DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 2

Procesamiendo de Imágenes, análisis de componentes principales y reducción de la dimensionalidad

---

Métodos Numéricos

Joaquín Carrasco: joaquin.e.carrasco@gmail.com
Santiago Pensotti: santipensotti@gmail.com
Valentina Vul: valenvul@gmail.com

Resumen:

El objetivo de este trabajo es comparar y analizar diferentes metodologías para la compresión de imágenes que se encuntran en estudio hoy en día. Se implementaron algoritmos de PCA y 2DPCA, y del método de la potencia para conseguir los autovalores necesarios para estos.

En base a diferentes métricas se realizaron comparaciones entre los dos métodos y se llegó a la conclusión de que el 2DPCA presenta muchos beneficios por encima del otro método. No solo es más eficiente en cuanto a costo de cómputo, sino que tambien presenta un error menor y hace una mejor clasificación de imágenes con variaciones con respecto al conjunto de entrenamiento.

Palabras clave:

Método de la potencia, autovalores, autovectores, PCA, 2DPCA, compresión de imágenes

Índice

1. Introducción Teórica

En este informe se busca analizar y contrastar diferentes métodos que proponen respuestas a algunas problemáticas de las que se encarga el campo del procesamiento de imágenes. Entre estas se encuentran la detección de rostros, la compresión de imágenes y el cómputo de la similitud entre estas. Para embarcar estas tareas, se trabaja con matrices numéricas cuyos elementos codifican los valores de los pixeles de las imágenes.

Una técnica ampliamente usada en este ámbito es análisis de componentes principales o PCA. Este permite encontrar las características más distintivas de un conjunto de datos y, basandose en estas, proyectar la muestra en un subespacio de menor dimensión, eliminando así el ruido. Las componentes principales de un conjunto de datos van a ser aquellas que presentan mayor variabilidad, es decir, aquellas direcciones en las que se maximice la *varianza*.

Dado un conjunto de imágenes, a cada una se la puede aplanar, de manera que la muestra quede compuesta por vectores de dimensión n, la cantidad de pixeles de las imágenes: $X = \{x_1, ..., x_m\}$. Estos vetores van a ser proyectados sobre direcciones dadas por vectores u_i, también de dimensión n. De esta forma, la varianza de los datos proyectados se calcula de la siguiente manera

$$\frac{1}{m}\sum_{j=1}^{n}(u_i^t x_j - u_i^t \bar{x})^2 = u_i^t C u_i \tag{1}$$

Siendo $\bar{x}$ la media de las imágenes.

A esta matriz C se la denomina matriz de covarianza. Tomando $X \in \mathbb{R}^{m\times n}$ como la matriz en la que cada fila representa una imágen de la muestra y $\bar{x}$ el vector fila con las medias de cada variable de la imágen, se define a la matriz X_c como la matriz de los datos centrados, es decir $X_{c[i]} = X_{[i]} - \bar{x}$

$$C = \frac{X_c^t X_c}{n-1} \tag{2}$$

Al maximizar la ecuación de la varianza (ecuación 1), se llega a que el valor máximo se consigue cuando u_i es el autovector asociado al autovalor de mayor módulo de la matriz C. Esta matriz es simétrica, por lo que es diagonalizable y sus autovectores forman una base ortonormal de un subespacio. De esta forma, se la puede reescribir como

$$C = VDV^t$$

D es la matriz diagonal cuyos elementos no nulos son los autovalores de C, y las columnas de V son los autovectores asociados a cada uno de estos. Al ordenar los autovalores en D de mayor a menor, se pueden tomar las k primeras columnas de V como las k características principales de la muestra.

Una vez calculadas estas matrices, los datos pueden ser proyectados a un espacio de dimensionalidad $k < n$, mediante la multiplicación de la matriz X, con los datos originales, por una matriz V_k que representa las k caracteristicas principales de la muestra

$$P_k = XV_k \tag{3}$$

Como V_k es ortogonal, y por ende inversible, es posible conseguir las imágen recontruida multiplicando a la proyección de una imágen por la inversa de esta, es decir su traspuesta $A = P_{k[i]} \times V_k^t$.

Para encontrar a D y a V, es necesario encontrar los autovalores y autovectores de C comenzando por el autovalor dominante, es decir, el de mayor magnitud. Como se estableció antes, los autovectores asociados a esta matriz forman una base de un subespacio, por lo que se puede utilizar el método de la potencia para calcularlos.

Este método se basa en definir una sucesión $\{q^k\}$, empezando con $q^0 \in \mathbb{R}^n$ tal que $||q^0||_2 = 1$, tal que

$$z^k = Aq^{k-1} \tag{4}$$
$$q^k = \frac{z^k}{||z^k||_2}$$

Cuando k tiende a infintio, q^k converge al autovector asociado al autovalor de mayor módulo de la matriz A y $\lambda_k = (q^k)^t A q^k$ converge en dicho autovalor.

Una vez que se obtiene este primer autovalor, es necesario calcular el resto. Para esto se hace uso de una técninca de deflación, la cual propone la formación de una nueva matriz cuyos autovalores sean los mismos que los de la original, exceptuando el dominante, que pasa a ser 0.

$$B = A - \lambda_1 * v_1 * v_1^t \tag{5}$$

Luego, se puede volver a aplicar el método de la potencia sobre la matriz B. Como λ_1 ahora vale 0, el autovector dominante de esta nueva matriz será el segundo autovector dominante de la original. Así se pueden calcular λ_2 y v_2 y, si se repiten estos pasos sucesivamente, se pueden obtener todos los autovalores de A.

Al trabajar con imágenes, representadas como matrices, no resulta cómodo el hecho de tener que transformarlas y operar sobre ellas como vectores. Es por esto que se desarrolló otro método basado en este pero que trabaja con las imágenes en su formato original, el 2DPCA.

Nuevamente, el objetivo es encontrar las direcciones a las cuales proyectar las imágenes de manera que se maximice la varianza. Dado un conjunto de m imágenes, cada una una matriz de $\mathbb{R}^{a \times b}$, se define la matriz de covarianza de la imagen como

$$G_t = \frac{1}{m} \sum_{j=1}^{m} (A_j - \bar{A})^t (A_j - \bar{A}) \tag{6}$$

Con A_j una imagen particular y $\bar{A}$ la imagen promedio de toda la muestra. De esta forma, el eje de proyección óptimo, es decir la dirección que maximiza la varianza, se encuentra dado por el autovector asociado al autovalor de mayor magnitud de dicha matriz.

Al proyectar la imagen sobre cada vector óptimo obtenemos las características principales de la imágen. A diferencia de PCA, estas resultan en vectores y no escalares. Realizando esta proyección sobre todos los ejes óptimos, se consigue una familia de vectores que definen las componentes principales de una imagen y, al organizar estos vectores como columnas, se consigue una matriz denominada la *imagen característica* de un dato particular de la muestra.

Al igual que en el PCA, a la imagen se la puede reconstruir tomando la inversa de V, es decir su traspuesta, y multiplicandola por la imagen característica. Tomando solo los k primeros autovectores, y los primeros k vectores característicos, se consigue la proyección de la imagen sobre un subespacio de dimensión k.

$$\tilde{A} = P_k V_k^t + \bar{A} \tag{7}$$

Es necesario tener una medida bajo la cual examinar la calidad de las compresiones de las imágenes. Para esto, se define una matriz de similaridad que, dado un conjunto de datos $X \in \mathbb{R}^{m \times n}$, computa una función sobre cada par de datos. Una de las funciones que se puede usar es la correlación. La correlación entre dos datos se define como la covarianza normalizada, para que su rango se encuentre entre el 1 y el -1. Alta magnitud de correlación implica que los datos son paralelos mientras que el signo indica si comparten dirección.

Dado un conjunto de datos X, su matriz de correlación se calcula en base a la covarianza de la siguiente forma

$$C = \frac{X_c X_c^t}{n-1} \tag{8}$$

$$R_{ij} = \frac{C_{ij}}{\sqrt{C_{ii}C_{jj}}} \tag{9}$$

Luego, se puede medir la similaridad entre imágenes calculando el promedio de esta matriz para el conjunto de datos dado por las imágenes aplanadas y apiladas como filas de una matriz X.

2. Desarrollo

2.1. Método de la potencia

Como se explicó en la introducción, este algoritmo parte de un vector cualquiera y luego, mediante una secuencia de operaciones, se busca hacerlo converger al autovector asociado al autovalor dominante de una matriz.

Lo primero que se hace es generar el punto de partida de manera aleatoria. Con este, se comienza a ciclar una cantidad definida de veces, alterando al vector de manera que siga las ecuaciones de la sucesión definidas previamente (ecuación 4). Este máximo de iteraciones portege al código de quedar en un ciclo infinito, pero la convergencia se debería dar antes. Para verificar si se llegó a esta, en cada ciclo se realiza una comparación con el resultado de la iteración previa. Si la distancia entre los dos autovectores es 0, se asume que se llegó a la convergencia.

Mediante diferentes pruebas, se observó que el método suele converger antes de las 3000 iteraciones. Se decidió dejar un margen generoso y cortar el ciclo si no se consiguió el autovalor después de 10000 iteraciones. Por otro lado, la comparación se implementó mediante una tolerancia de 10^{-6}, ya que al realizar los cálculos con tecnología de punto flotante, se genera un pequeño error numérico (linea 6).

Algorithm 1 Metodo Potencia *(in: A, iter, tol, x, λ)*

1: $x \leftarrow vectorAleatorio(dimension(A))$
2: **for** $i = 0$ **to** $iter$ **do**
3: $res = \mathrm{A} \times \mathrm{x}$
4: $res = res.normalizado()$
5: $norma1 \leftarrow ||res - x||_2$
6: $norma2 \leftarrow ||res + x||_2$
7: **if** $norma1 < \mathrm{tol}$ || $norma2 < \mathrm{tol}$ **then**
8: **break**
9: **end if**
10: **end for**
11: $*\lambda \leftarrow x.transpuesta() \times A \times *x$
12: **Return** λ, x

Una vez encontrado el autovalor dominante y su autovector asociado, se aplica la deflación. Se modifica a la matriz según la ecuación de deflación (ecuación 5) y se vuelve a llamar al método de la potencia sobre esta, guardando los autovalores y autovectores calculados en cada llamado. Este proceso se repite hasta conseguir la cantidad de autovectores que corresponde a la dimensión a la que se busca proyectar las imágenes, k.

Algorithm 2 Deflación *(in: A, iter, tol, k)*

1: $(m) \leftarrow dimension(A)$
2: $autovalores \leftarrow vectorVacio(m)$
3: $autovectores \leftarrow matrizVacia(m \times k)$
4: $x \leftarrow vectorVacio(m)$
5: $\lambda \leftarrow float$
6: $B \leftarrow A$
7: **for** $\mathrm{i} = 0$ **to** k **do**
8: $\lambda, x \leftarrow metodoPotencia(B, iter, tol, x, \lambda)$
9: $B \leftarrow B - \lambda \times x.transpuesto() \times x$
10: $autovalores.pushback(\lambda)$
11: $autovectores.pushback(x)$
12: **end for**
13: **Return** autovalores, autovectores

2.2. PCA

El primer paso de este algoritmo consiste en calcular la matriz de covarianza para el conjunto de datos. Al trabajar con imágenes, representadas como matrices, previamente es necesario aplanarlas, es decir, convertir a cada una en un vector de dimensión igual a la cantidad de pixeles de esta. Una vez hecho esto, se las agrupa en una matriz donde cada fila representa una imágen y se puede realizar el cálculo de la matriz de covarianza.

Se comienza centrando los datos. Para esto es necesario calcular la media de cada columna de la matriz de los datos, o lo que es lo mismo, el promedio de los valores que toma un pixel en particular. A estos se los almacena en un vector, donde cada elemento es la media del pixel de su misma posición (lineas 3 a 6). Luego, a cada vector imagen se le resta el vector promedio y se multiplica a la matriz centrada por su transpuesta y se la divide por el largo de cada imágenes menos 1.

Algorithm 3 matriz_de_cov $(in : A)$

```
1:  (n) ← dimension(A[0])
2:  (m) ← dimension(A)
3:  (medias) ← vectorVacio(n)
4:  for i = 0 to n do
5:      u_j = promedio(X[ , j])
6:      medias.append(u_j)
7:  end for
8:  (X_c) ← vectorVacio(m)
9:  for j = 0  to  m do
10:     x_i = A[i] - medias
11:     X_c.append(x_i)
12: end for
13: G ← (X_c.transpose * X_c)/(n − 1)
14: return G
```

Una vez calculada esta, se utiliza el algoritmo de método de la potencia y de deflación descrito (algoritmos 1 y 2) previamente para obtener los autovalores y autovectores asociados de la matriz.

Algorithm 4 PCA $(in : A, k)$

```
G ← matriz_de_cov(A)
Λ, V ← metodoPotencia(G, k)
return V
```

Ya obtenidos todos los factores, se pueden proyectar las imágenes al nuevo subespacio. Para esto se realiza el producto entre la matriz que contiene a todos los vector-imágenes centrados y V, obteniendo una nueva matriz en la que cada fila representa una imagen proyectada. Como V tiene en sus columnas los k primeros autovectores de A, esta proyección implica una reducción de dimensión.

Algorithm 5 pca_img_proyectada *(in: A, $\bar{A}$, V)*

```
1: P ← (A + Ā) × V
2: Return P
```

Para reconstruir la imagen se multiplica a esta proyección por V traspuesta y se le suma el promedio restado en la proyección, y se obtienen así una aproximación de la imágen original. Por último, se la devuelve a su forma matricial de las mismas dimensiones que la imagen original.

Algorithm 6 reconstruir_img *(in: proyección, , V)*

1: $X \leftarrow proyección \times V^t + \bar{A}$
2: $X \leftarrow X.reshape()$
3: **Return** X

2.3. 2DPCA

El algoritmo de 2DPCA puede ser pensado en tres partes: la generación de la matriz de covarianza (algoritmo 7), el cálculo de los vectores característicos (algoritmo 8) y por último, la proyección de la imagen al nuevo espacio z (algoritmo 9).

Para este primer paso, se comienza calculando la imagen promedio $\bar{A}$ a partir de las imágenes de entrenamiento, es decir el conjunto de imágenes original (lineas 3 a 6). Esta se utiliza para centrar cada dato de la muestra antes de multiplicarlo por su traspuesto a izquierda, y así generar los operandos necesarios para calcular la matriz de covarianza. Una vez sumados, se los divide por la cantidad de imágenes del conjunto consiguiendo la matriz buscada (linea 7 a 11). Por último, se utiliza el algoritmo de método de la potencia descripto previamente (Algoritmo 2) para calcular los autovectores de esta matriz que representan las direcciones de proyección óptimas de la muestra. Estos se disponen como columnas de un nuevo arreglo que es el devuelto por la función.

Algorithm 7 2dpca ($in : A$ lista de imágenes)

1: $(a, b) \leftarrow dimensiones(A[0])$
2: $\bar{A} \leftarrow matrizVacia(a, b)$
3: $M \leftarrow longitud(A)$
4: **for** $i = 0$ **to** M **do**
5: $\quad \bar{A} += A_i$
6: **end for**
7: $\bar{A} \leftarrow \bar{A}/M$
8: $G = matrizVacia(b, b)$
9: **for** $j = 0$ **to** $longitud(A)$ **do**
10: $\quad G += (A_j - \bar{A})^T (A_j - \bar{A})$
11: **end for**
12: $G \leftarrow G/M$
13: $V \leftarrow metodoPotencia(G)$
14: **return** V

Una vez obtenidas las direcciones de proyección óptimas, es necesario calcular la matriz característica de la imagen que se desea comprimir. Esto se logra mediante la multiplicación a derecha de esta imagen centrada por la matriz calculada en el paso anterior. El resultado es una matriz cuya i-ésima columna corresponde a la i-ésima característica principal de la imagen.

Algorithm 8 2dpca_vec_proyectados *(in: A, $\bar{A}$, V)*

1: $P \leftarrow (A - \bar{A}) \times V_k$
2: **Return** P, V_k

Finalmente, para la reconstrucción de la imagen en el nuevo subespacio se hace uso de las dos matrices previamente calculadas. Por lo establecido en la introducción, sabemos que podemos generar una aproximación de la imagen original tomando la matriz característica y los primeros k autovectores de la matriz de covarianza. A estos se los dispone como filas de una matriz y se los multiplica a izquierda por las proyecciones calculadas en el paso anterior y se le suma el promedio previamente restado, regenerando una aproximación de la imagen.

Algorithm 9 2dpca_img_aproximada *(in: P, $\bar{A}$, V_k)*

1: $Aprox \leftarrow P \times V_k^T + \bar{A}$
2: **Return** $Aprox$

2.4. Análisis de similaridad

Para analizar la calidad de la compresión lo primero que se hace es apilar las imágenes que se busca comparar como filas de una matriz. En el caso de 2DPCA, es necesario convertir a cada imagen en un vector previamente. Una vez obtenida la matriz de los datos, se llama a la función de similaridad.

A partir de esta se calcula su matriz de correlación siguiendo los pasos establecidos en la introducción (ecuaciones 8 y 9). Primero calculando la covarianza (lineas 10 a 13) y, a partir de esta, la correlación (linea 15).

Algorithm 10 similaridad *(in: A)*

1: $(n) \leftarrow dimension(A[0])$
2: $(m) \leftarrow dimension(A)$
3: $(medias) \leftarrow vectorVacio(n)$
4: **for** $i = 0$ **to** n **do**
5: u_j = promedio(X[, j])
6: $medias.append(u_j)$
7: **end for**
8: $(X_c) \leftarrow vectorVacio(m)$
9: **for** $j = 0$ **to** m **do**
10: x_i = A[i] - medias
11: $X_c.append(x_i)$
12: **end for**
13: $G \leftarrow (X_c \times X_c^t)/(n-1)$
14: $m \leftarrow dimension(G)$
15: $R \leftarrow matrizVacia(m, m)$
16: **for** $i = 0$ **to** m **do**
17: **for** $j = 0$ **to** m **do**
18: $denominador \leftarrow \sqrt{G_{ii} * G_{jj}}$
19: $R_{ij} \leftarrow \frac{G_{ij}}{denominador}$
20: **end for**
21: **end for**
22: **Return** R

Por último, se calcula el promedio de cada fila de R y se lo almacena en un vector *prom_filas*, al que se le calcula nuevamente el promedio, para así obtener la medida de similaridad de las imágenes.

Algorithm 11 promedio_simil *(in: R)*

1: $n \leftarrow dimension(R)$
2: $prom\_filas \leftarrow vectorVacio()$
3: **for** $i = 0$ **to** n **do**
4: $promedio \leftarrow \frac{\sum_{j=0}^{n} |R_{ij}|}{n}$
5: $prom\_filas.append(primedio)$
6: **end for**
7: $res \leftarrow \frac{\sum_{i=0}^{n} prom\_filas_i}{n}$
8: **Return** res

2.5. Error de compresión

Para medir la calidad de las imagenes reconstruidas, se usa el error cuadratico medio. Se calcula la diferencia entre los pixeles de la imagen original con la imagen reconstruida y se eleva al cuadrado. Luego se hace la raíz cuadrada del promedio de las diferencia.

Algorithm 12 error_cuadratico_medio *(in: X, X_rec)*

1: $error \leftarrow X - X_{Rec}$
2: $error \leftarrow error * error$
3: $error \leftarrow \sqrt{promedio(error)}$
4: **Return** $error$

3. Resultados y Discusión

Para verificar el correcto funcionamiento de la implementación del método de la potencia, se ideó un test que verificara si los autovalores calculados toman el valor correcto. Para esto, se generaron matrices aleatorias simétricas mediante la librería de Python numpy. Se eligieron este tipo de matrices debido a las propiedades de los autovalores que aseguran un buen funcionamiento del método de la potencia. Se calcularon los autovalores y autovectores con el método propio y después se compararon con los autovalores y autovectores proporcionados por la función de numpy. Además, se observó el comportamiento del método cuando no se cumplen las precondiciones del mismo. Se ideó una matriz diagonal con elementos repetidos, lo que sucedió fue que los autovalores que no eran repetidos se encontraron bien, mientras que los repetidos no fueron calculados de manera correcta. Por último, se probó un caso con una matriz totalmente aleatoria donde el método no pudo encontrar correctamente los autovalores y autovectores.

El método de la potencia con deflación es un algoritmo computacionalemente costoso. La complejidad de encontrar un único autovalor es $\mathcal{O}(n^2)$ debido a que se realiza una multiplicación de un vector x de tamaño n, con una matriz A de tamaño nxn. Este proceso se repite por cada autovalor que se busca encontrar, es decir, k veces. De esta forma la complejidad teórica del algoritmo resulta de $\mathcal{O}(kn^2)$.

En cuanto al PCA, su complejidad depende del cálculo de la matriz de covarianza, el cálculo de sus autovectores (que, por lo dicho previamente, es $\mathcal{O}(kn^2)$), y por último por la proyección de la imagen. Si llamamos m a la cantidad de imágenes, n a la cantidad de pixeles de cada una y k a la dimensión de la proyección, este primer paso toma $m * n$ operaciones para centrar los datos y otras $m * n^2$ para calcular la matriz. Luego, la proyección toma $k * n$ operaciones. Es decir, la complejidad total del algoritmo es de $\mathcal{O}(kn^2 + mn^2)$.

En comparación, el calculo de la matriz de covarianza del 2DPCA y de sus autovectores resulta mucho más barato ya que esta tiene la misma dimensión que las imágenes en su forma matricial. Si llamamos M a la cantidad de imágenes y, $a \times b$ a la dimensión de cada una, el cálculo de la matriz de covarianza toma $M * a * b$ operaciones en centrar los datos y otras $a * b^2$ en calcular la matriz. Como se puede observar $a * b$ es la cantidad de pixeles de una imagen, es decir n, por lo que estas complejidades quedan de $M * n$ y $n * b$ respectivamente. Luego, el cálculo de autovalores toma $k * b^2$ y la proyección otras $k * n$ operaciones. Como las imágenes con las que se operó en este trabajo eran de formato retrato, $a > b$ y la complejidad resulta de $\mathcal{O}(Mn + bn)$

Como se puede inducir por el análisis previo, la diferencia temporal en la ejecución de los dos métodos es muy notoria. Esto se debe a que el tamaño de la matriz de covarianza es mucho menor, por lo que es necesario calcular menos autovalores y estos, a su vez, tienen una dimensión también son más pequeños.

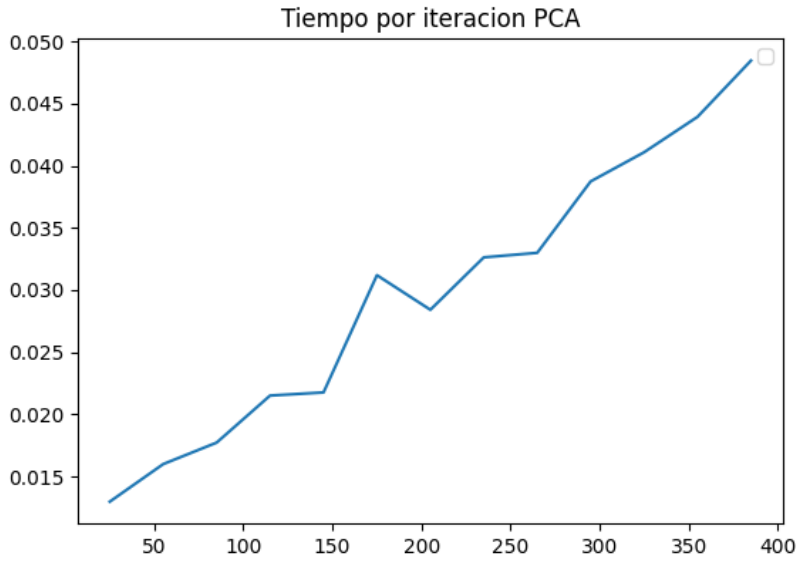

Figura 1: Tiempo de ejecución de PCA en base al tamaño de la matriz

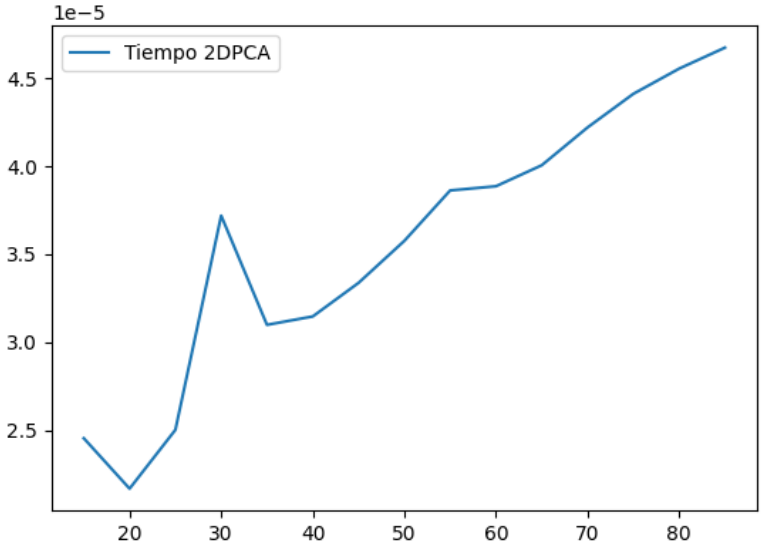

Figura 2: Tiempo de ejecución de 2DPCA en base al tamaño de la matriz

Estas conclusiones se ven reflejadas claramente en los gráficos 1 y 2. El segundo de estos fue diagramado en base a una ecala temporal mucho menor, reflejando asi el ahorro temporal generado al usar este segundo método.

El concepto principal en el que se basan ambos métodos, es en el hecho de que ciertas componentes o direcciones son las que aportan la mayor cantidad de información sobre los datos, mientras que el resto solo aportan detalles. Esto es muy evidente al observar las magnitudes de los autovalores de la matriz de covarianza ordenadas de mayor a menor. Como indican las figuras 3 y 4, hay un punto a partir del cual los autovalores no aportan casi nada información. En el caso de PCA esto ocurre al rededor del autovalor número 80 mientras que en 2DPCA al rededor del 20.

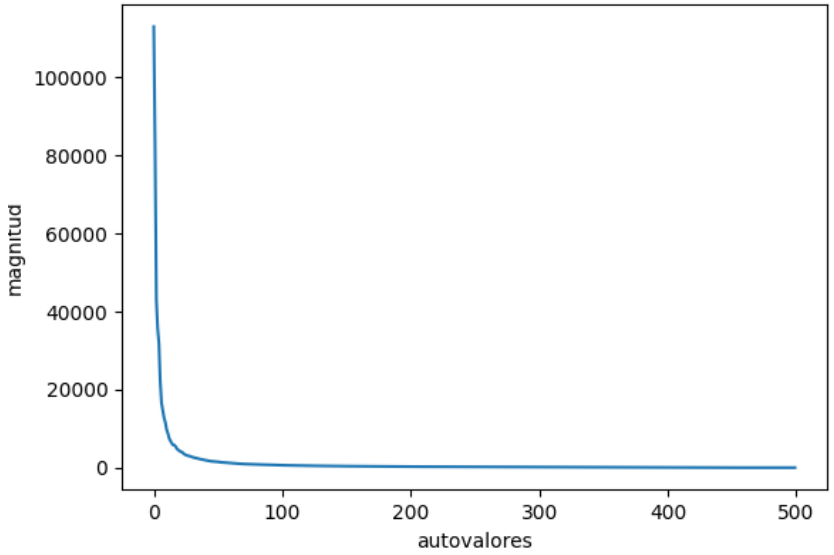

Figura 3: Magnitud de los autovalores de la matriz de covarianza PCA

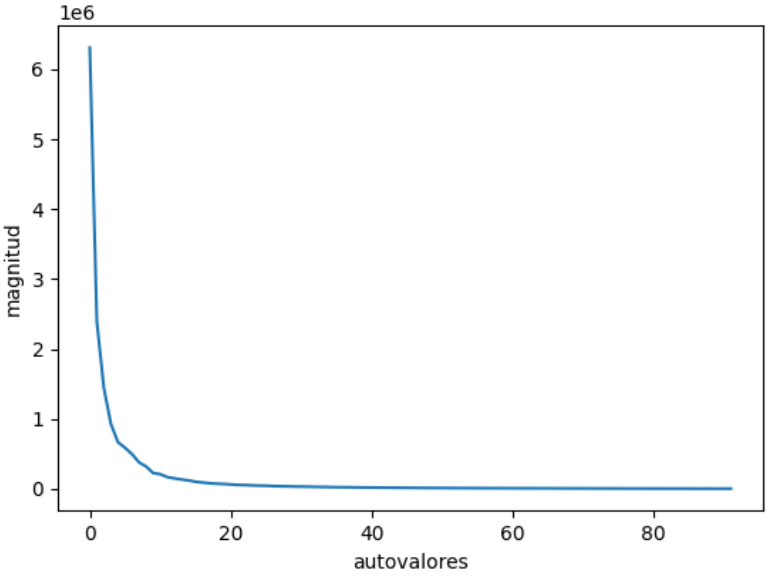

Figura 4: Magnitud de los autovalores de la matriz de covarianza 2DPCA

Otra forma de observar esta diferencia entre ambos métodos es a partir del error generado por las imágenes comprimidas reconstruidas. Tanto en la figura 5 como en la 6 se puede observar que cuanto mayor sea la dimensión del espacio de proyección más precisa es la reconstrucción. Aún así, el aumento de presicion se estanca en la misma cantindad de autovectores que los destacados en el análisis de las figuras anteriores.

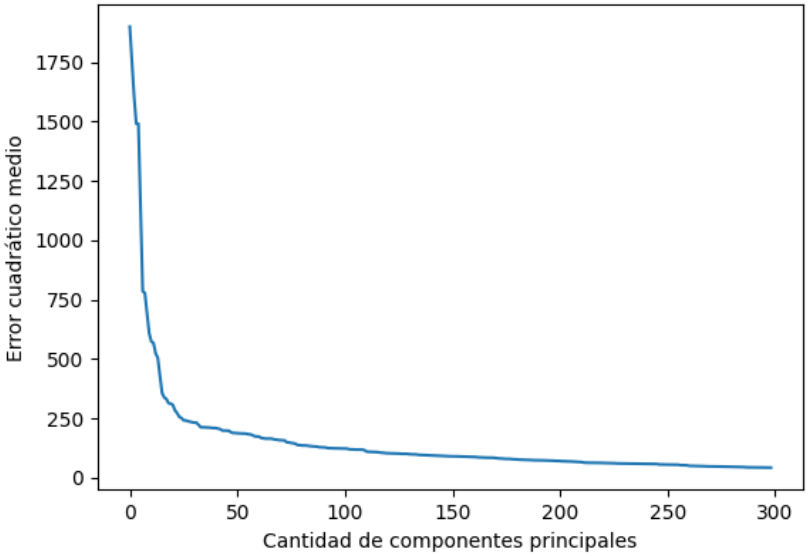

Figura 5: Promedio error para distintos autovectores (PCA)

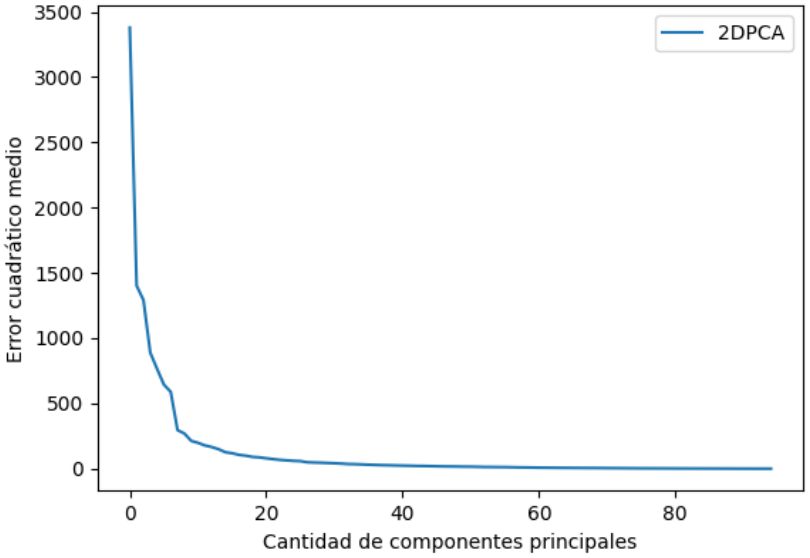

Figura 6: Promedio Error para distintos autovectores (2DPCA)

A partir de estas mediciones, se puede concluir que el método de 2DPCA no solo ahorra tiempo de cómputo a la hora de cálcular las componentes principales de las imágenes, sino que también requiere de un espacio de menor dimensión para conseguir una aproximación de suficiente calidad. Es decir, el costo de la proyección y de la descompresión también se verá reducido.

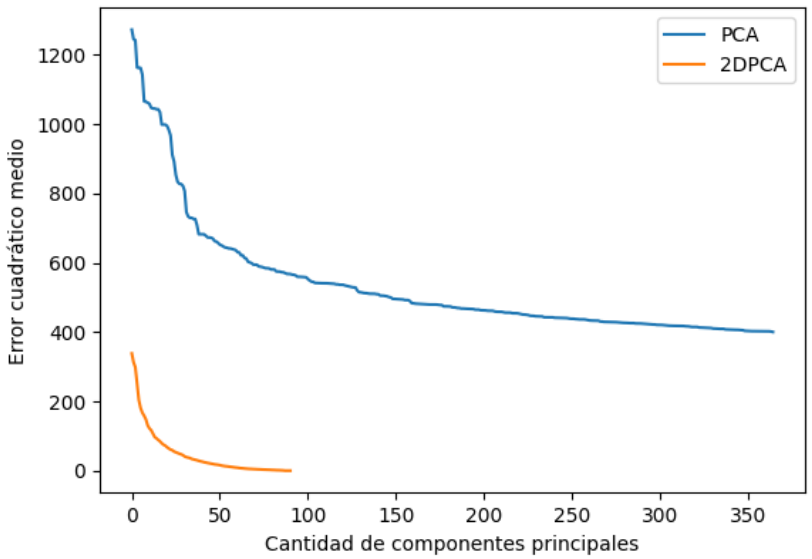

Figura 7: Promedio Error para distintas compresiones

En la figura 7 se visualiza la diferencia entre el error de compresión generado por PCA y 2DPCA. Como se puede observar, el segundo de estos es mucho menor. Esto se debe a que, al generar la matriz de covarianza basandose en matrices y no vectores, considera también las correlaciones entre las filas y las columnas de la imagen, lo que le pemite capturar las características no lineales presentes en los datos. De forma contraria, PCA basa su matriz de covarianza en estructuras de una dimensión y asume que las correlaciones solo existen en una dirección lineal, lo que limita su capacidad para capturar movimientos no lineales en la imagen.

De esta forma, el método de 2DPCA no solo es conveniente en cuestiones de ahorro twmporas sino que también en su exactitud en la compresión, como bien indican las figuras **??** y 9. En ambas métricas los valores tomados por la segunda de estas métodologías son mucho menores que los de la primera.

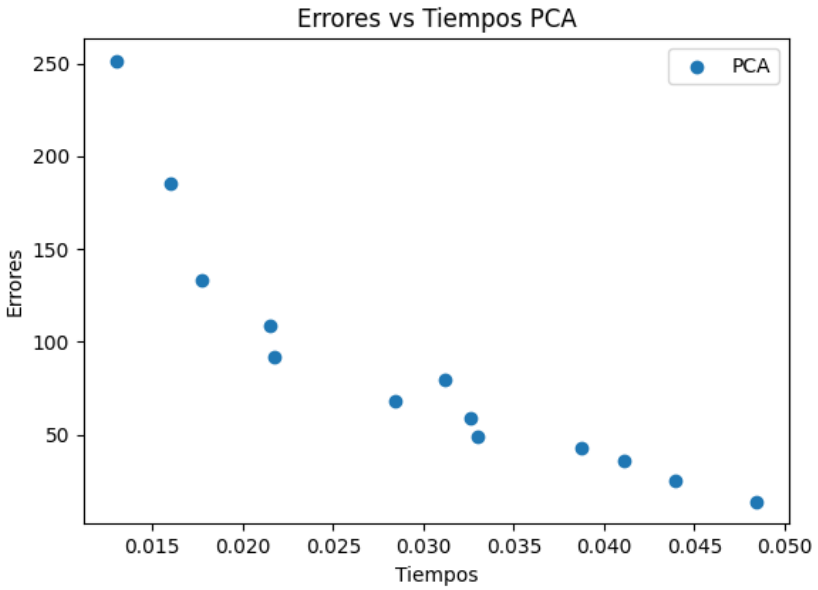

Figura 8: Error generado por el PCA en base al tiempo de ejecución

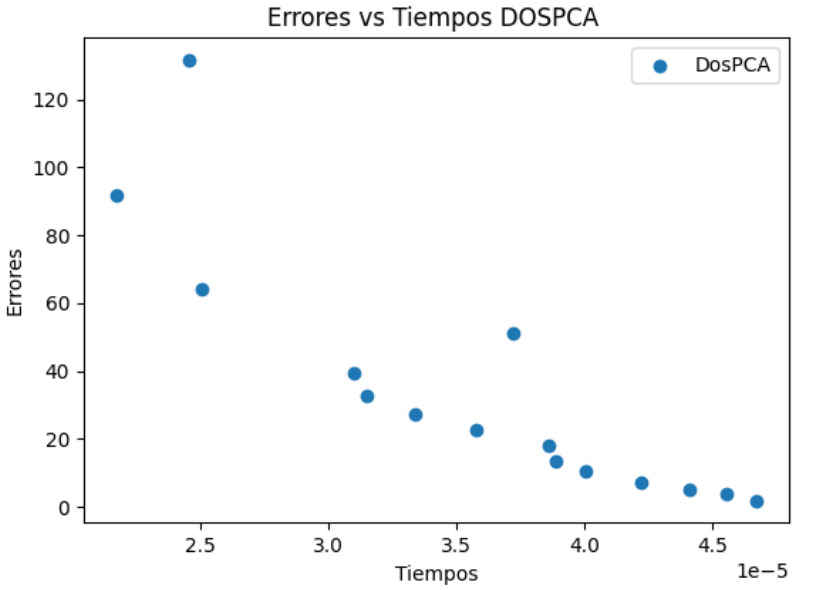

Figura 9: Error generado por el 2DPCA en base al tiempo de ejecución

Esta diferencia implica que el método de PCA va a tener una mayor dificultad a la hora de tener que evaluar variaciones que no eran parte del conjunto de entrenamiento, mientras que el modelo generado por 2DPCA podrá reconstruir dicha imágen con un error menor. Esto mismo visualiza la figura 10. En la fila de arriba se muestran imágenes comprimidas y reconstruidas con un modelo generado por PCA y en la de abajo por 2DPCA. Las imágenes de la izquierda muestran como el error aumenta cuando la imágen no se usó al entrenar el modelo, pero con 2DPCA esta sigue siendo reconocible.

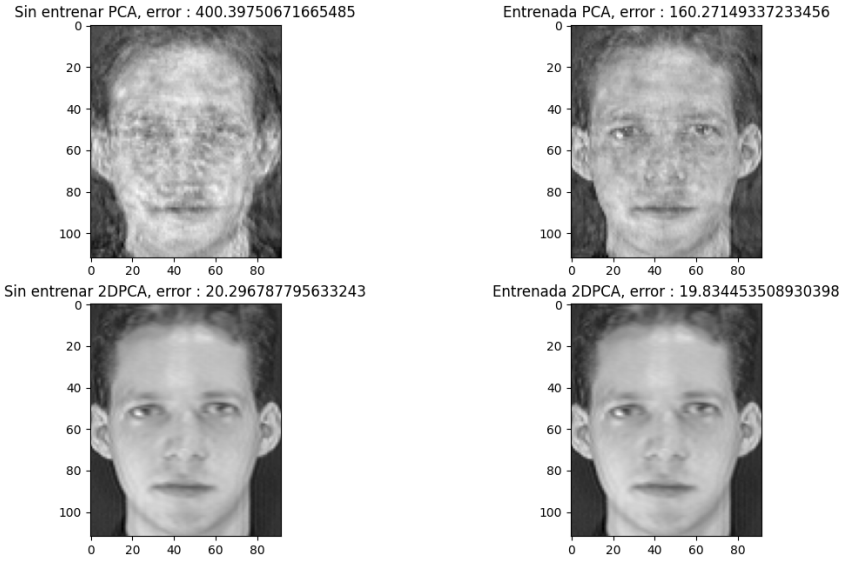

Figura 10: Caras reconstruidas con ambos métodos, formando parte y estando por fuera del conjunto de entrenamiento

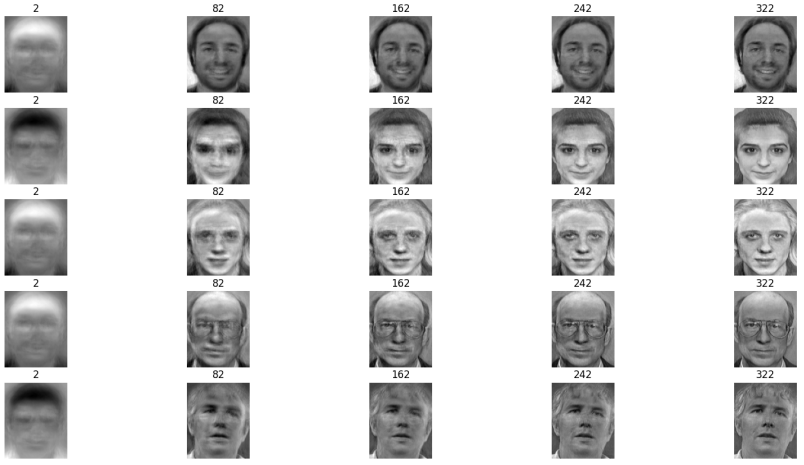

Figura 11: Reconstrucción con distintos k de caras PCA

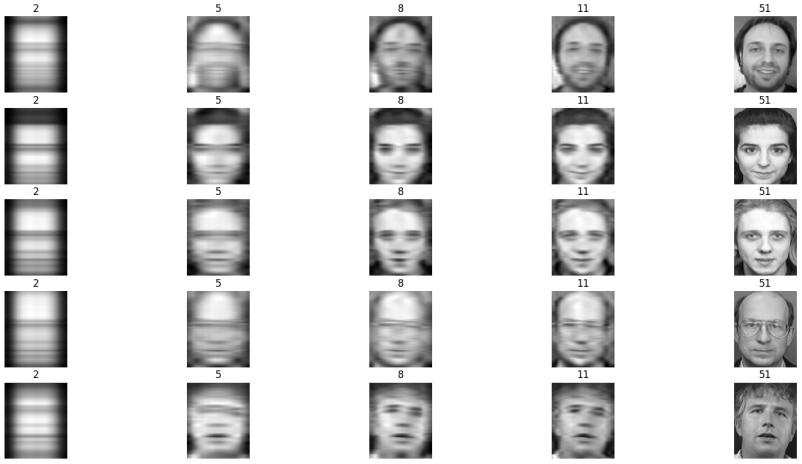

Figura 12: Reconstrucción con distintos k de caras usando 2DPCA

En los graficos 11 y 12 se observan las reconstrucciones de imágenes de distintas personas en base a un rango de componentes principales. Como se podía observar en las figuras 3 y 4, se pueden usar una porción pequeña de los autovalores totales para conseguir una imagen simil a la original. Nuevamente, se resalta la diferencia en la cántidad de componentes necesarias para reconstruir la imagen con cada método, con 2DPCA son necesarios menos de 10 vectores característicos para visualizar algo semejante a una cara, mientras que con PCA hacen falta al menos 80.

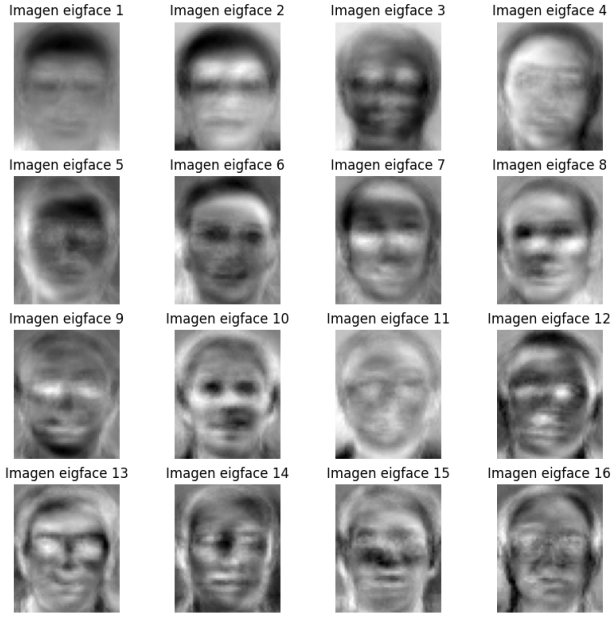

Figura 13: eigen faces PCA

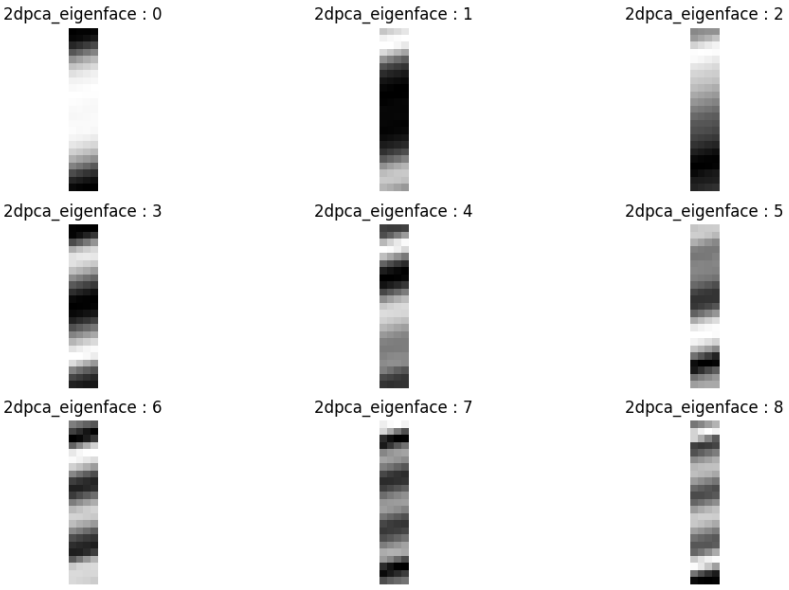

Figura 14: eigen faces 2DPCA

En el gráfico 14 se pueden observar las denominadas eigenfaces. Estos son los componentes principales del conjunto de datos. En el caso de PCA estos son del mismo formato que tienen las im'agenes a la hora de ralizar las operaciones, por lo que mantienen la forma de caras. En cambio, los vectores de proyección de 2DPCA no denotan nada en particular.

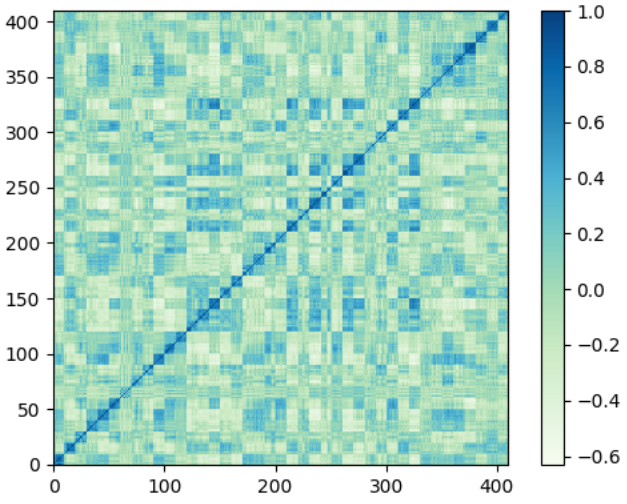

Figura 15: Matriz de similaridad de los datos originales centrados

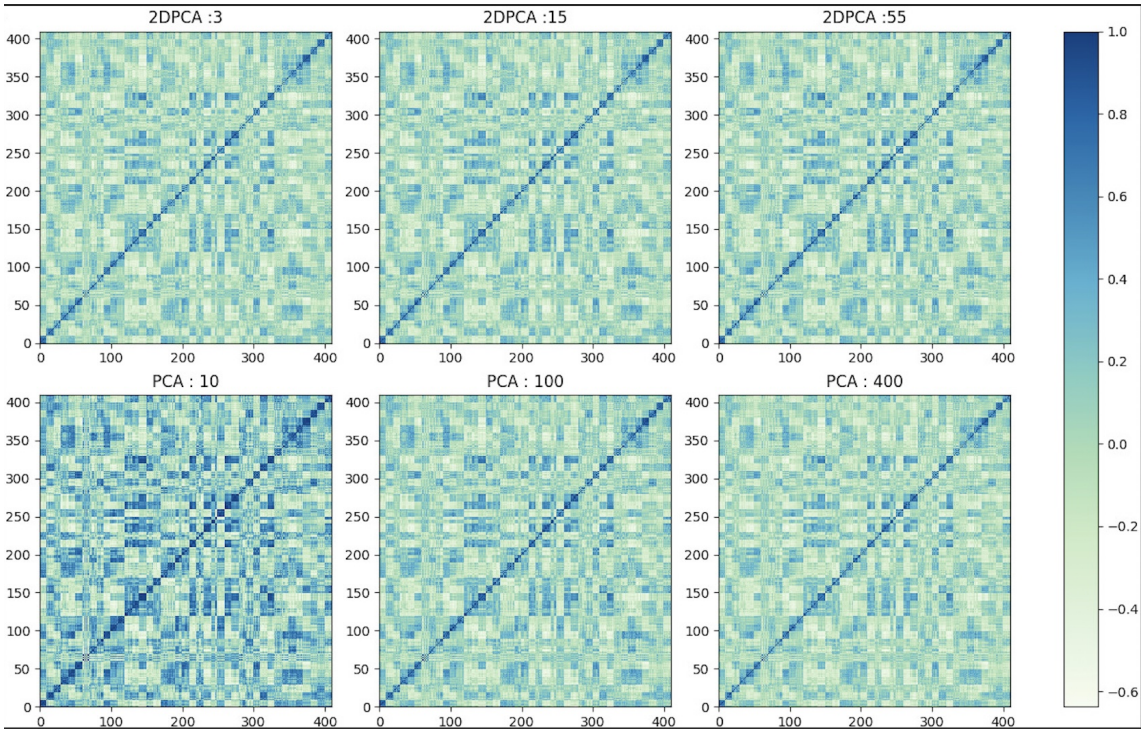

Figura 16: Matriz de similaridad para diferentes k

Las figuras 15 y 16 son visualizaciones de la matriz de similitud, primero para los datos originales centrados y después para distintas compresiones de estos con cada uno de los métodos. Se puede observar que se genera una distribución por bloques, ya que el conjunto está ordenado por persona y, al comparar una persona con sigo misma, la similaridad es mayor. Esto se puede ver especialmente en la antidiagonal, en donde la similaridad es máxima ya que se compara a una imagen con sigo misma. Por otro lado, al comparar las diferentes compresiones, se ve cómo la similaridad disminuye en cuanto la calidad de la aproximación aumenta. Esto tiene sentido, ya que cuantas más componentes se agreguen, más detalles de la imagen son tomados en cuenta. Las imágenes regeneradas en base a pocos autovalores solo permiten visualizar algo semejante a una cara, por lo que las imágenes, teniendo menos rasgos distinctivos, se van a parecer más, así como se muestra en las figuras 11 y 12.

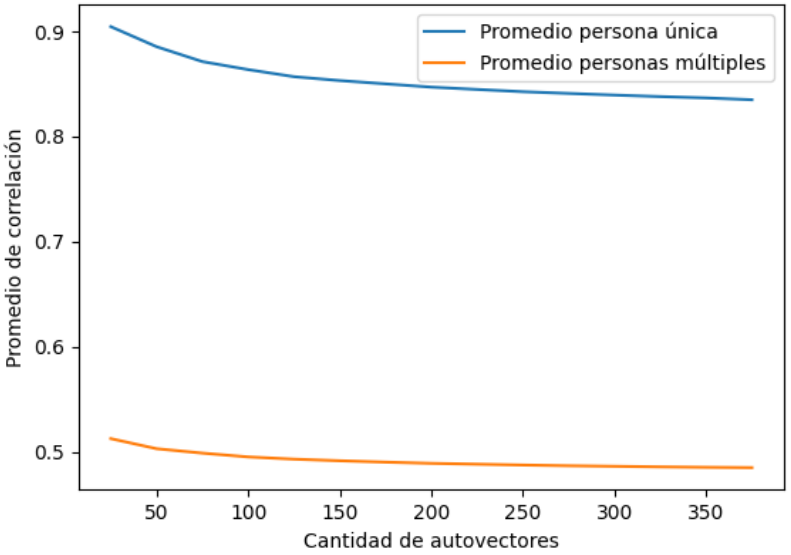

Figura 17: Medida de similaridad PCA

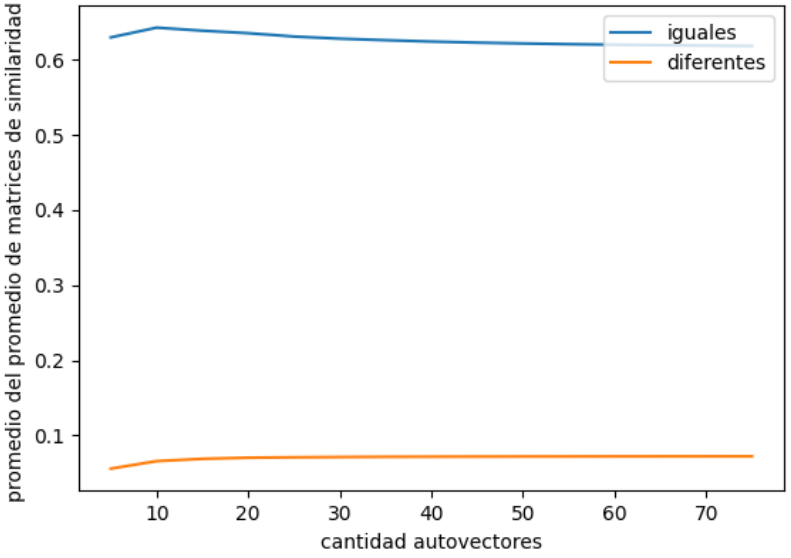

Figura 18: Medida de similaridad 2DPCA

Esto mismo denotan las figuras 17 y 18, que muestran la medida de semejanza entre diferentes compresiones de imágenes de la misma persona y de personas diferentes. En ambos casos, la similaridad disminuye cuanto más aumenta la calidad de las compresiones.

Estas figuras también permiten visualizar que ambos métodos generan un mecanismo de clasificación de imágenes. Hasta cuando la resolución de las compresiones es alta, se hace una diferenciación clara entre la comparación de imágenes de la misma persona e imágenes de personas distintas. De esta forma, sería posible comprimir una imágen y, mediante la média de similaridad, distinguir a que grupo de rostros pertenece.

4. Conclusiones

Tanto el método PCA como el 2DPCA permiten disminuir la dimensión de imágenes manteniendo la habilidad de regenerarlas de forma aproximada cuando se lo desee. Además permiten realizar una categorización de las imágenes según su similitud, es decir, reconocer a qué persona le corresponde la imágen de un rostro. Pero, a través de múltiples métricas, se observó que el segundo de estos algoritmos tiene muchos beneficios por sobre el otro.

En un principio, este presenta una reducción importante del costo computacional, permitiendo realizar estas compresiones de manera mucho más veloz. También implica un ahorro de espacio de memoria, ya que es necesario almacenar una matriz de covarianza de menor dimensión y, por ende, menos autovectores con menos elementos cada uno.

Además, al evaluar la covarianza sin modificar la estructura original de los datos y al tener que calcular una menor cantidad de autovectores, el 2DPCA genera un error más insignificante que el del PCA. Esto habilita la posibilidad de usar el modelo entrenado bajo un conjunto particular para comprimir imágenes por fuera de este y que la imágen siga siendo reconocible.

5. Bibliografía

1. Capítulo 12. Bishop, C. M. & Nasrabadi, N. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p.738). New York: springer.

2. Capítulo 9. Burden, R. L., Faires, J. D., & Burden, A. M. (2017). Análisis numérico (10.ª ed.). Cengage Learning.

3. Yang, Jian & Zhang, David & Frangi, Alejandro & Yang, Jing yu. (2004). Two-Dimensional PCA: A New Approach to Appearance-Based Face Representation and Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence. 26, 131-137.

4. Safayani, M., Manzuri Shalmani, M.T., & Khademi, M. (s.f). Extended Two-Dimensional PCA for Efficient Face Representation and Recognition. Computer Engineering Department of Sharif University of Technology.