國立臺灣大學電資學院電機工程學研究所

資料科學 期末專案

Graduate Institute of Industrial Engineering

College of Electrical Engineering

National Taiwan University

Final Project of Data Science

非監督式學習自關係網路中的社交圈

Unsupervised Learning Social Circles in Ego-networks

曾亮軒 (R11921067)、潘宣蓉 (B07502037)、

石旻翰 (B08502141)、陳志臻 (B07502071)

指導教授: 陳銘憲、駱明凌 老師

中華民國 112 年 1 月

January, 2023

目錄

第一章　Introduction

1.1　Introduction

在這個 work 中，我們的目標是要透過一個使用者的 ego-network 與這些用戶的特徵，來找出他的朋友圈。透過對社群網路使用者進行分群有許多用途，例如可以推荐可能認識的人，促進用戶之間的交流與連繫，或是更容易的投放廣告，針對類似族群投放相近性質的廣告。在我們的 work 中，我們會以兩個方向來研究這比資歷，分別是 Graph Clustering-based 的演算法，能根據圖形的形狀來進行分群，以及 Density Peaks-based 的方法，並改進現有的算法。

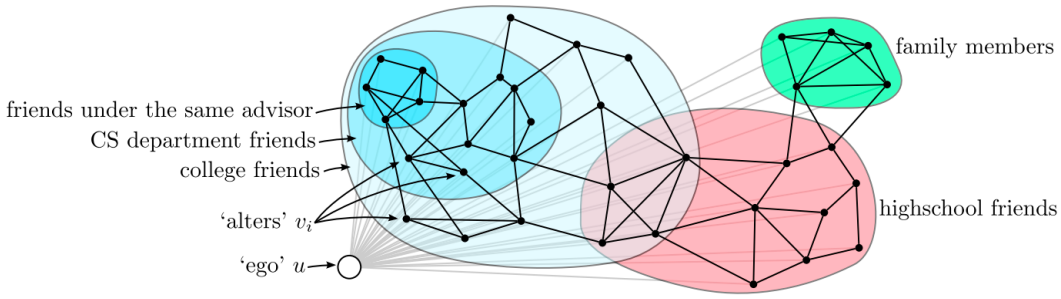

Figure 1.1: An illustration of ego-network and social circles.

第二章　Dataset and Metric

2.1　Dataset Description

在助教提供的 dataset 中，包含了以下資料夾/檔案:

1. egonets/: 資料夾中每個檔案為一個使用者的 ego-networks，用來儲存這個使用者所有朋友與其他朋友之間的關係。
2. Training/: 為 Ground Truth，是由使用者自行標記出的 social circle，每一個 circle 代表了使用者自己認定的朋友圈。
3. featureList.txt: 紀錄所有可能的用戶特徵。
4. features.txt: 所有用戶所擁有的特徵，可作為特徵向量使用。
5. sample_submission.csv: 範例繳交檔案。
6. testSet_users_friends.csv: 測試資料。(但因為 Kaggle 無法繳交 prediction，也找不到當初 testset 的 ground truth，故我們沒有使用到這個檔案)

2.2　Evaluation Metric

我們使用 dataset 所提供的 sample code socialCircles_metric.py 來評估我們預測的結果。在 socialCircles_metric.py 裡，會計算 predicted circles 與 ground-true circles 兩兩之間的 cost，任兩個 circles 之間的 cost function 如下式 (一)：

$$Cost_{ij} = \text{number of } (C^i_{pred} \cup C^j_{gt}) - \text{number of } (C^i_{pred} \cap C^j_{gt}) \tag{2.1}$$

獲得了 cost matrix 後，我們可以透過 Munkres 演算法 [] 來計算可以造成最小 cost 的路徑，作為 ground-true circles 以及 predicted circles 的最佳的 alignment 關係；同時這個最小的 cost 也就是我們的 evaluation metric 以及要 minimize 的 objective。在以下的實驗中，我們會將 training set 所有使用者的 circles 與預測出來的 social circles 去做計算並總和所有使用者的 cost 當作模型的表現數據參考依據。

第三章　Graph Clustering-based Social Circle Discovery

3.1　Graph of ego-network and clustering

首先，我們能將 egonet 轉換成一個一個 graph，並嘗試以 graph clustering 的方式來預測 social circles。我們的輸入是一個 egonet 的 graph G = (V, E)，以及每個用戶 $v(v \in V)$ 的個人資料，又或是所謂的 features。Egonet 的中心點，也就是使用者本人 u，並不包含在 graph G 裏面，graph G 僅由使用者 u 的朋友們組成。而我們的目標是預測一組 circle $C = \{C_1, C_2, ...C_k\}$，代表這個使用者可能的 social circles。

在這個段落裡，我們將嘗試直接使用 Non-Negative Matrix Factorization (NMF) 與 Spectral Clustering 兩種不同的分群演算法，來對我們的 data 進行分群以預測 social circles，並且將結果進行比較。

3.2　Spectral Clustering [10] [12] [3]

Spectral clustering 是一種非監督式的分群演算法，透過資料之間的親和度 (affinity) 來對資料分群，親和度一般以資料間的相似度來計算。在 Spectral Clustering 中，相似度計算方法為高斯核相似函數 (Gaussian Kernel Simlarity)[14]，如下式 (3.1)

$$K(x_i, x_j) = exp\{\frac{-d(x_i, x_j)}{\sigma^2}\} \tag{3.1}$$

其中，$d(x_i, x_j)$ 為歐氏距離 (Euclidean distance)，定義如下式 (3.2)

$$d(x_i, x_j) = \|x_i - x_j\|^2 = \Sigma_{k=0}^{D}(x_{ik} - x_{jk})^2 \tag{3.2}$$

由於 Spectral Clustering 可以應用在 Graph clustering 上，透過將圖 G 轉換成 adjacency matrix 並對其 Laplacian Graph 做降維，找出前 K 個最小的特徵值與特徵向量，將原本的資料投影到這 K 個特徵向量上，最後再對投影的結果進行分群，如 k-mean clustering，選擇使用 Spectral Clustering 除了適合做在圖 (Graph) 上外，也因為 Spectral Clustering 適合處理資料點形狀複雜的分類問題。

3.3 Dynamic Spectral Clustering [4]

我們參考 [7][4] 內的作法去做預測，作法如下：

1. 設有一個變數 n，1< n < n_cluster 的數量。
2. 對每一個 n 都去做 Spectural Clustering，並做出預測。
3. 再將預測的結果做 partion，做出一個 partition dict。
4. 最後，將上述的 partion dict 與 Graph G 之間的 modularity 算出來。
5. 將所有 n 所對應的 modularity 最大的取出來，並將此 modularity 所對應 n 的 pred_circle 當作最後的預測結果。計算預測結果與 ground truth 之間的 cost。

我們做了 original spectural clustering 與 dynamic spectural clustering 的比較實驗，結果如下表 (一):

	n_cluster	Cost		n_cluster	Cost
Original	8	14244	Dynamic	8	14488
	10	14700		10	14480
	15	15812		15	14508
	20	17082		20	14674

可以看出當 K-means 的 n_cluster 數量越大的時候，Dynamic 的幫助越大，但在其較小的時候，使用原本的 spectural clustering 演算法就有不錯的表現了。然而，我們也發現，整體來說 Dynamic 的表現比較平均，較不會受到 cluster 數量的影響。

3.4 Non-Negative Matrix Factorization (NMF)[5]

非負值矩陣方解 (Non-Negative Matrix Factorization)，是一種矩陣的分解方法，它假設數據與成份的值都不是負的。在矩陣不包含負值情況下，我們可以使用 NMF 來分解一個矩陣，通過優化矩陣之間的乘積，使分解後的矩陣乘積與原矩陣更為相似，來更新分解的權重。給定一個矩陣 X，X = WH，透過優化 X 與 WH 乘積之間的距離，來獲得最佳的分解方式。在優化的過程中，會先固定 W 並更新 H，再固定 H 更新 W。一般常用的距離計算方式為 squared Frobenius norm，如下式 (3.3[9])。

$$d_{Fro}(X,Y) = \frac{1}{2}\|X-Y\|_{Fro}^2 = \frac{1}{2}\Sigma_{i,j}(X_{ij}-Y_{ij})^2 \tag{3.3}$$

更新 W, H 的方法如下式 (3.4)、式 (3.5)[11]。透過不斷迭代更新，我們就可以求出最佳的 H，W。

$$H_{[i,j]}^{n+1} = H_{[i,j]}^{n} * \frac{((W^n)^T V)_{[i,j]}}{((W^n)^T W^n H^n)_{[i,j]}} \tag{3.4}$$

$$W_{[i,j]}^{n+1} = W_{[i,j]}^{n} * \frac{(V(H^{n+1})^T)_{[i,j]}}{(W^n H^{n+1}(H^{n+1})^T)_{[i,j]}} \tag{3.5}$$

我們參考 [1]，首先將 graph G 轉換成負的 Laplacian Metric L = A - D，在此 D 是 G 的 degree matrix，A 是 G 的 adjacency matrix。接著我們將 L 乘上一個 hyperparameter β 並計算其 matrix exponential 獲得 Lexp，接著我們將 matrix exponential 計算得到的 Lexp 做以下處理，將每個元素除以對應的行與列上的對角元素相乘開根號，如下式 (3.7)，並得到一個全為正值且對角皆為 1 的矩陣 B：

$$B_{[i,j]} = \frac{Lexp_{[i,j]}}{\sqrt{Lexp_{[i,i]} * Lexp^{[j,j]}}} \tag{3.6}$$

接著，先隨機初始化一個 NMF 模型，大小為 N_comp(default=20)。再來，我們使用此 NMF 將 B 分解成 W 和 H。對 W 的每一列 i，取出 W[:, i] 中大於

threshlod(default=0.5) 的值所對應的 index，並將此 index 所對應的 node u 塞進 pred_circle 裡。最後，將 pred_circle 中，長度大於 min_circle (default=5) 的 circle 當作我們的 output，並與 ground truth 計算 cost。另外，我們藉由調整 NMF 的 hyperparameters，來研究各個參數如何影響預測的好壞。如上所述，各個參數的預設值為: $(beta, threshold, min\_circle) = (0.2, 0.5, 5)$ 我們將在以下的實驗中，每次只動一個變數，去探討各個參數對 cost 的影響，並嘗試找到各個參數的設定應該如何調整，如：應該調大/調小，並在最後的討論中，使用我們找到的最好參數設定去做預測。

beta	Cost	Threshold	Cost	min_circle	Cost
0.1	16095	0.4	15269	5	15468
0.2	15468	0.5	15468	8	15572
0.3	15255	0.6	15852	10	15395

由上表 (二) 可知，我們應該要把 beta 稍微調大、Thershold 稍微調小，以及將 min_circle 調成 10。故我們最後的參數設定會是: $(beta, threshold, min\_circle) = (0.3, 0.4, 10)$ 並將 clusterg 數量設定為 8，實驗的 cost 為：14146

第四章　Density Peaks-based Social Circle Discovery (DPSCD) [13]

4.1 Background

在正式介紹 DPSCD 之前，我們要知道這篇 paper 提出的方法是基於 Density Peaks-based Clustering (DPC) [8]。而 DPC 最主要的想法則是要解決 DBSCAN [2] 的一些問題。例如: 在 DBSCAN 中我們在最一開始就需要定義好 min-point 數量以及 distance-threshold 這兩個關鍵的參數，才能更進一步的使用 DBSCAN 來作 clustering，但如果這兩個參數設得不好，將會很大幅度的影響 DBSCAN clustering 的結果。而 DPC 提出了一個概念，讓我們可以先找出合理的 cluster centers，在來作後續的 assignment。這樣的方法使得我們可以兼具 DBSCAN 的優勢 (快速、不受空間限制)，同時又能夠確保找到合理的 cluster centers。

接下來，就讓我們來了解 DPC 如何找出 cluster centers。首先，我們可以看到在 Figure 4.1中，我們可以先根據圖 C 找出每個 data point 的 density，進而得到圖 A 的 density 等高線圖。然後，DPC 的基本概念就是要找出 density 的高峰來當作 cluster centers。那怎麼樣才算是一個好的 density 高峰呢? 除了 density 要越高越好之外，還有一個條件就是: 這個高峰到離他最近的更高峰的距離要越遠越好。因此我們可以定義一個式子來描述這個條件，如 Equation 4.1所示。在對於所有的點都計算完 δ 值後，我們就可以畫出 δ 與 ρ 的二維圖，並且透過這張圖當作 decision graph 去找出往右上方分離出具有高峰性值的這些點當作 cluster centers (如 Figure 4.1圖 E 中彩色的點)。

$$\delta_i = \begin{cases} \max_v \left(d_{i,v}\right) & \text{if user } i \text{ of the largest density} \\ \min_{v:\rho_v > \rho_i} \left(d_{i,v}\right) & \text{otherwise} \end{cases} \tag{4.1}$$

再來，剩下要做的事情就只剩下把剩下非中心的點指派到各個已經找出的 cluster

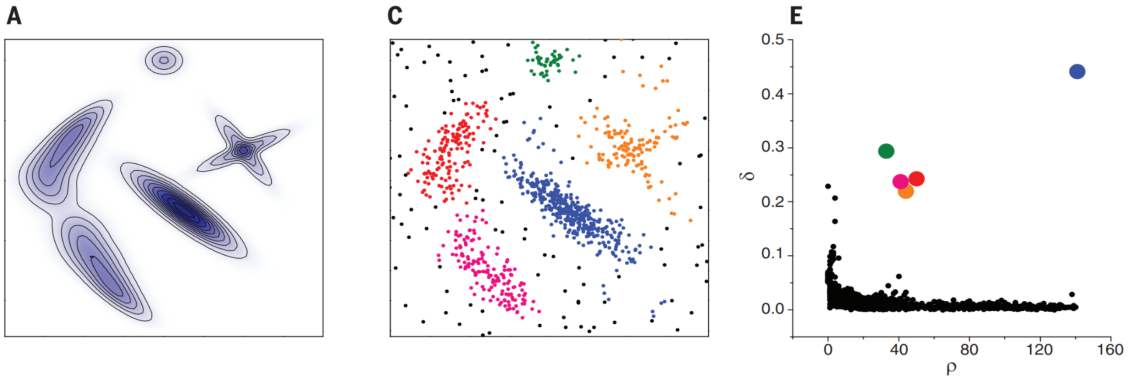

Figure 4.1: An illustration of how DPC finds the cluster centers.

centers 了。而 DPC 在指派的過程也不複雜，就是對每個點找出最接近且具有比該點還高的 density 的點作為 parent，並把自己的 cluster 指派成跟這個找到的 parent 一樣就結束了。

4.2 DPSCD

Density Peaks-based Social Circle Discovery (DPSCD) 則是將前面介紹的 DPC 的概念拓展到 Ego-network 上，用於找出 social circles。具體而言，可分為以下兩步驟: 1. 透過定義 distance 和 density 函數，使得 DPC 可以使用於 Ego-network 這種 graph 上作分群 2. 利用 Social Circle Integration，使得每個 node 可以存在於多個 cluster 之中。接下來，我們將分別詳述這兩個步驟具體的做法。

4.2.1 Apply DPC on Ego-networks

我們知道說原先 DPC 是用來對 embedding space 中的資料點作 clustering，而非能夠直接適用於 graph 上的分群。因此，DPSCD 透過自定義 distance function 以及 density function 來達到可以利用 DPC 分群的目的。以下我們先講解是 distance function (Equation 4.2) 在 paper 中的定義:

$$d_{i,v} = \frac{1}{\alpha \times \text{simP}(i,v) + (1-\alpha) \times \text{simN}(i,v)} \tag{4.2}$$

其中 $\text{simP}(i,v) = \sum_m P_i(m)\Delta P_v(m)$ 代表 $user_i$ 以及 $user_v$ 之間的 profile similarity，意義在於一一檢視 M 個 profile feature 維度，如果一樣就加一，不一樣則不加，因此 profile feature 越相近，proflie similarity 越高；而 $\text{simN}(i,v) = \frac{|\Gamma(i) \cap \Gamma(v)|}{\sqrt{|\Gamma(i)| \times |\Gamma(v)|}}$ 代表 $node_i$ 以及 $node_v$ 在 graph 中的 topological similarity，這邊 $\Gamma(i)$ 代表的是 $node_i$

的鄰居所形成的 set，因此當 $node_i$ 與 $node_v$ 的鄰居組成越相近，則 topological similarity 越高。最後，$\alpha \in [0, 1]$ 是一個 constant 用來對 profile similarity 以及 topological similarity 作 linear combination，而 distance 則定義為 similarity 的倒數，similarity 越高，distance 越近；反之，則越遠。

透過 distance function 的定義，我們可以接續著定義 density function，原始 paper 中的定義如下 (Equation 4.3):

$$\rho_i = \sum_v \exp\left(-\frac{\|d_{i,v} - d_c\|^2}{2\sigma^2}\right) \tag{4.3}$$

那 density 作的事情就是對每個 node 去加總該 node 到其他 node 的距離的總和，其中距離不會直接相加，而是經過一個 Gaussian kernel 後再相加。透過式子我們可以觀察到一個距離其他 node 都相對較近的 node，他的 density 也會比較高；如果一個 node 距離其他的 nodes 較為疏遠，他的 density 計算出來就會相對較低。以上的觀察也符合我們利用 DPC 來 cluster 的需求。最後，要利用 DPC 找出 cluster centers，除了 density (ρ) 之外我們還缺少了最後一樣東西，那就是 δ。DPSCD 的 δ value 的定義就跟 DPC 裡面一模一樣 (Equation 4.1)，因此我們這邊就不多作贅述了。有了 ρ 以及 δ 後，我們就可以跟 DPC 找 center 的方法一樣，畫出二維圖作為 decision graph，並利用該圖抓出 cluster centers。後續，我們依樣可以利用跟 DPC 中提到的一樣的方法來把每個非中心的 node 透過 linked-list 去指派並得到最後的一個 clusting 結果。

4.2.2 Social Circle Integration

作完了 clustering 後，DPSCD 還 proposed 了一個演算法，來進一步的提升 performance，也就是 Social Circle Integration。概念上來說，這個演算法的目的，就是讓每個 node 的可以去從屬於多於一個的 cluster。在 [6] 這篇論文中有提到一個概念是說 social circles 之間可能會有 overlap，而且這個情況並不少見。但是透過 DPC 得到的結果是一個 node 對到最多一個 cluster，因此 DPSCD proposed 了 circle integration 的方法來解決這個問題，讓 integrate 完的 circles 彼此之間能夠有 overlap。具體而言，原始 paper 中的作法就是將 circle 按照 center 的 density 大小作排序，接者 iterate 過每個 circle，讓小的 circle 中的 node 可以有機會從屬於大的 circle，如同 Figure 4.2 所示。原先屬於紅色 cluster 的其中一個 node 距離藍色的 cluster 比藍色 cluster 中最遠的 node 還要近，在經由 social circle integration 的 process 後該 node 也將從屬於藍色的 cluster。透過這樣的演算法，我們就能根

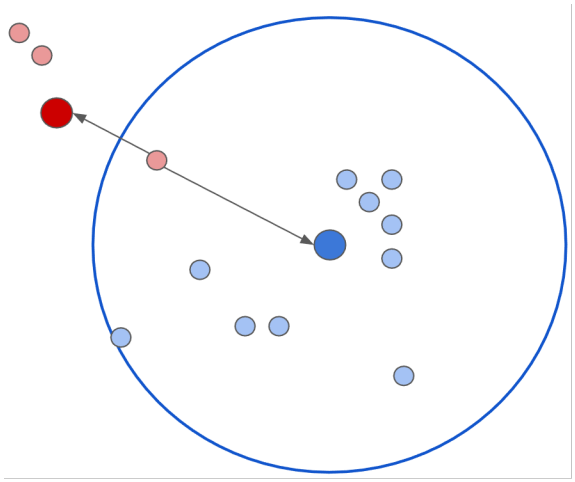

Figure 4.2: An illustration of the process of social circle integration in DPSCD.

據 node 之間的距離製造出 circle 之間合理的 overlap，這邊也可以看出，distance function 對於 DPSCD 的重要性，不只會影響前面 DPC 的結果，同時也會影響 social circle integration 的好壞。

4.3 Issue and Improvement

在最一開始，我們先嘗試自行從頭實作出 DPSCD 原始論文裡面的作法。而在我們實作完成後，我們發現原始論文中有一些潛在的問題以及可能可以改進的地方。首先是我們認為原始論文中提出的 distance function 實作上會有一些疑慮，效果可能也還可以更好；再者，我們認為最後作完 circle integration 後，應該還有一些提升 performance 的空間。對此，我們將分為兩個小節來分別探討這兩個部分。

4.3.1 Improving distance function (ID)

實作上，我們發現很多 nodes 之間的 similarity 是極低或甚至是零的。根據這樣個情況對應到原始論文的 distance function (Equation 4.2) 就會發現 distance 會非常大或甚至會出現 ZeroDivisionError (請見 Figure 4.3-A)。這樣的結果，distance 幾乎都特別大的結果就會造成 DPC 這樣的方法無法找出合適的 centers。在我們的觀察中，使用原始論文所分離出的的中心通常數量都只有一個 (Figure 4.3-B)，使得找到的 social circles 數量極度受限，表現也不佳。因此，我們提出了一個新的

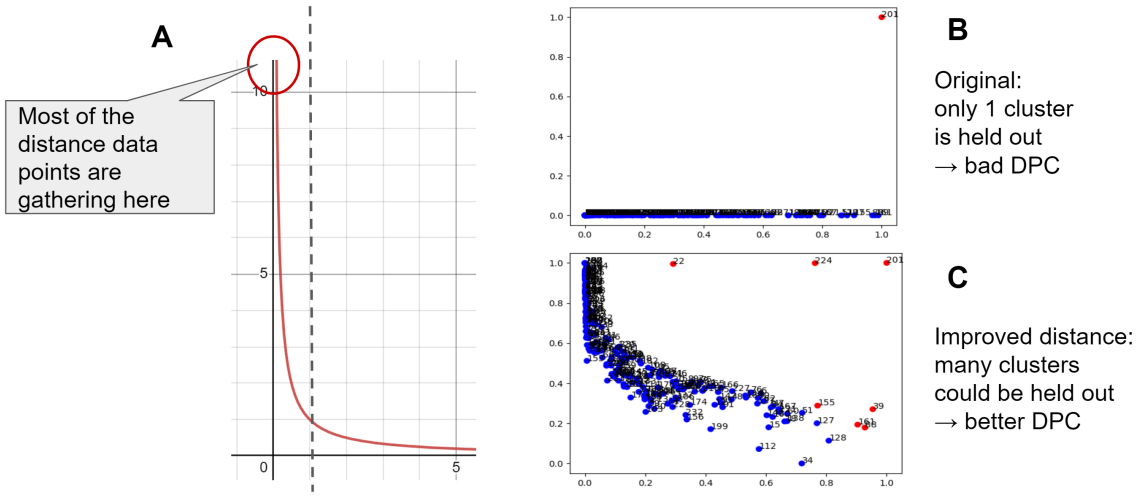

Figure 4.3: Issue of the original distance function (A, B) and the illustration of our improved results (C).

distance function (Equation 4.4)。

$$d_{i,v} = \sqrt{1 - (\alpha \cdot \text{simP}(i,v) + (1-\alpha) \cdot \text{simN}(i,v))^2} \tag{4.4}$$

在我們的 improved distance function (ID) 中，我們依然有運用到 profile similarity 以及 topological similarity。且利用 $cos(\theta)$ 與 $sin(\theta)$ 互為大小關係，我們可以經過簡單的轉換，使得新的 distance function 依然在 similarity 大的時候有較小的 distance；similarity 小的時候有較大的 distance，且沒有 ZeroDivisionError。我們使用 ID 能夠得到更好的 DPC 結果 (Figure 4.3)，能夠分離出更多的 cluster center，利於後續的 social circle discovery。

4.3.2 Merging Circles after Integration (MC)

最後，我們發現 integrate 完的 circles 有兩個主要的問題: 1. 存在一些多餘的 (空的) circles；2. 存在重度重合的 circles。對此，我們提出了第二個 improvement 的方式，也就是 merging circles (MC)。我們將我們 merge circle 的過程分為三個步驟。首先，我們先經過簡單的計算，移除掉空的 cluster；再來，我們會對 circles 根據大小進行排序；最後再依序 iterate 過每個 circle pairs，假設重合程度超過八成，我們將把小的 circle merge 進大的 circle。經過這樣的一到額外的程序，我們發現的確 performance 能夠有一些進步。

Table 4.1: The performance results of DPSCD and our improved versions. ID indicates improved distance while MC stands for merging circles

Method	Loss	relative improve rate	running time (secs)
Original paper	16587	–	60.04
+ MC	16101	3.48%	58.64
+ ID	14451	12.86%	60.05
+ ID + MC	**13554**	**18.29%**	59.03

4.4 Results

最後，我們把各個方法所得到的結果整理進同一個表格 (Table 4.1)。對照表格中的數據我們可以發現使用 ID 或是加上 MC 對於原始論文的方法都有幫助，而且這兩種方法還可以同時使用並且達到最好的效果。更進一步的，由於 time complexity 主要是發生在計算 distance 以及 density 的部分，因此使用我們提出的改進的方法並不會造成總體 running time 的提升，也就是我們不僅得到了更好的結果，還可以維持原先 DPSCD 速度上的優勢 (平均跑一個 Ego-network 只要花費一秒鐘的時間)。

此外，我們也將所有我們有做過的方法整理畫成圖 (Figure 4.4)。可以看到使用了 graph clustering-based 的方法可以得到不錯的效果；而在 DPSCD 相關方法之中，原先的效果並沒有很好，是所有方法中最差的，但在經過我們 proposed 的兩種 improve 的方法後，可以大幅降低 total loss (+ID +MC)。

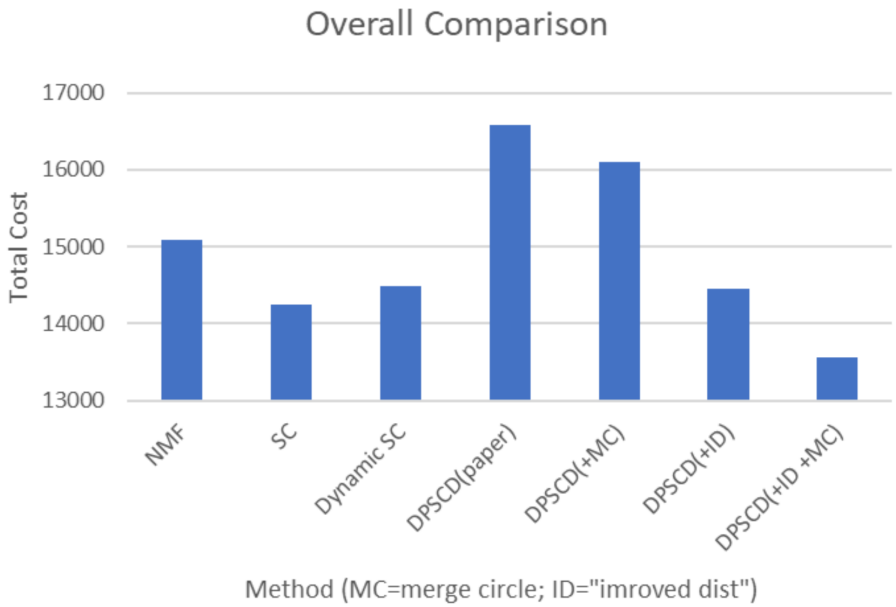

Figure 4.4: Compare the results from all the different methods, including the graph clustering-based methods.

第五章　Conclusion

5.1　Conclusion

最後，我們在此總結我們所作的資料科學期末專案。我們的題目是「非監督式學習自關係網路中的社交圈」。對此，我們首先探討了三種不同的 graph clustering-based 的方法，包括:Spectral Clustering (SC)、Dynamic SC 以及 Non-egative Matrix Factorization (NMF)。我們試著分析並探討這些方法，並取得了合理且不錯的結果。再來，我們針對 Density Peaks-based Social Circle Discovery (DPSCD) 的方法從頭開始實作。這項方法他以快速 (僅需一個 iteration 就可以初步 cluster 完) 以及可以同時利用到使用者資料和網路的拓樸資訊為主要特色。最後，我們也針對了這項方法進行了分析，同時提出了兩個改進的面向，且都取得更好的成果，甚至可以同時使用我們所 proposed 的這兩項方法來得到最佳的結果。

參考文獻

[1] bmkessler. Kaggle_social_circles.

[2] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In kdd, volume 96, pages 226–231, 1996.

[3] D. Huang. 譜分群 (spectral clustering)─ 運用圖論 (graph theory) 進行分群.

[4] A. LaViers, A. R. Rahmani, and M. B. Egerstedt. Dynamic spectral clustering. Georgia Institute of Technology, 2010.

[5] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. Nature, 401(6755):788–791, 1999.

[6] J. Leskovec and J. Mcauley. Learning to discover social circles in ego networks. Advances in neural information processing systems, 25, 2012.

[7] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. Physical review E, 69(2):026113, 2004.

[8] A. Rodriguez and A. Laio. Clustering by fast search and find of density peaks. science, 344(6191):1492–1496, 2014.

[9] scikit-learn developers (BSD License). Non-negative matrix factorization (nmf or nnmf).

[10] scikit-learn developers (BSD License). sklearn.cluster.spectralclustering.

[11] D. Seung and L. Lee. Algorithms for non-negative matrix factorization. Advances in neural information processing systems, 13:556–562, 2001.

[12] U. Von Luxburg. A tutorial on spectral clustering. Statistics and computing, 17(4):395–416, 2007.

[13] M. Wang, W. Zuo, and Y. Wang. An improved density peaks-based clustering method for social circle discovery in social networks. Neurocomputing, 179:219–227, 2016.

[14] Z. Zhang, X. Liu, and L. Wang. Spectral clustering algorithm based on improved gaussian kernel function and beetle antennae search with damping factor. Computational Intelligence and Neuroscience, 2020, 2020.