Assignment #2 - Community Detection

a. 說明如何執行程式(並附上程式碼檔案)

將程式碼與 "train.csv", 'test.csv'置於同資料夾

python louvain.py

b. 簡介你所使用的程式架構及演算法流程(如果有進行前處理也請解釋原因)

Louvain 是一種基於多層次優化 modularity 的演算法。modularity 在圖論里面用於衡量社群品質, 具體定義如下:

$$Q = \frac{1}{2m}\sum_{i,j}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j)$$

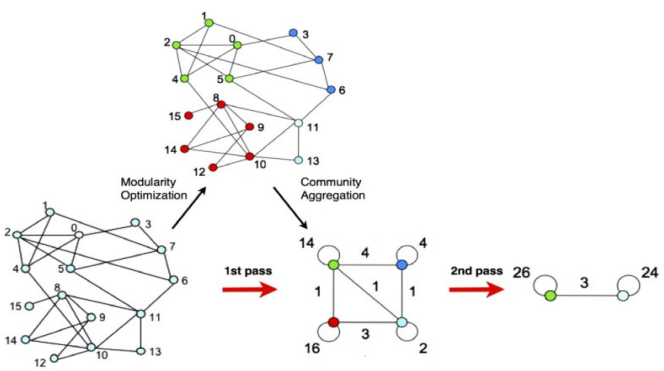
其中， m 表示網絡中邊的數量， K 為節點度數, A 為鄰接矩陣, delta 函數若(i,j)在一個社群則為 1, 否則為 0。 算法過程為:

1. 剛開始為每個節點定義 1 個社群, 然後移動節點到鄰居節點社群, 並通過最大 delta modularity 改變量來決定節點合並到哪個鄰居社群中。 delta modularity 定義如下:

$$\Delta Q = \frac{k_{i,in}}{2m} - \gamma\frac{\Sigma_{tot}\cdot k_i}{2m^2}$$

式中 ki,in 是節點 i 與 i 要移入社群中所有節點之間邊權重和, sigma tot 是節點 i 要移入的社群內所有節點的邊的權重和。 上式可以理解為括號內第一項 ki,in 表示實際節點 i 與要移入社群之間的連接邊數, 第二項 sigma tot/m 為其它節點與該社群連接一條邊的概率, sigma tot/m * ki 則為基於平均情況下節點 i 在度為 k 的情況下與該社群連接的邊數。 第一項若比第二項大則說明節點 i 與該社區連接程度超過平均預測, 則加入到該社群, 反之保持不變。

2. 在完成第一輪迭代後每個節點劃分為特定社群中。 接著把整個社群當成一個大節點, 大節點相連構成新的圖。 各大節點之間的連接以及權重通過社群之間連接來決定。 然後對新圖重覆 1 中過程做下一層次的社群劃分。 如下圖所示:



c.結果分析

之前使用過以下演算法:

girvan newman,leiden,louvain,surprise communities,walktrap。

其中 girvan newman 因為次迭代都要計算 edge betweenness 導致時間複雜度過高,無法使用在本次作業之資料集。其他演算法中又以 louvain 演算法結果最佳。我以 louvain 演算法嘗試各種參數,最後調出最佳參數為 resolution=1, threshold=0.0001, seed=7,最後 kaggle public score 為 0.77。