Universidad de Valladolid

FACULTAD DE CIENCIAS

TRABAJO FIN DE GRADO

Grado en Matemáticas

Aproximaciones de rango bajo de una matriz basadas en la descomposición en valores singulares.

Autor: Pablo González Castaño
Tutora: María Paz Calvo Cabrero
Año 2022

Índice general

Introducción

El Trabajo Fin de Grado que presentamos tiene por objetivo obtener aproximaciones de rango bajo a una matriz dada haciendo uso de su descomposición en valores singulares, o de una versión aleatorizada de dicha descomposición que requiere un coste computacional bastante menor.

En el Capítulo 1, se incluyen los resultados más relevantes sobre la descomposición en valores singulares de una matriz: el teorema que garantiza su existencia para cualquier matriz real, y el teorema de Eckart-Young, que proporciona a través de dicha descomposición la mejor aproximación a la matriz de partida por matrices de rango fijado, tanto en la norma espectral, como en la norma de Frobenius.

En el Capítulo 2, se introducen dos algoritmos que permiten obtener aproximaciones de rango bajo de una matriz de forma más eficiente, a cambio de perder algo de precisión. En concreto, se presenta un primer algoritmo para efectuar el producto aleatorizado de dos matrices, que se utiliza después para calcular la descomposición en valores singulares aleatorizada de una matriz. Además, se prueban algunos resultados teóricos sobre el error en las aproximaciones consideradas y se incluyen experimentos numéricos que ilustran dichos resultados.

En el Capítulo 3, se presentan dos aplicaciones prácticas de la descomposición en valores singulares. La primera se refiere a la incorporación de marcas de agua a imágenes, con vistas a probar la autoría o propiedad de las mismas, y se analizan los esquemas basados en la SVD. La segunda está relacionada con la extracción de información, y se centra en el análisis semántico latente, una propuesta que busca mejorar los sistemas de búsqueda basados en palabras individuales.

Finalmente, en el Apéndice A, se incluyen las funciones en Matlab que implementan los algoritmos introducidos en los Capítulos 2 y 3, y que han servido como base para crear las figuras que aparecen en esos mismos capítulos.

Capítulo 1

La descomposición en valores singulares

Comenzamos este primer capítulo enunciando y demostrando el teorema que garantiza la existencia de la descomposición en valores singulares de cualquier matriz real.

1.1. Teorema de existencia

Teorema 1.1.1. *Sea $A \in \mathbb{R}^{m\times n}$ una matriz de rango r. Existen matrices ortogonales $U \in \mathbb{R}^{m\times m}$ y $V \in \mathbb{R}^{n\times n}$ tales que $A = U\Sigma V^T$, donde $\Sigma \in \mathbb{R}^{m\times n}$ tiene elementos σ_{ij}, $1 \leq i \leq m$, $1 \leq j \leq n$, dados por*

$$\begin{aligned} \sigma_{ij} &= 0 \text{ para } i \neq j, \\ \sigma_{ii} &\neq 0 \text{ para } i \leq r, \\ \sigma_{ii} &= 0 \text{ para } i > r. \end{aligned}$$

Demostración. Vamos a tomar la matriz A^TA, que es simétrica y, por tanto, es diagonalizable ortogonalmente, es decir, existe $V \in \mathbb{R}^{n\times n}$ ortogonal de forma que

$$V^T(A^TA)V = D,$$

donde D es diagonal, con elementos diagonales $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r > \lambda_{r+1} = \cdots = \lambda_n = 0$, que son los autovalores de A^TA. Además, si $V = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n]$, donde $\mathbf{v}_i$ denota la columna i-ésima de V, se verifica que $\mathbf{v}_i$ es un autovector asociado a λ_i.
Ahora buscamos la matriz U. Para ello comprobamos en primer lugar que las imágenes por A de las columnas de V son ortogonales dos a dos

$$(A\mathbf{v}_i)^T(A\mathbf{v}_j) = \mathbf{v}_i^TA^TA\mathbf{v}_j = \mathbf{v}_i^T\lambda_j\mathbf{v}_j = \lambda_j\mathbf{v}_i^T\mathbf{v}_j = 0 \quad \text{si } i \neq j, \tag{1.1}$$

puesto que la matriz V es ortogonal. Tenemos por tanto un conjunto de n vectores ortogonales de $\mathbb{R}^m$ y además como el rango de A es r, $\lambda_i = \mathbf{0}$ para $i > r$ y, por tanto, $A\mathbf{v}_i = 0$ si $i > r$. Vemos que tanto las columnas de V como sus imágenes por A nos dan una expresión diagonal de A

$$(AV)^T AV = V^T A^T AV = D.$$

Normalicemos ahora los vectores $\{A\mathbf{v}_i\}_{i=1}^n$.
Si en (1.1) hacemos $j = i$, se comprueba que

$$\|A\mathbf{v}_i\|_2^2 = \lambda_i > 0, \qquad \text{si } i \leq r.$$

De este modo, $\lambda_i \geq 0$ y como hemos elegido V para que

$$\lambda_i \geq \lambda_j \quad \text{si} \quad i < j,$$

tenemos que

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r > 0,$$

con $\lambda_i = 0$ si $i > r$.
Tiene sentido entonces tomar para $1 \leq i \leq r$

$$\mathbf{u}_i = \frac{A\mathbf{v}_i}{\|A\mathbf{v}_i\|_2} = \frac{1}{\sqrt{\lambda_i}} A\mathbf{v}_i,$$

de forma que

$$A\mathbf{v}_i = \sqrt{\lambda_i}\mathbf{u}_i, \quad 1 \leq i \leq r.$$

Si ahora tomamos $\sigma_i = \sqrt{\lambda_i}$, $1 \leq i \leq r$, se tiene que

$$A\mathbf{v}_i = \sigma_i \mathbf{u}_i, \quad 1 \leq i \leq r,$$

que es exactamente lo que buscábamos.
En el caso de que $r < m$, basta con ampliar el sistema ortonormal $\{\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_r\}$ hasta una base ortonormal de $\mathbb{R}^m$.
Para concluir, tomamos como matriz U la que tiene por columnas los vectores $\mathbf{u}_i$, $1 \leq i \leq m$, con lo que tenemos

$$AV = U\Sigma,$$

donde Σ es la matriz de tamaño $m \times n$ con todos los elementos nulos, excepto los r primeros elementos diagonales que son iguales a $\sigma_1, \ldots, \sigma_r$, las raíces cuadradas de los autovalores no nulos de $A^T A$ ordenados en orden decreciente. Usando ahora la ortogonalidad de V se llega a la expresión buscada.

$$A = U\Sigma V^T. \tag{1.2}$$

□

Observación 1.1.1. *Llamaremos a las columnas de U, $\{\mathbf{u}_i\}_{i=1}^m$, y a las columnas de V, $\{\mathbf{v}_i\}_{i=1}^n$, los vectores singulares de A por la izquierda y por la derecha, respectivamente. Denominaremos valores singulares de A a los elementos diagonales no nulos de la matriz Σ, ordenados en orden decreciente, y los denotaremos por $\{\sigma_i\}_{i=1}^r$ o por $\{\sigma_i(A)\}_{i=1}^r$ en caso de que estemos hablando de varias matrices y pueda existir confusión. Notemos además que, dada la estructura de ceros de la matriz Σ, la igualdad (1.2) se puede escribir de manera más compacta como*

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \tag{1.3}$$

es decir, A está escrita como la suma de r matrices de rango 1.

Observación 1.1.2. *Basándonos en la demostración del Teorema 1.1.1, tenemos que los primeros r vectores singulares por la izquierda $\{\mathbf{u}_i\}_{i=1}^r$ son una base del subespacio generado por las columnas de A. Del mismo modo, los r primeros vectores singulares por la derecha $\{\mathbf{v}_i\}_{i=1}^r$ son una base del subespacio generado por las filas de A (el generado por las columnas de A^T). Es decir, los vectores singulares retienen gran parte de la información de la matriz original.*

1.2. Teorema de Eckart-Young

Una vez probado que existe la descomposición en valores singulares de cualquier matriz real A, vamos a ver que a partir de dicha descomposición podemos construir la mejor aproximación de rango k a dicha matriz. Pero antes, vamos a recordar la definición de dos normas matriciales que vamos a usar, y a introducir un lema auxiliar.

Definición 1.2.1. *Sea $A \in \mathbb{R}^{m\times n}$ una matriz con elementos a_{ij}, $1 \le i \le m$, $1 \le j \le n$. Se define la norma de Frobenius de A, que denotaremos por $\|A\|_F$, como*

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2}. \tag{1.4}$$

Definición 1.2.2. *Sea A una matriz como en la Definición 1.2.1. Se define la norma espectral de A (o la norma inducida por la norma euclídea), que denotaremos por $\|A\|_2$, como*

$$\|A\|_2 = \sup_{\boldsymbol{x}\neq 0} \frac{\|A\boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} = \max_{\|\boldsymbol{x}\|_2=1} \|A\boldsymbol{x}\|_2. \tag{1.5}$$

donde $\|\cdot\|_2$ denota la norma euclídea tanto en $\mathbb{R}^n$ como en $\mathbb{R}^m$.

Lema 1.2.1. *Sea $A \in \mathbb{R}^{m\times n}$ una matriz de rango r, y sea $A = U\Sigma V^T$ su descomposición en valores singulares. Entonces*

1. $\|A\|_2 = \sigma_1$.

2. $\|A\|_F = \sqrt{\sigma_1^2 + \cdots + \sigma_r}$.

Demostración.

1. Por como hemos elegido U y V, tomando el primer vector singular por la derecha $\mathbf{v}_1$, tenemos que
$$\frac{\|A\mathbf{v}_1\|_2}{\|\mathbf{v}_1\|_2} = \frac{\|\sigma_1\mathbf{u}_1\|_2}{\|\mathbf{v}_1\|_2} = \sigma_1,$$
puesto que $\mathbf{u}_1$ y $\mathbf{v}_1$ son unitarios, y $\sigma_1 > 0$. Tenemos entonces que
$$\|A\|_2 = \sup_{\mathbf{x}\neq 0} \frac{\|A\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \geq \sigma_1.$$
Para ver la otra desigualdad, recordamos que las columnas de V son una base ortonormal de $\mathbb{R}^n$ y por tanto para todo $\mathbf{x} \in \mathbb{R}^n$, existen $a_1, \ldots, a_n \in \mathbb{R}$ tales que
$$\mathbf{x} = a_1\mathbf{v}_1 + \cdots + a_n\mathbf{v}_n,$$
de modo que
$$\|\mathbf{x}\|_2^2 = |a_1|^2 + \cdots + |a_n|^2.$$
Si ahora multiplicamos $\mathbf{x}$ por A tenemos
$$A\mathbf{x} = a_1A\mathbf{v}_1 + \cdots + a_nA\mathbf{v}_n = a_1\sigma_1\mathbf{u}_1 + \cdots + a_r\sigma_r\mathbf{u}_r.$$
De nuevo aprovechando la ortonormalidad de los vectores singulares por la izquierda, podemos asegurar que
$$\|A\mathbf{x}\|_2^2 = |a_1\sigma_1|^2 + \cdots + |a_r\sigma_r|^2.$$
Además, $\sigma_i \leq \sigma_1$, para $i = 2, \ldots, r$, luego
$$\|A\mathbf{x}\|_2^2 \leq \sigma_1^2\left(|a_1|^2 + \cdots + |a_r|^2\right) \leq \sigma_1^2\|\mathbf{x}\|_2^2.$$
Como los valores singulares son no negativos, tomando raíces cuadradas concluimos que
$$\frac{\|A\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \leq \sigma_1, \qquad \text{para todo } \mathbf{x} \in \mathbb{R}^n,$$
y, por tanto, $\|A\|_2 \leq \sigma_1$.

2. Comencemos viendo que $\|A\|_F^2 = tr(A^TA)$. Para ello partimos de que el elemento diagonal j-ésimo de la matriz A^TA viene dado por

$$\left(A^TA\right)_{jj} = \sum_{i=1}^{m} a_{ij}^2 = \sum_{i=1}^{m} |a_{ij}|^2 .$$

Así, se tiene que

$$tr(A^TA) = \sum_{j=1}^{n} \left(A^TA\right)_{jj} = \sum_{j=1}^{n}\sum_{i=1}^{m} |a_{ij}|^2 = \|A\|_F^2.$$

Esto nos va a llevar a que si tenemos una matriz ortogonal $Q \in \mathbb{R}^{m\times m}$, entonces

$$\|QA\|_F = \sqrt{tr\left((QA)^T QA\right)} = \sqrt{tr\left(A^TQ^TQA\right)} = \sqrt{tr\left(A^TA\right)} = \|A\|_F.$$

De modo que no queda más que usar la descomposición en valores singulares de A para concluir que puesto que U y V son matrices ortogonales

$$\|A\|_F = \|U\Sigma V^T\|_F = \|\Sigma\|_F = \sqrt{\sum_{i=1}^{r} |\sigma_i|^2} = \sqrt{\sigma_1^2 + \cdots + \sigma_r^2}.$$

□

Ahora ya estamos en condiciones de enunciar y demostrar el siguiente resultado

Teorema 1.2.1. *(de Eckart-Young). Sea $A \in \mathbb{R}^{m\times n}$ una matriz de rango r, $A = U\Sigma V^T$ su descomposición en valores singulares y para $1 \leq k \leq r$ sea*

$$A_k = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T. \tag{1.6}$$

Entonces para toda matriz $B \in \mathbb{R}^{m\times n}$ de rango menor o igual que k, se tiene

$$\|A - A_k\|_F \leq \|A - B\|_F, \tag{1.7}$$

$$\|A - A_k\|_2 \leq \|A - B\|_2. \tag{1.8}$$

Es decir, A_k es la mejor aproximación de A por matrices de rango menor o igual que k en la norma de Frobenius y en la norma espectral.

Demostración. Antes de nada, notemos que, siguiendo la definición dada en la Observación 1.1.1, $A - A_k$ no es más que la matriz que tiene por valores singulares a $\{\sigma_i\}_{i=k+1}^{r}$
$(A - A_k = U(\Sigma - \Sigma_k)V^T)$, luego siguiendo el lema anterior, se tiene que

$$\|A - A_k\|_F^2 = \sigma_{k+1}^2 + \cdots + \sigma_r^2 \quad \text{y} \quad \|A - A_k\|_2^2 = \sigma_{k+1}^2.$$

Entonces para probar (1.7) y (1.8), es suficiente ver que para cualquier $B \in \mathbb{R}^{m\times n}$ de rango menor o igual que k se tiene

$$\|A-B\|_F^2 \geq \sigma_{k+1}^2 + \cdots + \sigma_r^2, \tag{1.9}$$

$$\|A-B\|_2^2 \geq \sigma_{k+1}^2, \tag{1.10}$$

respectivamente. Comencemos viendo (1.9).
Usando la descomposición en valores singulares de B, podemos expresar

$$B = \sum_{i=1}^{k} \mathbf{x}_i \mathbf{y}_i^T,$$

donde los vectores $\{\mathbf{y}_i\}_{i=1}^k$ son ortonormales y los vectores $\{\mathbf{x}_i\}_{i=1}^k$ son ortogonales y de norma euclídea $\sigma_i(B)$. Tenemos entonces

$$\begin{aligned}\|A-B\|_F^2 &= tr\left(\left(A-\sum_{i=1}^{k}\mathbf{x}_i\mathbf{y}_i^T\right)\left(A-\sum_{i=1}^{k}\mathbf{x}_i\mathbf{y}_i^T\right)^T\right)\\ &= tr\left(AA^T - A\sum_{i=1}^{k}\left(\mathbf{x}_i\mathbf{y}_i^T\right)^T - \sum_{i=1}^{k}\left(\mathbf{x}_i\mathbf{y}_i^T\right)A^T + \sum_{i=1}^{k}\left(\mathbf{x}_i\mathbf{y}_i^T\right)\left(\mathbf{x}_i\mathbf{y}_i^T\right)^T\right).\end{aligned} \tag{1.11}$$

Ahora bien,

$$\sum_{i=1}^{k}\left(\mathbf{x}_i\mathbf{y}_i^T\right)\left(\mathbf{x}_i\mathbf{y}_i^T\right)^T = \sum_{i=1}^{k}\mathbf{x}_i\|\mathbf{y}_i\|_2^2\mathbf{x}_i^T = \sum_{i=1}^{k}\mathbf{x}_i\mathbf{x}_i^T,$$

y también se tiene que

$$\begin{aligned}\sum_{i=1}^{k}\left(\mathbf{x}_i - A\mathbf{y}_i\right)\left(\mathbf{x}_i - A\mathbf{y}_i\right)^T &= \sum_{i=1}^{k}\left(\mathbf{x}_i\mathbf{x}_i^T - A\mathbf{y}_i\mathbf{x}_i^T - \mathbf{x}_i\left(A\mathbf{y}_i\right)^T + A\mathbf{y}_i\left(A\mathbf{y}_i\right)^T\right)\\ &= \sum_{i=1}^{k}\left(\mathbf{x}_i\mathbf{x}_i^T - A\left(\mathbf{x}_i\mathbf{y}_i^T\right)^T - \mathbf{x}_i\mathbf{y}_i^TA^T + A\mathbf{y}_i\mathbf{y}_i^TA^T\right).\end{aligned}$$

En consecuencia,

$$\sum_{i=1}^{k}\left(\mathbf{x}_i\mathbf{x}_i^T - A\left(\mathbf{x}_i\mathbf{y}_i^T\right)^T - \mathbf{x}_i\mathbf{y}_i^TA^T\right) = \sum_{i=1}^{k}\left(\left(\mathbf{x}_i - A\mathbf{y}_i\right)\left(\mathbf{x}_i - A\mathbf{y}_i\right)^T - A\mathbf{y}_i\mathbf{y}_i^TA^T\right),$$

y siguiendo los cálculos de (1.11),

$$\begin{aligned}\|A-B\|_F^2 &= tr(AA^T + \sum_{i=1}^{k}\left(\mathbf{x}_i - A\mathbf{y}_i\right)\left(\mathbf{x}_i - A\mathbf{y}_i\right)^T - \sum_{i=1}^{k}A\mathbf{y}_i\mathbf{y}_i^TA^T)\\ &= \|A\|_F^2 + \sum_{i=1}^{k}\|\mathbf{x}_i - A\mathbf{y}_i\|_F^2 - \sum_{i=1}^{k}\|A\mathbf{y}_i\|_F^2.\end{aligned}$$

Como el segundo sumando de la última igualdad es positivo, basta probar que

$$\sum_{i=1}^{r} \|A\mathbf{y}_i\|_F^2 \leq \sum_{i=1}^{k} \sigma_i^2,$$

pues entonces tendríamos

$$\begin{aligned}\|A - B\|_F^2 &\geq \|A\|_F^2 - \sum_{i=1}^{k} \|A\mathbf{y}_i\|_F^2 \\ &\geq \sum_{i=1}^{r} \sigma_i^2 - \sum_{i=1}^{k} \sigma_i^2 = \sum_{i=k+1}^{r} \sigma_i^2 = \|A - A_k\|_F^2.\end{aligned}$$

Para ello, siguiendo la descomposición en valores singulares de A y particionando V y Σ de forma conveniente en $V = (V_1|V_2)$, $\Sigma = (\Sigma_1|\Sigma_2)$ donde en V_1 y Σ_1 nos hemos quedado con las k primeras columnas, tenemos que

$$\|A\mathbf{y}_i\|_F^2 = \|U\Sigma V^T\mathbf{y}_i\|_F^2 = \|\Sigma V^T\mathbf{y}_i\|_F^2 = \|\Sigma_1 V_1^T\mathbf{y}_i\|_F^2 + \|\Sigma_2 V_2^T\mathbf{y}_i\|_F^2.$$

Del mismo modo,

$$\sigma_k^2\|V^T\mathbf{y}_i\|_F^2 = \sigma_k^2\|V_1^T\mathbf{y}_i\|_F^2 + \sigma_k^2\|V_2^T\mathbf{y}_i\|_F^2,$$

de manera que podemos asegurar que

$$\|A\mathbf{y}_i\|_F^2 = \sigma_k^2\|V^T\mathbf{y}_i\|_F^2 + \left(\|\Sigma_1 V_1^T\mathbf{y}_i\|_F^2 - \sigma_k^2\|V_1^T\mathbf{y}_i\|_F^2\right) + \left(\|\Sigma_2 V_2^T\mathbf{y}_i\|_F^2 - \sigma_k^2\|V_2^T\mathbf{y}_i\|_F^2\right).$$

Como los elementos de Σ_2 son todos menores o iguales que σ_k, $\left(\|\Sigma_2 V_2^T\mathbf{y}_i\|_F^2 - \sigma_k^2\|V_2^T\mathbf{y}_i\|_F^2\right) \leq 0$. Por otro lado, V es ortogonal y los vectores $\{\mathbf{y}_i\}_{i=1}^k$ son ortonormales, luego $\|V^T\mathbf{y}_i\|_F^2 = 1$. Por tanto,

$$\|A\mathbf{y}_i\|_F^2 \leq \sigma_k^2 + \left(\|\Sigma_1 V_1^T\mathbf{y}_i\|_F^2 - \sigma_k^2\|V_1^T\mathbf{y}_i\|_F^2\right) \leq \sigma_k^2 + \sum_{j=1}^{k} \left(\sigma_j^2 - \sigma_k^2\right) \left|\mathbf{v}_j^T\mathbf{y}_i\right|^2,$$

y podemos concluir que

$$\begin{aligned}\sum_{i=1}^{k} \|A\mathbf{y}_i\|_F^2 &\leq k\sigma_k^2 + \sum_{i=1}^{k} \left(\sum_{j=1}^{k} \left(\sigma_j^2 - \sigma_k^2\right) \left|\mathbf{v}_j^T\mathbf{y}_i\right|^2\right) \\ &= \sum_{j=1}^{k} \left(\sigma_k^2 + \left(\sigma_j^2 - \sigma_k^2\right) \sum_{i=1}^{k} \left|\mathbf{v}_j^T\mathbf{y}_i\right|^2\right) \\ &\leq \sum_{j=1}^{k} \left(\sigma_k^2 + \left(\sigma_j^2 - \sigma_k^2\right) \|\mathbf{v}_j\|_F^2\right) \\ &= \sum_{j=1}^{k} \sigma_j^2.\end{aligned}$$

Veamos ahora (1.10). Comenzamos eligiendo un vector $\mathbf{x} \neq \mathbf{0}$ tal que $B\mathbf{x} = \mathbf{0}$ y

$$\mathbf{x} = \sum_{j=1}^{k+1} c_j \mathbf{v}_j.$$

Claramente este $\mathbf{x}$ existe, ya que el rango de B es menor o igual que k, luego la dimensión de su núcleo es mayor que $n-k$. Por otro lado, como V es ortogonal, $\{\mathbf{v}_1, \ldots, \mathbf{v}_{k+1}\}$ generan un subespacio vectorial de dimensión $k+1$. Por tanto, sumando las dimensiones, se tiene que

$$(n-k) + (k+1) = n+1 > n,$$

luego al menos comparten un subespacio de dimensión 1.
Una vez visto esto, no hace falta más que tomar normas y tener en cuenta la ortonormalidad de los vectores singulares,

$$\|(A-B)\mathbf{x}\|_2^2 = \|A\mathbf{x}\|_2^2 = \|A\sum_{j=1}^{k+1} c_j \mathbf{v}_j\|_2^2 = \|\sum_{j=1}^{k+1} c_j \sigma_j \mathbf{u}_j\|_2^2 = \sum_{j=1}^{k+1} c_j^2 \sigma_j^2.$$

Por la estructura de la descomposición en valores singulares, sabemos que

$$\sigma_j \geq \sigma_{k+1} \quad \text{para} \quad j \leq k+1,$$

por lo que podemos acotar

$$\|(A-B)\mathbf{x}\|_2^2 = \sum_{j=1}^{k+1} c_j^2 \sigma_j^2 \geq \sigma_{k+1}^2 \left(\sum_{j=1}^{k+1} c_j^2 \right) = \|\mathbf{x}\|_2^2 \sigma_{k+1}^2.$$

Es decir,

$$\frac{\|(A-B)\mathbf{x}\|_2^2}{\|\mathbf{x}\|_2^2} \geq \sigma_{k+1},$$

Y tomando superiores para concluir que

$$\|A-B\|_2 = \sup_{\mathbf{x}\neq 0} \frac{\|(A-B)\mathbf{x}\|_2^2}{\|\mathbf{x}\|_2^2} \geq \sigma_{k+1}.$$

□

1.3. Dos resultados adicionales sobre la SVD

En esta sección introducimos dos resultados que nos dan más información acerca de los valores singulares de una matriz A. Además, van a ser útiles más adelante en el trabajo.
El primero es un resultado de perturbación que muestra cómo afectan a los valores singulares de una matriz los pequeños cambios que se puedan introducir en sus coeficientes.

Teorema 1.3.1. *Sean $A,\ \delta A \in \mathbb{R}^{m\times n}$. Entonces*

$$|\sigma_i(A+\delta A) - \sigma_i(A)| \leq \|\delta A\|_2 \quad \textit{para} \quad i = 1, ..., \text{mín}\{m, n\}.$$

Demostración. Consideramos las siguientes matrices

$$\widetilde{A} = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} \quad \text{y} \quad \widetilde{\delta A} = \begin{pmatrix} 0 & \delta A \\ (\delta A)^T & 0 \end{pmatrix},$$

ambas matrices de tamaño $(m+n)\times(m+n)$ y simétricas. Notemos que si $A = U\Sigma V^T$ es la descomposición en valores singulares, $\{\mathbf{u}_i\}_{i=1}^r$, $\{\mathbf{v}_i\}_{i=1}^r$ son los vectores singulares por la izquierda y por la derecha de A, entonces

$$\widetilde{A}\begin{pmatrix} \mathbf{u}_i \\ \mathbf{v}_i \end{pmatrix} = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}\begin{pmatrix} \mathbf{u}_i \\ \mathbf{v}_i \end{pmatrix} = \begin{pmatrix} A\mathbf{v}_i \\ A^T\mathbf{u}_i \end{pmatrix} = \begin{pmatrix} \sigma_i(A)\mathbf{u}_i \\ \sigma_i(A)\mathbf{v}_i \end{pmatrix} = \sigma_i(A)\begin{pmatrix} \mathbf{u}_i \\ \mathbf{v}_i \end{pmatrix}.$$

Del mismo modo,

$$\widetilde{A}\begin{pmatrix} -\mathbf{u}_i \\ \mathbf{v}_i \end{pmatrix} = \begin{pmatrix} \sigma_i(A)\mathbf{u}_i \\ -\sigma_i(A)\mathbf{v}_i \end{pmatrix} = -\sigma_i(A)\begin{pmatrix} -\mathbf{u}_i \\ \mathbf{v}_i \end{pmatrix}.$$

Es decir, los autovalores de $\widetilde{A}$ son $\pm\sigma_i(A)$ para $i = 1, \ldots, r$ y 0 con multiplicidad $m+n-2r$, donde r es el rango de A. Además, sabemos que se cumple la siguiente relación

$$-\sigma_1(A) \leq \cdots \leq -\sigma_r(A) < 0 < \sigma_r(A) \leq \cdots \leq \sigma_1(A).$$

Por tanto, si ordenamos los autovalores de $\widetilde{A}$ de forma decreciente, se tiene que $\lambda_1(\widetilde{A}) = \sigma_1(A)$ y $\lambda_{m+n}(\widetilde{A}) = -\sigma_1(A)$.
Del mismo modo se prueban relaciones análogas para los autovalores de las matrices $\widetilde{\delta A}$ y $\widetilde{A} + \widetilde{\delta A}$, es decir, $\lambda_1(\widetilde{\delta A}) = \sigma_1(\delta A)$ y $\lambda_{m+n}(\widetilde{\delta A}) = -\sigma_1(\delta A)$
El Teorema 1.8.5 de [5] aplicado a las matrices $\widetilde{A}$ y $\widetilde{A}+\widetilde{\delta A}$ afirma que sus autovalores satisfacen las siguientes desigualdades

$$\lambda_i(\widetilde{A}) + \lambda_{m+n}(\widetilde{\delta A}) \leq \lambda_i(\widetilde{A} + \widetilde{\delta A}) \leq \lambda_i(\widetilde{A}) + \lambda_1(\widetilde{\delta A}), \quad \text{para} \quad i = 1, .., m+n.$$

Expresando los autovalores de las matrices con tilde en función de los valores singulares de las matrices A, δA y $A+\delta A$, las desigualdades anteriores se reescriben como

$$\sigma_i(A) - \sigma_1(\delta A) \leq \sigma_i(A+\delta A) \leq \sigma_i(A) + \sigma_1(\delta A), \quad \text{para} \quad i = 1, .., \text{mín}\{m, n\},$$

lo cual es equivalente a

$$-\sigma_1(\delta A) \leq \sigma_i(A+\delta A) - \sigma_i(A) \leq \sigma_1(\delta A), \quad \text{para} \quad i = 1, .., \text{mín}\{m, n\}.$$

Podemos concluir entonces que

$$|\sigma_i(A+\delta A)-\sigma_i(A)| \leq \sigma_1(\delta A) = \|\delta A\|_2 \quad \text{para} \quad i=1,...,\text{mín}\{m,n\}.$$

□

El siguiente resultado proporciona una caracterización variacional de los valores singulares.

Teorema 1.3.2. *Sea $A \in \mathbb{R}^{m\times n}$ y σ_k su k-ésimo valor singular. Entonces*

$$\sigma_k = \max_{\substack{\mathbf{u} \in < \mathbf{u}_1,\ldots,\mathbf{u}_{k-1} >^{\perp} \\ \mathbf{v} \in < \mathbf{v}_1,\ldots,\mathbf{v}_{k-1} >^{\perp}}} \frac{\mathbf{u}^T A\mathbf{v}}{\|\mathbf{u}\|_2\|\mathbf{v}\|_2}, \quad k>1, \tag{1.12}$$

donde $< \mathbf{u}_1,\ldots,\mathbf{u}_{k-1} >$ y $< \mathbf{v}_1,\ldots,\mathbf{v}_{k-1} >$ son los subespacios generados por los $k-1$ primeros vectores singulares de A por la izquierda y por la derecha, $\{\mathbf{u}_1,\ldots,\mathbf{u}_{k-1}\}$ y $\{\mathbf{v}_1,\ldots,\mathbf{v}_{k-1}\}$, respectivamente.

Demostración. Hemos visto en (1.3) que podemos escribir

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T.$$

Por tanto, si tomamos $\mathbf{u} \in < \mathbf{u}_1,\ldots,\mathbf{u}_{k-1} >^{\perp}$ y $\mathbf{v} \in < \mathbf{v}_1,\ldots,\mathbf{v}_{k-1} >^{\perp}$, se tiene que

$$\mathbf{u}^T A\mathbf{v} = \sum_{i=1}^{r} \left(\sigma_i \mathbf{u}^T\mathbf{u}_i\mathbf{v}_i^T\mathbf{v}\right) = \mathbf{u}^T\left(\sum_{i=k}^{r}\sigma_i\mathbf{u}_i\mathbf{v}_i^T\right)\mathbf{v} = \mathbf{u}^T\left(A-A_{k-1}\right)\mathbf{v}, \tag{1.13}$$

donde se ha utilizado (1.3) y (1.6). Si ahora empleamos la desigualdad de Cauchy-Schwarz, nos queda

$$|\mathbf{u}^T A\mathbf{v}| \leq \|\mathbf{u}^T\|_2\|A-A_{k-1}\|_2\|\mathbf{v}\|_2 = \sigma_k\|\mathbf{u}\|_2\|\mathbf{v}\|_2,$$

es decir,

$$\sigma_k \geq \frac{|\mathbf{u}^T A\mathbf{v}|}{\|\mathbf{u}\|_2\|\mathbf{v}\|_2}, \quad \mathbf{u} \in < \mathbf{u}_1,\ldots,\mathbf{u}_{k-1} >^{\perp}, \quad \mathbf{v} \in < \mathbf{v}_1,\ldots,\mathbf{v}_{k-1} >^{\perp}.$$

Además, si en (1.13) tomamos $\mathbf{u}=\mathbf{u}_k$, $\mathbf{v}=\mathbf{v}_k$ el k-ésimo vector singular de A por la izquierda y por la derecha, respectivamente, tenemos

$$\mathbf{u}_k^T A\mathbf{v}_k = \sum_{i=1}^{r}\left(\sigma_i\mathbf{u}_k^T\mathbf{u}_i\mathbf{v}_i^T\mathbf{v}_k\right) = \sigma_k\mathbf{u}_k^T\mathbf{u}_k\mathbf{v}_k^T\mathbf{v}_k = \sigma_k,$$

lo cual concluye la demostración. □

Capítulo 2

La descomposición en valores singulares aleatorizada

Antes de nada, vamos a introducir una notación que usaremos a lo largo de este capítulo. Sea $A \in \mathbb{R}^{m\times n}$ una matriz, Denotaremos

1. Las columnas de A por $\mathbf{a}_i$ para $1 \leq i \leq n$.
2. Las filas de A por $\mathbf{a}^i$ para $1 \leq i \leq m$.
3. Los elementos de A por a_{ij} para $1 \leq i \leq m$, $1 \leq j \leq n$.
4. Dada una matriz $B \in \mathbb{R}^{n\times p}$, los elementos de la matriz producto AB los denotaremos por $(AB)_{ij}$ para $1 \leq i \leq m$, $1 \leq j \leq p$.

Además, como en esta sección también vamos a utilizar variables aleatorias, denotaremos como es habitual en la literatura su esperanza por $\mathbf{E}[X]$ y la varianza por $\mathbf{Var}[X]$, donde X es dicha variable aleatoria.

2.1. Aproximación aleatorizada del producto de matrices

Vamos a considerar un algoritmo sencillo para aproximar el producto de dos matrices de forma aleatorizada. Multiplicar matrices es un problema fundamental dentro del Álgebra Lineal, por lo que el algoritmo tiene interés en sí mismo. Pero además, esto es algo que se usa en muchos otros algoritmos, luego también va a ser útil como una herramienta auxiliar en otros problemas. El problema es el siguiente: dadas matrices $A \in \mathbb{R}^{m\times n}$ y $B \in \mathbb{R}^{n\times p}$, queremos obtener la matriz

producto AB. Sabemos que el producto usual de matrices requiere $m \times n \times p$ productos y otras tantas sumas, que se traduce en un tiempo de cálculo $O(m \times n \times p)$. La pregunta es, ¿podemos hacerlo más rápido, aunque hallemos el producto solo de manera aproximada?.
Para ello, vamos a interpretar el producto de dos matrices como una suma de matrices de rango uno, esto es

$$AB = \sum_{i=1}^{n} \mathbf{a}_i \mathbf{b}^i, \tag{2.1}$$

El algoritmo que vamos a introducir toma una muestra aleatoria de las columnas y las filas de las matrices A y B, respectivamente, y hace el producto con las matrices resultantes. Análogamente, cuando estamos tratando una suma de números, podemos tomar una muestra aleatoria de estos siguiendo cualquier distribución y vamos a obtener un estimador insesgado de dicha suma. Sin embargo, si queremos minimizar la varianza del estimador, vamos a necesitar elegir la muestra (y reescalar) en función del tamaño de los números. Para el producto de matrices va a ser igual. Comenzaremos describiendo el algoritmo que nos va a permitir aproximar el producto de dos matrices.

Algoritmo 1. Multiplicación Aleatorizada de Matrices.
Dadas matrices $A \in \mathbb{R}^{m\times n}$ y $B \in \mathbb{R}^{n\times p}$, un entero positivo $c < n$ y probabilidades $\{p_i\}_{i=1}^n$, hacemos

Para t desde 1 hasta c

1. Elegimos $i_t \in \{1, \ldots, n\}$ con probabilidad $P[i_t = k] = p_k$ en experimentos independientes e idénticamente distribuidos con reemplazamiento.
2. Hacemos $\quad \mathbf{c}_t = \dfrac{\mathbf{a}_{i_t}}{\sqrt{cp_{i_t}}} \quad \text{y} \quad \mathbf{r}^t = \dfrac{\mathbf{b}^{i_t}}{\sqrt{cp_{i_t}}}.$

Devolvemos C la matriz de tamaño $m \times c$ que tiene por columnas $\mathbf{c}_t,\ 1 \le t \le c$ y R la matriz de tamaño $c \times p$ que tiene por filas $\mathbf{r}^t,\ 1 \le t \le c.$ (2.2)

La matriz CR es una aproximación de la matriz AB. Podemos introducir ahora los siguientes resultados.

Lema 2.1.1. *Sean $A \in \mathbb{R}^{m\times n}$ y $B \in \mathbb{R}^{n\times p}$ dos matrices, C y R como en el Algoritmo 1. Entonces*

$$\mathbf{E}\left[(CR)_{ij}\right] = (AB)_{ij} \quad \text{y} \quad \mathbf{Var}\left[(CR)_{ij}\right] = \frac{1}{c}\sum_{k=1}^{n} \frac{a_{ik}^2 b_{kj}^2}{p_k} - \frac{1}{c}(AB)_{ij}^2. \tag{2.3}$$

Demostración. Fijamos i, j, y para $t = 1, \ldots, c$ definimos las variables aleatorias

$$X_t = \left(\mathbf{c}_t \mathbf{r}^t\right)_{ij} = \left(\frac{\mathbf{a}_{i_t}\mathbf{b}^{i_t}}{cp_{i_t}}\right)_{ij} = \frac{a_{ii_t} b_{i_t j}}{cp_{it}}.$$

Entonces,

$$\mathbf{E}\left[X_t\right] = \sum_{k=1}^{n} p_k X_k = \sum_{k=1}^{n} p_k \frac{a_{ik} b_{kj}}{c p_k} = \frac{1}{c} \sum_{k=1}^{n} a_{ik} b_{kj} = \frac{1}{c} \mathbf{a}^i \mathbf{b}_j = \frac{1}{c} \left(AB\right)_{ij}.$$

Además,

$$\mathbf{E}\left[X_t^2\right] = \sum_{k=1}^{n} p_k X_k^2 = \sum_{k=1}^{n} \frac{a_{ik}^2 b_{kj}^2}{c^2 p_k}.$$

Por otro lado, por definición, tenemos

$$\left(CR\right)_{ij} = \sum_{t=1}^{c} X_t,$$

luego

$$\mathbf{E}\left[\left(CR\right)_{ij}\right] = \sum_{t=1}^{c} \mathbf{E}(X_t) = \sum_{t=1}^{c} \frac{1}{c} \left(AB\right)_{ij} = \left(AB\right)_{ij}.$$

Por otro lado, como $\left(CR\right)_{ij}$ es la suma de c variables aleatorias independientes,

$$\begin{aligned}
\mathbf{Var}\left[\left(CR\right)_{ij}\right] &= \sum_{t=1}^{c} \mathbf{Var}(X_t) = \sum_{t=1}^{c} \left(\mathbf{E}\left[X_t^2\right] - \mathbf{E}\left[X_t\right]^2\right) \\
&= \sum_{t=1}^{c} \left(\sum_{k=1}^{n} \frac{a_{ik}^2 b_{kj}^2}{c^2 p_k} - \left(\frac{\left(AB\right)_{ij}}{c}\right)^2 \right) \\
&= c \sum_{k=1}^{n} \frac{a_{ik}^2 b_{kj}^2}{c^2 p_k} - \frac{\left(AB\right)_{ij}^2}{c^2} \\
&= \frac{1}{c} \sum_{k=1}^{n} \frac{a_{ik}^2 b_{kj}^2}{p_k} - \frac{\left(AB\right)_{ij}^2}{c}.
\end{aligned}$$

□

Estamos ahora en condiciones de dar una cota superior para $\mathbf{E}[\|AB - CR\|_F^2]$ y ver que el error va a depender de las probabilidades elegidas.

Lema 2.1.2. *Sean $A \in \mathbb{R}^{m\times n}$ y $B \in \mathbb{R}^{n\times p}$ dos matrices, $c \in \mathbb{Z}^+$, C y R como en el Algoritmo 1. Entonces*

$$\mathbf{E}\left[\|AB - CR\|_F^2\right] = \sum_{k=1}^{n} \frac{\|\mathbf{a}_k\|_2^2 \|\mathbf{b}^k\|_2^2}{c p_k} - \frac{1}{c}\|AB\|_F^2. \tag{2.4}$$

Además, si se eligen

$$p_k = \frac{\|\mathbf{a}_k\|_2 \|\mathbf{b}^k\|_2}{\sum_{l=1}^{n} \|\mathbf{a}_l\|_2 \|\mathbf{b}^l\|_2}, \quad 1 \le k \le n, \tag{2.5}$$

entonces

$$\mathbf{E}\left[\|AB - CR\|_F^2\right] = \frac{1}{c} \left(\sum_{k=1}^{n} \|\mathbf{a}_k\|_2 \|\mathbf{b}^k\|_2 \right)^2 - \frac{1}{c}\|AB\|_F^2.$$

Demostración. Comencemos notando que

$$\|AB - CR\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{p}\left|(AB)_{ij} - (CR)_{ij}\right|^2 = \sum_{i=1}^{m}\sum_{j=1}^{p}(AB - CR)_{ij}^2\,.$$

Además, usando el lema anterior y que $\mathbf{Var}\left[(CR)_{ij}\right] = \mathbf{E}\left[(CR)_{ij}^2\right] - \mathbf{E}\left[(CR)_{ij}\right]^2$, tenemos que

$$\begin{aligned}
\mathbf{E}\left[(AB - CR)_{ij}^2\right] &= \mathbf{E}\left[\left((AB)_{ij} - (CR)_{ij}\right)^2\right] = \mathbf{E}\left[(AB)_{ij}^2 - 2\,(AB)_{ij}\,(CR)_{ij} + (CR)_{ij}^2\right] \\
&= (AB)_{ij}^2 - 2\,(AB)_{ij}\,\mathbf{E}\left[(CR)_{ij}\right] + \mathbf{E}\left[(CR)_{ij}^2\right] \\
&= (AB)_{ij}^2 - 2\,(AB)_{ij}\,(AB)_{ij} + \mathbf{Var}\left[(CR)_{ij}\right] + \mathbf{E}\left[(CR)_{ij}\right]^2 \\
&= (AB)_{ij}^2 - 2\,(AB)_{ij}^2 + (AB)_{ij}^2 + \mathbf{Var}\left[(CR)_{ij}\right] \\
&= \mathbf{Var}\left[(CR)_{ij}\right].
\end{aligned}$$

Por tanto, podemos asegurar que

$$\mathbf{E}\left[\|AB - CR\|_F^2\right] = \sum_{i=1}^{m}\sum_{j=1}^{p}\mathbf{E}\left[(AB - CR)_{ij}^2\right] = \sum_{i=1}^{m}\sum_{j=1}^{p}\mathbf{Var}\left[(CR)_{ij}\right].$$

Siguiendo de nuevo el Lema 2.1.1, se tiene que

$$\begin{aligned}
\mathbf{E}\left[\|AB - CR\|_F^2\right] &= \sum_{i=1}^{m}\sum_{j=1}^{p}\left(\frac{1}{c}\sum_{k=1}^{n}\frac{a_{ik}^2 b_{kj}^2}{p_k} - \frac{1}{c}\,(AB)_{ij}^2\right) \\
&= \frac{1}{c}\sum_{k=1}^{n}\frac{1}{p_k}\left(\sum_{i=1}^{m}a_{ik}^2\right)\left(\sum_{j=1}^{p}b_{kj}^2\right) - \frac{1}{c}\|AB\|_F^2 \\
&= \frac{1}{c}\sum_{k=1}^{n}\frac{1}{p_k}\|\mathbf{a}_k\|_2^2\|\mathbf{b}^k\|_2^2 - \frac{1}{c}\|AB\|_F^2.
\end{aligned}$$

Si ahora tomamos p_k, $1 \le k \le n$, como en (2.5), entonces

$$\begin{aligned}
\mathbf{E}\left[\|AB - CR\|_F^2\right] &= \frac{1}{c}\sum_{k=1}^{n}\frac{1}{\frac{\|\mathbf{a}_k\|_2\|\mathbf{b}^k\|_2}{\sum_{l=1}^{n}\|\mathbf{a}_l\|_2\|\mathbf{b}^l\|_2}}\|\mathbf{a}_k\|_2^2\|\mathbf{b}^k\|_2^2 - \frac{1}{c}\|AB\|_F^2 \\
&= \frac{1}{c}\left(\sum_{k=1}^{n}\|\mathbf{a}_k\|_2\|\mathbf{b}^k\|_2\right)^2 - \frac{1}{c}\|AB\|_F^2.
\end{aligned}$$

□

A continuación enunciamos un resultado que afirma que efectivamente las probabilidades tomadas en (2.5) son las que minimizan la esperanza del error al aproximar la matriz producto AB.

Teorema 2.1.1. *Las probabilidades* $\{p_i\}_{i=1}^n$ *de la forma de* (2.5) *minimizan* $\mathbf{E}\left[\|AB - CR\|_F^2\right]$.

Demostración. Empezamos definiendo la función

$$f(p_1, \ldots, p_n) = \sum_{k=1}^{n} \frac{1}{p_k} \|\mathbf{a}_k\|_2^2 \|\mathbf{b}^k\|_2^2,$$

que caracteriza la dependencia de $\mathbf{E}\left[\|AB - CR\|_F^2\right]$ respecto de las probabilidades. Para minimizar f sujeto a $\sum_{k=1}^n p_k = 1$ introducimos el multiplicador de Lagrange λ y definimos

$$g(p_1, \ldots, p_n) = f(p_1, \ldots, p_n) + \lambda \left(\sum_{k=1}^{n} p_k - 1 \right).$$

Tenemos que pedir que se cumpla para $1 \leq i \leq n$,

$$0 = \frac{\partial g}{\partial p_i} = -\frac{1}{p_i^2} \|\mathbf{a}_i\|_2^2 \|\mathbf{b}^i\|_2^2 + \lambda,$$

por lo que

$$p_i = \frac{\|\mathbf{a}_i\|_2 \|\mathbf{b}^i\|_2}{\sqrt{\lambda}}.$$

Ahora, imponiendo la condición de que

$$\sum_{k=1}^{n} p_k = 1,$$

nos queda lo siguiente

$$1 = \sum_{k=1}^{n} \frac{\|\mathbf{a}_k\|_2 \|\mathbf{b}^k\|_2}{\sqrt{\lambda}},$$

y podemos concluir que

$$\sqrt{\lambda} = \sum_{k=1}^{n} \|\mathbf{a}_k\|_2 \|\mathbf{b}^k\|_2.$$

Además, $\dfrac{\partial^2 g}{\partial p_i^2} > 0$ para todo $i = 1, \ldots, n$, luego podemos asegurar que la función presenta un mínimo en

$$p_i = \frac{\|\mathbf{a}_i\|_2 \|\mathbf{b}^i\|_2}{\sum_{k=1}^n \|\mathbf{a}_k\|_2 \|\mathbf{b}^k\|_2}, \quad 1 \leq i \leq n.$$

□

Introducimos a continuación un resultado que proporciona una cota superior para la esperanza del error en función de la norma de Frobenius de las matrices que se quieren multiplicar y del tamaño de la muestra.

Teorema 2.1.2. *Sean $A \in \mathbb{R}^{m\times n}$ y $B \in \mathbb{R}^{n\times p}$, $c \in \mathbb{Z}^+$ tal que $1 \leq c \leq n$ y $\{p_i\}_{i=1}^n$ como en (2.5). Si construimos C y R como en el Algoritmo 1, entonces*

$$\mathbf{E}\left[||AB - CR||_F^2\right] \leq \frac{1}{c}\|A\|_F^2\|B\|_F^2.$$

Demostración. No hay más que notar que usando (2.4),

$$\begin{aligned}\mathbf{E}\left[\|AB - CR\|_F^2\right] &= \frac{1}{c}\left(\sum_{k=1}^{n}\|\mathbf{a}_k\|_2\|\mathbf{b}^k\|_2\right)^2 - \frac{1}{c}\|AB\|_F^2 \leq \frac{1}{c}\left(\sum_{k=1}^{n}\|\mathbf{a}_k\|_2\|\mathbf{b}^k\|_2\right)^2 \\ &\leq \frac{1}{c}\|A\|_F^2\|B\|_F^2,\end{aligned}$$

donde en la última desigualdad hemos usado la desigualdad de Cauchy-Schwarz. □

Este resultado afirma algo que podíamos intuir desde un principio: cuanto más grande sea la muestra de columnas de A y filas de B utilizadas, menor va a ser el error relativo en la aproximación $\dfrac{\mathbf{E}\left[||AB - CR||_F^2\right]}{\|A\|_F^2\|B\|_F^2}$.

Ahora vamos a ilustrar estos resultados con la ayuda de Matlab. Partimos de una imagen cuadrada de dimensión 916×916 de la cual obtenemos la matriz de valores numéricos asociados, A. De esta matriz calculamos la factorización LU y hacemos el producto aproximado de L y U mil veces para ver cómo de precisa es la media de dichas aproximaciones.

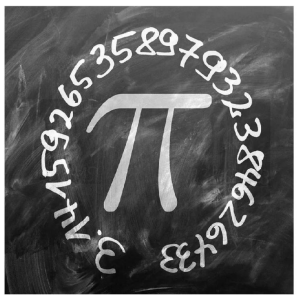

Figura 2.1: Imagen original 916×916.

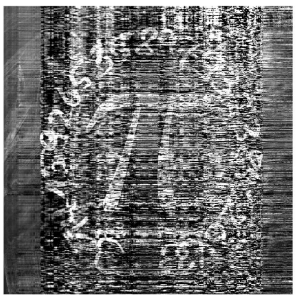

Figura 2.2: Aproximación hallada con $c = 10$.

La Figura 2.1 representa la imagen original, mientras que el resto de figuras son la media de las aproximaciones del producto de L y U para distintos valores de c. En particular, la Figura 2.2 es para $c = 10$, la Figura 2.3 toma $c = 50$ y en la Figura 2.4 $c = 100$.

Es evidente que cuando muestreamos solo con 10 columnas la aproximación no es buena, apenas somos capaces de ver nada de la imagen original. Sin embargo, en cuanto aumentamos a 50 la imagen ya es bastante más visible, aunque ciertos números pueden dar lugar a duda. Con el valor de $c = 100$ sí que se reconocen todos los dígitos.

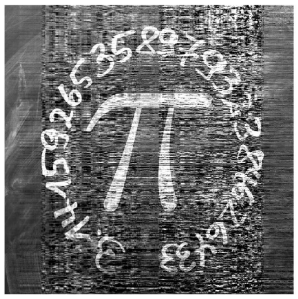

Figura 2.3: Aproximación hallada con $c = 50$

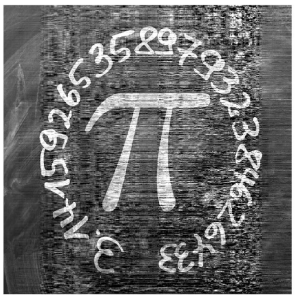

Figura 2.4: Aproximación hallada con $c = 100$.

Si seguimos aumentando el valor de c, es evidente que las aproximaciones seguirán mejorando, pero ya hemos visto que $c = 100$ es suficientemente buena en este caso.

Todo esto se ve reflejado en la Figura 2.5, que muestra el error relativo cometido $\dfrac{\mathbf{E}\left[||AB - CR||_F^2\right]}{\|A\|_F^2 \|B\|_F^2}$ frente al número de columnas c que hemos muestreado.

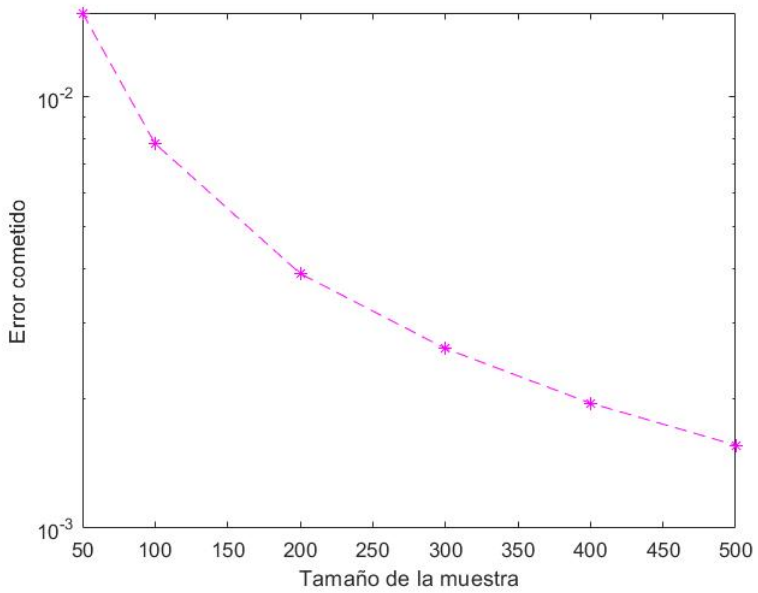

Figura 2.5: Relación entre el tamaño de la muestra y el error cometido

Observamos que al multiplicar por 2 el tamaño de la muestra, el error cometido se divide

aproximadamente entre 2. Sin embargo, con una muestra de tamaño $c = 100$ (que en este caso equivale a poco más que la décima parte del tamaño de la matriz), podemos conseguir precisión del orden de 10^{-2}, y hemos visto que es suficiente como para ser capaz de distinguir los dígitos que aparecen en la imagen. Por otro lado, es evidente que siguiendo este algoritmo, el producto aproximado requiere solo un número de operaciones del orden de $O(m \times c \times p)$, y generalmente vamos a usar $c \ll n$, lo cual es una mejora sustancial.
En [3] se pueden encontrar más detalles acerca de la multiplicación aleatorizada de matrices (ya hemos mencionado que es un problema que en sí mismo tiene mucho interés). Nuestro objetivo principal es, sin embargo, utilizar la multiplicación aleatorizada de matrices para dar un algoritmo aleatorizado para la descomposición en valores singulares de una matriz.

2.2. Un algoritmo aleatorizado para la SVD

Un análisis del tiempo de cálculo de la SVD de una matriz $A \in \mathbb{R}^{m\times n}$ nos lleva a que dicho tiempo va a ser del orden de $O(\text{mín}\{mn^2, m^2n\})$ [9]. De nuevo nos centramos en ver si somos capaces de aproximar la SVD de una matriz de un modo más rápido sin perder mucha información en el proceso.

Dada una matriz $A \in \mathbb{R}^{m\times n}$, queremos elegir ciertas columnas de A de forma que la proyección de la matriz sobre el espacio generado por dichas columnas *capte* la máxima información posible. En particular, si podemos hacer una buena aproximación de rango k de A, nos gustaría que $A \sim P_S A$, donde S es el espacio generado por las c columnas que hemos elegido de A y P_S es una proyección de A en dichas columnas. Para ello, vamos a seguir un algoritmo que se asemeja al presentado en la sección anterior.
Algoritmo 2. La SVD aleatorizada.
Dada $A \in \mathbb{R}^{m\times n}$, c, k tales que $1 \le k \le c \le n$, $\{p_i\}_{i=1}^n$ un conjunto de probabilidades,

Para t desde 1 hasta c

1. Elegimos $i_t \in \{1, \dots, n\}$ con probabilidad $P[i_t = j] = p_j$.
2. Hacemos $\mathbf{c}_t = \dfrac{\mathbf{a}_{i_t}}{\sqrt{cp_{i_t}}}$.

3. Calculamos C^TC y su SVD, véase, $C^TC = \displaystyle\sum_{t=1}^{c} \sigma_t^2(C)\mathbf{y}_t\mathbf{y}_t^T$.

4. Calculamos $\mathbf{h}_t = \dfrac{C\mathbf{y}_t}{\sigma_t(C)}$, para $t = 1, \dots, k$.

5. Devolvemos la matriz $H_k = [\mathbf{h}_1, \dots \mathbf{h}_k]$ y $\sigma_t(C)$ para $t = 1, \dots, k$. (2.6)

En esencia, estamos obteniendo los k primeros valores singulares de C y sus respectivos k vectores singulares por la izquierda, $\mathbf{h}_1, \ldots, \mathbf{h}_k$.

Queremos poder afirmar que de alguna forma C es similar a A. Puede parecer complicado puesto que no son matrices con las mismas dimensiones, pero los subespacios generados por sus columnas viven ambos en $\mathbb{R}^m$, así que diremos que son similares si sus vectores singulares por la izquierda generan subespacios parecidos, o lo que es lo mismo, $AA^T \sim CC^T$.
Antes de entrar en los resultados acerca de la calidad de la aproximación, veamos que efectivamente el tiempo de cálculo va a ser lineal.

1. Si trabajamos con probabilidades proporcionales a la norma euclídea de las columnas de A (como hacíamos en el Algoritmo 1), tenemos que hacer una lectura de la matriz. Además, es necesario elegir y guardar los índices de las columnas, lo cual supone tiempo de cálculo y espacio adicional del orden de $O(c)$.

2. Una vez elegidos los índices, es necesaria otra lectura y a mayores se eligen las columnas y se construye la matriz C, que corresponde a un tiempo de cálculo y espacio adicional del orden de $O(m \times c)$.

3. Dada la matriz C, el cálculo de C^TC requiere $O(m \times c^2)$ tiempo de computación y espacio adicional. Ahora, para el cálculo de su SVD es necesario un tiempo de computación y espacio adicional del orden de $O(c^3)$.

4. Dada la SVD de C^TC, para calcular H_k son necesarias k multiplicaciones de matriz por vector, esto es, $O(m \times c \times k)$ tiempo y espacio adicional.

5. En resumen, podemos considerar $c,\ k = O(1)$, de modo que solo necesitamos espacio y tiempo de cálculo del orden de $O(m)$. Es decir, el tiempo de cálculo de este algoritmo es lineal en la dimensión de la matriz.

Damos ahora un resultado sobre la calidad de la aproximación en la norma de Frobenius.

Teorema 2.2.1. *Sea $A \in \mathbb{R}^{m\times n}$ y sea H_k la matriz obtenida tras aplicar el Algoritmo 2. Entonces*

$$\|A - H_k H_k^T A\|_F^2 \leq \|A - A_k\|_F^2 + 2\sqrt{k}\|AA^T - CC^T\|_F, \tag{2.7}$$

donde A_k es la matriz definida en el Teorema 1.2.1.

Demostración. Antes de nada, recordemos que para matrices $X, Y \in \mathbb{R}^{m\times n}$, $\|X\|_F^2 = tr(X^TX)$, $tr(X+Y) = tr(X) + tr(Y)$ y que, por cómo la hemos construido, $H_k^T H_k = I_k$, con I_k la matriz

identidad de tamaño k. Entonces

$$\begin{aligned}\|A - H_k H_k^T A\|_F^2 &= tr\left[\left(A - H_k H_k^T A\right)^T \left(A - H_k H_k^T A\right)\right] \\ &= tr\left[A^T A - 2A^T H_k H_k^T A + A^T H_k H_k^T H_k H_k^T A\right] \\ &= tr(A^T A) - tr(2A^T H_k H_k^T A - A^T H_k H_k^T A) = \\ &= \|A\|_F^2 - tr(A^T H_k H_k^T A) = \\ &= \|A\|_F^2 - \|A^T H_k\|_F^2. \end{aligned} \tag{2.8}$$

Además, notemos que $A^T H_k = \left(A^T \mathbf{h}_1 | \dots | A^T \mathbf{h}_k\right)$, donde $\mathbf{h}_1, \dots, \mathbf{h}_k$ son las columnas de H_k, de modo que

$$\|A^T H_k\|_F^2 = \sum_{j=1}^{k} \|A^T \mathbf{h}_j\|_F^2 = \sum_{j=1}^{k} \|A^T \mathbf{h}_j\|_2^2, \tag{2.9}$$

donde la última igualdad se da porque la norma de Frobenius de un vector no es más que su norma euclídea. De este modo, si tomamos $\mathbf{1} = (1, \dots, 1) \in \mathbb{R}^k$,
$\mathbf{v} = \left(\|A^T \mathbf{h}_1\|_2^2 - \sigma_1^2(C), \dots, \|A^T \mathbf{h}_k\|_2^2 - \sigma_k^2(C)\right) \in \mathbb{R}^k$, tenemos que, haciendo uso de la desigualdad de Cauchy-Schwarz,

$$\begin{aligned}\left|\|A^T H_k\|_F^2 - \sum_{j=1}^{k} \sigma_j^2(C)\right| &= \left|\sum_{j=1}^{k} \left(\|A^T \mathbf{h}_j\|_2^2 - \sigma_j^2(C)\right)\right| = |\langle \mathbf{1}, \mathbf{v}\rangle| \\ &\leq \|\mathbf{1}\|_2 \|\mathbf{v}\|_2 = \sqrt{k} \left(\sum_{j=1}^{k} \left(\|A^T \mathbf{h}_j\|_2^2 - \sigma_j^2(C)\right)^2\right)^{\frac{1}{2}}. \end{aligned} \tag{2.10}$$

Además, siguiendo la definición de $\mathbf{h}_j$ en el Algoritmo 2, tenemos que

$$C^T \mathbf{h}_j = \frac{C^T C \mathbf{y}_j}{\sigma_j(C)} = \frac{\left(\sum_{i=1}^{c} \sigma_i^2(C) \mathbf{y}_i \mathbf{y}_i^T\right) \mathbf{y}_j}{\sigma_j(C)} = \frac{\sigma_j^2(C) \mathbf{y}_j \mathbf{y}_j^T \mathbf{y}_j}{\sigma_j(C)} = \sigma_j(C) \mathbf{y}_j, \tag{2.11}$$

donde en las últimas dos igualdades hemos usado que como $\{\mathbf{y}_j\}_{j=1}^{c}$ son los vectores singulares de $C^T C$, entonces son un sistema ortonormal. Por tanto podemos deducir que $\|C^T \mathbf{h}_t\|_2^2 = \sigma_t^2(C)$ y, juntando (2.10) y (2.11), tenemos

$$\begin{aligned}\left|\|A^T H_k\|_F^2 - \sum_{j=1}^{k} \sigma_j^2(C)\right| &\leq \sqrt{k} \left(\sum_{j=1}^{k} \left(\|A^T \mathbf{h}_j\|_2^2 - \|C^T \mathbf{h}_j\|_2^2\right)^2\right)^{\frac{1}{2}} \\ &= \sqrt{k} \left(\sum_{j=1}^{k} \left(\left(A^T \mathbf{h}_j\right)^T \left(A^T \mathbf{h}_j\right) - \left(C^T \mathbf{h}_j\right)^T \left(C^T \mathbf{h}_j\right)\right)^2\right)^{\frac{1}{2}} \\ &= \sqrt{k} \left(\sum_{j=1}^{k} \left(\mathbf{h}_j^T \left(AA^T - CC^T\right) \mathbf{h}_j\right)^2\right)^{\frac{1}{2}}. \end{aligned} \tag{2.12}$$

Por otro lado, sea $H = \left[H_k|\widetilde{H}\right]$ una matriz ortogonal cuyas primeras k columnas son $\{\mathbf{h}_j\}_{j=1}^k$. Entonces podemos escribir $AA^T - CC^T$ como

$$AA^T - CC^T = HBH^T - HDH^T = HEH^T, \tag{2.13}$$

para ciertas matrices B, D y $E = B - D$, de modo que $\|AA^T - CC^T\|_F = \|HEH^T\|_F = \|E\|_F$, por ser H ortogonal. Si combinamos (2.12) y (2.13) nos queda

$$\begin{aligned}\sqrt{k}\left(\sum_{j=1}^{k}\left(\mathbf{h}_j^T\left(AA^T - CC^T\right)\mathbf{h}_j\right)^2\right)^{\frac{1}{2}} &= \sqrt{k}\left(\sum_{j=1}^{k}\left(\mathbf{h}_j^T\left(HEH^T\right)\mathbf{h}_j\right)^2\right)^{\frac{1}{2}} \\ &= \sqrt{k}\left(\sum_{j=1}^{k}\left(\mathbf{e}_j^T E\mathbf{e}_j\right)^2\right)^{\frac{1}{2}} = \sqrt{k}\left(\sum_{j=1}^{k}(E)_{jj}^2\right)^{\frac{1}{2}} \\ &\leq \sqrt{k}\|E\|_F^2 = \sqrt{k}\|AA^T - CC^T\|_F^2, \end{aligned} \tag{2.14}$$

donde $\mathbf{e}_j$ es el j-ésimo elemento de la base ordenada canónica de $\mathbb{R}^m$.
Por otro lado, siguiendo un razonamiento muy similar al utilizado para obtener (2.10), tenemos que aplicando la desigualdad de Cauchy-Schwarz,

$$\left|\sum_{j=1}^{k}\sigma_j^2(C) - \sum_{j=1}^{k}\sigma_j^2(A)\right| \leq \sqrt{k}\left(\sum_{j=1}^{k}\left(\sigma_j^2\left(C\right) - \sigma_j^2\left(A\right)\right)^2\right)^{\frac{1}{2}}.$$

Usando ahora que $\sigma_j(XX^T) = \sigma_j^2(X)$, $1 \leq j \leq n$ para toda $X \in \mathbb{R}^{m\times n}$ y el lema (1.3.1), tenemos

$$\begin{aligned}\left|\sum_{j=1}^{k}\sigma_j^2(C) - \sum_{j=1}^{k}\sigma_j^2(A)\right| &\leq \sqrt{k}\left(\sum_{j=1}^{k}\left(\sigma_j\left(CC^T\right) - \sigma_j\left(AA^T\right)\right)^2\right)^{\frac{1}{2}} \\ &\leq \sqrt{k}\left(\sum_{j=1}^{m}\left(\sigma_j\left(CC^T\right) - \sigma_j\left(AA^T\right)\right)^2\right)^{\frac{1}{2}} \\ &\leq \sqrt{k}\|CC^T - AA^T\|_F. \end{aligned} \tag{2.15}$$

Si combinamos los resultados (2.12), (2.14) y (2.15), obtenemos

$$\begin{aligned}\left|\|A^T H_k\|_F^2 - \sum_{j=1}^{k}\sigma_j^2(A)\right| &\leq \left|\|A^T H_k\|_F^2 - \sum_{j=1}^{k}\sigma_j^2(C)\right| + \left|\sum_{j=1}^{k}\sigma_j^2(C) - \sum_{j=1}^{k}\sigma_j^2(A)\right| \\ &\leq 2\sqrt{k}\|AA^T - CC^T\|_F. \end{aligned} \tag{2.16}$$

Y por tanto no queda más que combinar (2.8) y (2.16) y utilizar el Lema 1.2.1 para concluir

$$\begin{aligned}\|A - H_k H_k^T A\|_F^2 - \|A - A_k\|_F^2 &= \|A\|_F^2 - \|A^T H_k\|_F^2 - \|A - A_k\|_F^2 \\ &= \sum_{j=1}^{m} \sigma_j^2(A) - \sum_{j=k+1}^{m} \sigma_j^2(A) - \|A^T H_k\|_F^2 \\ &= \sum_{j=1}^{k} \sigma_j^2(A) - \|A^T H_k\|_F^2 \\ &\leq 2\sqrt{k}\|AA^T - CC^T\|_F.\end{aligned}$$

□

Este teorema nos dice que el error cometido por la aproximación de rango k proporcionada por el Algoritmo 2 (más allá del que cometemos con la mejor aproximación, que sabemos que es la proporcionada por la SVD) se puede relacionar con el error cometido al aproximar el producto de AA^T por CC^T. Por tanto, si podemos conseguir un error pequeño en dicha aproximación, garantizaremos una buena aproximación de rango bajo a la matriz A.

Introducimos ahora un resultado muy similar para la norma espectral. Notemos que en este caso el factor $\sqrt{k}$ no va a aparecer.

Teorema 2.2.2. *Sea $A \in \mathbb{R}^{m\times n}$ y sea H_k la matriz obtenida tras aplicar el Algoritmo 2. Entonces*

$$\|A - H_k H_k^T A\|_2^2 \leq \|A - A_k\|_2^2 + 2\|AA^T - CC^T\|_2, \tag{2.17}$$

Demostración. Sea $\mathcal{H}_k$ el subespacio generado por $\{\mathbf{h}_1, \ldots, \mathbf{h}_k\}$ y sea $\mathcal{H}_{m-k}$ el complemento ortogonal de dimensión $m-k$ de $\mathcal{H}_k$ en $\mathbb{R}^m$. Sea $\mathbf{x} \in \mathbb{R}^m$ tal que $\mathbf{x} = \alpha\mathbf{y} + \beta\mathbf{z}$ con $\mathbf{y} \in \mathcal{H}_k$, $\mathbf{z} \in \mathcal{H}_{m-k}$ y $\alpha^2 + \beta^2 = 1$.
Notemos además que como los valores singulares de una matriz X cualquiera y de X^T son los mismos, siguiendo el Lema 1.2.1, deducimos que $\|X\|_2^2 = \|X^T\|_2^2$. Entonces

$$\begin{aligned}\|A - H_k H_k^T A\|_2 = \|(A - H_k H_k^T A)^T\|_2 &= \max_{\mathbf{x}\in\mathbb{R}^m, \|\mathbf{x}\|_2} \|(A - H_k H_k^T A)^T \mathbf{x}\|_2 \\ &= \max_{\mathbf{x}\in\mathbb{R}^m, \|\mathbf{x}\|_2} \|\mathbf{x}^T(A - H_k H_k^T A)\|_2 \\ &= \max_{\mathbf{y}\in\mathcal{H}_k, \|\mathbf{y}\|_2, \mathbf{z}\in\mathcal{H}_{m-k}, \|\mathbf{z}\|_2, \alpha^2+\beta^2=1} \| \left(\alpha\mathbf{y} + \beta\mathbf{z}\right)^T (A - H_k H_k^T A)\|_2 \\ &\leq \max_{\mathbf{y}\in\mathcal{H}_k, \|\mathbf{y}\|_2} \|\mathbf{y}^T(A - H_k H_k^T A)\|_2 + \max_{\mathbf{z}\in\mathcal{H}_{m-k}, \|\mathbf{z}\|_2} \|\mathbf{z}^T(A - H_k H_k^T A)\|_2, \end{aligned} \tag{2.18}$$

donde la última desigualdad se debe a que $\alpha,\ \beta \leq 1$. Por otro lado, como $\mathbf{y} \in \mathcal{H}_k$ y $\mathbf{z} \in \mathcal{H}_{m-k}$, se tiene que

$$\mathbf{y}^T H_k H_k^T = \mathbf{y}^T \quad \text{y} \quad \mathbf{z}^T H_k H_k^T = \mathbf{0}, \tag{2.19}$$

luego combinando (2.18) y (2.19), tenemos

$$\begin{aligned}\|A - H_k H_k^T A\|_2 &\leq \max_{\mathbf{y}\in\mathcal{H}_k,\|\mathbf{y}\|_2} \|\mathbf{y}^T A - \mathbf{y}^T A\|_2 + \max_{\mathbf{z}\in\mathcal{H}_{m-k},\|\mathbf{z}\|_2} \|\mathbf{z}^T A\|_2 \\ &= \max_{\mathbf{z}\in\mathcal{H}_{m-k},\|\mathbf{z}\|_2} \|\mathbf{z}^T A\|_2. \end{aligned} \tag{2.20}$$

A continuación acotamos (2.20). Por un lado,

$$\|\mathbf{z}^T A\|_2 = \mathbf{z}^T A A^T \mathbf{z} = \mathbf{z}^T C C^T \mathbf{z} + \mathbf{z}^T \left(AA^T - CC^T\right) \mathbf{z}. \tag{2.21}$$

Por otro lado, siguiendo el Teorema (1.3.2) aplicado a la matriz CC^T, tenemos que

$$\sigma_{k+1}(CC^T) = \max_{\mathbf{z}\in\mathcal{H}_{m-k}} \mathbf{z}^T C C^T \mathbf{z},$$

pero sabemos que $\sigma_{k+1}(CC^T) = \sigma_{k+1}^2(C)$. Por tanto, siguiendo (2.21),

$$\|\mathbf{z}^T A\|_2 \leq \sigma_{k+1}^2(C) + \|AA^T - CC^T\|_2. \tag{2.22}$$

Ahora, aplicando el Lema 1.3.1 a los valores singulares de AA^T y CC^T, tenemos

$$\left|\sigma_{k+1}(AA^T) - \sigma_{k+1}(CC^T)\right| \leq \|AA^T - CC^T\|_2,$$

y, por tanto,

$$\sigma_{k+1}(CC^T) \leq \sigma_{k+1}(AA^T) + \|AA^T - CC^T\|_2. \tag{2.23}$$

Si ahora juntamos (2.22) y (2.23), llegamos a

$$\|\mathbf{z}^T A\|_2 \leq \sigma_{k+1}(AA^T) + 2\|A - A_k\|_2 = \|AA^T - CC^T\|_2 + 2\|AA^T - CC^T\|_2. \tag{2.24}$$

Finalmente, no hay más que combinar (2.20) y (2.24) para concluir (2.17) y finalizar la demostración del teorema. □

Podemos observar que ambos teoremas son ciertos independientemente de las probabilidades elegidas. Como además $\|A - A_k\|_p$ es una propiedad intrínseca de la matriz A tanto en el caso de la norma espectral como en el de la de Frobenius, las probabilidades solo van a jugar un papel en el error cometido a la hora de aproximar el producto AA^T. Y ya hemos visto que las probabilidades óptimas para esa aproximación son de la forma de (2.5). Además, también hemos comprobado que cuanto mayor sea la muestra, es decir, cuanto más grande sea c, mejor va a ser la aproximación del producto, por tanto esperamos que este comportamiento se repita.

Vamos a visualizar la calidad de la aproximación con la ayuda de Matlab. Para ello, a partir de la matriz A de valores numéricos asociada a una imagen de dimensiones 960×720, hemos

hallado H_k para distintos valores de k y c y vamos a ver qué imagen obtenemos al hacer $H_k H_k^T A$. Además, podemos comparar los resultados con las aproximaciones de rango k que proporciona la SVD exacta, es decir, con A_k como en (1.6).

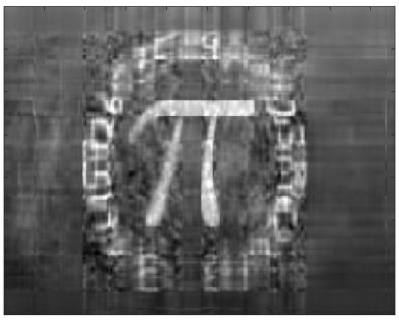

Figura 2.6: Aproximación A_k con $k = 10$.

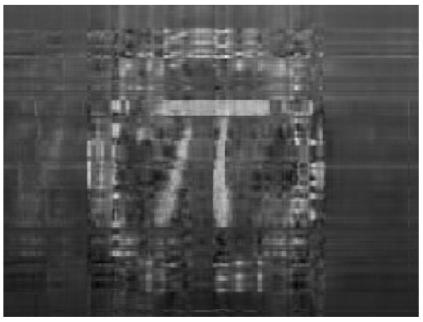

Figura 2.7: Aproximación con $c = 50$, $k = 10$.

Como la aproximación A_{10} es mala (Figura 2.6), es lógico que si hacemos el algoritmo aleatorizado con $c = 50$ y $k = 10$ la aproximación tampoco sea buena (Figura 2.7).
Si ahora tomamos A_{50}, ya obtenemos una aproximación mucho mejor en la que somos capaces de apreciar con claridad todos los dígitos (Figura 2.8), y ocurre lo mismo con la aproximación aleatorizada para $c = 100$ y $k = 50$ (Figura 2.9).

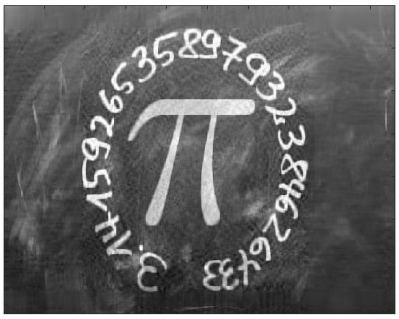

Figura 2.8: Aproximación A_k con $k = 50$.

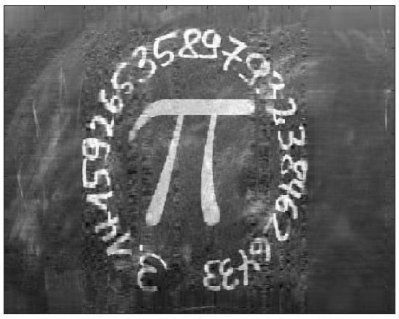

Figura 2.9: Aproximación con $c = 100$, $k = 50$.

Es evidente que la aproximación aleatorizada no es igual de buena que la exacta, pero el coste computacional es mucho menor y en este caso somos capaces de ver el contenido de la imagen sin problema. En general, ajustaremos el valor de c y k según si valoramos más la precisión o la eficiencia.

Capítulo 3

Aplicaciones

En este capítulo presentamos dos aplicaciones en las que se utiliza la descomposición en valores singulares, una del ámbito de tratamiento de imágenes y otra relacionada con el análisis semántico latente. En ambos ejemplos se utilizará la SVD exacta y en el primero de ellos también haremos uso de la aproximación aleatorizada a la SVD descrita en la Sección 2.2.

3.1. Incorporación de una marca de agua a una imagen.

3.1.1. El problema de la propiedad legítima

Uno de los problemas que surgen hoy en día con la gran facilidad de acceso que tenemos a cualquier tipo de información es que terceras partes puedan hacer uso fraudulento e incluso robar contenido con derechos de autor y obtener beneficio como si fuera propio. Es por eso que en los últimos años se han propuesto distintas técnicas para incorporar marcas de agua digitales que permitan garantizar los derechos de autor, y una de dichas técnicas es la descomposición en valores singulares.
Para que una marca de agua sea de utilidad, debe satisfacer unos requisitos básicos:

1. Imperceptibilidad: no deberíamos ser capaces de distinguir a simple vista si un documento lleva o no marca de agua.

2. Confianza: un diseño de marca de agua debería garantizar que es imposible generar marcas de agua falsas y debe permitir aportar pruebas fiables para proteger la propiedad legítima del material marcado.

3. Robustez: dado un documento con marca de agua, un tercero sin derechos sobre dicho documento no debería ser capaz de deshacerse de la marca de agua sin hacer inservible

el documento que resulta tras este proceso.

El principal objetivo de las marcas de agua es proteger los derechos de autor. Sin embargo, muchos de los diseños existentes permiten manipular de forma sencilla la imagen con marca de agua y reclamarla como propia. Algunos procedimientos necesitan la imagen original para poder verificar la propiedad, pero incluso en estas condiciones a veces siguen surgiendo problemas. Este tipo de diseños que permiten la creación de *falsificaciones originales* se denominan invertibles. Craver y sus coautores introducen en [2] el concepto de no invertibilidad. De manera informal, esto significa que es computacionalmente imposible para un atacante encontrar una imagen falsa y una marca de agua tales que juntas sean idénticas a la imagen con marca de agua original. Daremos ahora una definición más precisa.
Sea A la matriz original y W la marca de agua usada por el autor para obtener la imagen con marca de agua A_W. Escribimos entonces

$$A_W = E(A, W), \tag{3.1}$$

donde E representa el algoritmo de implementación de marca de agua. Si un atacante que tiene acceso a la imagen A_W publicada sin conocer A, crea una marca de agua falsificada W_F y una imagen falsa A_F que cumplen

$$A_W = E(A_F, W_F), \tag{3.2}$$

entonces puede usar A_F como imagen original para reclamar la propiedad de A_W.
Si (3.2) se sostiene, el diseño se denomina invertible. De lo contrario, se denomina no invertible. Las ecuaciones (3.1) y (3.2) son las ecuaciones básicas para definir la no invertibilidad.

3.1.2. Esquemas de marca de agua basados en la SVD

Antes de introducir algunos procedimientos para incorporar una marca de agua a una imagen, revisamos algunas de las propiedades más importantes de la descomposición en valores singulares desde el punto de vista del tratamiento de imágenes:

1. Los valores singulares son muy estables, es decir, pequeñas perturbaciones en la imagen se traducen en pequeñas perturbaciones en los valores singulares de la matriz asociada (ver Teorema 3.1.1).

2. Los valores singulares representan propiedades algebraicas intrínsecas de la imagen [8].

3. No es necesario que la imagen sea cuadrada.

El esquema de Liu y Tan

Liu y Tan proponen en [8] el siguiente procedimiento para incorporar una marca de agua a una imagen dada. Partimos de la matriz $A \in \mathbb{R}^{m\times n}$ asociada a la imagen y hallamos su descomposición en valores singulares, $A = U\Sigma V^T$. A continuación, si $W \in \mathbb{R}^{m\times n}$ es la matriz asociada a la marca de agua, añadimos W escalada por un parámetro real a a la matriz diagonal Σ, obteniendo $\Sigma + aW$, y hallamos a continuación la descomposición en valores singulares de $\Sigma + aW = U_W\Sigma_W V_W^T$. La imagen con su marca de agua será la asociada a la matriz $A_W = U\Sigma_W V^T$. Siguiendo la notación de [8] los pasos serían los siguientes:

$$A \to U\Sigma V^T, \tag{3.3}$$

$$\Sigma + aW \to U_W\Sigma_W V_W^T, \tag{3.4}$$

$$A_W \leftarrow U\Sigma_W V^T. \tag{3.5}$$

Notamos que los pasos (3.3) y (3.4) requieren la SVD de las matrices A y $\Sigma + aW$. En la detección de marcas de agua, conocidas U_W, Σ, V_W, a y la imagen posiblemente distorsionada A_W^*, se puede extraer una marca de agua corrupta W^* invirtiendo los pasos anteriores:

$$\begin{aligned} A_W^* &\to U^*\Sigma_W \left(V^*\right)^T, \\ D^* &\leftarrow U_W\Sigma_W V_W^T, \\ W^* &\leftarrow \frac{1}{a}\left(D^* - \Sigma\right). \end{aligned} \tag{3.6}$$

De nuevo, es necesaria la SVD en el primer paso de (3.6).
Probamos a continuación un resultado acerca de la imperceptibilidad de la marca de agua descrita previamente.

Teorema 3.1.1. *Sean A, W y A_W como en (3.3)-(3.5). Entonces los valores singulares de la matriz original y de la matriz con la marca de agua satisfacen*

$$|\sigma_i(A_W) - \sigma_i(A)| \leq a\|W\|_2 \quad \textit{para} \quad i = 1, ..., \text{mín}\{m, n\}. \tag{3.7}$$

Demostración. No hay más que ver que, puesto que A y Σ (respectivamente A_W y Σ_W comparten los valores singulares,

$$|\sigma_i(A_W) - \sigma_i(A)| = |\sigma_i(\Sigma_W) - \sigma_i(\Sigma)| = |\sigma_i(\Sigma + aW) - \sigma_i(\Sigma)| \leq a\|W\|_2,$$

donde la última desigualdad la deducimos del Teorema 1.3.1. □

El Teorema 3.1.1 indica que podemos usar a y $\|W\|_2$ para determinar la diferencia entre A y A_W. Por tanto vamos a poder ajustar la norma espectral de la marca de agua para tener más o

menos imperceptibilidad. Al final, lo más fácil va a ser fijar W y reducir el valor del escalar a. Desde el punto de vista de la imperceptibilidad, cuanto más pequeña sea $a\|W\|_2$, más parecida va a ser la imagen marcada a la original.
Liu y Tan propusieron este modelo en el año 2002, bajo la premisa de que era no invertible [8]. Sin embargo, Zhang y Li demostraron en 2005 [12] que no solo era un diseño invertible, sino que además se puede obtener cualquier marca de agua deseada mediante el proceso de extracción (3.6). Esto se debe a que en (3.4), como Σ es una matriz diagonal, las matrices U_W y V_W representan subespacios muy similares a los que generaría la SVD de W. Además, sabemos que la SVD retiene la mayor parte de la información de una imagen (como hemos mencionado en la Observación 1.1.2). Entonces, en la extracción, da igual la matriz Σ^* que usemos, la matriz final D^* está en el mismo subespacio definido por $\Sigma + aW$. En otras palabras, (3.6) *estampa* la información de la marca de agua W en D^*, independientemente de A_W y de la marca de agua original W. Más adelante incluimos un ejemplo para ilustrar este hecho.

El esquema de Jain, Arora y Panigrahi.

Jain y colaboradores [6] propusieron una mejora al esquema de Liu y Tan, buscando solventar los fallos que este último presentaba tanto en la robustez como en la confianza.
La solución que los autores proponen es que, como la mayoría de información reside en los vectores singulares, solo vamos a usar los vectores singulares por la izquierda para la extracción de la marca de agua. El esquema es el siguiente:
Dadas $A \in \mathbb{R}^{m\times n}$ la matriz asociada a la imagen original, $a \in \mathbb{R}$ el factor de escala y $W \in \mathbb{R}^{m\times n}$ la matriz asociada a la marca de agua, hacemos la SVD de las matrices A y W, y se construye la matriz de las componentes principales $A_{W,a}$.

$$\begin{aligned} A &\to U\Sigma V^T, \\ W &\to U_W \Sigma_W V_W^T, \\ A_{W,a} &= U_W \Sigma_W. \end{aligned} \tag{3.8}$$

A continuación, se forma Σ_1 incorporando la matriz $A_{W,a}$ a Σ y a partir de Σ_1 obtenemos la imagen con marca de agua,

$$\begin{aligned} \Sigma_1 &= \Sigma + aA_{W,a}, \\ A_W &\leftarrow U\Sigma_1 V^T. \end{aligned} \tag{3.9}$$

Si ahora A_W^* es una imagen con marca de agua posiblemente distorsionada, para recuperar la

marca de agua original, dadas A (y por tanto U y V), V_W y a, seguimos los tres pasos

$$\begin{aligned} A_W^* - A &\to A_1. \\ \frac{1}{a}\left(U^T A_1 V\right) &\to A_{W,a}^*. \\ A_{W,a}^* V_W^T &\to W^*. \end{aligned} \tag{3.10}$$

Podemos introducir un resultado acerca de la precisión de la marca de agua extraída con (3.10).

Teorema 3.1.2. *Sean A_W como en (3.9), A_W^* la imagen con marca de agua distorsionada y W^* como en (3.10). Entonces*

$$\|W - W^*\|_F = \frac{\|A_W^* - A_W\|_F}{|a|}. \tag{3.11}$$

Demostración. Consideremos $A = U\Sigma V^T$ y $W = U_W \Sigma_W V_W^T$ las descomposiciones en valores singulares de A y W, respectivamente. Si extraemos la marca de agua distorsionada de la matriz perturbada A_W^* siguiendo (3.9), hacemos $P = A_W^* - A_W$ y utilizamos (3.8), tenemos

$$\begin{aligned} W^* &= \frac{1}{a}\left(U^T\left(A_W^* - A\right)V\right)V_W^T = \frac{1}{a}U^T\left(A_W - A + P\right)VV_W^T \\ &= \frac{1}{a}U^T\left(U\left(\Sigma + aU_W\Sigma_W\right)V^T - A + P\right)VV_W^T \\ &= \frac{1}{a}U^T\left(A + aUU_W\Sigma_W V^T - A + P\right)VV_W^T = \frac{1}{a}U^T\left(aUU_W\Sigma_W V^T + P\right)VV_W^T \\ &= U_W\Sigma_W V_W^T + \frac{1}{a}U^T PVV_W^T \\ &= W + \frac{1}{a}U^T PVV_W^T. \end{aligned}$$

Si ahora calculamos el error en la marca de agua medido en la norma de Frobenius, nos queda

$$\|W^* - W\|_F = \|\frac{1}{a}U^T PVV_W^T\|_F = \frac{\|P\|_F}{|a|} = \frac{\|A_W^* - A_W\|_F}{|a|},$$

ya que las matrices U, V, V_W son ortogonales y $P = A_W^* - A_W$. □

3.1.3. Una ilustración numérica

Para mostrar el funcionamiento de los esquemas descritos en la Sección 3.1.2 consideramos la matriz A asociada a la imagen mostrada en la Figura 3.1 y la marca de agua W representada en la Figura 3.2. Hemos implementado en Matlab el algoritmo de Jain y sus colaboradores, (3.8)-(3.9), y mostramos en las Figuras 3.3 y 3.4 las imágenes con la marca de agua incorporada cuando $a = 0{,}1$ y $a = \frac{1}{255}$ (tomado de [12]), respectivamente.

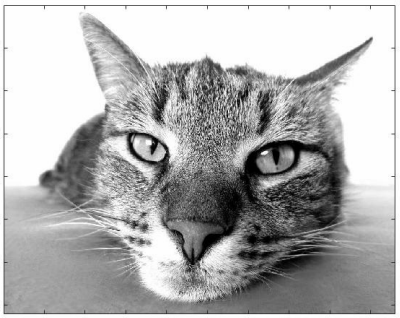

Figura 3.1: Imagen original

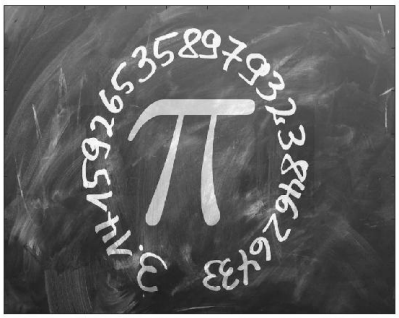

Figura 3.2: Marca de agua

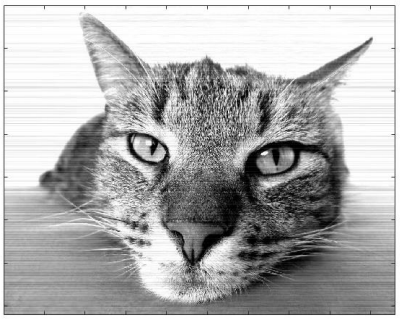

Figura 3.3: Imagen con marca de agua $a = 0{,}1$.

Figura 3.4: Imagen con marca de agua $a = \frac{1}{255}$.

Sí que podemos observar cómo para el valor de $a = 0{,}1$ la marca de agua es bastante perceptible en la imagen, pero en la Figura 3.4 ya no se aprecia apenas diferencia con la imagen original de la Figura 3.1.

Ahora perturbamos la matriz de la imagen con marca de agua sumándole otra matriz aleatoria B generada siguiendo una distribución uniforme con un factor de escala $b = 0{,}1$, es decir, $A_W^* = A_W + bB$ (donde A_W es la imagen con la marca de agua de la Figura 3.4), obtenemos lo siguiente

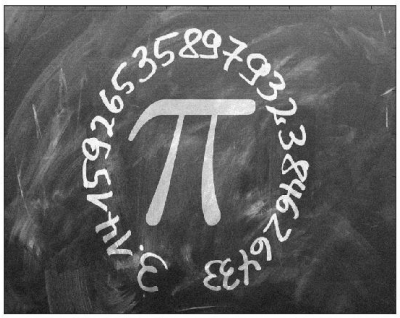

Figura 3.5: Marca de agua real extraída.

Es decir, hemos sido capaces de obtener la marca de agua implementada en la Figura 3.4 sin problema.

A continuación, en las Figuras 3.6 y 3.7 podemos ver representada tanto la perceptibilidad relativa de la marca de agua, $\frac{\|A-A_W\|_F}{\|A\|_F}$, como el error relativo cometido en la extracción de dicha marca, $\frac{\|W-W^*\|_F}{\|W\|_F}$, ambos como función del parámetro a. En color azul se ha representado la información para el método de Liu y Tan y en color magenta la correspondiente al método de Jain.

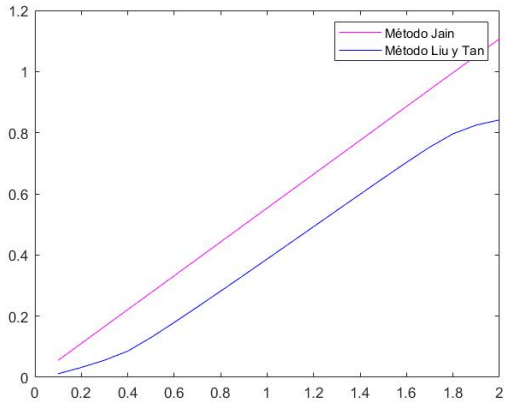

Figura 3.6: Perceptibilidad

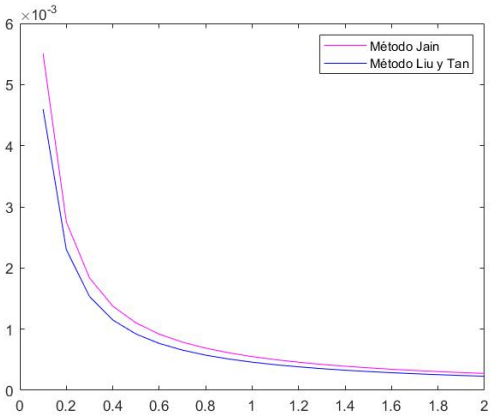

Figura 3.7: Error en la extracción.

Estas dos figuras ilustran perfectamente las afirmaciones que se hacen en los Teoremas 3.1.1 y 3.1.2. Cuanto mayor sea el valor de a, la marca de agua va a ser más perceptible, pero también vamos a poder recuperarla con mayor precisión. También observamos que con el método de Liu y Tan la perceptibilidad es menor, aunque ya hemos visto que aquel esquema tenía otras carencias.

Ahora, vamos a ver ćomo afecta a la extracción de la marca de agua el hecho de partir de una

imagen perturbada A_W^* que sea una aproximación de rango k a la imagen con la marca de agua incorporada. Para ello, tomaremos como A_W^* tanto la mejor aproximación $A_{W,k}$ a partir de la SVD exacta como la aproximación de rango k que obtenemos utilizando la SVD aleatorizada. Las siguientes figuras representan la marca de agua extraída a partir de la aproximación de rango k de la imagen con marca de agua mostrada en la Figura 3.3 ($a = 0{,}1$) cuando $A_{W,k}$ se calcula con la SVD exacta (Figura 3.8) o con la SVD aleatorizada (Figura 3.9), para $k = 100$.

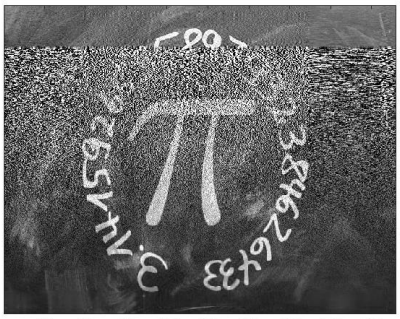

Figura 3.8: SVD exacta, $k = 100$.

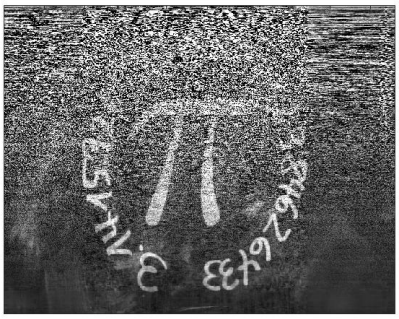

Figura 3.9: SVD aleatorizada, $c = 200$, $k = 100$.

Observamos que la calidad no es suficientemente buena en ninguno de los dos casos, aunque en la Figura 3.8 sí que es posible leer la mayoría de los dígitos.
Hemos repetido el mismo experimento con distintos valores de a y variando también el rango de la aproximación a la imagen con marca de agua $A_{W,k}$ desde $k = 1$ hasta $k = 150$. En la Figura 3.10 se muestran los resultados cuando $A_{W,k}$ se halla con la SVD exacta y en la Figura 3.11 cuando se utiliza la SVD aleatorizada.

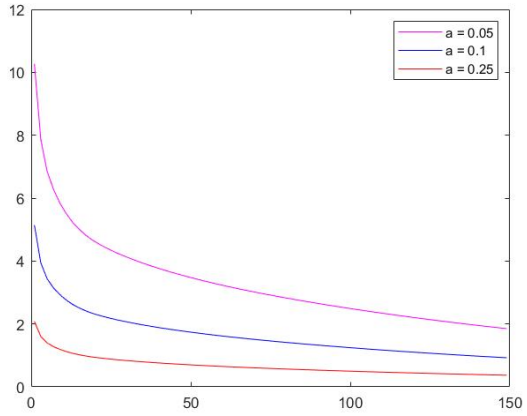

Figura 3.10: SVD exacta

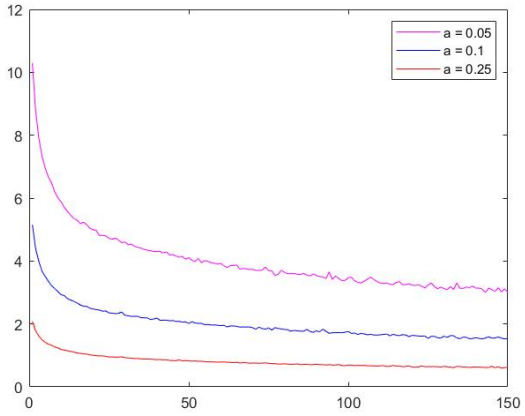

Figura 3.11: SVD aleatorizada

Vemos que el error es más pequeño cuando a es mayor, y que disminuye al aumentar k de forma más estable y rápida en la Figura 3.10 (sin las oscilaciones de la Figura 3.11), debido a que partimos de la SVD exacta para hacer la aproximación, y no de la aleatorizada. Es decir, para a fijo vamos a necesitar un valor mayor de k si queremos obtener el mismo error partiendo de la aproximación calculada con la SVD aleatorizada que con la SVD exacta.
En cuanto al valor de a, ya vimos en la Figura 3.3 que para $a = 0{,}1$ en el caso del esquema de Jain la imperceptibilidad no es buena, y de hecho la Figura 3.6 nos dice que para $a = 0{,}25$ va a ser todavía peor. Por tanto el valor de k va a tener que ser alto si queremos extraer la marca de agua de forma precisa manteniendo un alto grado de imperceptibilidad. Sobre este último punto, notamos que existe un procedimiento que mejora al de Jain [7], en cuanto a la imperceptibilidad, aunque no se ha considerado en este trabajo.
Finalmente, podemos comprobar que si ejecutamos el algoritmo de extracción de la marca de agua que proponen Liu y Tan, esto es, las ecuaciones (3.6), podemos *inventarnos* cualquier marca de agua en la Figura 3.4 y recuperarla a partir de ella.

Figura 3.12: Imagen que vamos a *extraer* de la Figura 3.4

Figura 3.13: Marca de agua extraída de la Figura 3.4 con el método de Liu y Tan.

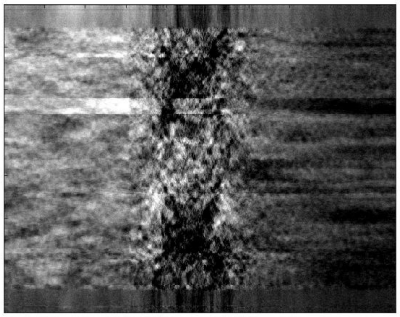

Figura 3.14: Marca de agua falsa extraída de la Figura 3.4 con el método de Jain.

En la Figura 3.12 se muestra la marca de agua *inventada* que vamos a extraer de la Figura 3.4, aunque realmente no está oculta en ella y en la Figura 3.13 la falsa marca de agua extraída. Sin embargo, como vemos en la Figura 3.14, si intentamos extraer con el algoritmo de Jain dicha marca de agua, no lo conseguimos. No hemos podido *inventarnos* una marca de agua que no existe, como sí hemos podido hacer con el método de Liu y Tan en la Figura 3.13.

3.2. LSI: Análisis semántico latente

Habitualmente la información se recupera emparejando términos que aparecen en los documentos con los de una búsqueda. Sin embargo, este método puede ser impreciso, debido a que hay palabras que tienen distintos significados (polisemia) o varias palabras que tienen significados similares (sinonimia).
El análisis semántico latente (*Latent semantic indexing*, LSI) intenta resolver estos problemas usando la relación entre las palabras, la frecuencia con que aparecen en los documentos y su relación con otras palabras con las que suelan aparecer de manera conjunta en vez de considerar palabras individuales para la recuperación.

3.2.1. Introducción.

Lo primero que necesita el LSI es construir una matriz que relacione términos con documentos. Pongamos, por ejemplo, una matriz $A = (a_{ij})$, $1 \leq i \leq m$, $1 \leq j \leq n$, donde a_{ij} representa la frecuencia con la que aparece el término i en el documento j. Como no todas las palabras aparecen en todos los documentos, la matriz A generalmente va a ser una matriz dispersa.
La descomposición en valores singulares obtiene y almacena la información de A en las matrices U, Σ y V. Estas representan un desglose de las relaciones originales en vectores linealmente independientes (las bases de los subespacios columna y fila de las que hablamos en la Observación 1.1.2). El uso de los k primeros valores y vectores singulares es equivalente a aproximar la matriz de términos-documentos original por A_k. De alguna forma, la SVD es una técnica para obtener una serie de vectores donde cada término y documento está representando por un vector usando los vectores singulares por la derecha o la izquierda. En particular, las columnas de U y V se consideran los vectores términos y documentos, respectivamente.
Es importante tener en cuenta que A_k no reconstruye exactamente la matriz original A, pero la SVD truncada recoge la mayor parte de la estructura en la asociación entre términos y documentos. Al mismo tiempo, elimina el ruido que abunda en los métodos de recuperación de información basados en palabras.
De forma intuitiva, como $k \ll m$, el número de términos, las pequeñas diferencias en termino-

logía se ignorarán. Por otro lado, términos que aparecen en distintos documentos estarán *cerca* en el subespacio de dimensión k. Esto se traduce en que documentos que no contengan ninguna palabra de la búsqueda de un usuario puedan estar cerca en el subespacio de dimensión k si existen otros documentos que tengan palabras en común con los primeros.
Consideremos las palabras coche, automóvil, conductor y elefante. Coche y automóvil son sinónimos, tienen relación con conductor y no tienen nada que ver con elefante. En la mayoría de sistemas de recuperación de información, la búsqueda de la palabra *coche* no va a proporcionar más documentos sobre automóviles que sobre elefantes si no aparece la palabra coche en el documento. Sería preferible que sí lo hiciera, incluso que también incluyera algún documento sobre conductores. La principal idea del LSI es modelar las relaciones entre términos (mediante la SVD truncada) y aprovecharlo para mejorar la búsqueda.

3.2.2. Búsqueda y actualización

Para poder recuperar la información, la búsqueda debe representarse como un vector en el subespacio de dimensión k. Una búsqueda es una serie de palabras. Por ejemplo, se puede representar como

$$\hat{\mathbf{q}}^T = \mathbf{q}^T U_k (\Sigma_k)^{-1} \tag{3.12}$$

donde $\mathbf{q}$ es el vector con las palabras de la búsqueda, U_k, Σ_k son las matrices de tamaño $m \times k$ y $k \times k$ de la SVD truncada (1.6). En esencia, el vector búsqueda es una suma ponderada de los términos de la búsqueda, que podemos comparar posteriormente con los vectores documentos existentes y ordenarlos por similitud. Una forma de medir esta similitud es el coseno entre el vector de búsqueda y el vector documento (*midiendo* la ortogonalidad entre ambos vectores).
Por otro lado, suponiendo que ya tenemos una base de datos generada. Si queremos añadir nuevos términos o nuevos documentos, tenemos dos opciones: crear una nueva matriz, y repetir todo el proceso desde el principio (incluida la SVD) o actualizar las matrices que ya tenemos. Para actualizar una base de datos ya creada, incorporaremos nuevos términos y documentos de la siguiente forma. Dado $\mathbf{d} \in \mathbb{R}^{m\times 1}$ el vector del nuevo documento, calculamos

$$\hat{\mathbf{d}}^T = \mathbf{d}^T U_k (\Sigma_k)^{-1} \tag{3.13}$$

y lo incorporamos a la derecha de V_k^T, es decir, hacemos $\tilde{V}_k^T = \left(V_k^T | \hat{\mathbf{d}}\right)$.
Para los términos hacemos algo muy similar. Dado el vector del nuevo término $\mathbf{t} \in \mathbb{R}^{n\times 1}$, calculamos

$$\hat{\mathbf{t}}^T = \mathbf{t}^T V_k (\Sigma_k)^{-1}, \tag{3.14}$$

e incorporamos $\hat{\mathbf{t}}^T$ por debajo a la matriz U_k.

3.2.3. Una ilustración de LSI.

Hemos cogido de [1] la información que se recoge en la Tabla 3.1, que se refiere a 16 términos que aparecen en los títulos de una colección de 17 libros. A partir de esa información, se genera la matriz A de términos-documentos de tamaño 16×17 con la que vamos a trabajar.

Términos	Documentos																
	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15	B16	B17
Algoritmos	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0
Aplicación	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Retraso	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
Diferencial	0	0	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0
Ecuaciones	1	1	0	1	0	0	0	1	0	1	1	1	1	1	1	0	0
Implementación	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
Integral	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
Introducción	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
Métodos	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
No lineal	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0
Ordinario	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
Oscilación	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
Parcial	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
Problema	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	0
Sistemas	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0
Teoría	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1

Tabla 3.1: Frecuencia de los términos en cada documento

Calculamos la aproximación de rango $k = 2$ a partir de la SVD de esta matriz, es decir, nos quedamos con $U_2 = [\mathbf{u}_1, \mathbf{u}_2]$, $V_2 = [\mathbf{v}_1, \mathbf{v}_2]$ y $\Sigma_2 = \begin{pmatrix} \sigma_1(A) & 0 \\ 0 & \sigma_2(A) \end{pmatrix}$. Si ahora tomamos como base del subespacio generado por las columnas de U_2 $\{\sigma_1(A)\mathbf{u}_1, \sigma_2(A)\mathbf{u}_2\}$, las coordenadas en dicha base de los vectores asociados a cada documento están determinadas por la relación $A_2 = \sigma_1(A)\mathbf{u}_1\mathbf{v}_1^T + \sigma_1(A)\mathbf{u}_1\mathbf{v}_2^T$. Del mismo modo, tomando como base del subespacio generado por las columnas de V_2 $\{\sigma_1(A)\mathbf{v}_1, \sigma_2(A)\mathbf{v}_2\}$, las coordenadas de los vectores asociados a cada término en esa base vienen dadas por la relación $A_2^T = \sigma_1(A)\mathbf{v}_1\mathbf{u}_1^T + \sigma_1(A)\mathbf{v}_2\mathbf{u}_2^T$. Por tanto podemos usar $\mathbf{u}_1$ como las coordenadas en el eje x y $\mathbf{u}_2$ como coordenadas en el eje y de los términos, y lo mismo para los documentos con $\mathbf{v}_1$ y $\mathbf{v}_2$.

Si ahora queremos hacer una búsqueda que contenga las palabras *aplicación* y *teoría*, tomamos el vector $\mathbf{q} = (0, 1, 0, \ldots, 0, 1)^T \in \mathbb{R}^{16\times 1}$ y las coordenadas del vector de búsqueda vienen dadas por $\hat{\mathbf{q}}^T = \mathbf{q}^T U_2 \Sigma_2^{-1}$. Ahora podemos compararlo con el resto de documentos en el plano cartesiano. El resultado de nuestra búsqueda serán todos los documentos cuyo coseno con respecto al vector de búsqueda sea menor que $0,9$. Todo esto se puede ver representado en la Figura 3.15, donde en magenta están representados los términos, en azul los documentos, y en rojo el vector de búsqueda junto con el ángulo permitido por el umbral $0,9$ utilizado.

Observamos que con este umbral del coseno, la búsqueda devuelve los documentos B3, B5, B6, B7, B16 y B17. Si aumentamos el umbral del coseno a $0,55$ (línea roja discontinua de la Figura 3.15), los títulos B11 y B12 también habrían aparecido en la búsqueda.

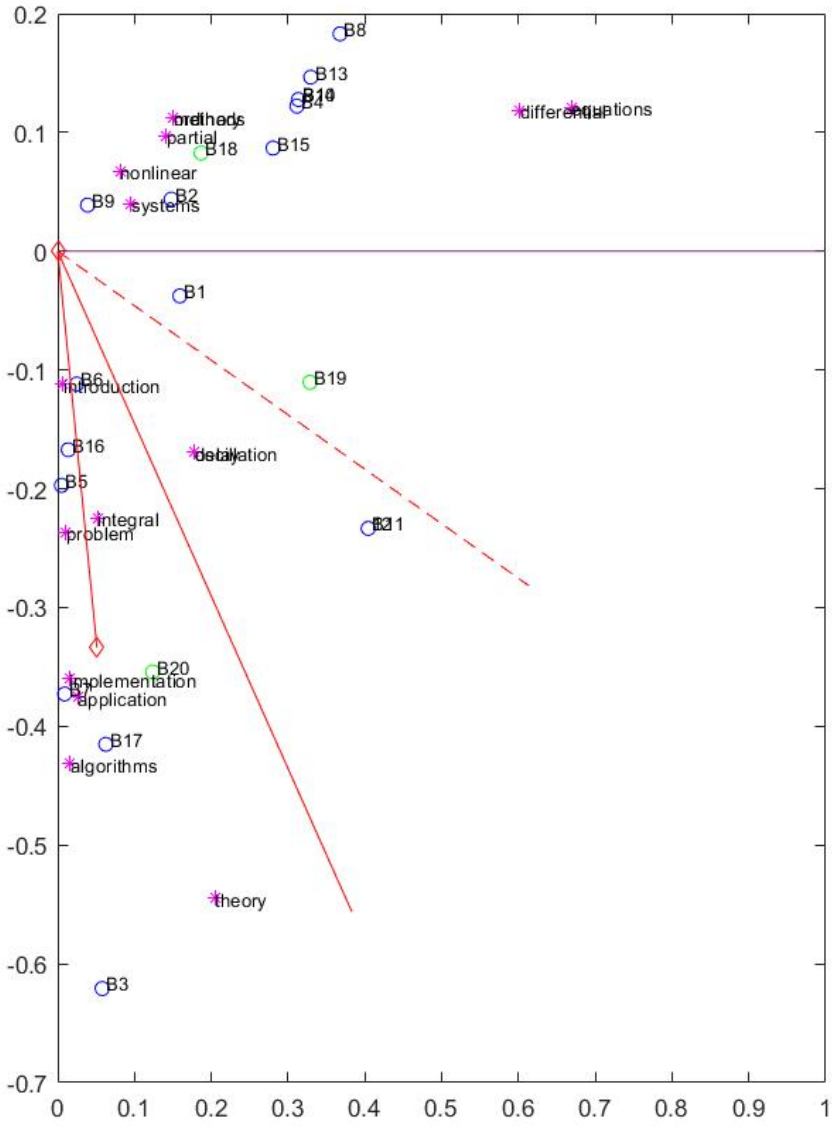

Figura 3.15: Gráfico bidimensional de los términos y documentos, con la búsqueda

Sin embargo, si hubiéramos hecho una búsqueda con palabras individuales solo habríamos obtenido los títulos B3, B11, B12 y B17. Es decir, el LSI consigue extraer cuatro documentos

adicionales que son relevantes en la búsqueda, aunque no comparten ninguna palabra con ella. Si ahora queremos incorporar los documentos que aparecen en la Tabla 5 de [1] siguiendo el proceso descrito en (3.13), obtenemos la Figura 3.16, donde se han incorporado los nuevos títulos en color verde.

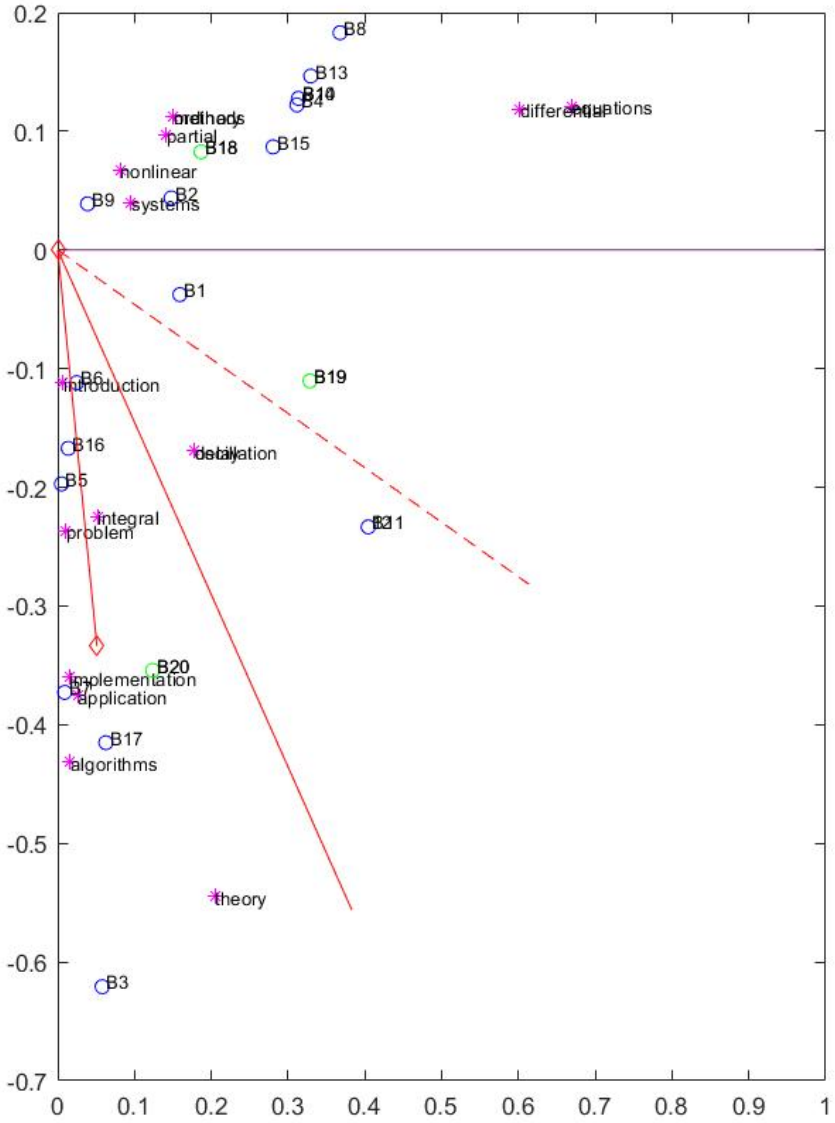

Figura 3.16: Gráfico con los nuevos documentos incorporados

Ahora, con el umbral del coseno original la búsqueda también nos devolvería el título B20. La forma ideal de producir la mejor aproximación de rango k actualizada con los nuevos térmi-

nos y documentos sería volver a calcular la SVD de la nueva matriz términos-documentos. Sin embargo esto supone un coste computacional muy alto cuando estamos tratando con matrices grandes. Es por eso que los métodos de actualización se vuelven más interesantes cuando tenemos restricciones en el tiempo o en la memoria para almacenar la matriz. En nuestro ejemplo, dadas las dimensiones, no supone un problema volver a construir la matriz y repetir el proceso desde el principio. Esto se ve representado en la Figura 3.17.

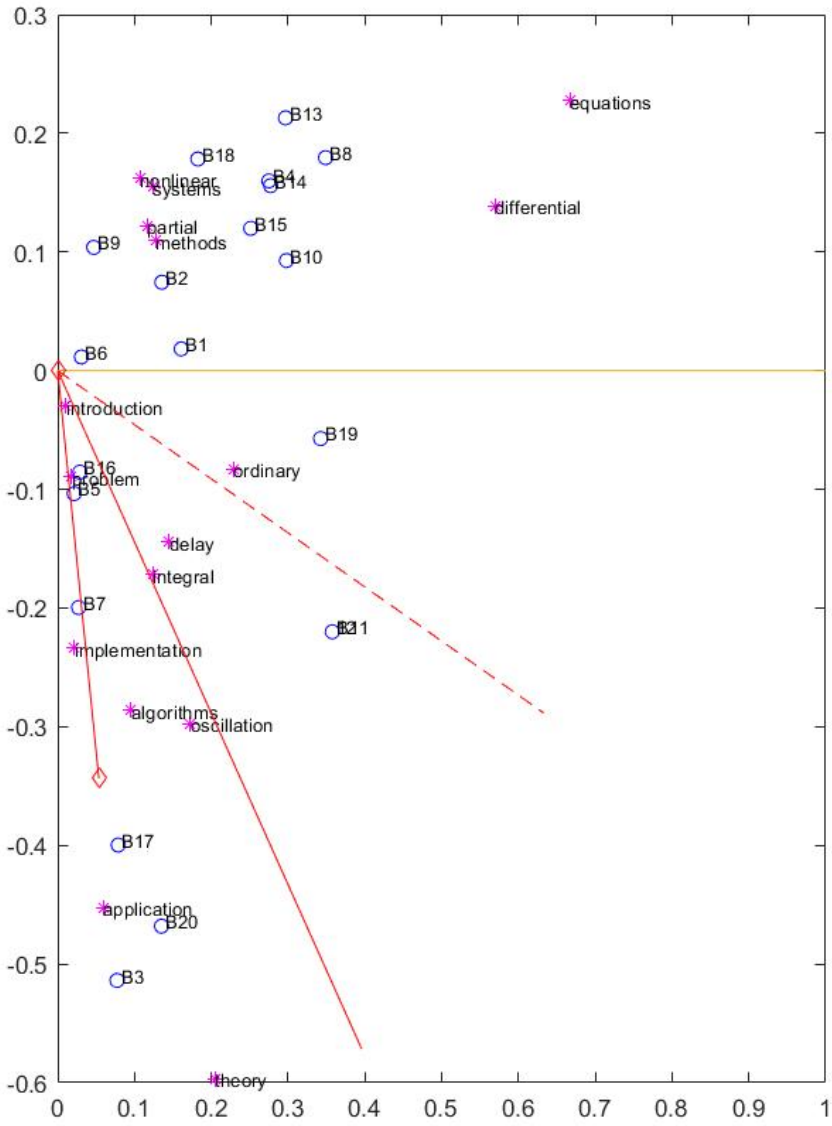

Figura 3.17: Gráfico bidimensional para la matriz ampliada con los nuevos documentos.

Dado que la matriz es pequeña y no se han introducido muchos documentos nuevos, no apreciamos cambios significativos con respecto al caso en el que simplemente incorporamos los nuevos documentos. De hecho, el resultado de nuestra búsqueda original nos va a devolver exactamente los mismos títulos en ambos casos.

Bibliografía

[1] M. W. Berry, S. T. Dumais and G. W. O'Brien, "Using Linear Algebra for Information Retrieval", SIAM Review Vol. 37, No. 4, pp. 573-595, 1995.

[2] S. Craver, N. Memon, B. Yeo, and M. Yeung, "Can invisible watermarks resolve rightful ownership?", IBM Research Division, Tech. Rep. RC 20509, 1996.

[3] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast Monte Carlo Algorithms for Matrices I: Approximating Matrix Multiplication", SIAM J. Comput. Vol. 36, No. 1, pp. 132–157, 2006

[4] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast Monte Carlo Algorithms for Matrices II: Computing a Low-rank Approximation to a Matrix", SIAM J. Comput. Vol. 36, No. 1, pp. 158–183, 2006.

[5] G. H. Golub and C. F. Van Loan, "Matrix Computations", The Johns Hopkins University Press, Third Edition, 1996.

[6] C. Jain, S. Arora, and P. K. Panigrahi, "A Reliable SVD based Watermarking Scheme", arXiv:0808.0309v1, 2008.

[7] K. Keegan, D. Melendez and J. Zheng, "Randomized Singular Value Decomposition and its Applications", Brown University Summer Tech. Rep.@ICERM (RI), 2020.

[8] R. Liu and T. Tan, "An SVD-Based Watermarking Scheme for Protecting Rightful Ownership", IEEE Transactions on Multimedia, Vol. 4, No. 1, pp. 121-128, 2002.

[9] M. W. Mahoney, "Lecture Notes on Randomized Linear Algebra, 14. Additive-error Low-rank Matrix Approximation", 2013.

[10] G. Strang, "The fundamental Theorem of Linear Algebra", The American Mathematical Monthly, Vol. 100, No. 9., pp. 848-855, 1993.

[11] G. Strang, "Linear Algebra and Learning from Data", Wellesley - Cambridge Press, 2019.

[12] X. Zhang and K. Li, Comments on "An SVD-Based Watermarking Scheme for Protecting Rightful Ownership", IEEE Transactions on Multimedia, Vol. 7, No. 2, pp. 593-594, 2005

Apéndice A

Código de Matlab.

En este apéndice incluimos los programas de Matlab que implementan los algoritmos básicos que se han estudiado en este trabajo y que se han utilizado para elaborar las distintas figuras que se han incluido en la memoria.

A.1. Producto aleatorizado de dos matrices.

En este primer código se ha implementado el Algoritmo 1, presentado en el Capítulo 2. La función acepta como entradas las matrices A y B que se quieren multiplicar, el tamaño c de la muestra y el vector de probabilidades para el muestreo. La salida son las matrices C y R, cuyo producto aproxima el de A y B.

```
function [C,R] = basicmatrixprod(A,B,c,prob)
[m,n1] = size(A);
[n2,p] = size(B);

if n1 == n2
    n = n1;
    C = zeros(m,c);
    R = zeros(c,p);
    y = randsample(n,c,true,prob);
    for i=1:c
        C(:,i) = A(:,y(i))/(sqrt(c*prob(y(i))));
        R(i,:) = B(y(i),:)/(sqrt(c*prob(y(i))));
    end
else
```

```
15      fprintf('Las dimensiones de las matrices deben de ser adecuadas
            para su producto.')
16  end
17
18  end
```

Incluimos a continuación una función que, dada una matriz, calcula las probabilidades óptimas (2.5) tanto para el producto aleatorizado de matrices como para la SVD aleatorizada.

```
1  function prob = probabilidades(A,B)
2  [~,n]=size(A);
3  prob = zeros(1,n);
4  denom = 0;
5
6  for i=1:n
7      denom = denom + norm(A(:,i))*norm(B(i,:));
8  end
9
10 for j=1:n
11     prob(j) = (norm(A(:,j))*norm(B(j,:)))/denom;
12 end
13
14 end
```

A.2. Descomposición en valores singulares aleatorizada.

En el siguiente código implementa el Algoritmo 2, presentado en la Sección 2.2. La función acepta como entradas la matriz A de la que se quiere calcular la SVD aproximada, el rango k de la aproximación buscada, el valor c del tamaño de la muestra con la que se va a aproximar el producto de AA^T y el vector de probabilidades para el muestreo. La salida la forman la matriz H_k y un vector con los k primeros valores singulares de C.

```
1  function [Hk,Sigma] = LinearTimeSVD(A,k,c,prob)
2
3  [m,n] = size(A);
4  C = zeros(m,c);
5  Hk = zeros(m,k);
```

```
Sigma = zeros(1,k);
y = randsample(n,c,true,prob);

for i=1:c
    C(:,i) = A(:,y(i))/(sqrt(c*prob(y(i))));
end

CC = transpose(C)*C;

[U,aux,~] = svd(CC);
aux = sqrt(aux);



for j=1:k
    Hk(:,j) = C*U(:,j)/aux(j,j);
    Sigma(j) = aux(j,j);
end

end
```

A.3. Implementación y extracción de una marca de agua con el esquema de Liu y Tan.

En el siguiente programa se implementa el esquema de Liu y Tan descrito en la Sección 3.1.2. La función acepta como entrada dos imágenes, la original y la marca de agua que se quiere implementar, y el factor de escala a. Devuelve la imagen IMGWM con la marca de agua incorporada, junto con las matrices necesarias para poder extraer posteriormente la marca de agua.

```
function [UW,IMGWM,VW,Sigma] = marcadeagua(IMG,MDA,a)

A = imread(IMG);
W = imread(MDA); %leemos las imagenes
A = im2gray(A);
```

```
6  W = im2gray(W); %pasamos la marca de agua y la imagen a escala de
       grises
7  A = double(A);
8  W = double(W);
9
10 [U,Sigma,V] = svd(A);
11 [UW,SigmaW,VW] = svd(Sigma+a*W);
12 IMGWM = U*SigmaW*transpose(V); %hacemos el algoritmo
13
14 end
```

Esta segunda función acepta como entrada los datos que se suponen conocidos para poder realizar la extracción de la marca de agua junto con la imagen de la que queremos extraerla, y devuelve la marca de agua extraída.

```
1  function [MDA] = marcadeaguainv(UW,Sigma,VW,IMG,a)
2
3  [~,SigmaW,~] = svd(IMG);
4  D = UW*SigmaW*transpose(VW);
5  MDA = (1/a)*(D-Sigma);
6
7  end
```

A.4. Implementación y extracción de una marca de agua con el esquema de Jain *et al.*

Las siguientes funciones son análogas a las de la sección anterior, pero implementan el esquema de marca de agua de Jain, presentado en la Sección 3.1.2.

```
1  function [IMGWM,V,U,VW] = marcadeaguaJaine(IMG,MDA,a)
2
3  A = imread(IMG);
4  W = imread(MDA); %leemos las imagenes
5  A = im2gray(A);
6  W = im2gray(W); %pasamos la marca de agua y la imagen a escala de
       grises
7  A = double(A);
```

```
W = double(W);

[U,Sigma,V] = svd(A);
[UW,SigmaW,VW] = svd(W);
AWa = UW*SigmaW;
Sigma1 = Sigma+a*AWa;
IMGWM = U*Sigma1*transpose(V);

end
```

```
function MDAW = inversaJaine(AW,A,U,V,VW,a)

A1 = AW-A;
AWa = (U'*A1*V)/a;
MDAW = AWa*transpose(VW);

end
```

A.5. Implementación del análisis semántico latente

Finalmente incluimos el código donde reproducimos el ejemplo de [1].

```
A(1,:) = [0,0,1,0,1,0,1,0,0,0,0,0,0,0,0,0,0];
A(2,:) = [0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1];
A(3,:) = [0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0];
A(4,:) = [0,0,0,1,0,0,0,1,0,1,1,1,1,1,1,0,0];
A(5,:) = [1,1,0,1,0,0,0,1,0,1,1,1,1,1,1,0,0];
A(6,:) = [0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0];
A(7,:) = [1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1];
A(8,:) = [0,0,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0];
A(9,:) = [0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,0,0];
A(10,:) = [0,0,0,0,0,0,0,0,1,0,0,0,1,0,0,0,0];
A(11,:) = [0,0,0,0,0,0,0,1,0,1,0,0,0,0,0,0,0];
A(12,:) = [0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0,0];
A(13,:) = [0,0,0,1,0,0,0,0,0,0,0,0,1,0,0,0,0];
A(14,:) = [0,0,0,0,0,1,1,0,0,0,0,0,0,0,0,1,0];
```

```
A(15,:) = [0,0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0];
A(16,:) = [0,0,1,0,0,0,0,0,0,0,1,1,0,0,0,0,1];

[m,n] = size(A);

terminos = ["algorithms" "application" "delay" "differential"...
    "equations" "implementation" "integral" "introduction" ...
    "methods" "nonlinear" "ordinary" "oscillation" "partial" ...
    "problem" "systems" "theory"];
documentos = ["B1" "B2" "B3" "B4" "B5" "B6" "B7" "B8" ...
    "B9" "B10" "B11" "12" "B13" "B14" "B15" "B16" "B17"];

k = 2;
[Uk,Sigmak,Vk] = svds(A,k);

figure(1)
plot(Uk(:,1),-Uk(:,2),'*','color','magenta');
for i=1:16
text(Uk(i,1),-Uk(i,2),terminos(i),'FontSize',8);
hold on
end
plot(Vk(:,1),-Vk(:,2),'o','color','blue')
hold on
for i=1:17
text(Vk(i,1)+0.005,-Vk(i,2)+0.005,documentos(i),'FontSize',8);
hold on
end
plot([0,1],[0,0],'-');
hold on

query = [0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1];

q = query*Uk/Sigmak;
cos = 0.9;
sen = sqrt(1-cos^2);
```

```
50 threshold = [q(1)*cos+q(2)*sen, q(1)*sen-q(2)*cos];
51
52 cos2 = 0.55;
53 sen2 = sqrt(1-cos2^2);
54 threshold55 = [q(1)*cos2+q(2)*sen2, q(1)*sen2-q(2)*cos2];
55
56 figure(1)
57 plot([0,q(1)],[0,-q(2)],'d','LineStyle','-','Color','red')
58 hold on
59 plot(2*[0,threshold(1)],2*[0,threshold(2)],'LineStyle','-','Color','
   red')
60 hold on
61 plot(2*[0,threshold55(1)],2*[0,threshold55(2)],'LineStyle','--','
   Color','red')
62 hold on
63
64 pause
65
66 d = [0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0;
67       1,0,0,1,1,0,1,0,0,0,1,0,0,0,0,0;
68       0,1,0,0,0,0,0,0,0,0,1,1,0,0,0,1];
69 dd = d*Uk/Sigmak;
70
71 Vk = [Vk;dd];
72
73 docsad = ["B18", "B19", "B20"];
74
75
76 figure(1)
77 plot(Vk(18:20,1),-Vk(18:20,2),'o','color','green')
78 hold on
79 for i =1:3
80     text(Vk(i+17,1)+0.005,-Vk(i+17,2)+0.005,docsad(i),'FontSize',8);
81     hold on
82 end
```

```
83
84 AA = [A,d'];
85 documentos2 = [documentos,docsad];
86 [UUk,Sigma2k,VVk] = svds(AA,k);
87
88 figure(2)
89 plot(UUk(:,1),-UUk(:,2),'*','color','magenta');
90 for i=1:16
91 text(UUk(i,1),-UUk(i,2),terminos(i),'FontSize',8);
92 hold on
93 end
94 plot(VVk(:,1),-VVk(:,2),'o','color','blue')
95 hold on
96 for i=1:20
97 text(VVk(i,1)+0.005,-VVk(i,2)+0.005,documentos2(i),'FontSize',8);
98 hold on
99 end
100 plot([0,1],[0,0],'-');
101 hold on
102
103 query = [0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,1];
104
105 qq = query*UUk/Sigma2k;
106 threshold2 = [qq(1)*cos+qq(2)*sen, qq(1)*sen-qq(2)*cos];
107 figure(2)
108 plot([0,qq(1)],[0,-qq(2)],'d','LineStyle','-','Color','red')
109 plot(2*[0,threshold2(1)],2*[0,threshold2(2)],'LineStyle','-','Color'
    ,'red')
110 hold on
```