Aplicación del algoritmo K-means y K-means++ como método no supervisado para clasificación de datos

Mirko Iván Czajkowski, María Belén Irala, Carolina Andrea Ruiz

Universidad Gastón Dachary PSS N3300, Argentina
Ingeniería en Informática
Inteligencia Artificial II
{Mbelenirala, lurikoan, carolinaandrearuiz}@gmail.com

Abstract. El algoritmo de K-means es una de las técnicas que permite a partir del conjunto de datos lograr definir particiones dado una característica similar. Por ejemplo, la semejanza de las preferencias de los usuarios para definir perfiles de usuarios, semejanza en color y ubicación de cada píxel para detectar objetos en imágenes, frecuencia en las interacciones para analizar vínculos en redes sociales.

Genera k grupos que está representado por el promedio de los puntos que lo componen. Existe un centroide por cada k grupo que se define como la representación de este. Dicho valor k es un parámetro que debe ser definido con anterioridad. Para definir los centroides iniciales se toman en base a una inicialización aleatoria o un análisis de sensibilidad de la inicialización.

Este informe contiene una introducción al tema y las definiciones iniciales que justifican a su implementación en un lenguaje de programación Python, cuales fueron los resultados y las conclusiones obtenidas. Dado esto último hay que destacar que la elección del centroide o semilla inicial determina el agrupamiento final de los datos para ello se presenta una mejora al algoritmo denominado K-means++ donde se presenta una optimización de la selección de centroides iniciales.

Keywords: K-means, K-means++, inicialización aleatoria, inicialización heurística, clustering.

1 Introducción

A lo largo de los años los conjuntos de datos cada vez se han vuelto más grandes y esto conlleva a la aplicación de algoritmos para el procesamiento de ellas.

El agrupamiento de los datos es una técnica que divide un conjunto de objetos en grupos, donde los objetos que pertenecen a un grupo son muy similares entre sí, y al mismo tiempo, son diferentes respecto a los objetos que pertenecen a otros grupos [1].

El algoritmo de K-means es una técnica de clustering o particionado de datos no supervisado dado que ellos no se encuentran definidos en una clase. Su objetivo es identificar patrones en los datos y particionarlos con el criterio de que sean similares

dado una característica particular, por ejemplo, la distancia entre ellas. Algunos ejemplos del uso de esta técnica son para la definición de perfiles de usuario dado las semejanzas en sus preferencias, detectar objetos en una imagen dado la semejanza de los pixeles en su posición y color, entre otros.

Inicialmente se debe definir un numero k de grupos (definido por el usuario) y este divide los datos iterativamente en k grupos según una función de distancia y un criterio de parada [2].

El resultado de la configuración de los k grupos esta significativamente sensible a los centroides que se seleccionan inicialmente de manera aleatoria. Dada la situación anterior existe una mejora del algoritmo denominado K-means++ que define los centroides o semillas iniciales no de forma aleatoria sino dado un procedimiento optimizado para su selección.

El objetivo del trabajo es lograr implementar el algoritmo de K-means y Kmeans++ desarrollando un aplicativo donde nos permita comparar dichas implementaciones y definir conclusiones sobre ellas.

A continuación, en el informe, se definirá en detalle la implementación del algoritmo, generando simulaciones y presentando los resultados y conclusiones obtenidas.

2 Marco Teórico

El ser humano puede definir una categorización a través de la visualización de los datos en dos o tres dimensiones, pero si contamos con datos con más de tres dimensiones esto ya no se vuelve tan intuitivo y para ellos se puede recurrir a mecanismos matemáticos que nos permitan decir algo de los datos.

El algoritmo K-means es un algoritmo no supervisado dado que sus datos no se encuentran definidos por una clase, se vale de encontrar características a partir de los propios datos (dataset). Para ello define un número de grupos o clusters definidos por una característica similar y establecer una categorización de los mismos.

Cada grupo o cluster está constituido por un centroide que representa al grupo. Esta se puede inicializar aleatoriamente o bajo una configuración establecida (inicialización heurística).

Para lograr identificar estos grupos a través de un algoritmo se define el siguiente diagrama de flujo:

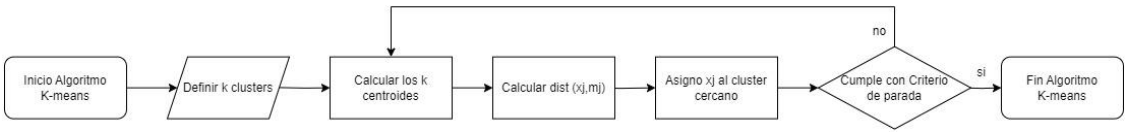

Fig. 1. Diagrama de flujo Algoritmo de K-means.

Dando una descripción de la figura 1, para iniciar el algoritmo primero se debe definir los k clusters para los cuales se debe particionar a los datos. Luego de ello se calcula los k centroides iniciales, tomando centroides de los propios datos, definido de

forma aleatoria, o estableciendo alguna heurística. A partir de obtener los centroides se calcula la distancia euclidiana (xj,mj) de cada elemento del dataset contra cada centroide. Una vez obtenido dichas distancias se aplica la clasificación de las instancias a los diferentes clusters dado la menor distancia entre ellos. Por último, se valida los criterios de parada y si cumple con uno de ellos el algoritmo finaliza arrojando la configuración de clusters resultantes, pero en el caso que no cumpla con alguno de los criterios se vuelve a calcular los nuevos centroides de cada cluster estableciendo la media de cada componente de cada cluster.

Para el cálculo de las semillas iniciales aleatorias se aplica una función de elección desde el dataset aleatoriamente (K-means).

Para el caso de la selección de las semillas iniciales dado una heurística se calcula la media de los datos y se toma la instancia más cercana como centroide inicial y para los siguientes k centroides se toma aquel que este a mayor distancia de él (K-means++).

Luego se establece la distancia euclidiana definida como [2]:

$$dist(x_i, m_j) = \sqrt{(x_{i1} - m_{j1})^2 + (x_{i2} - m_{j2})^2 + \cdots + (x_{ir} - m_{jr})^2} \quad (1)$$

Donde cada elemento del dataset contra cada k centroide semilla se establece la distancia de la formula (1) y se asigna al grupo o cluster con menor distancia.

Se vuelve a computar los centroides como la media del grupo o cluster [2]. Para el cálculo de la media del centroide:

$$m_j = \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i \quad (2)$$

Esto se repite hasta cumplir con un criterio de parada que pueden ser [2]:

- No se genera ninguna reasignación de instancias a un nuevo cluster o es mínimo.
- No se generan cambios en los centroides o es mínimo.
- La suma de los errores al cuadrado es mínima, establecida como SSE.

$$SSE = \sum_{j=1}^{k} \sum_{x \in C_j} dist\ (x, m_j)^2 \quad (3)$$

Cabe destacar en este tipo de algoritmo que la inicialización de los centroides define la configuración de los clusters resultantes, como así también la elección del k cluster, por ello existe una medida llamada Calinski-Harabasz (CH) como criterio de relación de varianza para la selección del k óptimo, donde una puntuación o score alta se define cuando una partición interna del cluster es compacta y estos grupos están mayormente separados.

El Score indica una relación entre la suma de la dispersión entre cluster y dentro del cluster [3]. Calinski-Harabasz (CH) está definida como:

$$CH(k) = \frac{\frac{1}{k-1} g(\Pi*)}{\frac{1}{m-k} f_{LS}(\Pi*)}, 1 < k < m \quad (4)$$

Básicamente esta medida nos permite definir la asignación de un k óptimo. A partir de estas definiciones se establece la solución implementada.

3 Solución Implementada

Para la implementación del algoritmo de K-means se desarrolló un programa ejecutable de tipo desktop en lenguaje de programación Python.

El programa permite realizar la elección de 3 tipos de dataset precargados o cargar un nuevo dataset elegido por el usuario. Además, permite elegir para dicho dataset la selección del valor de k clusters establecido desde 1 > k > 6.

Al momento de seleccionar "Graficar Resultados" realiza la ejecución del algoritmo K-means y K-means++.

Luego de ello permite la visualización gráfica de los resultados de ambos algoritmos junto con el paso a paso de cada uno de ellos. Esto significa que brinda la información de cada iteración y selección de centroides junto con el cálculo de Calinski-Harabasz Score (CH) y el criterio de parada utilizado en dicha corrida. Gráficamente se puede evidenciar la comparación de los centroides iniciales y finales de una corrida.

Las librerías utilizadas para el desarrollo del aplicativo son las siguientes:

Numpy [4], Csv [5], Matplotlib [6], Math [7], Sklearn [8], Tkinter [9], Screeninfo [10], Pillow [11] y Pyinstaller [12].

A partir de la solución implementada se procede a presentar los resultados obtenidos.

4 Resultados Obtenidos

Se analizará tres dataset precargados compuesto por instancias en dos dimensiones. Gráficamente podemos ver a cada uno de los dataset como lo siguiente:

Dataset 1: Compuesto por 630 instancias de dos dimensiones (x;y).

Dataset 2: Compuesto por 600 instancias de dos dimensiones (x;y).

Dataset 3: Compuesto por 800 instancias de dos dimensiones (x;y).;

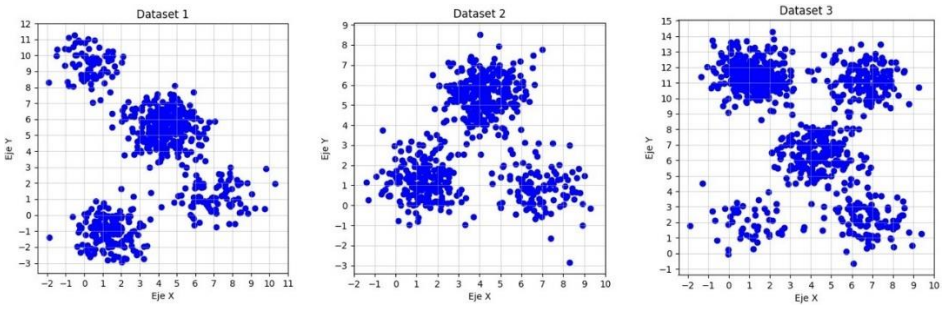

Fig. 2. Representación gráfica de los tres datasets.

Se establece un 1 < k < 6 de clusters preestablecidas. Definimos dos criterios de parada: no hay más asignaciones de instancias y no hay más cambios en los centroides.

Luego de ejecutar el programa nos permite generar comparativas aplicando el algoritmo de K-means y K-means++ para cada uno de los dataset. Se quiere probar la eficiencia de cada uno de los algoritmos porque la selección de la semilla juega un papel fundamental para lograr definir la categorización de las instancias del dataset. Las pruebas se realizarán definiendo lo siguiente:

- Comparación de número de iteraciones entre ambos algoritmos.
- Comparación de Calinski-Harabasz Score (CH).

Para llevar a cabo las pruebas se realiza 10 corridas por cada uno de los dataset y cada valor de k.

Con los resultados obtenidos se procede a calcular el promedio de los resultados de las 10 iteraciones dando como resultado las siguientes tablas para cada dataset:

La columna K representa el número de cluster a agrupar para el dataset. La columna N° Iteraciones representa el promedio de iteraciones de 10 corridas por cada k elegido. La columna CH representa el promedio de Calinski-Harabasz de 10 corridas por cada k elegido.

En la tabla 1 se puede visualizar el valor más alto de CH cuanto se elige un k = 4. Esto se cumple tanto para el algoritmo de K-means como K-means++. El número de iteraciones es el más bajo cuando CH es el mayor.

Table 1. Dataset 1.

K	N° Iteraciones		CH	
	K-means	K-means++	K-means	K-means++
2	6	10	743,582	743,582
3	8	4	871,819	892,740
4	6	4	1759,255	1888,762
5	12	7	1417,059	1549,201

En la tabla 2 se puede visualizar el valor más alto de CH cuanto se elige un k = 3. Esto se cumple tanto para el algoritmo de K-means como K-means++. El número de iteraciones es el más bajo cuando CH es el mayor.

Table 2. Dataset 2.

K	N° Iteraciones		CH	
	K-means	K-means++	K-means	K-means++
2	4	4	512,743	272,838
3	5	3	1377,751	1377,747
4	9	14	1080,075	1151,288
5	12	13	957,190	969,443

En la tabla 3 se puede visualizar el valor más alto de CH cuanto se elige un k = 5 para el algoritmo de K-means++, pero no así para K-means. El número de iteraciones es el más bajo cuando CH es el mayor para el algoritmo de K-means++.

Table 3. Dataset 3.

K	N° Iteraciones		CH	
	K-means	K-means++	K-means	K-means++
2	5	8	1076,713	1076,713
3	11	9	1136,766	713,972
4	12	12	1393,606	1357,619
5	12	4	1284,440	1981,957

5 Conclusiones

En este informe se ha descrito como implementar el algoritmo de K-means basados en inicialización aleatoria o bajo una heurística. También se pruebas del aplicativo determinando dos criterios y plasmando los resultados obtenidos.

A partir de las pruebas y los resultados obtenidos podemos concluir que la semilla inicial determinará como serán definidas las agrupaciones o clusters.

En cuanto a la comparación entre K-means y K-means++, notamos que si bien el algoritmo K-means++ requiere un mayor procesamiento inicial para calcular la semilla inicial, logra obtener una solución en menos iteraciones en comparación con la inicialización aleatoria para números de k impares. Esto puede ser un factor que considerar en la elección del algoritmo a utilizar, aunque también depende de la distribución del dataset.

Finalmente, las pruebas realizadas sugieren una fuerte relación entre la elección del número de clusters (k) y la obtención del índice CH alto. Como el caso de las tablas 1, 2 y 3, se puede apreciar que seleccionar un valor de k que se ajuste de manera adecuada a la estructura de los datos conduce a una mejor descripción de los agrupamientos, lo que se refleja en una alta puntuación de CH. Esto se ejemplifica claramente al definir k = 4 en la tabla 1, donde se observa una división precisa de los datos en cuatro secciones, lo mismo sucede en la tabla 2 con k = 3 y en la tabla 3 con k = 5, lo cual resalta la importancia de elegir cuidadosamente el número de clusters en nuestro análisis. El motivo de ello es porque CH definida como la dispersión entre cluster sobre la dispersión promedio de cada cluster valida que, si contamos con una menor distancia entre cluster y una mayor dispersión en un cluster da como resultado un menor resultado de CH, caso contrario si la dispersión entre cluster es mayor y la dispersión en un cluster es menor podemos describir mejor la distribución de los datos y con ello da un mayor valor de CH.

Referencias

1. Autor, Joaquin Perez-Ortega, Miguel Hidalgo-Reyes, Noe Alejandro Castro-Sanchez, Rodolfo Pazos-Rangel, Ocotlan Diaz-Parra, Victor Olivares-Peregrino, Nelva Almanza-Ortega. Una heurística eficiente aplicada al algoritmo K-means para el agrupamiento de grandes instancias altamente agrupadas (2017). SciELO, ISSN: 20079737, DOI: 10.13053/CyS-22-2-2548., México (2023)
2. Autor, Liu, B., & Mining, W. D. (2011). Web Data Mining. Exploring Hyperlinks, Contents, and Usage Data. Ser. Data-Centric Systems and Applications. Springer Berlin Heidelberg. Capítulo 4 - Unsupervised Learning (Págs. 133-138).
3. Autor, Rudolf Scitovski, Kristian Sab, Francisco Martínez Alvarez, Šime Ungar. (2021) Cluster Analysis and Applications. Springer Capítulo 5 - Indexes (Págs. 102-104).
4. NumPy, NumPy, https://numpy.org/. Accedido por última vez el 23 Oct 2023.
5. Python documentation, csv - Lectura y escritura de archivos CSV, https://docs.python.org/es/3/library/csv.html. Accedido por última vez el 23 Oct 2023.
6. Matplotlib, Visualization with Python, https://matplotlib.org/. Accedido por última vez el 23 Oct 2023.
7. Python documentation math - Funciones matemáticas, https://docs.python.org/es/3.10/library/math.html. Accedido por última vez el 23 Oct 2023.
8. Scikit-learn documentation, Machine learning in Python, https://scikit-learn.org/stable/. Accedido por última vez el 23 Oct 2023.
9. Python documentation, tkinter - Interface de Python para Tcl/Tk, https://docs.python.org/es/3/library/tkinter.html. Accedido por última vez el 23 Oct 2023.
10. PyPI, screeninfo, https://pypi.org/project/screeninfo/. Accedido por última vez el 23 Oct 2023.
11. PyPI, Pillow, https://pypi.org/project/Pillow/. Accedido por última vez el 23 Oct 2023.
12. PyPI, Pyinstaller, https://pypi.org/project/pyinstaller/. Accedido por última vez el 23 Oct 2023.