 | 1500 |
| 11 | 500 |
| 8 | 2000 |
| 1 | 2900 |
| 3 | 500 |
| 2 | 1700 |
| 0 | 5100 |

**Question 2:**

The process of performing group average hierarchical clustering on given similarity matrix as below,

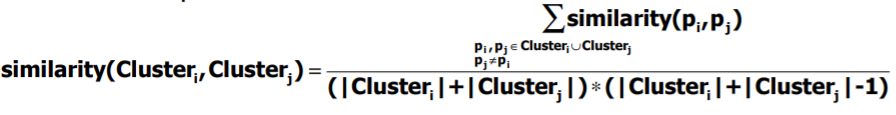
Step 1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | p1 | p2 | p3 | p4 | p5 |
| p1 | 1.00 | 0.10 | 0.41 | 0.55 | 0.35 |
| p2 | 0.10 | 1.00 | 0.64 | 0.47 | 0.98 |
| p3 | 0.41 | 0.64 | 1.00 | 0.44 | 0.85 |
| p4 | 0.55 | 0.47 | 0.44 | 1.00 | 0.76 |
| p5 | 0.35 | 0.98 | 0.85 | 0.76 | 1.00 |

From table above, we can know that combine p5 with p2 to a new cluster, named p2 instead.

Step 2:

Calculate the new similarity matrix by the given formula,



Cluster now as below,

New p1:{p1}, new p2:{p2, p5}, new p3:{p3}, new p4:{p4}

So, the only change inter-cluster similarity are all about p2,



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | p1 | p2 | p3 | p4 |
| p1 | 1.00 | 0.48 | 0.41 | 0.55 |
| p2 | 0.48 | 1.00 | 0.82 | 0.74 |
| p3 | 0.41 | 0.82 | 1.00 | 0.44 |
| p4 | 0.55 | 0.74 | 0.44 | 1.00 |

From table above, we know that we should combine p2 with p3, named p2 instead.

Step 3:

Cluster now are,

New p1:{p1}, new p2:{p2, p3, p5}, new p3:{p4}





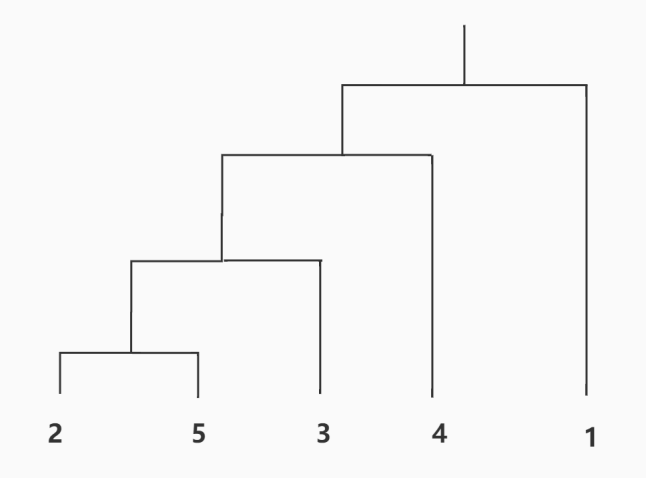
|  |  |  |  |
| --- | --- | --- | --- |
|  | p1 | p2 | p3 |
| p1 | 1.00 | 0.56 | 0.55 |
| p2 | 0.56 | 1.00 | 0.69 |
| p3 | 0.55 | 0.69 | 1.00 |

From table above, we can know that it should combine p2 with p3, name p2 instead.

Step 4:

Cluster now are:

New p1:{p1}, new p2:{p2, p3, p4, p5}, and the dendrogram is:



**Question 3:**

(1)

The stopping criterion is till the algorithm converges to the final k clusters. So the algorithm is as below,

Algorithm 1: k-means(D, k)

1 Initialize k centers C = [c1, c2, . . . , ck];

2 canStop ← false;

3 while canStop = false do

4 Initialize k empty clusters G = [g1, g2, . . . , gk];

5 for each data point p ∈ D do

6 cx ← NearestCenter(p, C);

7 gcx .append(p);

Copy list C to list C’

8 for each group g ∈ G do

9 ci ← ComputeCenter(g);

If evey center in list C’equal to list C:

canStop = True;

10 return G;

(2)

In every iteration, the k-means algorithm only do two things, update the classification of m points and take the mean point as center of cluster. As below, through these two steps, I will explain why the cost of k cluster as evaluated at the end of each iteration never increases.

1. The n points are assigned to the existing k centers, according to the rules that assign the point to the closest center. Compared with assigning the point with other ways, the method can reduce the cost function. This is because  is smaller with the way of clustering the node to the closest center. So  and  will decrease.
2. Assume a category have N points, and its center is . Now the cost function of this category is  , and we know which x is a random point. So this steps decrease the cost function, too.

In conclusion, the total cost will not increase.

(3)

We know the distance will not less than 0, so 

not less than 0. Also, is not less than 0.

After that, the loop in the algorithm is finite. This is because the possible of cluster is finite(It is , k is number of cluster and n is number of node). Using conclusion of question 2, we can know that the total cost of cluster will never increased.

So the worst case is that it will get the value under  decrease. It will get the local minimum which is not less than 0.

In conclusion, it always converges to a local minimum.