

# ALGORITMO GOLUB-KAHAN BIDIAGONAL APLICADO A EIGENFACES

Carlos Enrique Nosa Guzman  
Brayan Alejandro Romero Castro  
Juan Carlos Galvis Arrieta

Tercera conferencia colombiana de Matemáticas aplicadas e industriales - MAPI 3

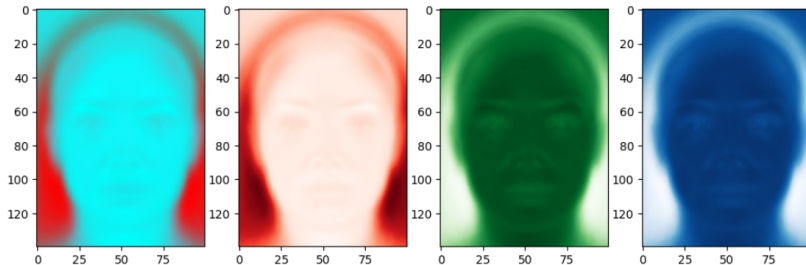
14 de junio del 2024

# Contenidos

- 1 Eigenfaces
- 2 SVD
- 3 Algoritmo
  - Primera parte
  - Segunda parte
- 4 Ventajas del algoritmo
- 5 Resultados
- 6 Referencias

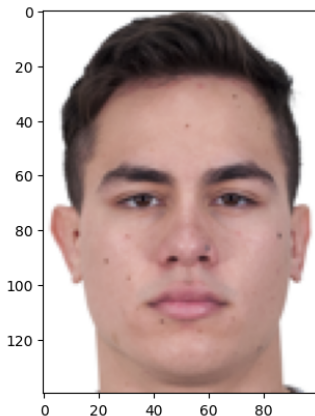
## ¿Qué es una cara propia (eigenface)?

Una cara propia es el nombre dado a un componente principal de un conjunto de imágenes. El cálculo de eigenfaces se hace con el objetivo de reducir la dimensionalidad de un conjunto de imágenes y poder describir con ‘poca’ información los rasgos principales de los rostros en cuestión. Esta técnica es usada en el reconocimiento y reconstrucción de rostros.



**Figura:** Cara propia en diferentes visualizaciones. Realización propia.

# Conjunto de datos




**Figura:** Cara de ejemplo de conjunto de datos (dimensiones:  $m \times n \times k = 140 \times 100 \times 3$ ). Tomado de [1].

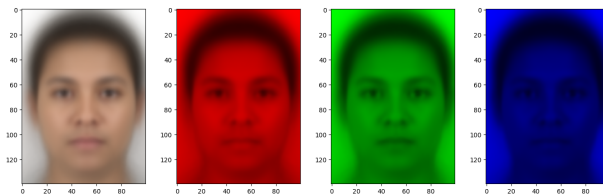
- La base de datos fue desarrollada por la Universidad de Chicago. (Véase [1])
- Es una herramienta diseñada para la investigación científica. Es de libre acceso.
- En total posee  $p = 821$  fotografías de individuos masculinos y femeninos entre los 17 y 65 años provenientes de Estados Unidos e India.

# Procedimiento

- Construir la matriz  $M$  que tenga en sus columnas a cada imagen.

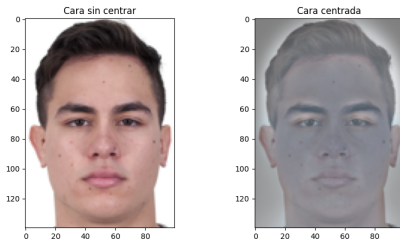

$$\widetilde{M} = (I_1 \ I_2 \ \dots \ I_p) \longrightarrow M = \begin{pmatrix} I_1^1 & I_2^1 & \dots & I_p^1 \\ I_1^2 & I_2^2 & \dots & I_p^2 \\ \vdots & \vdots & \ddots & \vdots \\ I_1^{mn} & I_2^{mn} & \dots & I_p^{mn} \end{pmatrix}$$

- Calcular la cara promedio.



# Procedimiento

- Centrar cada una de las imágenes.



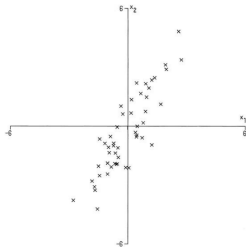
- Calcular la matriz de varianza del conjunto de datos.

$$\hat{Var}[\tilde{M}] = E[M^T M] - E[M]^T E[M] = E[M^T M] = \frac{1}{mn} M^T M.$$

La matriz  $\frac{1}{mn} M^T M$  tiene dimensiones  $p \times p$ .

# Procedimiento

- Una vez obtenida la matriz de varianza estimada  $\hat{Var}[\tilde{M}] = \frac{1}{mn} M^T M$  del vector aleatorio  $\tilde{M}$ , se hace **análisis de componentes principales (ACP)**.
  - El ACP es una técnica de medición de variabilidad de un conjunto de datos.



**Figura:** Conjunto de datos en  $\mathbb{R}^2$ . Imagen tomada de [2].

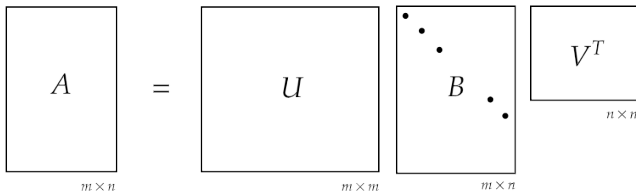
- A nivel computacional, el **ACP** se puede hacer mediante la **descomposición en valores singulares (SVD)** de una matriz asociada al conjunto de datos.

# Descomposición en valores singulares (SVD)

## Teorema.

Para toda matriz  $A_{m \times n}$  existen matrices  $U$ ,  $B$  y  $V$  tales que  $A_{m \times n} = U_{m \times m} B_{m \times n} V_{n \times n}^T$  con  $U$  y  $V$  son ortogonales y  $B$  una matriz diagonal. De otra manera,

$$A = \sum_{i=1}^{\min\{m,n\}} b_{ii} \mathbf{u}_i \mathbf{v}_i^T.$$





# Procedimiento

- Finalmente, si  $M = U\Sigma V^T$  con  $U \in \mathbb{R}^{mn \times mn}$ ,  $\Sigma \in \mathbb{R}^{mn \times p}$  y  $V \in \mathbb{R}^{p \times p}$ , entonces
  - $U$  contiene en sus columnas las caras propias.
  - $M^T M = V\Sigma^T \Sigma V^T$ .

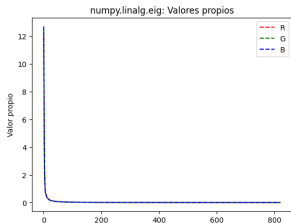


Figura: Valores propios de  $M^T M$ .

Truncamiento	R	G	B
1	26.2997	28.512	29.016
5	64.7753	65.6549	66.2674
10	72.4903	73.3829	74.0402
20	79.4073	79.975	80.6594
30	83.1439	83.4289	83.9529
40	85.5571	85.7124	86.1336
50	87.3247	87.404	87.7408
100	92.0806	92.0154	92.1941
150	94.3655	94.2783	94.4015
200	95.7495	95.6663	95.7595
250	96.7332	96.6525	96.7274
300	97.4582	97.3808	97.4399
500	99.0418	98.9709	98.982
750	99.8234	99.8137	99.8146
821	100	100	100

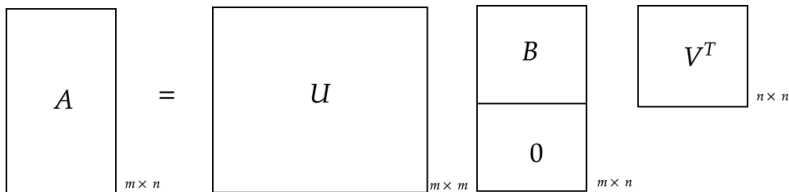
Figura: Tabla de porcentaje de varianza.

# Algoritmo

La idea del algoritmo es primero bidiagonalizar a  $A$ , y luego aplicar un iteración  $QR$  implícita a la matriz bidiagonal resultante, para obtener una descomposición SVD de  $A$ .

- Se bidiagonaliza por Householder.

$$A = UB'V^T$$



# Algoritmo

- Se hace SVD de la parte bidiagonal  $B' = U'_B \Sigma' V_B^T$ , quedando la descomposición de  $A$  requerida.

$$A = U U'_B \Sigma' V_B^T V^T$$

$A$	$=$	$U$	$U_B$	$0$	$\Sigma$	$V_B^T$	$V^T$
			$0$	$I_{n \times n}$	$0$		

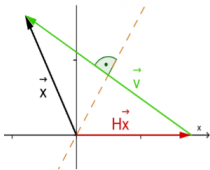
# Primera parte

Para bidiagonalizar  $A$ , se aplican reflexiones de Householder, que ayudarán a **cerar** filas y columnas.

$$B' = \begin{bmatrix} B \\ 0 \end{bmatrix} = U_n \cdots U_1 A V_1 \cdots V_{n-2}$$

*Se refleja en vectores canonicos*

*Reflexión con eje de simetría el normal de  $v$*



$$H_v = I - \frac{2vv^T}{v^Tv}$$

# Primera parte

$$B' = \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}$$

$$B' \leftarrow B'V_1 = \begin{bmatrix} \times & \times & 0 \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix}$$

$$B' \leftarrow U_1B' = \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix}$$

$$B' \leftarrow U_2B' = \begin{bmatrix} \times & \times & 0 \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & \times \end{bmatrix}$$

$$B' \leftarrow U_3B' = \begin{bmatrix} \times & \times & 0 \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & 0 \end{bmatrix}$$

## Segunda parte

Se puede emplear la iteración  $QR$  para diagonalizar  $T = B^T B$ , y a  $\hat{T} = BB^T$ , como

$$T = QDQ^T$$

$$\hat{T} = \hat{Q}D\hat{Q}^T$$

Por medio del teorema  $Q$  implícito de [3] (Pag. 454 y Pag. 416) es posible asegurar que la descomposición SVD de  $B$ , está dada por:

$$B = \hat{Q}D^{1/2}Q^T$$

Esto resulta lento pues hace uso de productos grandes, como  $T_k = R_{k-1}Q_{k-1}$ , además que se hace  $T$  y  $\hat{T}$  por separado.

## Segunda parte

Para evitar aplicar el algoritmo QR a las matrices  $T$  y  $\hat{T}$  se realiza de manera implícita:

- En el algoritmo QR se computaban matrices de Givens  $G_1, G_2, \dots, G_{n-1}$ , tales que  $Q_1 = G_1 G_2 \dots G_{n-1}$ , y  $T_1 = Q_1^T T Q_1$  era tridiagonal, con  $\text{off}(T_1) \leq \text{off}(T)$ .
- Solo es necesario calcular la matriz  $G_1$ , aplicarla a  $B$  y luego de esto, llevar al producto  $BG_1$  nuevamente a una matriz bidiagonal.

## Segunda parte

Se hace iteración QR, de forma implícita por medio del siguiente procedimiento:

*Primer paso de la iteración QR de forma implícita*

$$B = \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix} \quad B \leftarrow BG_1 = \begin{bmatrix} \times & \times & 0 & 0 \\ + & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}$$

$$B \leftarrow U_1^T B = \begin{bmatrix} \times & \times & + & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix} \quad B \leftarrow BV_2 = \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & + & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}$$

$$B \leftarrow U_1^T B = \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & + \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix} \quad B \leftarrow BV_3 = \begin{bmatrix} \times & \times & 0 & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix}$$

$$B_1 = (U_{n-1}^T \cdots U_1^T) \cdot B \cdot (G_1 \cdot V_2 \cdots V_{n-1}) = U^T B V$$

$$T_1 = B_1 B_1^T$$

Luego lo volvemos  
aplicar a  $B_1$



# Ventajas del algoritmo

- Sirve para cualquier tipo de matriz  $m \times n$ , característica que no tienen muchos métodos, que requieren condiciones muy específicas sobre las matrices.
- Aunque la primera parte del algoritmo puede tener multiplicaciones grandes, esta es una parte mínima del algoritmo, pues solo se realizan  $2n - 2$  multiplicaciones de estas, y luego se aprovecha el enfoque en pequeñas secciones de las matrices que tienen las rotaciones de Givens, haciendo que la convergencia sea cúbica (Pag 263 de [3])
- Tiene un sustento teórico muy fuerte, estable y bastante estudiado, al estar basado completamente en matrices ortogonales, su modificación o cambios según requiera el tipo de matriz puede ser fácilmente adaptado.

# Resultados

La comparación entre métodos se hizo tomando un máximo de 1000 iteraciones y un error umbral fuera de la diagonal de  $D$  de  $10^{-5}$ ; se tomó el promedio de correr el programa con cada método cinco veces, para la misma matriz  $M$  de tamaño  $821 \times 821$ .

	Tiempo en minutos	Precisión $\ \Sigma_r V - VD\ _F^2$
Potencias con Deflación	11.13	$1.8765 \cdot 10^{-4}$
Potencias matriz Ortogonal	13.44	$1.3527 \cdot 10^{-6}$
Jacobi	0.66	$5.1705 \cdot 10^{-27}$
Golub and Kahan	1.96	$9.9853 \cdot 10^{-10}$

$$\sum_{i=1}^{821} \|M^T M v_i - v_i \sigma_i^2\|_2^2 = \|M^T M V - V \Sigma^T \Sigma\|_F^2$$

$$= \|\Sigma_r V - VD\|_F^2$$

# Referencias

- [1] Cfd | Chicago Face Database. (s.f.). Recuperado 22 de noviembre de 2023, de <https://www.chicagofaces.org/>
- [2] Golub, Gene Howard, y Charles F. Van Loan. Matrix Computations. JHU Press, 2013.
- [3] Jolliffe I. T. Principal Component Analysis. Springer-Verlag, 2002. DOI.org, <https://doi.org/10.1007/b98835>.