MDA_HW3

102062130 劉鳳軒

1. **Overview 架構：**

(1) 先把 centeroid 資料的檔案在本地用 filesystem 吃進來(一開始是 c1.txt 或 c2.txt，之後會用新的 centroid-r-00000)， 之後用 conf.setDouble 的方式將 10 個 centroid 的存進 conf 中(其中 name property 為 C_i_j ，i 為 centroid 編號、j 為該 centroid 的 feature 編號)
(2) 進入 Map Reduce 產生出新的 centroid 檔案(centroid-r-00000)，同時產生一個 cost 檔案紀錄每個 cluster 的 cost(cost-r-00000)
(3) 把 cost-r-00000 中每個 cost 加起來即是該次的總 cost，並記錄在本地的 cost.csv 中(一次一筆，最後會有 20 筆)
(4) 回到 1 並做 20 次

檔案說明:

Kmeans.java: kmeans code
Cost_c1_E.xlsx: 用 c1 做 Euclidean distance 20 次的結果(附圖)
Cost_c2_E.xlsx: 用 c2 做 Euclidean distance 20 次的結果(附圖)
Cost_c1_M.xlsx: 用 c1 做 Manhattan distance 20 次的結果(附圖)
Cost_c2_M.xlsx: 用 c2 做 Manhattan distance 20 次的結果(附圖)

2. **Mapper**

Input: LongWritable, Text
Output: IntWritable, Text
let point a has features a1, a2, …, a58

	Key		Value	
Input	Type	Format	Type	Format
	LongWritable		IntWritable	a1, a2, …., a58
Output	IntWritable	Format	IntWritable	Format
	IntWritable	centroidID	Text	a1, a2, …., a58, distance(centroid, a)

將一個個 point 讀進來後，對每個 centroid 做距離計算(用

Euclidean 或 Manhattan)，並把該 point 分到最近的 centroid 的 cluster 中，並以 centorid id 為 key, 該 point 的 features 加上最近 distance 為 value 輸出。

3. **Reducer**

Input: IntWritable, Iterable< Text >
Output: Null, Text

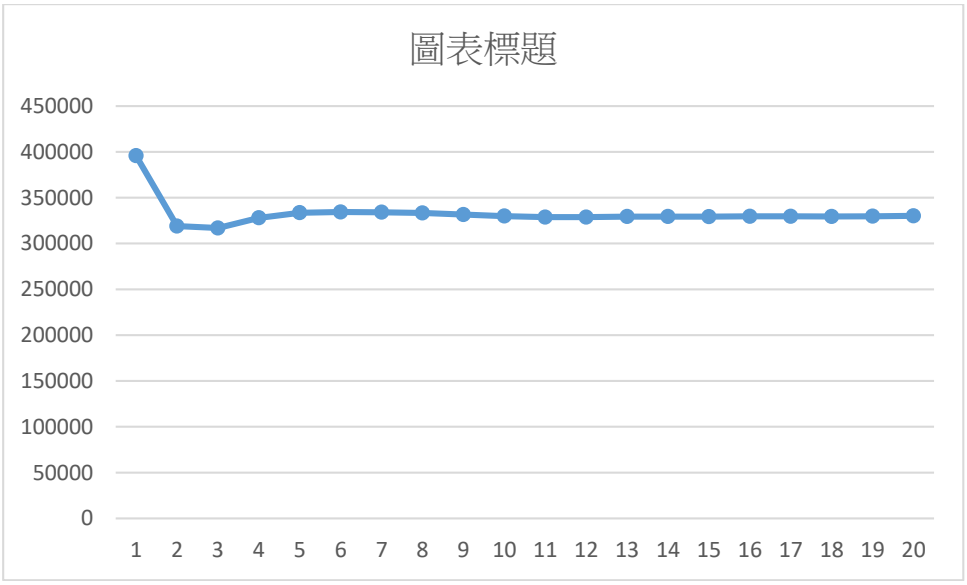
	Key		Value	
Input	Type	Format	Type	Format
	IntWritable	centroidID	Text	**Iterable**<(a1, a2, …., a58, distance(centroid, a))>
Output1: New Centroids	Type	Format	Type	Format
	NullWritable	Null	Text	58 float values (new centroid)
Output2: Costs per Cluster	Type	Format	Type	Format
	NullWritable	Null	Text	New Cost

用 cluster 中的每個點算出新的 centroid(取平均)並把每個點的 distance 加起來成為 cost，最後把 centroid 輸出一個檔案，cost 輸出到另一個檔案

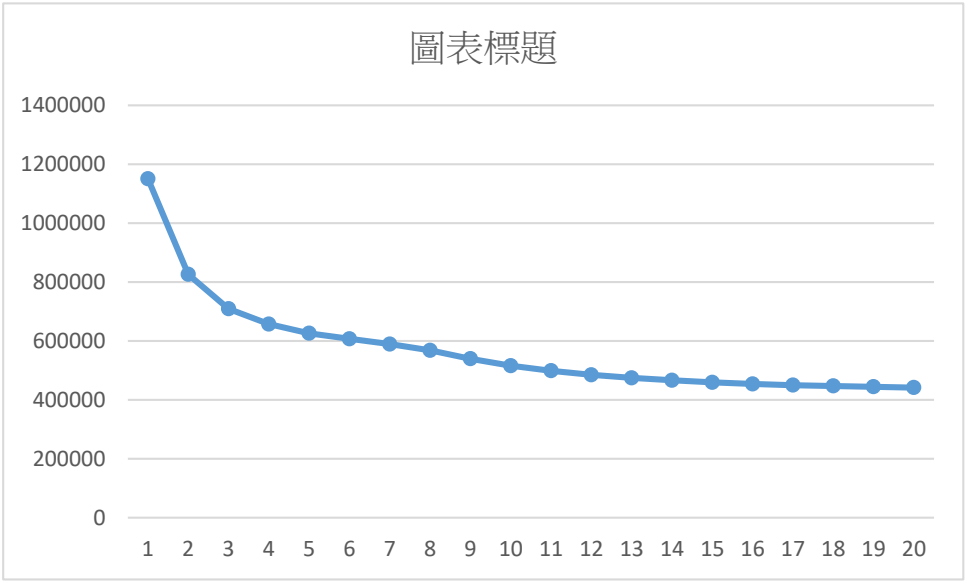
4. 問題討論

- Euclidean 中用 c1 跟 c2 差別

  c1:



C2:

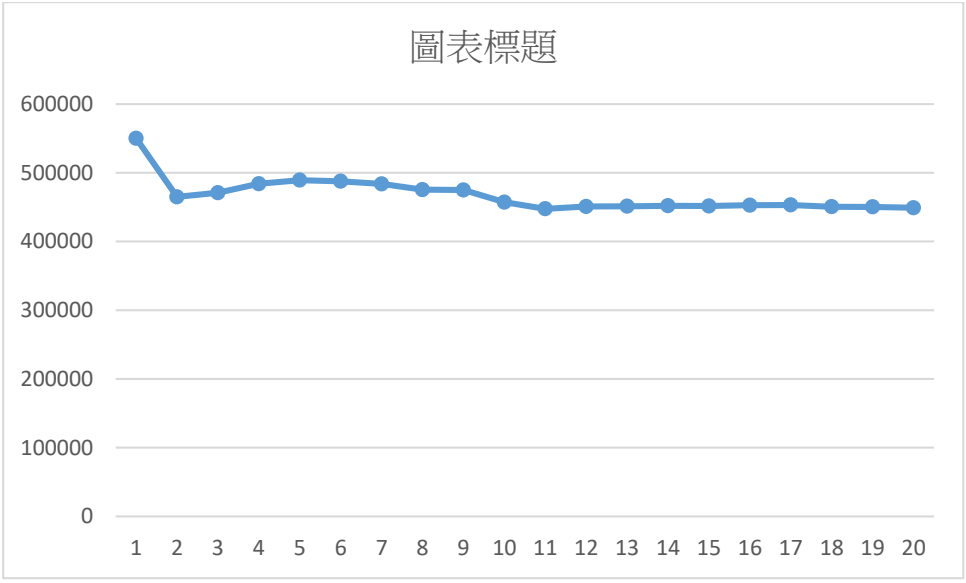


可以看到 c2 的曲線更平滑地收斂，且下降的比率也高很多。

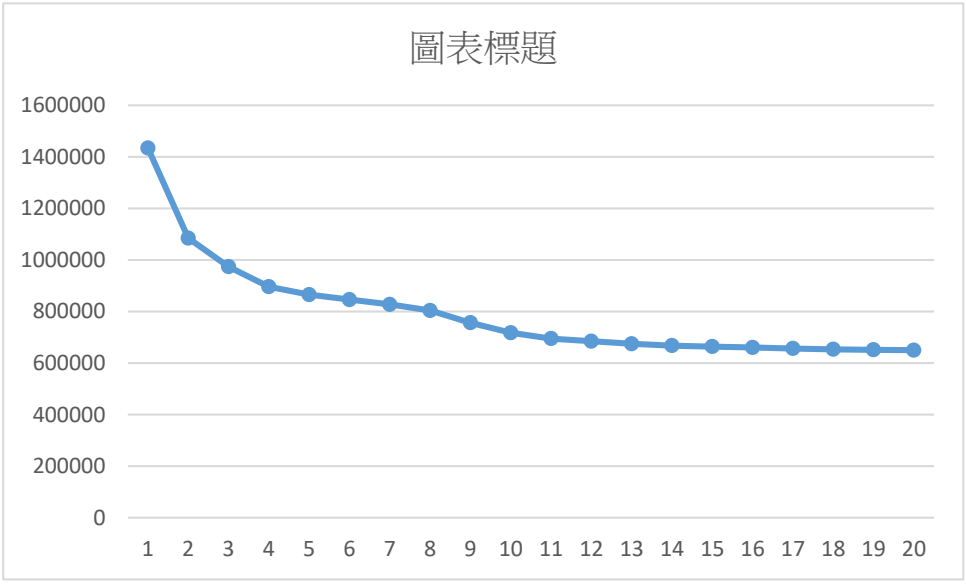
不過最後是 c1 的 cost 較低，代表平均而言 cluster 內的相似度更高(歸類更精確)，所以較佳。

- Manhattan 中用 c1 跟 c2 差別

  c1:



c2:



同樣地，可以看到 c2 的曲線更平滑地收斂，且下降的比率也高很多。

不過最後是 c1 的 cost 較低，代表平均而言 cluster 內的相似度更高(歸類更精確)，所以較佳。