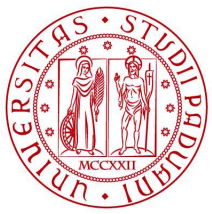

Università degli Studi di Padova

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Laurea Triennale in Ingegneria dell'Informazione

Studio di un algoritmo di streaming per k-median clustering

Relatore
Andrea Alberto Pietracaprina, Prof.

Laureando
Tommaso Galligioni

Anno Accademico
2022/2023

29 Settembre 2023

Abstract

Questa tesi presenta un'analisi dell'algoritmo di streaming Partial Least Squares Plus (PLS+). Ci concentriamo in particolare sul k-median problem in spazi metrici generali, nel quale cerchiamo di trovare un insieme di k centri, ognuno delle quali raggruppa un certo numero di punti pesati dello stream iniziale, per ridurre al minimo la somma delle distanze da ciascun punto nel set di dati al suo centro più vicino.

Viene utilizzato al suo interno anche l'algoritmo Online Facility Location (OFL) sia come controllore sul costo sia per scegliere se un punto può far parte di una facility già creata o deve essere il centro di una nuova facility.

L'obiettivo della tesi è di fare un'analisi sia teorica che sperimentale delle proprietà dell'output di PLS+.

Contents

1 Introduzione

Il clustering, dato un insieme di punti appartenenti ad uno spazio, con una nozione di distanza tra i punti, è un insieme di tecniche di analisi dei dati con il compito di raggruppare nello stesso gruppo (cluster) punti "vicini" per determinate caratteristiche. La distanza comprende anche una nozione di somiglianza: più i punti sono vicini, più sono simili tra di loro. Un problema di clustering viene solitamente definito richiedendo che i cluster ottimizzino una data funzione obiettivo soddisfando alcune proprietà. Sono stati definiti numerosi problemi di clustering, impiegati in vari scopi nel corso degli anni.

D'altronde il clustering non è un algoritmo specifico, ma può essere visto come un processo iterativo che presenta prove e fallimenti prima di trovare il giusto compromesso richiesto. Si basa infatti sul multi-objective optimization, una tecnica che dato un insieme di funzioni da ottimizzare tutte assieme, sacrifica alcune funzioni per valorizzarne altre (trade off) per ottenere il risultato migliore possibile.

Un'importante classe di tecniche di clustering, usata anche all'interno dell'algoritmo PLS+, è il clustering center-based, nel quale cerchiamo di suddividere un insieme di dati di input in gruppi, in modo tale che la somiglianza tra gli elementi nello stesso gruppo viene definita dalla vicinanza degli elementi al centro di quell'insieme.

1.1 Contributi della tesi

Questa tesi si occupa di analizzare un modo per costruire un coreset dato uno streaming di punti in input, attraverso il problema di k-median clustering. Il coreset in questo caso è un insieme pesato che rappresenta un campione dell'insieme di dati di input X, e che è molto più piccolo di X stesso, ma è allo stesso tempo capace di riassumere in maniera sufficientemente precisa l'insieme di partenza.

Per fare questa analisi, abbiamo preso spunto dal lavoro di Luca Badin [2] sul medesimo argomento, cercando di rendere più chiara l'analisi teorica dell'algoritmo di PLS+ [3], un algoritmo interessante perchè si basa sul parallellismo che c'è tra i problemi di k-median clustering e quelli di facility location. Infatti, la presenza dell'algoritmo di OFL all'interno di PLS+, è presente una proprietà che mette in relazione il costo ottenuto con le facility scelte nel problema di facility location con il costo ottimo del problema di k-median.

PLS+ lavora a fasi, diventando sempre più accurato ogni volta che ne comincia

una nuova, che succede quando il numero di facility è troppo elevato, oppure il costo totale per l'inserimento dei punti in una facility è troppo grande. Infatti all'inizio di ogni fase i parametri di ingresso vengono aggiornati per renderli più vicini alla soluzione richiesta.

Inoltre la tesi punta a valutare sperimentalmente il ruolo di alcuni parametri di ingresso dell'algoritmo stesso, cercando di ottimizzare maggiormente il limite di costo per la creazione del coreset trovato nell'analisi teorica precedente. La creazione del dataset di partenza è sempre stata presa dalla tesi di Luca Badin.

1.2 Struttura della tesi

In questa tesi nel capitolo 2 viene esposto e presentato l'algoritmo di streaming OFL per il problema di facility location [5], enunciando, senza prova, alcune sue proprietà importanti, propedeutico per l'analisi e per lo pseudocodice di PLS+.

Nel capitolo 3 invece viene analizzato a fondo lo pseudocodice dell'algoritmo principale della tesi, il PLS+. Successivamente, nel capitolo 4 viene implementato l'algoritmo e vengono mostrate le osservazioni fatte al variare dei parametri di partenza.

2 Nozioni preliminari

Uno **spazio metrico** è una coppia ordinata (M, d) nella quale M rappresenta un insieme, mentre $d(\cdot, \cdot)$ è la funziona distanza che soddisfa le seguenti proprietà per ogni coppia $x, y \in X$:

- $d(x, y) > 0$, tranne se $x = y$, in quel caso $\delta(x, y) = 0$;
- è simmetrica: $d(x, y) = d(y, x)$;
- soddisfa la disuguaglianza triangolare: $d(x, y) + d(y, z) \geq d(x, z)$;

Il **k-median clustering** è un metodo che mira a partizionare n punti iniziali in k gruppi (cluster), nel quale ciascun punto appartiene al cluster con la mediana più vicina.

Definizione 1. *Il problema del k-median clustering prevede che, dati un insieme X di punti da uno spazio metrico (M, d) e un intero k, con $1 \leq k < |X|$, si determini un insieme di punti $K \subset X$, con $|K| = k$, che minimizzi la funzione obiettivo*

$$v_X(K) = \sum_{x \in X} \min_{y \in K} d(x, y)$$

In tutta questa tesi indicheremo con $v^*_{X,k}$ il costo del clustering su X con un insieme di centri ottimo per il k-median clustering problem, che corrisponde all'insieme di centri K per cui il costo della funzione diventa minimo:

$$v^*_{X,k} = \min_{K \subset X : |K| = k} \sum_{x \in X} d(x, K)$$

D'ora in poi, nella tesi, quando la presenza di X e/o k è chiara dal contesto, X e k verranno omessi nella notazione di $v^*_{X,k}$.

Il problema di k-median clustering si può vedere inoltre come un caso specifico dell'uncapacitated facility location problem. Per **facility location** si intendono quei problemi nei quali, dato un insieme arbitrario di punti, si deve selezionare un sottoinsieme di essi che li rappresenti (facility).

Definizione 2. *Il problema di facility location, prevede che, dati uno insieme X di punti da uno spazio metrico (M, d) , un costo di apertura di una nuova facility f*

si determini un insieme di facility $F \subset X$ da aprire al fine di minimizzare il costo totale

$$f|F| + \sum_{x \in X} d(x, F)$$

Ogni punto paga un costo di servizio per essere connesso alla facility già aperta più vicina, che è uguale alla distanza d che c'è tra il punto stesso e la facility. Alternativamente viene aperta una nuova facility, che viene rappresentata da quel punto, che paga un costo equivalente al costo fissato per l'apertura di una nuova facility, f, che supponiamo costante. I punti dell'insieme di partenza X inoltre possono essere pesati ma se è così, e un punto ha peso $w(x)$, lo possiamo considerare con un insieme di $w(x)$ punti consecutivi di peso unitario.

D'ora in poi considereremo $c = \sum_{x \in X} d(x, F)$ il costo di servizio complessivo.

Possiamo osservare che c è analogo alla funzione di costo per un problema di k-median, ovvero v: la somma di tutte le distanze tra ogni punto $x \in X$ e la facility più vicina. D'altra parte però, a differenza del problema di k-median clustering nel quale è presente il vincolo $|K| = k$, questo nel problema di facility location è sostituito dalla somma dei costi per creare nuove facility, $f|F|$.

Entrambi i problemi, general metric k-median clustering e facility location, sono problemi computazionalmente difficili, nel senso che non si conoscono algoritmi polinomiali in grado di calcolare la soluzione ottima, ed è pure molto difficile che esistano questi algoritmi. Si ricorre quindi, per motivi di efficienza, a soluzioni approssimate calcolabili in tempo polinomiale.

Inoltre, in questa tesi, studieremo algoritmi sviluppati nel modello computazionale di **data streaming**. Questo modello viene utilizzato quando devono essere elaborati dataset molto grandi la cui taglia supera quella della memoria RAM, oppure flussi continui (e potenzialmente illimitati) di dati. Nel primo caso, il dataset è mantenuto in un supporto di memoria lento, ad esempio su disco, e, tramite una scansione sequenziale, viene trasferito alla memoria RAM come flusso continuo di dati. Il modello architetturale dello streaming è formato da 3 elementi principali, ovvero lo stream di dati, il processore e la working memory con una capacità limitata. I compiti svolti sono quelli di rispondere alle richieste che arrivano al processore, e aggiornare continuamente ogni volta che arrivano elementi ciò che deve essere modificato, come si vede nella figura 2.1.

Negli algoritmi di data streaming vogliamo che vengano ottimizzate 3 metriche fondamentali:

- la dimensione s della memoria di lavoro con l'obiettivo che sia $s \ll |\sum|$
- il numero di passate che si vogliono fare sull'input, con l'obiettivo che se ne faccia solamente una
- il tempo di elaborazione per un singolo elemento, con l'obiettivo che sia $O(1)$

Per questo motivo l'analisi dei dati deve essere fatta immediatamente, utilizzando un limitato spazio di memoria, senza salvare i dati per una successiva lavorazione

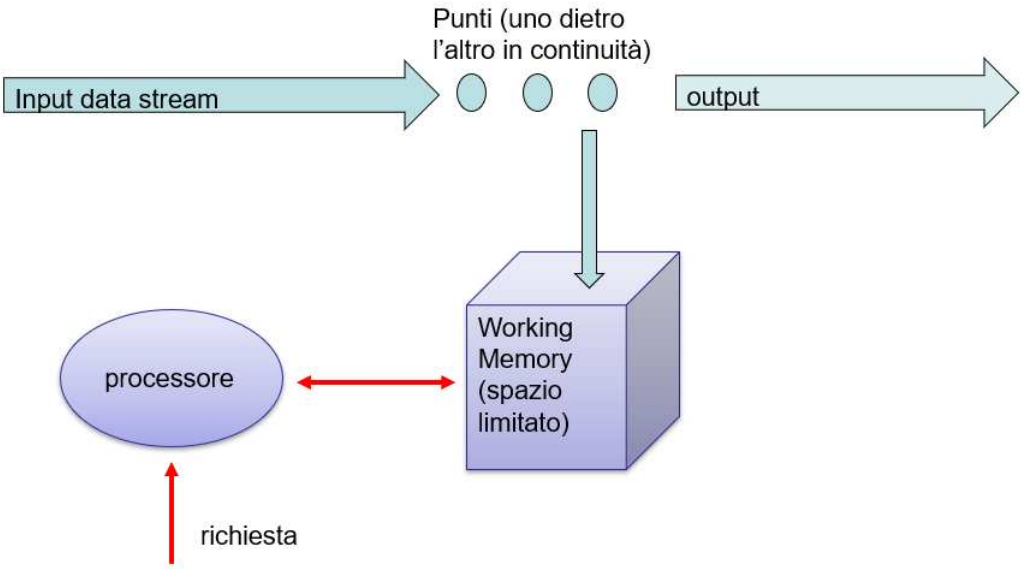

Figure 2.1: Modello architetturale dello streaming

con il dataset completo e senza considerare i dati che devono ancora essere analizzati mentre si processa un qualsiasi dato.

Nel caso si risolvano problemi di ottimizzazione, la soluzione trovata sarà sicuramente una approssimazione di quella ottima (che sarebbe possibile solo con uno spazio lineare e non polilogaritmico). [4]

3 Online Facility Location

In questo capitolo vedremo la variante online dell'algoritmo della facility location (OFL) [5]. Questa si differenzia dalla versione originaria, perchè richiede che ciascun punto deve essere assegnato a una facility appena arriva, senza aspettare i punti successivi e senza modificare il suo assegnamento più tardi con l'arrivo degli altri punti. I punti infatti arrivano uno alla volta continuamente e attraverso l'algoritmo diventano o una facility nuova o un punto che fa parte di una facility già creata, modificandone il peso e il costo totale di servizio. L'algoritmo non fa altro rendendo le decisioni prese immutabili.

3.1 Algoritmo OFL

L'algoritmo lavora nel modo seguente. Ogni volta che un nuovo punto x arriva, si calcola la distanza $d(x, y)$ con y che è la facility più vicina al punto stesso. A questo punto, con probabilità $\min\{d/f, 1\}$, viene aperta una nuova facility, pagando il costo associato f, altrimenti, pagando il costo di servizio d, il punto x, viene connesso alla facility già aperta più vicina.

Quindi, se $f \leq d$ apriremo sicuramente una nuova facility, altrimenti la probabilità di creare una nuova facility è proporzionale al rapporto d/f.

Il costo totale di un problema di facility location è $c + f|F|$, cioè la somma tra il costo di servizio per inserire un punto in una facility, e l'insieme dei costi per creare le facility.

Principalmente siamo interessati a c, e chiameremo c il costo della soluzione F dell'algoritmo OFL.

Algorithm 1 Online Facility Location. f è il costo per creare una nuova facility

Input: Stream X di elementi da un metric space (M, d), ciascun $x \in X$ associato a un peso $w(x)$
Output: insieme di facility F
$c \leftarrow 0$ ▷ costo del servizio
$F \leftarrow \emptyset$ ▷ Insieme delle facility
while ci sono ancora punti nello stream **do**
    $x \leftarrow$ prossimo punto
    $y \leftarrow \arg\min_{y \in F} \delta(x, y)$
    $p \leftarrow \text{RANDOM}([0, 1])$
    if $p \leq \min\{1, w(x) \cdot \delta(x, y)/f\}$ **then**
        $F \leftarrow F \cup x$ ▷ Apri nuova struttura
    else
        $c \leftarrow c + w(x) \cdot \delta(x, y)$ ▷ Paga il costo del servizio
        $w(y) \leftarrow w(y) + w(x)$ ▷ Incrementa il peso della facility
    end if
end while
return F

3.2 Analisi

Notiamo, con anche il prossimo teorema, che scegliendo in maniera appropriata il costo f di creazione di una facility, il numero totale di facility aperte non è troppo superiore rispetto al valore k che si troverebbe utilizzando solamente un algoritmo di k-median sull'insieme di partenza. Inoltre, notiamo un'importante proprietà dell'algoritmo OFL, ovvero che mette in relazione il costo ottenuto con le facility scelte con il costo ottimo del problema di k-median, importante per l'analisi successiva dell'algoritmo PLS+.

Teorema 1. *Supponiamo che l'OFL abbia come costo di creazione di una facility il seguente:*

$$f = \frac{L}{k(1 + \log n)}, \qquad L \leq v*$$

dove L è un limite inferiore del costo per fare un cluster su X ottendendo la soluzione ottima con il k-median clustering, e n rappresenta il peso totale di tutti i punti nello stream. Allora, con alta probabilità [1], si ottiene che:

$$c \leq \left(3 + \frac{2e}{e-1}\right) v^*, \qquad |F| \leq 7k(1 + \log n)\frac{v^*}{L}$$

[1] con alta probabilità si intende probabilità $\geq 1 - \frac{1}{n}$, dove n è la taglia di input

4 Algoritmo Partial Least Squares Plus (PLS+)

4.1 Descrizione e pseudocodice

PLS+ è un algoritmo basato sull'Online Facility Location, che, preso uno stream di punti appartenente a un insieme X, restituisce un insieme di facility F di cardinalità $O(k \log n)$. Ogni punto $x \in X$ è associato a una facility (e una facility è associata a se stessa), e tale associazione definisce una partizione di X in $|F|$ cluster. Ogni facility è inoltre restituita con un peso che rappresenta la taglia del cluster associato a quella facility. E inoltre ogni facility è restituita con un peso che rappresenta la taglia del cluster associato a quella facility. Nell'algoritmo questo insieme di facility è scelto in maniera tale che il cluster sull'insieme X dei punti sia ottimo, sia per numero di facility (non devono essere troppe), sia per il costo di inserire nuovi punti all'interno di una facility (non deve essere troppo alto).

A differenza dell'OFL però, che risolve il problema di Facility Location, PLS+ punta invece a risolvere il problema di k-median clustering. Tuttavia la soluzione restituita, ovvero l'insieme di punti F, ha taglia che può essere superiore a k (infatti il set di facility è di cardinalità $O(k \log n)$. A questo punto si aprono due possibilità: o si tollera un numero di centri superiore a k, valutando però il costo rispetto alla soluzione ottima per k; oppure si considera l'insieme F come un coreset pesato rappresentativo di tutto lo stream, e si estrae da esso una soluzione di taglia k eseguendo su F uno degli algoritmi di approssimazione sequenziali noti per k-median.

L'algoritmo PLS+ si basa su una ipotesi sul costo che serve per aprire una nuova facility. Se questa ipotesi non rispetta i parametri visti precedentemente sul costo e sul numero di facilities, si fa partire una nuova fase e si ricomincia il processo con una nuova ipotesi migliorata.

Definizione 1. *Per fase si intende una singola esecuzione dell'algoritmo OFL con un valore fissato di L (una costante che definisce un limite superiore al costo del clustering indotto dalle facility correnti), da cui dipende il costo per l'apertura di ogni facility.*

Ovviamente, quando si cambia l'ipotesi, cambia anche la fase, e tutte le facility vengono inserite nuovamente nello stream, con i loro pesi dati dall'insieme dei pesi

dei punti associati a una specifica facility. Le facility, in realtà, più che all'interno dello stream stesso, possono essere inserite tutte all'interno di una pila che lavora parallelamente allo stream. Infatti vengono messe all'interno di questa pila e finché la pila non si svuota, nella fase successiva vengono estratte tutte prima di prendere elementi ancora non pesati all'interno dello stream stesso.

Algorithm 2 PLS+. β, γ sono costanti fornite dall'utente

Input: Stream X di elementi da un metric space (M, d), ciascun $x \in X$ associato a un peso $w(x)$
Output: insieme F di facility
$L_1 \leftarrow 1$ ▷ Guess iniziale di L
$i \leftarrow 1$ ▷ numero di fasi
while solution not found **do**
    $c \leftarrow 0$ ▷ costo totale della fase corrente
    $f \leftarrow \frac{L_i}{k(1+\log n)}$ ▷ costo per creare una facility
    $F \leftarrow \emptyset$ ▷ insieme di facility
    while there are points still in the stream **do** ▷ OFL: inizia
        $x \leftarrow$ next point
        $y \leftarrow \arg\min_{y \in F} \delta(x, y)$
        $p \leftarrow \text{RANDOM}([0, 1])$
        if $p \leq \min\left\{1, \frac{w(x) \cdot \delta(x, y)}{f}\right\}$ **then**
            $F \leftarrow F \cup \{x\}$
        else
            $c \leftarrow c + w(x) \cdot \delta(x, y)$
            $w(y) \leftarrow w(y) + w(x)$
        end if ▷ OFL: finisce
        if $c > \gamma L_i$ or $|F| > (\gamma - 1)k(1 + \log n)$ **then**
            Break and raise flag ▷ e bisogna iniziare una nuova fase
        end if
    end while
    if flag raised **then** ▷ non sono rispettati i vincoli
        Push facilities in F on to stream
        $L_{i+1} \leftarrow \beta L_i$
        $i \leftarrow i + 1$
    else ▷ il while si conclude perchè non ci sono più punti
        solution found
    end if
end while
return F

4.2 Analisi

Il più importante risultato di questo algoritmo è il seguente:

Teorema 4. *Esistono delle opportune costanti β e γ tali che, con alta probabilità, PLS+ completa la fase finale con una soluzione F il cui costo è*

$$\sum_{x \in X} \delta(x, F) \leq \alpha v_X^*,$$

dove α è $\dfrac{\beta\gamma}{\beta - 1}$ e quindi F è un insieme limitato di cardinalità $O(k \log n)$ per il problema di k-median clustering sull'istanza di partenza X.

Per dimostrarlo, partiamo enunciando e dimostrando alcuni lemmi che saranno utili in seguito.

Innanzitutto, bisogna vedere se prima o poi una fase si conclude senza causare l'uscita improvvisa dal "while" con il "break". Quindi si deve capire se, quando viene creata una nuova fase, ci siano effettivamente dei progressi nel processare nuovi punti. Se il problema avuto nella fase precedente fosse stato il numero di facility create troppo elevato, si può notare che, se viene aumentato il costo di creare una nuova facility, il prodotto tra il peso e la distanza tra punto e facility, diviso il costo stesso, inevitabilmente diminuirà e quindi diminuirà la probabilità di creare una nuova facility.
Quindi ci si trova di fronte a due casi.

Caso 1:Il costo è ancora troppo basso e tutte le facility nella pila vengono ricreate nello stesso numero rispetto a prima. Non ci sono stati progressi, ma sicuramente il numero delle facility non è aumentato rispetto a prima, e faremo partire un'altra fase, sempre con il costo ancora aumentato esponenzialmente, finché inevitabilmente, con un tempo che non deve essere troppo grande, ci troveremo nel caso 2.

Caso 2: Arriveremo al punto che, presa la prima facility, f_1, e presa una qualsiasi altra facility f_2, possiamo notare che, aumentando il costo della creazione di una nuova facility di un fattore $\beta > 1$ a ogni fase, prima o poi avremo che il costo per creare una nuova facility è troppo più alto rispetto alla distanza tra f_1 e f_2, e quindi ci troveremo a non aprire f_2 e associarla a f_1, diminuendo in questo modo il numero di facility.

Ripetendo questo processo molte volte ci troveremo al punto che il limite sul numero di facility sarà soddisfatto anche dopo aver processato tutti i punti nello stream.
Se il problema invece è il costo di inserimento degli elementi ci viene in aiuto, inserendo un limite superiore per il costo di servizio c dopo che vengono proces-

sate tutte le vecchie facility nuovamente, che ci permette sicuramente di processare almeno un altro punto nuovo dello stream, il seguente lemma.

Lemma 1. *All'inizio di una nuova fase, l'algoritmo PLS+ riesce a fare un clustering con successo di tutte le facility presenti la fase precedente, senza uscire dal ciclo while con il break, prima di esaminare i nuovi punti presi dallo stream. Inoltre, il costo, una volta che tutte le precedenti facility sono state processate, è sicuramente:*

$$c < \gamma L_i$$

Dimostrazione: Osserviamo innanzitutto che per ogni punto si deve pagare al massimo f. Infatti se si apre una nuova facility il costo è esattamente quello, mentre se inserisco un nuovo elemento questo costerà $w(x) \cdot \delta(x, y)$ che è sicuramente un numero minore a f.

Infatti se per assurdo fosse strettamente maggiore, il rapporto $(w(x) \cdot \delta(x, y))/f$ sarebbe superiore a 1, e quindi, per come è stato scritto l'algoritmo, ci troveremo obbligati a creare una nuova facility.

Inoltre, come visto sempre nel codice, ci sono al massimo $(\gamma - 1)k(1 + \log n) + 1$ facility già presenti nella fase precedente.

Esse hanno come costo complessivo:

$$c \leq f(y-1)k(1+\log n) = \frac{L_i}{k(1+\log n)}(\gamma-1)k(1+\log n) = (\gamma-1)L_i$$

Da qui quindi si può dedurre che $c \leq (\gamma - 1)L_i < \gamma L_i$, e il lemma è dimostrato.

□

Definizione 2.: *Consideriamo $x \in X$ un qualsiasi punto del dataset di partenza. Si dice che una facility $y \in F$* ***rappresenta*** *x nella Fase l se y è la facility associata a x oppure è la facilty associata al rappresentante di x nella fase $l - 1$.*

*Il punto x è anche detto "**original point**".*

Interessante notare quindi che, una volta che x viene rappresentato in una certa fase da una facility f_1, avrà sempre un rappresentante anche nelle fasi successive. Analogamente se x è una facility, esso rappresenta sé stesso.

Ora enunciamo il lemma conseguente la cui dimostrazione si trova in [2] [3].

Lemma 2. *Consideriamo un sottoinsieme arbitrario P dei punti, pesati o ancora da leggere, presenti nello stream all'inizio di una Fase i. Allora, il costo di un k-median clustering su P è al massimo:*

$$v_P^* \leq v_X^* + \frac{\gamma}{\beta - 1} L_i$$

Analizziamo ora l'aumento del costo della creazione di una nuova facility nel tempo. In ogni fase, il limite sul costo cresce di un fattore β. Potrebbe comportare un aumento eccessivo, superando di troppo il valore del costo ottimo che l'algoritmo punra a raggiungere.

Fortunatamente la probabilità che questo accada è quasi nulla nell'algoritmo PLS+. Andiamo a dimostrare in seguito quanto detto.

Definizione 3. *Chiamiamo i, l'ultima fase in cui $L_i < v_X^*$,* ***fase critica****.*

Osserviamo innanzitutto che non è detto che si arrivi per forza a questa fase. Infatti una soluzione può essere trovata precedentemente, come mostra il lemma seguente.

Lemma 3. *Prendiamo*

$$c_{OFL} = \left(3 + \frac{2e}{e-1}\right) \backsimeq 6.164, \qquad k_{OFL} = 7$$

che sono i fattori costanti che compaiono nei limiti superiori rispettivamente al costo c e al numero di facility $|F|$, determinati nell'analisi in alta probabilità dell'algoritmo OFL. Quindi, considerando delle opportune costanti beta e gamma l'algoritmo PLS+ termina con alta probabilità prima o durante la fase critica.

Dimostrazione. Sia i l'indice della fase critica e X_i l'insieme di tutti i punti, sia quelli pesati, sia quelli non letti, che sono presenti nello stream all'inizio della fase i.

Osserviamo che, dato che questa è la fase critica, $v_X^* \leq \beta L_i$.

Possiamo applicare il Lemma 2 a X_i:

$$v_{X_i}^* \leq v_X^* + \gamma \left(\frac{1}{\beta - 1}\right) L_i \leq \left(\beta + \gamma \frac{1}{\beta - 1}\right) L_i$$

In questa fase, l'algoritmo OFL in esecuzione fornisce una soluzione tale che, con alta probabilita,

$$c \leq c_{OFL} v_{X_i}^*, \qquad |F| \leq k_{OFL}(1 + \log n) v_{X_i}^* / L_i$$

e questo lo può maggiorare solo con $(\beta + \gamma/(\beta - 1))k_{OFL}(1 + \log n)$.

A questo punto, vogliamo che i limiti imposti dall'algoritmo PLS+ sul costo e sul numero di nuove facility siano soddisfatti, per non fare partire una nuova fase. Quindi:

$$c_{OFL}v^*_{X_i} \leq \gamma L_i, \qquad (\beta + \gamma/(\beta - 1))k_{OFL}(1 + \log n) \leq (\gamma - 1)k(1 + \log n)$$

Ora vediamo che è sufficiente scegliere γ e β in modo tale che

$$\gamma \geq 1 + k_{OFL}(\beta + \gamma/(\beta - 1)).$$

E quindi può concludere che il lemma segue per opportuni valori di β e γ costanti.

Se l'OFL all'interno del PLS+ con alta probabilità termina con successo nella fase critica, allora anche il PLS+ termina con successo con la stessa probabilità.

□

Abbiamo adesso tutti gli elementi per dimostrare il Teorema 4, obiettivo ultimo dell'analisi, che fornisce un limite superiore al costo totale dell'algoritmo per qualisasi insieme di partenza X.

Dimostrazione (Teorema 4). Supponiamo di essere nella fase i, e supponiamo che sia la fase finale. Consideriamo ogni punto $x \in X$ dello stream, e consideriamo j la fase in cui x è stato clusterizzato la prima volta.

Consideriamo $y_j, y_{j+1}, \ldots, y_{i-1}, y_i$ come le rappresentazioni di x nelle fasi $j, j + 1, \ldots, i - 1, i$.

Se perciò facciamo un clustering di X con F, il costo c_x dato da x, è esattamente la distanza dalla facility che lo rappresenta in questo momento, $\delta(x, y_i)$. Per la diseguaglianza triangolare, c_x è limitata superiormente dal costo della connessione tra x e la facility y_j che lo rappresentava originariamente, sommata alla somma di tutte le distanze tra le due facility che rappresentavano x in fasi successive. Ovvero

$$c_x = \delta(x, y_i) \leq \delta(x, y_j) + \sum_{l=1}^{i-j} \delta(y_{i-l}, y_{i-l+1})$$

Quindi in altre parole, il costo finale di un qualsiasi punto $x \in X$ dopo aver fatto un clustering in base a F è superiormente limitato dalla somma di tutti i costi di servizio passati, durante tutte le fasi dell'algoritmo PLS+.

Successivamente, se consideriamo tutto lo stream X, sommando c_x per tutti i punti $x \in X$ otterremo la somma di tutti i costi di servizio o di apertura di una nuova facility che sono stati pagati da ogni punti in ogni fase dalla prima all'ultima, che corrisponde alla fase i, per come l'avevamo definita ad inizio dimostrazione.

Ora possiamo utilizzare il Lemma 1, e quindi, presa una generica fase $l < i$,

$$c_l \leq \gamma L_l = \gamma L_i \frac{1}{\beta^{i-l}}$$

Considerando questa quantità in tutte le fasi $l = 1, \ldots, l$, e sapendo per il Lemma 3 che il PLS+ termina a una fase dove $L_i \leq v^*$ con alta probabilità, otteniamo i seguenti limiti sul costo di servizio totale della soluzione dell'algoritmo di PLS+:

$$\begin{aligned} c &\leq \sum_{l=1}^{i} c_l \\ &\leq \gamma L_i + \gamma L_{i-1} + \gamma L_{i-2} + ... + \gamma L1 \\ &\leq \gamma L_i \sum_{l=0}^{i-1} \frac{1}{\beta^l} \\ &\leq \frac{\beta\gamma}{\beta - 1} v^* \end{aligned}$$

□

5 Valutazioni sperimentali

In questo capitolo vengono fatte alcune analisi sperimentali per confrontare le prestazioni ottenute in pratica dall'algoritmo PLS+ con quelle previste dall'analisi teorica descritta nel capitolo precedente.

Nel corso dello studio del PLS+ abbiamo trovato un limite sul costo per ottenere un certo numero di facility, che rappresentano in maniera sufficientemente approssimata un dataset di partenza. in questi esperimenti vediamo come funziona in pratica l'algoritmo dato che l'analisi teorica è fatta worst-case e i limiti di conseguenza non sono così stretti.

Infatti vedremo come questi limiti risultanti dall'analisi del PLS+ in linea teorica, nella pratica possono essere notevolmente migliorati, al variare dei valori γ e β, e usando migliori approssimazioni, e diminuendo la cardinalità del dataset di partenza. Alla fine di tutto inoltre, per avere reali confronti tra l'efficienza dei vari valori di γ e β viene utilizzato all'interno di questa analisi sperimentale un algoritmo di k-median, sulle varie facility pesate che vengono accumulate dall'algoritmo di PLS+, per avere un numero finale omogeneo di facility (k).

Per l'analisi sperimentale il codice sia di PLS+ che dell'algoritmo di k-median è stato implementato in Java. Inoltre per il test è stato utilizzata una piattaforma AMD Ryzen 5 3500U with Radeon Vega Mobile Gfx che funziona a 2.10 GHz, 8GB di memoria RAM.

5.1 K-means++

Per la fase sperimentale di questa tesi, è stato utilizzato una versione modificata dell'algoritmo di k-means++ [1], noto algoritmo tipicamente utilizzato per produrre un insieme iniziali di centri per l'algoritmo di Lloyd per il problema del clustering k-means. Infatti abbiamo utilizzato il k-median, che si differenzia dal k-means++ per il fatto che per calcolare il costo considera la distanza tra i punti e non la distanza al quadrato. Lo stesso algoritmo però ha buon rapporto costo-prestazioni anche per il problema di k-median. Questo algoritmo a partire da un dataset qualsiasi X di punti non pesati, ritorna un insieme $K \subset X$, con $1 \leq k < |X|$, e con $|K| = k$. I punti di questo insieme K sono i centri che rappresentano il dataset di partenza.

Nella tesi abbiamo utilizzato però una versiona pesata dell'algoritmo. Infatti le facility che escono dall'algoritmo di PLS+ rappresentano un insieme di punti, tutti con il proprio peso. Per questo motivo, all'interno dell'algoritmo, la distanza tra due punti non è solamente la distanza $d(x, y)$, con $y \in K$, ma sarà

$$d_{pes}(x, y) = w(x)d(x, y)$$

dove $w(x)$ è il peso del punto x.

5.2 Datasets

Per testare il nostro codice, abbiamo utilizzato gli stessi dataset utilizzati nella propria tesi da Luca Badin [2]. Dati $n \in \mathbb{N}^+$ punti da generare, $k \geq 1$ centri, $D \in \mathbb{N}^+$ dimensioni, $r \in \mathbb{R}^+$ raggio, e $s \in \mathbb{N}^+$ un fattore di distribuzione, avremo:

- k centri cher sono generati uniformemente e indipendentemente uno dall'altro, all'interno di una circonferenza di raggio r con centro nell'origine;
- n punti, ognuno dei quali generato dopo aver scelto un centro in maniera casuale e uniforme, e con dimensioni generate traslando il centro scelto di una quantità casuale distribuita come una normale con media zero e varianza unitaria, moltiplicata per r/s. In pratica, il punto si troverà all'interno di una circonferenza di raggio r/s dal suo centro, applicando un Rumore Gaussiano a ciascuna dimensione in modo indipendente.

Tutte le distanze presenti per gli esperimenti sono considerate distanze Euclidee. I vari parametri utilizzati per creare un dataset invece sono i seguenti:

n	k	D	r	s
100,000	4	2	100	4
200,000	4	2	100	4
500,000	4	2	100	4
1,000,000	4	2	100	4

5.3 Risultati

Innanzitutto vediamo cosa comportano i vari cambiamenti di β e γ nell'algoritmo PLS+. Aumentando β il costo di creazione di una facility in ogni nuova fase cresce con una derivata maggiore, e di conseguenza diminuisce la probabilità di creare una facility nuova, ma rende anche il costo di servizio più alto, dato che metterò in una singola facility punti più lontani tra di loro. Aumentando γ invece rendiamo più ampi i limiti di costo di servizio totale e sul numero di facility, facilitando così la terminazione dell'algoritmo, nonostante un valore più alto del costo di servizio.

Le tabelle sono state fatte con $n = 100000$, perché non cambia con il numero di punti l'andamento crescente o calante del costo e del numero di facility. Inoltre i risultati sono tutti una media di 3 valutazioni differenti.

5.3.1 PLS+ con β fissato

Tenendo fissato $\beta = 5$ proviamo a fare variare γ. Notiamo subito che con l'aumentare di γ aumenta anche il numero di facility, come pensavamo 5.2. Infatti il limite sul numero massimo di facility possibili da creare prima di terminare la fase corrente è maggiore con l'aumentare del valore di γ, e quindi è più probabile che l'algoritmo finisca in una fase precedente. Inoltre interessante notare come il numero di facility create varia in maniera notevole quando l'algoritmo riesce a terminare l'analisi sul dataset con una fase in meno (si nota bene anche dal grafico del numero di facility). Anche questo ce lo aspettavamo, dato che il costo per aprire una facility ogni volta che aumenta la fase, viene moltiplicato di un fattore β, e quindi è più probabile che ci siano meno facility create quando si comincia una nuova fase successiva, come è stato già visto nell'analisi teorica.

Il costo di servizio invece si comporta in maniera diversa. Infatti diminuisce con l'aumentare delle facility 5.3. Anche questo comportamento è quello che ci aspettavamo, infatti, con l'aumentare delle facility, è molto più probabile per un punto trovare una facility più vicina, abbassando così il costo di servizion c. C'è però un caso particolare. Infatti quando sono molto vicino alla diminuzione del numero di fasi da parte dell'algoritmo (con $\gamma = 51$ PLS+ si risolve in 6 fasi anzichè 7) il costo è più basso rispetto a quello che trovo con un valore superiore di γ, e conseguentemente un valore superiore di facility, ad esempio con $\gamma = 80$. 5.1

Questo fenomeno però è plausibile accada, perché nella settima fase (nel caso di $\gamma = 50$ ci troveremo molti meno punti da analizzare, (non ho più tutto il dataset, ma i punti rimanenti del dataset + le facility già pesate nelle fasi precedenti), e quindi meno punti su cui calcolare il costo di servizio.

5.3.2 PLS+ con γ fissato

Proviamo ora a far variare β fissando $\gamma = 80$. Notiamo subito che come in precedenza, all'aumentare del numero di facility diminuisce il costo. Invece all'aumentare di β il comportamento del numero di facility non è lineare così come quando aumentava il valore di γ. Con il cambio di fase, infatti, il numero di facility aumenta, mentre nella stessa fase, con l'aumentare di β, $|F|$ diminuisce. Anche questo avremmo potuto aspettarcelo, perchè, se restiamo nella stessa fase, con un β più piccolo la probabilità di creare una nuova facility aumenta, dato che f, che è direttamente proporzionale a β, diminuisce.

Anche qua, come visto precedentemente, può succedere che per valori vicino al cambio di fase non venga rispettata la monotonia del crescere del costo c al diminuire del numero di facility.

γ	fasi	$\lvert F\rvert$	c
2	10	34	788,445
5	9	104	350,948
8	8	286	311,005
12	8	289	251,759
15	8	294	204,232
20	7	835	191,682
35	7	867	132,908
45	7	882	103,203
50	7	894	90,364
80	6	2,475	98,966
130	6	2,534	67,339

Figure 5.1: Risultati di PLS+ fissato β

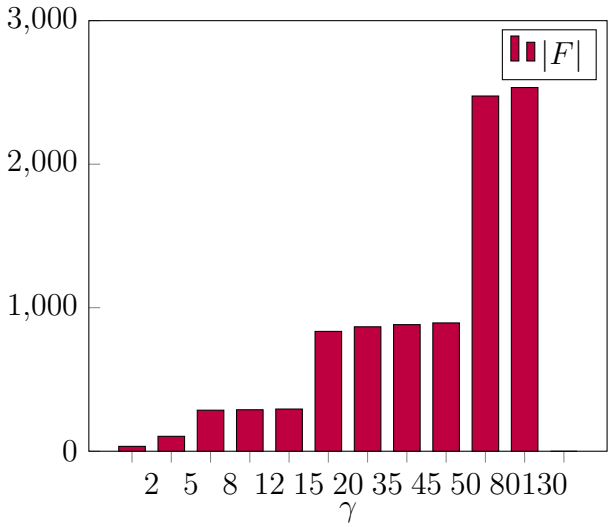

Figure 5.2: Numero di facility create con PLS+ al variare di γ

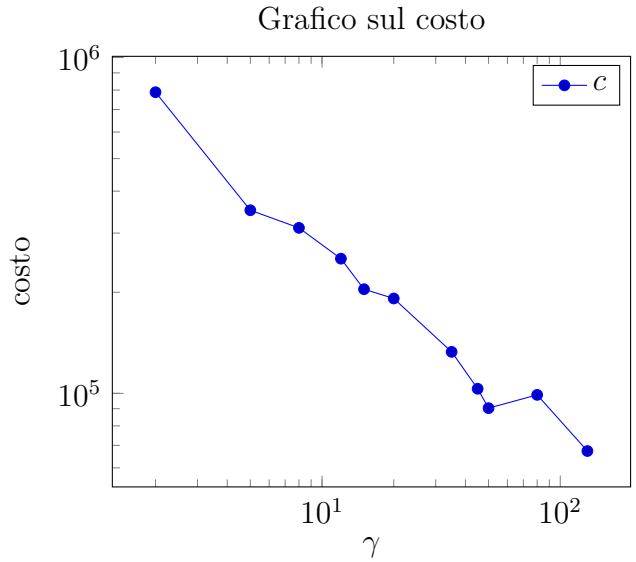

Figure 5.3: Costo di servizio al variare di γ. La scala del grafico è logaritmica.

β	fasi	$\lvert F \rvert$	c
2	12	3,055	39,734
5	6	2,475	98,966
8	5	2,095	123,508
15	4	2,344	141,039
20	4	1,348	179,894
30	4	615	252,906
50	3	2,895	136,501
100	3	1,178	222,312

Figure 5.4: Risultati di PLS+ fissato γ

γ	fasi	k	c
2	10	20	1,028,892
5	9	20	1,050,548
8	8	20	1,099,528
12	8	20	1,105,683
15	8	20	1,100,243
20	7	20	1,143,056
35	7	20	1,083,445
45	7	20	1,085,928
50	7	20	1,090,074
80	6	20	1,110,946
130	6	20	1,118,255

Figure 5.5: Costo finale fissato β

5.3.3 K-median dopo PLS+ e confonto con k-median puro

Ora proviamo a uniformare il numero di facility attraverso un algoritmo di k-median in modo da ottenere un insieme finale di k facility $F' = \{f'_1, f'_2, ..., f'_k\}$. A questo punto il costo finale sarà

$$c_{final} = \sum_{x \in X} \min_i d(x, f'_i)$$

Teniamo sempre $\beta = 5$ e facciamo variare γ come in precedenza. Notiamo valori di costo complessivo finali abbastanza alti. Questo era prevedibile, infatti i punti complessivi finali sono solo 20, quindi maggior probabilità di avere un centro lontano e quindi di aumentare il costo. Inoltre questi costi sono anche molto simili tra di loro. Probabilmente perchè, essendo l'insieme di facility uscenti dall'algoritmo PLS+ una rappresentazione realistica del dataset di partenza, e successivamente riducendo questo dataset a una base comune di k centri, anche i costi complessivi si assomigliano tra di loro.

Infatti ricordiamo che il costo di PLS+, che viene fuori dall'algoritmo di OFL, per il Teorema 1 3.2, è strettamente legato al costo ottimo del k-median sullo stesso dataset di partenza.

Vediamo ora come varia il costo complessivo con il variare di β fissato $\gamma = 80$. Notiamo il comportamento molto simile a quello nel quale a variare è il parametro γ, e possiamo avanzare le stesse conclusioni.

β	fasi	k	c
2	12	20	1,149,835
5	6	20	1,126,539
8	5	20	1,114,597
15	4	20	1,103,692
20	4	20	1,102,429
30	4	20	1,149,623
50	3	20	1,080,342
100	3	20	1,048,039

Figure 5.6: Costo finale fissato γ

In conclusione, proviamo a vedere se realmente c'è una corrispondenza tra i risultati ottenuti, usando PLS+ e k-median, e l'uso del solo k-median sullo stesso dataset di partenza con gli stessi parametri. E effettivamente, il valore ottimo stimato del costo di un k-median clustering, che chiameremo $\tilde{v}^*$, è molto simile ai valori ottenuti nelle tabelle 5.55.6. Infatti, facendo una media tra 20 risultati ottenuti tramite l'algoritmo k-median, il costo ottimo stimato risulta

$$\tilde{v}^* = 1,050,462$$

6 Conclusioni

In questa tesi abbiamo analizzato delle tecniche per poter arrivare, partendo da un insieme di punti facenti parte di uno spazio metrico arbitrario, a una soluzione ai problemi di k-median e facility location. Questa soluzione non è ottima, dato che i problemi di general metric k-median e facility location, sono problemi computazionalmente difficili, ma è comunque una soluzione approssimata calcolabile in tempo polinomiale. Per farlo abbiamo utilizzato l'algoritmo PLS+, con al suo interno l'algoritmo OFL, e un algoritmo di k-median. Abbiamo inoltre adottato come modello di calcolo il modello di data streaming, assumendo quindi che l'input arrivi come uno flusso continuo e che non possa essere salvato tutto in memoria.

Inoltre l'algoritmo di OFL, e quindi di conseguenza anche PLS+, possedeva l'importante proprietà che mette in relazione il costo ottenuto con le facility scelte dall'algoritmo per rappresentare l'insieme di partenza, con il costo ottimo del problema di k-median, scelte opportunamente alcune costanti, β e γ, che rappresentano rispettivamente il fattore moltiplicativo del costo di apertura di una facility in ogni fase, e il fattore moltiplicativo dei limiti sul numero di facility e sul costo per ogni fase di PLS.

Nella fase sperimentale, abbiamo fatto variare β e γ, e abbiamo portato l'insieme di facility, grazie all'algoritmo di k-median, a un numero fissato, per poter studiare senza solamente i limiti abbastanza deboli dell'analisi teorica fatta precedentemente, il comportamento del costo totale del processo.

L'obiettivo finale sarebbe poter riuscire a conoscere il costo di un k-clustering sul dataset di partenza in maniera quanto più vicina al valore corretto, per poter capire in linea pratica quanto e in che modo potrebbero essere limitati i valori di γ e β per ottenere la resa massima dall'algoritmo di PLS+.

Un tema di ricerca futura che può aiutare al miglioramento dell'algoritmo di PLS+ in fase sperimentale, può essere il comportamento anomalo del costo per quei valori di γ, quando è fissato β, critici perchè rendono l'algoritmo sul punto di fare una fase in più o una fase in meno, come evidenziato in 5.3.1.

7 Bibliography

[1] David Arthur and Sergei Vassilvitskii. K-means++ the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, 2007.

[2] Luca Badin. Streaming algorithms for center-based clustering in general metrics. 12 2022.

[3] Vladimir Braverman, Adam Meyerson, Rafail Ostrovsky, Alan Roytman, Michael Shindler, and Brian Tagiku. Streaming k-means on well-clusterable data. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 26–40. SIAM, 2011.

[4] Camil Demetrescu and Irene Finocchi. Algorithms for data streams. *Handbook of Applied Algorithms: Solving Scientific, Engineering and Practical Problems*, pages 241–269, 2008.

[5] Adam Meyerson. Online facility location. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 426–431. IEEE, 2001.