

Práctica 7

Agrupamiento

Javier Herrer Torres (NIP: 776609)



Aprendizaje automático
Grado en Ingeniería Informática



**Escuela de
Ingeniería y Arquitectura
Universidad** Zaragoza

Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza
Curso 2020/2021

1. Objetivo

El objetivo es utilizar técnicas de agrupamiento. Implementaremos el algoritmo Kmedias y lo utilizaremos para cuantificar imágenes.

2. Estudio previo: Compresión de imágenes

El algoritmo de Kmedias puede ser usado para comprimir una imagen aunque en esta compresión se produzcan pérdidas, es decir, no será posible recuperar la imagen original. Cuanto mayor sea el ratio de compresión mayor diferencia existirá respecto a la original [1]. Las ideas básicas son:

- La imagen original se encuentra en $D_{m \times n}$ que almacena los valores RGB de cada píxel ($n = 3$).
- El número de clústeres k debe ser menor que el número de píxeles de la imagen (m).
- Guardar los centroides en $\mu_{k \times n}$.
- Guardar las asignaciones de cada píxel i a su centroide en $c^{(i)}$.

3. Metodología

Se ha utilizado el algoritmo para la cuantificación de imágenes (*image quantization*). La idea central es representar la imagen como un grupo de K colores (clústeres) más pequeño que la paleta original. Para ello, se ha asignado cada píxel a un clúster, es decir, se ha asignado a ese píxel la representación de color (media o centroide) de ese clúster.

3.1. Inicialización de los centroides

Una estrategia inicial consistía en escoger aleatoriamente los centroides mediante la permutación de la matriz de datos. Sin embargo, este método se descartó al ser plausible la elección de varios centroides iguales. Esto se podía producir con facilidad si un determinado color domina una gran parte de la imagen.

Por lo tanto, se sustituyó por otra estrategia en la que los centroides fuesen elegidos equidistantes en sus atributos. En este caso, se deben valorar las 3 dimensiones de los atributos (RGB), y para ello se empleó la función `curvspace.m` [2]. Esta función genera K centroides iniciales equidistantes que interpolan una curva (descrita por la matriz de datos).

3.2. Actualización de clústeres y centroides

La función `updateCentroids.m` calcula los centroides dadas las correspondencias. Y `updateClusters.m` calcula las correspondencias dados los centroides. Consisten en:

- Para $i = 1$ hasta m
 - $c^{(i)} := k : \min_k ||x^{(i)} - \mu_k||^2$
- Para $k = 1$ hasta K
 - $\mu_k :=$ promedio de $\{x^{(i)} : c^{(i)} = k\}$

3.3. Implementación de kmeans

Para la implementación de `kmeans.m` se ha hecho uso de las dos funciones anteriores. Kmeans repite `updateClusters` y `updateCentroids` hasta que $c^{(i)}$ no cambie para ningún $x^{(i)}$.

4. Resultados

Se ha utilizado KMeans para agrupar los píxeles RGB de la imagen de prueba (y también se ha probado con otras). Para ello, se ha convertido la imagen en una matriz donde cada fila representa los valores RGB de un píxel. Una vez agrupados, se ha utilizado el código que recupera la imagen y la muestra.

4.1. Imágenes de prueba

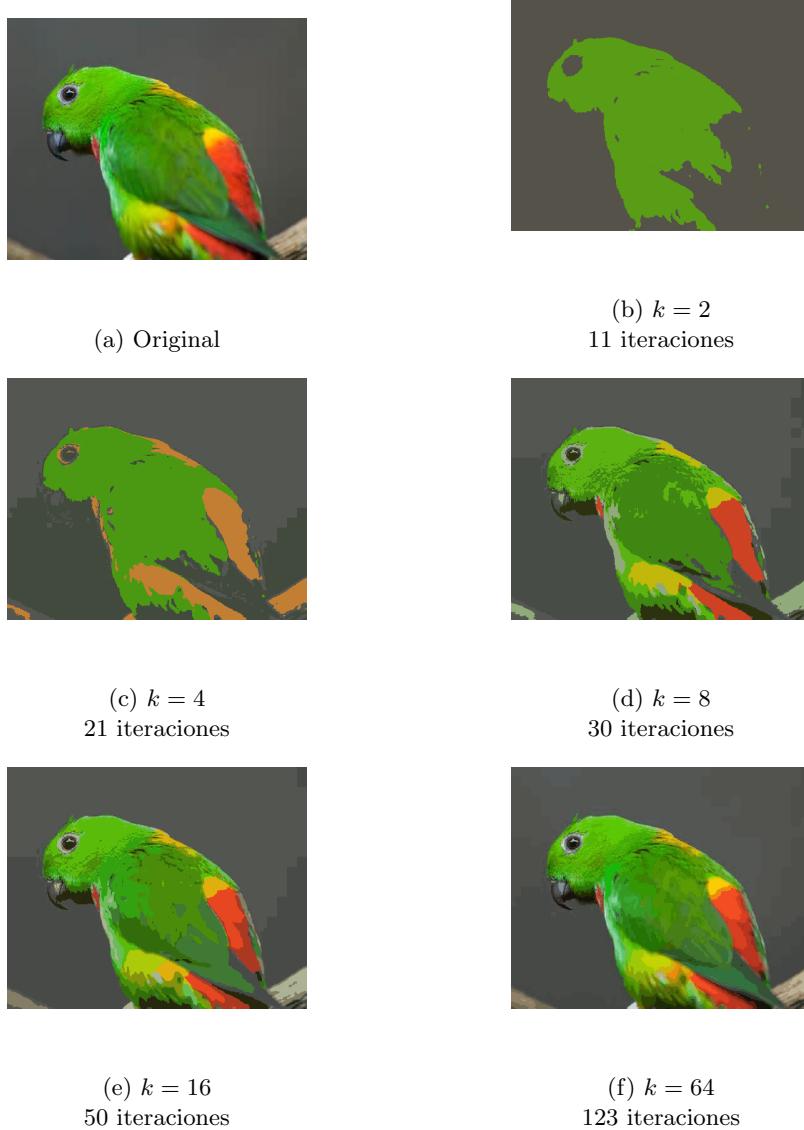
En la figura 1 se aprecia la reconstrucción de la imagen del lorículo en función del número de clústeres. Para dos clústeres, los colores distinguibles son el gris (fondo de la imagen) y el verde que es el color predominante en el cuerpo del lorículo. A medida que aumenta el número de clústeres, aparecen nuevos colores como el amarillo y el rojo de las plumas, además de tonos intermedios.

Por otro lado, en la figura 2 se ha repetido el mismo procedimiento para una imagen del DIIS de la EINA. El color predominante en esta imagen es el blanco (color de los edificios) y, al igual que ocurría con el lorículo, a medida que aumenta el número de clústeres se pueden apreciar más detalles de color en la imagen como, por ejemplo, el color de las hojas de los árboles.

4.2. Elección del número de clústeres

Siguiendo con el punto 2, y sabiendo que el tipo de dato empleado para almacenar las matrices en Matlab es `double` (8 bytes) [3]. Se puede deducir que:

- La imagen original almacena en su matriz para cada píxel su color. Este color ocupa 8 bytes, lo que hace un total de 24 bytes por píxel. Es decir, esta matriz ocupa $24m$ bytes siendo m el número de píxeles.
- μ almacena para cada centroide k su color. Haciendo el mismo cálculo que antes, esta matriz ocupará $24k$ bytes.
- c almacena el centroide k asignado a cada píxel. Como se almacena en un `double`, esta matriz ocupará $8m$ bytes.

Figura 1: Representación del lorículo en k clústeres.



(a) Original

(b) $k = 8$
91 iteraciones(c) $k = 16$
100 iteraciones(d) $k = 64$
135 iteracionesFigura 2: Representación del DIIS en k clústeres.

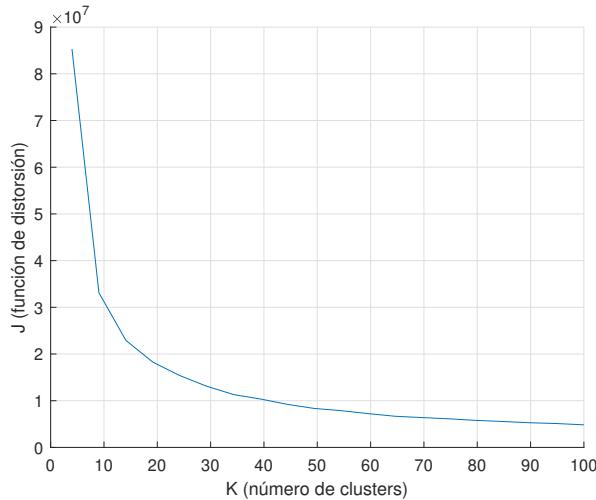


Figura 3: Evolución en la imagen del lorículo.

Por lo tanto, se obtiene el siguiente ratio de compresión:

$$r = \frac{8mn}{8nk + 8n} = \frac{mn}{kn + m}$$

Es decir, existe una relación inversamente proporcional entre el número de clústeres y el ratio de compresión. Si se quiere saber a partir de qué valor de k no «merece la pena» realizar la compresión, ya que se obtendría un mayor tamaño que la original, se puede calcular:

$$\frac{mn}{kn + m} = 1$$

Imagen del lorículo

Se ha realizado el cálculo para la imagen del lorículo (figura 1) y se tiene que a partir de un número de clústeres $k > 124160$ el ratio de compresión es inferior a 1. Por otro lado, en la figura 3 se observa que el error disminuye a medida que aumentan el número de clústeres hasta llegar a $k = m$ donde $J = 0$. Siguiendo el *método del codo*, se ha escogido un número de clústeres $k = 20$ con el que se obtienen los siguientes resultados para la imagen:

- 79 iteraciones.
- Ratio de compresión de 2,99. Es decir, la imagen original es casi 3 veces mayor que la comprimida.
- El valor de la función de distorsión es de $J = 1,7158 \cdot 10^7$.

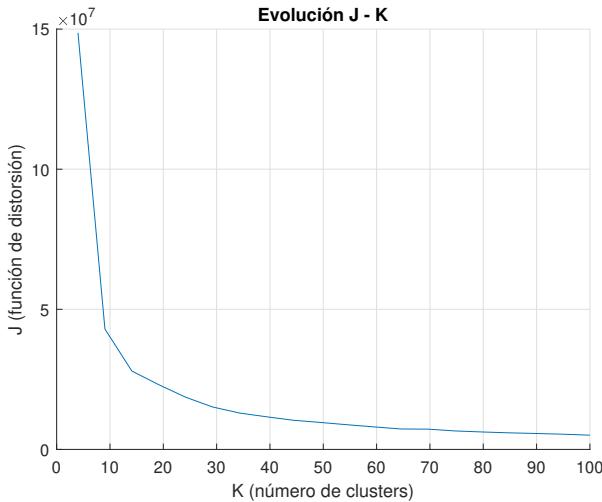


Figura 4: Evolución en la imagen del DIIS.

Imagen del DIIS

Se han repetido los cálculo para la imagen del DIIS (figura 2) y se tiene que a partir de un número de clústeres $k > 2,4 \cdot 10^5$ el ratio de compresión es inferior a 1. Siguiendo el *método del codo* en la figura 4, se ha escogido un número de clústeres $k = 15$ con el que se obtienen los siguientes resultados para la imagen:

- 176 iteraciones.
- Ratio de compresión de 2,99. Es decir, la imagen original es casi 3 veces mayor que la comprimida.
- El valor de la función de distorsión es de $J = 2,6634 \cdot 10^7$.

5. Conclusiones

Comparando el número de clústeres escogido para cada imagen tiene sentido que una imagen como la del lorículo, que tiene un mayor colorido, necesite de un valor de k más alto que la imagen del DIIS.

Si se comparan los resultados obtenidos, en términos de compresión, con los de la práctica anterior (Análisis de componentes principales) se aprecia un peor ratio de compresión utilizando Kmedias. Aunque, esto cambia según qué criterios se utilicen para escoger el número de clústeres o el valor k componentes principales.

En la práctica anterior se usaba como criterio de elección aquel valor de k que mantiene al menos el 90 % de la variabilidad. Y esta práctica se ha escogido el valor de k mediante el método del codo.

Referencias

- [1] Vibhor Agarwal. *Image Compression using K-means Clustering*. Mayo de 2018. URL: <https://medium.com/@agarwalvibhor84/getting-started-with-machine-learning-using-sklearn-python-7d165618eddf>.
- [2] Yo Fukushima. *curvspace - Generate evenly spaced points along an existing curve in 2D or 3D*. URL: <https://es.mathworks.com/matlabcentral/fileexchange/7233-curvspace>.
- [3] Xing Wan. “Application of K-means Algorithm in Image Compression”. En: *IOP Conference Series: Materials Science and Engineering* 563 (ago. de 2019), pág. 052042. DOI: [10.1088/1757-899x/563/5/052042](https://doi.org/10.1088/1757-899x/563/5/052042). URL: <https://doi.org/10.1088/1757-899x/563/5/052042>.