Clustering

Artículo de Inderjit Dhillon, Yuqiang Guan y Brian Kulis: *A Unified View of Kernel k-means, Spectral Clustering and Graph Cuts*, UTCS Technical Report, 2005.

Se introducen y unifican dos métodos comúnmente utilizados para agrupar datos no linealmente separables: el algoritmo kernel k-means y la partición de grafos.
Algoritmo Kernel K-Means:
El algoritmo kernel k-means es una extensión del algoritmo k-means estándar. A diferencia del k-means estándar, el kernel k-means mapea puntos de datos a un espacio de mayor dimensión, lo que le permite identificar grupos que no son linealmente separables en el espacio de entrada original. Esto es una ventaja significativa, especialmente cuando se le proporciona una matriz definida positiva de valores de similitud.
Algoritmos de Partición de Grafos:
Los algoritmos de partición de grafos se centran en agrupar nodos en un grafo. Los métodos espectrales, que incluyen el corte de proporción y el corte normalizado, se utilizan comúnmente para objetivos de partición de grafos. Estos métodos tienen aplicaciones en diversos campos, como el diseño de circuitos y la segmentación de imágenes.
En casos en los que el cálculo de eigenvectores es prohibitivo (por ejemplo, si se requieren muchos eigenvectores de una matriz grande), el algoritmo weighted kernel k-means puede ser más deseable que los métodos espectrales. En situaciones en las que el cálculo de eigenvectores es factible, se pueden utilizar los métodos espectrales para una inicialización efectiva de los clústeres. Posteriormente, nuestro análisis sugiere que mediante una elección adecuada de pesos y kernel, podemos utilizar kernel k-means para mejorar de manera monótona objetivos específicos de partición de grafos, como el ratio cut, normalized cut, ratio association y el objetivo de Kernighan-Lin.
Aunque kernel k-means tiene la ventaja de poder agrupar datos que no son linealmente separables, no ha tenido un gran impacto en la comunidad de investigación para aplicaciones específicas. Se ha trabajado en escalar kernel k-means a conjuntos de datos grandes, pero el algoritmo no se ha aplicado a muchos problemas del mundo real. Nuestro resultado proporciona evidencia convincente de que weighted kernel k-means es, de hecho, un enfoque potente y útil para la agrupación, especialmente para la agrupación de grafos.
K-Means
El algoritmo K-Means, dado un conjunto de vectores $\{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_n\}$, busca encontrar clusters $\{\pi_1, \pi_2, ..., \pi_k\}$ que minimicen la función objetivo:

$$\mathcal{D}(\{\pi_c\}_{c=1}^k) = \sum_{c=1}^{k} \sum_{\mathbf{a}_i \in \pi_c} \|\mathbf{a}_i - \mathbf{m}_c\|^2$$

Aquí, $\mathbf{m}_c$ representa el centroide del cluster π_c, calculado como la media de los vectores en ese cluster:

$$\mathbf{m}_c = \frac{1}{|\pi_c|} \sum_{\mathbf{a}_i \in \pi_c} \mathbf{a}_i$$

Es importante notar que el conjunto c-ésimo se denota como π_c, y la agrupación o particionamiento se representa por $\{\pi_c\}_{c=1}^k$.
Una desventaja del K-Means estándar es que los clusters deben estar separados por un hiperplano; esto se debe al hecho de que se utiliza la distancia euclidiana al cuadrado como medida de distorsión. Para contrarrestar esto, el Kernel K-Means utiliza una función para mapear puntos a un espacio de características de mayor dimensión. Cuando se aplica K-Means en este

espacio de características, los separadores lineales en dicho espacio corresponden a separadores no lineales en el espacio de entrada.
La función objetivo del Kernel K-Means se puede expresar como una minimización de:

$$\mathcal{D}(\{\pi_c\}_{c=1}^{k}) = \sum_{c=1}^{k} \sum_{\mathbf{a}_i \in \pi_c} \|\phi(\mathbf{a}_i) - \mathbf{m}_c\|^2$$

Aquí, $\mathbf{m}_c$ se calcula de manera similar al caso de K-Means estándar, pero se utiliza la función de mapeo ϕ para llevar los puntos al espacio de características de mayor dimensión.

Complejidad computacional
La complejidad computacional del algoritmo ponderado del kernel k-means se analiza a continuación. Este algoritmo, presentado como Algoritmo 1, es una generalización directa del k-means estándar. Al igual que en el k-means estándar, dada la posición actual de los centroides, se calcula el centroide más cercano para cada punto. Después de esto, se vuelve a calcular el agrupamiento. Estos dos pasos se repiten hasta que el cambio en el valor de la función objetivo es suficientemente pequeño. Se puede demostrar que este algoritmo converge de manera monótona siempre y cuando $\mathbf{K}$ sea semi-definida positiva para que se pueda interpretar como una matriz de Gram.
Es evidente que el cuello de botella en el algoritmo del kernel k-means es el Paso 2, es decir, el cálculo de las distancias $\phi(\mathbf{a}_i, \mathbf{m}_c)$. El primer término, K_{ii}, es una constante para $\mathbf{a}_i$ y no afecta la asignación de $\mathbf{a}_i$ a los clústeres. El segundo término requiere $O(n)$ cálculos para cada punto de datos, lo que lleva a un costo de $O(n^2)$ por iteración. El tercer término es fijo para el clúster c, por lo que en cada iteración se puede calcular una vez y almacenar; en todos los clústeres, esto toma $O(n^2)$ operaciones. Por lo tanto, la complejidad es de $O(n^2)$ operaciones escalares por iteración.
Sin embargo, para una matriz $\mathbf{K}$ dispersa, cada iteración se puede adaptar para tener un costo de $O(nz)$ operaciones, donde nz es el número de entradas no nulas de la matriz ($nz = n^2$ para una matriz de kernel densa). Si estamos utilizando una función de kernel κ para generar la matriz de kernel a partir de nuestros datos, calcular $\mathbf{K}$ generalmente toma tiempo $O(n^2 m)$, donde m es la dimensionalidad de los puntos originales. Si el número total de iteraciones es τ, entonces la complejidad temporal del Algoritmo 1 es $O(n^2(\tau + m))$ si se dan vectores de datos como entrada, o $O(nz\tau)$ si se da una matriz definida positiva como entrada.

Partición de grafos
Ahora cambiamos nuestro enfoque hacia un enfoque aparentemente muy diferente para agrupar datos: la partición de grafos. En este modelo de agrupamiento, se nos proporciona un grafo $G = (V, E, A)$, que está compuesto por un conjunto de vértices V y un conjunto de aristas E de tal manera que una arista entre dos vértices representa su similitud. La matriz de afinidad A es de tamaño $|V| \times |V|$ cuyas entradas representan los pesos de las aristas (una entrada de A es 0 si no hay arista entre los vértices correspondientes).
Denotamos links(A, B) como la suma de los pesos de las aristas entre nodos en A y B. En otras palabras,

$$\text{links}(A, B) = \sum_{i \in A, j \in B} A_{ij}.$$

Además, definimos el grado de A como los enlaces de los nodos en A con todos los vértices, es decir,

$$\text{degree}(A) = \text{links}(A, V).$$

El problema de la partición de grafos busca dividir el grafo en k particiones o clústeres disjuntos $V_1, \ldots, V_k$ de manera que su unión sea V. Se han propuesto y estudiado varios objetivos diferentes para la partición de grafos, y nos centraremos en los más prominentes.

1. **Ratio Association (Asociación de Ratio):** El objetivo de Ratio Association (también llamada asociación promedio) busca maximizar la asociación dentro del clúster en relación con el tamaño del clúster. El objetivo se expresa como:

$$RAssoc(G) = \max_{V_1,\ldots,V_k} \sum_{c=1}^{k} \frac{\text{links}(V_c, V_c)}{|V_c|}$$

2. **Ratio Cut (Corte de Ratio):** Este objetivo difiere de la asociación de ratio en que busca minimizar el corte entre clústeres y los vértices restantes. Se expresa como:

$$RCut(G) = \min_{V_1,\ldots,V_k} \sum_{c=1}^{k} \frac{\text{links}(V_c, V \setminus V_c)}{|V_c|}$$

3. **Objetivo Kernighan-Lin:** Este objetivo es casi idéntico al objetivo de corte de ratio, excepto que se requiere que las particiones tengan tamaños iguales. Aunque este objetivo se presenta generalmente para $k = 2$ clústeres, se puede generalizar fácilmente para k arbitrario. Para simplificar, asumimos que el número de vértices $|V|$ es divisible por k. Entonces, escribimos el objetivo como:

$$KLObj(G) = \min_{V_1,\ldots,V_k} \sum_{c=1}^{k} \frac{\text{links}(V_c, V \setminus V_c)}{|V_c|}$$

   sujeto a $|V_c| = |V|/k$ para todo $c = 1, \ldots, k$.

4. **Normalized Cut (Corte Normalizado):** El objetivo de normalized cut es uno de los objetivos de partición de grafos más populares y busca minimizar el corte relativo al grado de un clúster en lugar de su tamaño. El objetivo se expresa como:

$$NCut(G) = \min_{V_1,\ldots,V_k} \sum_{c=1}^{k} \frac{\text{links}(V_c, V \setminus V_c)}{\text{degree}(V_c)}$$

   Cabe destacar que el problema de corte normalizado es equivalente al problema de asociación normalizado, ya que

$$\text{links}(V_c, V \setminus V_c) = \text{degree}(V_c) - \text{links}(V_c, V_c).$$

5. **Cortes/Asociación de Grafos Ponderados Generalizados:** Podemos generalizar los problemas de asociación y corte a variantes ponderadas más generales. Esto resultará útil para establecer una conexión general con el kernel k-means ponderado. Introducimos un peso w_i para cada nodo del grafo y, para cada clúster V_c, definimos

$$\text{w}(V_c) = \sum_{i \in V_c} w_i.$$

   Generalizamos el problema de asociación de la siguiente manera:

$$WAssoc(G) = \max_{V_1,\ldots,V_k} \sum_{c=1}^{k} \mathrm{w}(V_c)\mathrm{links}(V_c, V_c)$$

De manera similar, para los cortes:

$$WCut(G) = \min_{V_1,\ldots,V_k} \sum_{c=1}^{k} \mathrm{w}(V_c)\mathrm{links}(V_c, V \setminus V_c)$$

La ratio association es un caso especial de $WAssoc$ donde todos los pesos son iguales a uno (y, por lo tanto, el peso de un clúster es simplemente el número de vértices en él), y la asociación normalizada es un caso especial donde el peso de un nodo i es igual a su grado (la suma de la fila i de la matriz de afinidad A). Lo mismo es cierto para $WCut$.

Poniendo todo junto

En esta sección, se abordan algunos problemas importantes pendientes. Se destaca el uso de métodos espectrales como paso de inicialización para objetivos de partición de grafos. Se discuten dos métodos para obtener una agrupación discreta a partir de eigenvectores y se generalizan estos métodos para obtener agrupaciones iniciales para el weighted kernel k-means. Para mejorar la calidad de los resultados con el weighted kernel k-means, se examina el uso de la búsqueda local como una forma efectiva de evitar óptimos locales. Combinando estas estrategias, se presenta un algoritmo mejorado para el weighted kernel k-means.
En la sección 5.1, se describen dos métodos para obtener particiones a partir de eigenvectores: el postprocesamiento de Bach y Jordan, que generaliza el método para todos los casos de weighted kernel k-means, y el postprocesamiento de Yu y Shi, una alternativa para cortes normalizados.
En la sección 5.2, se aborda el problema de quedar atrapado en óptimos locales al ejecutar k-means estándar o kernel k-means, proponiendo el uso de búsqueda local mediante incremental weighted kernel k-means para mover puntos entre clusters y mejorar la función objetivo.
En la sección 5.3, se presenta el algoritmo final que consta de cuatro etapas: configuración de pesos y kernel, inicialización de clusters, ajuste de la matriz K para que sea definida positiva y la ejecución alternante de batch weighted kernel k-means y incremental weighted kernel k-means hasta la convergencia.
En la siguiente sección se presentan los resultados experimentales de un estudio que aborda la partición de grafos y el problema de k-medias ponderado en el ámbito de los núcleos (kernel k-means). Los experimentos ilustran varias implicaciones de los resultados obtenidos:

- Se demuestra que realizar un desplazamiento negativo (σ) en la diagonal de la matriz del núcleo puede mejorar significativamente los resultados de agrupamiento en aplicaciones de clustering de documentos.
- La partición de grafos utilizando el algoritmo final proporciona una mejora sobre los métodos espectrales cuando se utiliza una inicialización espectral, y en algunos casos, la inicialización con METIS es tan efectiva como la inicialización espectral.
- Se muestra que la segmentación de imágenes mediante el corte normalizado puede llevarse a cabo sin la necesidad de utilizar eigenvectores.

Los algoritmos se implementaron en MATLAB, y se realizaron experimentos en una máquina Linux con un procesador Pentium IV de 2.4 GHz y 1 GB de memoria principal.

En la sección de clustering de documentos, se observa que aplicar un desplazamiento negativo en la diagonal de la matriz del núcleo mejora el rendimiento en comparación con la ausencia de dicho desplazamiento. Se utilizan conjuntos de documentos C30, C150 y C300, y se evalúa el rendimiento mediante la información mutua normalizada (NMI), que indica la correspondencia entre los resultados de clustering y las etiquetas de clase reales.
En la sección de partición de grafos, se evalúa el rendimiento de la k-medias ponderada mediante la mejora en los valores de la asociación normalizada y la asociación normalizada, utilizando inicializaciones aleatorias, METIS y espectrales. Se emplean nueve grafos de prueba, y los resultados muestran que, a pesar de que METIS y la inicialización espectral son métodos de inicio superiores, el algoritmo logra mejorar significativamente los resultados incluso con inicialización aleatoria.
En resumen, los experimentos destacan la efectividad de las estrategias propuestas en la mejora de los resultados de clustering y partición de grafos en diversas aplicaciones.
En conclusión, el trabajo presentado se inspira en avances recientes en clustering espectral y métodos de relajación. Se destaca el enfoque de reformular la función objetivo de k-medias mediante la maximización de trazas de la matriz Gram, especialmente en el contexto de kernels para separadores no lineales. Se aborda la versión ponderada de kernel k-medias, que amplía la capacidad para cumplir con varios objetivos de clustering espectral, como la minimización del corte normalizado.
Comparado con trabajos anteriores, se resalta la eficiencia de los métodos propuestos, evitando costosas factorizaciones de matrices en el caso de kernel k-medias. Se menciona la contribución inicial de usar kernels para mejorar la función objetivo de k-medias y su aplicación en componentes no lineales.
Se discuten investigaciones previas sobre reformulación de la función objetivo de kernel k-medias, y se compara con los enfoques propuestos. Se destaca la necesidad de imponer positividad definida en el clustering, abordando el problema de incrustar puntos basados en matrices de disimilitud en el espacio euclidiano.
El artículo actual expande trabajos previos al introducir la noción de kernel k-medias ponderado y explorar objetivos adicionales, como corte de ratio y asociación de ratio. Se presentan resultados experimentales en grafos grandes y segmentación de imágenes para respaldar la eficacia de los métodos propuestos.

Implementación de métodos

Para esta parte se implementaron 5 métodos para poder ser comparados, esto con el fin de poder observar la eficiencia del metodo de Kernel K-Means.
Primero los datos que utlizaremos fueron generados por la libreria de sklearn.dataset, y son un claro ejemplo de un comportamiento no lineal debido a que forman dos medias lunas y en el otro caso son unos datos tipo dona. Los datos se muestran como sigue:

Vemos en este grupo de datos a primera vista dos grupos muy predominantes, los cúales son las dos medias lunas, y usamos estos datos ya que es clara la separación y con ello podemos poner a prueba los metodos que se enunciarán a continuación.
Para esta prueba de métodos, primero daremos los resultados dados por 5 métodos diferentes de clustering, iniciaremos con métodos clasicos y depsues avanzaremos con métodos mas profundos vistos en clases.

El método de Kernel K-Means es una extensión del algoritmo K-Means que utiliza una función de kernel para permitir la agrupación en espacios no lineales. Aunque puede ser efectivo en ciertos casos, puede presentar desafíos cuando se utilizan conjuntos de datos complejos como el de las dos medias lunas que has generado.

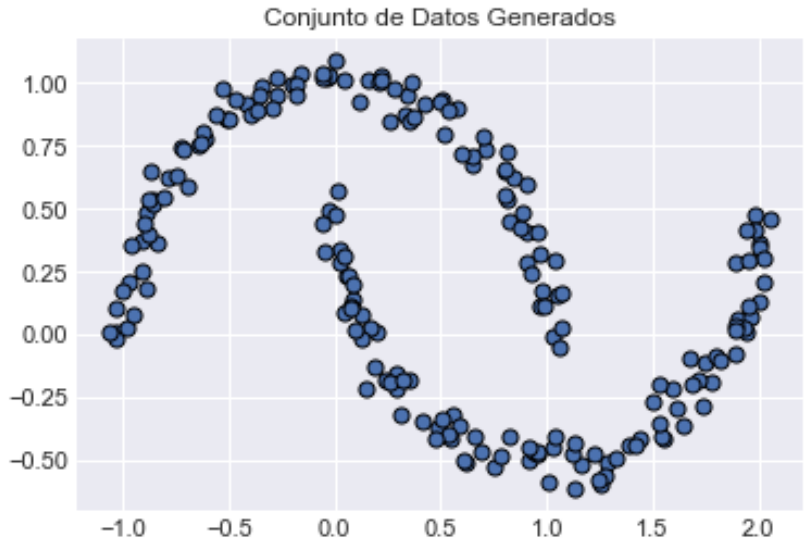


Figura 0.1: Datos generados para la prueba de kernel k means, estos datos fueron hechos con la libreria de sklearn en python tipo espiral

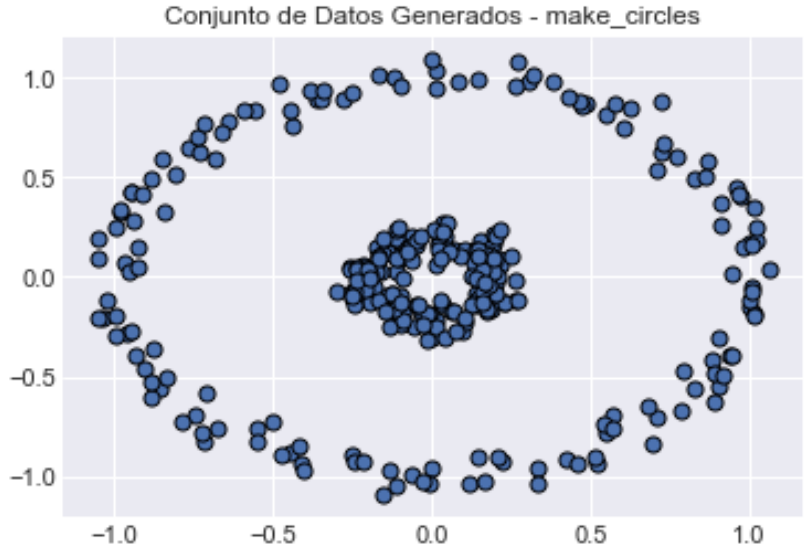


Figura 0.2: Datos generados para la prueba de kernel k means, estos datos fueron hechos con la libreria de sklearn en python tipo dona

Aquí hay algunas razones por las cuales el método de Kernel K-Means puede no haber funcionado bien es porque:
La elección del kernel es crucial en el método de Kernel K-Means. Para conjuntos de datos no lineales como las dos medias lunas, es común utilizar kernels como el kernel radial (RBF) o el kernel polinomial. La elección inadecuada del kernel puede afectar significativamente la capacidad del algoritmo para separar los datos. Además de la selección del kernel, los hiperparámetros asociados con el kernel (por ejemplo, el ancho de banda en el caso del kernel RBF) deben ajustarse adecuadamente. La elección de valores inapropiados para estos hiperparámetros puede conducir a resultados subóptimos.
Aunque has generado los datos con un bajo nivel de ruido (noise=0.05), el ruido presente en los datos puede afectar la capacidad del algoritmo de separar las dos clases de manera

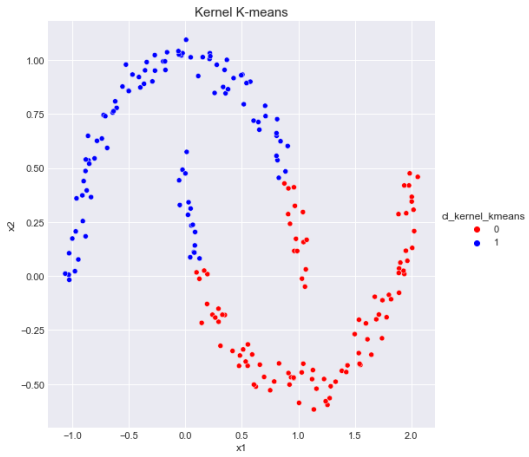


Figura 0.3: Prueba Clustering con Kernel K-Means, de los datos de media luna generados por sklearn

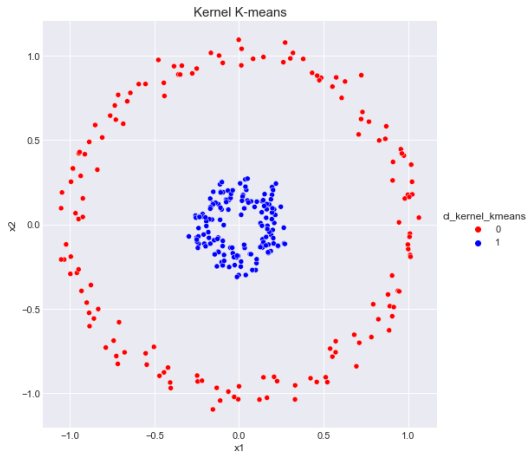


Figura 0.4: Prueba Clustering con Kernel K-Means, de los datos de dona generados por sklearn

efectiva. Al igual que en el algoritmo K-Means estándar, el método de Kernel K-Means puede ser sensible a la inicialización aleatoria de los centroides. Una mala inicialización puede llevar a una convergencia a mínimos locales subóptimos.
Para el caso de media luna se ajustaron con un $\sigma = 1$ y con un kernel gaussiano lo cual dio buenos resultados como se muestran en la figura 2.4. esto puede ser debido al ajuste que le hicimos al código debido a que entre modifiquemos el tipo de kernel y otros parametros el método de kernel k - means funciona de manera correcta.

K - Mediods

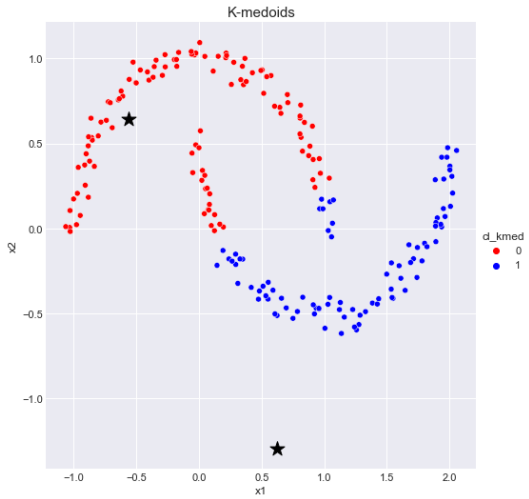


Figura 0.5: Prueba Clustering con K - Mediods, de los datos de media luna generados por sklearn

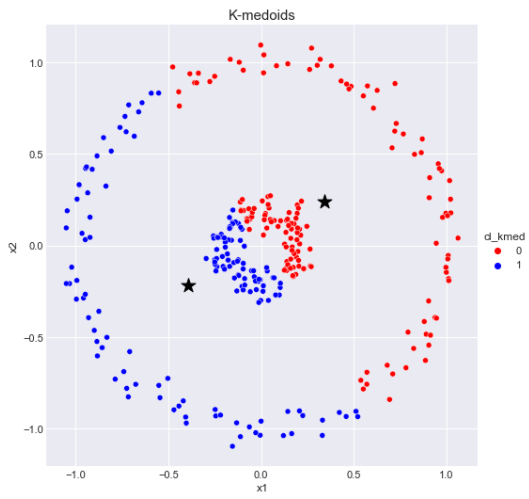


Figura 0.6: Prueba Clustering con K - Mediods, de los datos de dona generados por sklearn

El algoritmo de K-Medoids es una variante de K-Means que utiliza medoides en lugar de centroides para representar los clústeres. Aunque K-Medoids es robusto frente a valores atípicos y puede funcionar bien en ciertos escenarios, también puede enfrentar desafíos, especialmente en conjuntos de datos complejos como el de las dos medias lunas de la figura 2.3. Además en

esta figura podemos notar que los centros de K mediods, están bastantes disparejos en el caso de el grupo rojo muy centrico y en el cluster azul demasiado alejado
K-Medoids, al igual que K-Means, puede ser sensible a la inicialización de los medoides. La elección inicial de medoides puede afectar la convergencia del algoritmo y los resultados finales.Además que, las dos medias lunas generadas forman un conjunto de datos no convexo. Algoritmos que asumen estructuras convexas pueden tener dificultades para representar y separar correctamente estos tipos de conjuntos de datos.
La geometría de las medias lunas puede hacer que la identificación de medoides representativos sea más difícil, ya que los medoides deben estar ubicados dentro de las regiones densas de datos.

Fuzzy K - Means

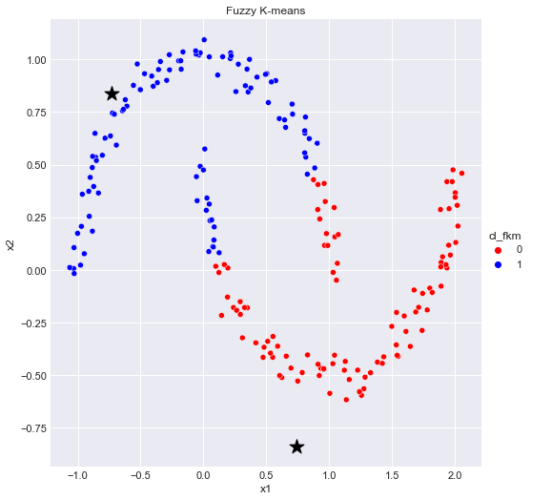


Figura 0.7: Fuzzy K- Means, de los datos de media luna

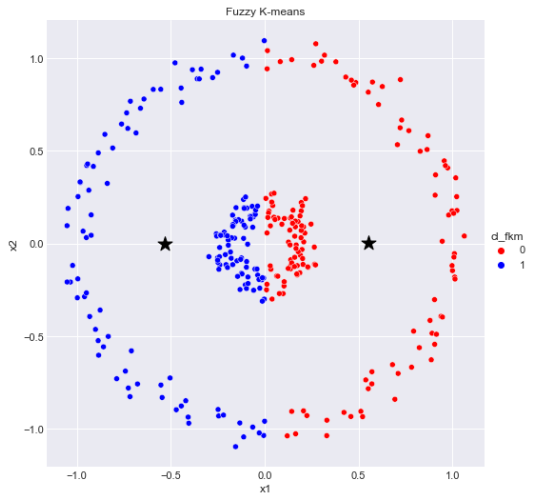


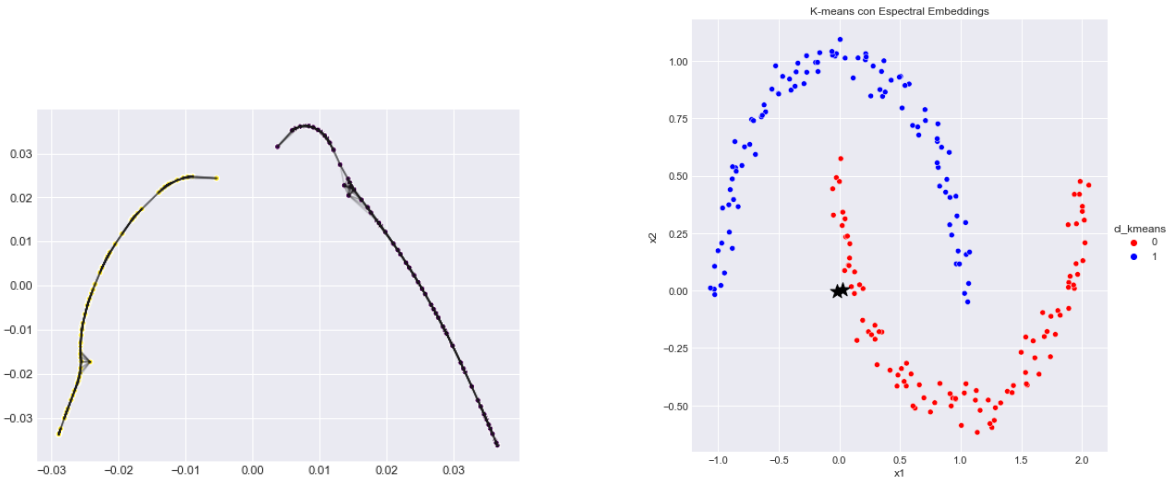
Figura 0.8: Fuzzy K- Means, de los datos de tipo dona

Fuzzy K-Means también puede ser sensible a la inicialización de los centroides y la matriz de pertenencia. Una mala inicialización puede llevar a la convergencia a mínimos locales subóptimos. Además, los datos tienen una estructura convexa, como las dos medias lunas, Fuzzy K-Means puede tener dificultades para asignar grados de pertenencia adecuados a los puntos.
Fuzzy K-Means puede ser más sensible a valores atípicos y ruido en comparación con K-Means. Notamos en la figura 2.4, que los centros de Fuzzy K-Means estás mas cercanos que en el caso de K - Mediods, pero aún asi este método resulta sensible ante esta no linealidad de los datos.

Clustering Espectral

El método de Espectral Embeddings es particularmente eficaz cuando se trata de datos que tienen una estructura no lineal. Las medias lunas generadas son un claro ejemplo de datos no lineales, y Espectral Embeddings puede capturar estas estructuras más complejas. Espectral Embeddings utiliza la construcción de un grafo de afinidad basado en la distancia entre los puntos. El hecho de que las medias lunas estén bien separadas en el espacio transformado indica que la afinidad y el vecindario entre los puntos se han conservado de manera efectiva. La elección de $n_neighbors = 10$ para la construcción del grafo de afinidad y $n_clusters_kmeans = 2$ para K-Means puede haber sido adecuada para este conjunto de datos específico. La aplicación de K-Means después de la transformación espectral puede haber convergido a una solución

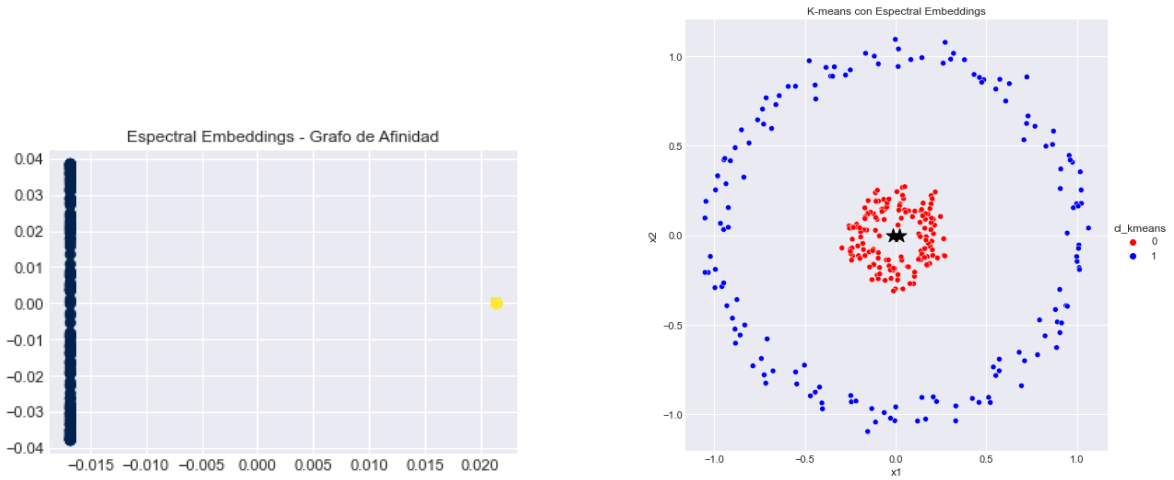
que separa bien las dos medias lunas. Además, los centroides iniciales pueden haber sido adecuadamente seleccionados. Capacidad de K-Means para datos bien separados:
K-Means es eficaz cuando los clústeres están bien separados y son de forma esférica. En este caso, como en la figura 2.5 b podemos ver que las medias lunas generadas tienen una separación clara y una forma que facilita la tarea de K-Means.



(a) Grafo creado en el embedding a partir de usar vecino mas cercano

(b) Resultado de la prueba de clustering espectral

Figura 0.9: Prueba con Clustering Espectral, de los datos de media luna generados por sklearn



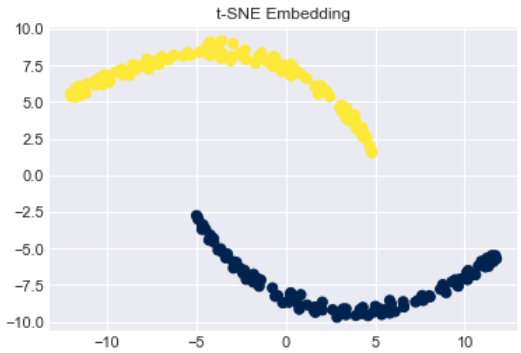
(a) Grafo creado en el embedding a partir de usar vecino mas cercano

(b) Resultado de la prueba de clustering espectral

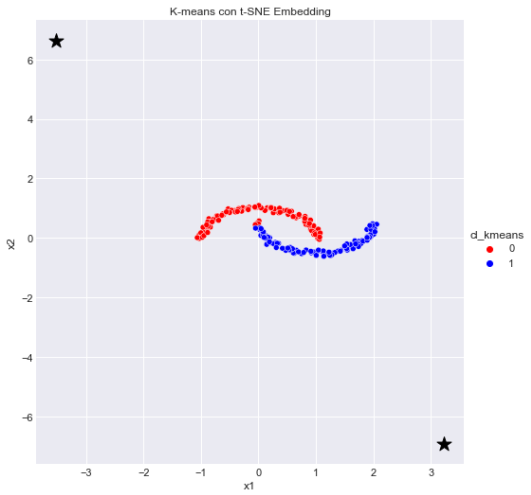
Figura 0.10: Prueba con Clustering Espectral, de los datos de dona generados por sklearn

T - SNE

T-SNE con K-Means funcionó adecuadamente en los datos de media luna, aunque no tan bien como el clustering espectral como se muestra en la figura 2.6. Vemos que en la parte de la reduccion de dimensionalidad al aplicar t-sne trabaja de buena forma pero en la parte de aplicar k means tenemos ciertos problemas. El clustering espectral es conocido por su capacidad para

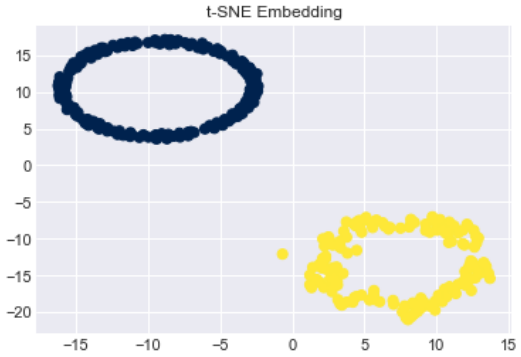


(a) Resultado de reducción de dimensionalidad al aplicar t - sne

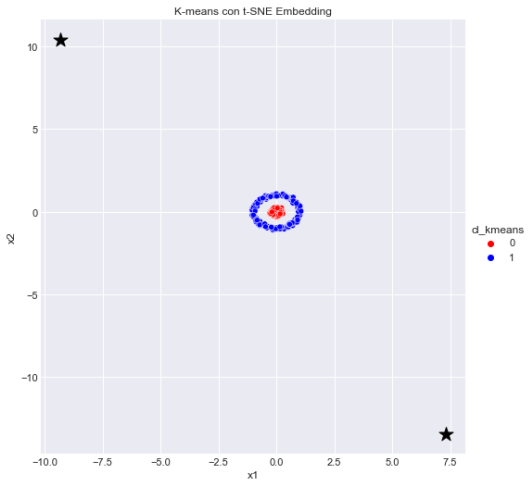


(b) Cluster aplicando K -means al resultado de reducción de dimensionalidad

Figura 0.11: Prueba con Clustering Espectral, de los datos de media luna generados por sklearn



(a) Resultado de reducción de dimensionalidad al aplicar t - sne



(b) Cluster aplicando K -means al resultado de reducción de dimensionalidad

Figura 0.12: Prueba con Clustering Espectral, de los datos tipo dona generados por sklearn

modelar estructuras complejas y no lineales en datos. Al trabajar con medias lunas, que tienen una estructura no lineal, el clustering espectral pudo haber capturado de manera más efectiva las relaciones intrínsecas en el espacio de características. K-Means funciona bien cuando los clústeres son de forma esférica y tienen tamaños y densidades aproximadamente iguales. Las medias lunas pueden presentar desafíos para K-Means debido a su forma no convexa y a la variabilidad en la densidad de las regiones. T-SNE tiene una componente estocástica, lo que significa que puede proporcionar resultados ligeramente diferentes en ejecuciones diferentes. Esto podría haber afectado los resultados obtenidos. Por otro lado conocemos que k-means es un método sensible a este tipo de datos puede que no sea suficiente la reducción de dimensionalidad para que k - means logra captar de manera correcta los cluster en este tipo de datos. En los datos tipo dona podemos observar que si esta funcionando de manera correcta como

se puede ver en la figura 2.12. entonces podemos decir que es unn buen metodo para clasificar este tipo de datos.