ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

SCUOLA DI SCIENZE

Corso di Laurea in Matematica

Studio e Confronto di Tre Algoritmi di Spectral Clustering con Vincolo di Group Fairness

Tesi di Laurea in Matematica Computazionale

Relatrice:
Chiar.ma Prof.ssa
Valeria Simoncini

Presentata da:
Giovanni Lombardi

Anno Accademico 2022/2023

Introduzione

Il machine learning (ML) è una branca dell'intelligenza artificiale che utilizza algoritmi in grado di effettuare previsioni e decisioni a partire da un insieme di dati in ingresso, senza seguire istruzioni esplicite per farlo. Il ML ha ad oggi molteplici applicazioni: ad esempio, è utilizzato da molti motori di ricerca per interpretare le richieste degli utenti al fine di fornire risultati pertinenti; permette la realizzazione di pubblicità mirate nel targeted advertising; è alla base dei sistemi di guida autonoma. Per quanto potente, il ML è vulnerabile a pregiudizi contro certi gruppi demografici o individui, che possono essere presenti nei dati o incorporati negli stessi algoritmi. Ad esempio, nell'ottobre del 2018 l'agenzia di notizie finanziarie internazionali Reuters ha annunciato la rinuncia di Amazon al proprio strumento di intelligenza artificiale per l'analisi automatizzata dei CVs: l'algoritmo, basandosi sulle informazioni relative alle assunzioni di Amazon su un periodo di dieci anni, aveva ereditato dai dati un comportamento parziale a favore dei canditati maschi[1]. Nonostante ciò, i sistemi di intelligenza artificiale vengono utilizzati sempre di più, in vari ambiti, per prendere decisioni che possono avere un impatto significativo sulla vita delle persone coinvolte. Per questo è cruciale assicurarsi che queste decisioni non riflettano comportamenti discriminatori. Esistono numerosi studi che approfondiscono il problema, cercando di sviluppare strategie che promuovono l'equità (*fairness*) negli algoritmi di ML. Essi variano a seconda degli algoritmi analizzati e della definizione di fairness considerata.

Questo lavoro si concentra sullo *spectral clustering* con vincolo di *group fairness*. In generale, un algoritmo di clustering è un algoritmo di ML volto al riconoscimento e raggruppamento (*clustering*) di osservazioni omogenee in un insieme di dati. Nel caso dello spectral clustering, tale raggruppamento è determinato tramite lo studio spettrale di un'opportuna matrice che codifica le informazioni

[1] `https://www.reuters.com/article/us-amazon-com-jobs-automation-insightst-women-idUSKCN1MK08G`

relative alla similarità fra i dati. Il vincolo di group fairness assicura che in ogni insieme del raggruppamento (*cluster*) la percentuale di elementi con una determinata caratteristica di interesse, detta *attributo sensibile*, sia sempre la stessa. Intuitivamente, questa restrizione vuole garantire che il criterio con cui i dati sono raggruppati sia "indifferente" a tale caratteristica, evitando che alcune osservazioni ricevano una considerazione particolare a causa di un attributo sensibile. In effetti, da un punto di vista probabilistico, il vincolo appena descritto può essere interpretato nel seguente modo: si richiede che la probabilità che una certa osservazione sia assegnata ad un certo cluster (ricevendo così un giudizio "positivo" o "negativo") sia indipendente dagli attributi sensibili considerati. In questo senso il vincolo di group fairness può essere interpretato come un vincolo di "equità". Ad esempio, nel caso in cui il dataset rappresenti un insieme di persone, attributi sensibili potrebbero essere il sesso, l'etnia, l'orientamento sessuale o l'età.

Lo scopo di questo elaborato di tesi è quello di illustrare nel dettaglio e motivare matematicamente tre diversi algoritmi di spectral clustering che implementano il vincolo di equità descritto, nonché confrontarne le performance su alcuni datasets reali e sintetici. Nel confronto, si pone l'attenzione a quanto i raggruppamenti determinati dai tre algoritmi riflettano effettivamente un'eventuale similarità o dissimilarità fra i dati, a quanto essi siano equi rispetto agli attributi sensibili considerati e al tempo richiesto per ottenerli.

Il problema dell'implementazione di vincoli di equità in algoritmi di clustering ha iniziato a ricevere una maggiore attenzione all'interno della comunità del ML a partire da [Chierichetti et al., 2018]. In tale studio è stata proposta e formalizzata la nozione di equità di un clustering descritta sopra nel caso di un unico attributo sensibile che può assumere solo due stati diversi. Sono state poi sviluppate delle varianti degli algoritmi di k-center e k-median che tengono conto di tale nozione. Successivamente, in [Kleindessner et al., 2019] si è generalizzata la nozione di equità presentata in [Chierichetti et al., 2018] al caso di più attributi sensibili che possono assumere più di due stati. In questo elaborato di tesi chiamiamo "vincolo di group fairness" il vincolo per cui il raggruppamento considerato sia equo secondo tale nozione. Gli autori dell'articolo hanno poi cercato di implementare il vincolo in alcuni algoritmi di spectral clustering. A seguire, in [Wang et al., 2023] è stata proposta una versione efficiente dell'algoritmo di spectral clustering normalizzato con vincolo di group fairness sviluppato in [Kleindessner et al., 2019]. Per una trattazione molto più completa sulla storia

dello studio dell'equità nel contesto del clustering si consulti [Chhabra et al., 2021].

Questo lavoro di tesi si basa sugli articoli già citati [Kleindessner et al., 2019] e [Wang et al., 2023]. Nel capitolo 1 si descrive l'algorimo di spectral clustering normalizzato secondo [Shi and Malik, 2000], in breve SC. Il capitolo 2 è invece focalizzato sullo spectral clutering con vincolo di equità. Innanzitutto, si descrive nel dettaglio il vincolo di group fairness e si illustra la variante di SC presentata in [Kleindessner et al., 2019] che lo implementa; la chiamiamo FairSC. L'algoritmo ottiene il clustering equo dei dati risolvendo un opportuno problema di minimizzazione di traccia, riconducendolo ad un problema agli autovalori. Per quanto capace di realizzare raggruppamenti più equi rispetto al precedente algoritmo, FairSC può essere utilizzato soltanto su datasets di dimensioni moderate a causa di un elevato costo computazionale, in termini di tempo, legato al calcolo di diverse operazioni con matrici dense e di grandi dimensioni. A seguire, si studia una variante di FairSC che si basa sulla riformulazione del problema di minimizzazione in un opportuno problema generalizzato agli autovalori. Il nuovo algoritmo prende il nome di gep-FairSC. Sempre nel capitolo 2 è illustrato il modello proposto in [Wang et al., 2023], che risolve il problema di complessità computazionale di FairSC riuscendo a ricondurre il problema di minimizzazione di traccia ad un problema agli autovalori risolvibile in modo efficiente. L'algoritmo così sviluppato è chiamato Scalable FairSC o s-FairSC. Il nome dell'algoritmo deriva dal fatto che esso risulta *scalabile*, ovvero capace di gestire anche datasets di grandi dimensioni, nel senso che risulta avere un tempo di esecuzione paragonabile a quello del corrispondente algoritmo senza vincolo di equità, SC. Infine, nel capitolo 3 si illustrano alcuni esperimenti con cui si verifica che FairSC, gep-FairSC e s-FairSC recuperano sempre clusterings più equi rispetto a SC. Inoltre, si mostra che, nonostante le differenze in termini di costo computazionale, i raggruppamenti determinati dai tre algoritmi risultano del tutto analoghi sia per quanto riguarda l'omogeneità dei clusters che per quanto riguarda l'equità.

Indice

Preliminari

Illustriamo qualche utile preliminare di teoria dei grafi. Richiamiamo innanzitutto la definizione di grafo semplice e pesato e la sua rappresentazione tramite matrice di adiacenza. Definiamo poi alcune matrici Laplaciane sui grafi e concludiamo dimostrandone qualche importante proprietà.

Nella sua definizione più generale, un *grafo* è il dato di una collezione di oggetti, detti *vertici*, che interagiscano a coppie tramite dei collegamenti, detti *lati*. Si chiama *cappio* un lato che connette un vertice a sé stesso. Un grafo con al più un lato fra ogni coppia di vertici e che non ammette cappi è detto *semplice*. I grafi che tratteremo saranno spesso semplici e *pesati*: a ogni lato sarà associato un numero reale positivo che rappresenta la "forza" o l'"intensità" della connessione (l'interpretazione di tali termini può variare a seconda del contesto).

Un modo molto diffuso di rappresentare matematicamente un grafo consiste nell'utilizzo della *matrice di adiacenza*. Consideriamo un grafo semplice e pesato di vertici $V = \{v_1, \dots, v_n\}$. Denotiamo con $w_{ij} > 0$ il peso associato al lato che collega v_i a v_j se tale lato esiste; poniamo $w_{ij} = 0$ altrimenti. Allora, la matrice di adiacenza del grafo è definita come $W := (w_{ij}) \in \mathbb{R}^{n \times n}$. Si osservino le seguenti:

- Per ogni $i = 1, \dots, n$, si ha $w_{ii} = 0$ perché il grafo non ammette cappi;
- La matrice di adiacenza è simmetrica perché se v_i è collegato a v_j da un lato di peso w_{ij} vale anche il viceversa (ai lati non è associata una "direzione").

Usiamo la notazione $\mathcal{G}(V, W)$ per indicare il grafo semplice e pesato di vertici V e con matrice di adiacenza W.

Definiamo il *grado* del nodo $v_i \in V$ come

$$d_i := \sum_{j=1}^{n} w_{ij},$$

cioè il peso totale dei lati che hanno un estremo in v_i. Definiamo inoltre la *matrice dei gradi* come

$$D \coloneqq \operatorname{diag}(d_1, \dots, d_n) = \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{bmatrix}.$$

Nel seguito supporremo sempre che il grafo non contenga nodi isolati, cioè nodi non collegati a nessun altro vertice, in modo tale che D risulti definita positiva.

Definizione. La *matrice Laplaciana* del grafo semplice e pesato $\mathcal{G}(V, W)$ è

$$L = D - W;$$

la *matrice Laplaciana normalizzata simmetrica* è

$$L_n = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I_n - D^{-\frac{1}{2}} W D^{-\frac{1}{2}};$$

la *matrice Laplaciana normalizzata non simmetrica* (o *random walk*) è

$$L_{rw} = D^{-1} L = I_n - D^{-1} W.$$

Dimostriamo alcune basilari proprietà di queste tre matrici, che si possono ritrovare con ulteriori dettagli nell'articolo [von Luxburg, 2007].

Proposizione. *La matrice Laplaciana L soddisfa le seguenti proprietà:*

1. *per ogni $x \in \mathbb{R}^n$ si ha*
$$x^T L x = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} (x_i - x_j)^2; \tag{1}$$
2. *L è simmetrica semidefinita positiva;*
3. *il vettore con coefficienti unitari $\mathbf{1}_n \in \mathbb{R}^n$ è autovettore di L relativo all'autovalore nullo.*

Dimostrazione. Dimostramo innanzitutto il punto 1. Si ha

$$\begin{aligned} 2x^T L x &= 2 \sum_{i=1}^{n} d_i x_i^2 - 2 \sum_{i,j=1}^{n} w_{ij} x_i x_j = 2 \sum_{i,j=1}^{n} w_{ij} x_i^2 - 2 \sum_{i,j=1}^{n} w_{ij} x_i x_j \\ &= \sum_{i,j=1}^{n} w_{ij} x_i^2 - 2 \sum_{i,j=1}^{n} w_{ij} x_i x_j + \sum_{i,j=1}^{n} w_{ij} x_j^2 = \sum_{i,j=1}^{n} w_{ij} (x_i - x_j)^2, \end{aligned}$$

da cui il risultato. Segue immediatamente che $x^T L x \geq 0$ per ogni $x \in \mathbb{R}^n$, che prova il punto 2. Per concludere, si osservi che

$$L \mathbf{1}_n = D \mathbf{1}_n - W \mathbf{1}_n = 0,$$

perché $(D \mathbf{1}_n)_i = d_i = (W \mathbf{1}_n)_i$ per ogni $i = 1, \dots, n$. □

Ricordiamo che un *cammino* nel grafo $\mathcal{G}(V, W)$ è una sequenza di nodi tale che ogni coppia di nodi consecutivi nella sequenza sia collegata da un lato. Un insieme di vertici $A \subseteq V$ si dice *connesso* se ogni coppia di vertici di A è collegata da un cammino costituito soltanto da vertici di A. Una *componente connessa* del grafo è un insieme di vertici connesso massimale.

Proposizione. *La molteplicità k dell'autovalore* 0 *della matrice L è uguale al numero di componenti connesse $A_1, \dots, A_k$ del grafo. L'autospazio corrispondente è generato dai vettori indicatori delle componenti connesse $\mathbf{1}_{A_1}, \dots, \mathbf{1}_{A_k}$.*

Dimostrazione. Consideriamo dapprima il caso in cui il grafo è connesso. Sia $x \in \mathbb{R}^n$ tale che $Lx = 0$. Allora,

$$x^T L x = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(x_i - x_j)^2 = 0,$$

da cui si deduce che $x_i = x_j$ se i nodi v_i e v_j sono collegati da un lato (cioè $w_{ij} > 0$). Segue che $x_i = x_j$ se i nodi v_i e v_j sono connessi da un cammino. Poichè il grafo è connesso, deve essere $x \in \text{span}(\mathbf{1}_n)$. D'altra parte, per la proposizione precedente, $\mathbf{1}_n$ è autovettore di L relativo a 0. Perciò 0 è autovalore semplice di L e l'autospazio corrispondente è $\text{span}(\mathbf{1}_n)$.

Consideriamo ora il caso generale in cui il grafo ha k componenti connesse $A_1, \dots, A_k$ di cardinalità $n_1, \dots, n_k$ rispettivamente. A meno di rinumerare i nodi, possiamo supporre che

$$\begin{aligned} A_1 &= \{v_1, \dots, v_{n_1}\}, \\ A_2 &= \{v_{n_1+1}, \dots, v_{n_1+n_2}\}, \\ &\vdots \end{aligned}$$

La Laplaciana L risulterà quindi diagonale a blocchi,

$$L = \begin{bmatrix} L_1 & & \\ & \ddots & \\ & & L_k \end{bmatrix},$$

dove $L_i \in \mathbb{R}^{n_i \times n_i}$ è la matrice Laplaciana del sottografo con nodi A_i. Poiché tale sottografo è connesso per definizione di A_i, utilizzando la prima parte della dimostrazione si ottiene che 0 è autovalore semplice di L_i con autovettore $\mathbf{1}_{n_i}$. Segue la tesi ricordando la relazione tra gli autovalori e autovettori di una matrice diagonale a blocchi e quelli dei blocchi stessi. □

Concludiamo con analoghe proprietà delle matrici Laplaciane normalizzate. Le due proposizioni che seguono si possono dimostrare con ragionamenti analoghi a quelli già utilizzati.

Proposizione. *Per le matrici Laplaciane normalizzate L_n e L_{rw} valgono le seguenti:*

1. *per ogni $x \in \mathbb{R}^n$ si ha*

$$x^T L_n x = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij} \left(\frac{x_i}{\sqrt{d_i}} - \frac{x_i}{\sqrt{d_j}} \right)^2;$$

2. *Le autocoppie di L_{rw} sono le soluzioni del problema generalizzato agli autovalori $Lx = \lambda Dx$;*

3. *(λ, x) è un'autocoppia di L_{rw} se e soltanto se $(\lambda, D^{\frac{1}{2}} x)$ è un'autocoppia di L_n;*

4. *$\mathbf{1}_n$ è autovettore di L_{rw} relativo all' autovalore 0 e $D^{\frac{1}{2}} \mathbf{1}_n$ è autovettore si L_n relativo all'autovalore 0;*

5. *L_n è simmetrica semidefinita positiva e L_{rw} ha autovalori reali non negativi.*

Proposizione. *Per entrambe le matrici Laplaciane normalizzate L_n e L_{rw}, la molteplicità k dell'autovalore 0 è pari al numero di componenti connesse $A_1, \ldots, A_k$ del grafo. Inoltre, nel caso di L_{rw} il corrispondente autospazio è generato dai vettori indicatori $\mathbf{1}_{A_1}, \ldots, \mathbf{1}_{A_k}$; nel caso di L_n tale autospazio è generato dai vettori $D^{\frac{1}{2}} \mathbf{1}_{A_1}, \ldots, D^{\frac{1}{2}} \mathbf{1}_{A_k}$.*

Capitolo 1

Spectral clustering

Assegnato un insieme di dati e una qualche nozione di similarità tra di essi, lo scopo di un algoritmo di clustering è quello di determinarne un raggruppamento in insiemi omogenei. In altre parole, l'algoritmo cerca una partizione dell'insieme dei dati tale per cui osservazioni nello stesso sottoinsieme siano più simili tra loro rispetto a dati in sottoinsiemi distinti. Fra tutti gli algoritmi di clustering, gli algoritmi di spectral clustering presentano alcuni vantaggi: spesso riescono a ottenere risultati migliori rispetto agli algoritmi di clustering "tradizionali" come quelli di linkage o delle k-medie, come mostrato con un esempio nella sezione 1.2 a pagina 11, e possono essere realizzati con semplici strumenti di algebra lineare.

All'interno di questo primo capitolo si vuole sviluppare l'algoritmo di spectral clustering normalizzato proposto in [Shi and Malik, 2000], che costituisce la base degli algoritmi di fair spectral clustering presentati successivamente.

Dato $k \in \mathbb{N}$, in questo elaborato scriveremo sempre "i primi k autovalori" per riferirci ai k autovalori più piccoli della matrice studiata (assicurandoci che la matrice abbia autovalori reali).

1.1 Costruzione dell'algoritmo SC

Consideriamo un grafo semplice e pesato $\mathcal{G}(V, W)$ i cui nodi

$$V = \{v_1, \dots, v_n\},$$

rappresentano i dati a disposizione e la cui matrice di adiacenza

$$W = (w_{ij}) \in \mathbb{R}^{n \times n},$$

contiene le informazioni relative alla similarità fra i dati. Assumiamo $w_{ij} \geq 0$ e $w_{ii} = 0$ per ogni i. Maggiore è w_{ij}, più i nodi v_i e v_j saranno "simili", dove la

nozione di similarità non è univoca e può dipendere dalla natura dei dati e dal contesto di applicazione. Ad esempio, nel caso in cui i dati siano punti in $\mathbb{R}^m$, tipicamente sono considerate "simili" osservazioni vicine rispetto alla metrica euclidea. La coppia (v_i, v_j) è un lato del grafo se $w_{ij} > 0$. Denotiamo con

$$d_i := \sum_{j=1}^{n} w_{ij},$$

il grado del vertice v_i e con $D := \mathrm{diag}(d_1, \ldots, d_n)$ la matrice dei gradi del grafo. Assumendo che non ci siano nodi isolati, il grado di ciascun nodo è positivo, dunque la matrice D è diagonale definita positiva. L'algoritmo determina una partizione (*clustering*) dell'insieme dei vertici V in k sottoinsiemi (*clusters*)

$$V = C_1 \sqcup \cdots \sqcup C_k. \tag{1.1}$$

Tale partizione può essere rappresentata dalla matrice $H = (h_{il}) \in \mathbb{R}^{n \times k}$, detta *matrice indicatrice del clustering*, dove

$$h_{il} := \begin{cases} 1, & \text{se } v_i \in C_l, \\ 0, & \text{altrimenti,} \end{cases} \tag{1.2}$$

per $i = 1, \ldots, n$ e $l = 1, \ldots, k$.

Data una partizione (1.1) dei nodi, introduciamo le seguenti funzioni:

$$\mathrm{vol}(C_l) := \sum_{v_i \in C_l} d_i = \sum_{\substack{v_i \in C_l, \\ v_j \in V}} w_{ij}, \tag{1.3}$$

che misura il peso complessivo dei lati che hanno almeno un estremo nel cluster C_l;

$$\mathrm{Cut}(C_l, V \setminus C_l) := \sum_{\substack{v_i \in C_l, \\ v_j \in V \setminus C_l}} w_{ij}, \tag{1.4}$$

che misura il peso complessivo dei lati che hanno un solo estremo in C_l;

$$\mathrm{NCut}(C_1, \ldots, C_k) := \sum_{l=1}^{k} \frac{\mathrm{Cut}(C_l, V \setminus C_l)}{\mathrm{vol}(C_l)}, \tag{1.5}$$

detta *Normalized Cut*, che misura il peso totale normalizzato dei lati che hanno estremi in clusters distinti. Si osservi che la funzione NCut assume valori bassi in corrispondenza di raggruppamenti significativi dei dati, nel senso di clusterings in cui si ha alta similarità all'interno di ciascun cluster e bassa similarità fra

clusters diversi. Dunque, ricordando che lo scopo dell'algoritmo è quello di raggruppare dati simili fra loro, l'obiettivo dell'algoritmo SC può essere formalizzato matematicamente come quello di determinare una partizione di V in k clusters che minimizza la funzione NCut. Si osservi che normalizzare la funzione Cut in (1.5) è utile per scoraggiare la formazione di raggruppamenti con pochi nodi "isolati". Tuttavia questo non è l'unico modo di farlo. Ad esempio, un'altra funzione obiettivo molto utilizzata è ottenuta considerando la "dimensione" del cluster C_l come il numero di vertici da esso contenuti, $|C_l|$, anzichè $\mathrm{vol}(C_l)$. In letteratura essa prende il nome di *Ratio Cut*:

$$\mathrm{RatioCut}(C_1 \ldots, C_k) := \sum_{l=1}^{k} \frac{\mathrm{Cut}(C_l, V \setminus C_l)}{|C_l|}.$$

A seconda della scelta della funzione obiettivo cambierà la matrice di cui si svolge l'analisi spettrale per ottenere il clustering ottenendo così vari algoritmi di spectral clustering [von Luxburg, 2007]. Delle due possibilità presentate, è preferibile l'utilizzo di NCut perché la quantità $|C_l|$ non dà informazioni sulla similarità fra i dati del cluster C_l, che si vorrebbe massimizzare. La proposta di usare NCut come funzione obiettivo, seguita anche in questo lavoro, è stata avanzata per la prima volta in [Shi and Malik, 2000].

La funzione obiettivo NCut può essere espressa in modo più compatto come traccia di una matrice opportuna. Per farlo, scaliamo innanzitutto la matrice H nel modo seguente:

$$H \leftarrow H\widehat{D}^{-1}, \tag{1.6}$$

dove $\widehat{D} := \mathrm{diag}\Big(\sqrt{\mathrm{vol}(C_1)}, \ldots, \sqrt{\mathrm{vol}(C_k)}\Big)$. Affinché questo passaggio abbia senso, è fondamentale ricordare che si sta considerando un grafo privo di nodi isolati, cioè $d_i > 0$ per ogni i. Con abuso di notazione, indichiamo d'ora in poi la nuova matrice indicatrice sempre con la lettera H. Consideriamo poi $L \in \mathbb{R}^{n\times n}$ la matrice Laplaciana del grafo, $L := D - W$. Allora vale l'uguaglianza

$$\mathrm{NCut}(C_1, \ldots, C_k) = \mathrm{Tr}\big(H^T L H\big). \tag{1.7}$$

Per vederlo, poniamo $h_l := He_l$ la colonna l-esima di H per $l = 1, \ldots, k$, dove e_l denota l'l-esimo vettore della base canonica di $\mathbb{R}^n$. Osservando che l'i-esimo

coefficiente di h_l è $\frac{1}{\sqrt{\mathrm{vol}(C_l)}}$ se $v_i \in C_l$ e 0 altrimenti, si ha

$$\begin{aligned} h_l^T L h_l &\overset{(1)}{=} \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(h_{il}-h_{jl})^2 \\ &= \frac{1}{2}\sum_{\substack{v_i\in C_l,\\ v_j\in V\setminus C_l}} w_{ij}\left(\frac{1}{\sqrt{\mathrm{vol}(C_l)}}-0\right)^2 + \frac{1}{2}\sum_{\substack{v_i\in V\setminus C_l,\\ v_j\in C_l}} w_{ij}\left(0-\frac{1}{\sqrt{\mathrm{vol}(C_l)}}\right)^2 \\ &\quad \text{(ricordando che } W \text{ è simmetrica)} \\ &= \sum_{\substack{v_i\in C_l,\\ v_j\in V\setminus C_l}} \frac{w_{ij}}{\mathrm{vol}(C_l)} = \frac{\mathrm{Cut}(C_l, V\setminus C_l)}{\mathrm{vol}(C_l)}. \end{aligned}$$

per $l = 1, \ldots, k$. Pertanto,

$$\mathrm{Tr}\big(H^T L H\big) = \sum_{l=1}^{k} h_l^T L h_l = \sum_{l=1}^{k} \frac{\mathrm{Cut}(C_l, V\setminus C_l)}{\mathrm{vol}(C_l)} = \mathrm{NCut}(C_1\ \ldots, C_k)\,.$$

Dunque il problema di minimizzazione della funzione NCut è stato ricondotto al problema equivalente

$$\min_{H\in\mathbb{R}^{n\times k}} \mathrm{Tr}\big(H^T L H\big) \quad \text{t.c. } H \text{ ha la forma (1.1).} \tag{1.8}$$

Per quanto elegante sia questa nuova formulazione del problema, essa risulta nella pratica ancora poco utile a causa di un elevato costo computazionale: risolverlo direttamente richiede un costo più che polinomiale (il problema è NP-hard). Per questo se ne risolve invece una versione rilassata:

$$\min_{H\in\mathbb{R}^{n\times k}} \mathrm{Tr}\big(H^T L H\big) \quad \text{t.c. } H^T D H = I_k. \tag{1.9}$$

Si noti che, in effetti, se H è della forma (1.1) allora soddisfa l'equazione in (1.9).

Il problema (1.9) è un classico problema di minimizzazione ed è stato affrontato per la prima volta nell'articolo [Fan, 1950]. Per risolverlo, utilizziamo il seguente risultato:

Teorema 1.1.1. *Data una matrice simmetrica $M \in \mathbb{R}^{n\times n}$ e un intero positivo $k \le n$,*

$$\min_{X^T X = I_k} \mathrm{Tr}\big(X^T M X\big) = \mathrm{Tr}\big(X_*^T M X_*\big) = \sum_{i=1}^{k} \lambda_i,$$

dove $\lambda_1 \le \cdots \le \lambda_n$ sono gli autovalori di M ordinati in ordine crescente e $X_* \in \mathbb{R}^{n\times k}$ ha colonne ortonormali tali che $X_* e_l$ è autovettore relativo a λ_l per ogni $l = 1, \ldots, k$.

Dimostrazione. Se $k = n$, allora per ogni $X \in \mathbb{R}^{n\times n}$ tale che $X^TX = I_n$ vale

$$\operatorname{Tr}\big(X^TMX\big) = \sum_{i=1}^{n} \lambda_i,$$

da cui segue il risultato. Supponiamo ora $k < n$. Sia $X = [x_1, \dots, x_k] \in \mathbb{R}^{n\times k}$ con colonne ortonormali. Completiamo $\{x_1, \dots, x_k\}$ a una base ortonormale $\{x_1, \dots, x_n\}$ di $\mathbb{R}^n$. Poniamo $U \coloneqq [x_1, \dots, x_n] \in \mathbb{R}^{n\times n}$, $A \coloneqq U^TMU$ e $B \coloneqq X^TMX$. Allora

$$A = U^TMU = \begin{bmatrix} B & C^T \\ C & D \end{bmatrix},$$

per opportune matrici $C \in \mathbb{R}^{(n-k)\times k}$ e $D \in \mathbb{R}^{(n-k)\times(n-k)}$. Siano $\mu_1 \le \cdots \le \mu_k$ gli autovalori della matrice simmetrica B. Poiché la U è ortogonale, gli autovalori di A sono gli stessi della matrice M. Denotiamo $v_1, \dots, v_k$ gli autovettori di A relativi rispettivamente a $\lambda_1, \dots, \lambda_k$. Allora,

$$\lambda_i = \min_{\substack{x\in\mathbb{R}^n\setminus\{0\} \\ x\perp v_1,\dots,v_{i-1}}} \frac{x^TAx}{x^Tx} \le \min_{\substack{x\in\mathbb{R}^n\setminus\{0\} \\ x\perp v_1,\dots,v_{i-1} \\ x\perp e_{k+1},\dots,e_n}} \frac{x^TAx}{x^Tx} = \min_{\substack{y\in\mathbb{R}^k\setminus\{0\} \\ [y;0;\dots;0]\perp v_1,\dots,v_{i-1}}} \frac{y^TBy}{y^Ty} \le \mu_i,$$

per ogni $i = 1, \dots, k$, dove l'ultima disuguaglianza è vera per il teorema di min-max di Courant-Fischer. Sommando su i si ottiene

$$\sum_{i=1}^{k} \lambda_i \le \sum_{i=1}^{k} \mu_i = \operatorname{Tr}(B) = \operatorname{Tr}\big(X^TMX\big).$$

Resta da provare che tale valore è raggiunto per $X = X_*$, dove X_* è come nell'enunciato. Sia $\widetilde{X} \in \mathbb{R}^{n\times(n-k)}$ tale che le colonne di X_* assieme a quelle di $\widetilde{X}$ costituiscano una base ortonormale di autovettori di M. Pertanto,

$$M = \begin{bmatrix} X_* & \widetilde{X} \end{bmatrix} \Lambda \begin{bmatrix} X_*^T \\ \widetilde{X}^T \end{bmatrix},$$

dove possiamo supporre $\Lambda = \operatorname{diag}(\lambda_1, \dots, \lambda_n)$. Allora

$$X_*^TMX_* = X_*^T \begin{bmatrix} X_* & \widetilde{X} \end{bmatrix} \Lambda \begin{bmatrix} X_*^T \\ \widetilde{X}^T \end{bmatrix} X_* = \begin{bmatrix} I_k & 0 \end{bmatrix} \Lambda \begin{bmatrix} I_k \\ 0 \end{bmatrix} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_k \end{bmatrix},$$

Concludiamo che $\operatorname{Tr}\big(X_*^TMX_*\big) = \sum_{i=1}^{k} \lambda_i$. □

Per applicare il teorema appena dimostrato al problema (1.9), facciamo il cambiamento di variabili $X = D^{\frac{1}{2}}H$. Otteniamo

$$\min_{X \in \mathbb{R}^{n \times k}} \operatorname{Tr}\left(X^T L_n X\right) \quad \text{t.c. } X^T X = I_k, \tag{1.10}$$

dove $L_n \coloneqq D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$ è la matrice Laplaciana normalizzata simmetrica. La buona posizione di questa definizione è garantita dall'ipotesi di assenza di nodi isolati. Ricordando che W è simmetrica, è chiaro che anche L e L_n lo sono. Allora, per il teorema precedente, una qualunque matrice che ha come colonne degli autovettori ortonormali relativi ai primi k autovalori di L_n è una soluzione di (1.10). Determinata una tale soluzione X, recuperiamo

$$H = D^{-\frac{1}{2}} X. \tag{1.11}$$

Si osservi che così facendo le colonne di H sono autovettori della matrice Laplaciana normalizzata non simmetrica, $L_{rw} \coloneqq I - D^{-1}W$. La matrice H così trovata non è però della forma (1.1) a causa del rilassamento del problema. Occorre perciò ricondurre la soluzione H, con coefficienti reali, ad una matrice indicatrice del tipo (1.1), con coefficienti discreti. La procedura standard per farlo è la seguente:

1. Vediamo le n righe di H come punti in $\mathbb{R}^k$ e ne determiniamo un raggruppamento in k clusters $\widehat{C}_1, \dots, \widehat{C}_k$ utilizzando l'algoritmo di clustering delle k-medie;

2. Assegniamo il dato $v_i \in V$ al cluster C_l se l'i-esima riga di H appartiene a $\widehat{C}_l$ per $i = 1, \dots, n$ e $l = 1, \dots, k$.

Abbiamo dunque la partizione cercata, $V = C_1 \sqcup \cdots \sqcup C_k$.

L'Algoritmo 1 riassume la procedura illustrata. Esso si deve ai già citati Shi e Malik; è un algoritmo di clustering di grande successo, efficiente in quanto i moderni metodi numerici per il calcolo di autovalori e autovettori permettono di sfruttare l'eventuale sparsità della matrice L_n. Quest'ultima risulta spesso sparsa perché in molte applicazioni la matrice di adiacenza W si può costruire con pochi non-zeri (si pensi ad esempio al fatto che se i dati fossero tutti molto simili fra loro, cioè se W fosse densa, non sarebbe di particolare interesse cercarne dei sottogruppi omogenei) e L_n ha la stessa sparsità di W. In effetti, la ragione per cui il rilassamento (1.9) è molto diffuso non è il fatto che assicuri sempre soluzioni molto vicine a quelle del problema di partenza (mancano risultati teorici al riguardo); la sua popolarità si deve piuttosto al fatto che conduce a un problema di algebra lineare standard risolvibile in modo efficiente.

Algoritmo 1 SC (Spectral clustering)

Inizialization: matrice di adiacenza $W \in \mathbb{R}^{n\times n}$; matrice di grado $D \in \mathbb{R}^{n\times n}$; $k \in \mathbb{N}$

Until convergence: un raggruppamento degli indici $1 : n$ in k clusters

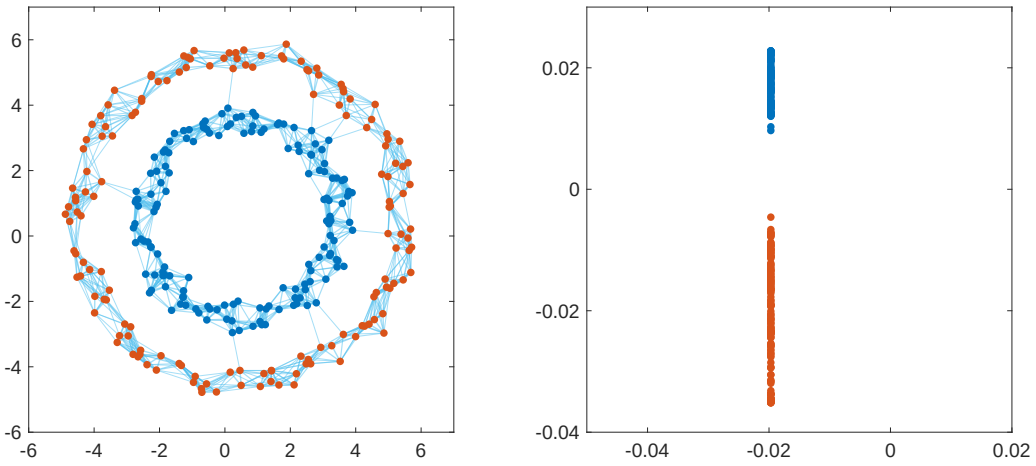
1: calcola la matrice Laplaciana $L = D - W$
2: calcola la matrice Laplaciana normalizzata $L_n = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$
3: calcola i k autovalori più piccoli di L_n e i corrispondenti autovettori $X \in \mathbb{R}^{n\times k}$
4: applica il clustering con k-medie alle righe di $H = D^{-\frac{1}{2}} X$

L'algoritmo illustrato in questa sezione è trattato nel dettaglio nell'articolo [von Luxburg, 2007].

1.2 Chiarimenti sul legame fra SC e *k*-medie

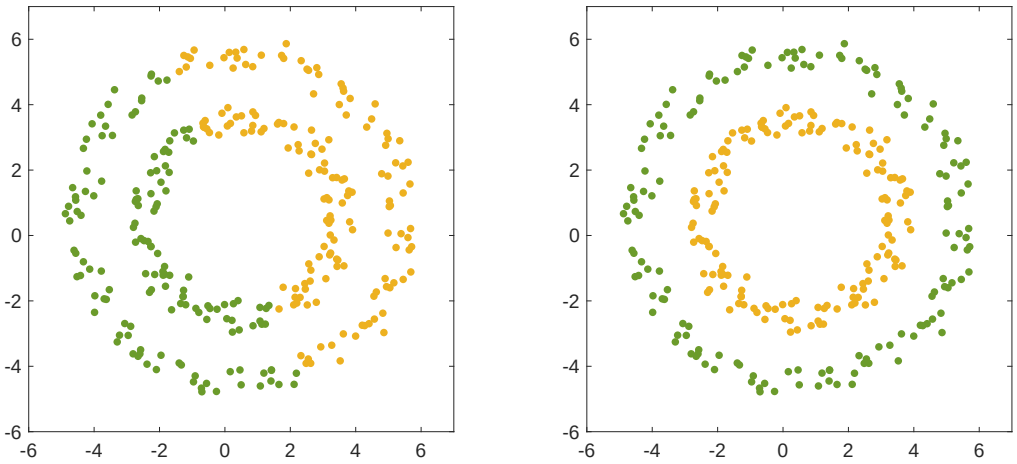
A una prima lettura dell'algoritmo, potrebbe lasciare perplessi il fatto che al punto 4 si concluda ricorrendo ad un altro algoritmo di clustering, quello delle k-medie, per ottenere il risultato finale. Potrebbe non essere immediatamente chiaro perché allora non si applichi direttamente il clustering con k-medie ai dati iniziali (nel caso in cui il dataset sia un sottoinsieme di $\mathbb{R}^m$). La motivazione sta nel fatto che tale algoritmo determina un clustering dei dati cercando di minimizzare la distanza delle osservazioni dal "centro" del cluster a cui appartengono (più preciamente, dalla media aritmetica delle osservazioni del cluster a cui appartengono). Perciò l'algoritmo fallisce nel riconoscimento dei clusters "naturali", cioè intrinsechi ai dati, quando questi ultimi non occupano regioni convesse di $\mathbb{R}^m$. Il cambiamento di variabili che trasforma i dati in punti k-dimensionali (le righe di H) serve proprio perché, anche se nelle variabili di partenza i clusters naturali potrebbero non avere questa proprietà, nelle nuove variabili essi tenderanno a soddisfarla e saranno quindi ben identificabili dall'algoritmo delle k-medie. La Figura 1.1 ne illustra un esempio. Nella Figura 1.1a è rappresentato il grafo di similarità utilizzato[1]. Notiamo che l'insieme dei dati si può pensare come l'unione di due datasets più piccoli disposti grossolanamente lungo due circonferenze concentriche, evidenziati in figura con colori diversi. Queste due famiglie costituiscono un clustering "naturale" dei dati. In Figura 1.1b sono

[1] Il grafo è stato costruito collegando v_i a v_j se v_j è uno dei 10 nodi più vicini a v_i (v_i escluso) o viceversa (*mutual* 10-*nearest neighbors graph*). Per due tali nodi è stato posto $w_{ij} = \exp(-\|v_i - v_j\|^2 / (2\sigma^2))$, con $\sigma = 1$ (*funzione di similarità Gaussiana*).



(a) Dati iniziali

(b) Dati dopo il cambio di variabili



(c) k-medie

(d) Spectral Clustering

Figura 1.1: Esempio di confronto fra k-medie e SC

rappresentati i dati dopo il cambio di variabili (righe di H)[2], avendo scelto $k = 2$. Possiamo notare che i due clusters naturali nelle nuove variabili occupano regioni convesse del piano ben separate. In Figura 1.1c e 1.1d sono illustrati con colori diversi i due clusters individuati rispettivamente dall'algoritmo delle k-medie[3] e da SC. Osserviamo che il clustering individuato da SC corrisponde a quello "naturale", mentre quello determinato dalle k-medie è poco significativo.

[2]Si osservi che i punti sono disposti su una retta verticale. Ciò non ci sorprende perché $\mathbf{1}_n$ è un autovettore di L_{rw} relativo all'autovalore 0, che ha molteplicità 1 dato che il grafo è connesso. Ne segue che la prima colonna di H, che deve essere un autovettore di L_{rw} relativo a 0, è un multiplo di $\mathbf{1}_n$.

[3]Come implementato dalla funzione `kmeans` di MATLAB.

Capitolo 2

Fair Spectral Clustering

In questo capitolo, si mostra come incorporare nell'algoritmo di spectral clustering analizzato nel capitolo 1 la nozione di group fairness proposta in [Chierichetti et al., 2018] e generalizzata in [Kleindessner et al., 2019]. Secondo tale nozione, fissata una partizione $\mathcal{P}$ del dataset, un clustering è equo rispetto alla partizione considerata se ogni insieme di $\mathcal{P}$ è rappresentato sempre nella stessa proporzione in ciascun cluster. Ci occupiamo di sviluppare algoritmi di spectral clustering capaci di ottenere raggruppamenti equi in tal senso. Il vincolo di equità appena descritto lo chiamiamo *vincolo di group fairness*.

Nella sezione 2.1, si definisce rigorosamente e si elabora il vincolo di group fairness. Il vincolo è implementato in SC tramite l'aggiunta al problema (1.9) di una restrizione lineare su H. Nella sezione 2.2, si risolve tale problema come in [Kleindessner et al., 2019], ottenendo così l'algoritmo FairSC. Si affrontano sinteticamente i problemi dell'algoritmo legati all'elevato costo computazionale. Nella sezione 2.3, si mostra un modo alternativo di risolvere il problema con vincolo lineare, che conduce all'algoritmo gep-FairSC. A seguire, si mostra una semplice variante di FairSC che costituisce il primo passo verso l'algoritmo s-FairSC [Wang et al., 2023]. A quest'ultimo è dedicata la sezione 2.4 , in cui si costruisce l'algoritmo e se ne discutono alcuni aspetti implementativi.

2.1 Il vincolo di group fairness

Torniamo a considerare un grafo semplice e pesato $\mathcal{G}(V, W)$ con nodi $V = \{v_1, \ldots, v_n\}$ e matrice di adiacenza $W = (w_{ij}) \in \mathbb{R}^{n \times n}$ come all'inizio del capitolo 1. Sia data una partizione di V in h sottoinsiemi, che chiameremo *gruppi*

(*groups*),

$$V = V_1 \sqcup \cdots \sqcup V_h. \tag{2.1}$$

Così come era per i clusters, tale partizione può essere codificata con una matrice,

$$G = (g_{is}) \in \mathbb{R}^{n\times h} \quad \text{dove} \quad g_{is} := \begin{cases} 1, & \text{se } v_i \in V_s, \\ 0, & \text{altrimenti,} \end{cases} \tag{2.2}$$

per $i = 1, \ldots, n$, $s = 1, \ldots, h$. Ora, la definizione di fairness introdotta in [Chierichetti et al., 2018] consiste nel vincolo per i nodi di ciascun gruppo siano presenti in ogni cluster sempre nella stessa percentuale.

Definizione 2.1.1. Un clustering $V = C_1 \sqcup \cdots \sqcup C_k$ si dice *group fair* rispetto alla partizione (2.1) se

$$\frac{|V_s \cap C_l|}{|C_l|} = \frac{|V_s|}{|V|}, \tag{2.3}$$

per ogni $s = 1, \ldots, h$, $l = 1, \ldots, k$.

Come già trattato nell'introduzione della tesi, da un punto di vista probabilistico questo vincolo fa sì che la probabilità che una qualche osservazione sia assegnata a un certo cluster sia indipendente dal gruppo a cui essa appartiene. In questo senso, il vincolo garantisce che il risultato dell'algoritmo sia "equo" rispetto alla partizione in gruppi considerata. Per questo i gruppi vengono anche chiamati *gruppi protetti*.

La condizione di fairness appena illustrata può essere riformulata in modo compatto usando le matrici indicatrici G e H definite da (2.2) e (1.2) rispettivamente. Per farlo introduciamo le matrici

$$M := G^T H,$$
$$Z := (G^T \mathbf{1}_n)(H^T \mathbf{1}_n)^T,$$

dove $\mathbf{1}_n \in \mathbb{R}^n$ denota il vettore con tutti i coefficienti unitari. Si osservi che, equivalentemente,

$$M := (m_{sl}) \in \mathbb{R}^{h\times k} \quad \text{dove} \quad m_{sl} = \frac{|V_s \cap C_l|}{\sqrt{\text{vol}(C_l)}},$$
$$Z := (z_{sl}) \in \mathbb{R}^{h\times k} \quad \text{dove} \quad z_{sl} = \frac{|V_s| \cdot |C_l|}{\sqrt{\text{vol}(C_l)}}.$$

Conseguentemente, ricordando che $|V| = n$, la condizione di group fairness (2.3) si può scrivere nella forma

$$nM = Z. \tag{2.4}$$

Denotiamo inoltre

$$F_0 \coloneqq G - \mathbf{1}_n z^T \in \mathbb{R}^{n\times h} \quad \text{dove} \quad z \coloneqq \frac{G^T \mathbf{1}_n}{n} = \frac{1}{n}\begin{bmatrix} |V_1| \\ \vdots \\ |V_h| \end{bmatrix}. \tag{2.5}$$

Allora, la condizione (2.4) è equivalente anche a

$$F_0^T H = 0. \tag{2.6}$$

Infatti,

$$\begin{aligned} nM - Z &= nG^T H - (G^T \mathbf{1}_n)(H^T \mathbf{1}_n)^T \\ &= nG^T H - nz(H^T \mathbf{1}_n)^T \\ &= nG^T H - nz\,\mathbf{1}_n^T H \\ &= n(G^T - z\,\mathbf{1}_n^T)H = nF_0^T H, \end{aligned}$$

da cui si ha immediatamente che $nM = Z$ se e soltanto se $F_0^T H = 0$.

Il lemma che segue mostra che il vincolo di group fairness si può scrivere in modo ancora più compatto sostituendo, nell'equazione (2.6), alla matrice F_0 la sua sottomatrice ottenuta eliminandone l'ultima colonna. L'idea di utilizzare solo le prime $h-1$ colonne di F_0 risale a [Kleindessner et al., 2019]. A partire da tale idea, in [Wang et al., 2023] gli autori hanno provato il lemma a seguire mostrando che in effetti non se ne può eliminare più di una perché F_0 ha rango esattamente pari a $h-1$.

Lemma 2.1.1. *Sia H la matrice indicatrice del clustering* (1.1)*, definita in* (1.2)*, e G la matirce indicatrice dei gruppi* (2.1)*, definita in* (2.2)*. Sia $F_0 \in \mathbb{R}^{n\times h}$ come in* (2.5) *e $F \coloneqq F_0(:, 1:h-1)$ la sottomatrice di F_0 ottenuta eliminandone l'ultima colonna. Allora,*

1. $\operatorname{rank}(F_0) = \operatorname{rank}(F) = h - 1$;
2. *Il clustering considerato è group fair rispetto ai gruppi* (2.1) *se e soltanto se*

$$F^T H = 0. \tag{2.7}$$

Dimostrazione. Segue subito dalla definizione di matrice indicatrice dei gruppi e dal fatto che i gruppi formano una partizione di V che ogni riga della matrice G ha uno e un solo coefficiente non nullo, pari a 1. Ne segue che

$$G\,\mathbf{1}_h = \mathbf{1}_n \quad \text{e} \quad \mathbf{1}_n^T G\,\mathbf{1}_h = n.$$

Inoltre, quanto già osservato sulle righe di G implica l'ortogonalità delle sue colonne. In particolare le colonne di G sono linearmente indipendenti, dunque dalla prima delle due equazioni precedenti si ottiene

$$\mathbf{1}_h = G^\dagger \mathbf{1}_n, \tag{2.8}$$

dove $G^\dagger := (G^T G)^{-1} G^T$ è la pseudoinversa di G. Sia $x \in \mathbb{R}^h$ tale che $F_0 x = 0$. Dalla definizione (2.5) di F_0, ciò è equivalente a

$$Gx - \frac{1}{n} \mathbf{1}_n (G^T \mathbf{1}_n)^T x = 0,$$

cioè

$$Gx = \alpha \mathbf{1}_n \quad \text{dove} \quad \alpha := \frac{(G^T \mathbf{1}_n)^T x}{n},$$

da cui, moltiplicando a sinistra per $G^\dagger$,

$$x = \alpha G^\dagger \mathbf{1}_n \overset{(2.8)}{=} \alpha \mathbf{1}_h .$$

Possiamo dunque affermare che il nucleo di F_0 è contenuto nel sottospazio di $\mathbb{R}^h$ generato da $\mathbf{1}_h$, in formule $\ker(F_0) \subseteq \operatorname{span}(\mathbf{1}_h)$. D'altra parte $\mathbf{1}_h$ è un autovettore relativo a 0 di F_0 perché

$$\begin{aligned} F_0 \mathbf{1}_h &= G \mathbf{1}_h - \frac{1}{n} \mathbf{1}_n (G^T \mathbf{1}_n)^T \mathbf{1}_h \\ &= G \mathbf{1}_h - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^T G \mathbf{1}_h \\ &= \mathbf{1}_n - \frac{1}{n} \mathbf{1}_n \mathbf{1}_n^T \mathbf{1}_n \\ &= \mathbf{1}_n - \mathbf{1}_n = 0, \end{aligned}$$

dove si è usato il fatto, già osservato, che $G \mathbf{1}_h = \mathbf{1}_n$. Perciò $\ker(F_0) = \operatorname{span}(\mathbf{1}_n)$ e $\operatorname{rank}(F_0) = h - 1$. Inoltre da $F_0 \mathbf{1}_h = 0$ segue evidentemente che l'ultima colonna di F_0 è combinazione lineare delle altre, precisamente

$$F_0 e_h = -F \mathbf{1}_{h-1} .$$

Dunque lo span delle colonne di F_0 e delle colonne di F coincidono, perciò

i. $\operatorname{rank}(F_0) = \operatorname{rank}(F) = h - 1$;

ii. $F_0^T x = 0$ se e soltanto se $F^T x = 0$ per ogni $x \in \mathbb{R}^n$.

Da queste due si conclude immediatamente ricordando che la condizione di group fairness si può scrivere anche nella formulazione (2.6). □

Si osservi la struttura della matrice F:

$$F = (f_{is}) \in \mathbb{R}^{n\times(h-1)} \quad \text{dove} \quad f_{is} = \begin{cases} 1 - \frac{|V_s|}{n}, & \text{se } v_i \in V_s, \\ -\frac{|V_s|}{n}, & \text{altrimenti,} \end{cases} \tag{2.9}$$

per $i = 1, \dots, n$ e $s = 1, \dots, h-1$. Pertanto la matrice F contiene tutte le informazione relative al partizionamento di V in gruppi. Per questo è chiamata *group-membership matrix*.

2.2 L'algoritmo FairSC

2.2.1 Costruzione dell'algoritmo FairSC

Il Lemma 2.1.1 permette di incorporare il vincolo di group fairness nell'algoritmo SC in modo elegante. Ricordando che, per determinare il clustering ottimale, l'algoritmo SC applica il metodo delle k-medie alle righe della matrice H soluzione del problema rilassato (1.9), per implementare nell'algoritmo la nozione di group fairness è sufficiente aggiungere a tale problema di minimizzazione il vincolo lineare (2.7). Si ottiene

$$\min_{H\in\mathbb{R}^{n\times k}} \operatorname{Tr}(H^T L H) \quad \text{t.c.} \quad H^T D H = I_k \text{ e } F^T H = 0. \tag{2.10}$$

Notiamo che, affinché una soluzione possa esistere, dobbiamo supporre $k \leq n - h + 1$. Altrimenti, non potrebbero esistere k vettori D-ortonormali (quindi linearmente indipendenti) in $\ker(F^T)$, che ha dimensione $n - h + 1$, quindi non potrebbe esistere una matrice $H \in \mathbb{R}^{n\times k}$ che soddisfa le due equazioni in (2.10). Tuttavia, ciò non è particolarmente restrittivo perché in tutte le applicazioni reali sia h che k sono molto più piccoli della dimensione del dataset n.

Si osservi che $F^T H = 0$ se e solo se le colonne di H sono contenute in $\ker(F^T)$. Ricordando che la matrice F ha rango massimo, ciò è equivalente all'esistenza di una matrice $Y \in \mathbb{R}^{(n-h+1)\times k}$ tale che $H = ZY$, dove $Z \in \mathbb{R}^{n\times(n-h+1)}$ ha come colonne una base ortonormale di $\ker(F^T)$. Pertanto, H è soluzione di (2.10) se e soltanto se è della forma $H = ZY$, dove Y risolve

$$\min_{Y\in\mathbb{R}^{(n-h+1)\times k}} \operatorname{Tr}\big(Y^T \big[Z^T L Z\big] Y\big) \quad \text{t.c.} \quad Y^T \big[Z^T D Z\big] Y = I_k. \tag{2.11}$$

Infine, col cambio di variabili $X = QY$, dove $Q := \sqrt{Z^T D Z}$ (si ricordi che D è definita positiva e Z è alta con rango massimo, cosicché anche $Z^T D Z$ è definita

positiva), otteniamo

$$\min_{X \in \mathbb{R}^{(n-h+1)\times k}} \operatorname{Tr}\left(X^T M X\right) \quad \text{t.c.} \quad X^T X = I_k, \tag{2.12}$$

dove $M := Q^{-1} Z^T L Z Q^{-1}$. Poiché M è simmetrica e per ipotesi $k \leq n - h + 1$, il Teorema 1.1.1 assicura che una soluzione di tale problema è data da una matrice che ha come colonne degli autovettori ortonormali relativi ai primi k autovalori di M. Determinata una tale X, troviamo $H = ZQ^{-1}X$ soluzione di (2.10). I passaggi mostrati motivano l'Algoritmo 2, FairSC [Kleindessner et al., 2019].

Algoritmo 2 FairSC

Inizialization: matrice di adiacenza $W \in \mathbb{R}^{n\times n}$; matrice dei gradi $D \in \mathbb{R}^{n\times n}$ definita positiva; group-membership matrix $F \in \mathbb{R}^{n\times(h-1)}$; $k \in \mathbb{N}$, $k \leq n - h + 1$

Until convergence: un raggruppamento degli indici $1 : n$ in k clusters

1: calcola la matrice Laplaciana $L = D - W$
2: calcola una base ortonormale Z del nucleo di F^T
3: calcola la radice quadrata $Q = (Z^T D Z)^{\frac{1}{2}}$
4: calcola $M = Q^{-1} Z^T L Z Q^{-1}$
5: calcola i k autovalori più piccoli della matrice M e i corrispondenti autovettori $X \in \mathbb{R}^{(n-h+1)\times k}$
6: applica il clustering con k-medie alle righe di $H = ZQ^{-1}X$

2.2.2 Commenti sul costo computazionale di FairSC

Nel capitolo 3 si mostrerà con alcuni esperimenti che effettivamente tale algoritmo determina clusterings più equi rispetto all'Algoritmo 1. Tuttavia, in questa forma l'introduzione della nozione di fairness nell'algoritmo di spectral clustering ha come effetto collaterale un aumento significativo del tempo di esecuzione. L'algoritmo SC ha un costo computazionale in termini di tempo dominato dal calcolo dei primi k autovettori di L_n. Il costo di tale procedura è in generale $\mathcal{O}(n^3)$. Tuttavia, se k è piccolo o L_n è sparsa essa può essere svolta in modo più efficiente che con tempo cubico sfruttando i moderni metodi iterativi sviluppati per il problema (ad esempio, quelli implementati dal software ARPACK utilizzato dalla funzione `eigs` di MATLAB). Al contrario, anche in tal caso il costo computazionale di FairSC è elevato:

1. al terzo passo, l'algoritmo calcola la radice quadrata della matrice simmetrica definita positiva $Z^T DZ$, densa e di dimensione $(n-h+1) \times (n-h+1)$;

2. al quarto e sesto passo, l'algoritmo deve risolvere dei sistemi lineari con la matrice simmetrica definita positiva Q, densa e di dimensione $(n-h+1) \times (n-h+1)$, per svolgere i prodotti che coinvolgono Q^{-1};

3. al terzo, quarto e sesto passo, l'algoritmo calcola alcuni prodotti fra matrici dense e di grandi dimensioni.

Tutte queste operazioni hanno un costo computazionale $\mathcal{O}(n^3)$. Un altro fattore che contribuisce all'elevato tempo di calcolo, anche se meno impattante, è il fatto che FairSC necessita, a differenza di SC, il calcolo di una base ortonormale di uno spazio di grande dimensione (il nucleo di F^T). Tale procedura può essere svolta sfruttando la decomposizione QR o la SVD di F con costo $\mathcal{O}(n(h-1)^2)$. Infine, un algoritmo iterativo come quello citato sopra per il calcolo delle prime autocoppie di una matrice ha un costo computazionale per iterazione confrontabile in termini asintotici a quello del prodotto matrice-vettore con la matrice stessa. Nel caso di FairSC, la matrice oggetto dello studio spettrale è M, che in generale è densa. Pertanto, il costo computazionale del prodotto matrice-vettore risulta essere $\mathcal{O}((n-h+1)^2)$ (qui h è il numero di gruppi protetti ed è quindi solitamente molto più piccolo di n). Nel caso di SC, invece, vengono calcolati i primi autovettori della matrice L_n, spesso sparsa, perciò tale costo è meno che quadratico in n (ad esempio $\mathcal{O}(n\log(n))$ o $\mathcal{O}(n)$). Anche questo concorre all'aumento complessivo del costo computazionale.

Di conseguenza, l'Algoritmo 2 può essere utilizzato solo per problemi di dimensioni moderate.

2.3 L'algoritmo gep-FairSC

In questa sezione, studiamo una variante di FairSC in cui si riconduce il problema (2.10) ad un problema generalizzato agli autovalori

$$Ax = \lambda Bx \tag{2.13}$$

dove A e B sono della forma

$$A = \begin{bmatrix} K & C \\ C^T & 0 \end{bmatrix}, \quad B = \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix}, \tag{2.14}$$

con $m, n \in \mathbb{N}$, $m < n$, $K \in \mathbb{R}^{n\times n}$ simmetrica, $C \in \mathbb{R}^{n\times m}$ di rango massimo m, $M \in \mathbb{R}^{n\times n}$ simmetrica definita positiva.

2.3.1 Teoria del problema generalizzato agli autovalori

Per ricondurre il problema (2.10) a un problema generalizzato agli autovalori della forma illustrata, è necessario preliminarmente sviluppare un importante risultato sulle soluzioni del problema (2.13). La teoria relativa a tale problema presentata in questa sezione è tratta da [Cliffe et al., 1994].

Ricordiamo innanzitutto qualche basilare definizione riguardante il problema generalizzato autovalori $Ax = \lambda Bx$, con $A, B \in \mathbb{C}^{N\times N}$, $N \in \mathbb{N}$.

Definizione 2.3.1. La coppia (A, B) si dice *singolare* se

$$\det(A - \lambda B) = 0, \tag{2.15}$$

per ogni $\lambda \in \mathbb{C}$, altrimenti si dice *regolare*.

Nel seguito poniamo $\beta/0 = \infty$ per ogni $\beta \in \mathbb{C}$, $\beta \neq 0$.

Definizione 2.3.2. Sia (A, B) una coppia regolare e siano $\alpha, \beta \in \mathbb{C}$ non entrambi nulli, $x \in \mathbb{C}^N$, $x \neq 0$. Se vale l'uguaglianza

$$\alpha Ax = \beta Bx,$$

allora $\lambda = \frac{\beta}{\alpha} \in \mathbb{C} \cup \{\infty\}$ si dice *autovalore* della coppia (A, B) con *autovettore* x. In tal caso, (λ, x) si dice *autocoppia* di (A, B).

Le soluzioni del problema agli autovalori $Ax = \lambda Bx$ sono per definizione le autocoppie di (A, B).

Si notino le seguenti:

- $\lambda \in \mathbb{C}$ è un autovalore di (A, B) se e soltanto se esiste un vettore $x \in \mathbb{C}^N$, $x \neq 0$, tale che
$$Ax = \lambda x,$$
cioè se e soltanto se λ è una soluzione dell'equazione (2.15), e in tal caso x è un autovettore relativo a λ;

- $\lambda = \infty$ è un un autovalore di (A, B) se e soltanto se B è singolare e in tal caso un autovettore corrispondente è un qualunque vettore non nullo in $\ker(B)$.

Osservazione 2.3.1. Date $A, B \in \mathbb{C}^{N\times N}$, $\det(A - \lambda B)$ è un polinomio nella variabile λ. Esso prende il nome di *polinomio caratteristico* di (A, B). Se (A, B) è singolare, allora si tratta del polinomio identicamente nullo. Altrimenti, è un polinomio di grado al più N. In particolare, una coppia regolare ha al più N autovalori finiti.

Osservazione 2.3.2. Una condizione sufficiente affinché la coppia (A, B) sia regolare è che B o A siano invertibili. Infatti, se B è invertibile, allora $\det(A - \lambda B)$ coincide a meno di una costante non nulla con il polinomio caratteristico della matrice AB^{-1}, quindi è non nullo. Se A è invertibile, allora (B, A) è regolare. Osservando che

$$\alpha Ax = \beta Bx \quad \text{se e soltanto se} \quad \beta Bx = \alpha Ax,$$

si conclude che anche (A, B) è regolare e gli autovalori (A, B) sono i reciproci di quelli di (B, A).

Osservazione 2.3.3. È utile notare che se A è hermitiana e B è hermitiana definita positiva, allora (A, B) è regolare ed ha autovalori reali. La regolarità segue dall'osservazione precedente. Inoltre, se

$$B = LL^H$$

è la decomposizione di Cholesky di B, allora $Ax = \lambda Bx$ è equivalente a

$$L^{-1}AL^{-H}(L^H x) = \lambda(L^H x).$$

Ponendo $y := L^H x$, riconosciamo il problema agli autovalori con matrice hermitiana

$$L^{-1}AL^{-H}y = \lambda y. \tag{2.16}$$

Poiché B è invertibile, ∞ non è autovalore di (A, B). Infine, dai passaggi appena mostrati abbiamo che gli autovalori (finiti) di (A, B) coincidono con quelli di (2.16), e sono quindi reali.

Torniamo a considerare A e B della forma (2.14). Sia inoltre

$$C = QR = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix},$$

una fattorizzazione QR di C, con $Q_1 \in \mathbb{R}^{n\times m}$, $Q_2 \in \mathbb{R}^{n\times(n-m)}$ e $R_1 \in \mathbb{R}^{m\times m}$.

Teorema 2.3.1. *Valgono le seguenti:*

1. *La coppia* (A, B)*, con* A *e* B *della forma* (2.14)*, è regolare e ha al più* $n - m$ *autovalori finiti distinti, tutti reali, che coincidono con quelli di* $(Q_2^T K Q_2, Q_2^T M Q_2)$*;*

2. *se* (λ, u, p) *è soluzione del problema*

$$\begin{bmatrix} K & C \\ C^T & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \lambda \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix}, \tag{2.17}$$

   con λ *finito, allora* $(\lambda, Q_2^T u)$ *è soluzione del problema*

$$Q_2^T K Q_2 y = \lambda Q_2^T M Q_2 y; \tag{2.18}$$

3. *se* (λ, y) *è soluzione del problema* (2.18)*, allora* (λ, u, p) *è soluzione di* (2.17)*, dove*

$$\begin{aligned} u &:= Q_2 y, \\ p &:= -R_1^{-1} Q_1^T (K - \lambda M) Q_2 y. \end{aligned}$$

Dimostrazione. Per definizione, la coppia (A, B) è singolare se e soltanto se per ogni $\lambda \in \mathbb{C}$ esistono $u \in \mathbb{C}^n$ e $p \in \mathbb{C}^m$, non entrambi nulli, tali che valga l'equazione (2.17),

$$\begin{bmatrix} K & C \\ C^T & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \lambda \begin{bmatrix} M & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix}.$$

Sia V la matrice ortogonale

$$V := \begin{bmatrix} Q & 0 \\ 0 & I_m \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}.$$

Ponendo

$$z = V^T \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} Q^T u \\ p \end{bmatrix} = \begin{bmatrix} Q_1^T u \\ Q_2^T u \\ p \end{bmatrix} =: \begin{bmatrix} u_1 \\ u_2 \\ p \end{bmatrix},$$

notiamo che la (2.17) è equivalente a

$$V^T A V z = \lambda V^T B V z. \tag{2.19}$$

Mettendo in evidenza la struttura a blocchi delle due matrici nell'equazione (2.19), si ottiene

$$\begin{bmatrix} K_{11} & K_{12} & C_1 \\ K_{21} & K_{22} & C_2 \\ C_1^T & C_2^T & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ p \end{bmatrix} = \lambda \begin{bmatrix} M_{11} & M_{12} & 0 \\ M_{21} & M_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ p \end{bmatrix},$$

dove $K_{ij} \coloneqq Q_i^T K Q_j$, $M_{ij} \coloneqq Q_i^T M Q_j$, $C_i \coloneqq Q_i^T C$ per $i, j = 1, 2$. Osservando che

$$C_1 = Q_1^T C = R_1 \quad \text{e} \quad C_2 = Q_2^T C = 0,$$

la precedente equazione si riscrive come

$$\begin{bmatrix} K_{11} & K_{12} & R_1 \\ K_{21} & K_{22} & 0 \\ R_1^T & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ p \end{bmatrix} = \lambda \begin{bmatrix} M_{11} & M_{12} & 0 \\ M_{21} & M_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ p \end{bmatrix}.$$

Perciò $\lambda \in \mathbb{C}, u \in \mathbb{C}^n, p \in \mathbb{C}^m$, con u e p non entrambi nulli, soddisfano la (2.17) se e soltanto se

$$\begin{cases} u_1 = 0, & \text{(2.20a)} \\ K_{22} u_2 = \lambda M_{22} u_2, & \text{(2.20b)} \\ p = -R_1^{-1}(K_{12} - \lambda M_{12}) u_2, & \text{(2.20c)} \end{cases}$$

con $u_2 \neq 0$. Per l'Osservazione (2.3.3), la coppia (K_{22}, M_{22}) è regolare. Segue che la (2.17) è vera per un numero finito di $\lambda \in \mathbb{C}$ e quindi la coppia (A, B) è regolare. Inoltre, quanto mostrato ci permette di affermare che gli autovalori finiti di (A, B) coincidono con quelli di (K_{22}, M_{22}) e sono perciò al più $n - m$ e tutti reali. Questo prova il punto 1.

Sia ora (λ, u, p) una soluzione del problema (2.17), con $\lambda \neq \infty$. Allora, lo scalare λ ed il vettore non nullo $\left[\begin{smallmatrix} u \\ p \end{smallmatrix}\right]$ devono soddisfare la (2.17). Pertanto, $u_1 = Q_1^T u, u_2 = Q_2^T u$ e p devono soddisfare il sistema (2.20). Poiché $u_2 \neq 0$ (altrimenti anche p sarebbe nullo e si avrebbe $u = 0$ e $p = 0$) e u_2 soddisfa la (2.20b), si ha che (λ, u_2) è una soluzione di (2.18). È così provato il punto 2.

Sia (λ, u_2) soluzione di (2.20b). Allora, per quanto già mostrato, $\left(\lambda, \left[\begin{smallmatrix} 0 \\ u_2 \\ p \end{smallmatrix}\right]\right)$ è soluzione di (2.19), dove p è definito come in (2.20c). Segue che λ è autovalore di (2.17) con autovettore

$$V \begin{bmatrix} 0 \\ u_2 \\ p \end{bmatrix} = \begin{bmatrix} Q_2 u_2 \\ p \end{bmatrix}.$$

È così provato il punto 3. □

2.3.2 Costruzione dell'algoritmo gep-FairSC

Ritorniamo nelle notazioni della sezione 2.2. In particolare, ricordiamo che abbiamo denotato con la lettera Z una matrice che ha colonne che formano una base ortonormale di $\ker(F^T)$ e abbiamo definito $Q \coloneqq \sqrt{Z^T D Z}$. Fissiamo $k \leq n - h + 1$, come fatto nella sezione 2.2.

Proposizione 2.3.1. *Sia* $Y \in \mathbb{R}^{(n-h+1)\times k}$ *matrice* Z^TDZ*-ortogonale con colonne costituite da autovettori del problema*

$$Z^TLZy = \lambda Z^TDZy, \tag{2.21}$$

relativi ai primi k autovalori. Allora la matrice $ZY \in \mathbb{R}^{n\times k}$ *è una soluzione del problema* (2.10)*,*

$$\min_{H\in\mathbb{R}^{n\times k}} \mathrm{Tr}(H^TLH) \quad t.c. \quad H^TDH = I_k \; e \; F^TH = 0.$$

Dimostrazione. Abbiamo già dimostrato nella sottosezione 2.2.1 che una soluzione di (2.10) è $ZQ^{-1}X$, con X che ha colonne ortonormali date da autovettori del problema

$$Q^{-1}Z^TLZQ^{-1}x = \lambda x, \tag{2.22}$$

relativi ai primi k autovalori. Ponendo $y = Q^{-1}x$ e ricordando che

$$Q^2 = Z^TDZ,$$

otteniamo il problema (2.21). Se Y ha come colonne degli autovettori di (2.21) relativi ai primi k autovalori, allora le colonne di $X := QY$ sono autovettori di (2.22) relativi agli stessi autovalori. Inoltre, se Y è Z^TDZ-ortogonale, allora

$$X^TX = Y^TQ^2Y = Y^TZ^TDZY = I_k.$$

Perciò $ZQ^{-1}X = ZQ^{-1}QY = ZY$ è soluzione di (2.10). □

Poniamo $m := h - 1 < n$ e consideriamo il problema

$$\begin{bmatrix} L & F \\ F^T & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \lambda \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix}. \tag{2.23}$$

Abbiamo $k \leq n - h + 1 = n - m$.

Corollario 2.3.1. *Sia* $\left[\begin{smallmatrix} U \\ P \end{smallmatrix}\right] \in \mathbb{R}^{(n+m)\times k}$ *con colonne date da autovettori di* (2.23) *relativi ai primi k autovalori e tale che*

$$\begin{bmatrix} U^T & P^T \end{bmatrix} \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U \\ P \end{bmatrix} = U^TDU = I_k.$$

Allora U è una soluzione di (2.10)*.*

Dimostrazione. Applicando il Teorema 2.3.1 al problema generalizzato agli autovalori (2.23), abbiamo che $Y := Z^T U$ ha come colonne degli autovettori relativi ai primi k autovalori di (2.21). Inoltre, poiché per ipotesi $U^T D U = I_k$, risulta

$$\begin{aligned} Y^T Z^T D Z Y &= U^T Z Z^T D Z Z^T U \\ &\quad \text{(la } U \text{ soddisfa } F^T U = 0 \text{ dunque } Z Z^T U = U\text{)} \\ &= U^T D U = I_k. \end{aligned}$$

Usando la Proposizione 2.3.1, abbiamo quindi che $ZY = ZZ^T U = U$ è soluzione di (2.10). L'esistenza di una matrice $\left[\begin{smallmatrix} U \\ P \end{smallmatrix}\right]$ come nell'enunciato può essere vista come una conseguenza del Teorema 2.3.1. □

Il Corollario 2.3.1 ci permette di sviluppare un semplice algoritmo di spectral clustering con vincolo di group fairness, l'Algoritmo 3. Il nome dell'algoritmo richiama il fatto che esso si basa sull'idea di riformulare il problema di partenza (2.10) in un opportuno problema generalizzato agli autovalori (*generalized eigenvalue problem*).

Algoritmo 3 gep-FairSC

Inizialization: matrice di adiacenza $W \in \mathbb{R}^{n \times n}$; matrice dei gradi $D \in \mathbb{R}^{n \times n}$ definita positiva; group-membership matrix $F \in \mathbb{R}^{n \times (h-1)}$; $k \in \mathbb{N}$, $k \leq n - h + 1$

Until convergence: un raggruppamento degli indici $1 : n$ in k clusters

1: calcola la matrice Laplaciana $L = D - W$
2: calcola i k autovalori più piccoli del problema (2.23) e i corrispondenti autovettori $X \in \mathbb{R}^{(n+m) \times k}$
3: applica il clustering con k-medie alle righe di $H = X(1 : n, :)$

Notiamo che il costo computazione dell'algoritmo è ancora una volta dominato da quello della risoluzione del problema agli autovalori. In MATLAB, esso può essere risolto in modo efficiente con la funzione `eigs`.

2.4 L'algoritmo s-FairSC

2.4.1 Un primo passo verso s-FairSC

In [Wang et al., 2023] è proposta una semplice variante dell'algoritmo FairSC in cui si rimuove il vincolo lineare in (2.10) senza dover calcolare la radice

quadrata di una matrice di grande dimensione. Si riporta sotto l'equazione (2.10):

$$\min_{H\in\mathbb{R}^{n\times k}} \operatorname{Tr}\left(H^T L H\right) \quad \text{t.c.} \quad H^T D H = I_k \text{ e } F^T H = 0.$$

Col cambio di variabili $X = D^{\frac{1}{2}} H$, si ottiene

$$\min_{X\in\mathbb{R}^{n\times k}} \operatorname{Tr}\left(X^T L_n X\right) \quad \text{t.c.} \quad X^T X = I_k \text{ e } C^T X = 0, \tag{2.24}$$

dove $C \coloneqq D^{-\frac{1}{2}} F$ e L_n è la matrice Laplaciana normalizzata simmetrica,

$$L_n \coloneqq D^{-\frac{1}{2}} L D^{-\frac{1}{2}} = I_n - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}.$$

Per eliminare il vincolo $C^T X = 0$, determiniamo innanzitutto $V \in \mathbb{R}^{n\times(n-h+1)}$ con colonne che costituiscono una base ortonormale di $\ker(C^T)$. Allora X soddisfa $C^T X = 0$ se e soltanto se esiste una matrice $Y \in \mathbb{R}^{(n-h+1)\times k}$ tale che $X = VY$. Pertanto il problema (2.24) è equivalente a

$$\min_{Y\in\mathbb{R}^{(n-h+1)\times k}} \operatorname{Tr}\left(Y^T L_n^v Y\right) \quad \text{t.c.} \quad Y^T Y = I_k, \tag{2.25}$$

dove

$$L_n^v \coloneqq V^T L_n V \in \mathbb{R}^{(n-h+1)\times(n-h+1)}. \tag{2.26}$$

Per il Teorema 1.1.1, ricordando che $k \leq n - h + 1$, ci si è quindi ricondotti al problema agli autovalori con matrice simmetrica semidefinita positiva

$$L_n^v y = \lambda y. \tag{2.27}$$

Una soluzione del problema (2.25) è data da una matrice Y che ha come colonne una base ortonormale di autovettori relativi ai primi k autovalori di L_n^v. Determinata una tale Y, per ottenere H soluzione del problema di partenza basterà quindi calcolare

$$H = D^{-\frac{1}{2}} V Y. \tag{2.28}$$

Osservazione 2.4.1. Riprendiamo il problema (2.23) studiato in gep-FairSC,

$$\begin{bmatrix} L & F \\ F^T & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \lambda \begin{bmatrix} D & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix}.$$

Normalizzando opportunamente le due matrici, otteniamo un problema con stessi autovalori e i cui autovettori soddisfano una semplice relazione con quelli del problema di partenza:

$$\begin{bmatrix} L_n & C \\ C^T & 0 \end{bmatrix} \begin{bmatrix} v \\ q \end{bmatrix} = \lambda \begin{bmatrix} I_n & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v \\ q \end{bmatrix}.$$

Alla luce del Teorema 2.3.1, questo problema è equivalente a

$$L_n^v y = \lambda y,$$

nel senso enunciato dal teorema. Dunque possiamo dire che questo nuovo algoritmo risolve un problema agli autovalori che rappresenta una versione "proiettata" del problema (2.23). Entrambi i problemi vengono utilizzati per calcolare una soluzione del problema (2.10), ma il secondo ha dimensione $n - h + 1$ anziché $n + h - 1$. Il passaggio di normalizzazione permette di ottenere, una volta applicato il Teorema 2.3.1, un problema agli autovalori standard anziché generalizzato, rendendone più agevole la risoluzione. In effetti, applicare il Teorema 2.3.1 senza aver prima normalizzato le due matrici conduce al problema

$$Z^T L Z y = \lambda Z^T D Z y,$$

che è quello studiato in FairSC tramite il calcolo di $Q = \sqrt{Z^T D Z}$ che permette di ricondursi a

$$\underbrace{Q^{-1} Z^T L Z Q^{-1}}_{M} w = \lambda w.$$

Abbiamo così l'Algoritmo 4. Per quanto questa semplice variante dell'algoritmo FairSC permetta di evitare il calcolo della radice quadrata di una matrice densa, le altre operazioni onerose dal punto di vista computazionale di FairSC rimangono. In particolare, è richiesto il calcolo di una base ortonormale di uno spazio di dimensione $n - h + 1$ e la risoluzione del problema agli autovalori con una matrice densa.

Algoritmo 4

Inizialization: matrice di adiacenza $W \in \mathbb{R}^{n \times n}$; matrice dei gradi $D \in \mathbb{R}^{n \times n}$ definita positiva; group-membership matrix $F \in \mathbb{R}^{n \times (h-1)}$; $k \in \mathbb{N}$, $k \leq n - h + 1$

Until convergence: un raggruppamento degli indici $1 : n$ in k clusters

1: calcola la matrice Laplaciana $L = D - W$
2: calcola $L_n = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$ e $C = D^{-\frac{1}{2}} F$
3: calcola una base ortonormale V del nucleo di C^T
4: calcola i k autovalori più piccoli di L_n^v, definita in (2.33), e i corrispondenti autovettori $Y \in \mathbb{R}^{(n-h+1) \times k}$
5: applica il clustering con k-medie alle righe di $H = D^{-\frac{1}{2}} V Y$

2.4.2 Costruzione dell'algoritmo s-FairSC

In questa sezione, mostriamo come riformulare il problema agli autovalori (2.27) per calcolare in modo efficiente degli autovettori ortonormali di L_n^v relativi a $\lambda_1, \ldots, \lambda_k$. Questo ci permetterà di implementare un algoritmo di fair spectral clustering scalabile, con costo computazionale paragonabile a quello dell'algoritmo di spectral clustering senza vincolo di group fairness.

Partendo da

$$(V^T L_n V)y = \lambda y,$$

una moltiplicazione a sinistra per V porta a

$$V(V^T L_n V)y = \lambda V y,$$

o, ricordando che le colonne di V sono ortonormali,

$$(VV^T L_n VV^T)Vy = \lambda V y. \tag{2.29}$$

Sia $P \coloneqq VV^T$, matrice di proiezione ortogonale sullo spazio delle colonne di V (cioè sullo spazio nullo di C^T). Denotiamo

$$L_n^p \coloneqq PL_nP = VL_n^v V^T \in \mathbb{R}^{n\times n}.$$

Allora, l'equazione precedente conduce al problema agli autovalori

$$L_n^p x = \lambda x. \tag{2.30}$$

Osservazione 2.4.2. (λ, y) è un'autocoppia si L_n^v se e soltanto se è della forma $(\lambda, V^T x)$, dove (λ, x) è una soluzione di (2.30) tale che $x \in \text{range}(C)^\perp$.

Dunque si è ricondotto il problema (2.27) al problema (2.30).

Prima di mostrare come nella pratica si può risolvere il problema iniziale a partire dallo studio delle autocoppie di L_n^p, motiviamo perché il problema (2.30) è preferibile rispetto al precedente. Innanzitutto, nella sottosezione 2.4.3 dimostreremo che la matrice P si può scrivere nella forma

$$P = I - C(C^T C)^{-1} C^T. \tag{2.31}$$

Ora, un metodo iterativo per l'approssimazione delle autocoppie di L_n^p avrà bisogno di utilizzare tale matrice solo per prodotti matrice-vettore. Dunque non è necessario costruire L_n^p e P esplicitamente. La scrittura (2.31) della matrice P permette quindi di svolgere il prodotto $L_n^p w$, $w \in \mathbb{R}^n$, in modo efficiente dato

che la C ha molte poche colonne ($h-1$) e L_n è sparsa. Al contrario, il prodotto matrice-vettore con L_n^v non si riesce a svolgere in modo efficiente perché V è densa e di grande dimensione, senza una struttura particolare. Questo rende il secondo problema computazionalmente vantaggioso rispetto al primo. Inoltre, il fatto di non utilizzare la V nel problema agli autovalori permetterà di non costruire tale matrice, quindi di non calcolare una base ortonormale del nucleo di C^T. Questo porta ad ulteriori benefici in termini di costo computazionale.

Il seguente risultato mette in relazione gli autovalori ed autovettori di L_n^p con quelli di L_n^v.

Proposizione 2.4.1. *Sia*

$$L_n^v = Y\Lambda_v Y^T$$

la decomposizione spettrale di L_n^v, dove $\Lambda_v = \operatorname{diag}(\lambda_1, \dots, \lambda_{n-h+1})$ contiene gli autovalori di L_n^v e Y è ortogonale di ordine $n-h+1$. Allora,

$$L_n^p = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Lambda_v & \\ & 0 \end{bmatrix} \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix}, \tag{2.32}$$

dove $[U_1, U_2]$ è ortogonale di ordine n, $U_1 = VY$ e U_2 ha come colonne una base ortonormale arbitraria di $\operatorname{range}(C)$.

Dimostrazione. Innanzitutto, osserviamo che $[U_1, U_2]$ è effettivamente ortogonale:

$$\begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix} \begin{bmatrix} U_1 & U_2 \end{bmatrix} = \begin{bmatrix} Y^T(V^TV)Y & Y^T(V^TU_2) \\ (V^TU_2)^TY & U_2^TU_2 \end{bmatrix} = I_n,$$

dove si è usato anche il fatto che $V^TU_2 = 0$ dato che le colonne di V generano lo spazio nullo di C^T (cioè l'ortogonale di $\operatorname{range}(C)$) e le colonne di U_2 generano $\operatorname{range}(C)$. L'uguaglianza $V^TU_2 = 0$ mostra anche che

$$L_n^pU_2 = VL_n^v(V^TU_2) = 0,$$

dunque le colonne di U_2 sono autovettori di L_n^p relativi a 0. Infine,

$$L_n^pU_1 = VL_n^vV^TVY = VL_n^vY = VY\Lambda_vY^TY = VY\Lambda_v = U_1\Lambda_v,$$

da cui la tesi. □

Ricordando che la matrice L_n^v è simmetrica semidefinita positiva, possiamo supporre $0 \le \lambda_1 \le \cdots \le \lambda_{n-h+1}$. Per la proposizione precedente, gli n autovalori di L_n^p sono

$$\underbrace{0 = \cdots = 0}_{h-1} \le \lambda_1 \le \cdots \le \lambda_{n-h+1},$$

dove l'autospazio relativo agli $h-1$ autovalori 0 evidenziati è range(C). Siamo interessati a determinare degli autovettori relativi ai k autovalori più piccoli di L_n^v, cioè $\lambda_1, \ldots, \lambda_k$. Per farlo, possiamo quindi calcolare gli autovettori relativi ai primi $h-1+k$ autovalori di L_n^p e selezionare quelli ortogonali a range(C).

Un modo preferibile, perché meno costoso, di selezionare gli autovettori desiderati si basa sull'utilizzo del seguente risultato.

Proposizione 2.4.2. *Siano $n, k \in \mathbb{N}$, con $k < n$, e $M \in \mathbb{R}^{n\times n}$ una matrice simmetrica con decomposizione spettrale*

$$M = Q\Lambda Q^T = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} \Lambda_1 & \\ & \Lambda_2 \end{bmatrix} \begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix},$$

dove Q è ortogonale e Λ è diagonale, $Q_1 \in \mathbb{R}^{n\times k}$, $Q_2 \in \mathbb{R}^{n\times(n-k)}$, $\Lambda_1 \in \mathbb{R}^{k\times k}$, $\Lambda_2 \in \mathbb{R}^{(n-k)\times(n-k)}$. Fissato uno shift *$\sigma \in \mathbb{R}$, definiamo $M_\sigma := M + \sigma Q_1 Q_1^T$. Allora, la matrice M_σ ha decomposizione spettrale*

$$M_\sigma = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} \Lambda_1 + \sigma I_k & \\ & \Lambda_2 \end{bmatrix} \begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix}.$$

In altri termini, gli autovalori di M_σ sono dati dall'unione di quelli contenuti in Λ_2 e quelli contenuti in Λ_1 traslati di σ mentre gli autovettori corrispondenti sono lasciati invariati.

Dimostrazione. Poiché Q è ortogonale, $Q_1^T Q_1 = I_k$ e $Q_1^T Q_2 = 0$. Inoltre, dalla decomposizione spettrale di M si ha

$$MQ_1 = Q_1\Lambda_1 \quad \text{e} \quad MQ_2 = Q_2\Lambda_2.$$

Pertanto, valgono le due seguenti:

$$M_\sigma Q_1 = MQ_1 + \sigma Q_1(Q_1^T Q_1) = Q_1\Lambda_1 + \sigma Q_1 = Q_1(\Lambda_1 + \sigma I_k),$$
$$M_\sigma Q_2 = MQ_2 + \sigma Q_1(Q_1^T Q_2) = MQ_2 = Q_2\Lambda_2,$$

il ché prova il risultato. □

Tornando nelle notazioni della Proposizione 2.4.1 nella pagina precedente, questo risultato fornisce un metodo efficiente per il calcolo degli autovettori relativi a $\lambda_1, \ldots, \lambda_k$ di L_n^p. Infatti, fissato $\sigma \in \mathbb{R}$ e posto

$$L_n^\sigma := L_n^p + \sigma U_2 U_2^T, \tag{2.33}$$

per la Proposizione 2.4.2 si ha

$$L_n^\sigma = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Lambda_v & \\ & \sigma I_{h-1} \end{bmatrix} \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix}. \tag{2.34}$$

Dunque, se $\sigma > \lambda_k$, gli autovalori $\lambda_1, \dots, \lambda_k$ corrisponderanno esattamente ai k autovalori più piccoli di L_n^σ. Per calcolare gli autovettori di L_n^p desiderati ci basterà quindi calcolare quelli relativi ai primi k autovalori di L_n^σ.

Si ricordi che siamo interessati a determinare degli autovettori ortonormali di L_n^v relativi a $\lambda_1, \dots, \lambda_k$. I risultati mostrati in questa sezione ci permettono di farlo in modo efficiente.

1. Scegliamo $\sigma \in \mathbb{R}$, $\sigma > \lambda_k$ (si veda la sottosezione 2.4.3 per ulteriori dettagli sulla scelta di σ).

2. Calcoliamo una matrice $X \in \mathbb{R}^{n \times k}$ costituita da autovettori ortonormali di L_n^σ relativi ai suoi k autovalori più piccoli (cioè $\lambda_1, \dots, \lambda_k$).

3. Per la Proposizione 2.4.1, lo spazio delle colonne di X è contenuto in $\text{range}(V) = \text{range}(C)^\perp$. Allora, per l'Osservazione 2.4.2 a pagina 28, le colonne di $Y \coloneqq V^T X$ sono autovettori di L_n^v relativi a $\lambda_1, \dots, \lambda_k$ e sono ortonormali perché

$$\begin{aligned} Y^T Y &= X^T V V^T X \\ &= X^T P X \\ &\quad (\text{range}(X) \text{ è contenuto in } \text{range}(V)) \\ &= X^T X = I_k. \end{aligned}$$

Fatto ciò, la procedura per trovare il fair clustering dei dati si conclude come nell'Algoritmo 4: si determina $H = D^{-\frac{1}{2}} V Y$, in accordo con la (2.28), per poi applicare il clustering con k-medie alle righe di H. Il prodotto fra matrici $D^{-\frac{1}{2}} V Y$ per ottenere H si può semplificare:

$$H = D^{-\frac{1}{2}} V Y = D^{-\frac{1}{2}} V V^T X = D^{-\frac{1}{2}} P X = D^{-\frac{1}{2}} X.$$

Perciò $H = D^{-\frac{1}{2}} X$. Si noti che questo permette di omettere il terzo punto della procedura illustrata (non è necessario calcolare Y). Questo è rilevante dal punto di vista computazionale perché si evitano alcuni prodotti fra matrici e si evita di utilizzare V che quindi non verrà costruita.

La proposizione seguente riassume quanto sviluppato all'interno di questa sezione.

Proposizione 2.4.3. *Sia $L_n^\sigma \in \mathbb{R}^{n\times n}$ come definita in* (2.33)*. Supponiamo $\sigma > \lambda_k(L_n^v)$, dove $\lambda_k(L_n^v)$ denota il k-esimo autovalore più piccolo di L_n^v. Se X ha colonne costituite da autovettori ortonormali relativi ai k autovalori più piccoli di L_n^σ, allora $H := D^{-\frac{1}{2}}X$ è soluzione del problema di minimizzazione* (2.10)*.*

L'Algoritmo 5, che sfrutta questo risultato, è stato proposto in [Wang et al., 2023] e prende il nome di Scalable FairSC, s-FairSC in breve.

Algoritmo 5 s-FairSC (scalable FairSC)

Inizialization: matrice di adiacenza $W \in \mathbb{R}^{n\times n}$; matrice dei gradi $D \in \mathbb{R}^{n\times n}$ definita positiva; group-membership matrix $F \in \mathbb{R}^{n\times(h-1)}$; $k \in \mathbb{N}$, $k \leq n-h+1$

Until convergence: un raggruppamento degli indici $1:n$ in k clusters

1: calcola la matrice Laplaciana $L = D - W$
2: calcola $L_n = D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$ e $C = D^{-\frac{1}{2}}F$
3: calcola i k autovalori più piccoli di L_n^σ, definita in (2.33), e i corrispondenti autovettori $X \in \mathbb{R}^{n\times k}$
4: applica il clustering con k-medie alle righe di $H = D^{-\frac{1}{2}}X$

2.4.3 Alcuni aspetti implementativi

Come già accennato, un metodo iterativo per il calcolo approssimato delle autocoppie della matrice L_n^σ, ad esempio quello implementato dalla funzione `eigs` di MATLAB, ha bisogno di accedere a L_n^σ solo per svolgere prodotti matrice-vettore. Dunque, dal punto di vista del costo computazionale, è conveniente evitare la costruzione esplicita della matrice. Dalla definizione di L_n^σ in (2.33), dato $w \in \mathbb{R}^n$ si ha

$$\begin{aligned} L_n^\sigma w &= PL_nPw + \sigma\left(I_n - P\right)w \\ &= P\left(L_n(Pw)\right) - \sigma Pw + \sigma w. \end{aligned}$$

Dunque il prodotto per L_n^σ richiede due prodotti matrice-vettore con la matrice di proiezione $P = VV^T = I - U_2U_2^T$ (dove U_2 ha come colonne una base ortonormale di $\text{range}(V)^\perp = \text{range}(C)$) e uno con L_n, sparsa.

Vogliamo ora osservare che il prodotto Pw si può svolgere senza costruire la matrice U_2 o V quindi senza calcolare una base ortonormale di $\text{range}(C)$ o del suo ortogonale. Innanzitutto, poiché le colonne di U_2 generano $\text{range}(C)$,

possiamo considerare $a \in \mathbb{R}^{h-1}$ tale che $U_2 U_2^T w = Ca$, da cui

$$Pw = w - Ca.$$

Poichè P proietta sull'ortogonale dello spazio delle colonne di C,

$$C^T Pw = C^T(w - Ca) = 0.$$

Ricordando che C ha rango massimo, si ottiene

$$a = (C^T C)^{-1} C^T w,$$

e si conclude che

$$Pw = w - Ca = w - C(C^T C)^{-1} C^T w.$$

Ecco dunque che il prodotto Pw si può scrivere usando operazioni che coinvolgono solo la matrice C. Infine, è utile osservare che il vettore

$$z \coloneqq (C^T C)^{-1} C^T w$$

è la soluzione del problema ai minimi quadrati

$$\min_{z \in \mathbb{R}^{h-1}} \|Cz - w\|_2.$$

Nel caso di problemi di dimensione moderata, essa può essere calcolata efficientemente in modo diretto via SVD o fattorizzazione QR di C. Nel caso di problemi di grandi dimensioni, la soluzione può essere approssimata con un metodo iterativo come LSQR.

Concludiamo la sezione con un semplice criterio per la scelta del parametro σ. Come è stato illustrato, tale parametro deve soddisfare la condizione

$$\sigma > \lambda_k,$$

così da garantire che i primi $h-1$ autovalori 0 di L_n^p siano traslati sufficientemente lontano dall'origine. Un criterio poco costoso per la scelta di σ si basa sulla Proposizione di Hirsch, per cui

$$|\lambda| \leq \|L_n^v\|,$$

per ogni λ autovalore di L_n^v e per ogni norma matriciale indotta. Allora si può scegliere ad esempio

$$\sigma \coloneqq \|L_n^v\|_1,$$

utilizzando la norma 1 per avere un costo computazione basso.

Capitolo 3

Esperimenti

In questo capitolo mostriamo alcuni risultati sperimentali ottenuti usando gli algoritmi illustrati nei due capitoli precedenti. Con i primi due esperimenti si vuole mettere a confronto su dati reali l'NCut e l'equità dei clusterings ottenuti dai vari algoritmi. Il terzo e ultimo esperimento è volto invece allo studio del loro costo computazionale. Gli esperimenti illustrati, eccetto l'Esperimento 3.2.2, sono stati costruiti a partire da quelli presentati in [Wang et al., 2023].

Gli esperimenti sono stati svolti su MATLAB, versione R2023 Update 5 (9.14.0.2337262), su un laptop Acer con processore Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59GHz, 16 GB di RAM, sistema operativo Microsoft Windows 11.

3.1 Datasets

Gli esperimenti condotti fanno uso dei seguenti *datasets*.

FacebookNet. FacebookNet[1] è un piccolo dataset relativo alla relazione di amicizia su Facebook fra gli studenti di una scuola superiore di Marsiglia, in Francia, nel 2013 [Mastrandrea et al., 2015]. Nell'Esperimento 3.2.1, abbiamo utilizzato un campione estratto da questo dataset costituito da due parti. La prima consiste delle informazioni riguardanti la relazione di amicizia su Facebook tra 155 studenti della scuola: per ogni coppia dei 155 studenti è noto se i due sono amici su Facebook o meno (per alcune coppie di studenti questa informazione è mancante, in tal caso i due studenti si considerano

[1] `http://www.sociopatterns.org/datasets/high-school-contact-and-friendship-networks/`

non-amici); la seconda contiene invece le informazioni relative al sesso degli studenti. Precisamente, dei 155 studenti 70 sono femmine e 85 maschi.

FisherIris. Questo dataset, anch'esso di dimensioni moderate, consiste di 50 campioni di piante di *Iris* per ognuna delle tre specie *Iris setosa*, *Iris virginica* e *Iris versicolor*. Per ogni istanza sono state misurate in centimetri quattro caratteristiche: lunghezza e larghezza del petalo e del sepalo. Il dataset è stato reso pubblico per la prima volta dallo statista Ronald Fisher in un articolo pubblicato nel 1936, *The use of multiple measurements in taxonomic problems*, e raccolto dal botanico Edgar Anderson nella penisola Gaspé, in Canada.

RandomLaplacian. Si tratta di un dataset sintetico costruito in modo random. Fissato $n \in \mathbb{N}$ e $d \in (0,1)$, si genera il dataset costruendo la matrice di adiacenza $W \in \mathbb{R}^{n\times n}$ del grafo di similarità in modo casuale con *desità* pari a d (cioè con circa $d \cdot n^2$ non-zeri) e coefficienti non nulli uniformemente distribuiti nell'intervallo $(0,1)$. Essa è costruita in modo da garantirne la simmetria e piccoli coefficienti non nulli possono essere aggiunti, se necessario, per evitare la presenza di nodi isolati. I gruppi protetti sono anch'essi scelti in modo random: fissato $h \in \mathbb{N}, 1 < h < n$, si crea la group membership matrix $F \in \mathbb{R}^{n\times(h-1)}$ con coefficienti uniformemente distribuiti nell'intervallo $(0,1)$. Si noti che in questo modo la F non rappresenta realmente una partizione del dataset in gruppi protetti (non è della forma (2.9)). Ciò non costituisce un problema perchè il dataset sarà utilizzato soltanto per lo studio del costo computazionale degli algoritmi illustrati. Pertanto, non è essenziale ai fini dell'esperimento che la matrice F sia interpretabile come una matrice indicatrice di una qualche partizione di V.

3.2 Descrizione degli esperimenti e risultati

A seguire sono descritti gli esperimenti condotti sui datasets illustrati e ne sono commentati i risultati.

Esperimento 3.2.1: FacebookNet

Il primo esperimento è stato condotto sul dataset FacebookNet. I dati sono stati rappresentati con un grafo di similarità non pesato $\mathcal{G}(V,W)$ in cui $|V| = 155$

Tabella 3.1: FacebookNet: studio del fairness per i tre algoritmi nel caso $k = 2$.

Quantità	SC	FairSC	gep-FairSC	s-FairSC
$\frac{\lvert V_1 \cap \widehat{C}_1 \rvert}{\lvert \widehat{C}_1 \rvert}$	0,6528	0,5616	0,5616	0,5616
$\frac{\lvert V_1 \cap \widehat{C}_2 \rvert}{\lvert \widehat{C}_2 \rvert}$	0,2771	0,3537	0,3537	0,3537
$\frac{\lvert V_2 \cap \widehat{C}_1 \rvert}{\lvert \widehat{C}_1 \rvert}$	0,3472	0,4384	0,4384	0,4384
$\frac{\lvert V_2 \cap \widehat{C}_2 \rvert}{\lvert \widehat{C}_2 \rvert}$	0,7229	0,6463	0,6463	0,6463

e $W = (w_{ij}) \in \mathbb{R}^{155\times 155}$ ha coefficienti

$$w_{ij} = \begin{cases} 1, & \text{se } v_i \text{ e } v_j \in V \text{ sono amici su Facebook,} \\ 0, & \text{altrimenti.} \end{cases}$$

La suddivisione di V in gruppi protetti è $V = V_1 \sqcup V_2$, dove V_1 contiene i nodi che rappresentano le studentesse femmine e V_2 gli altri, $|V_1| = 70$ e $|V_2| = 85$.

La Tabella 3.1 illustra le quantità relative alla Definizione 2.1.1, a pagina 14, per i raggruppamenti determinati dai vari algoritmi nel caso $k = 2$. In base a tale definizione, poichè

$$\frac{|V_1|}{|V|} = 0.4516 \quad \text{e} \quad \frac{|V_2|}{|V|} = 0.5484,$$

se uno dei quattro clusterings fosse perfettamente equo, allora i primi due e gli ultimi due coefficienti della colonna corrispondente nella Tabella 3.1 dovrebbero essere circa 0,45 e 0,55 rispettivamente. Osserviamo che FairSC, gep-FairSC e s-FairSC determinano clusterings più equi rispetto a SC. Inoltre, notiamo che i risultati ottenuti con i tre algoritmi sono identici. In effetti, FairSC, gep-FairSC e s-FairSC risolvono lo stesso problema rilassato (2.10) per determinare il clustering dei dati, seppur in modi diversi. Per questo, la differenza fra tali algoritmi non sta tanto nella bontà dei clusters (in termini di NCut e fairness) quanto piuttosto nel tempo di esecuzione.

Per misurare l'equità dei raggruppamenti, è utile introdurre la seguente definizione che generalizza l'analoga data in [Chierichetti et al., 2018] nel caso $h = 2$.

Definizione 3.2.1. Dato il clustering dei dati $V = \widehat{C}_1 \sqcup \cdots \sqcup \widehat{C}_k$ e la partizione di V in gruppi $V = V_1 \sqcup \cdots \sqcup V_h$, poniamo

$$\text{balance}\big(\widehat{C}_l\big) \coloneqq \min_{s \neq s' \in \{1,\dots,h\}} \frac{\big|V_s \cap \widehat{C}_l\big|}{\big|V_{s'} \cap \widehat{C}_l\big|} \in [0,1], \tag{3.1}$$

per $l = 1, \dots, k$. Definiamo inoltre il *balance* (o *overall balance*) del clustering come

$$\text{balance}\big(\widehat{C}_1, \dots, \widehat{C}_k\big) \coloneqq \min_{l=1,\dots,k} \text{balance}\big(\widehat{C}_l\big) \in [0,1], \tag{3.2}$$

e l'*avarage balance* come $\frac{1}{k}\sum_{l=1}^{k} \text{balance}\,\big(\widehat{C}_l\big)$.

Si osservi che se il clustering considerato è group fair, allora

$$\frac{\big|V_s \cap \widehat{C}_l\big|}{\big|V_{s'} \cap \widehat{C}_l\big|} = \frac{\big|V_s \cap \widehat{C}_l\big|}{\big|\widehat{C}_l\big|} \frac{\big|\widehat{C}_l\big|}{\big|V_{s'} \cap \widehat{C}_l\big|} = \frac{|V_s|}{|V|}\frac{|V|}{|V_{s'}|} = \frac{|V_s|}{|V_{s'}|},$$

da cui segue immediatamente

$$\text{balance}\big(\widehat{C}_1, \dots, \widehat{C}_k\big) = \min_{s \neq s' \in \{1,\dots,h\}} \frac{|V_s|}{|V_{s'}|}.$$

Inoltre si può dimostrare [Kleindessner et al., 2019] che per ogni clustering vale la disuguaglianza

$$\text{balance}\big(\widehat{C}_1, \dots, \widehat{C}_k\big) \leq \min_{s \neq s' \in \{1,\dots,h\}} \frac{|V_s|}{|V_{s'}|}.$$

Questo suggerisce che un maggiore balance è indice di un clustering più equo secondo la Definizione 2.1.1 di group fairness.

La Figura 3.1 mostra l'NCut e l'avarage balance dei clusterings ottenuti da ciascuno dei quattro algoritmi SC, FairSC, gep-FairSC e s-FairSC al variare del numero di clusters $k = 2, \dots, 6$. Notiamo che FairSC, gep-FairSC e s-FairSC determinano sempre raggruppamenti più equi rispetto ad SC, con avarage balance pressoché identici fra loro. Ciò è in accordo con quanto osservato nel caso $k = 2$, illustrato nel dettaglio in precedenza. È interessante il fatto che, per ogni valore di k, i tre algoritmi costruiscano dei clusterings con medesimo NCut. Ci si aspetterebbe infatti che l'algoritmo SC determini raggruppamenti con NCut più basso rispetto a quelli degli altri tre, che devono soddisfare il vincolo ulteriore di group fairness. Tuttavia, su questo dataset particolare, almeno per i valori di k studiati, la maggiore equità è raggiunta senza sacrifici in termini di NCut.

Nella Figura 3.2, a pagina 39, sono stati rappresentati i clusterings ottenuti da SC (a sinistra) e s-FairSC (a destra) per $k = 2$. Non si sono illustrati quelli

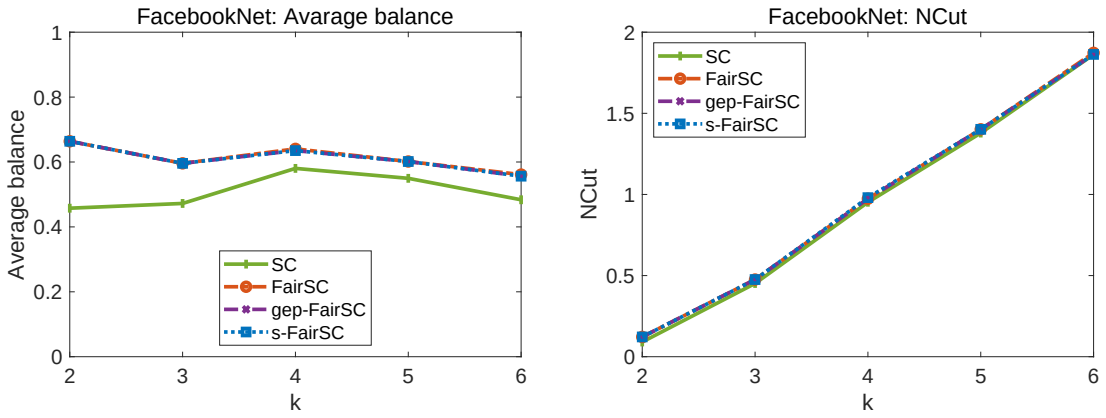


Figura 3.1: FacebookNet: avarage balance e NCut a confronto per SC e s-FairSC.

relativi agli altri due algoritmi perché, come suggeriscono i risultati già visti, essi risultano identici a quello individuato da s-FairSC. In questo modo possiamo vedere più precisamente come influisce sul raggruppamento il vincolo di equità. I colori diversi individuano i due clusters mentre la forma del marker individua i due gruppi protetti. Notiamo che nel passaggio dal clustering determinato da SC a quello rilevato da s-FairSC, 6 studentesse sono "passate" dal primo al secondo cluster e, viceversa, 7 studenti maschi sono passati dal secondo al primo (i due clusters sono stati numerati in accordo con la Tabella 3.1). Dunque si potrebbe dire che, in questo caso, s-FairSC "aggiusta meticolosamente" il clustering, nel senso che dal primo al secondo clustering sono stati cambiati i clusters di appartenenza di pochi nodi in modo tale da ottenere il massimo guadagno in termini di fairness (si ricordi che nel primo cluster del raggruppamento costruito da SC le femmine sono sovrarappresentate). Inoltre, osserviamo che i nodi che cambiano cluster sono tutti piuttosto isolati, non densamente collegati a nessuno dei due clusters. Questo potrebbe giustificare l'uguaglianza in termini di NCut dei clusterings calcolati dai due algoritmi per i vari valori di k: la maggiore equità è raggiunta cambiando il cluster di appartenenza di pochi nodi isolati; il fatto che tali nodi siano isolati fa sì che assegnarli all'altro cluster non comporti un aumento significativo di NCut.

Esperimento 3.2.2: FisherIris

In questo esperimento abbiamo confrontato i risultati ottenuti da SC e s-FairSC sul dataset FischerIris per $k = 3$ e per alcune scelte dei gruppi protetti. I dati in FisherIris sono stati traslati in modo tale che la media aritmetica delle

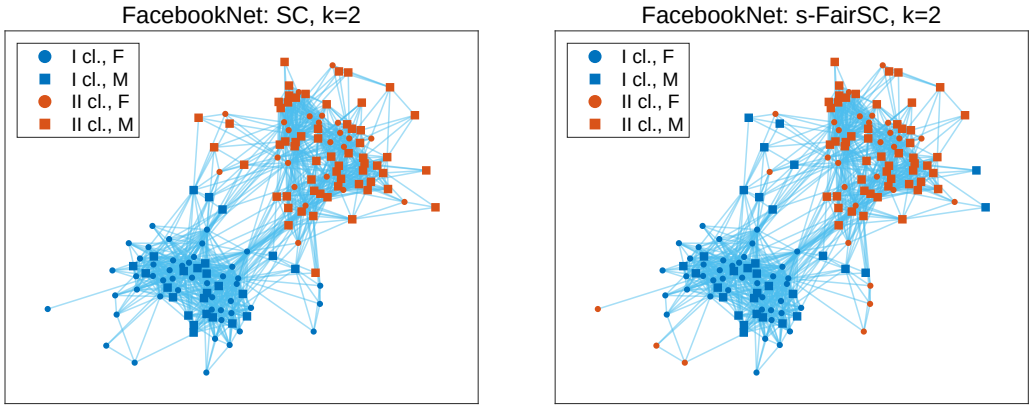


Figura 3.2: FacebookNet: visualizzazione dei clusters per SC e s-FairSC.

osservazioni fosse l'origine $\mathbf{0} \in \mathbb{R}^4$ (questo passaggio è stato fatto solo per semplificare alcuni calcoli, non influenza nella sostanza i risultati).

Nella Figura 3.3 è illustrato il grafo di similarità utilizzato, $\mathcal{G}(V, W)$. In tale grafo, due nodi v_i e v_j sono collegati da un lato se v_j è uno dei 4 nodi più vicini a v_i (v_i escluso) o viceversa per ogni $v_i, v_j \in V$. Questo tipo di grafo è noto in letteratura come *mutual* 4-*nearest neighbors graph* [von Luxburg, 2007]. Il peso di ciascun lato è stato calcolato usando la *funzione di similarità Gaussiana* con distanza euclidea, per cui

$$w_{ij} = \exp\left(-\frac{\|v_i - v_j\|_2^2}{2\sigma^2}\right),$$

con $\sigma = 1$. Per rappresentare il grafo in due dimensioni sono state utilizzate le prime due componenti principali. Notiamo che il grafo ha due componenti connesse.

Per studiare più approfonditamente il risultato, nella Figura 3.4 sono state rappresentate le colonne della matrice H definita in (1.11), che sono degli autovettori relativi ai primi 3 autovalori di L_{rw}. Si ricordi che l'algoritmo SC determina il clustering finale applicando il clustering con k-medie alle righe di H. Entrambe le sottofigure illustrano il `plot` delle colonne di $H = [h_1, h_2, h_3]$. Nella prima i colori evidenziano le tre colonne di H; nella seconda sono state colorate allo stesso modo le righe di H che vengono raggruppate nello stesso cluster da SC. Notiamo che, in accordo con i risultati teorici, i primi due autovettori sono costanti sulle due componenti connesse del grafo. Il terzo autovettore invece è costante soltanto sulla prima componente connessa. Conseguentemente, l'algoritmo non ha difficoltà a rilevare il cluster costituito dalla specie Iris setosa (primi 50 oggetti),

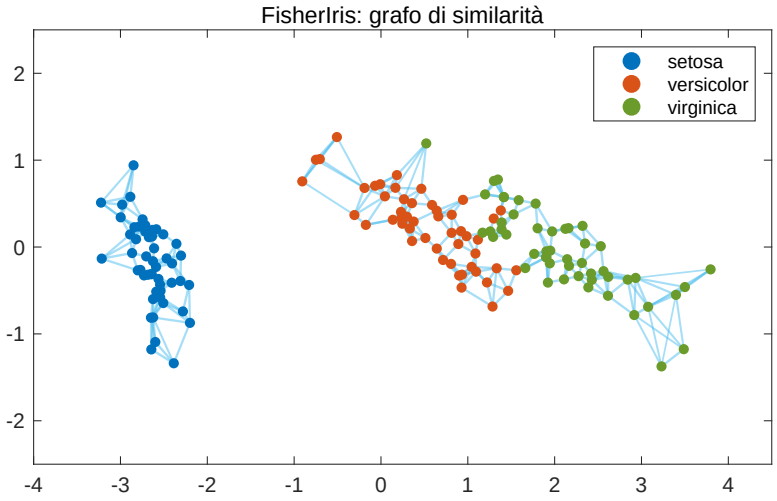


Figura 3.3: FisherIris: grafo di similarità nelle prime due componenti principali.

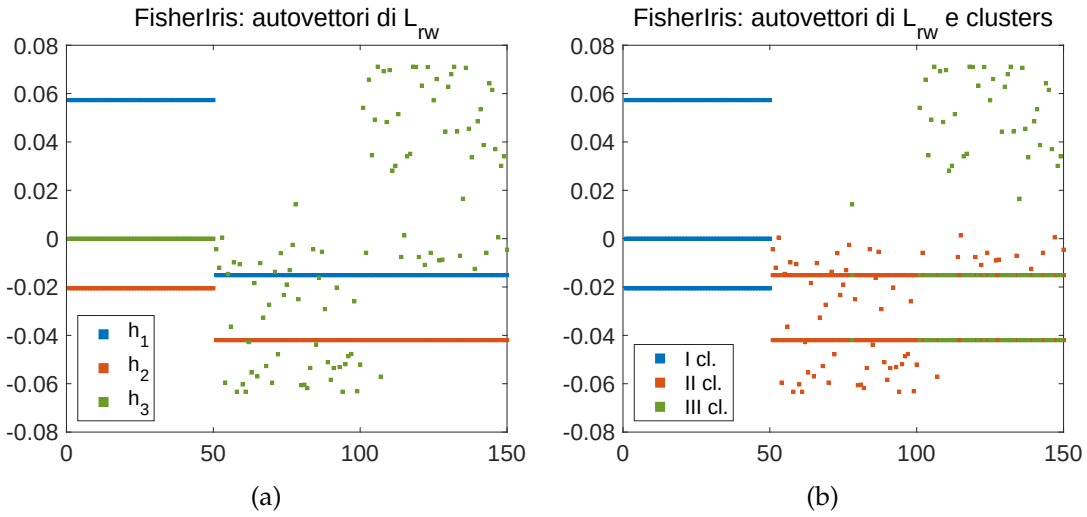


Figura 3.4: FischerIris: autovettori della matrice L_{rw}.

ma mischia alcune istanze delle altre due specie. Nella Figura 3.5 nella pagina successiva è illustrato il clustering così ottenuto con SC. Il colore interno di ogni nodo ne indica la specie (il colore blu corrisponde alla specie Iris setosa, quello arancione a Iris versicolor e quello verde a Iris virginica) mentre il colore del bordo ne indica il cluster di appartenenza. Si osservi che, per quanto il clustering non recuperi esattamente la suddivisione in specie, esso risulta ragionevole nel senso che gli ultimi due clusters dividono l'agglomerato di destra con un taglio poco costoso (cioè con Cut basso).

Abbiamo poi testato l'algoritmo s-FairSC, per certe scelte dei gruppi protetti. Innanzitutto, prendendo tre gruppi protetti costituiti tutti e tre per un terzo da

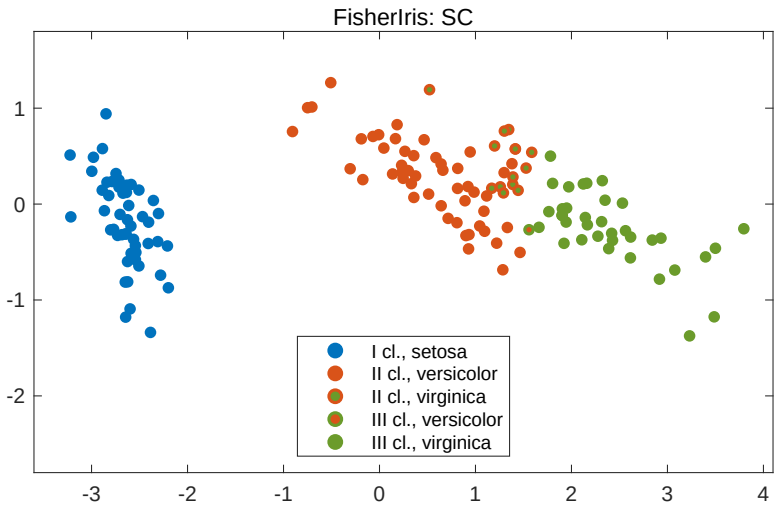
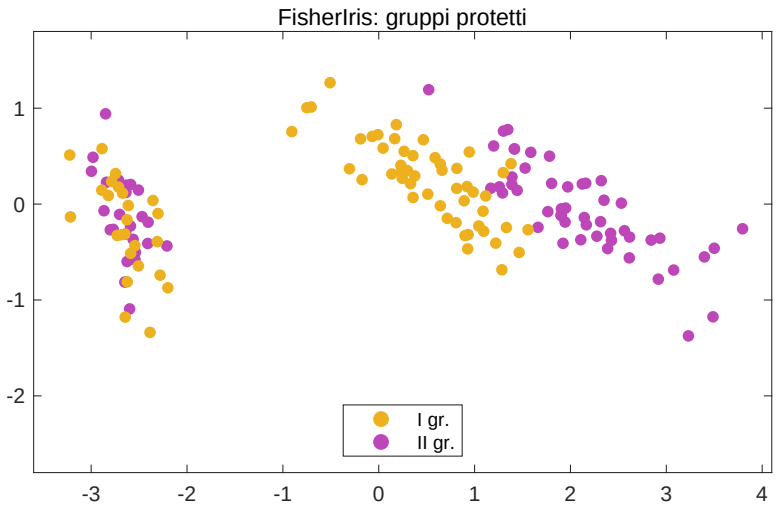
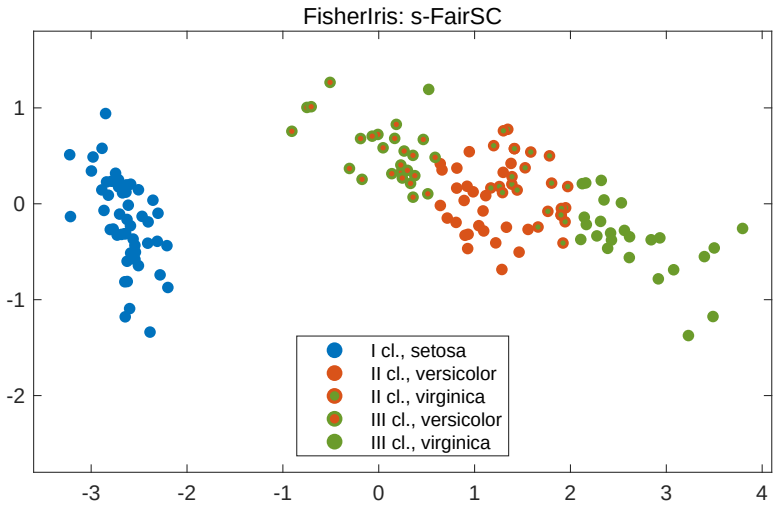


Figura 3.5: FischerIris: clustering con SC

ogni specie, s-FairSC ottiene un clustering quasi identico a quello senza vincolo di group fairness. In effetti, con questa scelta dei gruppi i clusters determinati da SC risultano già equi, ed è quindi ragionevole che s-FairSC recuperi lo stesso clustering. Si è poi provato a prendere come gruppi protetti le tre specie di fiori. Questa scelta, come ci si aspettava, ha portato a risultati di difficile interpretazione dato che la richiesta che le tre specie siano equamente rappresentate in ciascun cluster è fortemente in contrasto con quella di minimizzazione della funzione obiettivo NCut. Per concludere, si sono presi due gruppi protetti come nella Figura 3.6a nella pagina seguente. Il primo gruppo è costituito dall'intera famiglia di Iris versicolor e dal 50% dei fiori Iris setosa (selezinati in modo casuale), mentre il secondo è costituito dagli oggetti rimanenti. Il clustering ottenuto da s-FairSC con questa scelta dei gruppi è rappresentato nella Figura 3.6b. Notiamo che uno dei tre clusters è dato dalla famiglia Iris setosa. Questo non ci sorprende perché essa è perfettamente isolata della parte rimanente del dataset (dunque sceglierla come cluster contribuisce ad un NCut basso) ed è anche ben bilanciata rispetto ai gruppi scelti. Gli altri due clusters sono molto diversi da quelli visti in precedenza. Tuttavia, il risultato è ancora una volta coerente con la discussione teorica: s-FairSC "rinuncia" al clustering con NCut più basso in favore di una maggiore equità. Nella Tabella 3.2 sono riportati l'NCut ed il balance dei clusterings ottenuti dai due algoritmi. Nel passaggio dalla colonna relativa a SC a quella relativa a s-FairSC, notiamo un aumento rilevante dell'NCut, che raddoppia pur rimanendo sotto il 2.5% del massimo valore possibile (che è 3). Tuttavia, osserviamo anche un bilanciamento decisamente migliore, quasi perfetto (si



(a)



(b)

Figura 3.6: FisherIris: gruppi protetti e clustering corrispondente.

ricordi che il balance ha valore massimo unitario).

Esperimento 3.2.3: RandomLaplacian

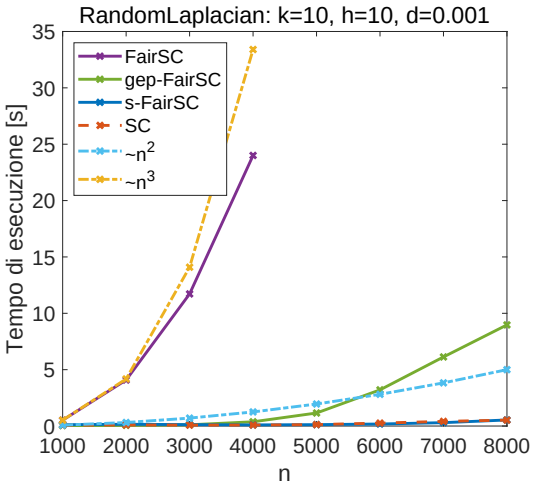
In questo esperimento usiamo il dataset RandomLaplacian per confrontare il costo computazionale degli algoritmi SC, FairSC, gep-FairSC e s-FairSC in termini di tempo. Al variare della dimensione n e della densità d, fissati $k = h = 10$, creiamo il grafo di similarità e la group-membership matrix in modo random come descritto nella sezione 3.1 ed eseguiamo i quattro algoritmi memorizzando i tempi di esecuzione. La scelta di fissare k e h a un valore piccolo è dovuta al fatto che spesso, nelle applicazioni reali, il numero di clusters da determinare ed il

Tabella 3.2: FisherIris: confronto fra SC e s-FairSC

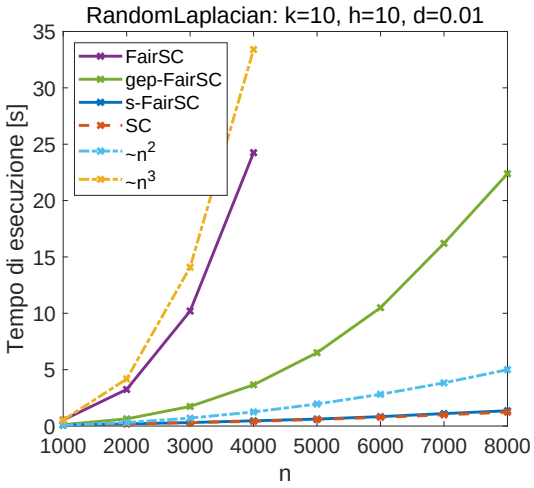
Quantità	SC	s-FairSC
NCut	0,0329	0,0703
balance	0,0277	0,9600

numero di gruppi protetti sono effettivamente molto più piccoli della dimensione del dataset.

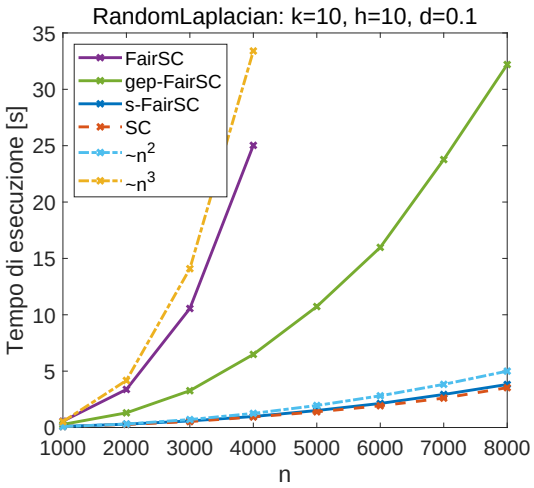
Nella Figura 3.7 sono riportati i risultati. È evidente come FairSC risulti essere l'algoritmo più oneroso, con un costo quasi cubico nella dimensione del dataset. Per via dell'elevato tempo di esecuzione, abbiamo condotto l'esperimento per FairSC solo fino a n al massimo pari a 4000. Mentre gep-FairSC risulta certamente più efficiente di FairSC, rimane comunque più costoso di s-FairSC. Ad esempio, per $n = 4000$ e $d = 0.1$, il tempo di esecuzione di s-FairSC è circa 25 volte inferiore a quello di FairSC e circa 6 volte inferiore a quello di gep-FairSC. Solo per valori di d pari a 0.001 o 0.01 e per piccole dimensioni del dataset, gep-FairSC ha un costo paragonabile a quello di s-FairSC. È interessante notare che s-FairSC ha un costo computazionale simile a quello di SC, nonostante l'implementazione del vincolo di equità. Possiamo anche notare che il tempo di calcolo per FairSC non è significativamente influenzato dalla sparsità di W, a differenza di quanto accade per gli altri tre algoritmi. In effetti, la matrice M di cui si calcano i primi autovettori in FairSC rimane densa anche quando W è sparsa.



(a) Densità $d = 0.001$



(b) Densità $d = 0.01$



(c) Densità $d = 0.1$

Figura 3.7: RandomLaplacian: confronto del costo computazionale degli algoritmi al variare della dimensione del dataset n e della densità della matrice di adiacenza d.

Conclusioni

I tre algoritmi di fair spectral clustering analizzati in questo lavoro, FairSC, gep-FairSC e s-FairSC, implementano efficacemente il vincolo di group fairness, con risultati fra loro del tutto simili, come illustrato nel terzo capitolo. La differenza fra i tre algoritmi riguarda principalmente la complessità computazionale: a differenza dei primi due algoritmi, s-FairSC è adeguato anche per datasets di grandi dimensioni avendo un tempo di esecuzione paragonabile a quello di SC.

Sviluppi futuri della ricerca relativa al fair spectral clustering potrebbero comprendere ad esempio nuovi algoritmi che implementino altre nozioni di equità in modo efficiente. Ad esempio, si potrebbe approfondire una variante meno stringente del vincolo di group fairness illustrato in questo elaborato in cui si permette che la proporzione di ciascun gruppo in ogni cluster vari all'interno di un certo intervallo:

$$\beta_s \leq \frac{|V_s \cap C_l|}{|C_l|} \leq \alpha_s,$$

dove $0 \leq \beta_s < \alpha_s \leq 1$ per ogni $s = 1, \ldots, h$ e $l = 1, \ldots, k$. Altre definizioni di fairness, spesso in contrasto con quella adottata all'interno di questo lavoro, si basano sull'idea di equità a livello individuale piuttosto che di gruppo, per cui si richiede che individui simili rispetto a certi attributi protetti siano trattati nello stesso modo. Dunque un algoritmo che rispetti questo vincolo tenderà a raggruppare nello stesso cluster osservazioni vicine rispetto a una funzione di similarità opportuna, che misura la vicinanza fra osservazioni rispetto ad un attributo sensibile. Anche se è già stato sviluppato un algoritmo di spectral clustering che implementa tale vincolo [Gupta and Dukkipati, 2021], esso risulta computazionalmente oneroso ed è ancora aperto il problema di determinarne una versione scalabile.

Bibliografia

A. Chhabra, K. Masalkovaite, and P. Mohapatra. An Overview of Fairness in Clustering. *IEEE Access*, 9:130698–130720, 2021. doi: 10.1109/access.2021.3114099. URL https://doi.org/10.1109%2Faccess.2021.3114099.

F. Chierichetti, R. Kumar, S. Lattanzi, and S. Vassilvitskii. Fair Clustering Through Fairlets. *CoRR*, abs/1802.05733, 2018. URL http://arxiv.org/abs/1802.05733.

K. A. Cliffe, T. J. Garratt, and A. Spence. Eigenvalues of block matrices arising from problems in fluid mechanics. *SIAM J. Matrix Anal. Appl.*, 15(4):1310–1318, oct 1994. ISSN 0895-4798. doi: 10.1137/S0895479892233230. URL https://doi.org/10.1137/S0895479892233230.

K. Fan. On a Theorem of Weyl Concerning Eigenvalues of Linear Transformations: II. *Proceedings of the National Academy of Sciences of the United States of America*, 36(1):31—35, January 1950. ISSN 0027-8424. doi: 10.1073/pnas.36.1.31. URL https://europepmc.org/articles/PMC1063126.

S. Gupta and A. Dukkipati. Protecting Individual Interests across Clusters: Spectral Clustering with Guarantees. *CoRR*, abs/2105.03714, 2021. URL https://arxiv.org/abs/2105.03714.

M. Kleindessner, S. Samadi, P. Awasthi, and J. Morgenstern. Guarantees for Spectral Clustering with Fairness Constraints. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3458–3467. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/kleindessner19b.html.

R. Mastrandrea, J. Fournet, and A. Barrat. Contact Patterns in a High School: A Comparison between Data Collected Using Wearable Sensors, Contact Diaries

and Friendship Surveys. *PLOS ONE*, 10(9):1–26, 09 2015. doi: 10.1371/journal.pone.0136497. URL https://doi.org/10.1371/journal.pone.0136497.

J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, aug 2000. ISSN 0162-8828. doi: 10.1109/34.868688. URL https://doi.org/10.1109/34.868688.

U. von Luxburg. A Tutorial on Spectral Clustering. *Statistics and Computing*, 17 (4):395–416, Dec 2007. ISSN 1573-1375. doi: 10.1007/s11222-007-9033-z. URL https://doi.org/10.1007/s11222-007-9033-z.

J. Wang, D. Lu, I. Davidson, and Z. Bai. Scalable Spectral Clustering with Group Fairness Constraints. In F. Ruiz, J. Dy, and J.-W. van de Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 6613–6629. PMLR, 25–27 Apr 2023. URL https://proceedings.mlr.press/v206/wang23h.html.