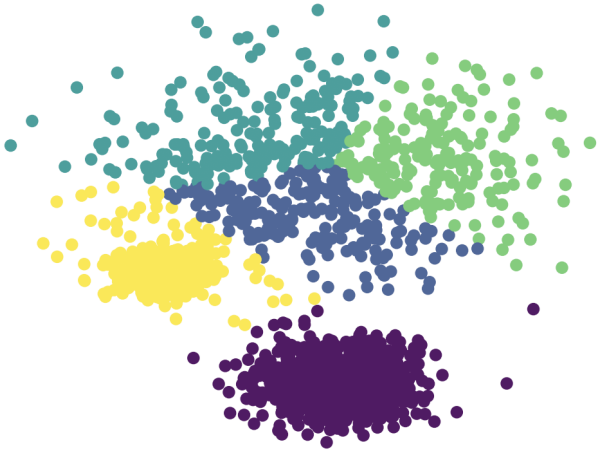
Projecte Algorísmia



Víctor Pla Sanchís

Grup PROJ.16

Facultat d'Informàtica de Barcelona

Universitat Politècnica de Barcelona

22 de març de 2024

Índex

1 Introducció

En aquest treball iniciarem explicant el problema a resoldre i definirem els algorismes implementats. Seguidament detallarem els algorismes i estudiarem la correctesa dels mateixos com també el seu cost computacional i espaial. Al final del projecte extreure'm conclusions respecte la qualitat dels mateixos amb alguns experiments.

1.1 Context

Algorísmia (A) és una assignatura obligatòria de la Facultat d'Informàtica de Barcelona (FIB) de la branca de computació. En A s'estudia diferents metodologies algorísmiques per resoldre certs problemes. S'apliquen tècniques de dividir i vèncer, programació dinàmica o problemes de fluxos com d'altres. En el projecte de A hi implementarem certs algorismes per resoldre un problema en concret, finalment hi estudiarem la qualitat i eficàcia dels algorismes implementats i experimentarem els resultats obtinguts.

En el grup PROJ.16 hi som els alumnes: Luca Ursache, Daniel Sanchez i Víctor Pla. Tots tres alumnes de l'assignatura A. Tot i ser un projecte pensat per realitzar-se amb 4 persones, el nostre grup està format per 3 alumnes. A més a més, donat que els alumnes Luca Ursache i Daniel Sanchez s'han mantingut absents a realitzar la feina, el treball s'ha realitzat de manera completament solitària per l'alumne Víctor Pla Sanchís.

El projecte s'ha dut a terme durant les primeres quatre setmanes de Març del 2024 i han hagut més de 125h de feina, aproximem que unes 25h han sigut temps dedicat a: autoaprenentatge, investigació i bibliografia. Altres 50h han sigut destinades a la creació de la documentació i l'informe final (com també de tot el que fa relació a l'informe com: els experiments, les figures, creació de *scripts* per agilitzar els experiments i temps d'aprenentatge de LaTeX). Finalment 50h han sigut destinades a la creació del codi font i dels algorismes com també de totes les resolucions de qualsevol error que s'hagi donat durant el procés dels mateixos (ja que Python ha sigut el llenguatge de programació usat i no és amb el que tenim més coneixement, tot i això, creiem que ha sigut un encert donada la facilitat de la visualització dels resultats i experiments que ens ofereix).

1.2 Problema

El problema en el que treballarem és l'anomenat *Clustering* que consisteix en l'agrupació de dades en K subconjunts. Cada subconjunt tindrà un cert nombre d'elements amb certa similitud entre ells, donat que els elements poden ser de qualsevol tipus com ara: perfils d'una xarxa social, punts geogràfics, gens o cèl·lules d'un persona... Cal doncs definir una funció de similitud f entre dos elements del conjunt de dades A.

$$f : A \times A \to \mathbb{R}$$

Donat que el nostre conjunt $A \subseteq \mathbb{R}$ podrem doncs usar la funció de distancia euclidiana per definir la similitud de dos elements de A tal que:

$$d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$
$$d(a, b) = \sqrt{(a_0 - b_0)^2 + (a_1 - b_1)^2 + \cdots + (a_n - b_n)^2}$$

Tot i això, el problema de *Clustering* es pot resoldre per a qualsevol tipus de conjunt A. Simplement cal definir correctament alguna funció f que defineixi la similitud de dos elements tal i com nosaltres necessitem. Per exemple, per definir la similitud de dues paraules es pot usar la funció de distancia de Levenshtein que calcula el numero d'operacions mínimes per transformar una paraula en l'altre (una operació pot ser definida com: inserció o eliminació d'un caràcter i substitució d'algun caràcter). Una mala definició de la funció de similitud pot fer que el nostre algorisme convergeixi en solucions que no ens interessin. Simplement puntualitzar que la millor funció de similitud és aquella que defineixi de millor manera la relació que volem recalcar de similitud en les nostres dades i per tant l'elecció d'ella és totalment flexible.

Iniciarem doncs el projecte implementant dues versions del famós algorisme *k-means*. Seguidament hi implementarem alguns algorismes per poder computar la qualitat de les solucions obtingudes i finalment usarem aquests qualificadors per experimentar l'eficiència de *k-means* respecte uns *datasets* donats.

Donat un *k-means*, com bé ens van explicar en l'enunciat del problema, l'algorisme es pot dividir en tres parts molt diferents:

1. Inicialització dels *centroides*
2. Computació dels clústers
3. Actualització dels *centroides*

Per tant, implementarem dos algorismes que realitzin els tres passos essencials de manera diferent, en la següent secció els especificarem.

2 Algorismes de clustering

En aquesta secció veurem els dos algorismes de *clustering* implementats (Random k-means i Weight k-means++) i finalment farem una pinzellada per entendre les diferències més pràctiques que els distingeix. Tot i analitzar les principals diferencies en aquesta secció, més endavant, en els experiments entendrem de manera més objectiva la diferencia dels seus comportaments.

2.1 Random k-means i Weight k-means++

Pel que fa el nostre primer algorisme de *k-means* hem decidit implementar cada pas de la següent manera:

Random k-means

1. Inicialitzem K centroides aleatoris corresponents a K punts diferents del *dataset*.

$$C = \{c_0, c_1, \cdots, c_k\}, \forall c_i \in c, c_i \in \textit{dataset}$$

2. Assignarem el punt $x_i \in \textit{dataset}$ al *centroide* més proper (més similar).
3. Calcularem la mitjana de tots els punts del clúster per calcular el nou *centroide* del clúster.

$$c_i = \frac{1}{|c_i|} \sum_{x_i \in c_i} d(x_i, c_i)$$

A aquest algorisme l'anomenarem **Random k-means**. Per altre banda, modificarem els passos 1 i 3 per generar un nou algorisme que l'anomenarem **PlusPlus k-means**, o més simple, **Weight k-means++**. Aquest últim l'anomenarem d'aquesta manera donat que utilitza una metodologia d'inicialització de *centroides* anomenada *k-means++* i usa una ponderació de pesos per actualitzar els *centroides* en cada iteració.

Weight k-means++

1. Inicialitzem 1 *centroide* $c_0 \in \textit{dataset}$ de manera aleatòria. Seguidament calcularem la probabilitat P d'un punt $x_i \in \textit{dataset}$ respecta al quadrat de la distància amb el *centroide* més proper. Finalment escollirem de manera seqüencial els $K-1$ *centroides* de manera aleatòria amb una probabilitat d'escollir el punt x_i de $P(x_i)$.

$$P(x_i) = \frac{\min(d(x_i, c_i)^2)}{\sum d(x_i, c_i)^2}, \forall c_i \in C$$

2. Assignarem el punt $x_i \in$ *dataset* al *centroide* més proper (més similar).
3. Calcularem el nou *centroide* com la mitjana de la suma ponderada a partir d'una funció de pes w.

$$w : A \times C \to \mathbb{R}$$

$$w(x_i, c_i) = \begin{cases} 0 & \text{si } c_i = x_i \\ \frac{1}{d(c_i, c_i)^2} & \text{si } c_i \neq x_i \end{cases}$$

$$c_i = \frac{\sum d(x_i, c_i) w(x_i)}{\sum w(x_i)}, \forall x_i \in A$$

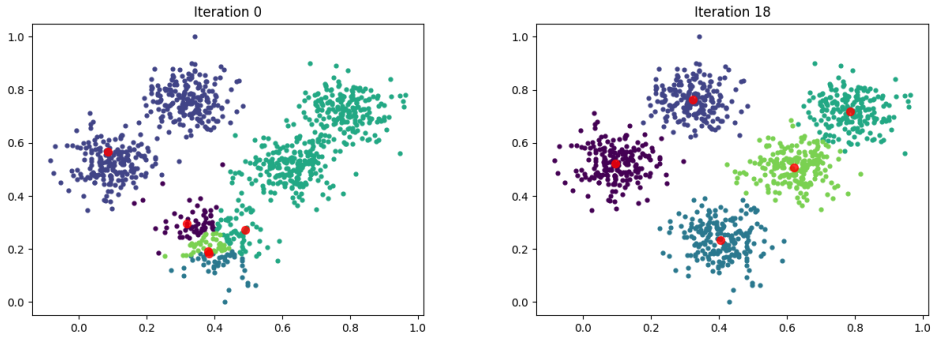
Per acabar, cal també definir la condició de convergència dels algorismes. En aquest cas, per a ambdós algorismes és el mateix. Hem escollit que l'algorisme convergeixi quan tots els clusters no hagin generat cap variació entre dues iteracions. És la condició de convergència menys permissiva donat que és la solució més optima. Existeixen altres condicions però generalment per falta de temps no ens hi hem aturat gaire en aquesta part del projecte, és cert que aquesta iteració genera totes les iteracions possibles i que podríem evitar-ne d'algunes per convergir abans en una solució suficientment bona.

2.2 Diferències entre els dos algorismes

Ja hem definit formalment les diferències entre Random k-means i Weight k-means++, tot i això aquesta secció es centrara en veure visualment i a la pràctica la diferència entre els dos algorismes.

Pel que fa la inicialització de *centroides* Weight k-means++ permet una millor distribució donat que evita potencialment (donat que no deixa de ser una selecció aleatòria però en aquest cas ponderada amb una probabilitat) que dos *centroides* comencin sent molt propers i caiguin en un mateix subconjunt de punts i no generin la convergència desitjada o triguin més en fer-ho.

Si ens hi fixem en la Figura 1, veiem que Random Weight k-means++ genera una inicialització de *centroides* bastant naïf donat que hi ha quatre *centroides* molt pròxims, inclús, a la figura costa de diferenciar dos *centroides* de lo pròxims que hi son. Per altre banda, si observem la Figura 2 podem veure com el algorisme Weight k-means++ genera una distribució de



(a) Inicialització de *centroides* (b) Convergència de Random k-means

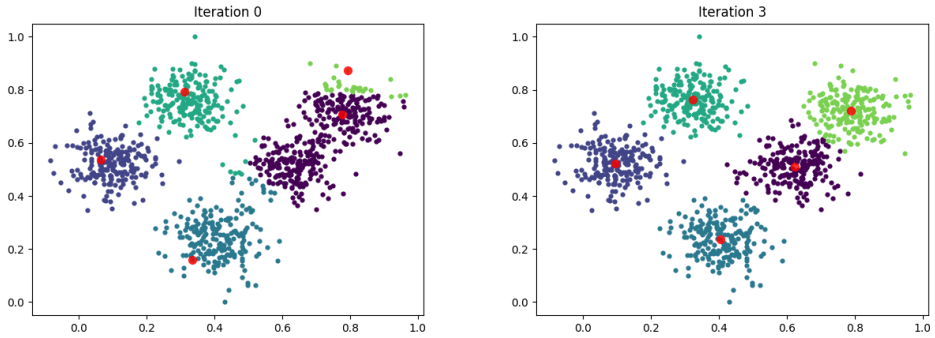
Figura 1: Exemple d'execució de l'algorisme Random k-means

centroides inicials molt mes dispersa, ambdós algorismes convergeixen de manera raonable tot i això la dispersada inicialització de Weight k-means++ ho permet fer tant sols en tres iteracions mentres que el Random k-means ho fa en divuit. Molt important recalcar que en aquest cas, tot hi haver mencionat els dos algorismes, simplement hem canviat la inicialització dels *centroides* i no hem variat el pas 3 (ambdós els hem executat amb una actualització de *centroides* idèntica) per tal de trobar una diferencia objectiva entre les diferents maneres de inicialitzar els *centroides*.

Per tant, aquesta secció ens permet formular una hipòtesi respecte els experiments sobre el temps de convergència dels algorismes Weight k-means++ i Random k-means. Inicialment diem que potencialment el temps de convergència de Weight k-means++ serà més curt respecte el del Random k-means.

Finalment, veurem visualment la diferència del calcul de *centroides* en mig d'una execució dels dos algorismes. Per fer-ho, hem agafat un moment específic on els *centroides* tenen una posició concreta i hem executat els dos algorismes.

Com veiem a la Figura 3, donat que Weight k-means++ pondera aquells punts més propers al *centroide* amb un valor més alt i aquells més llunyans amb un valor més baix, els *centroides* calculats són sempre més propers al anterior *centroide* comparat amb el algorisme Random k-means. Per tant, podriem formular la següent hipòtesi: calcular els nous *centroides* amb la ponderació de pesos de Weight k-means++ generara més iteracions potencialment per trobar la convergència de la solució comparat amb Random k-means. Tot i això pensem que és una hipotèsi bastant fluixa donat que un *learning rate* elevat també pot generar convergències molt lentes (quan el *centroide* varia molt respecte a l'anterior i oscil·la amb la solució de manera que li costa apropar-s'hi, com el cas de la Figura 4).



(a) Inicialització de *centroides* (b) Convergència de Weight k-means++

Figura 2: Exemple d'execució de l'algorisme Weight k-means++

3 Mesures de qualitat

En aquesta secció veurem dues mesures de qualitat implementades per analitzar les solucions dels algorismes implementats. Per una banda hem implementat una mesura de qualitat interna: *Silhouette*, i una mesures de qualitat externa: *Rand Index*.

3.1 Silhouette

Donada una solució de l'algorisme de *k-means*, *Silhouette* és una mesura de qualitat interna donat que calcula per cada punt d'un clúster quant de correcte és que hi pertanyi al cluster assignat i no a un d'altre. Donat un punt qualsevol p i el seu clúster C, es calcula la mitjana $\bar{C}$ de les distàncies de p amb tots els altres punts de C. Seguidament es calcula també la mitjana $\bar{C}'$ de les distancies de p amb qualsevol punt d'un clúster veí C' i finalment *Silhouette* representa la diferència normalitzada d'aquestes dues mitjanes. Sigui doncs $S(p)$ el valor de *Silhouette* per a un punt $p \in C$ on C' és el clúster veí de p ($C \cap C' = \emptyset$).

$$\bar{C}(p) = \frac{\sum_{q \in C, q \neq p} d(p,q)}{|C| - 1}, p \in C$$

$$\bar{C}'(p) = \frac{\sum_{q \in C'} d(p,q)}{|C'|}, p \notin C'$$

$$S(p) = \frac{\bar{C}'(p) - \bar{C}(p)}{\max(\bar{C}(p), \bar{C}'(p))}$$

Si estudiem l'expressió $S(p)$, veiem que si el valor de $S(p) = 0$, aleshores el punt p és igual de

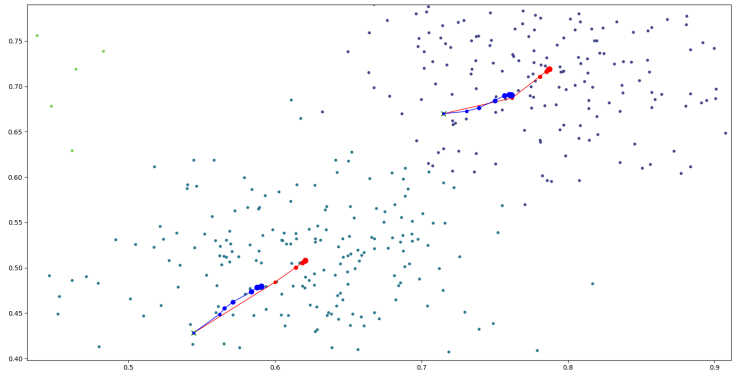


Figura 3: En vermell els *centroides* calculats per Random k-means i en blau els centroides calculats per Weight k-means++. Podem veure con van convergint des de un punt en comú marcat amb una creu de color verd.

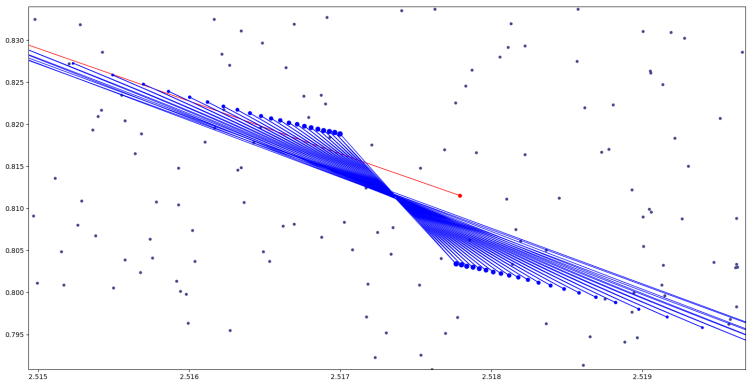


Figura 4: Exemple on el *centroide* de color blau varia massa respecte la iteració anterior i aleshores genera moltes iteracións per convergir.

proper als punts de C com als punts de C'. Si $S(p) > 0$, aleshores el punt p és més proper al clúster C que al clúster C' i indicaria una bona assignació, i si $S(p) < 0$, aleshores el punt p és més proper al clúster C' que al clúster C i per tant no indicaria una bona assignació de p. Finalment doncs, un valor de $S(p)$ que ens indiqui que el punt p està ben ubicat al clúster C és aquell valor suficientment alt i positiu. El valor que indica una assignació optima respecte l'estudi realitzat per Georgios Vardakas et al. [2] és per sobre de 0.74.

3.2 Rand Index

Una mesura de qualitat externa és aquella que compara la solució obtinguda amb una altre solució independent. *Rand Index* permet mesura com de iguals són dues solucions, calculant

si en una solució donada una parella de punts té la mateixa relació (hi són al mateix clúster o si hi son en diferents clústers) en ambdós solucions o de manera contraria no tenen la mateixa relació.

Donades dos solucions del mateix algorisme A i B, *Rand Index* defineix quatre variables: a,b,c i d. La variable a representa el numero de parelles de punts que en la solució A hi pertanyent al mateix clúster i en la solució B també. La variable b representa el numero de parelles de punts que en la solució A hi pertanyent al mateix clúster i en la solució B no. La variable c representa el numero de parelles de punts que en la solució A hi pertanyent a diferents clústers i en la solució B hi pertanyen al mateix clúster. La variable d representa el numero de parelles de punts que en la solució A no hi pertanyent al mateix clúster i en la solució B també.

Finalment doncs si definim el índex R com el *Rand Index* de dues solucions A i B, R es calcularia de la següent manera:

$$R(A,B) = \frac{a+d}{a+b+c+d}$$

Donat que $a+b+c+d$ són totes les possibilitats combinatòries de cada parell de punts, podem dir que $a+b+c+d = \binom{n}{2}$ sent $n = |A| = |B|$. Sabem que $\binom{n}{2} = \frac{n(n-1)}{2}$ i aleshores:

$$R(A,B) = \frac{2(a+d)}{n(n-1)}$$

I per tant nomes cal calcular les variables a i d. Tot i això computacionalment no genera cap mena de diferencia ja que sempre haurem d'iterar per tota parella d'elements.

Rand Index doncs ens diu quina similitud hi ha entre dos solucions A i B de clustering sense necessitat de tenir etiquetació en els clusters. Sigui doncs R una mesura de *Rand Index* de les solucions A i B, aleshores $R \in [0,1]$, on $R = 0$ indica que $A \neq B$ i $R = 1$ indica exactament que $A = B$. Un bon valor de R pot ser per sobre o igual a 0.7 (Inbeom Lee et al. [3]).

4 Elbow method

En *Machine Learning* existeixen molts algorismes per determinar o estudiar els valors òptims dels hiper-paràmetres del nostre algorisme (en l'article Marc Claesen et al. [1] s'hi descriu aquesta qüestió). En el nostre cas, en k-means, el hiper-paràmetre que podem usar és el valor de clústers per dividir el nostre conjunt de dades inicial. Existeixen varis mètodes per estuiar quin valor de K és el més òptim, nosaltres hem usat el mètode del colze o *Elbow method*.

Elbow method consisteix en calcular una mesura de qualitat interna per cada valor de K, per exemple la mitjana de les distancies mitjanes de cada clúster, i visualitzar els resultats per veure quin numero de clústers genera una solució òptima amb el menor numero de clústers possibles.

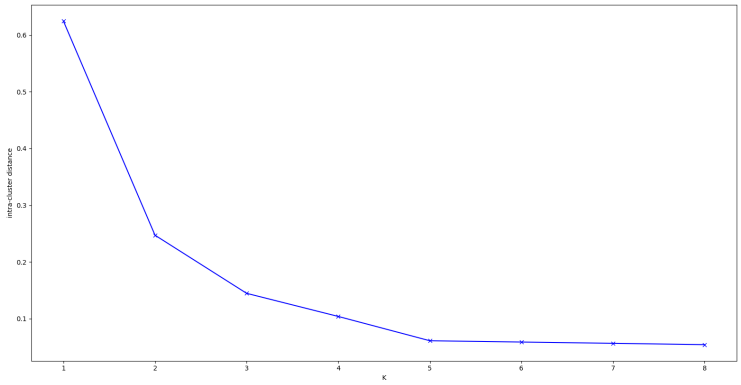
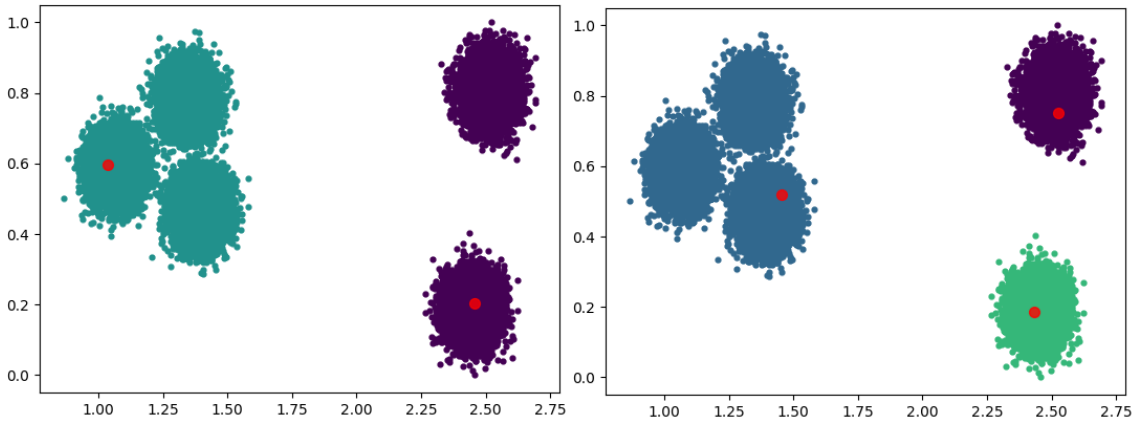


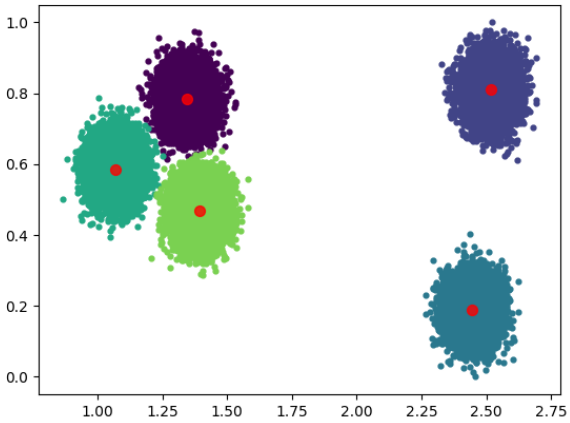
Figura 5: Exemple de *Elbow method* per a valors de $K \in [1, 8]$.

A la Figura 5 podem veure un exemple d'execució del mètode del colze per a valors de $K \in [1, 8]$. És evident veure que mai podrem executar un mètode del colze amb valors de $K \in [a, b]$ on $a < 1$. També és clar veure que sempre que augmentem el valor de K i l'algorisme convergeixi de manera lògica la mesura de qualitat interna disminueixi. Això és degut a que sempre que afegim un clúster totes les distàncies mitjanes de cada clúster disminueixen al tenir una opció de clúster més proper. En l'exemple de la Figura 5 podem discutir si el valor òptim de K és o bé: 2, 3 o inclús 5. Veiem que tenir 2 clústers en canvi de 1 genera una gran millora en la mesura de qualitat, però passar de 2 a 3 clústers també, un poc més lleugera en aquest cas. Finalment passar de 3 a 5 clústers també genera una millora significativa però molt més petita. En aquest cas, pensem que la K òptima seria el valor de $K = 3$. Anem a donar un cop d'ull com el conjunt de dades es de la Figura 5 es representa en un espai de dues dimensions (veure Figura 11).

Com veiem a la Figura 11 podem doncs afirmar que totes les opcions de K mencionades anteriorment serien correctes. En aquest cas hauríem de decidir el valor de K tenint en



(a) $K = 2$ (b) $K = 3$



(c) $K = 5$

Figura 6: K-menas amb valors de $K \in \{2, 3, 5\}$

compte el context del conjunt de dades amb el que treballem donat que hi veiem 3 possibles molt raonables de dividir les dades ($K \in \{2, 3, 5\}$).

També hem hagut de dissenyar un sistema automàtic per, donada una solució del mètode del colze, escollir el valor òptim de K. Per fer-ho hem calculat els següent:

Sigui S un conjunt de solucións de k-means on A_i és una solució amb un valor de $K = i$ i $q : A \to \mathbb{R}$ una mesura de qualitat on l'entrada és qualsevol solució.

$$S = \{A_2, A_3, \cdots, A_n\}$$

Definim els següents vectors Id i Ir, on la notació V_i representa el valor i-èssim del vector

V.

$$\begin{aligned} Id_i &= |q(A_{i+1}) - q(A_i)| \qquad 0 \leq i < |S| \\ Ir_i &= \frac{Id_i^2}{Id_{i+1}} + 2 \qquad 0 \leq i < |Id|) \\ K' &= \arg\max(Ir_i) \end{aligned}$$

Finalment el valor K' representa el valor òptim de K donat una solució S del mètode del colze. En el nostre cas hem usat la mesura interna *inertia* per definir $q : A \to \mathbb{R}$.

$$q(A) = \sum_{i=0}^{n} d(x_i, C(x_i))^2$$

On C és el conjunt de clústers de la solució A. Nota que hem sobrecarregat la notació definint $C(x)$ com el centroide que té assignat el punt x.

Donats els 6 *datasets* de l'enunciat, si executem tal mètode del colze les K òptimes que ens recomana són les següents:

- *dataset* 1: $K = \{2, 3\}$ (tenint en compte que la K òptima és 7).
- *dataset* 2: $K \in \{2, 3, 4\}$ (tenint en compte que la K òptima és 2).
- *dataset* 3: $K = \{2\}$ (tenint en compte que la K òptima és 2).
- *dataset* 4: $K = \{2\}$.
- *dataset* 5: $K = \{4, 5, 6\}$.
- *dataset* 6: $K = \{2, 3, 5\}$ (tenint en compte que la K òptima és 3).

Donat els anteriors resultats, hem decidit en alguns casos definir K com el conjunt dels resultats més comuns de K que recomana l'algorisme. L'únic conjunt de dades on gairebé mai encerta el valor òptim de K és en el *dataset* 1, creiem que el nostre algorisme mai arribará a generar solucions prou òptimes pel *dataset* 1, creiem que és donat per la naturalesa de les dades i que hauríem d'haver extret alguna dimensió no gaire significativa, tot i això, no hem tingut temps de fer tal feina o haver-ho fet amb més cura i deteniment.

5 Estudi de costos

En aquest punt del treball ja hem definit tots els algorismes i bases per començar a realitzar els experiments i treure conclusions objectives respecte l'eficiència en temps i espai dels algorismes implementats com també de la qualitat de les seves solucions.

Estudiarem els costos a partir de les següents variables:

- N: donat un conjunt de dades A, aleshores definim N com el numero de dades: $N = |A|$.
- D: donat un conjunt de dades A, aleshores definim D com la dimensionalitat dels elements de A: $a \in A, D = |a|$.
- K: el nombre de clústers de l'algorisme k-means.

5.1 K-means

Donat que hem implementat dos versions de *k-means*, primer analitzarem computacionalment la part comuna dels dos algorismes i després per separat analitzarem les diferències computacionals.

Sigui K el numero de clústers, N el numero de dades i D les dimensions de les dades. Hem generat la següent taula per definir els costos temporals i d'espai de l'algorisme comú (veure Taula 3).

Cost	Ini. d'estructures de dades	Comp. Clusters	Condició d'aturada
Temporal	O(K)	O($K + KND$)	O(N)
Espaial	O($K + N$)	O($K + N$)	O(N)

Taula 1: Cost del k-means comú

Finalment doncs obtenim un cost temporal O(KND) i un cost espaial O($K + N$). Per una banda, el cost espaial prové de guardar tant els centroides com també els K clústers i de guardar les N dades. Per altre banda, el cost temporal prové de:

- O(K): re-inicialitzar els *centroides* i guardar els de la iteració anterior per la condició d'aturada.
- O(KND): calcular la distancia d'un punt respecte a cada punt del seu mateix clúster, i finalment fer-ho per a cada clúster.
- O(N): en la condició d'aturada s'hi comparen el clústers antics amb els nous, en cost pitjor s'han d'iterar totes les dades (quan la condició d'aturada és certa, això nomes passa un cop però sempre es pot donar el cas d'iterar $N - 1$ dades sense que la condició sigui certa).

5.1.1 Random k-means

Pel Random k-means, com ja hem comentat, estudiarem el cost temporal i espaial dels pasos 1 i 3 que el difereixen del K-means comú i final extreurem el cost final tenint en compte els costos anteriorment estudiats a la secció 5.1.

Cost	Ini. Centroides	Comp. de Centroides
Temporal	$O(K)$	$(N^2 + DK)$
Espaial	$O(K + N)$	$O(K + N)$

Taula 2: Cost del Random k-means

El cost espaial en aquest sentit no varia donat que seguim usant el mateix nombre de centroides i clústers. Si analitzem més precisament cada cost temporal:

- $O(K)$: inicialitzar els clústers suposant que obtenir un valor aleatori en el rang $[0, N-1]$ te un cost constant.
- $O(N^2 + DK)$: es calculen la mitjana de les distàncies de cada punt del clúster (en cost pitjor totes les dades són al mateix clúster, per tant, $O(N^2)$), i finalment $O(DK)$ genera la mitjana de cada dimensió de cada *centroide*.

El cost temporal de Random k-means és $O(N^2 + KND + DK)$. El cost espaial de Random k-means és $O(K + N)$.

5.1.2 Weight k-means++

Pel Weight k-means++ tenim:

Cost	Ini. Centroides	Comp. de Centroides
Temporal	$O(DK^2)$	$(N^2 + DK)$
Espaial	$O(K + N)$	$O(K + N)$

Taula 3: Cost del Random k-means

El cost espaial en aquest sentit no varia donat que seguim usant el mateix nombre de centroides i clústers. Si analitzem més precisament cada cost temporal:

- $O(DK^2)$: inicialitzar els clústers suposant que obtenir un valor aleatori en el rang $[0, N-1]$ te un cost constant. El primer centroide és aleatori però els restants han de calcular les distàncies als altres centroides.
- $O(N^2+DK)$: es calculen la mitjana ponderada de les distàncies de cada punt del clúster (en cost pitjor totes les dades són al mateix clúster, per tant, $O(N^2)$), i finalment $O(DK)$ genera el calcul de la funció de pes w que te cost $O(D)$ i s'ha de calcular per a cada clúster.

El cost temporal de Weight k-means++ és $\mathrm{O}(N^2 + KND + DK^2)$. El cost espaial de Weight k-means++ és $\mathrm{O}(K + N)$.

5.2 Silhouette

Recordem que aquesta mesura de qualitat calcula la distància d'un punt a tots als punts del seu clúster i també del seu clúster veí (el clúster veí d'un punt és aquell que és el més proper sense ser el seu propi clúster). Aquesta mesura es calcula per a tots els punts del conjunt de dades per tant hem d'executar $\mathrm{O}(N^2)$ distàncies. Donat que calcular la distància d'un punt amb un d'altre té cost $\mathrm{O}(D)$, finalment tenim que el cost temporal de *Silhouette* és $\mathrm{O}(DN^2)$ (totes les altres operacións tenen temps constant $\mathrm{O}(1)$). El cost espaial del algorisme és $O(1)$ donat que no guardem cap resultat.

Una solució raonable que hem estudiat per disminuir el temps de computació de *Silhouette* si el volguéssim executar P vegades seria el següent.

1. Inicialitzar una matriu M amb N^2 elements.
2. *If* M és buida.
   2.1 Omplir M on $M_{ij} = d(i, j)$.
3. Calcular *Silhouette* accedint a M per computar les distàncies.

Aquest algorisme te cost $\mathrm{O}(N^2 + P)$, evitant el cost de $\mathrm{O}(PN^2)$ si no computéssim la matriu M. Aquest algorisme per això té cost espaial $\mathrm{O}(N^2)$, si les nostres dades són dins el conjunt $\mathbb{R}$, aleshores un conjunt de 25.000 dades podria generar una matriu M de mida 4.75GB aproximadament.

5.3 Rand Index

El cost de saber la relació combinatòria d'una parella de punts (a, b) de dues solucions A i B diferents és lineal respecte N en cas pitjor, donat que els clústers poden tenir mida N en el pitjor cas. Per tant, donat que s'han de comparar $\binom{n}{2} = \frac{n(n-1)}{2}$ relacions, finalment tenim un cost temporal de $\mathrm{O}(N^3)$. Tot i això podem millorar aquest cost inicial fent ús d'un cost espaial de $\mathrm{O}(N)$ amb l'ajuda de dos diccionaris D_A i D_B on el diccionari D_{Ai} guarda el clúster k del punt A_i i el diccionari D_{Bi} guarda el clúster k' del punt B_i.

1. Omplir dos diccionaris D_A i D_B.
2. *For* $a \in D_A$.
   2.1 *For* $b \in D_B$.

2.1.1 Guardar la relació combinatòria (a, b)

3. Retornar *Rand Index* de totes les relacions combinatòries.

Donat que accedir als dos diccionaris té cost O(1) el nostre algorisme ha reduit el seu cost temporal fins a $\mathrm{O}(N^2 + N) = \mathrm{O}(N^2)$.

5.4 Elbow method

Donat que aquest mètode executa varis cops l'algorisme de k-means, no analitzarem el seu cost donat que depèn del algorisme executat. A trets generals totes les operacions per obtenir la K òptima són constants i si el rang de valors de K és de mida M el cost essencialment és $\mathrm{O}(MT)$ on T és el cost de l'algorisme de k-means.

6 Experiments

En aquesta secció iniciarem explicant la metodologia emprada per a la realització dels experiments sobre els algorismes implementats. Seguidament també observarem la qualitat de les solucions.

6.1 Metodologia dels experiments

Iniciarem estudiant el comportament dels algorismes de k-means. Per realitzar aquests experiments hem realitzat un *script* que genera *datasets* amb la mida de les variables K, N i D que ens interessi.

A la secció 5 hem vist que no hi ha excessiva diferència en el cost temporal de Random k-means i Weight k-means i que per tant nomes estudiarem a la pràctica l'algorisme de Random k-means, també tristament degut a la falta de temps (justificat a la secció 1.1).

Per a cada valor donat de les tres variables a estudiar (K,N,D) executarem 10 cops la mateixa casuística i finalment farem la mitjana dels resultats donats. Així evitarem valors atípics en els resultats i obtindrem conclusions més solides i objectives.

6.2 K-means

Per donar a terme aquests experiments, executarem ambdós algorismes amb diferents valors de: K, de N i de D. Primer observarem com varia el cost temporal de l'algorisme i finalment estudiarem la qualitat de les solucions respecte cada *dataset* proporcionat a l'enunciat del problema.

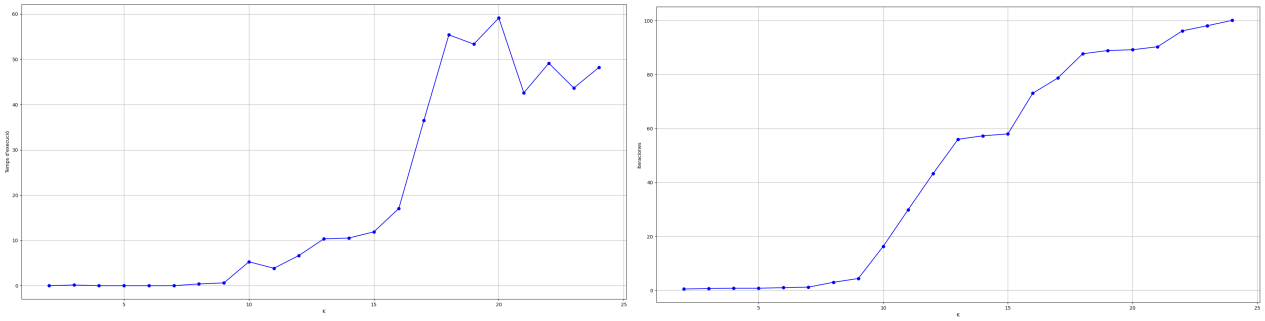
6.2.1 Cost temporal augmentant K

Per aquest experiment hem executat el *dataset* 1 proporcionat per l'enunciat de l'assignatura amb valors: $N = 13611$, $D = 16$ i $K =\in [2, 24]$. Recalquem que la K òptima d'aquest conjunt de dades és 7.

Recordant el cost temporal teòric de la secció 5.1.1 esperem trobar-nos que el temps d'execució de l'algorisme creixi en temps exponencials respecte K.

Efectivament, veiem que el temps creix exponencialment respecte el valor de K i que el numero d'iteracions ho fa linealment, un comportament lògic i raonable. És interessant veure que sembla ser que existeix un *threshold* donat un cert valor de K on fa que el temps d'execució sembli no augmentar més. Creiem que aquest fet es pot donar per varies raons, una d'elles és per la naturalesa de les dades, per com estan distribuïdes.

Per altre banda, pensem que per a valors baixos de N pot ser que l'algorisme no noti tanta exigència a l'hora de computar una iteració quan exactament $\frac{N}{K^2}$ és molt pròxim a N. El



(a) Temps d'execució per cada valor de K (b) Numero d'iteracions per cada valor de K

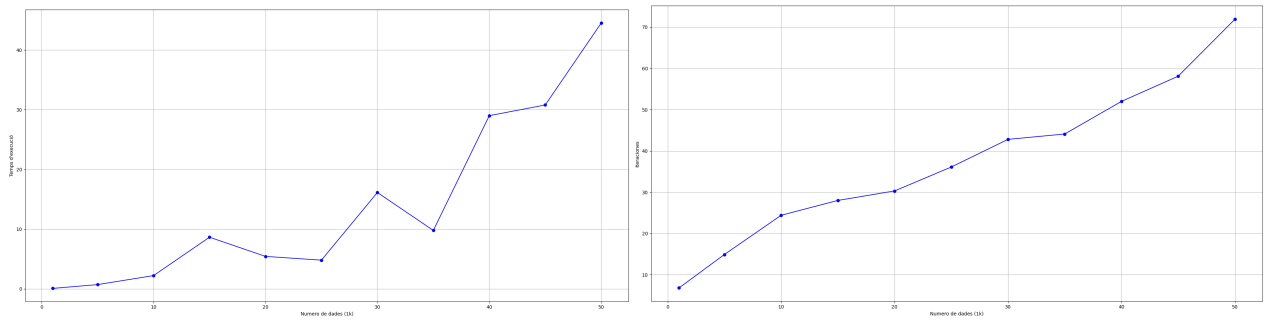
Figura 7: Experiment augmentant K amb Random k-means, on $K \in [2, 24]$

fet de que $\frac{N}{K^2}$ sigui pròxim a N fa que s'hagin de fer menys iteracions per convergir (podem veure que a la Figura 7b el numero d'iteracions sembla anar disminuint poc a poc per a valors grans de K) a la solució, cada iteració tingui un cost computacional menor i que l'algorisme tingui menor complexitat espaial.

6.2.2 Cost temporal augmentant N

Per aquest experiment hem generat un conjunt de dades amb les següents variables: $K = 10$, $D = 5$ i $N \in \{1000, 5000, 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000\}$.

Recordant el cost temporal teòric de la secció 5.1.1 esperem trobar-nos que el temps d'execució de l'algorisme creixi en temps exponencials respecte N.



(a) Temps d'execució per cada valor de N (b) Numero d'iteracions per cada valor de N

Figura 8: Experiment augmentant N amb Random k-means, on $N \in [1000, 50000]$

De mateixa manera que en la secció 6.2.2, el temps d'execució respecte al valor de N augmenta exponencialment. El numero de iteracions augmenta linealment cosa que similarment a la

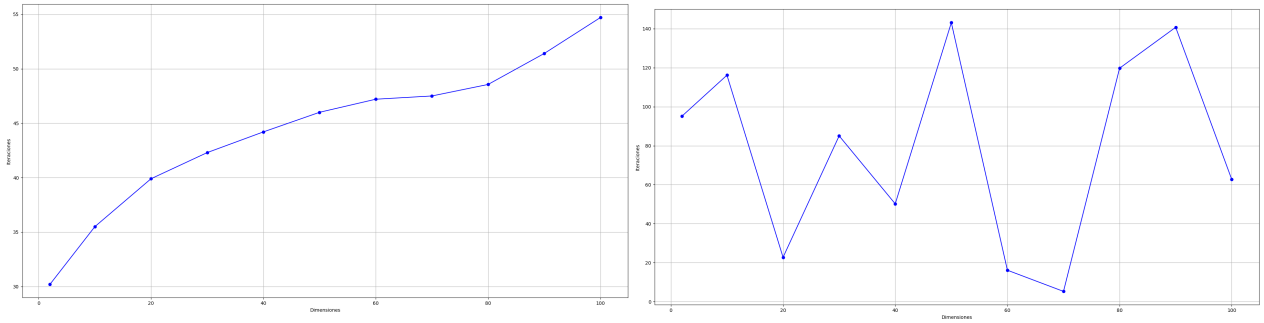
secció 6.2.2 recolza el fet de que el temps augmenti exponencialment respecte a N.

En aquest cas per això, no hem vist cap *threshold* i reforça les conclusions respecte el *threshold* format quan $\frac{N}{K^2}$ és pròxim a N, donat que aquest cop cada vegada N s'allunya més del valor $\frac{N}{K^2}$.

6.2.3 Cost temporal augmentant D

Per aquest experiment hem generat un conjunt de dades amb les següents variables: $K = 10$, $N = 25000$ i $D \in \{2, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$.

Recordant el cost temporal teòric de la secció 5.1.1 esperem trobar-nos que el temps d'execució de l'algorisme creixi en temps lineal respecte D.



(a) Temps d'execució per cada valor de D

(b) Numero d'iteracions per cada valor de D

Figura 9: Experiment augmentant D amb Random k-means, on $D \in [2, 100]$

Tal i com esperàvem, el temps d'execució augmenta linealment respecte D, i el nombre d'iteracions és constant. Tots els resultats obtinguts respecten els estudis teòrics, de la qual cosa ens sentim satisfets tant de l'estudi teòric com també de la realització dels experiments.

6.2.4 Qualitat de les solucions

Silhouette

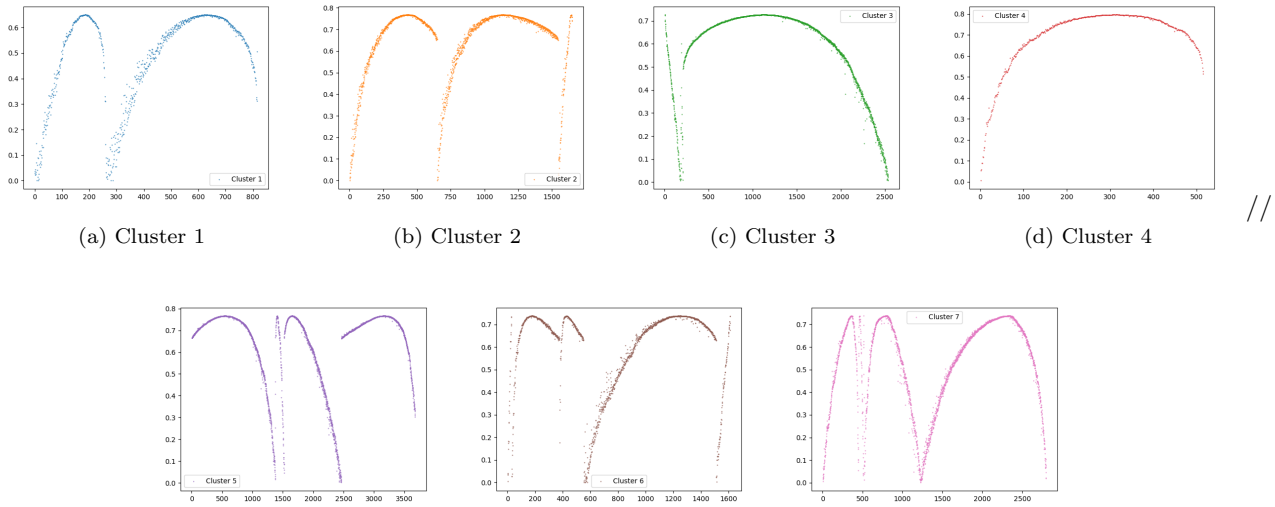
En aquesta secció veurem els resultats de la mesura interna *Silhouette* per a alguns *datasets* donats.

Si ens fixem en la Figura 10, veiem que qualsevol cluster no te cap punt amb un valor de *Silhouette* per sota de 0. Si existís algun punt amb algun valor per sota 0 tindriem el cas d'un punt on està situat a un clúster més llunya del seu clúster òptim. Aquest cas es pot donar en algunes configuracions de k-means, però donada la nostre condició de parada aquest fet mai es pot donar. En general, veiem que la majoria de punts estan per sota del 0.5 i una gran

quantitat de punts és per sobre del 0.74 recomanat. Per tant, pel *dataset* 1 podem estar prou satisfets amb la solució donada, tot i això, creiem que podria ser millorable si s'haguessin tractat les dades d'una manera més favorable estudiant la naturalesa de les dades.

Per altre banda, si observem la Figura 12, podem observar un resultat de la mesura interna molt positiu. La majoria de les dades estan per sobre del 0.7 i poques ronden pel rang $[0.4, 0.0]$.

En canvi, pel *dataset* 4 (veure Figura 13), que no , els resultats no són gaire bons. Donat que no hi havia una K determinada, s'ha executat abans el mètode del colze i automàticament s'ha escollit el valor de $K = 2$. Veient els resultats, potser podríem valorar augmentar un valor de K i provar-ho amb $K = 3$ o bé estudiar la naturalesa de les dades i tractar-les de la manera més convenient.



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 //

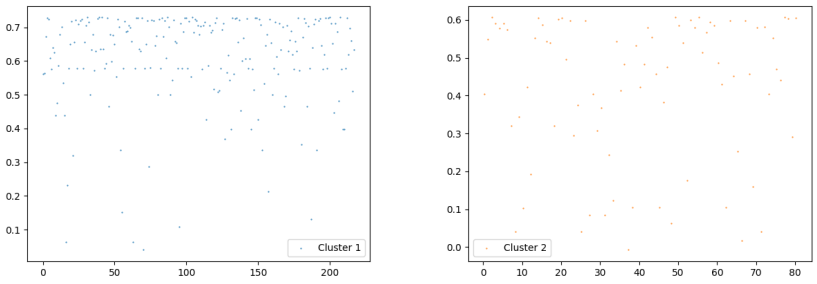
(e) Cluster 5 (f) Cluster 6 (g) Cluster 7

Figura 10: Silhouette value de cada cluster del dataset 1

Rand Index

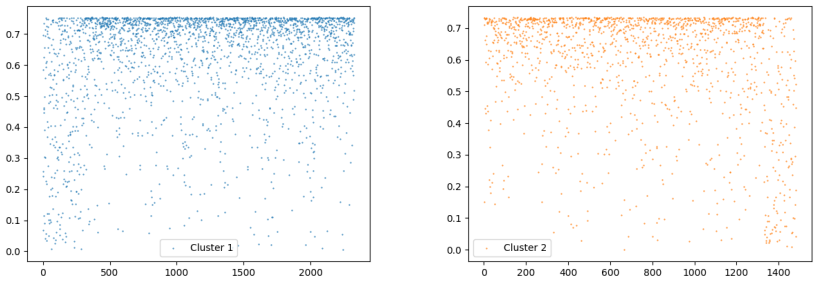
Anem a realitzar la mesura de qualitat externa *Rand Index* pels mateixos *datasets*. Per fer-ho generarem dues solucions A i B amb el mateix algorisme i executarem el *Rand Index*.

Com veiem a la Figura 14 podem veure que les validacions dels *datasets* que estaven marcats amb etiquetat semblen tenir un resultat molt positiu. Els *datasets* 1 i 3 tenen un resultat per sobre de 0.8 per tant podem dir que les solucions són molt bones. Per altre banda, el *dataset* 2 te un valor aproximat de 0.52 i el *Rand Index* indica que podria ser millorable.



(a) Cluster 0 (b) Cluster 1

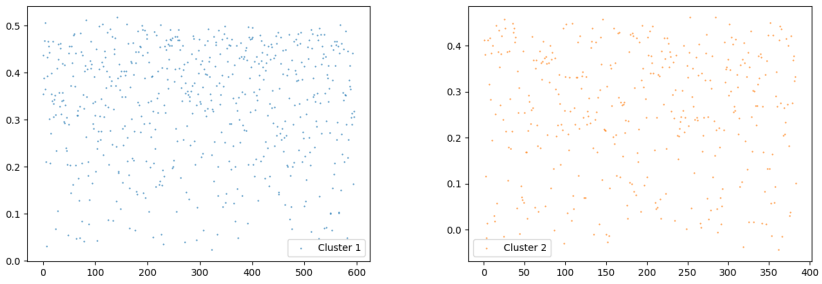
Figura 11: Silhouette value de cada cluster del dataset 2



(a) Cluster 1 (b) Cluster 2

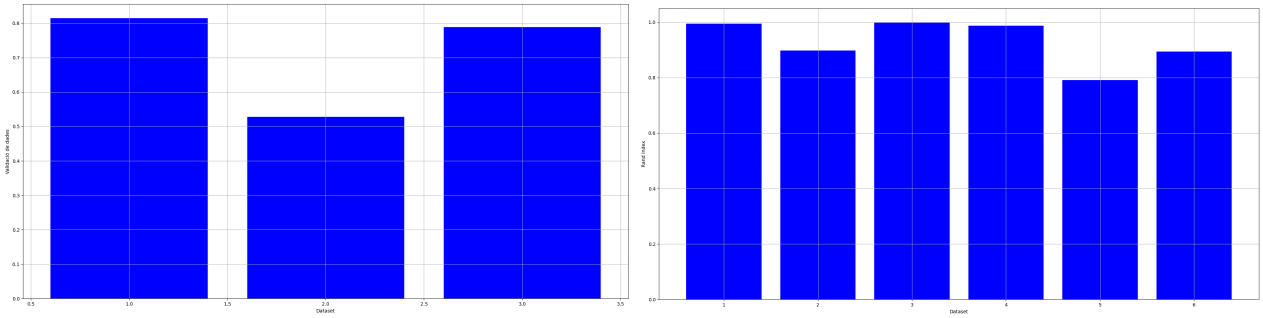
Figura 12: Silhouette value de cada cluster del dataset 3

També hem executat per a cada *dataset* un *Rand Index* amb dues solucions del mateix algorisme pel mateix *dataset*. Els resultats en general son extremadament positius i pensem que no son gaire significatius per treure cap conclusió objectiva, simplement que les convergències son sempre bastant similars i que això demostra que les solucions són solides. Hem executat varis cops aquests valors de *Rand Index* donat que no sempre convergeixen als mateixos clústers (comportament totalment natural) i així evitar valors atípics.



(a) Cluster 1 (b) Cluster 2

Figura 13: Silhouette value de cada cluster del dataset 4



(a) Rand Index de les validacions de cada *dataset*. (b) Rand Index amb dues solucions de cada *dataset*.

Figura 14: Resultats d'executar *Rand Index*.

7 Conclusions

Durant el projecte hem implementat i experimentat amb algoritmes relacionats amb el *Clustering* d'un conjunt de dades. Hem estudiat teòricament la complexitat temporal i espaial dels algorismes implementats i seguidament hem fet experiments per verificar-les a la pràctica. També hem vist la qualitat de les solucions dels nostres algorismes per verificar la correctessa dels mateixos.

Pel que fan els experiments, en tots hem pogut verificar els estudis teòrics. Per una banda hem vist que augmentar el valor de K linealment genera una progressió exponencial en el temps d'execució. Per altre banda, el numero d'iteracions del algorisme augmentaven linealment. El mateix pel que fa augmentar la mida de les dades. Si més no, augmentar el valor de les dimensions de les dades genera una progressió lineal al temps d'execució del algorisme sense augmentar el numero d'iteracions del mateix.

Per altre banda, totes les mesures de qualitat han mostrat sempre un resultat prou positiu i satisfactori. Cal remarcar que l'unic tractament a les dades que hem fet ha sigut la normalització d'elles. Hem intentat treballar amb altres mètodes com el mètode del PCA per reduir algunes dimensions o eliminar valors atípics en les dades. Tot i ser-hi en el codi no ho hem arribat a incloure a l'informe final donat que ens ha faltat temps per arribar a entendre i a trobar resultats significatius per treure conclusions i no malbaratar els resultats de les solucions.

El temps sempre ha sigut un impediment per treure endavant la feina i treure un treball molt més digne i de qualitat. Com per exemple, com ja hem comentat, podríem haver indagat més en el tractament de les dades o haver generat més variants del k-means. També creiem que haver suposat un comportament similar amb les dues versions del k-means (tot i haver discutit algunes diferencies) i no generar experiments per l'algorisme Weight k-means++ fa que el treball perdi moltíssim potencial pel que podria haver sigut en un final. Tot i això, estem contents amb els resultats obtinguts i de la feina feta.

8 Bibliografia

Durant el projecte em consultat les següents referencies per complementar i verificar la qualitat de la feina feta.

Referències

[1] Zhao Song and Chiwun Yang. *An Automatic Learning Rate Schedule Algorithm for Achieving Faster Convergence and Steeper Descent*. *CoRR*, volume abs/2310.11291, 2023. https://doi.org/10.48550/ARXIV.2310.11291

[2] Georgios Vardakas, John Pavlopoulos and Aristidis Likas. *Silhouette Aggregation: From Micro to Macro*. *CoRR*, volume abs/2401.05831, 2024. https://doi.org/10.48550/ARXIV.2401.05831

[3] Inbeom Lee, Siyi Deng and Yang Ning. *Optimal Variable Clustering for High-Dimensional Matrix Valued Data*. *CoRR*, volume abs/2112.12909, 2021. https://arxiv.org/abs/2112.12909

[4] Marc Claesen and Bart De Moor. *Hyperparameter Search in Machine Learning*. *CoRR*, volume abs/1502.02127, 2015. http://arxiv.org/abs/1502.02127