DESCOMPOSICIÓN EN VALORES SINGULARES DE UNA MATRIZ: UN REPASO POR LOS FUNDAMENTOS TEÓRICOS Y SUS APLICACIONES EN EL PROCESAMIENTO DE IMÁGENES

Juan José Fallas-Monge,* Jeffry Chavarría-Molina, Pablo Soto-Quiros
Instituto Tecnológico de Costa Rica, Costa Rica.

ABSTRACT

This review paper systematizes a detailed, didactical and own construction of the singular value decomposition (SVD). The theory is complemented with relevant applications to image processing. Also, the algorithms of those applications and numerical examples are shown. The way how the document is organized allows it to be useful for students and researchers with interest in the SVD and its applications to image processing.

KEYWORDS: SVD, singular values, image processing, Linear Algebra.

MSC: 15-01, 15A18, 15A23

RESUMEN

El presente artículo, de tipo *review*, sistematiza una construcción propia, detallada y didáctica de la descomposición en valores singulares de una matriz (SVD, por su siglas en inglés). A su vez, complementa el desarrollo teórico con aplicaciones relevantes de la SVD en el procesamiento de imágenes, junto con los algoritmos respectivos y ejemplos numéricos. La forma clara en que se estructura el documento le permite ser útil para aquellos estudiantes e investigadores dedicados al estudio de la SVD y sus aplicaciones en procesamiento de imágenes.

PALABRAS CLAVE: SVD, valores singulares, procesamiento de imágenes, Álgebra Lineal.

1. INTRODUCCIÓN

La descomposición en valores singulares de una matriz (SVD, por sus siglas en inglés) es un tipo de factorización que generaliza para cualquier matriz rectangular el concepto de valores propios, mediante una extensión de la descomposición polar ([16, 13]). Esta estrategia es similar a la diagonalización de matrices, la descomposición QR o la descomposición LU, sin embargo, tiene la ventaja que existe para

*jfallas@itcr.ac.cr, jchavarria@tec.ac.cr, jusoto@tec.ac.cr

cualquier matriz, sin imponer limitaciones sobre su dimensión. Como se mostrará posteriormente, la SVD es ampliamente utilizada en diversas aplicaciones, algunas de las cuales se abordarán con detalle en este documento. Parte de su fortaleza recae en que los valores singulares permiten organizar, en cierto sentido, la información almacenada en una matriz. Los valores singulares de mayor magnitud se asocian a subespacios propios que condensan la información más significativa de los datos. Esto permite reducir la dimensionalidad de problemas concretos como la compresión de imágenes, en donde es posible controlar la calidad de la compresión, mediante la magnitud de dichos valores.

El inicio del estudio de la SVD fue en 1873 mediante los trabajos desarrollados por Eugenio Beltrami ([4]) y Camille Jordan ([19]). Sobre esto, el matemático G. W. Stewart resalta que Beltrami y Jordan son los progenitores de la SVD, ya que Beltrami fue el primero en llevar a cabo la publicación de dicha descomposición y Jordan por la completitud y la elegancia de la representación ([25]). Curiosamente, el origen de la SVD precedió al uso de las matrices, ya que los resultados se desarrollaron en términos de determinantes, formas bilineales y formas cuadráticas ([25, 22]).

La SVD tiene diversas aplicaciones en Matemática tales como calcular la inversa de Moore-Penrose, determinar el rango de una matriz, calcular el espacio nulo y rango de una transformación lineal ([15, 16, 13]) e, inclusive, en Estadística ha sido utilizada como una generalización del método de Análisis de Componentes Principales ([18]). Sin embargo, en los últimos años este concepto también se ha extendido al campo de la ingeniería. La búsqueda y ordenamiento de documentos relevantes dentro de una determinada colección (como lo hacen los motores de búsqueda en Internet), así como en el desarrollo de sistemas de control de múltiple entrada y múltiple salida, con el fin de mejorar las ondas de transmisión y recepción en antenas para dispositivos inalámbricos, son ejemplos de ello ([27, 28]). En el procesamiento de imágenes también ha ido en aumento y esto es relevante porque afecta positivamente áreas como la medicina, telecomunicaciones, control de procesos industriales y al entretenimiento ([23]). Algunos ejemplos concretos son la compresión de imágenes digitales a través del rango matricial ([22, 8]), el modelado de fondo de videos ([30]), la eliminación de ruido de una imagen ([1, 30]) y el reconocimiento facial ([29]).

El presente artículo, de tipo *review*, muestra una revisión meticulosa y autónoma de la descomposición en valores singulares de una matriz, con particular énfasis en algunas aplicaciones en el procesamiento de imágenes. La construcción teórica realizada es propia de los autores, no en el sentido de la originalidad de los resultados y teoremas que se usan (todos ellos son ampliamente conocidos en Álgebra Lineal), si no en la forma didáctica en que se organizan y detallan en el documento para que cualquier lector con conocimientos básicos de Álgebra Lineal pueda asimilar la deducción, sin necesidad de estar consultando recurrentemente otros artículos o libros. Dichas explicaciones se complementan con ejemplos numéricos que ayudan al lector para una mejor comprensión de los resultados. Muchas otras publicaciones relevantes tratan la SVD (ver [25] y [22], por ejemplo), sin embargo, se enfocan en el teorema en sí y su demostración, dejando de lado los conceptos y resultados periféricos necesarios para el proceso. Por ello, el artículo da una visión teórica más completa y global de la SVD, que potencia al lector a entender cómo esta descomposición funciona en aplicaciones concretas.

Por otra parte, el ejemplo clásico en la literatura sobre la SVD en el área del procesamiento de imágenes

consiste en la compresión de imágenes, por lo que el presente artículo muestra detalladamente otras aplicaciones actuales y que no son frecuentemente referenciadas al mostrar ejemplos aplicados de esta descomposición. Por ello, se mostrará cómo la SVD puede ser utilizada en el modelado de fondo de imágenes para la detección de movimiento en videos, en la eliminación de ruido de una imagen y en el reconocimiento facial.

Cabe aclarar que los autores del presente documento no elaboraron ni propusieron los algoritmos y aplicaciones mencionados en la Sección 4. Los autores solo se dedicaron a realizar la reproducción de ellos, y realizar una presentación y organización original de estos métodos.

Finalmente, el artículo está organizado de la siguiente manera. En la sección 2 se presentan los conceptos previos necesarios para formalizar la SVD. En la sección 3 se formaliza y ejemplifica la descomposición. En la sección 4 se sintetiza un conjunto de aplicaciones de la SVD en al área del procesamiento de imágenes. Finalmente, en la sección 5 se presentan las conclusiones.

2. PRELIMINARES

En esta sección se resumen los conceptos y resultados necesarios para formalizar la SVD. Algunos de ellos se justificarán en el texto, otros son bastante conocidos y pueden ser fácilmente encontrados en la literatura.

Dados $m, n \in \mathbb{N}$, se denotará con $\mathrm{Mat}(\mathbb{C}, m, n)$ el conjunto de las matrices de entradas complejas de m filas y n columnas. Por su parte, $\mathrm{Mat}(\mathbb{C}, n)$ denota el conjunto de las matrices cuadradas de orden n de entradas complejas. Se escribirá $A_{m\times n}$ o A_n, si $A \in \mathrm{Mat}(\mathbb{C}, m, n)$ o $A \in \mathrm{Mat}(\mathbb{C}, n)$, respectivamente. A la matriz $\Sigma \in \mathrm{Mat}(\mathbb{C}, m, n)$ tal que $\sigma_{ij} = 0$, cuando $i \neq j$, se le denomina *matriz diagonal generalizada*. En el caso que Σ sea cuadrada, simplemente se le llama *matriz diagonal*.

Si $U \in \mathrm{Mat}(\mathbb{C}, m, n)$, la *transpuesta conjugada* de U se denotará U^*, y satisface que $u^*_{ij} = \overline{u_{ji}}$, $\forall i, j$. Si U solo tiene entradas reales, entonces $U^* = U^{\mathbf{T}}$. En caso que $U \in \mathrm{Mat}(\mathbb{C}, n)$, se le llama *hermitiana* o *autoadjunta* si $U = U^*$. Además, U es *unitaria* si $U^* \cdot U = I_n = U \cdot U^*$, esto es, si U^{-1} existe y $U^{-1} = U^*$. Es claro que las matrices cuadradas AA^* y A^*A son hermitianas, para toda $A \in \mathrm{Mat}(\mathbb{C}, m, n)$.

Dada $A \in \mathrm{Mat}(\mathbb{C}, m, n)$, una *representación de A por bloques* con p bloques-fila y q bloques-columna tiene la forma:

$$A = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1q} \\ \vdots & & & \vdots \\ A_{p1} & A_{p2} & \dots & A_{pq} \end{pmatrix},$$

tal que para i fijo $(1 \leq i \leq p)$ todas las submatrices A_{ij} $(1 \leq j \leq q)$ tienen que tener la misma cantidad de filas (no así con el número de columnas). Similarmente, para j fijo todas las submatrices A_{ij} tienen que tener la misma cantidad de columnas (no así con el número de filas). Evidentemente, la representación por bloques de una matriz no es única. Si las matrices $A \in \mathrm{Mat}(\mathbb{C}, m, n)$ y $B \in \mathrm{Mat}(\mathbb{C}, n, s)$ se organizan por bloques de manera que A tiene p bloques-fila y q bloques-columna,

y B tiene q bloques-fila y t bloques-columna, entonces el producto AB está dado por $(AB)_{ij} = \sum_{k=1}^{q} A_{ik}B_{kj},\ 1 \leq i \leq p,\ 1 \leq j \leq t$, siempre que el producto de bloques $A_{ik}B_{kj}$ esté bien definido para todo k.

Si $A \in \mathrm{Mat}(\mathbb{C}, m, n)$, el *rango columna* de A es el número máximo de columnas que son linealmente independientes. Similarmente, el *rango fila* de A es el número máximo de filas que son linealmente independientes. En [20] se demuestra que dichos valores son iguales y se le denomina el *rango de A*. Es común utilizar las notaciones $\mathrm{r}(A)$ y $\mathrm{rg}(A)$ para denotar a este valor. Es claro que $0 \leq \mathrm{r}(A) \leq \mathrm{mín}\{m, n\}$, y en caso que $\mathrm{r}(A) = \mathrm{mín}\{m, n\}$, entonces se dice que A es de *rango completo*. Algunas de las propiedades bastante conocidas sobre el rango de una matriz son:

- Si A es una matriz cuadrada, A es invertible si y solo si es de rango completo.
- Si $B \in \mathrm{Mat}(\mathbb{C}, m)$ es una matriz no singular, $\mathrm{r}(BA) = \mathrm{r}(A)$, para todo $A \in \mathrm{Mat}(\mathbb{C}, m, n)$.
- Si $\Sigma \in \mathrm{Mat}(\mathbb{C}, m, n)$ es una matriz diagonal generalizada, entonces su rango corresponde al número de entradas no nulas.

Para $A \in \mathrm{Mat}(\mathbb{C}, n)$, a un vector $v \in \mathbb{C}^n$, $v \neq \mathbf{0}$, se le llama *vector propio* de A si existe un escalar λ (llamado *valor propio*) tal que $Av = \lambda v$ o, equivalentemente, $(\lambda I_n - A)\, v = \mathbf{0}$. Esta definición implica analizar la ecuación matricial $(\lambda I_n - A)\, X = \mathbf{0}$, que tendrá soluciones no triviales si y solo si $\mathrm{r}(\lambda I_n - A) < n$. Por ende, λ es valor propio de A si y solo si $|\lambda I_n - A| = 0$. A la expresión $P(\lambda) = |\lambda I_n - A|$ se le denomina *polinomio característico* de A y a $|\lambda I_n - A| = 0$ la *ecuación característica*. Los valores propios de A son las raíces de su polinomio característico.

En caso que exista una matriz Q no singular tal que $A = Q\Sigma Q^{-1}$ (o, equivalentemente, $Q^{-1}AQ = \Sigma$), donde Σ es una matriz diagonal, entonces se dice que A es *diagonalizable*. Esto es, A es diagonalizable si es semejante a una matriz diagonal. En general, una condición necesaria y suficiente para que una matriz $A \in \mathrm{Mat}(\mathbb{C}, n)$ sea diagonalizable es que su polinomio característico tenga solo raíces reales y tal que para cada valor propio λ, de multiplicidad k, $k \geq 2$, deben existir k vectores propios linealmente independientes. Si esto sucede, entonces las columnas de Q están formadas por tales vectores propios y Σ contiene en su diagonal a los valores propios de A, ordenados de manera respectiva como los vectores propios correspondientes fueron colocados como columnas de Q. La diagonalización de una matriz cuadrada, en caso que sea posible, da una factorización de ella y esto es útil, por ejemplo, para la rotación de secciones cónicas y para el cálculo de A^n, $n \in \mathbb{N}$. No obstante, la diagonalización de matrices tiene tres desventajas significativas:

- La matriz tiene que ser cuadrada. En muchas aplicaciones la matriz de datos no es cuadrada, y por ende la diagonalización deja de ser una herramienta útil.
- Los vectores propios de A usualmente no son ortogonales. Para el caso que A sea simétrica y de entradas reales siempre es posible diagonalizarla ortogonalmente (u ortonormalmente, en caso que se necesiten vectores propios unitarios), sin embargo, para cada valor propio de A de multiplicidad k, $k \geq 2$, es necesario seleccionar k vectores propios ortogonales del subespacio

propio respectivo. Esta selección no siempre es inmediata, y puede que se tengan que realizar cálculo adicionales, como el proceso de ortogonalización de Gram-Schmidt.

- Usualmente no hay "suficientes" vectores propios. Esto pasa en caso que el subespacio propio asociado a un valor propio de multiplicidad k, $k \geq 2$, tenga un dimensión menor que k. Por ende, no es posible construir todas las columnas de la matriz Q que define la diagonalización, con la característica de ser linealmente independientes.

Como se verá luego, la descomposición en valores singulares de una matriz es una alternativa más eficiente a la diagonalización de matrices, dado que evita las tres limitaciones citados arriba.

Algunos de los teoremas que permiten justificar ciertas características de la SVD se demuestran utilizando las propiedades de un producto interno (conocido también como *producto escalar*). Por ello es necesario recordar que si V es un espacio vectorial sobre $\mathbb{C}$, un *producto interno* en V es una función $\langle \cdot, \cdot \rangle : V \times V \to F$ que satisface para todo $u, v, w \in V$ y para todo $\alpha \in \mathbb{C}$ lo siguiente: $\langle u, v \rangle = \overline{\langle v, u \rangle}$, $\langle \alpha u, v \rangle = \alpha \langle u, v \rangle$, $\langle u + v, w \rangle = \langle u, w \rangle + \langle v, w \rangle$, $\langle u, u \rangle \geq 0$ y $\langle u, u \rangle = 0 \iff u = \mathbf{0}$. De la definición se deduce fácilmente que: $\langle x, y + z \rangle = \langle x, y \rangle + \langle x, z \rangle$ y $\langle x, \alpha y \rangle = \overline{\alpha} \langle x, y \rangle$. La forma general de un producto interno en $\mathbb{C}^n$ se conoce como la forma hermitiana y es tal que $\langle u, v \rangle = v^* M u = \overline{u^* M v}$, para todo $u, v \in \mathbb{C}^n$, donde M es cualquier matriz hermitiana definida positiva[1] y v^* es el conjugado transpuesto de v. Es común seleccionar $M = I_n$ y, por ende, usualmente se esribe: $\langle u, v \rangle = v^* u$.

Si $S = \{u_1, \ldots, u_n\}$ es un *conjunto ortogonal* de vectores no nulos de un espacio vectorial V sobre $\mathbb{C}$ de dimensión n, entonces para $i \in \{1, \ldots, n\}$ se tiene que:

$$\mathbf{0} = \sum_{k=1}^{n} c_k \cdot u_k \Rightarrow u_i^* \cdot \mathbf{0} = c_i \cdot (u_i^* \cdot u_i) \Rightarrow 0 = c_i \cdot \langle u_i, u_i \rangle \Rightarrow c_i = 0$$

Por lo tanto, todo subconjunto ortogonal finito de un espacio vectorial de dimensión finita es linealmente independiente. De hecho, como la cantidad de vectores no nulos linealmente independientes en S es n, que coincide con la dimensión de V, entonces S es una base de dicho espacio vectorial, denominada *base ortogonal* de V y si, además, los vectores de S son unitarios, entonces S es una *base ortonormal* de V.

Se concluye esta sección con la prueba de cuatro resultados fundamentales para establecer la SVD de una matriz. En resumen:

- Todo valor propio de una matriz hermitiana es real.
- Si $A \in \text{Mat}(\mathbb{C}, m, n)$, los valores propios de las matrices AA^* y A^*A son reales no negativos.
- Los vectores propios asociados a valores propios distintos de una matriz hermitiana generan subespacios propios ortogonales.
- Las columnas de una matriz unitaria de orden n forman una base ortonormal de $\mathbb{C}^n$.

[1] La matriz M hermitiana se dice definida positiva si $u^* M u > 0$, para todo $u \in \mathbb{C}^n$ no nulo. Otra forma equivalente de definirla es que todos los autovalores (valores propios) de M sean positivos.

Teorema 1 *(Valor propio de una matriz hermitiana)*
Sea $U \in \mathrm{Mat}(\mathbb{C}, n)$ hermitiana. Si λ es un valor propio de U, entonces $\lambda \in \mathbb{R}$.

Prueba: Sea $v \in \mathbb{C}^n$ un vector propio de U ($v \neq \mathbf{0}$, por definición) asociado a λ. Note que:

$$\begin{aligned} \overline{\lambda}\ \langle v, v\rangle &= \langle v, \lambda v\rangle\ =\ \langle v, Uv\rangle\ = (Uv)^* v = v^* U^* v \\ &= v^* U v =\ \langle Uv, v\rangle\ =\ \langle \lambda v, v\rangle\ = \lambda\ \langle v, v\rangle \end{aligned}$$

Así, $(\overline{\lambda} - \lambda) \cdot \langle v, v\rangle\ = 0$. Como $v \neq \mathbf{0}$, entonces $\overline{\lambda} = \lambda$ y de ahí se tiene el resultado.

♣

Teorema 2
Sea $A \in \mathrm{Mat}(\mathbb{C}, m, n)$. Los valores propios de A^*A y de AA^* son no negativos.

Prueba: Sea λ valor propio de A^*A (el otro caso es similar) y v un vector propio asociado a λ. Como A^*A es hermitiana, en virtud del teorema 1 se sabe que $\lambda \in \mathbb{R}$. Luego,

$$\begin{aligned} \lambda \cdot \langle v, v\rangle &= \langle v, \lambda v\rangle\ =\ \langle v, A^*Av\rangle\ = (A^*Av)^* v = v^* A^* A v \\ &= (Av)^* Av =\ \langle Av, Av\rangle\ \geq 0 \end{aligned}$$

Así, $\lambda \cdot \langle v, v\rangle\ \geq 0$, de donde $\lambda \geq 0$.

♣

Teorema 3
Sea $U \in \mathrm{Mat}(\mathbb{C}, n)$ hermitiana. Si λ_1 y λ_2 son autovalores de U, tales que $\lambda_1 \neq \lambda_2$, y v_1 y v_2 autovectores asociados a λ_1 y λ_2, respectivamente, entonces $v_1 \perp v_2$.

Prueba: En efecto:

$$\begin{aligned} \lambda_1\ \langle v_1, v_2\rangle &= \langle \lambda_1 v_1, v_2\rangle\ =\ \langle Uv_1, v_2\rangle\ = v_2^* U v_1 = v_2^* U^* v_1 \\ &= (Uv_2)^* v_1 =\ \langle v_1, Uv_2\rangle\ =\ \langle v_1, \lambda_2 v_2\rangle\ = \overline{\lambda_2}\ \langle v_1, v_2\rangle \\ &= \lambda_2\ \langle v_1, v_2\rangle \end{aligned}$$

Así, $(\lambda_1 - \lambda_2) \cdot \langle v_1, v_2\rangle\ = 0$, de donde $v_1 \perp v_2$.

♣

Teorema 4
Sea $U \in \mathrm{Mat}(\mathbb{C}, n)$. U es unitaria si y solo si sus vectores columnas forman un conjunto ortonormal con el producto escalar complejo usual.

Prueba: Denote $U = (u_1\ u_2\ \ldots\ u_n)$, donde cada u_i, $1 \leq i \leq n$, corresponde a la i-ésima columna de C. Note que:

$$U^*U = \begin{pmatrix} u_1^* \\ u_2^* \\ \vdots \\ u_n^* \end{pmatrix} (u_1\ u_2\ \ldots\ u_n) = \begin{pmatrix} u_1^*u_1 & u_1^*u_2 & \ldots & u_1^*u_n \\ u_2^*u_1 & u_2^*u_2 & \ldots & u_2^*u_n \\ \vdots & & & \vdots \\ u_n^*u_1 & u_n^*u_2 & \ldots & u_n^*u_n \end{pmatrix}$$

$$= \begin{pmatrix} \langle u_1, u_1\rangle & \langle u_2, u_1\rangle & \ldots & \langle u_n, u_1\rangle \\ \langle u_1, u_2\rangle & \langle u_2, u_2\rangle & \ldots & \langle u_n, u_2\rangle \\ \vdots & & & \vdots \\ \langle u_1, u_n\rangle & \langle u_2, u_n\rangle & \ldots & \langle u_n, u_n\rangle \end{pmatrix}$$

De lo anterior se concluye que: U unitaria $\iff$ $U^*U = I_n$ $\iff$ $\big[\ \langle u_i, u_j\rangle = 0$ para $i \neq j$ y $\langle u_i, u_i\rangle = \|u_i\|^2 = 1$ para $i \in \{1, \ldots, n\}\big]$. Esto finaliza la prueba. ♣

En virtud del teorema 4, las columnas de una matriz unitaria $U \in \mathrm{Mat}(\mathbb{C}, n)$ forman una base ortonormal de $\mathbb{C}^n$ con respecto al producto escalar usual y, además, se deduce que toda matriz unitaria es de rango completo.

3. LA DESCOMPOSICIÓN EN VALORES SINGULARES

En esta sección se procederá a formalizar y ejemplificar la SVD. El teorema 5 establece la existencia de la descomposición para una matriz cualquiera. Para la prueba, suponga sin pérdida de generalidad que $n \leq m$ (adaptaciones mínimas permiten establecer la prueba para el caso que $n \geq m$).

Teorema 5 *(Existencia de la SVD)*
Sean $m, n \in \mathbb{N}$ y $A \in \mathrm{Mat}(\mathbb{C}, m, n)$ de rango r. Existen matrices unitarias $U_{m\times m}$ y $V_{n\times n}$ tales que $A = U\Sigma V^*$, donde $\Sigma_{m\times n}$ es una matriz diagonal generalizada de entradas reales no negativas. Dicha factorización se le llama descomposición en valores singulares y si A solo tiene entradas reales, entonces $A = U\Sigma V^{\mathbf{T}}$.

Prueba: Primero se tiene que $r = \mathrm{r}(A) = \mathrm{r}(A^*A)$ (ver [6] para la prueba de este hecho). Luego, en virtud del teorema 2 los valores propios de A^*A son no negativos. Denote dichos valores propios $\sigma_1^2 \geq \sigma_2^2 \geq \ldots \geq \sigma_r^2 > \sigma_{r+1}^2 = \sigma_{r+1}^2 = \ldots = \sigma_n^2 = 0$ y los respectivos vectores propios $v_1, \ldots, v_n$ *normalizados*. Así,

$$A^*Av_i = \sigma_i^2 v_i, \quad \text{para } i = 1, \ldots, n \tag{3.1}$$

Del teorema (3) se concluye que $\{v_1, \ldots, v_n\}$ es un conjunto ortogonal (en realidad, ortonormal, por la normalidad pedida sobre los v_i). Considere las matrices $V_r = (v_1\ v_2\ \ldots\ v_r)$, $V_{n-r} = (v_{r+1}\ v_{r+2}\ \ldots\ v_n)$ y $\Sigma_r = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$. De (3.1) note que:

$$A^*AV_r = V_r\Sigma_r^2 \tag{3.2}$$

Además, como $\{v_1, \ldots, v_n\}$ es ortonormal, entonces $V_r^* V_r = I$, por lo que

$$A^* A V_r = V_r \Sigma_r^2 \Rightarrow V_r^* A^* A V_r = V_r^* V_r \Sigma_r^2 \Rightarrow V_r^* A^* A V_r = \Sigma_r^2$$

Luego,

$$V_r^* A^* A V_r = \Sigma_r^2 \Rightarrow \Sigma_r^{-1} V_r^* A^* A V_r \Sigma_r^{-1} = I \tag{3.3}$$

Ahora, denote $U_r = A V_r \Sigma_r^{-1}$, de donde

$$U_r^* = \Sigma_r^{-1} V_r^* A^* \tag{3.4}$$

De (3.3), entonces se nota que $U_r^* U_r = I$ y por ende U_r es una matriz unitaria de tamaño $m \times r$. Considere ahora otra matriz unitaria U_{m-r} de tamaño $m \times (m-r)$ tal que U_{m-r} sea ortogonal a U_r. Esto es, que cada vector-columna de U_{m-r} sea ortogonal a todo vector-columna de U_r, de donde $U_{m-r}^* U_r = \mathbf{0}$. La existencia y forma para construir a U_{m-r}, la garantiza el proceso de ortogonalización de Grand-Schmidt. Ahora, defina $U = (U_r \;\; U_{m-r})$ y $V = (V_r \;\; V_{n-r})$. Así,

$$U^* A V = (U_r \;\; U_{m-r})^* A (V_r \;\; V_{n-r}) = \begin{pmatrix} U_r^* \\ U_{m-r}^* \end{pmatrix} A (V_r \;\; V_{n-r}) \tag{3.5}$$

$$= \begin{pmatrix} U_r^* A \\ U_{m-r}^* A \end{pmatrix} (V_r \;\; V_{n-r}) = \begin{pmatrix} U_r^* A V_r & U_r^* A V_{n-r} \\ U_{m-r}^* A V_r & U_{m-r}^* A V_{n-r} \end{pmatrix} \tag{3.6}$$

Como $A v_i = \mathbf{0}$, para $i = r+1, \ldots, n$, entonces $A V_{n-r} = \mathbf{0}$. Además, de las ecuaciones (3.2) y (3.4) se tiene que $U_r^* A V_r = \Sigma_r^{-1} V_r^* A^* A V_r = \Sigma_r^{-1} V_r^* V_r \Sigma_r^2 = \Sigma_r$. Por otra parte, como $U_r = A V_r \Sigma_r^{-1}$, entonces $U_r \Sigma_r = A V_r$, por lo tanto de la construcción de U_{m-r} se obtiene: $U_{m-r}^* A V_r = U_{m-r}^* U_r \Sigma_r = \mathbf{0} \cdot \Sigma_r = \mathbf{0}$. Regresando a la ecuación (3.5) se observa que:

$$U^* A V = \begin{pmatrix} U_r^* A V_r & U_r^* A V_{n-r} \\ U_{m-r}^* A V_r & U_{m-r}^* A V_{n-r} \end{pmatrix} = \begin{pmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$$

Si a esta última matriz se le denomina Σ, entonces

$$U^* A V = \Sigma \Rightarrow U U^* A V V^* = U \Sigma V^* \Rightarrow A = U \Sigma V^*,$$

donde $U_{m \times m}$ y $V_{n \times n}$ son matrices unitarias[2], tal y como se quería probar.

♣

A partir de la SVD de A, como U y V^* son de rango completo (por ser unitarias y por ende invertibles), entonces $\mathrm{r}(A) = \mathrm{r}(U \Sigma V^*) = \mathrm{r}(\Sigma)$. Así, $\mathrm{r}(A) = r$ es el número de entradas no nulas de la matriz Σ. Por su parte, la matriz A no tiene por qué ser de rango completo.

En la prueba del teorema (5) se observa que los vectores propios en V_{n-r} están asociados a los valores propios nulos de $A^* A$. Por ende, para la construcción de las últimas $n - r$ columnas de V basta

[2]Como $U_r^* U_r = I$, $U_{m-r}^* U_{m-r} = I$ y $U_{m-r}^* U_r = \mathbf{0}$, entonces fácilmente se ve que $U^* U = I$ y, por lo tanto, U es unitaria. Similar pasa con la matriz V.

seleccionar $n-r$ vectores ortonormales del núcleo de[3] A, para ello se resuelve el sistema $Av = \mathbf{0}$. Así, $v_1, \ldots, v_r, v_{r+1}, \ldots, v_n$ forman una base ortonormal para $\mathbb{C}^n$. A estos vectores se les suele llamar *vectores singulares por la derecha*.

Adicionalmente, en la prueba se parte del análisis de los valores y vectores propios de la matriz A^*A. Sin embargo, una construcción similar pudo haberse realizado a partir de la matriz AA^*. Note que:

$$A = U\Sigma V^* \quad \Rightarrow \quad A^* = V\Sigma^* U^* \Rightarrow AA^* = U\Sigma V^* V\Sigma^* U^*$$

$$\Rightarrow \quad AA^* = U\Sigma\Sigma^* U^* \Rightarrow AA^*U = U\Sigma\Sigma^*$$

La igualdad $AA^*U = U\Sigma\Sigma^*$ da una diagonalización de AA^*. En efecto, si se denota u_j a las columnas de U, entonces se tiene que $AA^*u_j = \sigma_j^2 u_j$, para $j = 1, \ldots, m$. Por lo tanto, las columnas de U corresponden a vectores propios de AA^*, asociados a los valores propios σ_j^2 de AA^*, que a su vez corresponden a los valores propios de A^*A. En resumen, los valores propios de las matrices AA^* y A^*A coinciden, y corresponden a los valores singulares de la matriz A. Por lo tanto, para determinar los valores singulares de una matriz A conviene trabajar con A^*A o AA^*, según cuál sea de menor tamaño. Esto con el objetivo de tomar el polinomio característico de menor grado.

Si el proceso se comienza con la matriz AA^*, entonces las últimas $m-r$ columnas de U se toman del núcleo de A^*, esto es, se determinan $m-r$ vectores ortonormales a partir del sistema $A^*u = \mathbf{0}$. De esta manera $u_1, \ldots, u_r, u_{r+1}, \ldots, u_m$ forman una base ortonormal para $\mathbb{C}^m$. A estos vectores se les suele llamar *vectores singulares por la izquierda*. Antes de establecer algunos ejemplos, tome en cuenta que:

- Para construir la matriz V si ya se tiene la matriz U, basta aplicar el hecho que

  $$A = U\Sigma V^* \Rightarrow A^* = V\Sigma^* U^* \Rightarrow A^*U = V\Sigma^*$$

  Así, $A^*u_j = \sigma_j v_j$. Por lo tanto, para los $\sigma_j \neq 0$, $1 \leq j \leq \text{mín}\{m, n\}$, se tiene:

  $$v_j = \frac{1}{\sigma_j} A^* u_j \tag{3.7}$$

  La ortogonalidad de los vectores u_j implica que los vectores v_j construidos en (3.7) sean ortogonales también. En efecto, si $i \neq j$, entonces:

  $$\langle v_j, v_i \rangle = v_i^* v_j = \left(\tfrac{1}{\sigma_i} A^* u_i\right)^* \tfrac{1}{\sigma_j} A^* u_j = \tfrac{1}{\sigma_i \sigma_j} u_i^* A A^* u_j = \tfrac{1}{\sigma_i \sigma_j} u_i^* \sigma_j^2 u_j = \tfrac{\sigma_j}{\sigma_i} \langle u_j, u_i \rangle = 0$$

  Por lo tanto, solamente falta normalizarlos. Si ya se tienen todos los v_j, listo; si no, se busca una base ortonormal del $\text{Ker}(A)$ y se agregan para formar a V.

- Si se tiene la matriz V, entonces la matriz U puede ser construida de la siguiente manera. Como $A = U\Sigma V^*$, entonces $AV = U\Sigma$. Luego, $Av_j = \sigma_j u_j$, donde $1 \leq j \leq \text{mín}\{m, n\}$. Para los $\sigma_j \neq 0$ se obtiene:

  $$u_j = \frac{1}{\sigma_j} A v_j$$

[3]En realidad, se deben tomar del núcleo de A^*A, sin embargo, $\text{Ker}(A) \subset \text{Ker}(A^*A)$. Por ello es suficiente con construir una base ortonormal para el $\text{Ker}(A)$.

Similar a como sucede en el caso anterior, la ortogonalidad de los vectores v_j implica la ortogonalidad de los vectores u_j, los cuales deben ser normalizados. Si ya se tienen todos los u_j, listo; si no, se busca una base ortonormal del $\mathrm{Ker}(A^*)$ y se agregan para formar a U.

3.1. Reducción de la SVD

En muchas aplicaciones es común utilizar una versión simplificada de la SVD. Utilizando las notaciones empleadas en el teorema 5 se tiene que:

$$A = U\Sigma V^* = (U_r \;\; U_{m-r}) \cdot \begin{pmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \cdot (V_r \;\; V_{n-r})^* \tag{3.8}$$

$$= (U_r\Sigma_r \;\; \mathbf{0}) \cdot \begin{pmatrix} V_r^* \\ \\ V_{n-r}^* \end{pmatrix} = U_r\Sigma_r V_r^* \tag{3.9}$$

En síntesis, dada $A \in \mathrm{Mat}(\mathbb{C}, m, n)$, la *descomposición reducida* en valores singulares de A está dada por $A = U_r\Sigma_r V_r^*$, donde $r = \mathrm{r}(A)$, que corresponde al número de valores singulares no nulos de A, que a su vez son los valores propios no nulos de las matrices AA^* y A^*A. Las matrices U_r, Σ_r y V_r^* son como se definieron previamente. Se advierte que dicha reducción favorece porque utiliza matrices de menor tamaño, sin embargo, se pierden propiedades sobre ellas. Por ejemplo, las matrices U y V en la descomposición completa son unitarias, pero U_r y V_r no necesariamente lo son (ni siquiera tienen que ser cuadradas).

Si se visualiza a U_r como una matriz de 1 bloque-fila y 1 bloque-columna (es decir, un único bloque, la matriz total) y a Σ_r como de 1 bloque-fila y r bloques-columna, entonces:

$$U_r\Sigma_r = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1r} \\ u_{21} & u_{22} & \dots & u_{2r} \\ \vdots & & \dots & \vdots \\ u_{m1} & u_{m2} & \dots & u_{mr} \end{pmatrix}_{1\times 1} \left(\begin{array}{c|c|c|c} \sigma_1 & 0 & \dots & 0 \\ 0 & \sigma_2 & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \dots & \sigma_r \end{array}\right)_{1\times r}$$

$$= \left(\begin{array}{c|c|c|c} \sigma_1 u_{11} & \sigma_2 u_{12} & \dots & \sigma_r u_{1r} \\ \sigma_1 u_{21} & \sigma_2 u_{22} & \dots & \sigma_r u_{2r} \\ \vdots & & & \vdots \\ \sigma_1 u_{m1} & \sigma_2 u_{m2} & \dots & \sigma_r u_{mr} \end{array}\right)_{1\times r} = (\sigma_1 u_1 \mid \sigma_2 u_2 \mid \dots \mid \sigma_r u_r)_{1\times r}$$

Ahora, como $V_r = (v_1 \;\; v_2 \;\; \dots \;\; v_r)$, entonces V_r^* puede ser visualizada como una matriz por bloques de r bloques-fila y 1 bloque-columna. Es decir, $V_r^* = \left(\begin{array}{c} v_1^* \\ \hline v_2^* \\ \hline \vdots \\ \hline v_r^* \end{array}\right)_{r\times 1}$.

Por ende,

$$\begin{aligned} U_r\Sigma_r V_r^* &= (\sigma_1 u_1 \mid \sigma_2 u_2 \mid \ldots \mid \sigma_r u_r)_{1\times r} \cdot \left(\begin{array}{c} v_1^* \\ \hline v_2^* \\ \hline \vdots \\ \hline v_r^* \end{array}\right)_{r\times 1} \\ &= \sigma_1 u_1 v_1^* + \sigma_2 u_2 v_2^* + \ldots + \sigma_r u_r v_r^* \end{aligned}$$

A la igualdad $U_r\Sigma_r V_r^* = \sigma_1 u_1 v_1^* + \sigma_2 u_2 v_2^* + \ldots + \sigma_r u_r v_r^*$ se le conoce como la SVD reducida y es fácilmente aplicable en contextos como la compresión de imágenes, tal y como se mostrará en la siguiente sección.

4. APLICACIONES DE LA SVD

En esta sección se sintetizan algunas de las aplicaciones de la SVD. Los experimentos numéricos desarrollados en esta sección se realizaron en una computadora del Instituto Tecnológico de Costa Rica. La computadora contiene un procesador Intel(R) Core(TM) i7-4712MQ CPU 2.30GHz, memoria 8GB, utilizando el programa de cálculo numérico MATLAB.

4.1. Compresión de imágenes

La combinación lineal

$$U_r\Sigma_r V_r^* = \sigma_1 u_1 v_1^* + \sigma_2 u_2 v_2^* + \ldots + \sigma_\alpha u_\alpha v_\alpha^* + \ldots + \sigma_r u_r v_r^*, \tag{4.1}$$

con $1 \leq \alpha \leq r$, muestra cómo la SVD reducida de una matriz puede ser vista como una técnica de compresión de imágenes. Dado que $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$, entonces dicha combinación agrupa de manera ordenada la información que se extrajo de la matriz de datos (que representa a la imagen tratada). Por lo tanto, la matriz $\sigma_1 u_1 v_1^*$ es la que contiene la información más significativa, en caso que se desee reconstruir la imagen original. Si $\sigma_1 > \sigma_2$, entonces la matriz $\sigma_2 u_2 v_2^*$ es la segunda en contener información de mayor calidad para realizar la reconstrucción, y así sucesivamente. En la Figura 1 se utiliza la imagen boat.512.tiff (ver [26]), que es de tamaño 512×512 pixeles, lo cual genera una matriz[4] A de 262144 entradas. Con respecto a la Figura 1, en (a) se muestra la imagen original y, posteriormente, se muestra la reconstrucción de la imagen usando diferentes truncamientos para la combinación lineal. Por ejemplo, para el caso $\alpha = 10$ se realiza el producto matricial $U_{10}\Sigma_{10}V_{10}^*$. Para efectos de almacenamiento, note que U_{10} es una matriz de $512 \cdot 10 = 5120$ entradas, Σ_{10} puede almacenarse como un vector de 10 entradas y V_{10}^* también tiene $10 \cdot 512 = 5120$ entradas. Es decir, en total 10250 entradas. Evidentemente, la compresión con $\alpha = 10$ genera mucha pérdida de información con respecto a la imagen original. De manera similar, en (f) se muestra la reconstrucción realizada con $\alpha = 100$, la cual es de mucho más calidad. En este caso se estarían almacenando 102500 entradas, que es menos de la mitad del número de entradas que representan a la matriz original. Evidentemente,

[4]Para el lector que no se encuentra familiarizado con la representación computacional de una imagen puede consultar alguna referencia afín. En particular puede ver [14].

entre más alto sea el valor de α (que está acotado por $\mathrm{r}(A)$), mayor calidad tendrá la reconstrucción, pero baja el nivel de comprensión.

A modo de ensayo, las matrices generadas con $\alpha = 10$ y $\alpha = 100$ fueron almacenadas por separado en archivos de texto. Lo mismo se realizó con la matriz A, la cual generó un archivo de tamaño 2.25 MB. La Tabla I muestra la información obtenida. Nótese que para el caso de $\alpha = 100$, los tres archivos almacenados en conjunto tienen un tamaño de 951 KB, que representa aproximadamente el 41 % del tamaño del archivo generado con la matriz A. En este ejercicio la comprensión resultó bastante beneficiosa con $\alpha = 100$. Una buena estrategia para seleccionar un valor adecuado para α es realizar el gráfico de las magnitudes de los valores singulares de A. La Figura 2 muestra, para el ejemplo, el comportamiento de los primeros 100 valores no nulos. De la gráfica se nota que σ_{50} y σ_{100} son pequeños en magnitud, comparados con los primeros valores singulares de A. Por ello, es razonable que haber pasado de $\alpha = 50$ a $\alpha = 100$ no generara una mejoría tan significativa en la reconstrucción de la imagen, como sí sucedió al pasar de $\alpha = 10$ a $\alpha = 50$.

Matriz	U_{10}, diag(S_{10}), V_{10}	U_{100}, diag(S_{100}), V_{100}
Tamaño del archivo	48.4 KB, 111 bytes, 48.2 KB	475 KB, 1011 bytes, 475 KB

Tabla 1: Tamaño de los archivos de textos generados con $\alpha = 10$ y $\alpha = 100$.

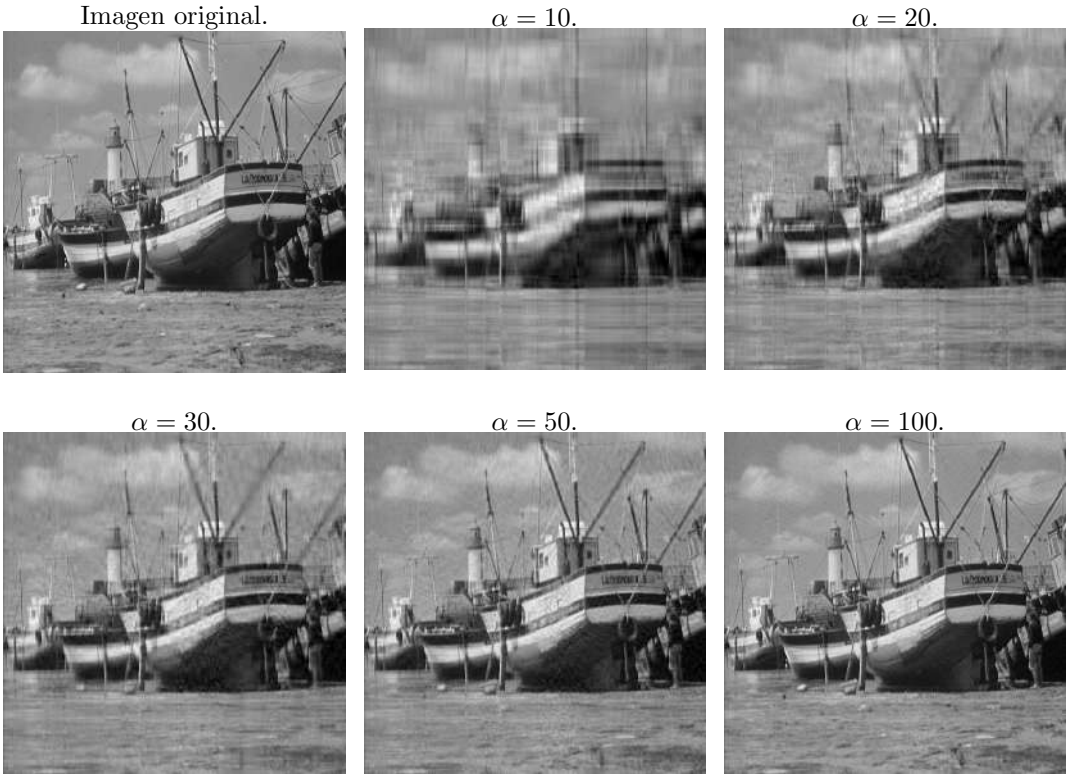

Figura 1: Reconstrucción de una imagen usando diferentes valores para α en la SVD reducida.

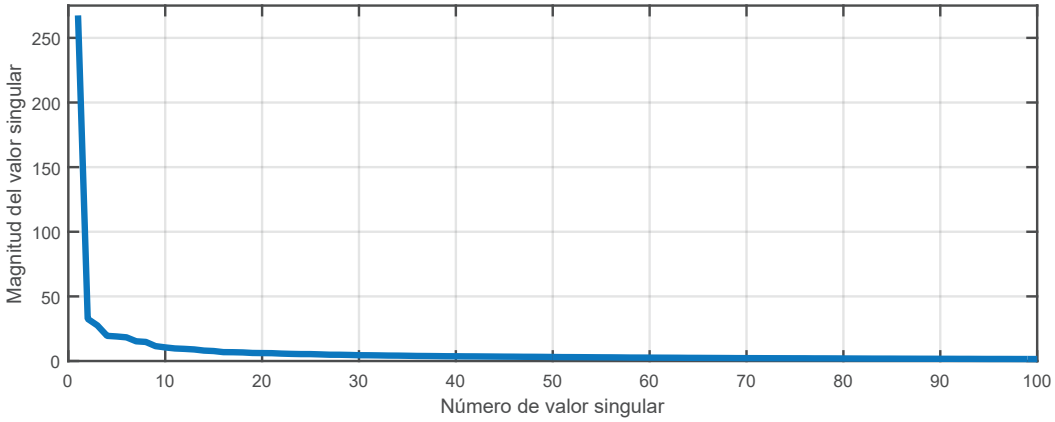

Figura 2: Gráfica de los valores singulares de la matriz A asociada a la imagen boat.512.tiff.

4.2. Cálculo de la pseudoinversa de una matriz

En muchas aplicaciones surgen sistemas de ecuaciones lineales de la forma $AX = B$, en los que la matriz de coeficientes es cuadrada y no singular, los cuales admiten la solución única $X = A^{-1}B$. Sin embargo, también es común que surjan sistemas lineales en los que la matriz A es singular. Para estos casos se puede recurrir a un concepto más general que la matriz inversa (que involucra incluso matrices no cuadradas), que permite generar matrices que se relacionan con las soluciones del sistema.

Dada $A \in \mathrm{Mat}(\mathbb{C}, m, n)$, una *inversa generalizada* de A es cualquier matriz de tamaño $n \times m$, denotada A^-, que satisface que $AA^-A = A$. Si la matriz A es no singular, entonces es claro que A^{-1} satisface la condición anterior. Adicionalmente, si A^- es cualquier otra inversa generalizada para la matriz no singular A, entonces $AA^-A = A$, de donde $A^- = A^{-1}AA^-AA^{-1} = A^{-1}AA^{-1} = A^{-1}$. En resumen, para matrices no singulares la inversa generalizada siempre existe, es única y corresponde a la matriz A^{-1}. En caso que A sea singular, la inversa generalizada siempre existe, pero no necesariamente es única (ver [15] para más detalles).

Este criterio se puede aplicar al sistema de ecuaciones lineales consistente $AX = B$, donde A no tiene que ser cuadrada ni invertible. Note que si A^- es una inversa generalizada de A, entonces $X' = A^-B$ es una solución del sistema $AX = B$. En efecto, si X_0 es cualquier solución de dicho sistema, entonces $AX_0 = B$, y así:

$$AX' = A(A^-B) = AA^-B = AA^-AX_0 = AX_0 = B$$

El concepto general presentado para una inversa generalizada es tan amplio (por la no unicidad), que puede ser poco aplicable. Por ello, existe un caso particular de inversa generalizada denominada *inversa de Moore-Penrose* o *pseudoinversa*, la cual siempre existe para cualquier matriz A y es única (ver [3]). Se define de la siguiente manera: dada una matriz $A \in \mathrm{Mat}(\mathbb{C}, m, n)$, la inversa de Moore-Penrose de A se denota $A^\dagger$ (de tamaño $n \times m$) y es la única matriz que satisface las condiciones $AA^\dagger A = A$ y $A^\dagger AA^\dagger = A^\dagger$ y que, además, las matrices $AA^\dagger$ y $A^\dagger A$ sean hermitianas (esto es, $(AA^\dagger)^* = AA^\dagger$ y $(A^\dagger A)^* = A^\dagger A$).

A partir de la SVD de la matriz $A \in \text{Mat}(\mathbb{C}, m, n)$ (refiérase al teorema 5), fácilmente se puede establecer una fórmula que permite calcular la inversa de Moore-Penrose. En efecto, si $A = U\Sigma V^*$, entonces $A^\dagger = V\Sigma^\dagger U^*$, donde $\Sigma^\dagger = \text{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}, 0, \ldots, 0) = \begin{pmatrix} \Sigma_r^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \in \text{Mat}(\mathbb{C}, m, n)$. Dada la unicidad de la pseudoinversa de Moore-Penrose, basta verificar que dicha matriz satisface la definición correspondiente. En efecto, como U y V son unitarias, entonces:

- $AA^\dagger A = U\Sigma V^* V\Sigma^\dagger U^* U\Sigma V^* = U\Sigma\Sigma^\dagger\Sigma V^* = U\Sigma V^* = A$, dado que:

$$\Sigma\Sigma^\dagger\Sigma = \begin{pmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \Sigma_r^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} I_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} = \begin{pmatrix} \Sigma_r & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} = \Sigma$$

  Siguiendo un razonamiento similar se tiene $\Sigma^\dagger\Sigma\Sigma^\dagger = \Sigma^\dagger$ y por ende

$$A^\dagger AA^\dagger = V\Sigma^\dagger U^* U\Sigma V^* V\Sigma^\dagger U^* = V\Sigma^\dagger U^* = A^\dagger$$

- Es claro que $\left(\Sigma\Sigma^\dagger\right)^* = \Sigma\Sigma^\dagger$, por lo tanto:

$$\begin{aligned} (AA^\dagger)^* &= (U\Sigma V^* V\Sigma^\dagger U^*)^* = (U\Sigma\Sigma^\dagger U^*)^* = U\left(\Sigma\Sigma^\dagger\right)^* U^* \\ &= U\Sigma\Sigma^\dagger U^* = U\Sigma V^* V\Sigma^\dagger U^* = AA^\dagger \end{aligned}$$

  Análogamente, como $\left(\Sigma^\dagger\Sigma\right)^* = \Sigma^\dagger\Sigma$, entonces se comprueba que $(A^\dagger A)^* = A^\dagger A$.

Con fines ilustrativos, a continuación se muestra el cálculo de $A^\dagger$ para dos de las matrices mostradas en los ejemplos de la sección 3.

$$A = \begin{pmatrix} 0 & 2 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \Rightarrow A^\dagger = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 \end{pmatrix}$$

$$A = \begin{pmatrix} 3 & 2 & 2 \\ 2 & 3 & -2 \end{pmatrix} \Rightarrow A^\dagger = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{3\sqrt{2}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & \frac{-1}{3\sqrt{2}} & \frac{-1}{\sqrt{3}} \\ 0 & \frac{2\sqrt{2}}{3} & \frac{1}{\sqrt{3}} \end{pmatrix} \begin{pmatrix} \frac{1}{5} & 0 \\ 0 & \frac{1}{3} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix} = \begin{pmatrix} \frac{7}{45} & \frac{2}{45} \\ \frac{2}{45} & \frac{7}{45} \\ \frac{2}{9} & \frac{-2}{9} \end{pmatrix}$$

Es importante aclarar que este método es muy preciso, pero poco eficiente en términos del rendimiento computacional, particularmente cuando se aplica a matrices grandes. Sin embargo, para acelerar el rendimiento en el cálculo de la pseudoinversa existen varios métodos iterativos, algunos de los cuales pueden consultarse en [5], [21] y [2].

Finalmente, el concepto de pseudoinversa se puede utilizar en el procesamiento de imágenes para la eliminación de ruido. Por ejemplo, considere la imagen en escala de grises de Einstein que se muestra en la Figura 3(a), la cual se representa numéricamente como una matriz X de dimensión 685×172. La imagen con ruido de la Figura 3(b) representa a la matriz Y construida como $Y = X + G$, donde G es una matriz aleatoria generada con una distribución normal. Para eliminar el ruido de la imagen se

usa un *filtro*, que corresponde a una matriz F de dimensión 685×685 y dada por $F = XY^{\dagger}$ (consultar [7] para más detalles), la cual se pre-multiplica a la matriz Y. La imagen 3(c) muestra el resultado del producto matricial $FY = XY^{\dagger}Y$.

Imagen original. Imagen con ruido. Imagen procesada.

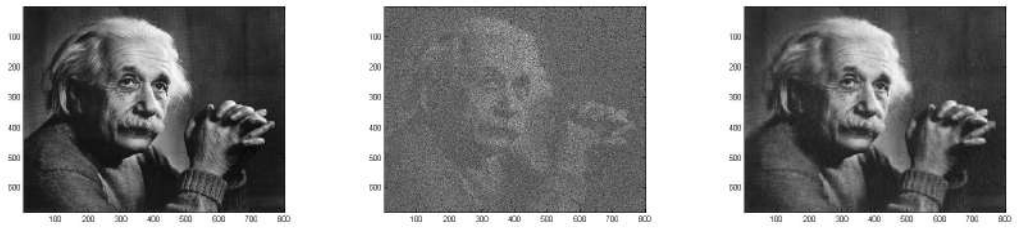

Figura 3: Ejemplo de aplicación de la pseudoinversa en procesamiento de imágenes.

4.3. Go decomposition (GoDec)

Dada una matriz A de rango r, en [9] y [17] se prueba cómo la SVD se puede utilizar para determinar una aproximación L_k de A de *rango reducido*, esto es, que su rango sea a lo sumo k, donde $k < r$. Esta aproximación se relaciona con la SVD reducida que se mostró en la ecuación (4.1). En efecto, si $A = U\Sigma V^*$ y $\mathrm{r}(A) = r$, de (4.1) se tiene que $A = U_r\Sigma_r V_r^* = \sigma_1 u_1 v_1^* + \sigma_2 u_2 v_2^* + \ldots + \sigma_r u_r v_r^*$. Luego, la mejor aproximación de A de rango *a lo sumo* k, está dada por:

$$L_k = \sigma_1 u_1 v_1^* + \sigma_2 u_2 v_2^* + \ldots + \sigma_k u_k v_k^* = U_k \Sigma_k V_k^*, \quad k < r$$

Así, $\min\limits_{\mathrm{r}(Y)\leq k} \|A - Y\|_{fr}^2 = \|A - L_k\|_{fr}^2$, donde $\|\cdot\|_{fr}$ denota a la norma de Frobenius[5]. Este resultado se conoce como el teorema de Eckart-Young y, además, la matriz L_k es única si y solo si $\sigma_k > \sigma_{k+1}$ [11].

El algoritmo iterativo *GoDec*, presentado en [30], emplea la SVD para determinar dos matrices L y S tales que $A \approx S + L$, donde L es de rango reducido y S es una matriz dispersa o rala (la mayor parte de sus entradas son ceros). Así, este algoritmo da una buena aproximación a la solución del problema:

$$\min_{\mathrm{r}(L)\leq k;\ \mathrm{card}(S)\leq c_0} \|A - L - S\|_{fr}^2,$$

donde card(S) denota la cantidad mínima de elementos no nulos en S. Esto es, se busca determinar una matriz L de rango a lo sumo k, $k < r$, y una matriz dispersa S con cardinalidad a lo sumo c_0 (parámetro de entrada en el algoritmo). Se suele escribir $A = L + S + G$, donde G es una matriz de ruido que modela el error de aproximación. En cada iteración del Algoritmo 1 se definen matrices L_t y S_t tales que $\mathrm{r}(L_t) \leq k$ y $\mathrm{card}(S_t) \leq c_0$. Las sucesiones $\{L_t\}$ y $\{S_t\}$ convergen linealmente a un mínimo local del problema de optimización [30]. La función $\mathcal{P}_{c_0}$ en dicho algoritmo (ver línea 6) recibe una matriz densa A' y retorna otra matriz dispersa de cardinalidad a lo sumo c_0, del mismo tamaño de

[5] Dada $A \in \mathrm{Mat}(\mathbb{C}, m, n)$, la norma de Frobenius de A se denota $\|A\|_{fr}$ y se define como $\|A\|_{fr} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}(a_{ij})^2}$. Como se puede notar, la norma de Frobenius de una matriz es similar a la norma euclidiana de un vector en $\mathbb{R}^n$.

A', la cual se construye manteniendo de A' las c_0 entradas de mayor magnitud (en valor absoluto) y las demás entradas iguales a cero.

Algoritmo 1 GoDec Clásico

Entrada: k, c_0, $A_{m\times n}$, ε.

Salida: L, S.

1: $L_0 := A$, $S_0 := \mathbf{0}_{m\times n}$, $t := 0$
2: **mientras** Verdadero **hacer**
3: $t := t + 1$.
4: $[U, \Sigma, V^*] = \text{svd}(A - S_{t-1})$.
5: $L_t := U_k \Sigma_k V_k^*$.
6: $S_t = \mathcal{P}_{c_0}(A - L_t)$.
7: $E_t := \|A - L_t - S_t\|_{fr}^2 / \|A\|_{fr}^2$.
8: **si** $|E_t - E_{t-1}| < \varepsilon$ **entonces**
9: **salir.**
10: **fin si**
11: **fin mientras**

Para ilustrar el Algoritmo 1, considere la matriz $A_{5\times 4}$, de rango 4, definida por:

$$A = \begin{pmatrix} 19 & 10 & 8 & 11 \\ -15 & 7 & 4 & -13 \\ -5 & -8 & 17 & 2 \\ 21 & 11 & -6 & 23 \\ 22 & -12 & 9 & 1 \end{pmatrix},$$

junto con los parámetros $k = 2$, $c_0 = 8$, $\varepsilon = 0.0001$, y las matrices $L_0 = A$ y $S_0 = \mathbf{0}_{5\times 4}$. Luego de 8 iteraciones se obtiene:

$$L_8 = \begin{pmatrix} 18.81311 & -1.66568 & 2.11459 & 11.26995 \\ -15.23693 & -2.94393 & 1.085 & -12.65785 \\ 1.19351 & -8.04379 & 5.30723 & -5.81271 \\ 20.94854 & 10.94608 & -5.98738 & 23.07557 \\ 22.04528 & -11.94952 & 8.99313 & 4.98487 \end{pmatrix}$$

y

$$S_8 = \begin{pmatrix} 0 & 11.66568 & 5.88541 & 0 \\ 0 & 9.94393 & 2.915 & 0 \\ -6.19351 & 0 & 11.69277 & 7.81271 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -3.98487 \end{pmatrix},$$

de tal manera que $\frac{\|A-L_8-S_8\|_{fr}^2}{\|A\|_{fr}^2} \approx 8.9948e^{-05}$. La Figura 4 muestra, para este ejemplo, el comportamiento del error a lo largo de las 8 iteraciones.

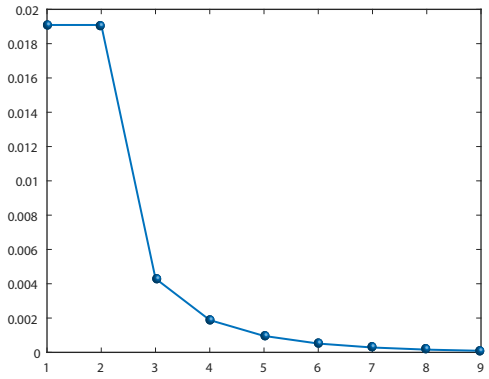

Figura 4: Comportamiento del error al aplicar el Algoritmo *GoDec* a la matriz A del ejemplo.

Expresar una matriz A como $A \approx S + L$, donde L es de rango reducido y S una matriz dispersa, tiene aplicación en el procesamiento de imágenes. En concreto, un video $\mathcal{V}$ (en el que se muestren objetos en movimiento) se puede descomponer en dos videos, de manera que uno de ellos muestre únicamente el fondo del video original (eliminando los objetos en movimiento) y el otro muestre solamente los objetos en movimiento. Para modelar este problema suponga que $\mathcal{V}$ está compuesto por K *frames*, todos del mismo tamaño y tal que el fondo sobre el que se mueven los objetos es invariante. Cada *frame* se modela por una matriz de tamaño $m \times n$, de tal manera que la matriz A sobre la cual itera el algoritmo *GoDec* es de tamaño $mn \times K$. La $i-$ésima columna de A corresponde a la matriz asociada al $i-$ésimo *frame*, la cual es reordenada como una matriz columna de tamaño $mn \times 1$. Así, cada columna de A corresponde a un *frame* de $\mathcal{V}$. Al aplicar el Algoritmo 1 a A se determinan las matrices L y S, de manera que para cada *frame* L almacena la información asociada al fondo y S contiene la información relativa a los objetos en movimiento. Por ende, la $i-$ésima columna de L (reescrita como matriz) corresponde al fondo del $i-$ésimo *frame* (pixeles que se mantienen invariantes), y la $i-$ésima columna de S (reescrita como matriz) corresponde a los pixeles de los objetos en movimiento del $i-$ésimo *frame*.

En la Figura 5 se observan parte de los resultados de la aplicación del algoritmo *GoDec* a un video de 60 *frames* (ver [24]), cada uno de tamaño 540×960, el cual consta de una plazoleta con personas en movimiento. En particular, se muestran los resultados asociados a los *frames* 1, 30 y 60, para visualizar cómo el algoritmo extrajo el fondo e identificó en cada *frame* los objetos en movimiento. Las imágenes (b), (e) y (h) muestran algunas personas como parte del fondo, dado que se mantienen estáticas a lo largo del video, o con movimientos muy leves. En caso que esto último ocurra, los objetos se observan como ligeras sombras en las imágenes que modelan el movimiento en cada uno de los *frames*. En la Figura 6 se resaltan dos ejemplos concretos en donde este fenómeno sucede.

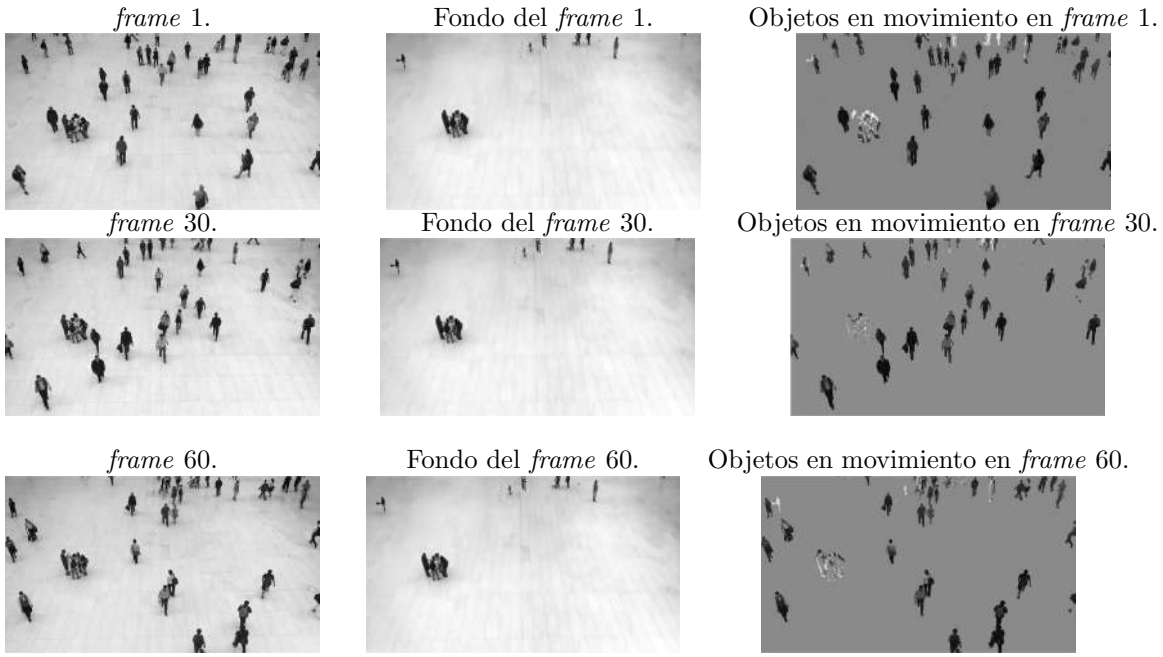

Figura 5: Procesamiento de un vídeo de 60 *frames* con el algoritmo *GoDec*.

Los parámetros utilizados para el procesamiento del video fueron[6]: $k = 2$, $\varepsilon = 10^{-9}$ y $c_0 = \lfloor 0.07 \cdot m \cdot n \cdot K \rfloor = 2177280$. Esto es, c_0 se definió como un porcentaje de la cantidad total de pixeles en el video (un 7 %, para el ejemplo). Tomando en cuenta que c_0 controla la cardinalidad de S, la matriz que modela los objetos en movimiento, entonces es razonable definir a c_0 en función de la cantidad total de pixeles. El valor 0.07 se ajustó empíricamente, luego de varias ejecuciones del algoritmo. Si el video procesado tiene muchos objetos en movimiento (u objetos en movimiento que ocupan mucha área en cada uno de los *frames*), entonces dicha constante de proporcionalidad debe ser mayor. Por su parte, la selección de $k = 2$ se hizo de esa manera (un valor bajo) porque el fondo de $\mathcal{V}$ permanece relativamente invariante a lo largo de los 60 *frames* (igual pudo haberse seleccionado $k = 1$ y los resultados obtenidos son similares). En caso que el fondo varíe a lo largo del video, entonces el valor de k debe incrementarse acorde con el número de variaciones significativas que se detecten. En resumen, tanto para la selección de c_0 como de k, debe mediar un análisis previo del video y, posteriormente, realizar varias ejecuciones del algoritmo, que en conjunto permitan ajustar los parámetros y así lograr la separación requerida entre el fondo y los objetos en movimiento.

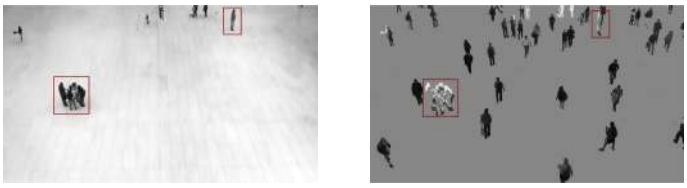

Figura 6: Objetos con poco movimiento en $\mathcal{V}$ que quedan como parte del fondo al aplicar *GoDec*.

[6]La notación $\lfloor \cdot \rfloor$ significa la función parte entera. Así, $\lfloor x \rfloor$ corresponde al mayor número entero menor o igual que x.

4.4. Algoritmo K-SVD

El algoritmo K-SVD, propuesto en [1], utiliza la descomposición en valores singulares para encontrar una representación esparcida de una matriz. Este algoritmo puede ser utilizado para la reconstrucción de imágenes con un cierto porcentaje de pixeles perdidos, tal y como se mostrará al final de la sección.

Formalmente, dada $Y \in \mathbb{R}^{m\times n}$, el algoritmo K-SVD permite aproximar matrices $D \in \mathbb{R}^{m\times t}$ y $X \in \mathbb{R}^{t\times n}$, $1 \leq t \leq \text{mín}\{m, n\}$, que minimicen la expresión $\|Y - DX\|_{fr}^2$, sujeto a las condiciones $\text{Card}(x_i) \leq c_0$, para $i = 1, \ldots, n$, donde $x_i \in \mathbb{R}^n$ representa la i-ésima columna de X y $\text{Card}(x_i)$ *representa la cantidad de entradas de* x_i *diferentes de cero*. En este caso, la constante c_0 es un número entero positivo que se conoce como constante de esparcidad y corresponde a un parámetro de entrada para el algoritmo. En términos prácticos, este problema de optimización busca realizar una representación esparcida de los datos de entrada, determinando un cierto número de patrones elementales (denominados *átomos*) que combinados entre sí permitan realizar una reconstrucción. Estos átomos corresponden a vectores que son organizados como columnas en la matriz D, a la cual se le denomina un *diccionario*. En otras palabras, se entenderá como diccionario a la matriz que permite encontrar la representación dispersa de un conjunto de datos, mediante una combinación lineal de sus columnas.

Este algoritmo corresponde a un método iterativo que alterna entre la escasa codificación de los datos basada en el diccionario actual D y un proceso de actualización de los átomos del diccionario para que haya un mejor ajuste. En este caso, la matriz D es el diccionario de la matriz esparcida X, que permite realizar una aproximación de la matriz Y. La actualización de las columnas de D se combina con una actualización de las representaciones esparcidas de las columnas de X, acelerando así la convergencia. Las matrices D y X se obtienen de manera iterativa, mediante el Algoritmo 2.

El Algoritmo 2, en el paso 3, plantea aproximar cada columna de X, sujeto a una condición de esparcidad. Este problema de optimización ha sido ampliamente estudiado y para su abordaje se utilizan algoritmos de *compressed sensing*. En particular para el ejemplo de aplicación del algoritmo K-SVD, que se mostrará posteriormente, en el paso 3 se utilizó el algoritmo *orthogonal matching pursuit* (OMP), que puede ser consultado en el capítulo 8 de [10]. Por otra parte, en el paso 8 del Algoritmo 2 tome en cuenta que:

- R_I es la matriz definida a partir de R tomando las columnas asociadas a los índices en I. Por ende, el tamaño de R_I es $m \times \text{Card}(I)$.
- d_j es la $j-$ésima columna de D (de dimensión $m \times 1$).
- $x_j(I)$ es el vector construido con las entradas no nulas de x_j (usando para ello los índices almacenados en I). Por lo tanto, los vectores I y $x_j(I)$ tienen tamaño $1 \times \text{Card}(I)$.

Finalmente, en el paso 11 se realiza la actualización de las entradas no nulas de la $j-$ésima columna de X a partir del vector $\sigma_1^2 v_1^T$. Esto es, de manera ordenada (acorde con los índices en I) se van actualizando las entradas no nulas de x_j, con las entradas de $\sigma_1^2 v_1^T$.

Una aplicación del algoritmo K-SVD consiste en la recuperación de pixeles perdidos en una imagen.

Algoritmo 2 Algoritmo K-SVD

Entrada: $Y \in \mathbb{R}^{m\times n}$, $D^{(0)} \in \mathbb{R}^{m\times t}$, $c_0 \in \{1, 2, ..., n\}$, $iter \in \mathbb{N}^*$.

Salida: $D^{(iter)} \in \mathbb{R}^{m\times t}$ y $X^{(iter)} \in \mathbb{R}^{t\times n}$.

1: $D = D^{(0)}$.
2: **para** $k = 1 : iter$ **hacer**
3: $\forall i = 1, ..., n$ determine $x_i = \underset{x}{\text{argmín}} \|y_i - Dx\|_2^2$, tal que $\text{Card}(x) \leq c_0$.
4: Defina $X = (x_1 \ \dots \ x_n)$.
5: Calcule $R = Y - DX$.
6: **para** $j = 1 : t$ **hacer**
7: Calcule I, el vector de índices asociados a las entradas de x_j no nulas.
8: Calcule $\widetilde{R} = R_I + d_j x_j(I)$.
9: Determine $\widetilde{R} = U\Sigma V^T$ (SVD de $\widetilde{R}$).
10: Sustituya $d_j = u_1$, donde u_1 es la primera columna de U.
11: Sustituya $x_j(I) = \sigma_1^2 v_1^T$, donde σ_1 es el primer valor singular de $\widetilde{R}$ y v_1 es la primera columna de V.
12: **fin para**
13: Defina $D^{(k)} = (d_1 \ \dots \ d_t)$.
14: **fin para**

Para el ejemplo que se presentará se utilizó una base de 55 imágenes (cada una de tamaño 112×88 y tomadas de [12]), formada por los rostros de 11 personas. Cada imagen se seccionó en bloques de tamaño 8×8 (obteniendo 154 bloques para cada imagen y 8470 bloques en total) y cada bloque se organizó como un vector columna de tamaño 64×1. Si se denota con $y_1, y_2, ..., y_{8470}$ tales vectores columna, entonces se puede definir la matriz $Y = [y_1, y_2, ..., y_{8470}]$ de tamaño $m \times n = 64 \times 8470$. La Figura 7 muestra el proceso descrito para definir a la matriz Y.

Al aplicar el algoritmo K-SVD a la matriz Y con los parámetros $c_0 = 10$, $iter = 40$, $t = 150$ y $D^{(0)}$ una matriz generada aleatoriamente usando una distribución normal, se obtuvieron matrices $D^{(40)}$ y X^{40} tales que $Y \approx D^{(40)}X^{(40)}$. La matriz $D^{(40)}$ corresponde al diccionario generado con el Algoritmo 2, que permite reconstruir una imagen con pixeles perdidos, que puede haber sido generada con alguna de las 55 imágenes originales o a partir de otra imagen con características similares. En nuestro caso se utilizó una imagen (ver Figura 8) que está en la base de datos, pero diferente a las 55 imágenes usadas para construir la matriz Y.

Para reconstruir la imagen con pixeles perdidos, que también es de tamaño 112×88, se divide la imagen en 154 bloques de tamaño 8×8 y se realiza la reconstrucción por bloques. Cada bloque se transforma en un vector z_j de tamaño 64×1 y para cada $j = 1, 2, ..., 154$ se realizan los siguientes pasos:

- El vector z_j puede tener algunas entradas iguales a 0, las cuales representan los pixeles perdidos. Utilizando esta información se define la *matriz diezmada* asociada a z_j, denotada D_{z_j}, de la

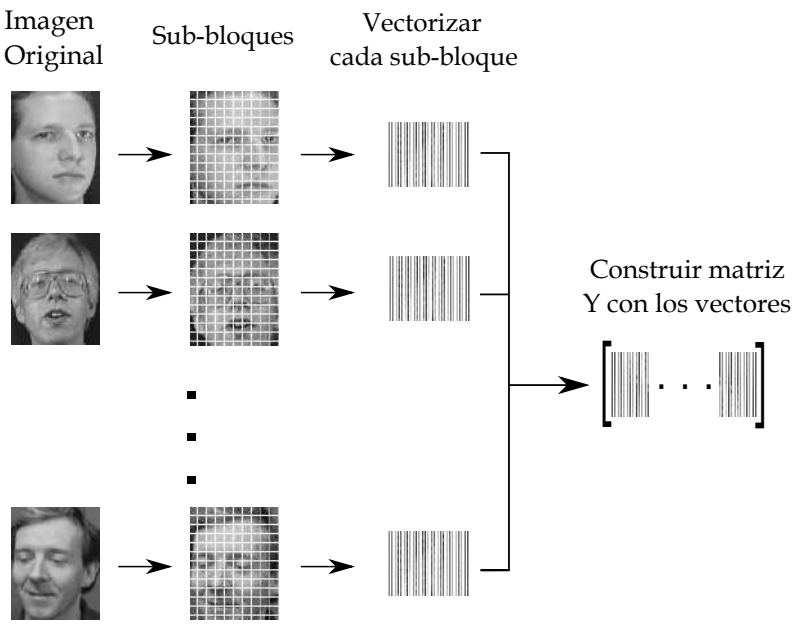

Figura 7: Representación matricial de las imágenes.

siguiente manera:

$$D_{z_j}(i,:) = \begin{cases} D^{(40)}(i,:) & \text{si } z_j(i) \neq 0 \\ \mathbf{0} & \text{si } z_j(i) = 0 \end{cases},$$

donde $D_{z_j}(i,:)$ y $D^{(40)}(i,:)$ representan la fila i de las matrices D_{z_j} y $D^{(40)}$, respectivamente, y $\mathbf{0}$ es el vector fila nulo. Esto es, la fila i de D_{z_j} se toma igual a la fila i de $D^{(40)}$, en caso que la $i-$ésima entrada del vector z_j no sea cero. Caso contrario, a la fila i de D_{z_j} se le asigna todas sus entradas iguales a cero.

- Utilizando un algoritmo de *compressed sensing* se determina x_{z_j}, tal que

$$x_{z_j} = \underset{x}{\text{argmín}} \|z_j - D_{z_j} x\|_2^2,$$

sujeto a $\text{Card}(x_{z_j}) \leq c_0$. En este caso nuevamente se utilizó el algoritmo OMP (orthogonal matching pursuit).

- Finalmente, para reconstruir el $j-$ésimo bloque se realiza la multiplicación $D^{(40)} \cdot x_{z_j}$. El resultado de este producto se convierte en un bloque de tamaño 8×8, y corresponde a la reconstrucción del $j-$ésimo bloque de la fotografía.

La Figura 8 muestra los resultados obtenidos en la reconstrucción de la fotografía seleccionada.

75% de pixeles perdidos

Reconstrucción con el algoritmo K-SVD

Figura 8: Reconstrucción de una imagen con 75 % de pixeles perdidos mediante el algoritmo K-SVD.

4.5. Reconocimiento facial

La última aplicación que se mostrará en el artículo se basa en [29], la cual utiliza la descomposición en valores singulares para realizar un reconocimiento facial en una base de imágenes. Para ello suponga que se tienen N imágenes de rostros, cada una de tamaño $m \times n$ pixeles. Sea $S = (f_1\ f_2\ \dots\ f_N)$ la matriz de tamaño $mn \times N$, donde f_i representa la $i-$ésima imagen redimensionada como columna de tamaño $mn \times 1$. Por su parte, el rostro promedio, $\overline{f}$, se define como $\overline{f} = \frac{1}{N}\sum_{i=1}^{N} f_i$, que para efectos de la matriz S representa una columna promedio. A partir de S y $\overline{f}$, se define la matriz $A = (a_1\ a_2\ \dots\ a_N)$, donde $a_i = f_i - \overline{f}$, para todo $i = 1, \dots, N$.

Si $r = \mathrm{r}(A)$, entonces la SVD de la matriz A, dada por $A = U\Sigma V^T$, genera matrices unitarias U y V de la forma $U = (u_1\ u_2\ \dots\ u_r\ u_{r+1}\ \dots\ u_{mn})$ y $V = (v_1\ v_2\ \dots\ v_r\ v_{r+1}\ \dots\ v_N)$. Se sabe que las columnas de $U_r = (u_1\ u_2\ \dots\ u_r)$ forman una base ortonormal del espacio generado por las columnas de A. En este caso a cada vector u_i, con $i \leq r$, se le denomina *rostro-base* del *espacio de rostros* formado por el conjunto inicial de imágenes de entrenamiento. Dado un nuevo rostro f, de tamaño $m \times n$, también redimensionado como vector columna de tamaño $mn \times 1$, es posible determinar las coordenadas para $f - \overline{f}$ asociadas a la base generada por las columnas de U_r. Esto es, se busca el vector de coordenadas $w = (w_1, w_2, \dots, w_r)^T$, con $w_i \in \mathbb{R}$ para todo i, tal que $f - \overline{f} = w_1u_1 + w_2u_2 + \cdots + w_ru_r$. En efecto,

$$\begin{aligned} & f - \overline{f} = w_1u_1 + w_2u_2 + \cdots + w_ru_r \\ \iff \quad & f - \overline{f} = (u_1\ u_2\ \dots\ u_r) \cdot (w_1, w_2, \dots, w_r)^T \\ \iff \quad & (u_1\ u_2\ \dots\ u_r)^T \left(f - \overline{f}\right) = (w_1, w_2, \dots, w_r)^T \end{aligned}$$

Por ende, basta tomar $w = (u_1\ u_2\ \dots\ u_r)^T \left(f - \overline{f}\right)$. Similarmente, para cada una de los rostros de entrenamiento, f_i, se tiene que sus coordenadas en el espacio de rostros están dadas por los vectores $x_i = (u_1\ u_2\ \dots\ u_r)^T \cdot (f_i - \overline{f})$, para $i = 1, \dots, r$. Por lo tanto, para determinar si f corresponde a alguna de las imágenes del conjunto de rostros de entrenamiento (o alguna variante de alguno de ellas), se procede a comparar el vector $\mathbf{w}$ con los vectores de coordenadas x_i. Si $x_j = \underset{1 \leq i \leq r}{\text{argmín}} \|w - x_i\|_2$, entonces el rostro f será identificado como el rostro f_j si $\|w - x_j\|_2 < \varepsilon_0$, donde ε_0 es una tolerancia previamente definida. En caso contrario, f debe ser reportado como un rostro desconocido. El Algoritmo 3 muestra

los pasos a seguir para la implementación.

Algoritmo 3 Reconocimiento facial

Entrada: $\{f_1, f_2, \ldots, f_N\}$, f, ε_0.

Salida: El rostro identificado o un mensaje indicando que el rostro es desconocido.

1: Construya $S = (f_1\ f_2\ \ldots\ f_N)$.
2: Calcule $\overline{f} = \frac{1}{N}\sum_{i=1}^{N} f_i$.
3: Construya $A = (a_1\ a_2\ \ldots\ a_N)$, donde $a_i = f_i - \overline{f}$.
4: Calcule $A = U\Sigma V^T$.
5: Defina $U_r = (u_1\ u_2\ \ldots\ u_r)$, donde $r = \mathrm{r}(A)$.
6: Para $i = 1, \ldots, N$ calcule $x_i = (u_1\ u_2\ \ldots\ u_r)^T \cdot (f_i - \overline{f})$
7: Calcule $w = (u_1\ u_2\ \ldots\ u_r)^T \cdot (f - \overline{f})$
8: Calcule $\varepsilon = \min\limits_{1\leq i\leq r} \|w - x_i\|_2 = \|w - x_j\|_2$, para algún $j \in \{1, 2, \ldots, N\}$
9: **si** $\varepsilon < \varepsilon_0$ **entonces**
10: **retornar** El rostro corresponde a la imagen f_j.
11: **si no**
12: **retornar** El rostro es desconocido.
13: **fin si**

Para fines ilustrativos, el Algoritmo 3 se aplicó a un conjunto de 500 imágenes de rostros extraídas de [12], las cuales fueron reescalas para que quedaran del mismo tamaño (156×111). Adicionalmente, dichas imágenes fueron convertidas a escala de gris. La base de datos en [12] incluye, para cada persona, 15 imágenes diferentes, con variantes en la posición y la expresión facial. La matriz S fue construida utilizando las primeras 10 imágenes de las 50 personas que se muestran en la Figura 9. Las cinco imágenes restantes de cada persona se utilizaron para realizar el experimento de reconocimiento facial.

Figura 9: Imágenes de rostros utilizadas en el experimento.

La Figura 10 muestra dos ejemplos particulares de los resultados obtenidos al aplicar el Algoritmo 3. En cada ejemplo, en la parte superior se muestran cinco imágenes que no fueron incluidas en la matriz S. Por su parte, en la parte inferior se muestran las imágenes con las que el Algoritmo 3 realizó la identificación. Las flechas utilizadas indican a cuál imagen detectó el algoritmo, para cada caso.

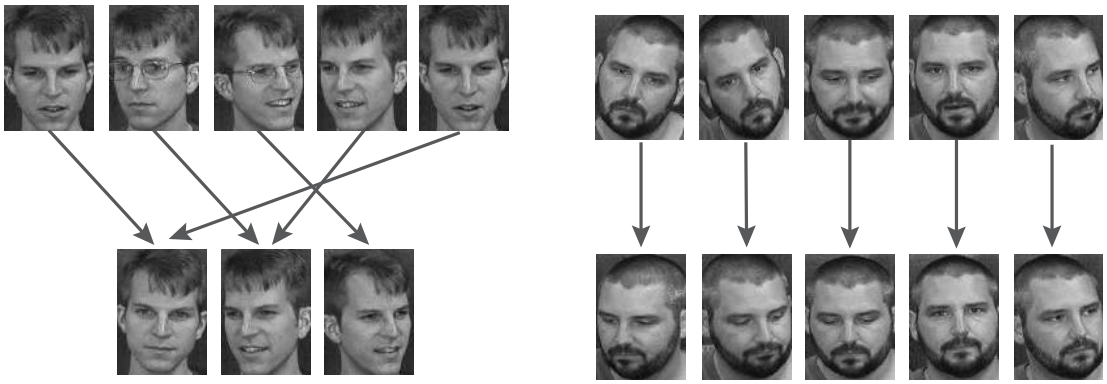

Figura 10: Resultados obtenidos al aplicar el algortimo de reconocimiento facial.

5. CONCLUSIONES

Este artículo presentó una revisión completa y autónoma de la descomposición en valores singulares de una matriz. En primera instancia se desarrolló con detalle los aspectos teóricos que permitieron justificar la existencia de la SVD y, posteriormente, se encausó al lector hacia aplicaciones concretas de la descomposición. En particular, la sección 3.1 mostró cómo la SVD logra condensar la información de una matriz en los subespacios asociados a los valores singulares de mayor magnitud, dando esto sentido a la aplicación de la compresión de imágenes (sección 4.1). Además, se presentaron otras aplicaciones de la SVD en el área del procesamiento de imágenes, abarcando los temas de modelado de fondo de imágenes para la detección de movimiento en videos (sección 4.3), eliminación de ruido de una imagen (secciones 4.2 y 4.4) y reconocimiento facial (sección 4.5).

Por la forma en que está estructurado, el artículo permite a un lector con conocimiento básico en Álgebra Lineal una mejor comprensión de este tipo de factorización. Por ello, puede ser considerado de interés como lectura complementaria en cualquier curso de Álgebra Lineal en el que se desee mostrar aplicaciones de esta área de la Matemática.

Reconocimientos:

Este artículo se realizó como parte del proyecto *FroSigPro* (código #1440037, periodo 2019-2020) inscrito en la Vicerrectoría de Investigación del Instituto Tecnológico de Costa Rica.

RECEIVED: JULY, 2020.

REVISED: NOVEMBER, 2020.

REFERENCIAS

[1] AHARON, M., ELAD, M. y BRUCKSTEIN, A. (2006): K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation, **IEEE Transactions on Signal Processing**, 54(11):4311–4322.

[2] ARTIDIELLO, S., CORDERO, A., and VASSILEVA, J. R. T. M. (2020): Generalized inverses estimations by means of iterative methods with memory, **Mathematics**, 8(1):1–13.

[3] BARATA, J.C. y HUSSEIN, M. S. (2011): The Moore–Penrose pseudoinverse: A tutorial review of the theory, **Brazilian Journal of Physics**, 42(1-2):146–165.

[4] BELTRAMI, E. (1873): Sulle funzioni bilineari, **Giornale di Matematiche ad Uso degli Studenti Delle Universita**, 11:98–106.

[5] BEN-ISRAEL, A. (1965): An iterative method for computing the generalized inverse of an arbitrary matrix, **Mathematics of Computation**, 19(91):452–455.

[6] BERNSTEIN, D.S. (2009): **Matrix Mathematics: Theory, Facts, and Formulas (Second Edition),** Princeton. Princeton University Press. New Jersey.

[7] CHUNG, J. y CHUNG, M. (2013): Computing optimal low-rank matrix approximations for image processing, In **2013 Asilomar Conference on Signals, Systems and Computers**, pages 670–674.

[8] DATTA, B.N. (2010): **Numerical Linear Algebra and Applications,** SIAM. Philadelphia.

[9] ECKART, C. y YOUNG, G. (1936): The approximation of one matrix by another of lower rank, **Psychometrika**, 1(3):211–218.

[10] ELDAR, Y.C. y KUTYNIOK, G. (2012): **Compressed Sensing: Theory and Applications,** Cambridge University Press. Cambridge.

[11] FRIEDLAND, S. y TOROKHTI, A. (2007): Generalized rank-constrained matrix approximations, **SIAM Journal on Matrix Analysis and Applications**, 29(2):656–659.

[12] GEORGIA TECH, Georgia tech face database, Disponible en `http://sipi.usc.edu/database/`. Consultado: 14-2,2020.

[13] GOLUB, G.H. y VAN LOAN, C.F. (2013): **Matrix Computations,** Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press. Baltimore.

[14] GONZALEZ, R., WOOD, R. y EDDINS, S. (2010): **Digital image processing using MATLAB,** Mc Graw Hill Education. India.

[15] HARVILLE, D.A. (2008): **Matrix Algebra From a Statistician's Perspective,** Springer. New York.

[16] HORN, R.A. y JOHNSON, C.R. (1990): **Matrix Analysis,** Cambridge University Press. Cambridge.

[17] JOHNSON, R. (1963): On a theorem stated by Eckart and Young, **Psychometrika**, 28(3):259–263.

[18] JOLLIFE, I.T. (2006): **Principal Component Analysis,** Springer. New York.

[19] JORDAN, C. (1874): Mémoire sur lesformes bilinéaires, **J. Math. Pures Appl., Deuxieme Série**, 19:35–54.

[20] LANG, S. (1987): **Linear Algebra,** Springer Undergraduate Texts in Mathematics and Technology. Springer. New York.

[21] LI, W. y LI, Z. (2010): A family of iterative methods for computing the approximate inverse of a square matrix and inner inverse of a non-square matrix, **Applied Mathematics and Computation**, 215(9):3433–3442.

[22] MARTÍNEZ-FERNÁNDEZ, J. J. (2005): La descomposición en valores singulares (svd) y algunas de sus aplicaciones, **La Gaceta de la RSME**, 8(3):796–810.

[23] PRATT, W.K. (2013): **Introduction to Digital Image Processing,** CRC Press.

[24] RYZE, M. (2018): Shopping, people, commerce, mall, many, crowd, walking free stock video footage youtube, Disponible en `https://www.youtube.com/watch?v=WvhYuDvH17I`. Consultado: 14-2,2020.

[25] STEWART, G. W. (1993): On the early history of the singular value decomposition, **SIAM Review**, 35(4):551–566.

[26] UNIVERSITY OF SOUTHERN CALIFORNIA, Signal and image processing institute, Disponible en `http://sipi.usc.edu/database/`. Consultado: 14-2,2020.

[27] WILLINK, T. J. (2008): Efficient adaptive SVD algorithm for MIMO applications, **IEEE Transactions on Signal Processing**, 56(2):615–622.

[28] WU, C. y TSAI, P. (2019): An svd processor based on Golub–Reinsch algorithm for MIMO precoding with adjustable precision, **IEEE Transactions on Circuits and Systems I: Regular Papers**, 66(7):2572–2583.

[29] ZENG, G. (2007): Facial recognition with singular value decomposition, In Elleithy, K., editor, **Advances and Innovations in Systems, Computing Sciences and Software Engineering**, pages 145–148, Dordrecht. Springer. Netherlands.

[30] ZHOU, T. y TAO, D. (2011): Godec: Randomized low-rank & sparse matrix decomposition in noisy case, **Proc. 28th ICML**, pages 33–40.