Centro de Investigación en Matemáticas (CIMAT).
Unidad Monterrey

Ciencia de Datos Tarea 2

Diego Godinez Bravo

12 de marzo de 2024

1. Problema 1

En la Figura 1 se muestran 6 conjuntos de datos en dos dimensiones denotados con A, B, C, D, E y F. En cada uno se usaros dos métodos de clustering, y uno de ellos fue k-means. Los centroides de cada clúster se señalan con una x, ambos ejes son proporcionales y en la misma escala para todos los conjuntos de datos. Indica, en cada conjunto, cuál solución corresponde a k-means y porqué.

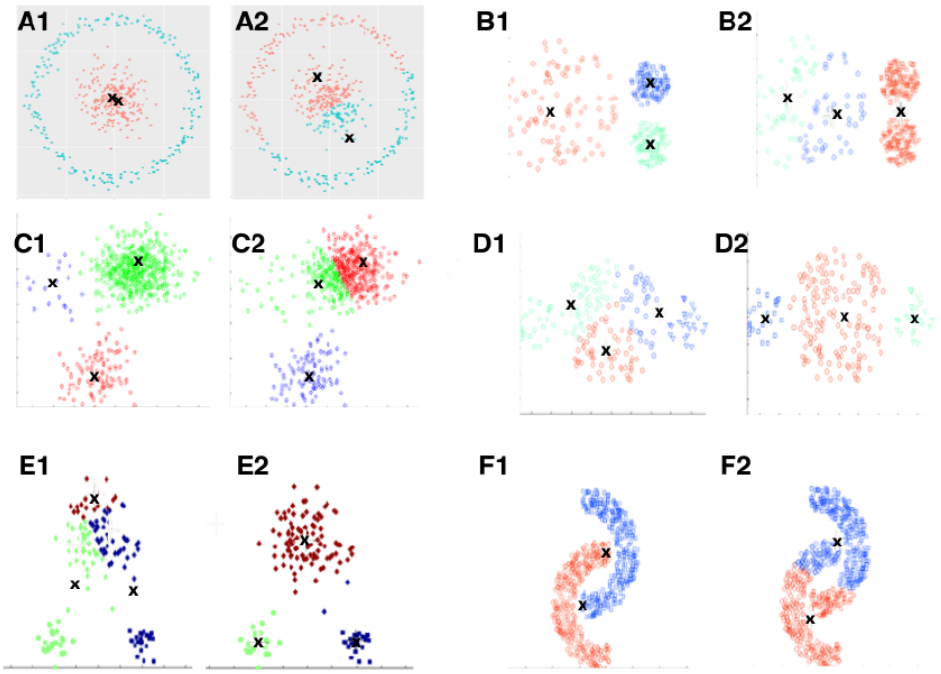

Figura 1.1. Clustering realizado con dos métodos en diferentes conjuntos de datos.

1.1. Solución

1.1.1. Conjunto de datos A

En este caso se aplicó k-means al subconjunto **A2**. Dado que k-means genera centroides los cuales se utilizan para minimizar las distancias entre las observaciones y asignarlas a grupos específicos, los centroides del subconjunto **A1** no serían plausibles debido a su proximidad entre ellos. El método de k-means se caracteriza por distribuir los centroides de manera que abarquen el espacio de caracteríísticas de manera más equilibrada. Esta distribución facilita una asignación clara de los puntos a los grupos correspondientes, además de favorecer una convergencia eficiente del algoritmo.

1.1.2. Conjunto de datos B

En el contexto del conjunto B, se aplicó el algoritmo k-means al subconjunto **B2**. Sea el subconjunto **B1** el resultado de k-means los centroides generados en este proceso podrían presentar dificultades para una asignación clara de los puntos a cada grupo. Esto debido a que los centroides ubicados a la derecha, están demasiado próximos entre sí. Además, el centroide restante esta categorizando puntos dispersos en el espacio de características, lo que complica la asignación. Como resultado, algunos puntos podrían encontrarse más cercanos a cualquiera de los centroides ubicados a la derecha en lugar de estar asignado al centroide central.

1.1.3. Conjunto de datos C

Debido a la distribución de los centroides generados en cada subconjunto, podríamos concluir que en este caso se aplicó el método de k-means al subconjunto **C2**. Esto se debe a que, a pesar de que los centroides en la parte superior se encuentran muy próximos entre sí, muestran un mayor equilibrio en la asignación de los puntos a sus respectivos grupos en comparación con el subconjunto **C1**, donde uno de los centroides abarca una amplia cantidad de puntos dispersos.

1.1.4. Conjunto de datos D

Considerando el conjunto D, se aplicó el método de k-means en el subconjunto **D1**. Esto fundamentado en la distribución de los centroides, los cuales abracan el espacio de características de manera uniforme, manteniendo una distancia constante entre ellos. Por otro lado, en el subconjunto **D2**, se observa un centroide central que abarca una amplia cantidad de puntos dispersos, lo que dificultaría la asignación de algunos puntos a sus respectivos grupos.

1.1.5. Conjunto de datos E

En el contexto del conjunto E, se aplicó el algoritmo k-means en el subconjunto **E2**. Esto basándonos en la distribución de los centroides del subconjunto **E1**, los cuales no podrían asignar de manera clara cada uno de los puntos a su respectivo grupo.

1.1.6. Conjunto de datos F

En este caso, se aplicó el algoritmo k-means al subconjunto **F2**. Recordando que k-means genera centroides que se utilizan para minimizar las distancias entre las observaciones y asignarlas a grupos específicos, la asignación de puntos a cada grupo en el subconjunto **F1** no sería factible, ya que los puntos en ubicados en las colas deberían de ser asignados al centroide más cercano, lo que coincide con lo observado en el subconjunto **F2**.

2. Problema 2

Leé el artículo de Inderjit Dhillon, Yuqiang Guan and Brian Kulis: *A Unified view of Kernel k-means, Spectral Clustering and Graph Cuts. UTCS Technical Report, 2005*; el cual está en la página del curso. Haz un breve resúmen recuperando las ideas más importantes respecto a los métodos de clústering que hemos visto.

Utiliza o implementa Kernel k-means. Escoge (o genera) algunos conjuntos de datos adecuados para verificar la eficiencia y ventajas del método. Comparalo con otros métodos que hemos visto para mostrar en qué casos es mejor su desempeño. Puedes usar o modificar la clase KernelKMeans que agrego en la tarea. Realiza un breve reporte de tus experimentos, ilustrándolo con gráficas informativas y tus hallazgos y/o conclusiones.

2.1. Resumen.

A Unified view of Kernel k-means, Spectral Clustering and Graph Cuts,
Dhillon I. & Guan Y. (2005).

El algoritmo kernel k-means amplía el enfoque del k-means clásico al mapear puntos en un espacio de mayor dimensión, permitiendo así la detección de grupos que no pueden separarse linealmente en el espacio de entrada. Por otro lado, los algoritmos de particionamiento de grafos se centran en agrupar nodos en un grafo. La versión ponderada del objetivo del kernel k-means es esencialmente equivalente, desde un punto de vista matemático, a un objetivo general de particionamiento de grafos que tiene en cuenta los pesos. En situaciones donde el cálculo de eigenvectores no es viable, el uso del algoritmo ponderado del kernel k-means podría ser preferible sobre los métodos espectrales. Sin embargo, cuando el cálculo de eigenvectores es factible, los métodos espectrales pueden utilizarse para inicializar los grupos de manera efectiva.

De manera general k-means busca encontrar grupos que minimicen una función objetivo, dado un conjunto de vectores. Una clara desventaja del método k-means clásico es que los grupos deben estar separados por un hiperplano, debido a que se utiliza la distancia euclidiana cuadrada como medida de distorsión. Para superar esta limitación, el algoritmo kernel k-means mapea los puntos a un espacio de características de mayor dimensión utilizando una función específica. Esto permite descubrir clusters que no pueden ser separados linealmente en el espacio de entrada.

Por otro lado, en el contexto de la partición de grafos, se nos presenta un grafo $G = (V, E, A)$, compuesto por un conjunto de vértices V y un conjunto de aristas E, donde una arista entre dos vértices representa su similitud. El problema de partición de grafos busca dividir el grafo en k particiones o clústeres $V_1, ..., V_k$ de manera que su unión sea igual a V. Se han propuesto y estudiado varios objetivos de partición de grafos, en este caso se enfocan en los más citados y referenciados a lo largo de la literatura. Uno de ellos es el corte normalizado ('normalized cut'). El objetivo del corte normalizado es uno de los más populares y busca minimizar el corte en relación al grado de un clúster en lugar de su tamaño.

Dhillon I. & Guan Y. (2005) presentan una visión unificada del particionamiento de grafos y el método de k-means ponderado con kernel. Proponen reformular los problemas de asociación ponderada de grafos y corte de grafos de manera idéntica como problemas de maximización de trazas de matrices, demostrando que ambos objetivos son matemáticamente equivalentes.

Los autores introducen una versión ponderada de la función objetivo del método kernel k-means, donde los pesos juegan un papel crucial al mostrar una equivalencia con el particionamiento de grafos.

$$D(\{\pi_c\}_{c=1}^k) = \sum_{c=1}^{k} \sum_{a_i \in \pi_c} w_i ||\phi(a_i) - m_c||^2, \qquad \text{donde } m_c = \frac{\sum_{a_i \in \pi_c} w_i \phi(a_i)}{\sum_{a_i \in \pi_c} w_i}$$

función objetivo del método kernel k-means ponderado.

El algoritmo representa una extensión directa del k-means clásico. Al igual que en el método k-means clásico, una vez que se han determinado los centroides actuales, se procede a calcular el centroide más cercano para cada punto. Posteriormente, se ajusta el agrupamiento. Estos dos pasos se repiten hasta que la variación en el valor de la función objetivo sea lo suficientemente pequeña.

En términos generales, el algoritmo consta de 4 etapas. Primero, se establecen los pesos y se define el kernel. Si se proporcionan pesos como entrada, se asume que la matriz de entrada es la matriz del kernel. De lo contrario, si se considera un objetivo estándar de particionamiento de grafos, se establece adecuadamente los pesos y la matriz del kernel. Otro caso a considerar es que los clústers iniciales también pueden especificarse como entrada. De lo contrario, se realiza la inicialización utilizando una inicialización aleatoria, inicialización espectral, inicialización basada en un desplazamiento negativo de sigma o inicialización por METIS. Después, de obtener los clústers iniciales, se modifica K de manera que sea definida positiva. Por último, se intercala entre ejecutar k-means ponderado con kernel en lotes y k-means ponderado con kernel incremental (búsqueda local) hasta la convergencia.

Dentro de los resultados experimentales presentados por los autores, se destacan varios hallazgos importantes. Reportan que un desplazamiento negativo de σ en la diagonal de la matriz del kernel puede tener un impacto significativo en la mejora de los resultados de agrupamiento. Además, afirman que el particionamiento de grafos utilizando el algoritmo propuesto muestra una mejora con respecto a los métodos espectrales, especialmente cuando se emplea inicialización espectral. Por último, hacen énfasis en que la segmentación de imágenes mediante corte normalizado puede llevarse a cabo prescindiendo del uso de eigenvectores.

En resumen, el algoritmo propuesto representa una extensión relevante del método clásico de k-means. Al mapear los puntos en un espacio de mayor dimensión mediante el uso de una función específica, el algoritmo kernel k-means permite la detección de grupos que no pueden separarse linealmente en el espacio de entrada. Esta característica es sumamente valiosa en aplicaciones donde la estructura de los datos no es lineal y los grupos están interrelacionados de manera compleja. Por último, el enfoque ponderado del objetivo del kernel k-means demuestra ser matemáticamente equivalente a los objetivos generales de particionamiento de grafos, lo que resalta su versatilidad y aplicabilidad en una variedad de contextos de análisis de datos. En general, estos hallazgos respaldan la eficacia y la utilidad del algoritmo kernel k-means como una herramienta robusta para el análisis, exploración e interpretación de conjuntos de datos complejos.

2.2. Solución.

Observamos cómo el enfoque Kernel k-Means logra separar efectivamente los conjuntos de datos circulares, mientras que k-means no lo logra debido a su incapacidad para manejar distribuciones no lineales (**Figura 2.1**).

La efectividad del método Kernel k-means radica en su capacidad para mapear los datos a un espacio de características de mayor dimensión, donde las relaciones no lineales entre los puntos se vuelven lineales. Este mapeo permite que el algoritmo k-means opere en un espacio de características más adecuado, lo que facilita la separación de los conjuntos de datos circulares.

Por otro lado, el enfoque clásico k-means se basa en la asignación de centroides y la minimización de la suma de las distancias cuadradas de los puntos al centroide más cercano. Sin embargo, debido a su naturaleza lineal, k-means tiende a agrupar los datos en regiones definidas por límites lineales, lo que resulta en una falta de capacidad para separar de manera efectiva conjuntos de datos con distribuciones no lineales, como en este caso.

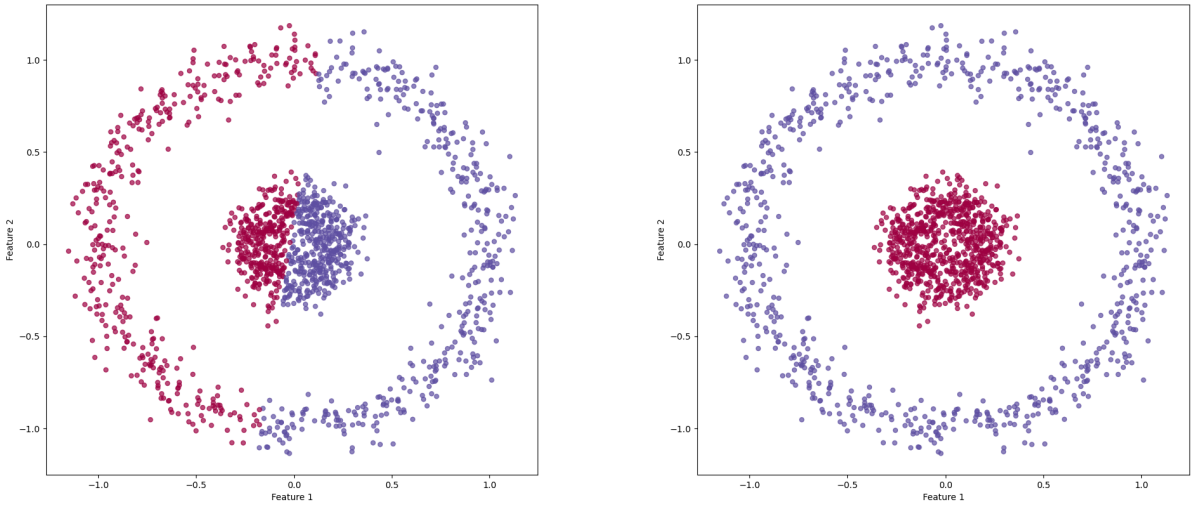

Figura 2.1. Comparación entre los métodos de agrupamiento k-means y Kernel k-means en datos circulares. En el lado izquierdo, se aplicó el método k-means, mientras que en el lado derecho se utilizó el método Kernel k-means.

Al aplicar ambos métodos utilizando datos con distribuciones gaussianas isotrópicas, observamos una similitud en los resultados obtenidos (**Figuras 2.2, 2.3**). Ambos se fundamentan en la idea de minimizar la varianza intra-cluster y maximizar la varianza entre clusters para obtener agrupaciones significativas. Esto se traduce en resultados similares en los casos donde las subpoblaciones gaussianas son claramente identificables. Esto resalta la capacidad de ambos enfoques para manejar de manera efectiva este tipo de situaciones (cuadrantes izquierdos).

Sin embargo, en situaciones donde las subpoblaciones gaussianas están muy próximas entre sí, tanto k-means como Kernel k-means pueden enfrentar dificultades para realizar una agrupación óptima. Este escenario presenta un desafío porque los límites entre los clusters se vuelven poco claras, lo que dificulta que los algoritmos de agrupamiento definan límites sólidos entre los grupos (cuadrantes derechos).

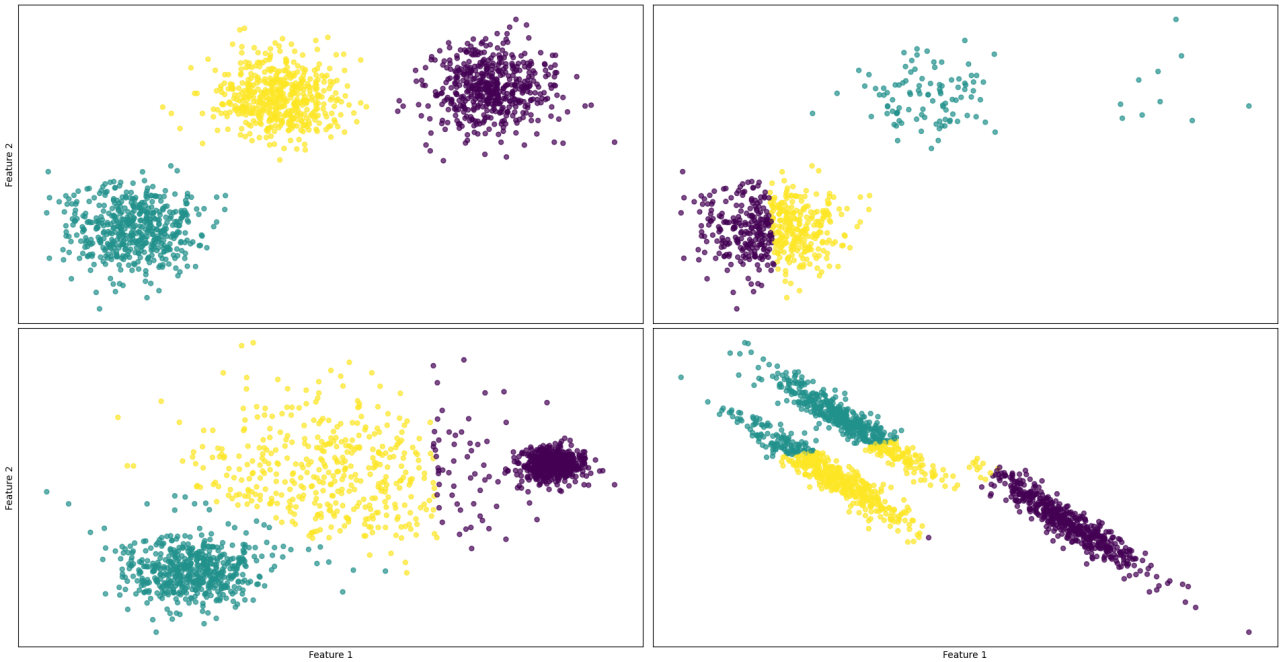

Figura 2.2. Método de agrupamiento k-means aplicado en subpoblaciones gaussianas isotrópicas.

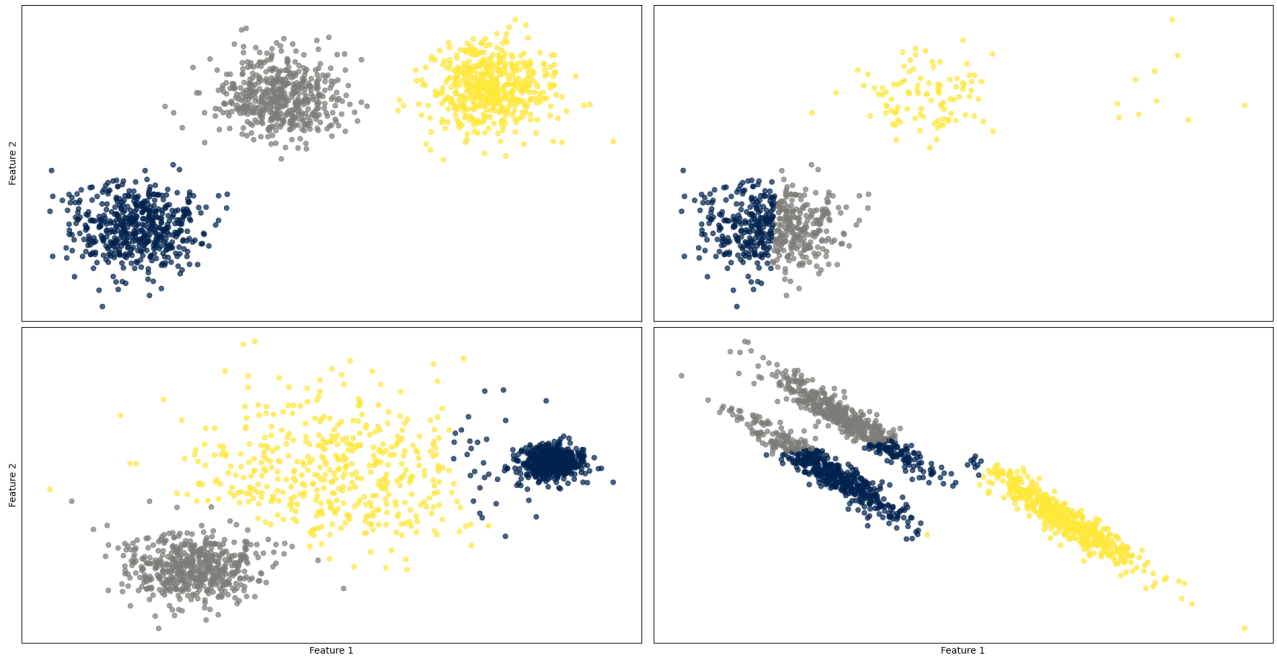

Figura 2.3. Método de agrupamiento Kernel k-means aplicado en subpoblaciones gaussianas isotrópicas.

En general, el método Kernel k-means puede manejar relaciones no lineales entre los datos al proyectarlos en un espacio de características de mayor dimensión, por otro lado k-means estándar se ve limitado por su incapacidad para tratar este tipo de distribuciones.

3. Problema 3

Considera un modelo de mezclas de k distribuciones:

$$f(x) = \sum_{k=1}^{K} w_k f_k(x)$$

donde $w_k \geq 0$ y $\sum_k w_k = 1$. En este caso, supondremos que $f_k = N(\mu_k, \Sigma_k)$.

Supón que tienes datos $x_1, x_2, ..., x_n \sim f(x)$, y queremos ajustar el modelo de mezclas de Gaussianas (MMG) para usarlo como soft-clustering.

a) Obtén la log-verosimilitud de los datos y los estimadores de máxima verosimilitud para los parámetros del modelo.

b) Implementa un método de clsutering usando el siguiente algoritmo (MMG-EM):

   1) Inicializa los parámetros del modelo y los pesos w_k.

   2) *Expectation*: asigna las 'responsabilidades' de cada dato, es decir, la asignación de un dato al cluster k, que en este esquema es la probabilidad de que una observación se genere de la distribución k:
   $$\gamma_i^k = P(C(i)) = (k|X = x_i) = \frac{w_k f_k(x_i; \mu_k \Sigma_k)}{\sum_k w_k f_k(x_i; \mu_k \Sigma_k)}$$

   3) *Maximization*: actualiza los parámetros μ_k^{new} y Σ_k^{new} usando las responsabilidades obtenidas. Observa que en este paso, usamos la 'asignación suave' de cada punto a un cluster k, por lo tanto, cada observación debe ser pesada por su correspondiente responsabilidad, y en consecuencia, el número de puntos 'asignados' a algún cluster k será $n_k = \sum_{i=1}^{n} \gamma_i^k$.

   4) Repite los pasos b) y c) hasta que la log-verosimilitud converja.

c) Considera el caso en que cada Gaussiana tiene la misma matriz de covarianzas esférica:

$$\Sigma_k = \sigma^2 I.$$

   Muestra que, cuando $\sigma^2 \to 0$, el método de MMG-EM y k-means coinciden.

Prueba tu implementación con un conjunto de datos sintéticos bien escogidos en dos dimensiones y compáralo contra fuzzy k-means. Discute los resultados.

3.1. Solución

Calculamos la verosimilitud, con los parámetros de la verosimilitud dados por

$$\theta = [w_j, \mu_j, \Sigma_j]_{j=1}^{k}$$

De manera que

$$L(x_1, x_2, \ldots, x_n; \theta) = \prod_{i=1}^{n} P(x_i; \theta) = \prod_{i=1}^{n} \sum_{k=1}^{k} w_k f_k(x_i) = \prod_{i=1}^{n} \left[\sum_{k=1}^{k} w_k \mathcal{N}(x_i | \boldsymbol{\mu} k, \boldsymbol{\Sigma}_{\mathbf{k}})(xi) \right]$$

Aplicamos un logaritmo para obtener la *log*-verosimilitud

$$\log(L(x_1, x_2, \ldots, x_n; \theta)) = \sum_{i=1}^{n} log \left[\sum_{k=1}^{k} w_k \mathcal{N}(x_i | \boldsymbol{\mu} k, \boldsymbol{\Sigma}_{\mathbf{k}})(xi) \right]$$

Obtenemos los estimadores que maximizan esta función basándonos en el método EM ('Expectation–maximization'). Por lo tanto, definimos γ_i^j, lo cual determina la probabilidad de que un punto x_i provenga del cluster k, i.e. definimos con la regla de bayes γ_i^j de la siguiente manera

$$\gamma_i^k = P\left(C(i) = k | X = xi\right) = \frac{w_k(\mathcal{N}(x_i \boldsymbol{\mu} k, \mathbf{\Sigma_k}))}{\sum k w_k(\mathcal{N}(x_i \boldsymbol{\mu}_k, \mathbf{\Sigma_k}))}$$

Posteriormente calculamos el valor esperado ponderando con respecto a la *log*-verosimilitud

$$E = \sum_{i=1}^{n}\left[\sum_{k=1}^{K}\gamma_i^k log\left(w_k(\mathcal{N}(x_i\boldsymbol{\mu}_k, \mathbf{\Sigma_k})\right)\right]$$

$$= \sum_{i=1}^{n}\left[\sum_{k=1}^{K}\gamma_i^k log\left(w_k\left(\frac{1}{(2\pi)^{\frac{d}{2}}\left|\Sigma j\right|^{\frac{1}{2}}}e^{-\frac{1}{2}(xi-\mu_j)^T\Sigma_j^{-1}(x_i-\mu_j)}\right)\right)\right]$$

Aplicamos la maximización del valor esperado, calculando las derivadas para encontrar los estimadores. Primero empezamos calculando el estimador para $\hat{\mu_j}$.

Obsrvamos que los únicos términos que involucran a μ_j son $= \left(\sum_{i=1}^{n} -\gamma_i^k \frac{1}{2}\left(x_i - \mu_j\right)^T \Sigma_j^{-1}\left(x_i - \mu_j\right)\right)$. Calculamo la derivada

$$\frac{\partial}{\partial \mu_j}E = \frac{\partial}{\partial \mu_j}\left(\sum_{i=1}^{n}\sum_{k=1}^{K} -\gamma_i^k \frac{1}{2}\left(x_i - \mu_j\right)^T \Sigma_j^{-1}\left(x_i - \mu_j\right)\right) = 0$$

Simplificando y agrupando términos

$$\sum_{i=1}^{n}\gamma_i^k \mathbf{\Sigma} k^{-1} x_i = \left(\sum i = 1^n \gamma_i^k\right)\boldsymbol{\mu}_k$$

Despejando en términos de μ

$$\hat{\boldsymbol{\mu}} k = \frac{\sum i = 1^n \gamma_i^k x_i}{\sum_{i=1}^{n}\gamma_i^k}$$

Calculamos el estimador de las covarianzas

$$\frac{\partial}{\partial \Sigma_j}E = \frac{\partial}{\partial \Sigma_j}\sum_{i=1}^{n}\left[\sum_{k=1}^{K}\gamma_i^k log\left(w_k\left(\frac{1}{(2\pi)^{\frac{d}{2}}\left|\Sigma j\right|^{\frac{1}{2}}}e^{-\frac{1}{2}(xi-\mu_j)^T\Sigma_j^{-1}(x_i-\mu_j)}\right)\right)\right]$$

$$= \frac{\partial}{\partial \Sigma_j}\left(\sum_{i=1}^{n} -\frac{1}{2}log|\Sigma_j| + \gamma_i^k \frac{1}{2}\left(x_i - \mu_j\right)^T \Sigma_j^{-1}\left(x_i - \mu_j\right)\right)$$

$$= \sum_{i=1}^{n} -\frac{1}{2}\gamma_i^k \Sigma_j^{-1} + \sum_{i=1}^{n}\gamma_i^k \frac{1}{2}\Sigma_j^{-1}\left(x_i - \mu_j\right)^T\left(x_i - \mu_j\right)\Sigma_j^{-1}$$

$$= \hat{\Sigma_k} = \frac{\sum_{i=1}^{n}\left(x_i - \mu_j\right)^2 \gamma_i^k}{\sum_{i=1}^{n}\gamma_i^k}$$

Igualamos a cero para obtener el estimador de la matriz de covarianzas

$$0 = \sum_{i=1}^{n} -\frac{1}{2}\gamma_i^k \Sigma_j^{-1} + \sum_{i=1}^{n}\gamma_i^k \frac{1}{2}\Sigma_j^{-1}\left(x_i - \mu_j\right)^T\left(x_i - \mu_j\right)\Sigma_j^{-1}$$

Manipulando algebraicamente la expresión encontramos que el resultado del estimador esta dado de la siguiente manera

$$\hat{\Sigma_k} = \frac{\gamma_i^k\left(x_i - \mu_j\right)\left(x_i - \mu_j\right)^T}{\sum_{i=1}^{n}\gamma_i^k}$$

Por último calculamos el estimador de w_i, recordando que existe la restricción $\sum_{k=1}^{K} w_k = 1$, por lo que utilizando los multiplicadores de Lagrange

$$\mathbb{L}(w, \alpha) = \sum_{i=1}^{n} \sum_{k=1}^{K} \gamma_i^k \log(w_i) + \alpha \left(1 - \sum_{i=1}^{n} w_i\right)$$

$$0 = \frac{\partial}{\partial w_j} \mathbb{L}(w, \alpha)$$

$$\bar{w}_j = \frac{1}{n} \sum_{i=1}^{n} \gamma_i^k$$

De manera que los estimadores obtenidos son

$$\bar{w}_j = \frac{1}{n} \sum_{i=1}^{n} \gamma_i^k$$

$$\hat{\Sigma}_k = \frac{\sum_{i=1}^{n} (x_i - \mu_j)^2 \gamma_i^k}{\sum_{i=1}^{n} \gamma_i^k}$$

$$\hat{\mu} = \frac{\sum_{i=1}^{n} \gamma_i^k x_i}{\sum_{i=1}^{n} \gamma_i^k}$$

3.2. Solución

Modelo de Mezclas Gaussianas (*'Gaussian Mixture Model'*).

Un modelo de mezcla Gaussiana es un modelo probabilístico que postula que todos los puntos de datos se originan a partir de una combinación de un número finito de distribuciones gaussianas, cada una con parámetros desconocidos. Este modelo es útil para representar subgrupos distribuidos normalmente dentro de una población más amplia. Lo notable es que el modelo no necesita información previa sobre la pertenencia de un punto a una subpoblación específica, lo que facilita que el modelo identifique automáticamente estas subpoblaciones durante el aprendizaje.

Fuzzy k-means.

El algoritmo Fuzzy k-means, una variante del método k-Means clásico, es una técnica de agrupamiento que permite asignar a cada punto de datos a múltiples clusters con diferentes grados de pertenencia. A diferencia del método k-means tradicional, donde cada punto de datos pertenece estrictamente a un único cluster, en Fuzzy k-Means, los puntos de datos pueden tener una pertenencia fraccional a varios clusters. Esto lo convierte en una herramienta poderosa para el análisis de datos en situaciones donde los límites de los clusters no son claros o cuando los puntos de datos pueden pertenecer a múltiples grupos simultáneamente. Los clusters producidos por el procedimiento de k-means clásico a veces se denominan clusters 'duros', ya que cualquier vector de características x es o no miembro de un cluster en particular. Esto contrasta con los grupos 'suaves' o 'difusos', en los que un vector de características x puede tener un grado de pertenencia a cada grupo.

Con el objetivo de evaluar y contrastar la eficacia de ambos métodos, se generó un conjunto de datos sintéticos utillizando el el conjunto de datos *'make_blobs'* del módulo scikitlearn. Esta función genera subpoblaciones gaussianas isotrópicas (**Figura 3.1**).

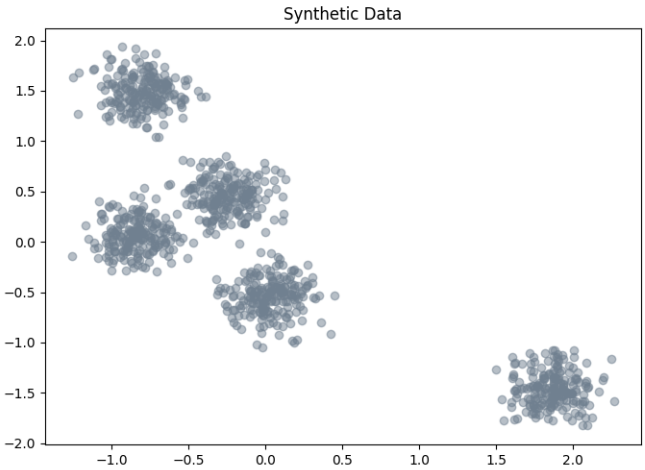

Figura 3.1. Conjunto de datos sintéticos generados utilizando el conjunto *'make_blobs'* de la librería scikitlearn.

Al aplicar ambos métodos al conjunto de datos sintéticos, se observa que el modelo GMM identifica con mayor precisión los clusters claramente definidos. Por otro lado, el modelo Fuzzy k-means identificó de manera errónea los clusters. Este resultado coincide con lo mencionado anteriormente, los clusters 'difusos' generados por el algoritmo permiten que los puntos de datos tengan un cierto grado de pertenencia a cada grupo **(Figura 3.2)**.

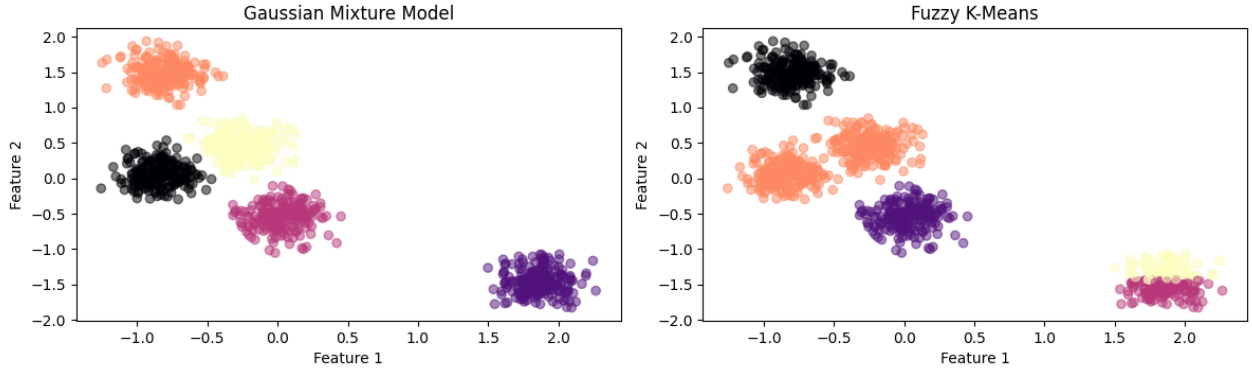

Figura 3.2. Comparación entre Modelo GGM (*'Gaussian Mixture Model'*) y Fuzzy k-means.

3.3. Solución

Recordando que γ_i^k se define de la siguiente manera

$$\gamma_i^k = P\left(C(i) = k | X = xi\right) = \frac{w_k(\mathcal{N}(x_i \boldsymbol{\mu} k, \boldsymbol{\Sigma_k}))}{\sum k w_k(\mathcal{N}(x_i \boldsymbol{\mu}_k, \boldsymbol{\Sigma_k}))}$$

y considerando que

$$\Sigma_k = \sigma^2 I$$

Entonces

$$gamma_i^k = \frac{w_k \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} e^{-\frac{1}{2}\frac{||x-\mu||}{\sigma^2}}}{\sum_k w_k \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} e^{-\frac{1}{2}\frac{||x-\mu||}{\sigma^2}})}$$

Si $\sigma^2 \to 0$, entonces

$$\lim_{\sigma^2 \to 0} \frac{1}{(2\pi\sigma^2)^{\frac{d}{2}}} = \infty$$

y

$$\lim_{\sigma^2 \to 0} e^{\left(-\frac{\|x_i - \boldsymbol{\mu} k\|^2}{2\sigma^2}\right)} = 0$$

Sabemos que cuando x_i esta cerca del centroide la distancia se aproxima a cero, por lo tanto obtiene un valor de uno. De manera que $\gamma_i^k = 1$ cuando $x_i \sim \mu_k$; y $\gamma_i^k = 0$ en otro caso.

Lo que coincide con la manera en que k-means realiza la asignación del clúúster. Por lo tanto podemos afirmar que el método MMG-EM se aproxima al método k-means cuando $\sigma^2 \to 0$.

4. Problema 4

Este ejercicio es sobre compresión de imágenes a color.

Considera una imagen en color como la que se muestra en la Figura 2 (derecha). Tu objetivo es reducir el tamaño (en Kb) de la imágen, tratando de mantener un balance entre tamaño y calidad de la misma, y para esto, utilizarás dos métodos.

a) *k-means*. En éste caso, se simplifica la imágen identificando k grupos de colores en los pixeles de la imágen, y posteriormente, se asigna cada pixel a su grupo de color correspondiente. Para esto, considera cada pixel $x_i \in N^3$, es decir, como un punto en el espacio RGB. Tu matriz de datos será entonces $X \in N^{(hxw)x3}$, donde w y h son el ancho y la altura de la imágen, respectívamente.

b) Componentes Principales. En éste método, realizarás la compresión simplificando la imágen mediante algunos componentes principales obtenidos en cada canal de color de la imágen. En éste caso, considera representaciones de la forma $X_{channel} \in N^{hxw}$.

Para cada inciso, verifica el resultado con distintos valores de k y p, reportando también el tamaño en Kb de la imágen. Escribe tus conclusiones de éste ejercicio donde consideres el balance entre compresión y calidad de la imágen. ¿Qué método prefieres y por qué? ¿Qué nivel de compresión te parece adecuado? ¿Qúe criterio (cuantitativo) se te ocurre para evaluar la compresión de la imágen original?

Detalles de implementación.

```
from io import BytesIO
def imageSize(img):
    img_file = BytesIO()
    image = Image.fromarray(np.uint8(img))
    image.save(img_file, 'png')
    return img_file.tell()/1024
```

---

donde img es una imagen RGB en forma de tensor, cuyas entradas son **enteros en un rango de 0 a 255**.

Todos los elementos para la reconstrucción de la imágen del paso 4 del Algoritmo 2, puedes obtenerlos de la clase PCA del módulo sklearn.decomposition. Alternativamente puedes usar el método inverse_transform() del mismo módulo.

4.1. Solución

El método de k-means es un algoritmo de agrupamiento utilizado comúnmente en análisis de datos y procesamiento de imágenes para clasificar un conjunto de datos en un número predeterminado de grupos, representados por el valor k. El objetivo principal es dividir un conjunto de datos en clusters de manera que los puntos de datos dentro de un mismo grupo sean similares entre sí y diferentes de los puntos de datos en otros grupos.

En el contexto de compresión de imágenes, k-means se puede utilizar para reducir el tamaño de una imagen manteniendo un porcentaje de calidad visual. Esto se logra agrupando los colores de los píxeles en la imagen en un número reducido de grupos representativos y luego asignar a cada píxel al grupo más cercano en términos de distancia en el espacio de color RGB.

Al aplicar el algoritmo k-means para la compresión de una imágen utilizando un valor de $k = 5$ clusters, se observa una disminución mínima en la calidad de la imágen. La esctructura general de la imágen se conserva, aunque se observan cambios sutiles en la nítidez, contraste y saturación de los colores (**Figura 4.1**).

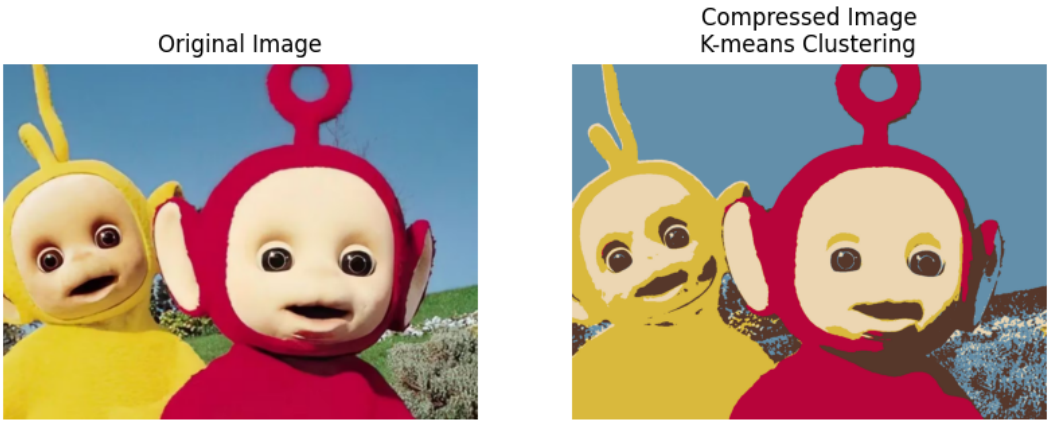

Figura 4.1. Compresión de imágenes utilizando k-means con $k = 5$ clusters.

El tamaño de la imágen original es de 830,71 Kb. Al modificar el número de clusters k, se puede observar cómo tanto la calidad como el tamaño de la imágen se ven afectados. Cuando se utiliza un valor pequeño de k, se logra una disminución significativa en el tamaño de la imágen en Kb, sin perder la estructura general de la misma. A medida que el valor de k aumenta, tanto la calidad como el tamaño de la imágen aumentan. Sin embargo, al comparar los valores de $k = 15$ y $k = 25$, no se percibe una diferencia significativa en cuanto a la calidad visual, siendo la última la que tiene un tamaño mayor. Por lo tanto, podría ser conveniente optar por el valor de $k = 15$ para reducir en mayor medida el tamaño de la imágen sin comprometer su calidad ni su estructura general (**Figura 4.2**).

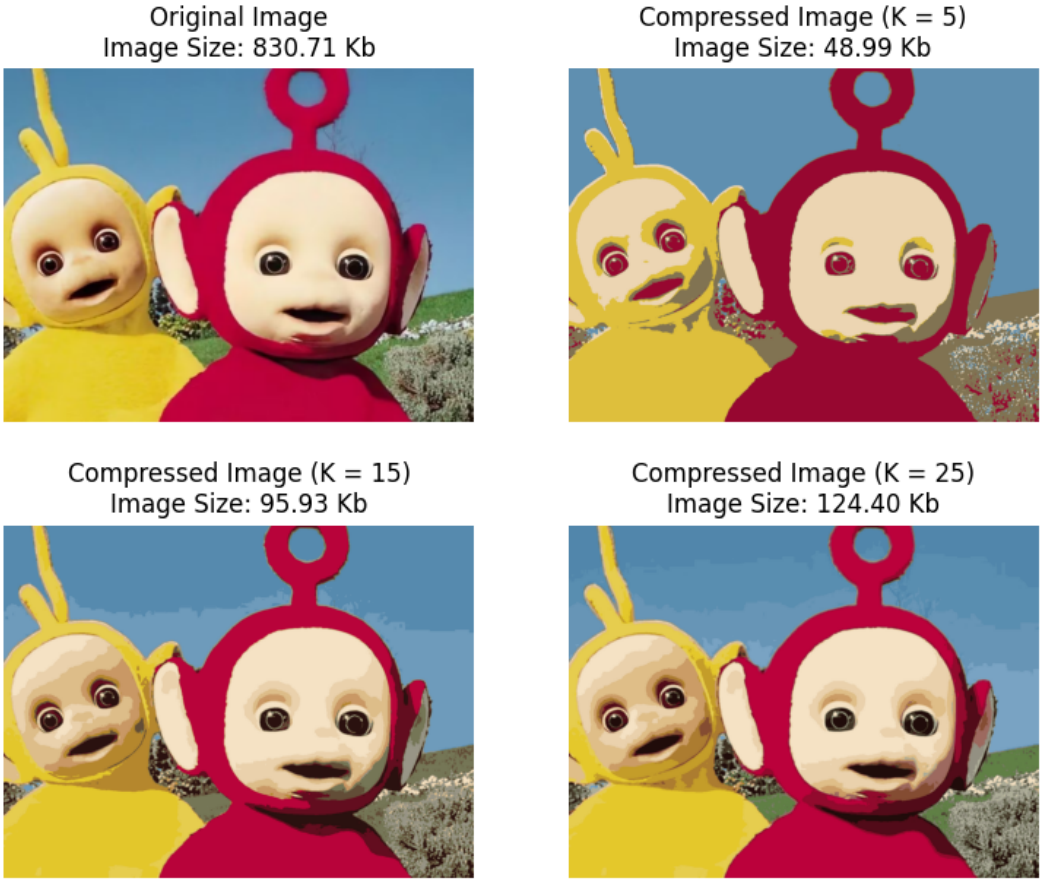

Figura 4.2. Comparación entre calidad y tamaño de imágen para distintos valores de k.

4.2. Solución

El análisis de componentes principales (PCA) es una técnica ampliamente utilizada en el análisis de datos para reducir la dimensionalidad de un conjunto de datos. Transforma un conjunto de variables correlacionadas (p) en un número k ($k < p$) más pequeño de variables no correlacionadas (componentes principales) manteniendo la mayor variación posible en el conjunto de datos original. Los componentes principales son vectores ortogonales que proporcionan una representación de los datos en dimensiones inferiores, al mismo tiempo que preservan una cantidad significatva de la variabilidad presente en el conjunto de datos original.

En el contexto de compresión de imágenes, PCA se puede utilizar para comprimir una imágen al reducir la cantidad de información necesaria para representarla, manteniendo su estructura general en medida de lo posible. En este caso se aplicó PCA a cada canal de color por separado del espacio de color RGB (rojo, verde y azul) para obtener los componentes principales de cada canal, conservando solo los primeros n componentes principales de cada canal para comprimir la imagen, lo que resultó en una representación simplificada de la imágen original con una menor cantidad de información.

Al aplicar el algoritmo de PCA para la compresión de una imágen utilizando un valor de $p = 5$ componentes principales, se observa una reducción significativa en la calidad visual. La estructura general de la imágen se ve comprometida en gran medida (**Figura 4.3**).

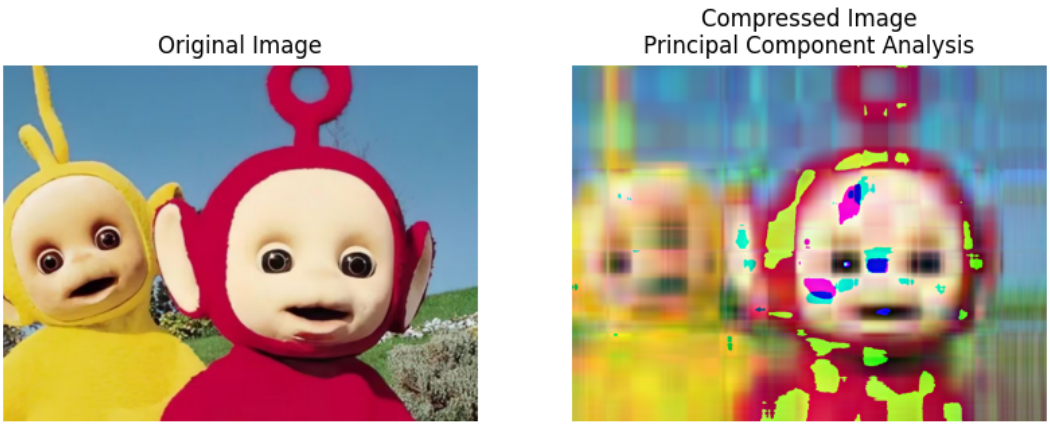

Figura 4.3. Compresión de imágen utilizando Análisis de Componentes Principales con $p = 5$ componentes principales.

Al variar el número de componentes principales p, al igual que en el caso anterior con el valor de clusters k (k-means), se puede notar que tanto la calidad como el tamaño de la imágen se ven afectados. Cuando se utiliza un valor bajo de p, la reducción en el tamaño de la imagen es mínima, pero esto compromete en gran medida la estructura general de la misma. A medida que p aumenta, la calidad visual aumenta, pero también lo hace el tamaño de la imágen, lo cual puede resultar contraproducente ya que se pierde el objetivo principal de la compresión de la imagen (**Figura 4.4**).

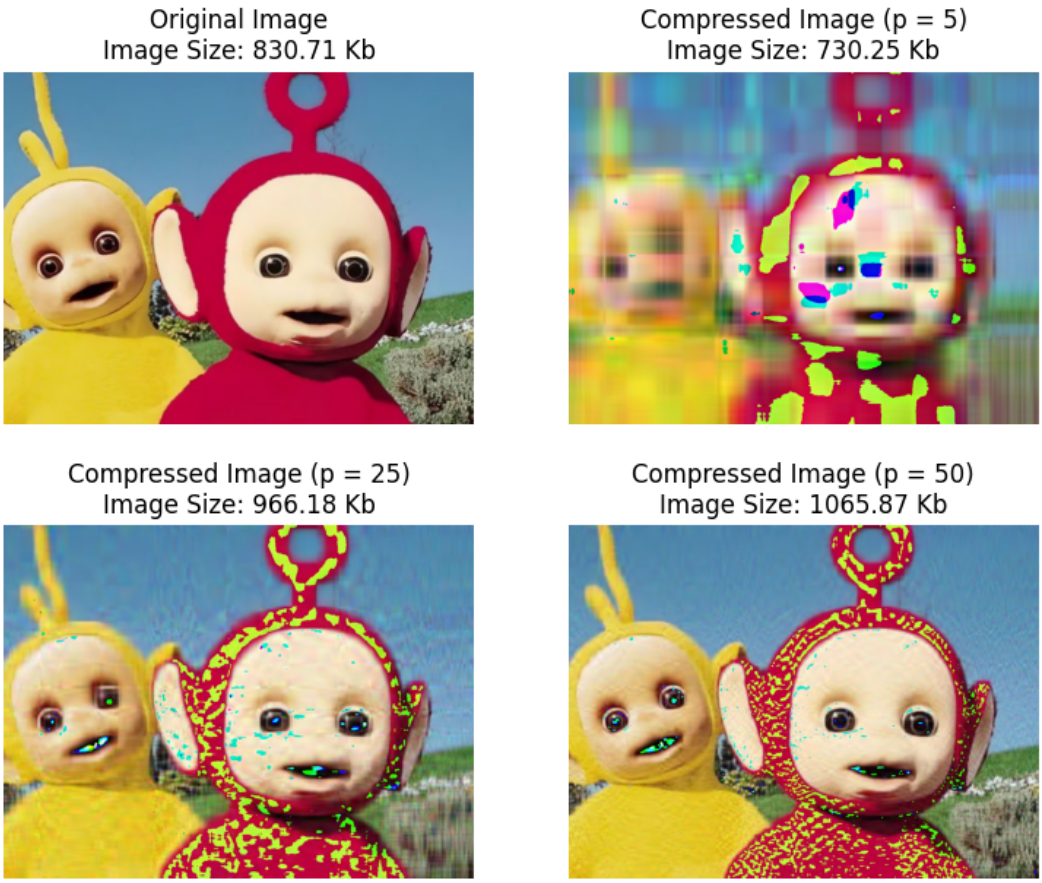

Figura 4.4. Comparación entre calidad y tamaño de imágen para distintos valores de p.

Al evaluar ambos enfoques, observamos que el método de k-means resulta más efectivo para la compresión de imágenes a color. Este método logra reducir considerablemente el tamaño del archivo de la imagen sin comprometer su estructura general, incluso al utilizar valores pequeños de k. Por otro lado, el análisis de componentes principales presenta limitaciones en este aspecto, ya que la reducción en el tamaño del archivo es mínima y afecta significativamente la calidad visual de la imágen. Esto sugiere que, en términos de equilibrio entre tamaño y calidad de imágen, el enfoque de k-means es más conveniente para la compresión de imágenes a color