Práctica 15: SVD y compresión de imágenes gráficas

1. Introducción

Una imagen de $m \times n$ pixels en escala de grises puede ser representada como una matriz $A \in \mathbb{R}^{m\times n}$ cuyos elemento $a_{ij} \in [0, 1]$ representa la intensidad del pixel (i, j). Este enfoque permite aplicar toda la potencia del Álgebra Lineal numérica al tratamiento de imágenes bidimiensionales (sin dudas, para las imágenes en colores necesitamos tres matrices, representando los tres colores básicos RGB).

En particular, una imagen contiene mucha información redundante, es decir, que puede ser eliminada sin que el efecto visual sea notable. En particular, podríamos sustituir A por otra matriz B de rango prefijado más pequeño. En este sentido nos ayuda el siguiente resultado:

Teorema: *Sean $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ valores singulares no nulos de $A \in \mathbb{C}^{m\times n}$. Entonces para cada $k < r$, la distancia desde A al conjunto de matrices de rango k en la norma $\|\cdot\|_2$ es*

$$\sigma_{k+1} = \min_{\mathrm{rank}(B)=k} \|A - B\|_2 \,.$$

Por tanto, si $A = UDV^H$ es la descomposición en valores singulares de A, la matriz de rango k de mejor aproximación a A es

$$B = U \left(\begin{array}{c:c} S & 0 \\ \hdashline 0 & 0 \end{array}\right) V^H \,, \quad S = \mathrm{diag}(\sigma_1, \ldots, \sigma_k) \,.$$

Más aun, si U_k y V_k son las matrices compuestas por las primeras k columnas de U y V, respectivamente, entonces

$$B = U_k S V_k^H \,, \tag{1}$$

logrando codificar la imagen aproximada por medio de matrices de menor tamaño.

2. Lectura y representación de imágenes en Matlab

Matlab implementa los comandos `imread` e `imwrite` para la lectura y escritura, respectivamente, de archivos gráficos. Para leer una imagen, almacenada en el fichero **prueba.jpg** y asignarla a una variable A podemos ejecutar el comando

```
A=imread('prueba.jpg','jpg');
```

si la imagen inicial es en escala de grises el resultado será un arreglo bidimensional de tipo `uint8`, que podemos visualizar en Matlab por medio de las instrucciones

```
imagesc(A); colormap(gray);
```

Finalmente, para poder manipularlo como una matriz, necesitamos convertirlo a doble precision, por medio de

```
A=double(A);
```

Una vez transformada la matriz A, para convertirla a formato gráfico el proceso es el inverso:

```
A=uint8(A);
```

para la conversión de formato, tras lo cual podemos verla por medio de las mismas instrucciones `imagesc` y `colormap` vistas más arriba, y en caso de querer salvarla, basta ejecutar

```
imwrite(A,'prueba_mod.jpg','jpg');
```

3. Ejemplo práctico

Prepare un script que implemente los siguientes pasos:

1. Lea la imagen almacenada en el fichero **prueba.jpg** que se anexa, y representela en una ventana gráfica de Matlab.

2. Almacene la imagen en la matriz A en formato doble. Calcule su descomposición en valores singulares $A = UDV^H$ utilizando el comando `svd` de Matlab, y determine el tamaño de cada una de las matrices.

3. Dibuje en la escala semilogarítmica los valores singulares de la matriz A en una nueva ventana (use el comando `figure`).

4. Determine visualmente de forma aproximada el valor singular σ_k tal que todos los valores singulares anteriores representan aproximadamente el 10 % de todos los valores singulares de A. Asigne ese valor a la variable `tol`.

5. Trunque las matrices U, D y V dejando sólo aquellas columnas que corresponden a los valores singulares $\geq$ `tol`, obteniendo las matrices U_k, S y V_k. Determine el tamaño de cada una de las matrices. Construya la matriz B que aproxima a A según la fórmula (1).

6. Represente la imagen resultante en la tercera ventana gráfica.

Una vez que el script esté funcionando, halle el mayor valor de la variable `tol` para el cual considere que la imagen sigue siendo suficientemente nítida para poder leer el nombre de la plaza en la foto, y calcule qué % de valores singulares ha necesitado. ¿A qué ciudad de España corresponde ese lugar?