Homework 3 : K-Means

108062608 黃柏翰

Map Reduce Algorithm and Code

演算法根據老師第七章 p.34

- **1)** For each point, place it in the cluster whose current centroid it is nearest
- **2)** After all points are assigned, update the locations of centroids of the ***k*** clusters
- **3)** Reassign all points to their closest centroid
  - Sometimes moves points between clusters
- **Repeat 2 and 3 until convergence**

因為題目要求使用不同的距離量測方式，所以在距離計算上面會寫成不同的 function

a. Map

此 Mapper 的功能為做計算各 node 到各 centroid 的距離，並 assign 各 node 到各 centroid，為題目要求使用不同的距離量測方式所以，會有兩種 mapper

程式碼

```python
def Mapper_Euclidean(data):
    # calculate_Euclidean_distance
    input_data = data.map(calculate_Euclidean_distance)

    # Assign to new centroid
    key_value = Assign2new_centroid(input_data)

    return key_value


def Mapper_Manhattan(data):
    # calculate_Euclidean_distance
    input_data = data.map(calculate_Manhattan_distance)

    # Assign to new centroid
    key_value = Assign2new_centroid(input_data)

    return key_value
```

所呼叫的副程式程式碼

```python
def Assign2new_centroid(input_data):
    # find the minimum distance to each centroid
    # Assign to new cluster
    # (cluster, node)
    # value, key, then reduce by key & average, you will get new centroid
    key_value = input_data.map(lambda x : (x[0][1].index(min(x[0][1])), x[0][0]) )

    return key_value
```

b. Reduce

此 Reducer 的功能為計算新的 centroid，因為找 centroid 跟距離量測方式無關，所以只需一個 Reducer

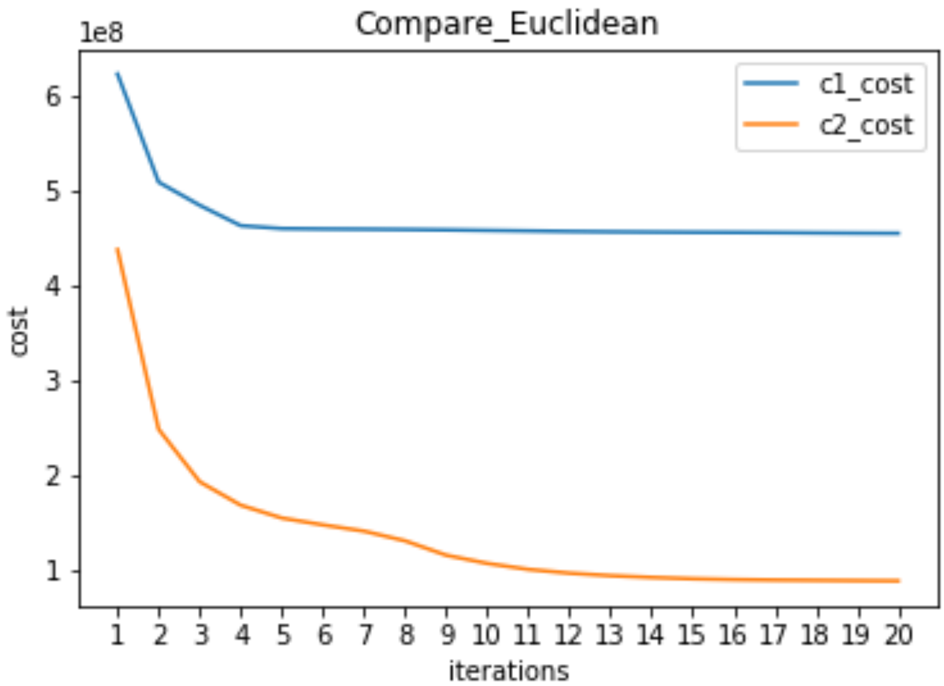
程式碼

```python
def Reducer(key_value):
    global countsByKey, centroid
    # find_new_centroid
    # find the number of value for each key
    countsByKey = key_value.countByKey()
    # accumulate same key value from same cluster
    key_value = key_value.reduceByKey(lambda x,y : add(x,y))

    key_value = key_value.map(find_new_centroid)

    #update centroid
    centroid = key_value.collect()

    return centroid
```

Homework 回答

以 Euclidean distance 做 K-means

1. Performance 比較



2. After 10 iterations, about centroid distance

2.1 探討進步程度

Using Euclidean with c1 as initial centroid improves 0.26 %
Using Euclidean with c2 as initial centroid improves 0.75 %

2.2 c1, c2 performance 比較

c2 在整體 performance 上不管是改善 cost，或者 cost 的大小上都比 c1 來的更好，因為 kmeans 的初始化很重要，所以隨機的初始化的結果很可能比較差

2.3 Centroid 距離比較表格

使用 c1 下用 Euclidean distance

Index	0	1	2	3	4	5	6	7	8	9
0	0	625.816	3582.48	168.285	276.57	421.768	358.917	474.076	1325.68	243.741
1	0	0	2956.94	789.057	901.809	1047.51	984.61	1099.87	726.91	394.749
2	0	0	0	3745.8	3858.72	4004.24	3941.39	4056.5	2317.87	3349.04
3	0	0	0	0	116.467	262.246	199.784	314.758	1476.14	398.084
4	0	0	0	0	0	146.37	83.5532	198.949	1592.47	513.183
5	0	0	0	0	0	0	62.9545	52.6113	1738.32	659.55
6	0	0	0	0	0	0	0	115.5	1675.91	596.681
7	0	0	0	0	0	0	0	0	1790.75	712.122
8	0	0	0	0	0	0	0	0	0	1082.21
9	0	0	0	0	0	0	0	0	0	0

使用 c1 下用 Manhattan distance

Index	0	1	2	3	4	5	6	7	8	9
0	0	701.963	3910.19	199.13	291.941	466.67	394.132	533.688	1741.41	325.649
1	0	0	3211.99	841.485	991.15	1166.76	1093.85	1233.66	1041.37	416.089
2	0	0	0	4047.01	4199.68	4374.45	4302.06	4441.23	2465.58	3589.35
3	0	0	0	0	154.623	330.521	257.26	397.688	1877.04	462.341
4	0	0	0	0	0	176.833	103.548	244.056	2029.84	615.201
5	0	0	0	0	0	0	73.8313	68.5145	2205.62	790.873
6	0	0	0	0	0	0	0	141.217	2132.31	717.766
7	0	0	0	0	0	0	0	0	2272.61	857.679
8	0	0	0	0	0	0	0	0	0	1418.22
9	0	0	0	0	0	0	0	0	0	0

使用 c2 下用 Euclidean distance

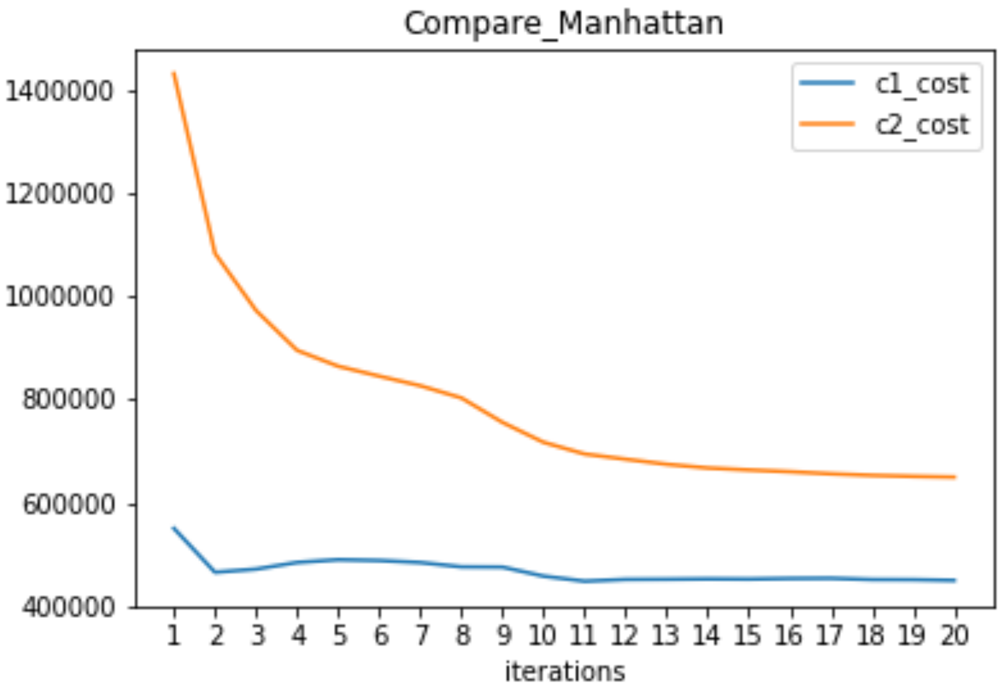
Index	0	1	2	3	4	5	6	7	8	9
0	0	15740.6	14093.8	9025.59	5548.15	1829.19	1385.96	584.708	2291.74	3138.74
1	0	0	11524.5	6743.88	10192.5	14499.4	14385.6	15161.6	13485.9	12606
2	0	0	0	9545.88	10883.4	12304.8	12959.8	13632.4	12300.1	11964
3	0	0	0	0	3494.22	7760.68	7657.99	8443.15	6753.8	5886.94
4	0	0	0	0	0	4434.41	4198.24	4969.25	3307.4	2415.82
5	0	0	0	0	0	0	1025.89	1408.37	1338	2167.13
6	0	0	0	0	0	0	0	803.39	908.038	1782.72
7	0	0	0	0	0	0	0	0	1707.37	2557.03
8	0	0	0	0	0	0	0	0	0	906.256
9	0	0	0	0	0	0	0	0	0	0

使用 c2 下用 Manhattan distance

Index	0	1	2	3	4	5	6	7	8	9
0	0	15750.4	20191.1	9508.51	5579.75	2921.72	1647.79	669.275	2654.66	3290.3
1	0	0	16003.5	7219.2	10221	16064.3	14674.5	15254.7	13875.7	12761
2	0	0	0	10690.5	14613.6	17571.1	18550.6	19527	17546.1	16907.5
3	0	0	0	0	3935.29	8855.64	7866.3	8843.42	6863.67	6224.91
4	0	0	0	0	0	5852.28	4462.2	5041.83	3663.56	2548.21
5	0	0	0	0	0	0	1396.8	2255.59	2197.68	3311.73
6	0	0	0	0	0	0	0	980.521	1011.91	1919.22
7	0	0	0	0	0	0	0	0	1988.05	2628.17
8	0	0	0	0	0	0	0	0	0	1116.81
9	0	0	0	0	0	0	0	0	0	0

以 Manhattan distance 做 K-means

1. Performance 比較



2. After 10 iterations, about centroid distance

2.1 探討進步程度

Using Manhattan with c1 as initial centroid improves 0.17 %
Using Manhattan with c2 as initial centroid improves 0.50 %

2.2 c1, c2 performance 比較

c1 performance 上(cost)表現比 c2 佳，進步上比 c2 差因為 kmeans 的初始化很重要，所以隨機的初始化的結果很可能使進步程度比較差，但是因為 Manhattan 距離量測方法的考量為各維度的點差距和，所以 cost 結果可能跟 Euclidean 不同

2.3 centroid 距離比較表格

使用 c1 下用 Euclidean distance

Index	0	1	2	3	4	5	6	7	8	9
0	0	2075.44	8105.39	283.877	287.871	526.528	443.09	586.47	1035.07	527.495
1	0	0	6046.23	2345.52	2361.3	2600.36	2517.36	2659.96	1419.4	1548.31
2	0	0	0	8362.95	8393.2	8631.8	8548.3	8691.77	7251.74	7583.42
3	0	0	0	0	130.219	291.475	215.713	348.231	1211.29	803.006
4	0	0	0	0	0	239.129	156.089	298.926	1289.84	813.043
5	0	0	0	0	0	0	83.5189	60.0299	1502.64	1052.14
6	0	0	0	0	0	0	0	143.528	1424.49	969.076
7	0	0	0	0	0	0	0	0	1559.28	1111.82
8	0	0	0	0	0	0	0	0	0	718.807
9	0	0	0	0	0	0	0	0	0	0

使用 c1 下用 Manhattan distance

Index	0	1	2	3	4	5	6	7	8	9
0	0	2213.76	9426.57	366.364	344.049	610.19	510.356	685.989	1559.21	549.721
1	0	0	7219.36	2427.98	2553.21	2818.57	2719.47	2894.15	1961.98	1668.07
2	0	0	0	9618.08	9766.7	10032.9	9934.03	10107.1	8011.32	8881.9
3	0	0	0	0	160.37	425.723	325.338	501.2	1748.31	877.023
4	0	0	0	0	0	267.392	167.905	343.326	1902.24	888.59
5	0	0	0	0	0	0	100.806	76.9709	2167.9	1155.06
6	0	0	0	0	0	0	0	176.919	2067.5	1055.31
7	0	0	0	0	0	0	0	0	2243.7	1230.86
8	0	0	0	0	0	0	0	0	0	1015.47
9	0	0	0	0	0	0	0	0	0	0

使用 c2 下用 Euclidean distance

Index	0	1	2	3	4	5	6	7	8	9
0	0	15732.9	14088.1	9017.85	5540.42	1995.19	1642.43	638.695	1541.17	3109.03
1	0	0	11524.5	6743.88	10192.5	14474.6	14093.9	15100.6	14342.3	12631.1
2	0	0	0	9545.88	10883.4	12167.8	12904.4	13584	12658.7	11940.2
3	0	0	0	0	3494.22	7742.63	7375.51	8381.75	7602.66	5909.02
4	0	0	0	0	0	4452.97	3901.87	4908.31	4180.9	2443.02
5	0	0	0	0	0	0	1421.41	1565.49	915.328	2198.45
6	0	0	0	0	0	0	0	1006.87	582.579	1466.61
7	0	0	0	0	0	0	0	0	945.484	2472.77
8	0	0	0	0	0	0	0	0	0	1755.61
9	0	0	0	0	0	0	0	0	0	0

使用 c2 下用 Manhattan distance

Index	0	1	2	3	4	5	6	7	8	9
0	0	15743	20183	9500.33	5571.62	3264.18	1748.62	731.677	2063.88	3304.93
1	0	0	16003.5	7219.2	10221	16325.3	14205.6	15203.1	14987.6	12832.8
2	0	0	0	10690.5	14613.6	17521.5	18438.4	19457	18133.1	16884.5
3	0	0	0	0	3935.29	9116.02	7754.1	8773.45	7778.79	6201.85
4	0	0	0	0	0	6110.83	3992.44	4990.31	4776.51	2619.96
5	0	0	0	0	0	0	2128.67	2537.03	1351.29	3503.29
6	0	0	0	0	0	0	0	1024.15	786.781	1558.35
7	0	0	0	0	0	0	0	0	1333.97	2578.91
8	0	0	0	0	0	0	0	0	0	2159.78
9	0	0	0	0	0	0	0	0	0	0