UNIVERSIDAD DE COSTA RICA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS DE LA COMPUTACIÓN E INFORMÁTICA

CI-0124 Computabilidad y Complejidad

Profesora Mauren Murillo Rivera

Grupo 2

Tarea programada 2: Algoritmo de clustering a datos de células

Grupo ZzZ:

Nathalie Alfaro Quesada B90221

José Pablo Mora Cubillo B75044

II Semestre 2024

Descripción

Las aneuploidías son alteraciones cromosómicas que se caracterizan por la ganancia o pérdida de cromosomas completos o segmentos cromosómicos. Este tipo de alteraciones son muy frecuentes en cáncer, hasta el 88% de las muestras del proyecto "The Cancer Genome Atlas" exhiben algún grado de aneuploidía. En la actualidad, se cuenta con tecnologías que permiten medir el número de copias (CN) a nivel de gen, en condiciones normales este valor debería ser de 2.

Se cuenta con un conjunto de datos que contiene datos de CN de 718 genes implicados en cáncer para 1398 líneas celulares cancerígenas. En este proyecto se busca agrupar estas líneas celulares en grupos de acuerdo a similitudes en los valores de CN, para lo que se va a utilizar algoritmos de agrupamiento.

La importancia de lo anterior radica en que se ha observado que los distintos patrones de CN tienen cierto grado de relación con el tipo de cáncer, tejido de origen, estadio y presiones selectivas a las cuales ha estado sujeto el tumor. Debido a esto, se puede hipotetizar que las líneas celulares con patrones parecidos de CN tienen comportamientos similares, por ejemplo, sensibilidad al mismo tipo de fármacos.

Ejemplo de los datos:

symbol model_id	ABCB1	ABI1	ABL1	ABL2	ACKR3	ACSL3	ACVR1	ACVR2A	AFDN	AFF1	...	ZNF208	ZNF331	ZNF384	ZNF429	ZNF521	ZNF626
SIDM00001	3.0000	2.0000	3.0000	2.0000	1.903779	2.0000	2.0000	2.5000	1.0000	2.0000	...	1.5000	1.0000	2.0000	1.5000	2.0000	1.5000
SIDM00002	3.0000	3.0000	3.0000	3.0000	3.000000	3.0000	3.0000	3.0000	2.0000	3.0000	...	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
SIDM00003	4.0000	3.0000	3.0000	3.0000	3.000000	3.0000	3.0000	3.0000	2.0000	2.0000	...	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
SIDM00008	5.0000	3.0000	2.0000	4.0000	4.000000	4.0000	4.0000	5.0000	3.0000	3.0000	...	3.0000	3.0000	3.0000	3.0000	4.0000	3.0000
SIDM00011	7.0000	3.0000	3.0000	4.0000	4.000000	4.0000	4.0000	4.0000	3.0000	3.0000	...	2.0000	3.0000	3.0000	2.0000	3.0000	2.0000
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
SIDM02076	3.8307	2.0521	4.2114	3.2612	3.079400	3.0782	3.1249	3.1249	2.9867	3.1873	...	3.3713	3.7299	3.8227	3.3713	3.6534	3.3713
SIDM02077	2.9821	2.4088	2.1983	1.6317	1.636200	1.6362	1.6053	0.9731	1.6669	1.0377	...	0.9191	1.9538	2.0136	1.8329	1.9477	1.9803
SIDM02078	2.9376	1.9788	2.8560	2.0221	2.882000	2.8820	2.8694	2.8694	2.0289	2.9833	...	1.9748	3.7649	4.9985	1.9748	1.0058	1.9748
SIDM02079	2.9775	2.7970	3.8654	3.1729	1.998400	4.0418	1.9879	1.9879	2.0014	1.1020	...	1.9649	3.8902	3.6761	1.9649	3.0050	1.9649
SIDM02080	1.8506	1.9958	2.0053	2.1766	2.071200	2.0117	2.0267	2.0267	2.0100	1.0082	...	1.9783	2.0057	1.0433	1.9783	3.8952	1.9783

1398 rows × 718 columns

Algoritmos de agrupamientos

El agrupamiento, también conocido como clustering o análisis de grupo, es una técnica utilizada en el análisis de datos para agrupar un conjunto de objetos en grupos (o clusters) de manera que los objetos dentro de un mismo grupo sean más similares entre sí que con los objetos de otros grupos. Los algoritmos enfocados en resolver este problema reciben el nombre de algoritmos de agrupamiento. Los problemas de agrupamiento son considerados NP-Hard, pero existen heurísticas y metaheurísticas a partir de las cuales se pueden obtener buenos resultados empleando una cantidad de recursos computacionales limitada.

Heurística

Como heurística se utilizará el algoritmo k-means (también llamado algoritmo de Lloyd), el cual es un algoritmo iterativo, específico para problemas de agrupamiento y que se puede quedar atrapado en óptimos locales (versión clásica del algoritmo). Generalmente el algoritmo recibe por parte del usuario los datos y un valor k que representa la cantidad de grupos a generar. A partir de lo anterior, se generan en una primera iteración k centroides aleatorios y cada entrada de los datos se asigna al cluster cuyo centroide se encuentra más cerca, para esto se emplea una métrica de distancia.

El proceso de asignación de un dato a un cluster es fundamental y lo utilizaremos tanto en k-means como en el algoritmo genético que se va a emplear como metaheurística. Existen múltiples métricas de distancia, pero la más común es la distancia euclidiana, esta consiste en:

$$d = \sqrt{(X1 - Y1)^2 + (X2 - Y2)^2 + \ldots + (Xn - Yn)^2}$$

Donde Xi corresponde al valor del centroide en la dimensión i y Yi corresponde al valor de la entrada en la dimensión i. Si tenemos K = 4 y 3 dimensiones (a veces también llamadas features), los cluster van a tener la siguiente forma:

C1 = (X1, Y1, Z1); C2 = (X2, Y2, Z2); C3 = (X3, Y3, Z3); C4 = (X4, Y4, Z4)

Con base en lo anterior, K-means consiste en los siguientes pasos:

1) Inicializar los centroides, ya sea aleatoriamente o seleccionando puntos al azar del conjunto de datos.
2) Asignar cada punto al cluster cuyo centroide esté más cercano, utilizando la distancia euclidiana.
3) Recalcular los centroides de los clusters, donde el nuevo centroide estará formado por los promedios por dimensión de cada entrada presente en el cluster.
4) Repetir los pasos de (2) y (3) hasta que las asignaciones de los puntos de datos ya no cambien significativamente.

Metaheurística

Como metaheurística utilizaremos los algoritmos genéticos, basándonos en las descripción de Maulik & Bandyopadhyay (2000) para clustering con algoritmos genéticos. Por practicidad, al igual que en K-means, se asume que el usuario va a brindar la cantidad k de clusters en los que se deben agrupar los datos. El proceso de asignación se realiza empleando una distancia euclidiana de la misma manera que se realiza en k-means.

En relación a lo visto en clases para algoritmos genéticos, se debe puntualizar que el espacio de búsqueda consiste en todos los posibles valores que van a tomar los k centroides. Por lo tanto, el cromosoma va a estar formado por estos mismos valores. Por ejemplo, en el caso de k = 4 y 3 dimensiones, un posible cromosoma sería:

[(X1, Y1, Z1), (X2, Y2, Z2), (X3, Y3, Z3), (X4, Y4, Z4)]

De manera que para la mezcla de la etapa de cruce, se pueden intercambiar clusters y/o valores dentro de los clusters.

Para calcular el valor de fitness de cada cromosoma, se puede escoger una función objetivo que maximice la diferencia entre los clusters o una que minimice la diferencia entre las entradas pertenecientes a cada grupo. En nuestro caso, utilizaremos esta última, la cual va a consistir el inverso de la sumatoria de las distancias euclidianas para cada muestra dentro de cada cluster:

$$Fitness = \frac{1}{\sum_{i=1}^{Nk} \sum_{j=1}^{Ei} dij}$$

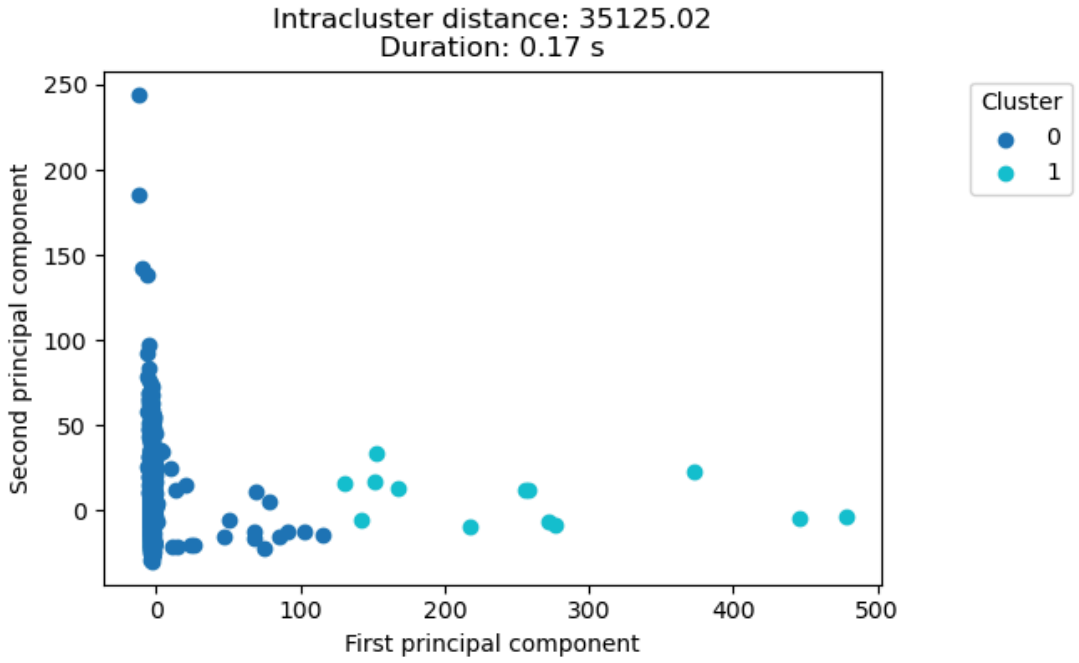
Donde Nk es el número de clusters, Ei es el número de entradas en el cluster i y dkj es la distancia euclidiana de la entrada j en el cluster i con el centroide i.

Fuerza bruta

A partir de lo anterior, existen dos posibilidades de fuerza bruta. (1) probar todos los valores que pueden tomar los centroides. (2) Probar todas las posibles combinaciones que se pueden realizar entre todos los datos y los k clusters.

Análisis de los gráficos

Algoritmo de K-means con n = 1398 y k = 2



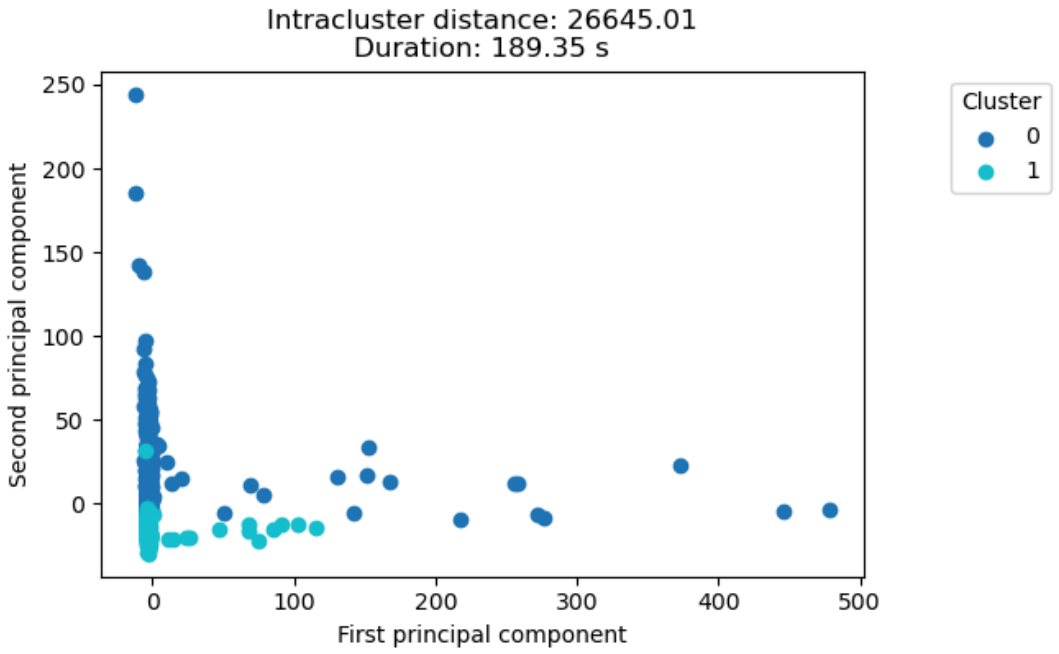
Este gráfico muestra el resultado de un análisis de k-means clustering aplicado a datos de células, reducidos a dos componentes principales utilizando PCA (Análisis de Componentes Principales). El eje X representa el primer componente principal, recoge la mayor variación posible en los datos y el eje Y representa el segundo componente principal que recoge la segunda mayor variación en los datos, independiente del primero. Estos componentes son combinaciones

lineales de las variables originales para reducir la dimensionalidad y facilitar la visualización.

Cada punto representa un dato de una célula en el espacio de los componentes principales, el color azul oscuro es el cluster 0, estos están mayormente agrupados cerca del origen demostrando que son datos más densos y el cluster 1 que es el celeste está más disperso que muestra más variabilidad.

La distancia intracluster tiene un valor de 35125.02, indicando que son más dispersos. Su tiempo de ejecución es de 0.17 segundos, lo cual indica que es eficiente para dividir 1398 datos en 2 clusters (K=2). Este algoritmo busca minimizar directamente las distancias.

Algoritmo genético con n = 1398 y k = 2

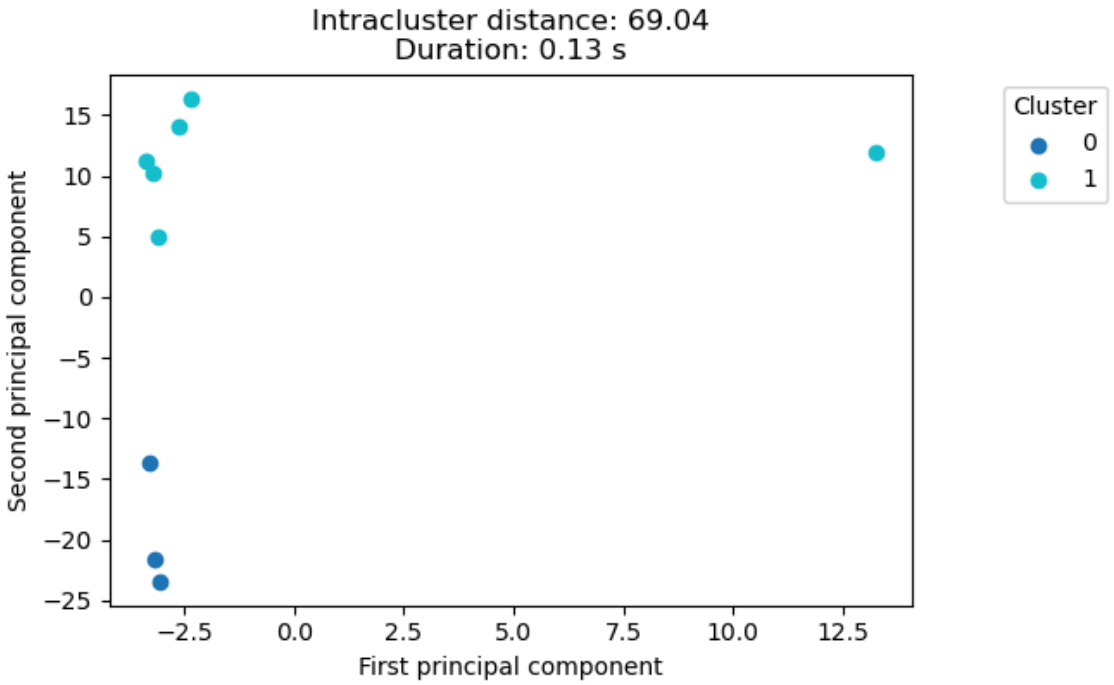


Este gráfico muestra el resultado de un algoritmo genético con clustering de K=2. Sus ejes representan el primer y segundo componente principal, el cluster 0 que es azul es más denso en el origen y el cluster 1 celeste es más disperso. Su distancia intracluster es de 26645.01, menor que el gráfico anterior, lo que indica que los puntos dentro de los clusters son más compactos. Tuvo una duración de

189.35 segundos, siendo mayor a K-means ya que este algoritmo es iterativo, por lo cual es más lento, porque busca una solución óptima evolutiva.

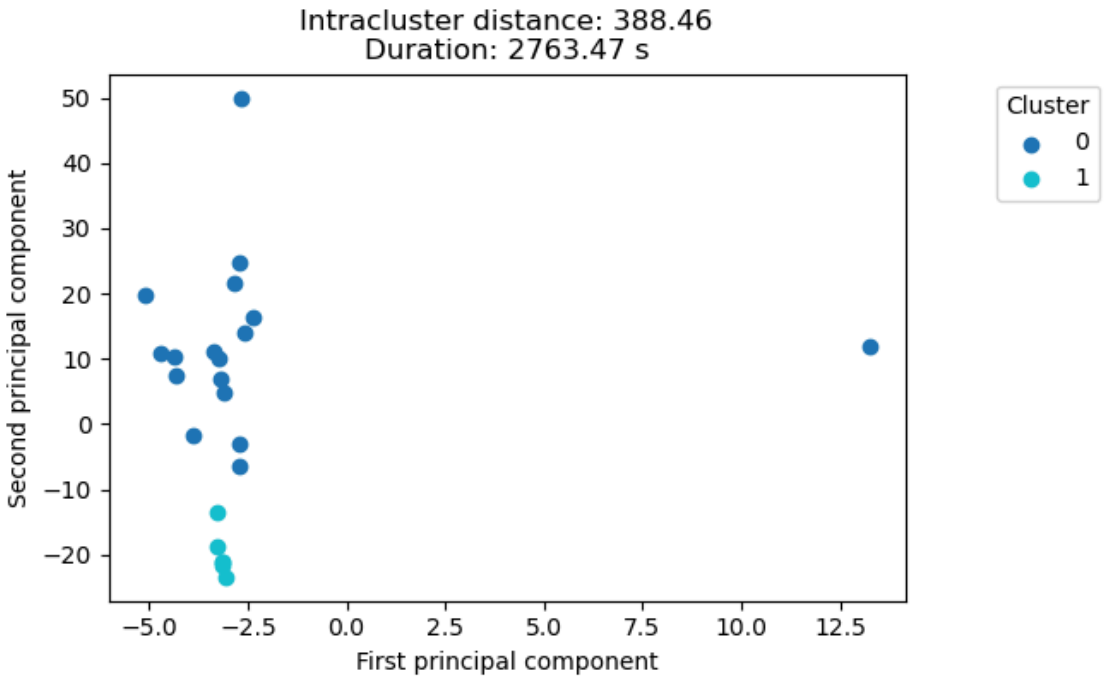
Comparando ambos, si el objetivo es una mayor precisión y clusters más compactos, el algoritmo genético es mejor, por su distancia intracluster. Sin embargo, si el tiempo de ejecución es una prioridad, K-means sería la opción más eficiente, por lo rápido y fácil de implementar, pero puede ser menos preciso.

Algoritmo de fuerza bruta con n = 8



Para el algoritmo de fuerza bruta con n = 8, tuvo 256 combinaciones a explorar con un tiempo de ejecución de 0.125 segundos y su distancia mínima intracluster es de 69.0469.0469.04. En el gráfico se puede observar que los puntos están separados entre los clusters y la baja distancia indica que los clusters son compactos.

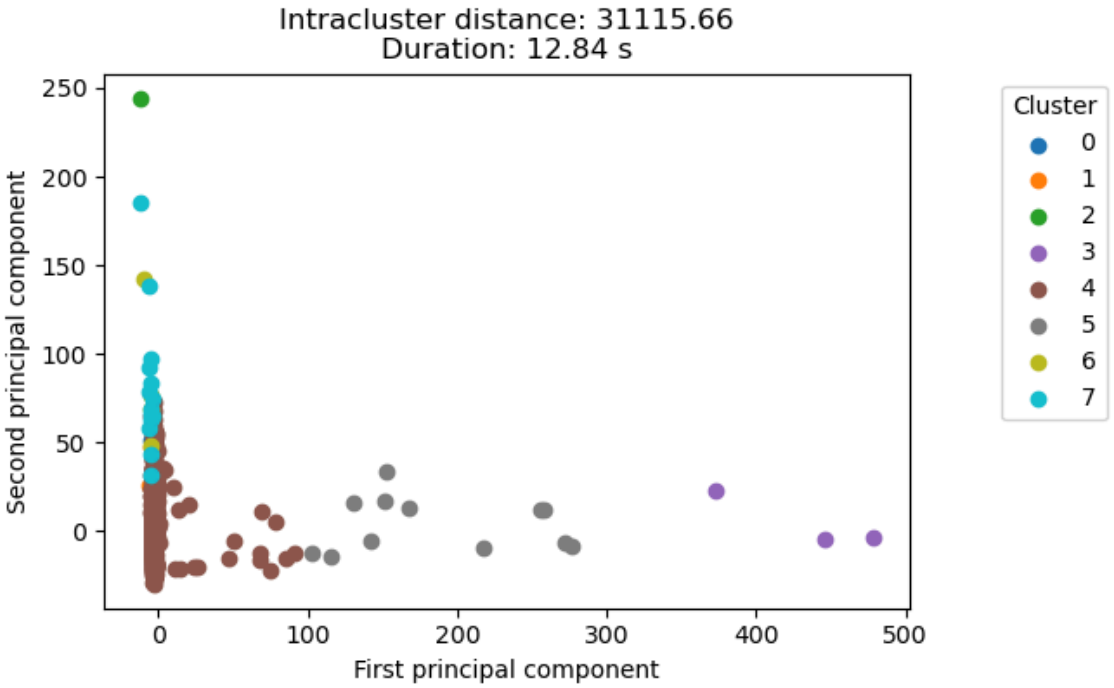
Algoritmo de fuerza bruta con n = 22



En este gráfico se obtuvieron 4,194,304 de combinaciones exploradas con un tiempo de ejecución aproximadamente de 46 minutos y su distancia mínima intracluster es de 388,46. Sus puntos muestran más dispersión dentro de los clusters y la alta distancia connota menor cohesión dentro de los grupos.

Comparando ambos gráficos de fuerza bruta con un n de 8 y un n de 22, el de mayor n aumenta exponencialmente la cantidad de combinaciones exploradas, generando un tiempo de ejecución mayor; mientras que el de menor n produjo clusters más compactos por su baja distancia, indicando que sus datos son más fáciles de agrupar. En el n mayor hay más dispersión que podría indicar que los datos probablemente son menos estructurados, además se puede observar que se tarda más que el de menor n.

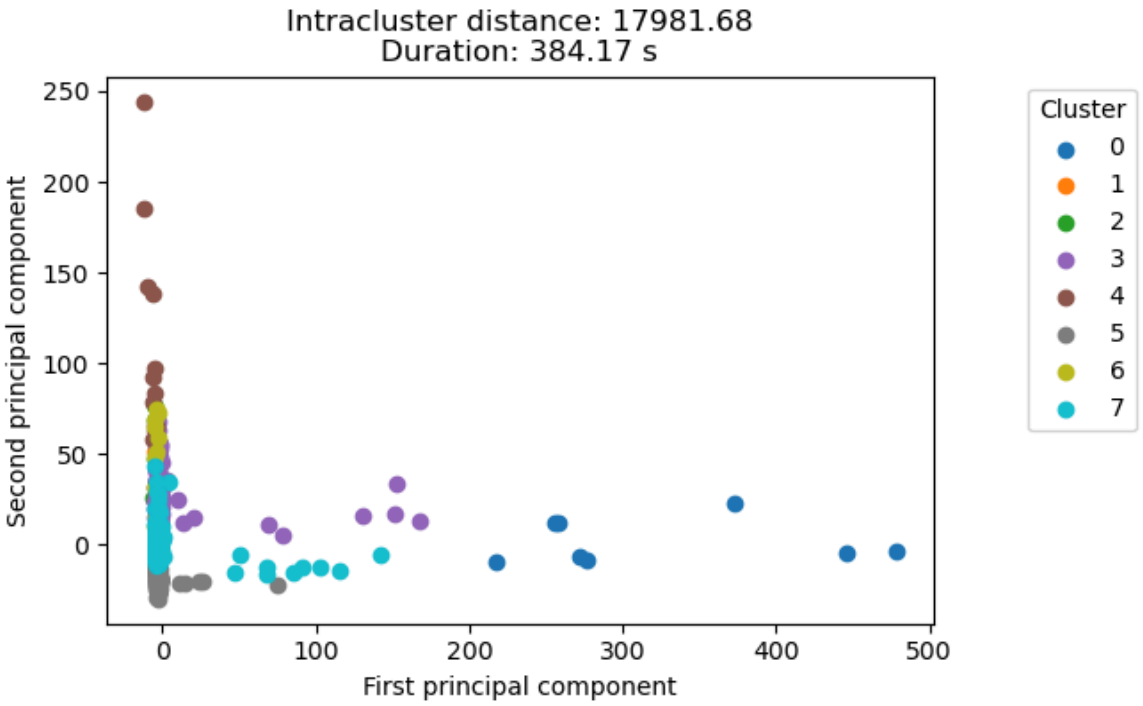
Algoritmo de K-means con n = 1398 y k = 8



En el gráfico de K-means con k = 8 se observa un tiempo de ejecución de 12.84 segundos con una distancia intracluster de 31,115.66 lo que indica mayor dispersión dentro de los clústeres.

Los clústeres se superponen cerca del origen porque demuestran alta densidad y tiene puntos atípicos como los que están alejados, esto demuestra que los clústeres son más dispersos.

Algoritmo genético con n = 1398 y k = 8

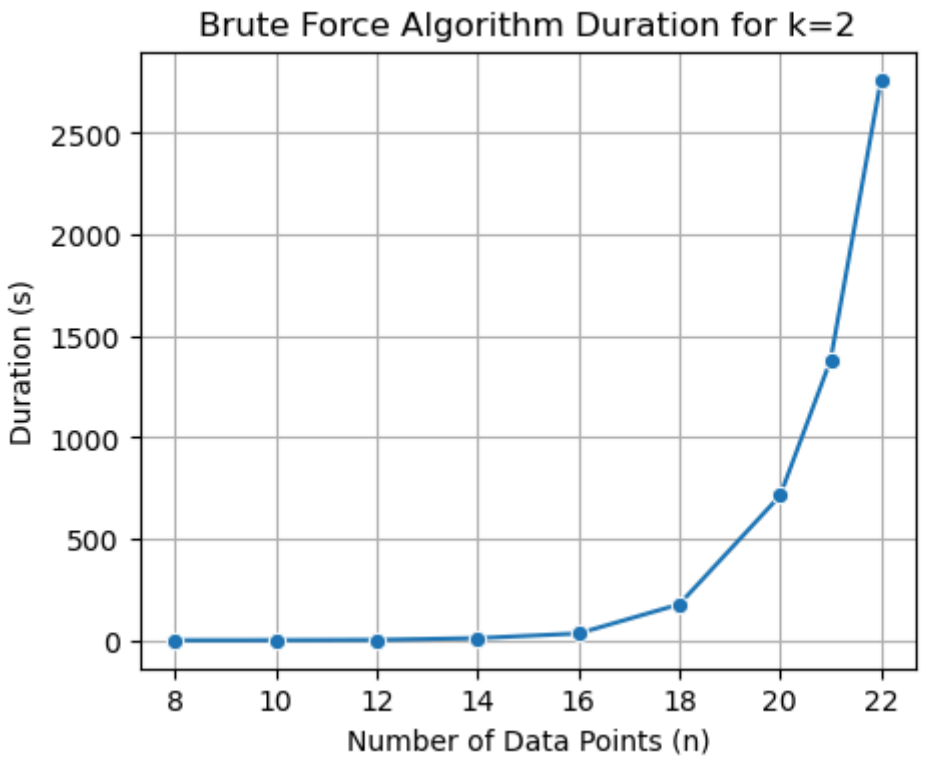


En este gráfico se observa un tiempo de ejecución de 384.17 segundos, su distancia intracluster es de 17,981.68, es decir clústeres más compactos y definidos en el origen, sus puntos atípicos están mejor agrupados y hay menos superposición entre los clústeres..

Comparando ambos se observa que el K-means es más rápido, perfecto para grandes datos, además es más fácil de implementar, mientras que el genético es más lento.

En el K-means tiene una desventaja y es su alta distancia, que hace tener clústeres menos compactos y menos definidos, mientras que en el genético sucede lo contrario.Sin embargo, es más costoso y complejo.

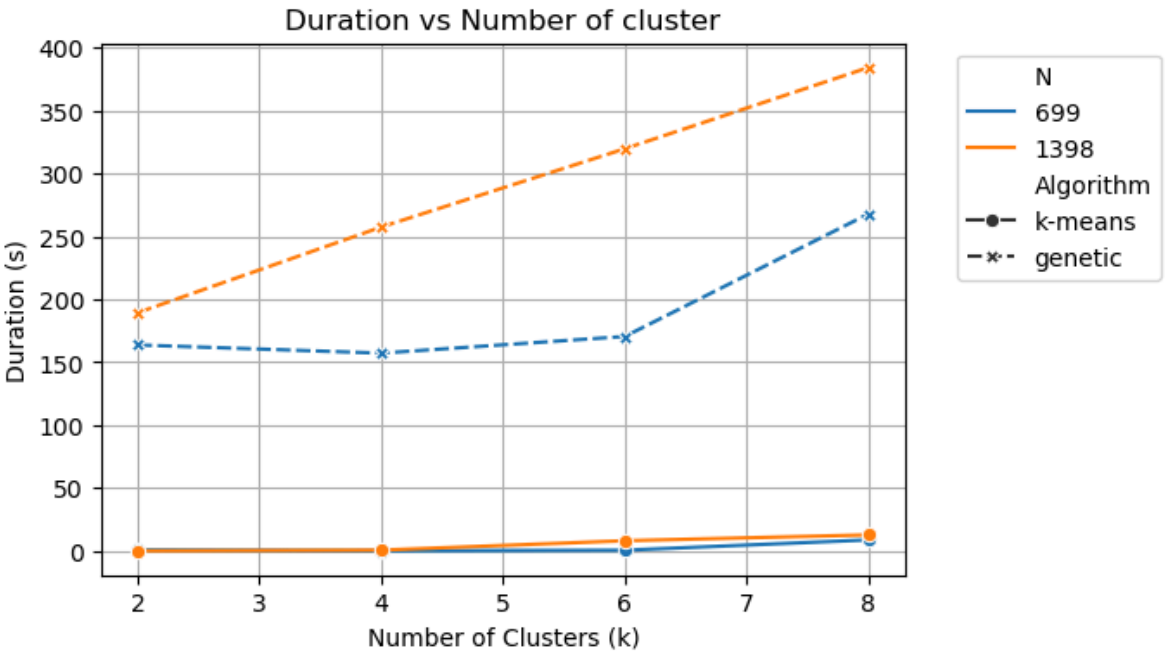
Rendimiento del algoritmo de fuerza bruta para k = 2

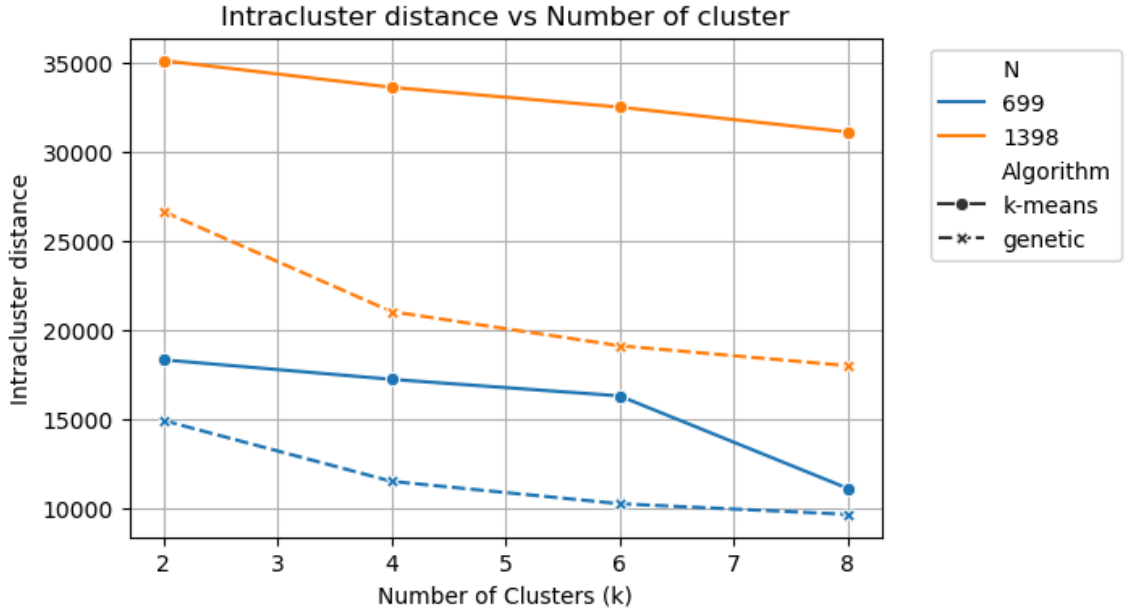


El gráfico muestra la duración del algoritmo de fuerza bruta para k = 2 en función del número de puntos de datos (n). Donde los valores bajos de n, el tiempo de ejecución es insignificante y casi constante. A medida que el número de puntos de datos aumenta (a partir de n = 16), la duración del algoritmo incrementa exponencialmente y en n = 22, el tiempo de ejecución supera los 2500 segundos, indicando una gran carga computacional para un número elevado de datos.

Este analiza todas las combinaciones posibles de puntos para calcular distancias o clústeres, tiene una complejidad computacional de O(n^k). Sin embargo en k = 2, solo se evalúa combinaciones de pares y el número de cálculos crece rápidamente con el aumento de n, por lo cual es mejor usarlos para n < 15.

Rendimiento de los K-means y algoritmos genéticos al incrementar el valor de K





Los gráficos muestran el comportamiento de los algoritmos K-means y Genético en distancia intraclúster y el tiempo de ejecución en función del número de clústeres (K) y del tamaño del conjunto de datos (n = 699 y n = 1398).

En el primer gráfico la distancia disminuye a medida que aumenta el número de clústeres (K) para ambos algoritmos y tamaños de datos ya que permiten un mejor ajuste de los puntos a los centroides. Por lo cual se ve que en el algoritmo genético tiene clústeres más compactos (menor distancia) que K-means en todos

los valores de k, dejándolo con la agrupación de calidad aunque de mayor costo computacional

Para el segundo gráfico el tiempo de ejecución aumenta con el número de clústeres (K), pero el incremento es mucho más pronunciado en el algoritmo genético. En K-means el tiempo es casi constante para ambos tamaños de datos, con un aumento leve conforme K crece.

Entonces se tiene que K-means es más rápido que el algoritmo genético en todos los casos, la duración del algoritmo genético crece pronunciadamente conforme K crece con el tamaño de los datos, lo que lleva a mayor complejidad computacional.

Conclusiones

Si busca rapidez y menor complejidad computacional, K-means es la mejor solución y en especial para grandes volúmenes de datos o de clústeres.

Si la prioridad es la calidad del agrupamiento con grupos definidos por su menor distancia y sin importar el tiempo, el Algoritmo Genético es mejor, pero hay que tomar en cuenta que este posee una alta complejidad computacional.

El algoritmo genético es más útil cuando la calidad de los clústeres es prioritaria y se dispone de suficientes recursos computacionales.

Referencias

Maulik, U., & Bandyopadhyay, S. (2000). Genetic algorithm-based clustering technique. Pattern recognition, 33(9), 1455-1465.