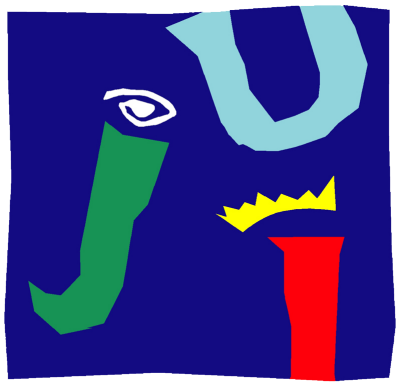

UNIVERSITAT JAUME·I

Grado en Matemática Computacional

Trabajo Final de Grado

Álgebra lineal y la compresión de imágenes

Autor:
Ariadna Tolós Vicente

Tutor académico:
Vicente Cervera Mateu

Fecha de lectura: __ de _____ de 20__
Curso académico 2021/2022

Resumen

En el presente documento se expone el trabajo final de grado en Matemática Computacional, dirigido por el profesor, D. Vicente Cervera Mateu. En este trabajo veremos la relación que existe entre las imágenes y las matrices, el álgebra lineal y las matemáticas. Partiremos con la introducción de varios conceptos teóricos para poder comprender mejor el contenido del trabajo. Continuaremos con una descomposición de matrices útil para la compresión de imágenes y un ejemplo numérico de este. Y finalizaremos con la elaboración de un programa que realiza la compresión de imágenes con el método que explicaremos durante el trabajo.

Palabras clave

Fotografía, SVD, matriz, compresión de imágenes, algebra lineal.

Keywords

Photography, SVD, matrix, image compression, linear algebra.

Agradecimientos

Me gustaría agradecer a todas las personas que han puesto un granito de arena para la realización de este trabajo al igual que han estado presente durante estos cuatro años de carrera.

Antes que nada, agradecer a mi tutor, D. Vicente Cervera Mateu, por guiarme en el presente trabajo, y ayudarme siempre que lo he necesitado en la realización de este.

A mi familia, gracias porqué gracias a ellos soy como soy, y todo esto es posible gracias a ellos. Especialmente a mis abuelos y mi hermana.

A mis abuelos que siempre me decían *"Si no se pot a la primera, a la segona, l'important és fer-ho."*, sé que ahora mismo estarían orgullosos de lo que he conseguido. Esto es por ellos.

A mi hermana, que ha sido la que realmente ha convivido conmigo todos los días durante estos cuatro años. Soportándome, día y noche, de todos mis agobios, llantos y malas contestaciones en mis momentos de desesperación.

A *los croquetis*: Andrea, Juan, Roderic y Sofía, por hacer que realizar un trabajo en pleno confinamiento fuera un momento de desconexión de todo lo otro, divertido aunque no por ello menos productivo.

Andrea, gracias por estar siempre preparada para hacernos reír incluso cuando no pensábamos ni que lo necesitábamos.

Juan, gracias por hacerme una *masterclass* de LaTeX y estar ahí siempre para resolverme todos los problema de compilación de este.

Roderic, por soportarme todas, absolutamente todas las clases de la carrera, incluso en mis peores días.

Sofía, gracias por tu paciencia y hacer siempre de mamá, la más responsable de la mayoría

de trabajos.

A todas mis amistades, nuevas y viejas, que siempre me han animado incluso cuando creía que no lo necesitaba y han confiado en mí incluso más que yo misma.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

Actualmente, estamos inmersos en la era de la información y la comunicación. En ello, toma una gran parte la fotografía. La fotografía es una técnica que gracias a ella, conservamos información. Es considerada un arte que consiste en capturar imágenes y guardarlas para su posterior revelación o conservación.

A día de hoy, la fotografía es tan importante que lo realmente raro es no capturar momentos y compartirlo con amigos, familiares, etc. Cada vez, estamos más inmersos en una mundo de redes sociales y *influencers*. En este mundo, es imprescindible las imágenes fotográficas, es por ello, que parece que todo gira entorno a la fotografía. Incluso, en muchos casos, cuando se quiere comprar un teléfono móvil, juzgamos la calidad de este en base a su calidad fotográfica.

Toda esta cantidad de información almacenada en imágenes ocupa espacio en nuestros dispositivos electrónicos. Y cuando hablamos de miles y miles de imágenes, puede ser un problema de memoria almacenar tanta cantidad de información. Es por ello, que surgen las técnicas para la compresión de imágenes.

Capítulo 2

Motivación y Objetivos

Desde el momento que se me planteó el tema de este trabajo, me causó una gran ilusión llevarlo a cabo. Me fascinó el hecho de poder aplicar en un trabajo académico una afición que siempre me ha apasionado, la fotografía. Unir dos ámbitos, la fotografía y las matemáticas, que me apasionan tenía que ser muy interesante.

Por lo que hace a la fotografía, fue cuando tuve mi primer teléfono móvil con cámara que sentí una gran admiración por hacer fotografías a todo lo que me rodeaba. Años más tarde, me regalaron mi primera cámara fotográfica y pasaba horas y horas frente a esta capturando momentos y aprendiendo a mejorar.

Por lo que hace a las matemáticas, siempre sentí una vocación por ellas. La frustración al no saber realizar un problema matemático, que aterraba a la mayoría de mis compañeros, a mí me provocaba una inquietud y necesidad de no parar de intentarlo hasta resolverlo. Y al conseguirlo, una satisfacción enorme por haberlo resuelto.

Pero lo que pocas veces pensé es que estos dos ámbitos estarían tan relacionados y terminaría profundizando tanto en esta investigación. La mayoría de la sociedad, nunca imaginaría que lo que se esconde detrás de una imagen, son números, cálculos y en general, matemáticas.

El objetivo de este presente trabajo es profundizar sobre un tema que no se ha dado o se ha dado poco durante los años de estudio de grado.

Capítulo 3

Desarrollo del TFG

3.1. Conceptos preeliminares

Antes que nada, en esta sección vamos a definir y enunciar definiciones y teoremas por si resultan de necesidad para comprender el presente trabajo.

Definición 3.1.1 *Sea $A \in \mathbb{R}^{n\times n}$ una matriz cuadrada. $\lambda \in \mathbb{R}$ es el **valor propio** correspondiente del **vector propio**, $x \in \mathbb{R}^n \backslash \{\bar{0}\}$, de A si:*

$$A \cdot x = \lambda \cdot x \tag{3.1}$$

Ejemplo 3.1.1 *El vector 3.2 es un vector propio de la matriz 3.3 con valor propio asociado $\lambda = 8$.*

$$x = \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix} \tag{3.2}$$

$$A = \begin{bmatrix} 3 & 2 & 4 \\ 2 & 0 & 2 \\ 4 & 2 & 3 \end{bmatrix} \tag{3.3}$$

Definición 3.1.2 *Sea V un espacio euclídeo n-dimensional y $(b_1, b_2, ..., b_n)$ una base de V. Se trata de una* ***base ortonormal*** *si cumple que:*

$$\langle b_i, b_j \rangle = 0 \quad i \neq j \tag{3.4}$$

$$\langle b_i, b_i \rangle = 1 \tag{3.5}$$

para todo $i, j = 1, ..., n$. En el caso de que solo se cumple 3.4 se llama ***base ortogonal****.*

Ejemplo 3.1.2 *Las bases 3.6 y 3.7 son bases ortonormales y 3.8 es una base ortogonal, respecto el producto escalar usual.*

$$B_1 = \{(cos\theta, sen\theta, 0), (-sen\theta, cos\theta, 0), (0, 0, 1)\} \tag{3.6}$$

$$B_2 = \{(\frac{3}{5}, \frac{4}{5}, 0), (-\frac{4}{5}, \frac{3}{5}, 0), (0, 0, 1)\} \tag{3.7}$$

$$B_3 = \{(2cos\theta, 2sen\theta, 0), (-sen\theta, cos\theta, 0), (0, 0, 1)\} \tag{3.8}$$

Teorema 3.1.1 ***Teorema espectral.*** *Si $A \in \mathbb{R}^{n \times n}$ es simétrica, existe una base ortonormal del correspondiente espacio vectorial V que consta de vectores propios de A, y cada valor propio es real.*
Una implicación directa del teorema es que existe la descomposición propia de una matriz A simétrica (con valores propios reales), y que podemos encontrar una base ortonormal de vectores propios tal que

$$A = P \cdot D \cdot P^T \tag{3.9}$$

donde D es una matriz diagonal que contiene los valores propios de$A$ y las columnas de P contienen los vectores propios. P es regular.

Ejemplo 3.1.3 *La matriz $A \in \mathbb{R}^3$ puede descomponerse en:*

$$A = \begin{bmatrix} \frac{5}{2} & -1 \\ -1 & \frac{5}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \cdot \begin{bmatrix} \frac{7}{2} & 0 \\ 0 & \frac{3}{2} \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} = P \cdot D \cdot P^T$$

Siendo P, D y P^T

$$P = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \qquad D = \begin{bmatrix} \frac{7}{2} & 0 \\ 0 & \frac{3}{2} \end{bmatrix} \qquad P^T = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

En las secciones 3.2 y 3.3 nos vamos a centrar en el contenido más teórico del presente trabajo. Después, en las secciones 3.4, 3.5 y 3.6 nos centraremos en la parte más práctica de la compresión de imágenes.

3.2. Compresión de imágenes

En esta sección nos centraremos en comprender en que consiste la compresión de imágenes[1] y veremos qué dos tipos existen[5].

Desde un punto de vista matemático, una fotografía de $m \times n$ píxeles es representada como matrices de enteros de dimensión $m \times n$. En el caso de una fotografía en escala de grises, está representada por una matriz cuyos valores oscilarán entre 0 y 255, dependiendo del nivel de blanco y negro, siendo 0 el blanco y 255 el negro. En el caso de tratarse de una fotografía a color (RGB[1]), tendrá 3 componentes, rojo, verde y azul. Cada componente, al igual que en blanco/negro, está representada por una matriz de la misma dimensión que la fotografía.

Como hemos dicho, en una fotografía en blanco/negro cada píxel se representa por un entero y este ocupa aproximadamente 1 byte. Por tanto, almacenar una fotografía de $m \times n$ píxeles ocupa aproximadamente, $m \times n$ bytes. Y en el caso de tratarse de color, se necesitará el triple de almacenamiento.

Para hacernos una idea con números, si consideramos una fotografía de 1280×1024 píxeles, el tamaño estándar de un monitor de pantalla, necesitaríamos aproximadamente 1,25 megabytes. Y si se tratará de una fotografía en color ocuparía 3,75 MB.

Si generalizamos y consideramos que aproximadamente cada fotografía en color ocupa 3,75 MB, ¿cuánta memoria necesitaríamos para almacenar todas las fotografías que conservamos en nuestros dispositivos electrónicos?

Actualmente, que es tan fácil el acceso a realizar una fotografía, que no resulta nada raro realizar más de una foto en diferentes ocasiones de un día normal como el amanecer, las comidas realizadas, etc. Por ello, una fantástica herramienta para optimizar el almacenamiento que ocupan nuestras fotografías es la compresión de imágenes.

La compresión de imágenes es una herramienta que trata de reducir la cantidad de datos necesaria para representar una imagen digital. La idea básica del proceso de reducción de datos es eliminar la redundancia de la información.

[1]Siglas en inglés: red, green, blue (rojo, verde, azul). Modelo cromático para representar diferentes colores a partir de la mezcla de estos tres colores primarios.

Ventajas	Desventajas
Partes de la imagen intactas Sin pérdida de calidad de la imagen Es un proceso reversible	Imagen comprimida demasiado grande La decodificación es un desafío

Cuadro 3.1: Ventajas y desventajas de la compresión de imágenes sin pérdida de la información

En la compresión de imágenes existen dos tipos bien diferenciados: compresión de imágenes sin pérdida de la información y con pérdida de la información. Esto varía según el cambio del tamaño del archivo de la imagen. En el primer tipo, se garantiza que la calidad de la imagen permanezca intacta, mientras que en el segundo se elimina algunas partes redundantes para obtener un tamaño más pequeño y ocupar menos memoria.

En las siguientes subsecciones 3.2.1 y 3.2.2 profundizaremos sobre estos tipos de compresión.

3.2.1. Compresión de imágenes sin pérdida de la información

La compresión de imágenes sin pérdida de la información es el proceso de cambiar de tamaño de las imágenes en una versión más pequeña. En este tipo de compresión, las imágenes conservan todos los datos, por lo que no varía la calidad de la imagen. En este tipo de compresión, se considera que las imágenes están basadas en la entropía, una técnica que codifica los datos sin conocer la naturaleza de los mismos. Como consecuencia de conservar todos los datos, la imagen final reconstruida es exactamente igual que la imagen original. Este tipo de compresión usa métodos estadísticos.

Se trata de un método excelente para reducir el tamaño de la imagen, pero el resultado no puede ser suficiente bueno. Eso se debe a que en este tipo de compresión no se elimina ninguna parte de la imagen.

Con este tipo de compresión, se consigue la máxima calidad en las imágenes aunque el grado de compresión es bajo, inferior al 50 %. Las únicas limitaciones que puede tener son propias de la imagen original, como: la resolución, número de colores, etc.

Un ejemplo sería convertir imagen de 15 MB a 10 MB. Sin embargo, con el tamaño de la imagen comprimida, seguiríamos sin poder subir la imagen a una página web.

Ventajas	Desventajas
Tamaño de la imagen muy reducido Tiempo de carga rápido Ideal para sitios web	La imagen pierde componentes Es irreversible

Cuadro 3.2: Ventajas y desventajas de la compresión de imágenes con pérdida de la información

3.2.2. Compresión de imágenes con pérdida de la información

La compresión de imágenes con pérdida de la información reduce el tamaño de la imagen eliminando algunas partes de esta. Utilizando este tipo de compresión, puede llegarse a obtener una versión más pequeña con una mínima diferencia de calidad.

En este tipo de compresión, la imagen resultante es más o menos diferente de la imagen original, aunque normalmente las pérdidas son tan aceptables que son casi invisibles para el ojo humano. Suelen utilizarse principalmente para las imágenes que tienen información redundante y tiende a ser eliminada o reducida. El problema de este tipo de compresión surge de imágenes borrosas, mosaicos, etc.

Este tipo de compresión suele basarse en una aproximación matemática de la imagen. En la mayoría de los casos, debe configurar un parámetro que determina el grado de compresión de la imagen. Un ejemplo, podría ser convertir una imagen de 15 MB en 2200 Kb o en 400 Kb.

Para comprender mejor las diferencias entre compresión de imágenes sin pérdida y con pérdida, podemos observar las tablas 3.1 y 3.2 comparativas de ambas técnicas.

En el presente trabajo la técnica que vamos a utilizar para la compresión de imágenes va a ser la descomposición en valores singulares. Se trata de una técnica de compresión de imágenes con pérdida de la información. En la sección 3.3 vamos a profundizar más en ello.

3.3. Descomposición en valores singulares

En esta sección vamos a explicar en que consiste el método de compresión de imágenes con descomposición en valores singulares[3, 4].

La descomposición en valores singulares, SVD abreviatura del inglés *Singular Value Descomposition*, es una herramienta para la factorización de matrices. Consiste en realizar operaciones

con matrices para llegar a factorizar una matriz en matrices más simples. A partir de estas más simples, analizarlas, eliminar los valores relevantes y finalmente volver a unirlas con menor información para que la matriz obtenida sea parecida a la matriz original. Esta descomposición es conocido como el "Teorema de descomposición en valores singulares"[4].

Teorema 3.3.1 ***Teorema SVD.*** *Sea $A \in \mathbb{R}^{m\times n}$ una matriz rectangular de rango r ,$0 < r \leq$ mín(m, n). El SVD de A es la descomposición de la forma:*

$$A = U \cdot \Sigma \cdot V^T \tag{3.10}$$

Siendo $U \in \mathbb{R}^{m\times m}$ y $V \in \mathbb{R}^{n\times n}$ dos matrices ortogonales con vectores ortonormales columna, u_i, $i = 1, ..., m$ y v_j, $j = 1, ..., n$, respectivamente. Σ es una matriz de dimensión $m \times n$ con Σ_{ii}=$\sigma_i \geq 0$ y $\Sigma_{ij} = 0, i \neq j$.
Los elementos de la diagonal $\sigma_i, i = 1, ..., r$, de Σ se llaman valores singulares, u_i se llaman vectores singulares izquierdos y v_i vectores singulares derechos.
Por convención, los valores singulares los ordenamos: $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r \geq 0$.

Demostración.

Lo demostraremos en el siguiente apartado donde veremos simultáneamente la construcción de U y V. □

La matriz de valores singulares Σ es única, pero tenemos que contemplar que puede ser una matriz rectangular. Σ será rectangular cuando A sea rectangular, ya que tendrán la misma dimensión. Además, como se trata de una matriz diagonal, exigirá añadir o bien filas, o bien columnas de 0, dependiendo de la estructura de la matriz.

En el caso en que $m > n$: Σ tendrá más filas que columnas, luego requerirá un relleno de $(m - n)$ filas de 0. Σ tendrá la siguiente estructura:

$$\Sigma = \begin{bmatrix} \sigma_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_n \\ 0 & \cdots & 0 \\ \vdots & & \vdots \\ 0 & \cdots & 0 \end{bmatrix}_{m\times n} \tag{3.11}$$

En el otro caso en que $n > m$: Σ tendrá más columnas que filas, luego requerirá un relleno de $(n - m)$ columnas de 0. Σ tendrá la siguiente estructura:

$$\Sigma = \begin{bmatrix} \sigma_1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & \sigma_m & 0 & \cdots & 0 \end{bmatrix}_{m \times n} \tag{3.12}$$

Supongamos que ya hemos obtenido la matriz A descompuesta en U, V^T y Σ, después de haber aplicado el *Teorema* 3,3,1. Al haber ordenado de mayor a menor los valores singulares, resulta interesante que los píxeles más relevantes de la imagen se almacenan en las primeras filas de la matriz A. Es decir, los vectores más significativos de U están en las primeras columnas y los de V^T en las primeras filas.

Aprofundizando con lo que hemos dicho antes, si nos quedamos solo con el valor más importante de Σ, σ_1, tendríamos unas submatrices de dimensiones: $m \times 1$, 1×1 y $1 \times n$. Y el resultado de multiplicarlas dará:

$$\begin{aligned} A = \begin{bmatrix} u_{11} \\ \vdots \\ u_{m1} \end{bmatrix}_{m \times 1} \begin{bmatrix} \sigma_1 \end{bmatrix}_{1 \times 1} \begin{bmatrix} v_{11} & \cdots & v_{1n} \end{bmatrix}_{1 \times n} = \begin{bmatrix} u_{11} \cdot \sigma_1 \\ \vdots \\ u_{m1} \cdot \sigma_1 \end{bmatrix}_{m \times 1} \begin{bmatrix} v_{11} & \cdots & v_{1n} \end{bmatrix}_{1 \times n} = \\ = \begin{bmatrix} u_{11} \cdot \sigma_1 \cdot v_{11} & \cdots & u_{11} \cdot \sigma_1 \cdot v_{1n} \\ \vdots & \ddots & \vdots \\ u_{m1} \cdot \sigma_1 \cdot v_{11} & \cdots & u_{m1} \cdot \sigma_1 \cdot v_{1n} \end{bmatrix}_{m \times n} \end{aligned} \tag{3.13}$$

Como vemos en 3.13, hemos obtenido una matriz de la misma dimensión que A que es una aproximación de A. En la sección 3.5, estudiaremos como calcular el error de la aproximación. A partir de este error, podremos considerar si aún es una buena aproximación o no y en caso negativo cogeríamos nuevas filas y columnas de U y V^T y valores de Σ.

3.4. Construcción de U y V

En esta sección vamos a buscar las matrices U y V al mismo tiempo que demostraremos el Teorema 3.3.1 de la sección 3.3. Si recordamos la ecuación del Teorema 3.1.1 y lo comparamos con la ecuación 3.10:

$$(3{,}9) : A = P \cdot D \cdot P^T \qquad\qquad (3{,}10) : A = U \cdot \Sigma \cdot V^T$$

Como podemos observar, sería la misma descomposición si: $D = \Sigma$ y $U = V = P$. Para poder desarrollar esa relación entre matrices trabajaremos con la matriz $M = A^T \cdot A$, que es cuadrada.

Por una parte, para poder diagonalizar M, es necesario y suficiente que exista una base ortonormal del espacio vectorial formada por vectores propios. Supongamos que existe esta base vectorial y apliquemos el Teorema 3.1.1 *Teorema de diagonalización de matrices.* Existirá una matriz P regular tal que:

$$M = P \cdot D \cdot P^T = P \cdot \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_n \end{bmatrix} P^T \tag{3.14}$$

Y por otra parte, aplicamos el Teorema 3.3.1, que queremos demostrar, por separado a A y A^T:

$$M = A^T \cdot A = (U \cdot \Sigma \cdot V^T)^T \cdot (U \cdot \Sigma \cdot V^T) = (V \cdot \Sigma^T \cdot U^T) \cdot (U \cdot \Sigma \cdot V^T)$$

Como U es ortonormal tenemos que $U \cdot U^T = I$:

$$M = A^T \cdot A = V \cdot \Sigma^T \cdot \Sigma \cdot V^T = V \cdot \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_n^2 \end{bmatrix} \cdot V^T \tag{3.15}$$

Si igualamos las ecuación 3.14 y 3.15 obtenemos que:

$$P = V \qquad P^T = V^T \qquad \sigma_i^2 = \lambda_i \rightarrow \sigma_i = \sqrt{\lambda_i}$$

Análogamente, volvemos a aplicar el teorema con $M' = A^T A$:

$$M' = A \cdot A^T = (U \cdot \Sigma \cdot V^T) \cdot (U \cdot \Sigma \cdot V^T)^T = (U \cdot \Sigma \cdot V^T) \cdot (V \cdot \Sigma^T \cdot U^T)$$

Como V también es ortonormal, $V \cdot V^T = I$:

$$M' = A \cdot A^T = U \cdot \Sigma \cdot \Sigma^T \cdot U^T = U \cdot \begin{bmatrix} \sigma_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sigma_n^2 \end{bmatrix} \cdot U^T \tag{3.16}$$

Luego, el *Teorema* 3,1,1 nos dice que podemos diagonalizar M y encontrar una base ortonormal de vectores propios de M que serán los que colocaríamos en P. Pero, $M = A^T \cdot A$ y $M' = A \cdot A^T$ tienen los mismos valores propios distintos de cero[2]. Luego, si escogemos M hemos conseguido un conjunto ortonormal de vectores de la derecha con los que construiremos en V. Si hubieramos escogido $M' = A \cdot A^T$ habríamos conseguido un conjunto ortonormal de vectores a la izquierda con los que construiríamos U. Para terminar la construcción del SVD, solo falta

[2]Como se trata de un resultado no muy obvio, está demostrado al final de esta sección en forma de en el teorema 3.4.1.

conseguir los vectores de U conectados con los de V o viceversa en el caso de escoger M'. Para conseguirlo, utilizaremos que la imágenes de v_i sobre A también tienen que ser ortogonales. Es decir, necesitamos que $< A \cdot v_i, A \cdot v_j > = 0$ para todo par de vectores tal que $i \neq j$.Veamos si es cierto:

$$< A \cdot v_i, A \cdot v_j >= (A \cdot v_i)^T \cdot (A \cdot v_j) = v_i^T \cdot A^T \cdot A \cdot v_j = v_i \cdot (\lambda_j \cdot v_j) = \lambda_j \cdot (v_i^T \cdot v_j) = \lambda_i \cdot 0 = 0$$

Sabemos que $v_i^T \cdot v_j = 0$ porque son ortogonales. Luego, hemos visto que $< A \cdot v_i, A \cdot v_j > = 0$ para todo par de vectores que $i \neq j$.

Para el caso de $m \geq r \leq \text{mín}(m, n)$ tenemos $\{Av_1, ..., Av_r\}$ es una base de dimensión r d'un subespacio de $\mathbb{R}^n$. Ahora definimos u_i como:

$$u_i = \frac{Av_i}{\|Av_i\|} = \frac{Av_i}{\|\lambda_i v_i\|} = \frac{Av_i}{\sqrt{\lambda_i}\|v_i\|} = \frac{1}{\sqrt{\lambda_i}} Av_i = \frac{1}{\sigma_i} Av_i \tag{3.17}$$

Ahora tenemos los llamados vectores singulares derechos que son los v_i, vectores propios, y la normalización de estos sobre la imagen de A serán los llamados vectores singulares izquierdos. Hemos obtenido dos bases ortonormales conectadas por la matriz Σ. Reordenando obtenemos:

$$\sigma_i \cdot u_i = A \cdot v_i \qquad\qquad i = 1, ..., r \tag{3.18}$$

Como vemos, esta ecuación es muy parecida a la ecuación de valores propios de la Definición 3.1.1, aunque los vectores a derecha e izquierda no son los mismos en este caso.

Para $n < m$, la ecuación 3.18 se cumple cuando $i \leq n$, pero no sabemos nada de u_i cuando $i > n$. Pero de lo contrario, si $m < n$ la ecuación 3.18 se cumple solo cuando $i < m$, y cuando $i > m$, tenemos $A \cdot v_i = 0$ y v_i forma un conjunto ortonormal.

Es decir, el SVD proporciona una base ortonormal del núcleo de A, el conjunto de x con $A \cdot x = 0$. Si pasamos la ecuación anterior a forma matricial, con u_i como columnas de U, v_i como columnas de V y σ_i como los valores de la diagonal de la matriz Σ, obtenemos:

$$A \cdot V = U \cdot \Sigma$$

Despejando A:

$$A \cdot V \cdot V^{-1} = U \cdot \Sigma \cdot V^{-1} \rightarrow A = U \cdot \Sigma \cdot V^{-1} \rightarrow A = U \cdot \Sigma \cdot V^T$$

Luego como V es ortonormal $V^{-1} = V^T$, luego hemos llegado a la ecuación del SVD:

$$A = U \cdot \Sigma \cdot V^T \tag{3.19}$$

□

Teorema 3.4.1 *Sean $A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{n \times m}$ las matrices $A \cdot B$ y $B \cdot A$ tienen los mismos valores propios no nulos.*

Demostración.
$\lambda \neq 0$ es un valor propio de $A \cdot B$. Por la definición 3.1.1 se cumple que:

$$\exists \quad x \in \mathbb{R}^n \backslash \{\bar{0}\}, \quad x \neq 0 \quad / \quad A \cdot B \cdot x = \lambda \cdot x$$

Llamamos y al vector $y = B \cdot x$, $y \neq 0$, ya que $y = 0$ implica $x = 0$. Y calculamos:

$$B \cdot A \cdot y = B \cdot A \cdot B \cdot x = B \cdot (A \cdot B \cdot x) = B \cdot (\lambda \cdot x) = \lambda \cdot (B \cdot x) = \lambda \cdot y \tag{3.20}$$

Hemos llegado a que:

$$B \cdot A \cdot y = \lambda \cdot y$$

Luego, hemos llegado a que λ también es valor propio distinto de nulo de la matriz $B \cdot A$. Finalmente, como hemos cogido un λ genérico hemos demostrado que $A \cdot B$ y $B \cdot A$ tiene los mismos valores propios distintos de cero. □

3.5. Aplicación del SVD

Sean $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$, $\Sigma \in \mathbb{R}^{m \times n}$ las matrices:

$$U = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1m} \\ u_{21} & u_{22} & \cdots & u_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1} & u_{m2} & \cdots & u_{mm} \end{bmatrix} \qquad V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{bmatrix}$$

Y Σ que dependerá de si $m > n$ o $m < n$ será de la forma de 3.11 o 3.12, respectivamente. Llamamos $r = rang(A)$.

Sea $A = [a_{ij}] \in \mathbb{R}^{m \times n}, i = 1, ..., m \ , \ j = 1, ..., n$ representa una imagen de $m \times n$ píxeles.

Aplicamos el *Teorema* 3,3,1.

$$A = U \cdot \Sigma \cdot V^T = \sum_{i=1}^{r} \sigma_i \cdot u_i \cdot v_i^T = \sum_{i=1}^{r} \sigma_i \cdot A_i$$

Siendo u_i, v_i y A_i:

$$u_i = \begin{bmatrix} u_{1i} \\ u_{2i} \\ u_{3i} \\ \vdots \\ u_{mi} \end{bmatrix} \qquad v_i = \begin{bmatrix} v_{1i} \\ v_{2i} \\ v_{3i} \\ \vdots \\ v_{ni} \end{bmatrix} \qquad A_i = u_i \cdot v_i^T$$

Como hemos visto en la ecuación 3.13, cogiendo solo un valor de σ_1, la aproximación de A se podía obtener multiplicando $\sigma_1 \cdot u_1 \cdot v_1^T$. Debido a que Σ es una matriz diagonal, la multiplicación de $U \cdot \Sigma \cdot V^T$ puede escribirse como el sumatorio de la multiplicación del i-ésimo autovalor por los i-ésimos vectores de u y v. Expandimos las matrices para entenderlo mejor, suponemos que $m > n$ (en el otro caso sería exactamente igual, solo variaría la forma de Σ):

$$A = U \cdot \Sigma \cdot V^T = \begin{bmatrix} u_1 & u_2 & \cdots & u_r & \cdots & u_m \end{bmatrix} \begin{bmatrix} \sigma_1 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & \sigma_r & \vdots \\ 0 & \cdots & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & \cdots & 0 \end{bmatrix}_{m \times n} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_r \\ \vdots \\ v_n \end{bmatrix} =$$

$$= \begin{bmatrix} u_1 \cdot \sigma_1 & u_2 \cdot \sigma_2 & \cdots & u_r \cdot \sigma_r \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_r \\ \vdots \\ v_n \end{bmatrix} = u_1 \cdot \sigma_1 \cdot v_1 + u_2 \cdot \sigma_2 \cdot v_2 + \cdots + v_n \cdot \sigma_n \cdot v_n = \tag{3.21}$$

$$= \sigma_1 \cdot A_1 + \sigma_2 \cdot A_2 + \cdots + \sigma_n \cdot A_n = \sum_{i=1}^{n} \sigma_i \cdot A_i$$

La mejor aproximación de rango k, será la imagen comprimida y la definimos como:

$$A_k = \sum_{i=1}^{k} \sigma_i \cdot u_i \cdot v_i^T = \sum_{i=1}^{k} \sigma_i \cdot A_i$$

Para poder calcular el error entre la imagen original, A y su aproximación $A(k)$, necesitamos definir una norma, que nos haga una idea de lo parecido o diferente que es la aproximación de la imagen real. Y esta norma la definimos de la siguiente forma:

Definición 3.5.1 *La **norma espectral de una matriz** $A \in \mathbb{R}^{m\times n}$ se define como:*

$$\|A\|_2 := \max_x \frac{\|Ax\|_2}{\|x\|_2}$$

Siendo $x \in R^n \setminus \{\bar{0}\}$.

Teorema 3.5.1 *La norma espectral de A es su valor singular más grande, es decir, σ_1.*

$$\|A\|_2 = \sigma_1 \tag{3.22}$$

Demostración. [6]

Por la definición 3.5.1:

$$\|A\|_2 = \max_x \frac{\|Ax\|_2}{\|x\|_2} = \max_x \left(\frac{\sqrt{x^T \cdot A^T \cdot A \cdot x}}{\sqrt{x^T \cdot x}}\right) = \max_x \sqrt{\frac{x^T \cdot A^T \cdot A \cdot x}{x^T \cdot x}} \tag{3.23}$$

Llamamos el valor α y la matriz Σ a:

$$\alpha = \frac{x^T \cdot A^T \cdot A \cdot x}{x^T \cdot x} \qquad\qquad \Sigma = A^T \cdot A \tag{3.24}$$

Para demostrarlo vamos a centrarnos en los x / $\|x\| = 1$. Entonces el valor α nos queda:

$$\alpha = \frac{x^T \cdot \Sigma \cdot x}{\|x\|^2} = x^T \cdot \Sigma \cdot x \tag{3.25}$$

Operamos sobre la ecuación 3.25:

$$\alpha = x^T \cdot \Sigma \cdot x = x^T \cdot V \cdot \Lambda \cdot V^T \cdot x = b^T \cdot \Lambda \cdot b = \sum_{i=1}^{n} \lambda_i \cdot b_i^2 \tag{3.26}$$

En la ecuación 3.26, hemos aplicado el Teorema 3.1.1 sobre Σ ya que es una matriz simétrica. Hemos nombrado b al vector resultante de la multiplicación de la matriz por vector: $b = V^T \cdot x$. Y como Λ es la matriz diagonal con los valores propios de Σ podemos escribirla como el sumatorio. Ahora vamos a operar sobre $\sum_{i=1}^{n} b_i^2$:

$$\sum_{i=1}^{n} b_i^2 = b^T \cdot b = x^T \cdot V \cdot V^T \cdot x = x^T \cdot x = 1 \tag{3.27}$$

En la ecuación 3.27 podemos ver que V, es la matriz de vectores propios en columna y como estos son ortogonales, V es ortogonal, por lo que $V \cdot V^T = I$, y llegamos a que b tiene que ser

un vector unitario.
Recapitulando resultados, hemos llegado a que α definido en la ecuación 3.25 es:

$$\alpha = \sum_{i=1}^{n} \lambda_i \cdot b_i^2 \tag{3.28}$$

Siendo b_i los elementos del vector $b = (\ b_1\ b_2\ ...\ b_n\ )^T$.
Ahora, para poder demostrar el teorema, queda obtener el máximo de la raíz cuadrada de el valor de α. Y este será máximo cuando α sea máximo. Luego, por la definición de α que hemos llegado en 3.28, esta será máxima cuando $i = 1$. Ya que los valores propios los ordenamos de modo que $\lambda_1 > \lambda_2 > ... > \lambda_n$. El valor propio más grande es λ_1 y se obtendrá cuando b será:

$$b = (\ 1\ 0\ 0\ ...\ 0\ )^T$$

Ahora que ya conocemos b, podemos buscar que vector será x a partir de b:

$$\begin{aligned} V^T \cdot x &= b \\ V \cdot V^T \cdot x &= V \cdot b \\ x &= v_1 \end{aligned} \tag{3.29}$$

En la ecuación 3.29 hemos llegado a que x es el vector propio asociado al mayor valor propio de Σ.
Luego sustituimos el resultado en la ecuación 3.22 y obtenemos que:

$$\|A\|_2 = \max_x \sqrt{\frac{x^T \cdot A^T \cdot A \cdot x}{x^T \cdot x}} = \max_x \sqrt{\frac{x^T \cdot A^T \cdot A \cdot x}{1}} = \max_x \sqrt{\alpha} = \max_{v_1} \sqrt{\alpha} = \sqrt{\lambda_1} = \sigma_1 \tag{3.30}$$

En la ecuación 3.30 hemos aplicado que la raíz de los valores propios de Σ son los valores singulares. Hemos demostrado a que la norma espectral de A es su valor singular más grande. □

Teorema 3.5.2 ***Teorema d'Eckart-Young*** *Consideramos la matriz $A \in \mathbb{R}^{m \times n}$ de rango r y sea $B \in \mathbb{R}^{m \times n}$ una matriz de rango k. Para cualquier $k \leq r$ con:*

$$A_k = \sum_{i=1}^{k} \sigma_i \cdot u_i \cdot v_i^T \tag{3.31}$$

Se tiene que:

$$A_k = argmin_{rk(B)=k} \|A - B\|_2 \tag{3.32}$$

$$\|A - A_k\|_2 = \sigma_{k+1} \tag{3.33}$$

El teorema d'Eckart-Young indica explícitamente el error de la aproximación de A de rango k. Si consideramos esta aproximación obtenida con el SVD como una proyección de la matriz A en un espacio de dimensiones inferiores de rango como máximo k. Entonces, el SVD minimizaría el error entre A y todas las otras aproximaciones posibles.

Demostración. [2]

Partimos de la ecuación 3.33 inicial del teorema 3.5.2 y operamos sobre esta.

$$\|A - A_k\|_2 = \left\| \sum_{i=1}^{n} \sigma_i \cdot u_i \cdot v_i^T - \sum_{i=1}^{k} \sigma_i \cdot u_i \cdot v_i^T \right\|_2 = \left\| \sum_{i=k+1}^{n} \sigma_i \cdot u_i \cdot v_i^T \right\|_2 \tag{3.34}$$

Si definimos D una nueva matriz diagonal de la forma:

$$D = \begin{pmatrix} 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & \sigma_{k+1} & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & \sigma_n \end{pmatrix} \tag{3.35}$$

Podemos escribir el sumatorio de la ecuación 3.34 en forma matricial de la siguiente forma que llamaremos C a la matriz resultante:

$$\left\| \sum_{i=k+1}^{n} \sigma_i u_i v_i^T \right\|_2 = \left\| U \cdot D \cdot V^T \right\|_2 = \|C\|_2 \tag{3.36}$$

Ahora si aplicamos el Teorema 3.5.1 con C obtenemos que $\|C\|_2$ es el mayor valor singular que este es: σ_{k+1}:

$$\|C\|_2 = \|A - A_k\|_2 = \sigma_{k+1} \tag{3.37}$$

Hemos terminado de demostrar la ecuación 3.33. Seguimos con la demostración de la ecuación 3.32. Para demostrarlo, primero vamos a intentar buscar un vector que nos facilite la demostración.
Sea $w \in \mathbb{R}^{k+1}$ vector unitario:

$$\|w\|_2 = 1 \rightarrow w_1^2 + w_2^2 + ... + w_k^2 + w_{k+1}^2 = 1$$

Además w tiene que cumplir que:

$$B \cdot w = 0 \tag{3.38}$$

$$w = \alpha_1 \cdot v_1 + \alpha_2 \cdot v_2 + ... + \alpha_k \cdot v_k + \alpha_{k+1} \cdot v_{k+1} \tag{3.39}$$

w podrá escribirse como combinación lineal del espacio vectorial formado por los vectores v_i, $i = 1, ..., k+1$, $w \in < v_1, v_2, ..., v_{k+1} >$.
Cogemos $V' = [v_1, v_2, ..., v_{k+1}]$, $V' \in \mathbb{R}^{n \times (k+1)}$. Luego podemos escribir w como: $w = V' \cdot \alpha$ siendo $\alpha = (\ \alpha_1\ \alpha_2\ ...\ \alpha_k\ \alpha_{k+1}\ ) \in \mathbb{R}^{k+1}$. Vamos a probar de encontrar α.

$$B \cdot w = 0 \rightarrow B \cdot V' \cdot \alpha = 0 \tag{3.40}$$

Llamamos $C = B \cdot V'$, $C \in \mathbb{R}^{m\times(k+1)}$, entonces la ecuación 3.40 queda:

$$B \cdot V' \cdot \alpha = 0 \rightarrow C \cdot \alpha = 0$$

Tenemos $k+1$ variables, y k ecuaciones independientes, luego se trata de un sistema indeterminado, por lo que α tiene infinitas soluciones. Consideramos la norma 3.5.1 y aplicamos la definición y operamos:

$$\|A - B\|_2 = \max_{\|w\|_2} \|(A - B)w\|_2 \geq \|(A - B)w\|_2 = \|Aw - Bw\|_2 = \|Aw\|_2 \tag{3.41}$$

En la ecuación 3.41 vemos que $\|(A - B)w\|_2$ se reduce a $\|Aw\|_2$, ya que hemos visto en la ecuación 3.40, $B \cdot w$ se anula. Hemos llegado a:

$$\|A - B\|_2 \geq \|A \cdot w\|_2 \tag{3.42}$$

Queremos encontrar cuál es el mínimo de $\|A \cdot w\|_2$ Para ello, como $A \cdot w$ puede escribirse como en 3.43, calculamos su norma y obtenemos 3.44.

$$A \cdot w = \alpha_1 \cdot \sigma_1 \cdot v_1 + ... + \alpha_{k+1} \cdot \sigma_{k+1} \cdot v_{k+1} \tag{3.43}$$

$$\|A \cdot w\|_2 = \sqrt{(\alpha_1 \cdot \sigma_1)^2 + ... + (\alpha_{k+1} \cdot \sigma_{k+1})^2} \tag{3.44}$$

Ahora buscamos el mínimo de 3.44.

$$\min_{\|w\|_2=1} \|A \cdot w\|_2 = \min \sqrt{(\alpha_1 \cdot \sigma_1)^2 + ... + (\alpha_{k+1} \cdot \sigma_{k+1})^2}$$

De un modo similar a como hemos demostrado el teorema 3.5.1. Como los σ_i están ordenados, $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_{k+1}$, la norma será menor cuando $\alpha_{k+1} = 1$ y $\alpha_i = 0 \; \forall$ i $\neq$ k+1. Y el valor de la norma será:

$$\min_{\|w\|_2=1} \|A \cdot w\|_2 = \sigma_{k+1}$$

Ahora recapitulando todo lo que hemos visto la ecuación 3.42 nos queda:

$$\|A - B\|_2 \geq \alpha_{k+1}$$

Pero hemos visto en 3.37, que:

$$\|A - A_k\|_2 = \alpha_{k+1} \rightarrow A_k = \underset{rang(B)=k}{\arg\min} \|A - B\|_2 \tag{3.45}$$

Nota Para determinar A_k solo necesitamos:

$$m \cdot k + n \cdot k = (m + n) \cdot k$$

Definición 3.5.2 *La imagen original de $m \times n$. El **grado de compresión** es:*

$$1 - \frac{(m+n) \cdot k}{m \cdot n} \tag{3.46}$$

*Y el **error relativo** es:*

$$\frac{\|A - A_k\|_2}{\|A\|_2} = \frac{\sigma_{k+1}}{\sigma_1} \tag{3.47}$$

Ejemplo 3.5.1 *En una imagen con dimensiones $m \times n$ siendo $m = 1024$ y $n = 768$:*

$$m \cdot n = 1024 \cdot 768 = 786432$$

$$m + n = 1024 + 768 = 1792$$

Si $k = 100$:

$$1 - \frac{179200}{786432} = 77\,\%$$

En una imagen de 1024×768 píxeles con $k = 100$ tiene un grado de compresión del $77\,\%$.

Ejemplo númerico de SVD

Encontramos la descomposición para A:

$$A = \begin{bmatrix} 1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix}$$

Necesitamos de los vectores singulares derechos v_i, los valores singulares σ_i y los vectores singulares izquierdos u_i.
Encontrar vectores singulares

Llamamos $M = A^T \cdot A$ para aplicar el *Teorema de Diagonalización* y así obtener P que será el V de la descomposición de la matriz original A.

$$M = A^T \cdot A = \begin{bmatrix} 1 & -2 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 5 & -2 & 1 \\ -2 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Ahora aplicamos sobre $A^T \cdot A$ el Teorema 1 para diagonalizar la matriz. Buscamos los valores propios.

$$M - \lambda \cdot I = \begin{vmatrix} 5-\lambda & -2 & 1 \\ -2 & 1-\lambda & 0 \\ 1 & 0 & 1-\lambda \end{vmatrix} = \lambda \cdot (1-\lambda) \cdot (-6+\lambda)$$

Resolvemos la ecuación:

$$\lambda \cdot (1-\lambda) \cdot (-6+\lambda) = 0 \rightarrow \lambda = \ 0,\ 1,\ 6$$

Los valores propios son: 0, 1, 6. Los ordenamos de mayor a menor y tenemos:

$$\lambda_1 = 6 \geq \lambda_2 = 1 \geq \lambda_3 = 0 \geq 0$$

Ahora calculamos los vectores propios respectivos. Para ello, diagonalizamos la matriz para resolver el sistema homogéneo y obtener el vector propio asociado a cada valor propio λ_i.

- $\lambda_1 = 6$

  El vector v_1 asociado a λ_1 es: $(-\frac{5}{2}, 1, -\frac{1}{2})$, que normalizado resulta:

$$v_1 = (\frac{-5}{\sqrt{30}}, \frac{2}{\sqrt{30}}, \frac{-1}{\sqrt{30}})$$

- $\lambda_2 = 1$

  El vector v_2 asociado a λ_2 es: $(0, 1, -2)$, que normalizado resulta:

$$v_2 = (0, \frac{1}{\sqrt{5}}, \frac{-2}{\sqrt{5}})$$

- $\lambda_3 = 0$

  El vector v_3 asociado a λ_3 es: $(1, 2, -1)$, que normalizado resulta:

$$v_3 = (\frac{1}{\sqrt{6}}, \frac{2}{\sqrt{6}}, \frac{-1}{\sqrt{6}})$$

Luego, la descomposición de M a partir del teorema 3.1.1 nos queda:

$$M = A^T A = \begin{bmatrix} \frac{-5}{\sqrt{30}} & 0 & \frac{1}{\sqrt{6}} \\ \frac{2}{\sqrt{30}} & \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{6}} \\ \frac{-1}{\sqrt{30}} & \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{6}} \end{bmatrix} \begin{bmatrix} 6 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{-5}{\sqrt{30}} & \frac{2}{\sqrt{30}} & \frac{-1}{\sqrt{30}} \\ 0 & \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{-1}{\sqrt{6}} \end{bmatrix} = P \cdot D \cdot P^T$$

Y de este modo hemos obtenido V:

$$V = P = \begin{bmatrix} \frac{-5}{\sqrt{30}} & \frac{2}{\sqrt{30}} & \frac{-1}{\sqrt{30}} \\ 0 & \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{-1}{\sqrt{6}} \end{bmatrix}$$

Encontrar la matriz de valores singulares

Como hemos dicho son las raíces de los valores propios no nulos de A^TA. Y como A tiene una dimensión de 2×3, Σ tendrá la estructura de 3.10 y quedará así:

$$\Sigma = \begin{bmatrix} \sqrt{6} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Encontrar los vectores singulares izquierdos

Calculamos los vectores u_i a partir de la ecuación (3.8):

$$u_1 = \frac{1}{\sigma_1} A \cdot v_1 = \frac{1}{\sqrt{6}} \begin{bmatrix} 1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix} \begin{pmatrix} \frac{5}{\sqrt{30}} \\ -\frac{2}{\sqrt{30}} \\ \frac{1}{\sqrt{30}} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{5}} \\ -\frac{2}{\sqrt{5}} \end{pmatrix}$$

$$u_2 = \frac{1}{\sigma_2} A \cdot v_2 = \frac{1}{\sqrt{1}} \begin{bmatrix} 1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix} \begin{pmatrix} 0 \\ \frac{1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{pmatrix} = \begin{pmatrix} \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{pmatrix}$$

Así, hemos obtenido U:

$$U = [u_1, u_2] = \begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ \frac{-2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{bmatrix} = \frac{1}{\sqrt{5}} \begin{pmatrix} 1 & 2 \\ -2 & 1 \end{pmatrix}$$

La descomposición en valores singulares, SVD, de A finalmente nos resulta:

$$A = \begin{bmatrix} 1 & 0 & 1 \\ -2 & 1 & 0 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ \frac{-2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \sqrt{6} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{-5}{\sqrt{30}} & \frac{2}{\sqrt{30}} & \frac{-1}{\sqrt{30}} \\ 0 & \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{6}} & \frac{2}{\sqrt{6}} & \frac{-1}{\sqrt{6}} \end{bmatrix} = U \cdot \Sigma \cdot V^T$$

Si ahora, a esta matriz descompuesta, la comprimimos y nos quedamos solo con el valor singular más grande obtendremos:

$$A_1 = \begin{bmatrix} \frac{1}{\sqrt{5}} \\ \frac{-2}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \sqrt{6} \end{bmatrix} \begin{bmatrix} -\frac{5}{\sqrt{30}} & \frac{2}{\sqrt{30}} & -\frac{1}{\sqrt{30}} \end{bmatrix} = \begin{bmatrix} -1 & \frac{2}{5} & \frac{-1}{5} \\ 2 & \frac{-4}{5} & \frac{2}{5} \end{bmatrix} \tag{3.48}$$

Luego si calculamos el grado de compresión acorde con la fórmula 3.46:

$$1 - \frac{(2+3) \cdot 1}{2 \cdot 3} = \frac{1}{6}$$

Tenemos que esta aproximación que hemos hecho tiene un grado de compresión de $\frac{1}{6}$. Como podemos ver en 3.48, la matriz A_1 que hemos obtenido difiere bastante de la matriz original A esto es debido al bajo grado de compresión que hemos obtenido. Un grado de compresión tan bajo puede ser debido a o bien, que se trata de una matriz de dimensiones pequeñas entonces no tiene información redundante que eliminar o bien, que solo hemos cogido un valor singular.

3.6. Programación en Python

Para la parte más práctica del presente trabajo hemos realizado un programa en el lenguaje de programación Python. El código del programa que hacemos referencia en este capítulo, podemos verlo en el anexo A. Decidimos utilizar este lenguaje de programación, porque tiene un amplio inventario de librerías muy fáciles de usar. Entonces, para la realización del programa hemos tenido de instalar e importar varias librerías de Python. Una vez importada tuvimos que averiguar que funciones eran las necesarias para el funcionamiento que queríamos en nuestro programa. A continuación, vamos a nombrar y dar una pincelada sobre en que consisten las librerías y funciones que hemos utilizado:

Para la parte del tratamiento de imágenes, necesitamos la librería *cv2*. Las funciones de esta librería que hemos utilizado son:

- `cv2.imread(nombreYExtensionImagen, cv2.IMREAD_COLOR)`

  Esta función la hemos utilizado para leer la imagen y convertirla en la matriz de píxeles. El argumento de la función *cv2.IMREAD_COLOR* nos indica que vamos a exportarla a 3 matrices, una por cada color RGB como hemos comentado en la sección 3.2. Si quisiéramos exportar una imagen en blanco y negro ese argumento sería *cv2.IMREAD_GRAYSCALE*.

- `cv2.normalize(matrizQueRepresentaLaImagen, None, alpha=0, beta=1, norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_32F)`

  Esta función la hemos utilizado para normalizar los valores de la matriz que representa la imagen, ya que para mostrar la imagen por pantalla estos valores debían estar normalizados.

- `cv2.imshow(cadena, imagen)`

  Esta función la hemos utilizado para mostrar las imágenes por pantalla. En el primer argumento le pasamos una cadena de texto que se mostrará en el título de la ventana en la que se abrirá la imagen.

- `cv2.imwrite(cadena, imagen)`

  Esta función la hemos utilizado para guardará las imágenes. En el primer argumento le pasamos una cadena de texto que es el nombre con la que se guardará la imagen. Se guarda en el mismos directorio donde está el programa ejecutado.

- `cv2.waitKey(0)`

  Esta función para el programa hasta que pulsamos sobre cualquier tecla del teclado. La hemos utilizado para poder observar las imágenes todo el tiempo que queramos hasta pulsar una tecla.

- `cv2.destroyAllWindows()`

  Esta función cierra todas las ventanas abiertas que están mostrando imágenes. La utilizamos justo después de la anterior, para que así cuando pulsemos una tecla se cierren automáticamente estas ventanas.

Para la parte del tratamiento de matrices, necesitamos la librería *numpy*. Las funciones de esta librería que hemos utilizado son:

- `np.zeros((n,m))`

  Esta función la hemos utilizado para crear una matriz de zeros de dimensiones $m \times n$.

- `np.dot(matriz1, matriz2)`

  Esta función la hemos utilizado para multiplizar dos matrices.

Para la parte de calcular la descomposición en valores singulares, también existe una librería, esta es *linalg*. De esta librería solo hemos utilizado una función y esta es:

- `la.svd(matriz)`

  Esta función calcula el SVD de la matriz que le pasamos como argumento. Devuelve 2 matrices, U y V^T y un vector que son el resultado de la descomposición en valores singulares como vector.

Con estas librerías importadas, las funciones explicadas y otras funciones simples de Python hemos realizado el programa en lenguaje Python.

En el capítulo 4 veremos los resultados de una imagen a las que hemos aplicado la compresión de imágenes.

Capítulo 4

Resultados

En este capítulo vamos a mostrar un resultado que hemos hecho con una imagen en el programa creado. Hemos elegido una imagen apaisada aérea donde podemos observar la Ágora y distintos edificios de la Universitat Jaume I.

Primero realizamos el experimento con 1 valor singular, un cuarto, un medio, tres cuartos y todos los valores singulares de la matriz asociada a la imagen. Pero como podemos observar en las imágenes 4.1 y 4.6, observamos que solo con la mitad de valores singulares ya no se apreciaba diferencia entre la imagen original y la imagen comprimida. Por ello decidimos que sería más interesante visualizar la compresión de la imagen con menos valores que la mitad del total, así podríamos observar visualmente la transición de la compresión de imágenes. Calcularemos el grado de compresión de cada respectiva imagen comprimida. La imagen tiene una dimensión de 728×1080 píxeles, por lo que:

$$m = 728 \quad y \quad n = 1080$$

Con el programa que hemos hecho también podemos obtener los valores singulares y con ellos calcular el error relativo de la aproximación con la ecuación 3.47. Para ello necesitamos los siguientes valores singulares de las matrices, como tenemos 3 valores singulares, uno por cada matriz de cada color RGB, realizaremos una media aritmética entre los 3, podemos ver dichos valores en la tabla 4. Luego aplicamos la Definición 3.5.2 y el error relativo con 3.47:

- Fig 4.2: Compresión con un valor singular ($k = 1$):

$$1 - \frac{(1080 + 728) \cdot 1}{1080 \cdot 728} = 1 - \frac{1808}{786240} = 0,9977 \approx 99,77\,\%$$

$$\frac{\|A - A_1\|_2}{\|A\|_2} = \frac{\bar{\sigma_2}}{\bar{\sigma_1}} = \frac{16658,21}{116763,71} = 0,1427$$

k	$\sigma_k Azul$	$\sigma_k Rojo$	$\sigma_k Verde$	$\bar{\sigma_k}$
1	118702,06	120337,47	111251,6	116763,71
2	17097,84	15552,23	17324,57	16658,21
45	2591,89	2772,85	2839,20	2734,65
91	1465,35	1489,51	1511,86	1488,91
273	309,52	309,73	307,17	308,81
364	156,13	153,68	153,74	154,52
728 (todos)	6,043	4,898	5,13	5,357

Cuadro 4.1: Tabla de los valores singulares de la imagen comprimida

El grado de compresión es $0,9977$ y el error relativo $0,1427$.

- Fig 4.3: Compresión con un cuarto de los valores singulares ($k = 45$):

$$1 - \frac{(1080 + 728) \cdot 45}{1080 \cdot 728} = 1 - \frac{81360}{786240} = 0,8965 \approx 89,65\,\%$$

$$\frac{\|A - A_{45}\|_2}{\|A\|_2} = \frac{\bar{\sigma_{45}}}{\bar{\sigma_1}} = \frac{2734,65}{116763,71} = 0,0234$$

El grado de compresión es $0,8965$ y el error relativo $0,0234$.

- Fig 4.4: Compresión con la mitad de valores singulares ($k = 91$):

$$1 - \frac{(1080 + 728) \cdot 91}{1080 \cdot 728} = 1 - \frac{164528}{786240} = 0,7907 \approx 79,07\,\%$$

$$\frac{\|A - A_{91}\|_2}{\|A\|_2} = \frac{\bar{\sigma_{91}}}{\bar{\sigma_1}} = \frac{1488,91}{116763,71} = 0,01275$$

El grado de compresión es $0,7907$ y el error relativo $0,01275$.

- Fig 4.5: Compresión con tres cuartos de los valores singulares ($k = 273$):

$$1 - \frac{(1080 + 728) \cdot 273}{1080 \cdot 728} = 1 - \frac{493584}{786240} = 0,372 \approx 37,2\,\%$$

$$\frac{\|A - A_{273}\|_2}{\|A\|_2} = \frac{\bar{\sigma_{273}}}{\bar{\sigma_1}} = \frac{308,81}{116763,71} = 2,644e-3$$

El grado de compresión es $0,372$ y el error relativo $2,644e-3$.

Nº de valores singulares cogidos	Grado de compresión	Error relativo
1	99,77 %	0,1427
45	89,65 %	0,0234
91	79,07 %	0,01275
273	37,20 %	2,644e3
364	16,29 %	1,323e17

Cuadro 4.2: Tabla de grados de compresión y errores relativos de las compresiones con distinto número de valores singulares cogidos

Figura 4.1: Imagen original de la Universitat Jaume I

- Fig 4.6: Compresión con todos los valores singulares ($k = 364$):

$$1 - \frac{(1080 + 728) \cdot 364}{1080 \cdot 728} = 1 - \frac{658112}{786240} = 0,1629 \approx 16,29\,\%$$

$$\frac{\|A - A_{364}\|_2}{\|A\|_2} = \frac{\bar{\sigma_{364}}}{\bar{\sigma_1}} = \frac{154,52}{116763,71} = 1,323e-3$$

  El grado de compresión es $0,1629$ y el error relativo $1,323e-17$.

A simple vista lo podemos ver y los errores relativos nos lo confirman que con la mitad de valores singulares de la matriz, la diferencia entre la imagen real y comprimida no es perceptible para el ojo humano. Eso quiere, decir, que con mucha menos memoria podríamos almacenar la misma imagen a vista del ojo humano.

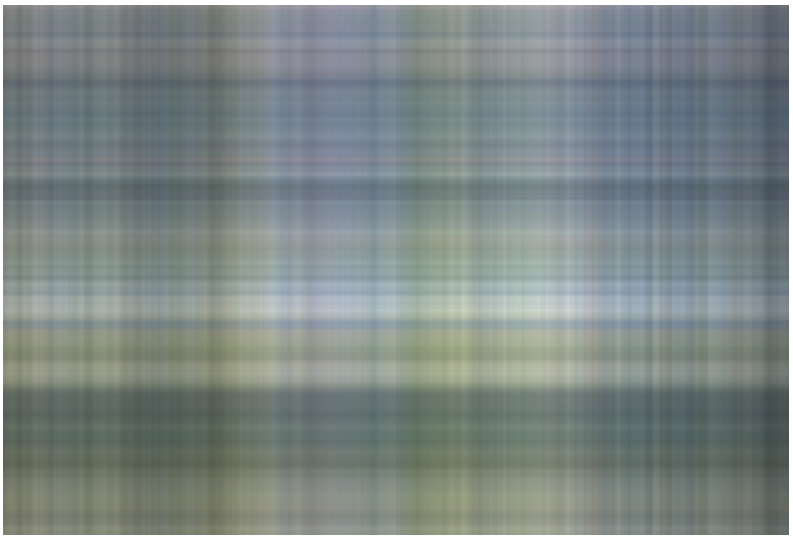

Figura 4.2: Imagen comprimida con 1 valor singular

Figura 4.3: Imagen comprimida con 45 (1/16 de todos los) valores singulares

Figura 4.4: Imagen comprimida con 91 (1/8 de todos los) valores singulares

Figura 4.5: Imagen comprimida con 273 (3/8 de todos los) valores singulares

Figura 4.6: Imagen comprimida con 364 (la mitad de los) valores singulares

Figura 4.7: Imagen comprimida con 728 (todos los) valores singulares

Capítulo 5

Conclusiones

En este último capítulo vamos a centrarnos en la descripción de las consideraciones más personales del proyecto final de grado, puesto que dentro de cada capítulo ya se han desarrollado las conclusiones más formales y académicas de cada parte.

El presente trabajo final de grado me ha resultado muy ameno a pesar que estaba profundizando sobre un tema que casi desconocía. Muchas veces había oído que detrás de todo están las matemáticas y que una imagen puede considerarse como una matriz de píxeles pero nunca había pensado que sería tan interesante investigarlo.

Sobretodo, lo que más me ha gustado de realizar el presente trabajo ha sido que pesé a que al principio tuve que averiguar y entender varios conceptos teóricos, me ha parecido bastante práctico. Por lo que hace a la parte de la programación, ha sido muy ocurrente realizar un programa, que funcione como queremos y tenga una utilidad, igual que así aplicar conceptos de las asignatura de informática que damos en la carrera.

Desde el primer momento, me llamó mucho la atención poder juntar dos cosas que siempre me han apasionado: la fotografía y las matemáticas. Ahora una vez acabado, puedo decir que ha sido igual o más interesante de lo que creía.

Bibliografía

[1] Nora La Serna, Luzmila Pro Concepción, and Carlos Yañez Durán. Compresión de imágenes: Fundamentos, técnicas y formatos. *Revista de investigación de Sistemas e Informática*, 6(1):21–29, 2009.

[2] Chi-Kwong Li and Gilbert Strang†. An elementary proof of mirsky's low rank approximation theorem. https://cklixx.people.wm.edu/uinorm-note.pdf.

[3] Cheng Soon Ong Marc Peter Deisenroth, A. Aldo Faisal. Image compression using singular value decomposition (svd).

[4] Cheng Soon Ong Marc Peter Deisenroth, A. Aldo Faisal. Mathematics for machine learning.

[5] Naveed Altaf A. Types of image compression. `https://www.gumlet.com/learn/types-of-image-compression/`.

[6] Richard Wilkinson. Svd optimization results. https://rich-d-wilkinson.github.io/MATH3030/3-4-svdopt.htmleq:eigenopt.

Anexo A

Anexo I

```
import cv2
import numpy as np
from numpy import linalg as la

# FUNCI N QUE APROXIMA CON valoresCoger VALORES SINGULARES LA IMAGEN
def aproximarImagenConIvalores(valoresCoger, matU, matSigma, matVT, n, m):
    U2 = np.zeros((n, valoresCoger))
    Sigma2 = np.zeros((valoresCoger, valoresCoger))
    VT2 = np.zeros((valoresCoger, m))
    for i in range(valoresCoger):
        for j in range(n):
            U2[j,i] = matU[j,i]
        Sigma2[i,i] = matSigma[i,i]
        for k in range(m):
            VT2[i, k] = matVT[i, k]

    USigma2 = np.dot(U2, Sigma2)
    aproximacion = np.dot(USigma2, VT2)
    return aproximacion


# FUNCION AUXILIAR QUE HACE LA LLAMADA A LA APROXIMACI N CON LOS N VALORES QUE
                                        LE PASAMOS
def aproxima(valores, n, m):
    Aaprox_azul = aproximarImagenConIvalores(valores, U_azul, Sigma_azul,
                                             VT_azul, n, m)
    Aaprox_rojo = aproximarImagenConIvalores(valores, U_rojo, Sigma_rojo,
                                             VT_rojo, n, m)
    Aaprox_verde = aproximarImagenConIvalores(valores, U_verde, Sigma_verde,
                                             VT_verde, n, m)
    aproximacion = np.zeros((n, m, 3))
    for i in range(n):
        for j in range(m):
```

```
            aproximacion[i][j][0] = Aaprox_azul[i][j]
            aproximacion[i][j][1] = Aaprox_rojo[i][j]
            aproximacion[i][j][2] = Aaprox_verde[i][j]
    Aaprox_norm = cv2.normalize(aproximacion, None, alpha=0, beta=1, norm_type=
                                        cv2.NORM_MINMAX, dtype=cv2.CV_32F)
   # print("Matriz de la imagen comprimida con {} valores singulares:".format(
                                        valores))
  #  print(aproximacion)
    cv2.imshow("Imagen comprimida con {} valores singulares:".format(valores),
                                        Aaprox_norm)
    cv2.imwrite('{}-{}-valores-singulares.png'.format(nombre_imagen, valores),
                                        aproximacion)


# LEER IMAGEN
nombre_imagen = 'UJI'
extension = 'jpg'
A = cv2.imread('{}.{}'.format(nombre_imagen, extension), cv2.IMREAD_COLOR)



# DIMENSIONES
n = A.shape[0]
m = A.shape[1]

# CREAMOS Y DIVIDIMOS LAS MATRICES EN UNA PARA CADA COLOR RGB
A_azul = np.zeros((n,m))
A_rojo = np.zeros((n,m))
A_verde = np.zeros((n,m))
for i in range(n):
    for j in range(m):
        A_azul[i][j] = A[i][j][0]
        A_rojo[i][j] = A[i][j][1]
        A_verde[i][j] = A[i][j][2]

 # CALCULAR SVD PARA CADA COLOR RGB
U_azul, sigma_azul, VT_azul = la.svd(A_azul)
U_rojo, sigma_rojo, VT_rojo = la.svd(A_rojo)
U_verde, sigma_verde, VT_verde = la.svd(A_verde)

 # CREAMOS LA MATRIZ SIGMA CON LOS VALORES SINGULARES DEL SVD
nValores = min(n, m)
Sigma_azul = np.zeros((n, m))
Sigma_rojo = np.zeros((n, m))
Sigma_verde = np.zeros((n, m))
for i in range(nValores):
    Sigma_azul[i, i] = sigma_azul[i]
    Sigma_rojo[i, i] = sigma_rojo[i]
    Sigma_verde[i, i] = sigma_verde[i]

 # CALCULAMOS LA IMAGEN A PARTIR DE LA DESCOMPOSICI N
USigma_azul = np.dot(U_azul, Sigma_azul)
```

```
Aaprox_azul = np.dot(USigma_azul, VT_azul)

USigma_rojo = np.dot(U_rojo, Sigma_rojo)
Aaprox_rojo = np.dot(USigma_rojo, VT_rojo)

USigma_verde = np.dot(U_verde, Sigma_verde)
Aaprox_verde = np.dot(USigma_verde, VT_verde)

#print("Matriz de la imagen:")
#print(A)
print('Dimensiones de la imagen :', A.shape)

# VOLVEMOS A UNIR LAS MATRICES DE LOS COLORES PARA MOSTRARLA POR PANTALLA
original = np.zeros((n, m, 3))
for i in range(n):
    for j in range(m):
        original[i][j][0] = A_azul[i][j]
        original[i][j][1] = A_rojo[i][j]
        original[i][j][2] = A_verde[i][j]

 # GUARDAMOS IMAGEN
cv2.imwrite('{}-orig.png'.format(nombre_imagen), original)


 # NORMALIZAMOS PORQUE PARA MOSTRAR LA IMAGEN POR PANTALLA LOS P XELES DEBEN
                                        SER DE 0 A 1
orig_norm = cv2.normalize(original, None, alpha=0, beta=1, norm_type=cv2.
                                       NORM_MINMAX, dtype=cv2.CV_32F)
print("Matriz de la imagen calculada por SVD")
print(orig_norm)
cv2.imshow("Imagen calculada por SVD (sin comprimir)", orig_norm )

# APROXIMACI N CON SOLO UN VALOR SINGULAR
print("Sigma_1 azul: ", sigma_azul[0])
print("Sigma_1 rojo: ", sigma_rojo[0])
print("Sigma_1 verde: ", sigma_verde[0])
aproxima(1, n, m)
print("Sigma_2 azul: ", sigma_azul[1])
print("Sigma_2 rojo: ", sigma_rojo[1])
print("Sigma_2 verde: ", sigma_verde[1])

# CON UN DIECISEISAVO DE LOS VALORES SINGULARES
aproxima( nValores // 16, n, m)
print("Sigma_1_2 azul: ", sigma_azul[nValores // 16])
print("Sigma_1_2 rojo: ", sigma_rojo[nValores // 16])
print("Sigma_1_2 verde: ", sigma_verde[nValores // 16])

# CON UN OCTAVO DE LOS VALORES SINGULARES
aproxima( nValores // 8, n, m)
print("Sigma_1_2 azul: ", sigma_azul[nValores // 8])
print("Sigma_1_2 rojo: ", sigma_rojo[nValores // 8])
print("Sigma_1_2 verde: ", sigma_verde[nValores // 8])
```

```
    # CON TRES OCTAVOS DE LOS VALORES SINGULARES
aproxima( 3 * nValores // 8, n, m)
print("Sigma_1_2 azul: ", sigma_azul[3 * nValores // 8])
print("Sigma_1_2 rojo: ", sigma_rojo[3 * nValores // 8])
print("Sigma_1_2 verde: ", sigma_verde[3 * nValores // 8])

# CON LA MITAD DE LOS VALORES SINGULARES
aproxima(nValores // 2, n, m)
print("Sigma_1_2 azul: ", sigma_azul[nValores // 2])
print("Sigma_1_2 rojo: ", sigma_rojo[nValores // 2])
print("Sigma_1_2 verde: ", sigma_verde[nValores // 2])

# CON TODOS LOS VALORES SINGULARES
aproxima(nValores, n, m)
print("Sigma_nValores azul: ", sigma_azul[nValores - 1])
print("Sigma_nValores rojo: ", sigma_rojo[nValores - 1])
print("Sigma_nValores verde: ", sigma_verde[nValores - 1])

#CERRAMOS LAS VENTANAS DONDE SE MUESTRAN LAS IM GENES Y SUS APROXIMACIONES
cv2.waitKey(0)
cv2.destroyAllWindows()
```