Proyecto final: Análisis de sargazo

Reconocimiento de Patrones - 0757

Facultad de Ingeniería,

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Martínez Ostoa N.I.
#315618648
Ing. en Computación
Facultad de Ingeniería, UNAM
Ciudad de México, México
nestor.martinez@iimas.unam.mx

Ramírez Bondi J. A.
#314634825
Ing. en Computación
Facultad de Ingeniería, UNAM
Ciudad de México, México
alejandrobondi@me.com

Profesores:
Dr. Boris Escalante Ramírez
Dra. Jimena Olveres Montiel
I.I.M.A.S. - U.N.A.M.

Resumen—

I. OBJETIVO

- Realizar un análisis de imágenes tomadas en distintos momentos de una playa en Puerto Morelos, QR para detectar la presencia de sargazo en el océano y ayudar en la estimación del arribo de este elemento que afecta al turismo y demás actividades económicas.
- A partir del análisis, realizar alguna de las siguientes actividades concretas:
  - Detección del sargazo
  - Segmentación de las manchas del sargazo
  - Seguimiento del movimiento del sargazo
  - Cuantificación, en píxeles, del sargazo.

II. INTRODUCCIÓN

La llegada atípica de sargazo a las playas mexicanas y el Caribe ha provocado afectaciones económicas significativas en las entidades cuya economía está mayoritariamente basada en los ingresos por actividades turísticas.

II-A. SVD

Uno de los componentes principales del algoritmo PCA es la descomposición en valores singulares o *singular value decomposition* (SVD, por sus siglas en inglés). La idea principal de SVD es descomponer una matriz (en este caso una imagen) en matrices más pequeñas de rango uno. Esto se hace a través de la siguiente ecuación:

$$A = U\Sigma V^T \tag{1}$$

en donde los vectores en U y V se conocen como vectores singulares izquierdos y vectores singulares derechos, respectivamente. Las dimensiones de cada matriz son:

$$A_{n\times p} = U_{n\times n}\Sigma_{n\times p}V^T_{p\times p} \tag{2}$$

SVD aplicado a imágenes tiene importancia en la compresión de la información. Es decir, poder mandar la misma calidad de la imagen con mucha menos información. Esto se hace mediante las submatrices de rango 1 que SVD encuentra. Por ejemplo, si tomamos en consideración la siguiente matriz de rango 1 A:

$$A = \begin{bmatrix} a & a & c & c & e & e \\ a & a & c & c & e & e \\ a & a & c & c & e & e \\ a & a & c & c & e & e \end{bmatrix}$$

En lugar de mandar A, la forma más eficiente de transmitir A es mandando lo siguiente:

$$A = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} a & a & c & c & e & e \end{bmatrix}$$

La cual se puede expresar como $A = u_1 v_1^T$. Cuando aumentamos a una matriz de rango 2, la ecuación anterior se convierte en: $A = u_1 v_1^T + u_2 v_2^T$. Ahora, sabiendo cómo podríamos obtener una posible descomposición de A, la pregunta es: ¿hemos escogido la mejor opción de vectores u y v? Una pista para poder contestar esta pregunta es que los vectores en U deben ser ortogonales con los vectores en V, pues esto producirá términos $c_2 u_2 v_2^T$ con menor valor. En tanto, SVD escoge términos de rango uno según el orden de importancia. Estos órdenes de importancia corresponden con las componentes principales de la imagen en cuestión.

Lo anterior, nos permite llegar a una técnica de aprendizaje de máquina de tipo estadístico no supervisado que utiliza una transformación ortogonal para llevar un conjunto de variables de un espacio lineal, las cuales seguramente tienen cierta

correlación, a uno en el que se logre separarlas. Es decir, se busca que ya no estén correlacionadas en el nuevo espacio. Más precisamente, el método se puede resumir como uno en el que se maximiza la varianza de las variables del conjunto de datos sobre sus componentes principales.

Aunque generalmente esta técnica es utilizada para reducir las variables que describen un conjunto de datos, también se puede utilizar para la obtención de imágenes conocidas como *eigenfaces*, la cual busca generar una o varias imágenes que representen cualquiera de las imágenes del conjunto bajo análisis original.

III. Desarrollo

Para esta práctica ocupamos las imágenes recopiladas en el proyecto *MIT Faces Recognition Database*, gracias al trabajo de [3]. Esta base de datos cuenta con $3,993$ imágenes de rostros de distintas identidades en escala de grises. Un ejemplo se muestra en la figura 1.

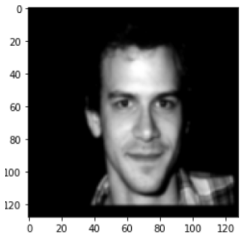

Figura 1: Ejemplo de imagen del proyecto *MIT Face Recognition Project*

III-A. Preprocesamiento de imágenes

El primer punto es encontrar las dimensiones de cada imagen, las cuales tienen un tamaño de 128×128 pixeles. Posteriormente, hacemos que todas las imágenes se acoplen a estas dimensiones como se muestra a continuación:

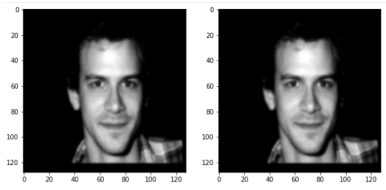

Figura 2: Imagen acoplada a una matriz de 128×128

Una vez que tenemos todas nuestras imágenes con el mismo tamaño, lo que hacemos es encontrar la imagen promedio:

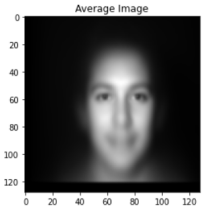

Figura 3: Imagen promedio tomada como la media aritmética de cada pixel entre todos los pixeles de todas las imágenes del dataset

La figura 3 se obtiene de realizar el promedio de todas las imágenes del dataset para un pixel en específico. Una vez que tenemos esta figura, lo que hacemos es centrar todas las imágenes mediante la resta de los valores actuales menos los valores de la imagen promedio:

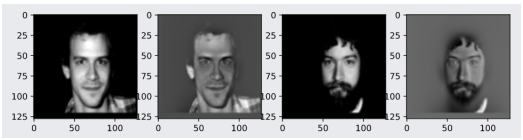

Figura 4: Imágenes de caras centradas: resta entre los valores originales y la cara promedio

III-B. PCA

Para el análisis de componentes principales, empleamos la descomposición por valores singulares para obtener las componentes en orden de importancia. Dichas componentes se almacenan en la matriz σ de la ecuación 1.

Al aplicar SVD sobre todo el dataset obtenemos las dimensiones de las matrices de la ecuación 1. Los resultados se muestran en la tabla I.

Matriz	Dimensiones
U	3993×3993
Σ	3993
V	3993×16384

Cuadro I: Dimensiones de las matrices U, Σ y V

Podemos observar que se mantienen las dimensiones de cada matriz de la ecuación **??**. Ahora, para visualizar la variablidad explicada por SVD, podemos recurrir a la figura 5 en la cual podemos observar que a partir de los primeros 5 valores singulares, ya no hay variación en los datos. Esto, relacionado con la teoría de SVD, nos indica que podríamos comprimir la información simplemente con 5 componentes principales y obtener una muy buena aproximación de la información en las caras.

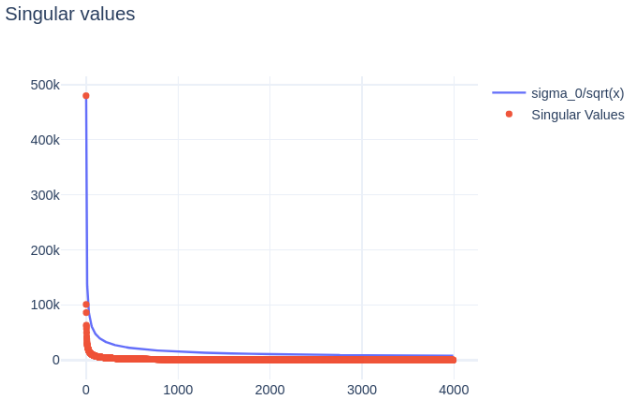

Figura 5: Nivel de variabilidad explicada por cada uno de los 3993 valores singulares

En la figura 12 podemos observar los rasgos explicados por las 5 componentes principales. A destacar que cada una de las componentes principales explica un rasgo único en las caras.

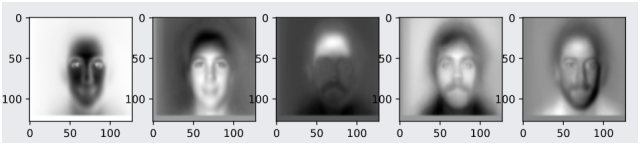

Figura 6: Rasgos de las caras explicados por las primeras 5 componentes principales

Continuando con el tutorial y los ejercicios propuestos en [1], después se trabajó con la obtención de los *eigenfaces*. Para ello, primero se normalizó y se calcularon los componentes principales de PCA. Esto derivó en que—considerando que se tenían imágenes de tamaño 64×64 por lo que el tamaño de la matriz era de $4,096$—se redujera el tamaño del vector de componentes principales. Es decir, se proyectaron los valores de los pixeles sobre los 64 vectores resultantes.

Si observamos la varianza que se va acumulando componente a componente, obtenemos lo siguiente:

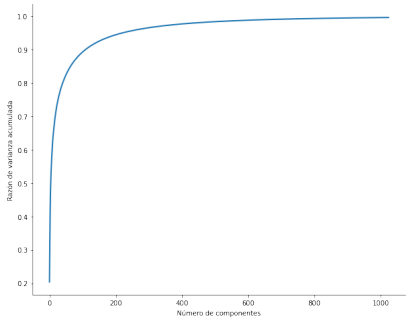

Figura 7: Acumulado de la varianza por componente calculado.

En la figura anterior, se muestra claramente que el $90\,\%$ de las varianzas de las imágenes bajo análisis se encuentran contenidas dentro de las primeras 64 componentes.

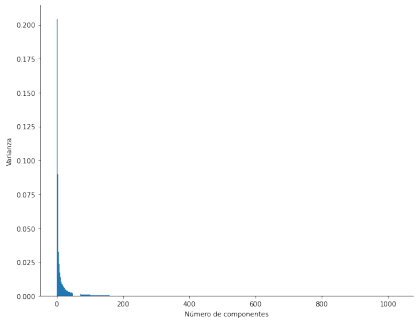

Figura 8: Distribución de la varianza dependiente del número de componentes principales considerado. Más del $90\,\%$ de los elementos se encuentran dentro de los 64 componentes.

Ya contando con el procesamiento inicial, se obtuvieron las imágenes promedio y de desviación estándar.

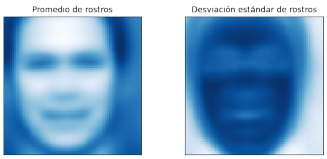

Figura 9: Caras generadas con el promedio y desviación estándar del total de rostros.

Por lo cual, ya es posible observar las *eigenfaces*, las cuales contienen o describen los elementos más importantes del conjunto original de rostros.

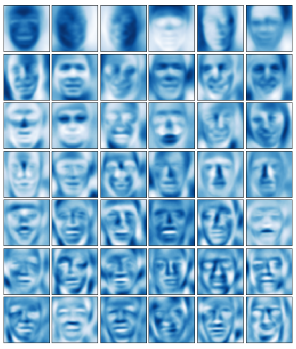

Figura 10: Se muestran las primeras *eigenfaces*.

Finalmnete, según lo que se ha incluido en la introducción, en teoría ya es posible reconstruir o generar, a partir de las *eigenfaces*, cualquiera de las imágenes originales en el dataset. Para ello, solo basta con generar una combinación lineal de las 64 imágenes de referencia basadas en el mismo número de componentes principales. [1]

[1]Esta función se trabajó a partir del método *inverse_transform* de scikit-learn.

Figura 11: Se muestran imágenes aleatorias regeneradas a partir de los vectores de componentes principales o *eigenfaces*.

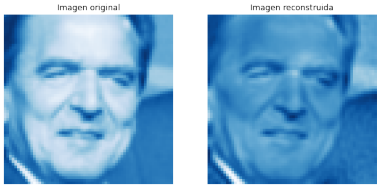

Figura 12: Estudio centrado en la reconstrucción de un rostro en específico a partir de la comparación de la imagen original y la obtenida después de realizar el procesamiento.

IV. Resultados

A partir del trabajo realizado, podemos considerar que se ha entendido plenamente el funcionamiento del método de análisis de componentes principales (PCA)—igual que el procesamiento inicial mediante SVD—para generar imágenes que, al ser transformadas linealmente por ciertos parámetros determinados, se pueden combinar para obtener cualquier imagen de una base de datos de fotografías de rostro o cualquier gráfico que se requiera inicial.

La técnica utilizada puede ayudar en la optimización y reducción del espacio para análisis de imágenes, almacenamiento y transmisión de datos gráficos. También, siendo la aplicación más importante de la técnica, este método es fundamental para el reconocimiento de patrones en los rostros y así poder identificar la identidad que se observa en una imagen.

V. Conclusiones

VI. Secciones de código relevante

VI-A. *Obtención de la cara promedio*

```
def get_avg_image(images):
    return np.mean(images, axis=0)
```

VI-B. *Centrado de imágenes*

```
def center_image(image, avg_image):
    return image-avg_image
```

VI-C. *SVD*

```
U, Sigma, VT = np.linalg.svd(X, full_matrices=False)
```

VI-D. *Variación de componentes principales*

```
x = np.arange(1, Sigma.shape[0])

fig = go.Figure()
fig.add_trace(go.Scatter(
    x=x,
    y=Sigma[0]/np.sqrt(x),
    name='sigma_0/sqrt(x)'
))
fig.add_trace(go.Scatter(
    x=np.arange(Sigma.shape[0]),
    y=Sigma,
    mode='markers', name='Singular Values'
))
fig.update_layout(
    template='plotly_white',
    title=dict(
        text='Singular values'
    ), width=600
)
fig.show()
```

VI-E. *Obtención de las n componentes principales*

```
def get_n_principal_components(n, VT):
    images = []
    d = int(np.sqrt(VT[0].shape[0]))
    for i in range(n):
        images.append(VT[i].reshape(d,d))
    return images
```

Referencias

[1] Sandipan D. EigenFaces and A Simple Face Detector with PCA/SVD in Python. sandipanweb, 2021. [Online]. Disponible: https://sandipanweb.wordpress.com/2018/01/06/eigenfaces-and-a-simple-face-detector-with-pca-svd-in-python/. [Visitado: 06- Aug- 2021]

[2] G. Strang, Introduction to Linear Algebra, Fifth edition. Wellesley: Wellesley-Cambridge Press, 2016.

[3] M. Szummer, "Face recognition project", Courses.media.mit.edu, 2002. [Online]. Disponible: https://courses.media.mit.edu/2004fall/mas622j/04.projects/faces/. [Accessed: 08- Aug- 2021]