

Comparando el algoritmo Lista de Clusters para el problema de clustering en el aprendizaje automático

Proyecto final curso de posgrado “Nuevas Propuestas Para Búsquedas Por Similitud En Bases De Datos Métricas”

Juan Martín Loyola

10 de abril de 2018



**FACULTAD DE CIENCIAS FÍSICO
MATEMÁTICAS Y NATURALES**



Organizar los datos en grupos significativos es uno de los modos de comprensión y aprendizaje más fundamentales. El análisis de grupos (en inglés, cluster analysis) es el estudio formal de métodos y algoritmos para agrupar, o “clusterizar”, objetos de acuerdo a similitudes o características intrínsecas medidas o percibidas. El análisis de grupos no utiliza etiquetas de categoría que asignen a objetos identificadores previos, es decir, etiquetas de clase. La ausencia de información de la categoría distingue la agrupación de datos (aprendizaje no supervisado) de la clasificación (aprendizaje supervisado). El objetivo de clusterizar, o de la agrupación, es encontrar la estructura en los datos y, por lo tanto, es de naturaleza exploratoria. En este trabajo, se proporciona una breve descripción del análisis de grupos, se resumen los métodos de agrupación más utilizados, se presenta el algoritmo *Lista de Clusters*, se lo aplica al problema de agrupamiento y se compara su desempeño frente a los otros algoritmos.

1. Introducción

Una definición operacional de agrupamiento puede establecerse de la siguiente manera: dada una representación de n objetos, encuentre K grupos basados en una medida de similitud tal que las similitudes entre objetos en el mismo grupo sean altas mientras que las similitudes entre objetos en diferentes grupos sean bajas. Pero, ¿cuál es la noción de similitud? ¿Cuál es la definición de un grupo? La Figura 1 muestra que los grupos pueden diferir en términos de su forma, tamaño y densidad. La presencia de ruido en los datos hace que la detección de los grupos sea aún más difícil. Un grupo ideal se puede definir como un conjunto de puntos que es compacto y aislado. En realidad, un grupo es una entidad subjetiva que está en el ojo del espectador y cuya importancia e interpretación requiere conocimiento del dominio. Pero, si bien los humanos somos excelentes buscadores de grupos en dos y posiblemente tres dimensiones, necesitamos algoritmos automáticos para datos de alta dimensión. Es este desafío junto con la cantidad desconocida de grupos para los datos dados que ha resultado en miles de algoritmos de agrupamiento que se han publicado y siguen apareciendo. [Jain, 2010]

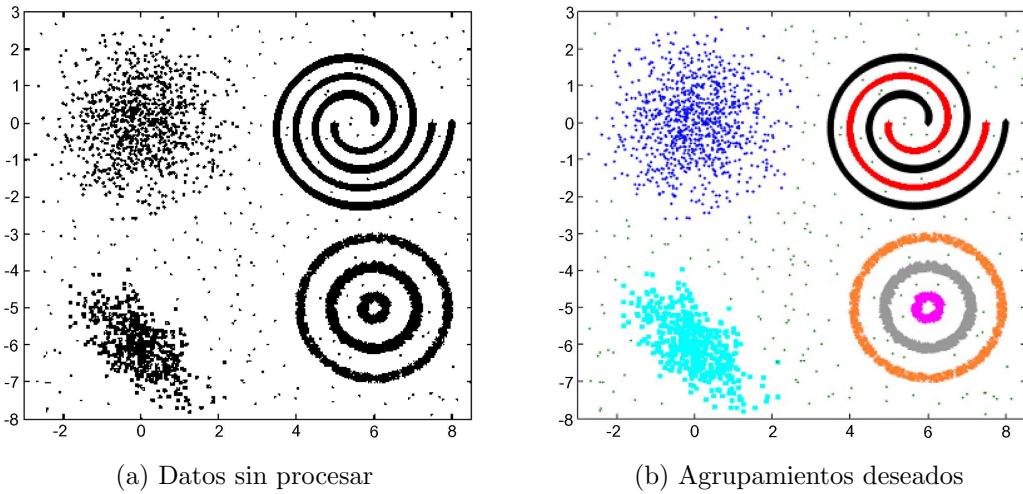


Figura 1: Ejemplo de clustering.

La agrupación de datos se ha utilizado principalmente para los siguientes propósitos:

- Estructura subyacente: para obtener información sobre los datos, generar hipótesis, detectar anomalías e identificar características destacadas.
- Clasificación natural: para identificar el grado de similitud entre formas u organismos.
- Compresión: como un método para organizar los datos y resumirlos a través de prototipos de grupos.
- Análisis exploratorio de datos: para aprender y obtener intuiciones sobre los datos.

- Segmentación de imágenes: para dividir una imagen digital en distintas regiones para la detección de fronteras o el reconocimiento de objetos.
- Sistemas de recomendación: para recomendar nuevos artículos basados en los gustos de los usuarios.
- Análisis de redes sociales: para reconocer comunidades dentro de grandes grupos de personas.

En la sección que sigue se proporciona una breve descripción del análisis de grupos y se resumen los métodos de agrupación más utilizados. En la Sección 3 se presenta el algoritmo Lista de Clusters. Luego, en la Sección 4, se aplica el algoritmo Lista de Clusters al problema de clustering y se compara su desempeño frente a los otros algoritmos. Por último, en la Sección 5, se presentan las conclusiones del trabajo y posibles mejoras.

2. Clustering

El agrupamiento o *clustering* (en inglés) es el proceso de agrupar objetos similares. Hay dos tipos de entradas que podríamos usar. En la agrupación basada en la similitud, la entrada al algoritmo es una matriz de desemejanzas $N \times N$ o una matriz de distancia D . En la agrupación basada en características, la entrada al algoritmo es una matriz de características $N \times D$ o matriz de diseño X . El agrupamiento basado en similitud tiene la ventaja de que permite la fácil inclusión de similitudes específicas de dominio o funciones de kernel. La agrupación basada en características tiene la ventaja de que es aplicable a datos “en bruto” potencialmente ruidosos.

Los algoritmos de agrupamiento se pueden dividir en dos grupos: jerárquicos y particionales. Los algoritmos de agrupamiento jerárquico recursivamente encuentran grupos anidados en modo aglomerativo: comenzando con cada punto de datos en su propio grupo y fusionando el par más similar de grupos sucesivamente para formar una jerarquía de grupo; o en modo divisivo, de arriba hacia abajo: comenzando por todos los puntos de datos en un grupo y dividir recursivamente cada grupo en grupos más pequeños. En comparación con los algoritmos de agrupamiento jerárquico, los algoritmos de agrupamiento de particiones encuentran todos los grupos simultáneamente como una partición de los datos y no imponen una estructura jerárquica. La entrada a un algoritmo jerárquico es una matriz de similitud de $n \times n$, donde n es la cantidad de objetos que se agruparán. Por otro lado, un algoritmo de partición puede usar una matriz de patrón $n \times d$, donde n objetos están incrustados en un espacio de característica d -dimensional o una matriz de similitud $n \times n$.

Cabe destacar que no existe un algoritmo de agrupamiento “correcto” objetivamente pero, como se señaló, “el agrupamiento está en el ojo del espectador”. El algoritmo de agrupamiento más apropiado para un problema en particular a menudo debe elegirse experimentalmente, a menos que haya una razón matemática para preferir un modelo de agrupación sobre otro. Cabe señalar que un algoritmo que está diseñado para un tipo de modelo generalmente fallará en un conjunto de datos que contiene un tipo de modelo radicalmente diferente. [Estivill-Castro, 2002]

Para utilizar un algoritmo de agrupamiento para realizar análisis exploratorio de datos (AED), el algoritmo debe intentar respetar las siguientes reglas:

- No equivocarse: Si se está haciendo AED, se está tratando de aprender y obtener intuiciones sobre los datos. En ese caso, es mucho mejor no obtener ningún resultado que un resultado que sea incorrecto. Los malos resultados conducen a intuiciones falsas que a su vez envían a uno por completo al camino equivocado. No sólo no se entienden los datos, sino que se malinterpretan los datos. Esto significa que un buen algoritmo de agrupamiento para AED debe ser conservador en su agrupamiento; debería estar dispuesto a no asignar puntos a los clusters; no debe agrupar puntos a menos que realmente estén en un clúster.
- Parámetros intuitivos: todos los algoritmos de agrupamiento tienen parámetros; se necesita mover algunas perillas para ajustar las cosas. La pregunta es: ¿cómo se selecciona la configuración para esos parámetros? Si se sabe poco acerca de los datos, puede ser difícil determinar qué valor o configuración debe tener un parámetro. Esto significa que los parámetros deben ser lo suficientemente intuitivos como para poder establecerlos sin tener que saber mucho sobre los datos.
- Grupos estables: si se ejecuta el algoritmo dos veces con una inicialización aleatoria diferente, se debería esperar recuperar aproximadamente los mismos grupos. Si se está muestreando los datos, tomar una muestra aleatoria diferente no debe cambiar radicalmente la estructura del clúster resultante (a menos que el muestreo tenga problemas). Si se modifican los parámetros del algoritmo de agrupamiento, se pretende que el clúster cambie de una manera predecible y estable.
- Rendimiento: los conjuntos de datos con los que se trabaja en la realidad son cada vez más grandes. En estos casos, se puede sub-muestrear (dada cierta estabilidad), pero finalmente se necesita un algoritmo de agrupamiento que pueda escalar a tamaños de datos grandes. Un algoritmo de agrupamiento no es muy útil si sólo puede ser usado tomando una sub-muestra tan pequeña que ya no sea representativa de los datos en general.

En lo que sigue se dará una introducción muy breve a los algoritmos de clustering utilizados:

- K-Means. El algoritmo de K-Means [Lloyd, 1982] agrupa los datos tratando de separar las muestras en k grupos de igual varianza, minimizando un criterio conocido como inercia o suma de cuadrados dentro del grupo. Este algoritmo requiere que se especifique la cantidad de grupos. Se adapta bien a una gran cantidad de muestras y se ha utilizado en una amplia gama de áreas de aplicación en muchos campos diferentes.
- Affinity Propagation. Affinity Propagation [Frey and Dueck, 2007] crea clústeres enviando mensajes entre pares de muestras hasta la convergencia. Luego, se describe un conjunto de datos utilizando un pequeño número de ejemplares, que se

identifican como los más representativos de otras muestras. Los mensajes enviados entre pares representan la capacidad para que una muestra sea el ejemplo de la otra, que se actualiza en respuesta a los valores de otros pares. Esta actualización ocurre de forma iterativa hasta la convergencia, momento en el que se eligen los ejemplos finales y, por lo tanto, se da la agrupación final.

- Spectral Clustering. Spectral Clustering realiza un “embedding” de baja dimensión de la matriz de afinidad entre muestras, seguida de un K-Means en el espacio de baja dimensión. Spectral Clustering requiere que se especifique la cantidad de grupos. Funciona bien para una pequeña cantidad de clusters, pero no se recomienda cuando se usan muchos grupos.
- Agglomerative Clustering. El algoritmo Agglomerative Clustering realiza una agrupación jerárquica utilizando un enfoque ascendente: cada observación comienza en su propio grupo y los clústeres se fusionan sucesivamente. Los criterios de vinculación determinan la métrica utilizada para la estrategia de combinación. En nuestro caso utilizamos la vinculación promedio que minimiza el promedio de las distancias entre todas las observaciones de pares de grupos.
- Gaussian Mixture. Un modelo de mezcla gaussiano (en inglés, Gaussian Mixture) es un modelo probabilístico que supone que todos los puntos de datos se generan a partir de una mezcla de un número finito de distribuciones gaussianas con parámetros desconocidos. Uno puede pensar en modelos de mezclas como la generalización de clusters K-Means para incorporar información sobre la estructura de covarianza de los datos, así como los centros de los gaussianos latentes.
- DBSCAN. El algoritmo DBSCAN [Ester et al., 1996] ve los grupos como áreas de alta densidad separadas por áreas de baja densidad. Debido a esta visión bastante genérica, los grupos encontrados por DBSCAN pueden ser de cualquier forma, a diferencia de K-Means, que asume que los clusters tienen forma convexa. El componente central del DBSCAN es el concepto de muestras de núcleo, que son muestras que se encuentran en áreas de alta densidad. Un grupo es, por lo tanto, un conjunto de muestras de núcleos, cada una cerca una de otra (medida por alguna medida de distancia) y un conjunto de muestras no centrales que están cerca de una muestra central (pero no son muestras de núcleo).
- HDBSCAN. HDBSCAN [McInnes et al., 2017] amplía DBSCAN convirtiéndolo en un algoritmo de agrupación jerárquica y luego utiliza una técnica para extraer un clúster plano basado en la estabilidad de los clústeres.

3. Lista de Clusters

En esta sección se presenta el algoritmo Lista de Clusters, se muestra cómo construir la lista de grupos, los distintos parámetros que controlan su comportamiento y cómo se va a hacer uso del mismo.

El algoritmo Lista de Clusters [Chavez and Navarro, 2000] es una técnica para indexar espacios métricos de gran dimensión. Se basa en la construcción de una lista de grupos, cada uno dado por un radio cobertor y un centro. Para su construcción, se debe elegir un centro $c \in D$ donde D es la colección de datos, y un radio r_c . Una bola centro (c, r_c) es el subconjunto de elementos de D que están a lo más a r_c de c . Se definen los siguientes subconjuntos de D :

$$I_{D,c,r_c} = \{x \in D - \{c\}, d(c, x) \leq r_c\}$$

como el grupo de elementos internos que son parte de la bola (c, r_c) , y

$$E_{D,c,r_c} = x \in D, d(c, x) > r_c$$

como los elementos externos. Una vez construidos el subconjuntos I_{D,c,r_c} y E_{D,c,r_c} , se vuelve a aplicar el mismo procedimiento con los elementos que terminaron en E_{D,c,r_c} . Cabe destacar que el primer centro elegido tiene preferencia sobre el resto cuando hay solapamiento. De este modo los elementos que pertenecen a la bola del primer centro son almacenados sólo en su cluster I , a pesar de que haya intersección con los grupos siguientes.

Hay dos formas de particionar el espacio: establecer un radio fijo para cada partición o adoptar un tamaño fijo de elementos para cada grupo. Mientras tanto, para la selección del centro existen distintas heurísticas:

- p1: aleatoriedad.
- p2: el elemento más cercano al centro anterior en el conjunto restante.
- p3: el elemento más alejado al centro anterior en el conjunto restante.
- p4: el elemento que minimice la suma de las distancias a los centros previos.
- p5: el elemento que maximice la suma de las distancias a los centros previos.

La configuración final que se elija depende del problema que se está resolviendo.

En nuestro caso no nos interesa la forma en la que se realiza la búsqueda en la Lista de Clusters, ya que no se utiliza para el agrupamiento de datos. Para llevar a cabo la tarea de construcción de grupos lo único que debemos hacer es construir la Lista de Clusters sobre el conjunto de datos que disponemos. Los distintos grupos estarán dados por los clusters generados.

4. Diseño experimental

A continuación se presentan los experimentos realizados para comparar el desempeño de la Lista de Clusters con respecto a algoritmos clásicos del problema de agrupamiento. Se brinda una descripción de los distintos conjuntos de datos sobre los que se evaluaron los algoritmos. Se presentan los algoritmos que fueron comparados juntos con sus parámetros. Además, se analiza la mejor combinación de parámetros para el algoritmo

de Lista de Clusters. Finalmente, se muestran los grupos obtenidos por cada algoritmo en cada conjunto de datos.

Para llevar a cabo las pruebas se utilizó el lenguaje de programación Python y en particular las “notebooks” interactivas de Jupyter [Kluyver et al., 2016]. Gran parte del pre-procesamiento de los datos y la construcción de los conjuntos de datos se llevó a cabo usando la librería scikit-learn [Pedregosa et al., 2011]. Además, ésta tiene implementados gran cantidad de algoritmos de agrupamientos que fueron utilizados para realizar este informe. Por último, con el fin de mostrar gráficamente los resultados se hizo uso de la librería matplotlib [Hunter, 2007].

4.1. Conjuntos de datos

Para evaluar el desempeño de los distintos algoritmos se construyeron siete conjuntos de datos. Cada conjunto de datos cuenta con mil quinientos puntos en \mathbb{R}^2 . A continuación se describen brevemente cada uno:

- **Círculos:** se trata de dos grupos en forma de círculo uno dentro del otro.
- **Lunas:** se trata de dos grupos en forma de lunas, representadas gráficamente por la fase creciente de la luna, enfrentadas.
- **Manchas con ruido:** se trata de tres grupos de manchas y puntos que corresponden a ruido que no deberían pertenecer a ningún grupo.
- **Manchas variantes:** se trata de tres grupos de manchas con centros y desviaciones estándares determinadas.
- **Agujas:** se trata de tres grupos de agujas.
- **Manchas fáciles:** se trata de tres grupos de manchas fácilmente distinguibles.
- **Aleatorio:** se trata de puntos aleatoriamente generados. No existe una buena agrupación de los puntos.

En la Figura 2 se pueden observar los distintos conjuntos de datos sin procesar.

4.2. Algoritmos a comparar

Los algoritmos que se escogieron para la comparación fueron los siguientes: K-Means, Affinity Propagation, Spectral Clustering, Agglomerative Clustering, Gaussian Mixture, DBSCAN, HDBSCAN y Lista de Clusters.

Los parámetros que mejores agrupamientos generaron fueron obtenidos para todos los algoritmos para cada conjunto de datos. Para los algoritmos normalmente utilizados para el problema de clustering, los parámetros fueron obtenidos de la documentación de scikit-learn¹. En cambio, para el caso del algoritmo Lista de Clusters estos valores se tuvieron que obtener por experimentación.

¹http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html

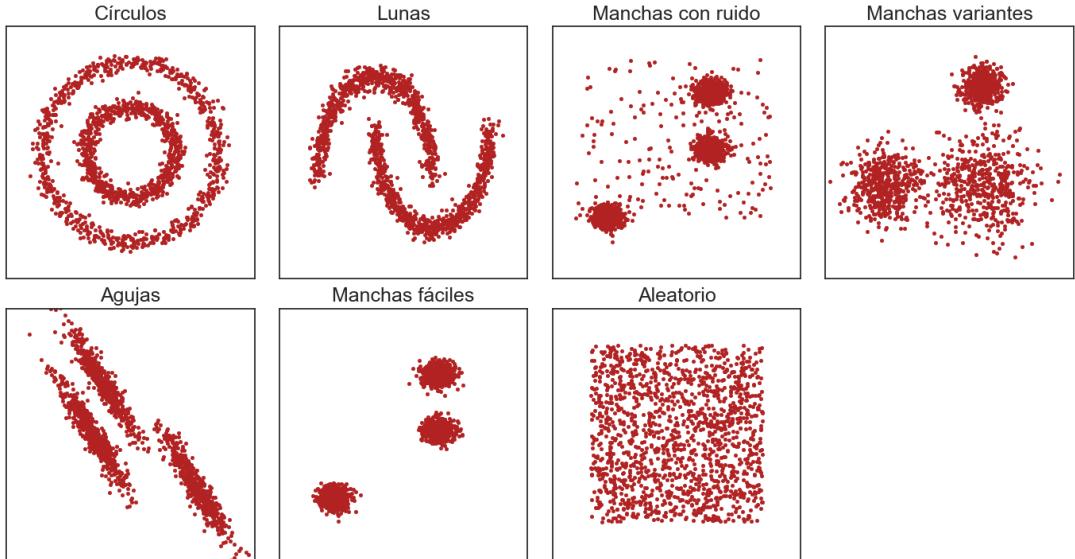


Figura 2: Conjuntos de datos sin procesar.

Debido a que no existe una implementación del algoritmo Lista de Clusters en el lenguaje de programación Python, se llevó a cabo su desarrollo. Éste se puede encontrar anexo al informe en el archivo `list_of_clusters.py`. Observando los distintos conjuntos de datos, se determinaron los valores óptimos para los parámetros `fixed_size` y `fixed_radius`. El valor del primero fue calculado de la siguiente forma: $\text{fixed_size} = \text{cantidad_puntos}/\text{numero_grupos}$. Así, para aquellos conjuntos de datos con dos grupos se utilizó `fixed_size = 750`, mientras que para los de tres grupos se usó `fixed_size = 500`. El valor de `fixed_radius`, en cambio, se obtuvo a ojo en cada conjunto de datos, seleccionando aquel valor que agrupara de mejor forma. Finalmente, para cada conjunto de datos se evaluaron todas las posibilidades en parámetros. Esto es, se probaron todas las combinaciones siguientes:

- Heurística de selección del centro: `{'p1', 'p2', 'p3', 'p4', 'p5'}`.
- Grupos de tamaño definido o de radio definido.
- Métrica de distancia: `{'euclidean', 'cityblock'}`.

Las combinaciones posibles para cada conjunto de datos se pueden encontrar en las Figuras 3 y 4. El código utilizado para generar estas gráficas de forma interactiva se encuentra anexo en el archivo `Params_LC.ipynb`²; si, en cambio, sólo se quiere mirar el código fuente se puede utilizar el archivo `Params_LC.html`. Observando las distintas agrupaciones, podemos concluir que los mejores parámetros para cada conjunto de datos son los que se muestran en el Cuadro 1.

Los parámetros que mayor impacto tuvieron en el agrupamiento fueron los que determinan los elementos que pertenecen a un grupo, esto es: `fixed_size` y `fixed_radius`. Para

²Se trata de una notebook interactiva de Jupyter.

Cuadro 1: Mejores parámetros de Lista de Clusters para cada conjunto de datos.

conjunto de datos	fixed_size	fixed_radius	center_choise	distance_metric
círculos	750	-	p1	euclidean
lunas	-	1,75	p3	euclidean
manchas con ruido	500	-	p1	euclidean
manchas variantes	-	1,4	p1	euclidean
agujas	500	-	p4	cityblock
manchas fáciles	500	-	p3	euclidean
aleatorio	-	1,0	p1	euclidean

la mayoría de los conjuntos de datos el uso de fixed_size mostró ser más efectivo, esto ya que permite aprovechar el conocimiento previo que tenemos de los grupos: la cantidad aproximada de elementos en cada grupo. La heurística de selección de centro no tuvo mucho peso debido a que los conjuntos de datos tienen baja cantidad de puntos bastantes cercanos en el espacio; sin embargo se puede ver que seleccionar los centro más alejados es mejor que los más cercanos, ya que produce menor cantidad de grupos. En cuanto a la métrica de distancia, la distancia euclídea fue la escogida en su mayoría ya que, cuando los datos se encuentran en dimensión \mathbb{R}^2 , genera círculos: forma que tienen la mayoría de los grupos. Para el caso del conjunto de datos “agujas” se utilizó la distancia de Manhattan ya que captura la pendiente de los grupos.

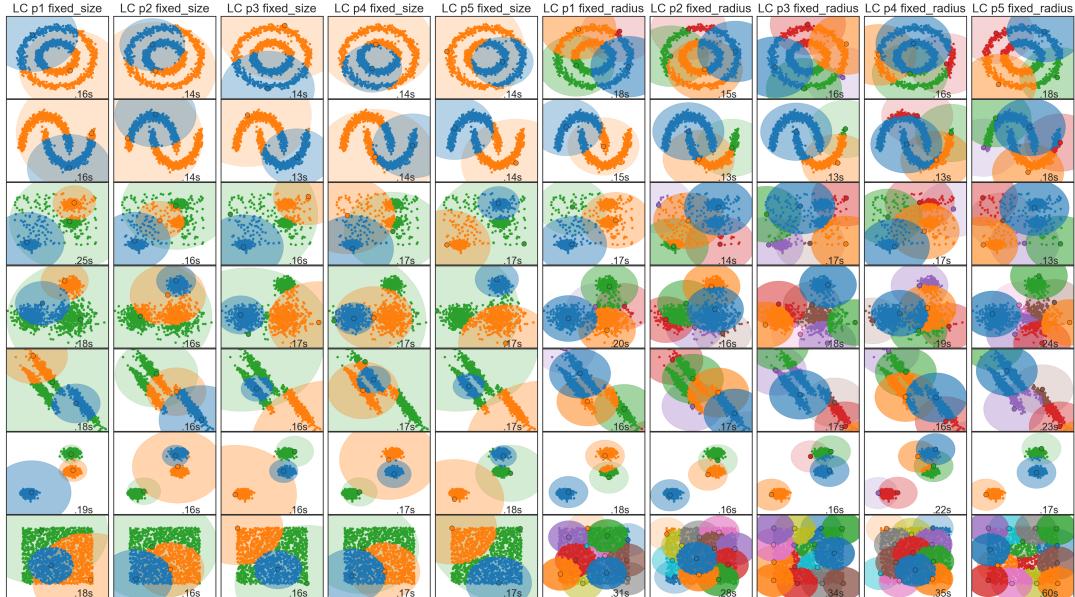


Figura 3: Comparación de distintos parámetros del algoritmo Lista de Clusters usando la distancia euclidiana.

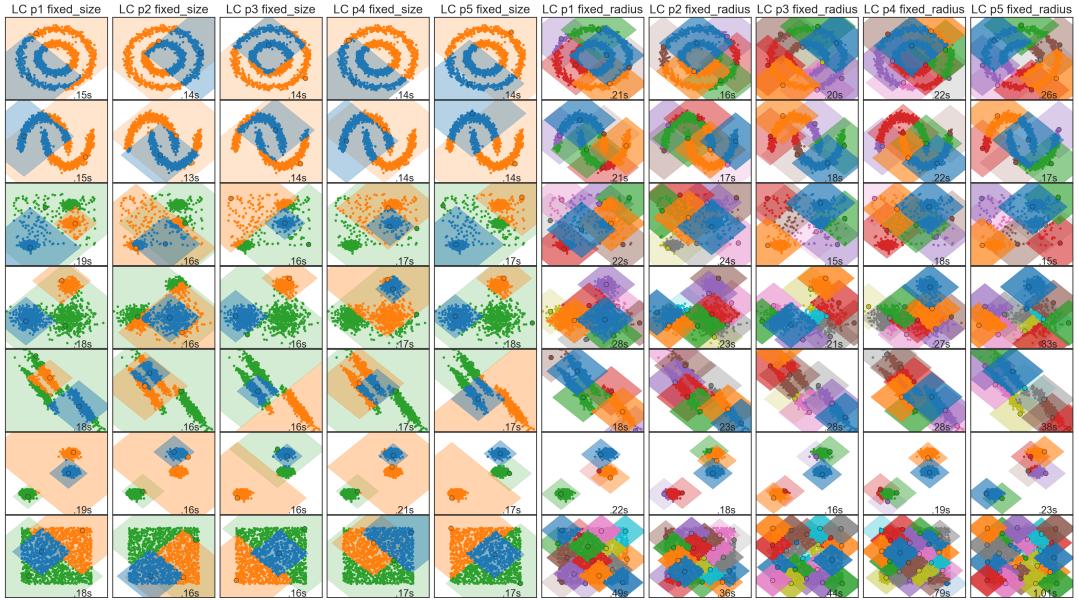


Figura 4: Comparación de distintos parámetros del algoritmo Lista de Clusters usando la distancia de Manhattan.

4.3. Resultados

Una vez obtenidos los mejores parámetros para cada algoritmo, se los comparó en los distintos conjuntos de datos. El resultado de este agrupamiento se puede observar en la Figura 5. El código utilizado para generar estas gráficas de forma interactiva se encuentra anexo en el archivo `List_of_Clusters_vs_Others.ipynb`; si, en cambio, sólo se quiere mirar el código fuente se puede utilizar el archivo `List_of_Clusters_vs_Others.html`.

Se observa que el desempeño del algoritmo Lista de Clusters es regular: sólo para el conjunto de datos “manchas fáciles” se generaron los grupos correctos. Los grupos que éste genera son similares a los que el algoritmo K-Means construyó. Sin embargo, cuando se compara la Lista de Clusters con el algoritmo que mejor desempeño tuvo, HDBSCAN, se nota una diferencia considerable; este último es capaz de encontrar todos los grupos e incluso de descartar los elementos que representan ruido³. Los únicos algoritmos, de los que se evaluaron, con esta propiedad de no agrupar a todos los elementos, dejando el ruido o los datos aberrantes sin grupo, son DBSCAN y HDBSCAN.

El tiempo requerido para construir los distintos grupos de cada algoritmo en cada conjunto de datos se puede apreciar en la Figura 6. De este gráfico, se puede concluir que el algoritmo Lista de Clusters tiene tiempos promedios. Los algoritmos Affinity Propagation y Spectral Clustering son considerablemente más lentos que la Lista de Clusters; para el caso de Affinity Propagation esto se debe a su alta complejidad en tiempo: $O(N^2T)$ donde N es el número de elementos y T es la cantidad de iteraciones

³Elementos graficados en color negro.

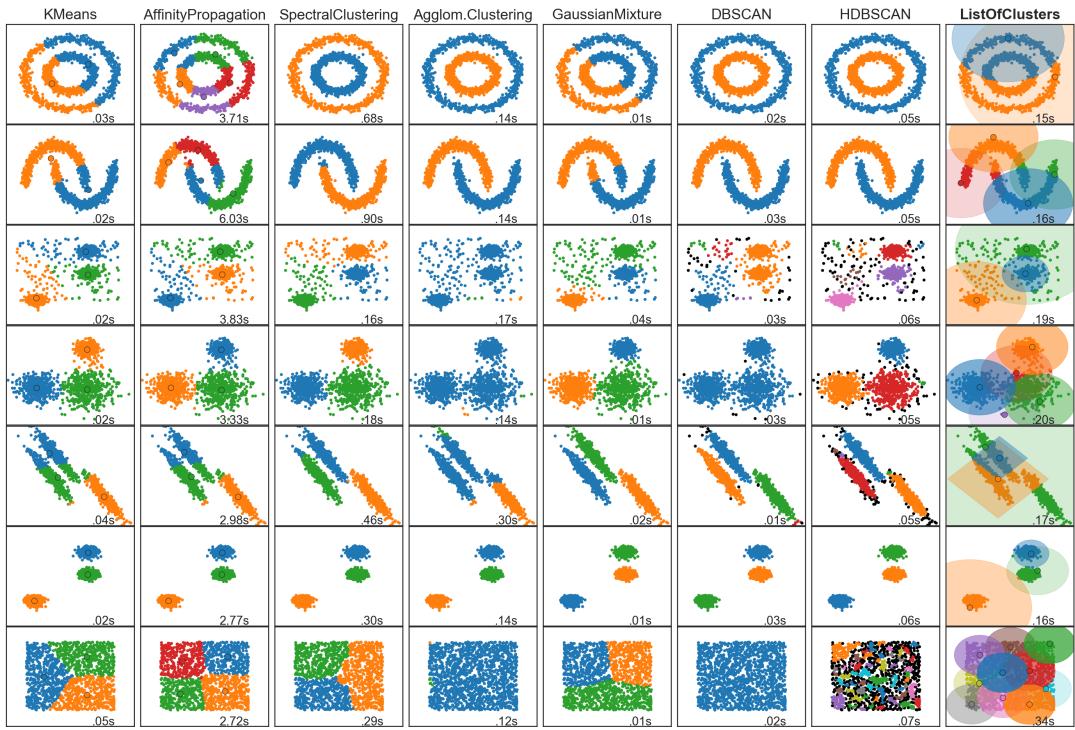


Figura 5: Comparación del algoritmo Lista de Clusters frente a otros.

para converger al agrupamiento final. Sin embargo, si se compara Lista de Clusters frente a los algoritmos más veloces como K-Means, vemos que la Lista de Clusters tardó en promedio siete veces más que K-Means. Esto se puede deber a una falta de optimización del código que implementa el algoritmo de Lista de Clusters. Un análisis más profundo de las complejidades de los algoritmos sería necesario para poder comparar de forma justa los algoritmos.

5. Conclusión

Se ha mostrado la aplicación del algoritmo Lista de Clusters, normalmente utilizado para indexar espacios métricos de gran dimensión, en el problema de agrupamiento de datos, relacionado al área del aprendizaje automático. Para esto se implementó el algoritmo en el lenguaje de programación Python y se realizaron numerosas pruebas para determinar los mejores parámetros para cada conjunto de datos. Finalmente, se comparó su desempeño con respecto a algoritmos comúnmente utilizados para el agrupamiento de datos en el aprendizaje automático. La Lista de Clusters mostró un desempeño comparable a K-Means considerando los grupos construidos; aunque, si se considera el tiempo requerido para generarlos, la Lista de Clusters tardó en promedio siete veces más que K-Means. Y si se compara la Lista de Clusters con el algoritmo HDBSCAN, se ve que los agrupamientos generados son de calidades muy distintas esta vez y que sigue siendo

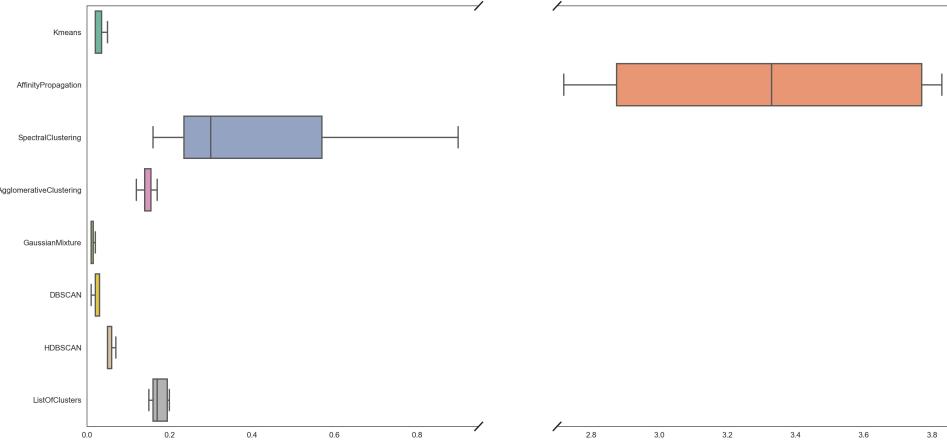


Figura 6: Tiempos de ejecución de los algoritmos medido en segundos.

más lento. HDBSCAN construye los grupos correctos y deja fuera de todo grupo a los puntos que son ruido. Esta diferencia de desempeño en cuanto a los grupos generados se debe a que el algoritmo Lista de Clusters no fue diseñado para generar agrupamientos; mientras que la diferencia de velocidad de los algoritmos puede deberse, en parte, a una falta de optimización del código de Lista de Clusters.

Como trabajo futuro se podría evaluar el desempeño de la Lista de Clusters para conjuntos de datos con mayor cantidad de dimensiones y con mayor cantidad de elementos. Pero para esto es necesario que previamente la implementación sea optimizada. Además, se podrían implementar otras medidas de distancia como la familia de distancias de Minkowski (forma general de la distancia euclídea y de Manhattan), la distancia de hamming, etc.

Referencias

- [Chavez and Navarro, 2000] Chavez, E. and Navarro, G. (2000). An effective clustering algorithm to index high dimensional metric spaces. In *Proceedings Seventh International Symposium on String Processing and Information Retrieval. SPIRE 2000*, pages 75–86.
- [Ester et al., 1996] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). Density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, KDD’96*, pages 226–231. AAAI Press.
- [Estivill-Castro, 2002] Estivill-Castro, V. (2002). Why so many clustering algorithms: A position paper. *SIGKDD Explor. Newsl.*, 4(1):65–75.

- [Frey and Dueck, 2007] Frey, B. J. and Dueck, D. (2007). Clustering by passing messages between data points. *Science*, 315.
- [Hunter, 2007] Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.
- [Jain, 2010] Jain, A. K. (2010). Data clustering: 50 years beyond k-means. *Pattern Recogn. Lett.*, 31(8):651–666.
- [Kluyver et al., 2016] Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., and Willing, C. (2016). Jupyter notebooks – a publishing format for reproducible computational workflows. In Loizides, F. and Schmidt, B., editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press.
- [Lloyd, 1982] Lloyd, S. (1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137.
- [McInnes et al., 2017] McInnes, L., Healy, J., and Astels, S. (2017). hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11).
- [Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.