**Spectral**

L’algoritmo di clustering spettrale permette di trovare due cluster in base al segno dell’autovettore associati all’autovalore minore, ma dovendo associare i nodi del grafo ad almeno quattro gruppi si hanno due possibili modi di operare: prendere in considerazioni gli autovettori associati ai due autovalori minori o effettuare una seconda volta il calcolo sui primi due cluster ottenuti.

Dalle prime osservazioni, si può subito concludere che il calcolo degli autovalori e autovalori su matrici sparse di tali dimensioni risulta sia temporalmente che spazialmente molto oneroso, decidendo quindi di utilizzare i primi due autovettori per ridurre i tempi di esecuzione.  
L’algoritmo naive ha mostrato, così come gli altri algoritmi di clustering, il problema dello sbilanciamento tra cluster, indice di una non netta separazione in gruppi dei nodi e una altissima probabilità che in tale grafo sia presente una componente gigante che contiene quasi tutti i nodi. Inoltre, è risultato essere estremamente lento a causa del troppo oneroso calcolo degli autovettori con un tempo di esecuzione di circa 13h e 30min. I risultati sono qui riportati.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Cluster1 | Cluster2 | Cluster3 | Cluster4 |
| Politician | 0 | 1 | 0 | 5767 |
| Government | 0 | 0 | 0 | 6880 |
| TVshow | 49 | 6 | 212 | 3060 |
| Company | 11 | 115 | 126 | 6243 |

Possiamo subito notare che sia i gruppi relativi ai ‘Politician’ che ai ‘Government’ risultano essere completamente contenuti nel cluster4 (ad eccezione di un elemento), dovendo quindi associare ai primi un cluster in cui non sono presenti loro elementi per ottenere l’accuratezza maggiore possibile. In particolare, l’associazione migliore possibile tra cluster e gruppi permette di ottenere un numero di nodi assegnati correttamente pari a 7207 su 22470

Per prima cosa, si cerca di ridurre il più possibile il tempo di esecuzione. Non potendo calcolare gli autovalori in maniera parallela, si effettua un leggero sampling dei nodi escludendo dal grafo e dal calcolo degli autovalori tutti i nodi di grado 1. Tale decisione è stata presa in base a due fattori principali: rimuovendo da un grafo connesso tutti i nodi di grado 1, si ha la certezza che il grafo rimanga connesso; questi nodi possono appartenere solo al cluster in cui è presente il loro unico vicino, potendo quindi ugualmente assegnare loro un cluster al termine dell’algoritmo.  
Pur essendo un sampling di piccolissime dimensioni, ciò ci permette di ridurre i nodi di poco più di un decimo, da 22470 a 19812, e, considerando la complessità computazionale pari ad O(n3), otteniamo O((9\*n/10)3)=O(729\*n3/1000). Ci aspettiamo quindi di ridurre i tempi di esecuzione di almeno un quinto.

Possiamo infatti vedere come l’algoritmo, con questo campionamento, impiega 10h per essere eseguito completamente. Bisogna quindi effettuare altri miglioramenti per ottenere tempi accettabili.  
Di seguito sono riportati i risultati dell’algoritmo con campionamento dei nodi.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Cluster1 | Cluster2 | Cluster3 | Cluster4 |
| Politician | 1 | 0 | 5767 | 0 |
| Government | 0 | 0 | 6866 | 14 |
| TVshow | 86 | 0 | 3241 | 0 |
| Company | 140 | 40 | 6282 | 33 |

Dai risultati però possiamo subito notare come in questa occasione, oltre ai ‘Politician’, anche i ‘TVshow’ risultano dover essere associati a cluster che non contengo loro elementi per massimizzare l’accuratezza. In particolare, abbiamo ottenuto 7006 nodi associati correttamente su 22470.

Per ridurre ulteriormente i tempi, l’unica soluzione possibile risulta essere il calcolo dei soli autovettori di cui abbiamo bisogno, invece degli n-1 che otteniamo dall’algoritmo della libreria scipy. Purtroppo, però, riducendo il numero di autovalori e autovettori da ottenere dalla funzione eigsh attraverso il relativo parametro e impostando di voler ricavare quelli con modulo più piccolo, otteniamo valori che si discostano troppo da quelli reali. Infatti, possiamo notare come, impostando k=2, il secondo autovettore risulta essere composto da soli elementi negativi, ottenendo due cluster vuoti. Di conseguenza questa soluzione viene scartata nonostante il suo brevissimo tempo di esecuzione di circa 2 minuti.

Per ridurre il tempo di esecuzione si decide quindi di utilizzare un algoritmo differente per il calcolo degli autovettori dato che l’Arnoldi method usato da scipy risulta troppo oneroso per il calcolo di n-1 autovettori e non accurato nel caso in cui si vogliano calcolare un numero inferiore di autovettori.  
Si è scelto quindi di utilizzare l’inverse power method il quale calcola un’approssimazione del solo autovettore dominante della matrice inversa attraverso k prodotti tra matrice e vettore senza la necessità di trovare il relativo autovalore. Si noti che l’autovettore dominante, cioè l’autovettore relativo all’autovalore con valore massimo, della matrice inversa corrisponde all’autovettore relativo all’autovalore minimo della matrice originaria.  
Questo metodo permette però di calcolare solo un autovettore e di conseguenza, per dividere il grafo in quattro cluster, una volta ottenuto il primo risultato si effettuano ulteriori esecuzioni dell’inverse power method. In particolare, tale metodo viene rieseguito su entrambi i cluster ottenuti se il cluster con dimensioni minori contiene almeno un decimo degli elementi totali, altrimenti viene rieseguito due volte sui cluster con dimensioni maggiori. Nel caso in esame, la differenza di dimensioni tra cluster risulta essere eccessiva e quindi si applica la seconda soluzione.  
Anche in questo metodo però risulta esserci un problematico collo di bottiglia rappresentato dal calcolo dell’inversa. Infatti, oltre a richiedere tempi di esecuzione molto lunghi, nei vari esperimenti questa operazione ha portato ad una completa saturazione della ram dopo un’ora di esecuzione e, data la necessità di effettuare questo calcolo per 3 volte nel corso dell’algoritmo, ciò risulta non accettabile.  
Si è quindi ricorsi ad ulteriori approssimazioni per il calcolo dell’inversa. Il metodo che è risultato più adatto per questo scopo è stato la LU-decomposition che ci permette di trovare un’approssimazione dell’inversa in tempi molto più brevi.

Una volta arrivati a questo punto, sono stati effettuati vari esperimenti al fine di scegliere il numero di cicli di aggiornamento ‘*k*’ dell’inverse power method. Si è così giunti alla conclusione che dopo un numero di cicli pari al numero dei nodi nel grafo, i cluster ottenuti da questo metodo non subiscono più modifiche all’aumentare dei cicli. Si è quindi posto k = n.

Questa versione dell’algoritmo spettrale è risultata essere molto più veloce con un tempo di esecuzione di circa 15 minuti. Qui di seguito sono riportati i relativi risultati.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Cluster1 | Cluster2 | Cluster3 | Cluster4 |
| Politician | 0 | 0 | 5731 | 37 |
| Government | 1 | 2 | 6836 | 41 |
| TVshow | 0 | 0 | 3304 | 23 |
| Company | 2 | 5 | 6463 | 25 |

Come i precedenti risultati, anche in questo caso abbiamo che un cluster contiene la quasi totalità dei nodi con un ulteriore riduzione della dimensione degli altri cluster e i nodi associati correttamente ai cluster sono scesi a 6878 su 22470.

Nonostante gli ottimi tempi di esecuzione, è stato effettuato anche un esperimento relativo alla versione sampled di questo algoritmo. In questo caso però i tempi di esecuzioni non si sono discostati di molto, con un tempo di esecuzione ridotto di soli 30 secondi, mentre in termini di accuratezza si sono ottenuti cluster ancora più sbilanciati in dimensioni rispetto a quest’ultima versione.