

Asignación 5 - SVD & PCA: Clasificación de Imágenes

Métodos de Machine Learning 2025-I

Juan Sebastián Peñaloza Quintana
JsebastianDS@proton.me

Resumen

En este proyecto se llevó a cabo el planteamiento de un algoritmo simple de clasificación de imágenes. De un grupo de fotografías, se transformó a su forma vectorizada y, posteriormente, se analizó y clasificó mediante técnicas de reducción de dimensionalidad. Se utilizan dos metodologías matemáticas clave: **Descomposición en Valores Singulares (SVD)** y **Análisis de Componentes Principales (PCA)**

1. Introducción

El objetivo final es clasificar las imágenes en dos grupos (etiquetas '0' y '1') basándose en la proyección sobre el primer componente principal y visualizar los resultados en un gráfico. PCA es una técnica estadística que transforma los datos originales a un nuevo sistema de coordenadas donde los primeros ejes (componentes principales) representan la mayor parte de la varianza [1]. El proceso de PCA se resume en los siguientes pasos:

1. **Centrar los datos:** Restar la media de cada característica.
2. **Calcular la matriz de covarianza o la matriz de productos internos:** Se extraen las correlaciones entre las variables.
3. **Extraer autovalores y autovectores:** Los autovectores asociados a los mayores autovalores son las direcciones de máxima varianza.
4. **Proyectar los datos:** Se transforman los datos originales al espacio formado por estos autovectores.

La SVD descompone una matriz X en tres matrices: U , S y V^T de modo que:

$$X = U S V^T$$

Esta descomposición permite identificar las direcciones (componentes) de mayor variabilidad en los datos y es muy útil para tareas de compresión y reducción de ruido.

2. Problemas

Problema 1: Extraer la mayor

Carga de Imágenes:

Se recorre el directorio `./fotos/` para leer todos los archivos con extensión `.jpeg`.

Conversión a Escala de Grises:

Cada imagen se convierte a escala de grises para simplificar el análisis, ya que se trabaja con un solo canal de intensidad.

Reducción de Tamaño:

Se redimensiona la imagen a 75×75 píxeles, reduciendo la cantidad de datos y facilitando el procesamiento.

Vectorización:

Cada imagen es aplanada en un vector unidimensional utilizando la función `flatten()`. Todos estos vectores se almacenan en una matriz X , donde cada fila representa una imagen.

Descomposición SVD:

Se aplica la descomposición SVD a la matriz X , obteniéndose [4] U , S y V^T . La matriz V^T se utiliza para realizar la reducción dimensional.

Reducción a 2 Componentes:

Se proyectan los datos sobre los dos primeros componentes principales, facilitando la visualización en 2D.

Clasificación Simple: Se define un umbral basado en la media del primer componente principal. Si el valor de la proyección en el primer componente es mayor que la media, la imagen se clasifica como '1', y '0' en caso contrario.

Visualización y Conteo:

Se genera un gráfico de dispersión para visualizar la distribución de imágenes en el espacio PCA y se imprime el número de imágenes en cada categoría.

3. Desarrollo y código

CELDA 1: Importación de librerías necesarias

```
1 # %%
2 import os
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import cv2
```

- `os`: Para la gestión de directorios y archivos.
- `numpy`: Para operaciones matemáticas y manejo de matrices.
- `matplotlib`: Para la visualización gráfica de los resultados.
- `cv2` (OpenCV): Para cargar y procesar imágenes

CELDA 2: Carga, procesamiento y clasificación

```
1  # %%
2  def cargar_procesar_imagenes(directorio, nuevo_tam=(50, 50)):
3      imagenes_vectorizadas = []
4
5      for archivo in os.listdir(directorio):
6          if archivo.endswith(".jpeg"):
7              # escala de grises
8              ruta_imagen = os.path.join(directorio, archivo)
9              img = cv2.imread(ruta_imagen, cv2.IMREAD_GRAYSCALE)
10             if img is not None:
11                 # Reducir el tamaño
12                 img_redimensionada = cv2.resize(img, nuevo_tam)
13                 # Aumentar contraste usando equalización de histograma
14                 img_contraste = cv2.equalizeHist(img_redimensionada)
15                 # Vectorizar
16                 imagenes_vectorizadas.append(img_contraste.flatten())
17
18     return np.array(imagenes_vectorizadas)
19
20 # Cargar y procesar las imágenes de un sub-directorio
21 directorio = "./fotos/"
22 X = cargar_procesar_imagenes(directorio)
23
24 # "centering"
25 X_mean = np.mean(X, axis=0)
26 X_centered = X - X_mean / 255.0
27
28 # Se usa la matriz de productos internos
29 XtX = X_centered.T @ X_centered
30
31 # Calcular los vectores y valores propios de la matriz de productos internos
32 eigvals, eigvecs = np.linalg.eig(XtX)
33
34 # Ordenar los valores propios y vectores en orden descendente
35 indices_orden = np.argsort(eigvals)[::-1]
36 eigvals = eigvals[indices_orden]
37 eigvecs = eigvecs[:, indices_orden]
38
39 # Seleccionar los 2 primeros componentes principales para visualización y
40 # proyectar los datos en el espacio de los componentes principales
41 PCs = eigvecs[:, :2]
42 X_pca = X_centered @ PCs
43
44 umbral = np.mean(X_pca[:, 0])
45 etiquetas = (X_pca[:, 0] > umbral).astype(int)
46
47 plt.figure(figsize=(8,6))
48 scatter = plt.scatter(X_pca[:, 0], X_pca[:, 1], c=etiquetas, cmap='bwr', alpha=0.7)
49 plt.xlabel("Primer Componente Principal (PC1)")
50 plt.ylabel("Segundo Componente Principal (PC2)")
51 plt.title("Distribución de imágenes en el espacio PCA\n(rojo=1, azul=0)")
52 plt.colorbar(scatter, label="Etiqueta")
53 plt.grid(True)
54 plt.show()
55 # -----
56
57
```

```

58 # -----
59 # Cantidad de imágenes clasificadas como '1' y '0'
60 num_1 = np.sum(etiquetas)
61 num_0 = len(etiquetas) - num_1
62 print(f"Imágenes clasificadas como '1': {num_1}")
63 print(f"Imágenes clasificadas como '0': {num_0}")

```

- **Función cargar_procesar_imagenes:** Se define una función modularizada que recorre el directorio, carga las imágenes en escala de grises, reduce su tamaño y aplica equalización de histograma para mejorar el contraste. Finalmente, cada imagen se vectoriza y se almacena en una matriz.
- **Verificación de Carga:** Se verifica que se hayan cargado imágenes y se informa el número de imágenes y la cantidad de características (píxeles) de cada imagen.
- **Estandarización de Datos:** Se centran los datos restando la media de cada característica, lo cual es esencial para que PCA funcione correctamente, ya que asume que los datos tienen media cero.
- **Matriz de Productos Internos:** En lugar de usar la matriz de covarianza, se utiliza la matriz $X^T X$ para obtener los autovalores y autovectores [2], lo cual es una alternativa computacionalmente eficiente.
- **Cálculo y Ordenamiento de Autovalores y Autovectores:** Se calcula la descomposición espectral de la matriz de productos internos y se ordenan en orden descendente para identificar las direcciones de mayor varianza.
- **Selección y Proyección en Componentes Principales:** Se seleccionan los dos primeros autovectores [3] y se proyectan los datos en este nuevo espacio, lo que permite visualizar la estructura subyacente en 2D.
- **Clasificación y Visualización:** Se clasifica cada imagen en función del valor del primer componente principal (PC1) respecto a su media, y se visualiza la distribución en un gráfico de dispersión.

4. Resultados

Visualización: El gráfico generado (scatter plot) muestra la distribución de las imágenes en el espacio definido por los dos componentes principales. Los puntos rojos corresponden a las imágenes clasificadas como '1' y los azules a las clasificadas como '0'. Luego de la ejecución, en este caso, se obtuvo (ver Anexo **Fig.2**) que *Se han cargado 99 imágenes con 2500 características cada una*. El diagrama de dispersión **Fig.1** obtenido se muestra a continuación

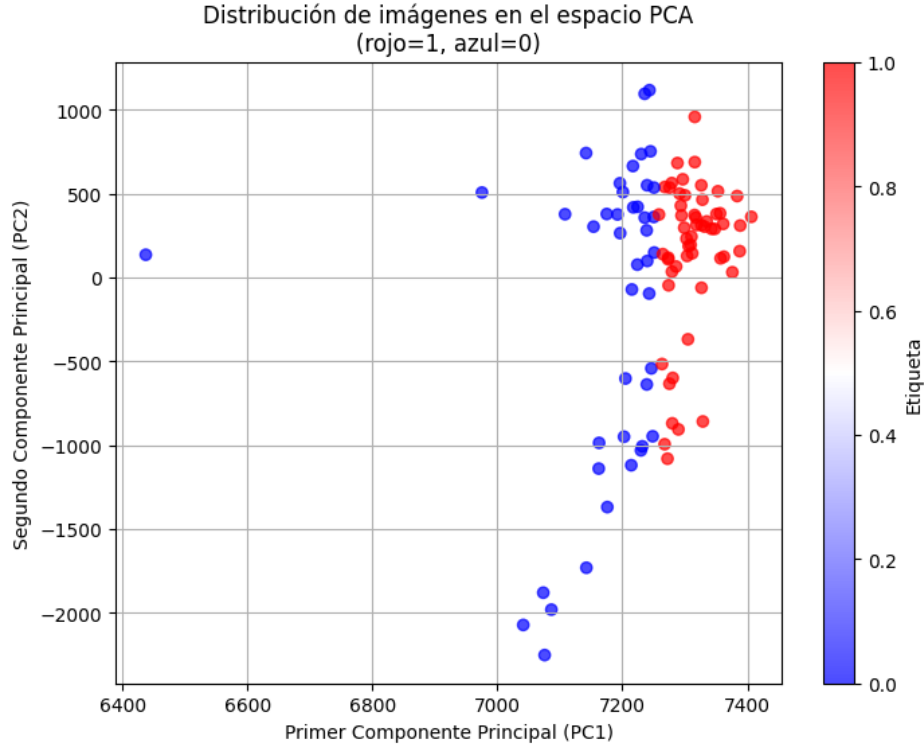


Figura 1

Conteo en cada categoría: Se imprime el número total de imágenes clasificadas en cada categoría, lo cual brinda una idea preliminar de la distribución y efectividad del umbral aplicado. Luego de la ejecución (ver Anexo **Fig.3**) se obtuvo como *salida* lo siguiente

- Imágenes clasificadas como '**1**': 56
- Imágenes clasificadas como '**0**': 43

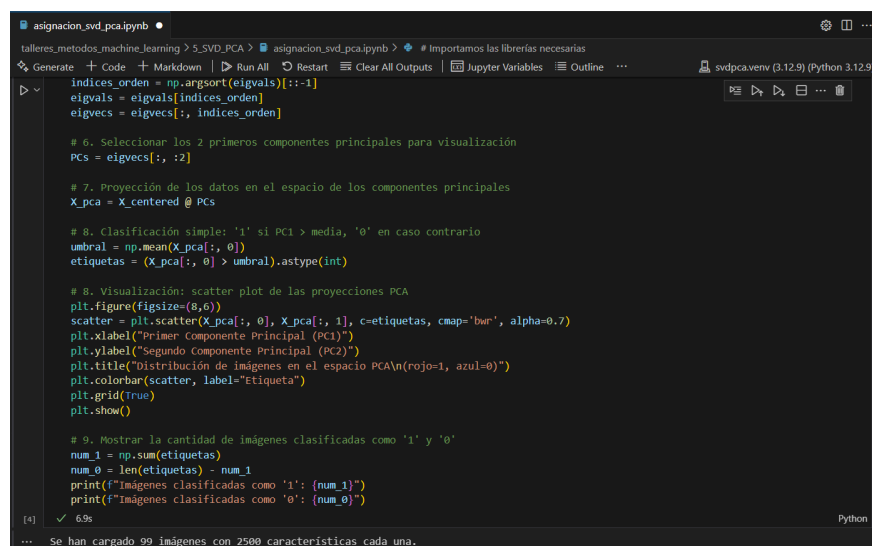
5. Conclusiones

El análisis realizado mediante SVD y PCA permitió transformar un conjunto de imágenes en vectores de características y reducir la dimensionalidad del problema para facilitar la visualización y el análisis. La aplicación de PCA permitió concentrar la mayor parte de la varianza en dos componentes principales, lo que facilita la visualización y la interpretación de los datos. La regla de clasificación basada en el primer componente principal permitió asignar de manera preliminar las etiquetas '0' y '1'. Aunque rudimentaria, esta metodología ilustra el uso de técnicas de reducción de dimensionalidad en tareas de clasificación. Respecto a la importancia del preprocesamiento, las técnicas de redimensionamiento, conversión a escala de grises y equalización del histograma fueron fundamentales para mejorar la calidad de las imágenes y, por ende, la efectividad de la extracción de características. Este enfoque puede ampliarse y combinarse con otros métodos de clasificación y algoritmos de aprendizaje automático para desarrollar sistemas de reconocimiento de imágenes más robustos y precisos.

Referencias

- [1] Goodfellow, I., Bengio, Y. & Courville, A. (2016). *Deep Learning*. MIT Press.
- [2] McKinney, W. (2017). *Python for Data Analysis (2nd ed.)*. O'Reilly Media. ISBN: 978-1491957660
- [3] NumPy Documentation (2023).numpy linear algebra. Matrix factorization problems.
- [4] Golub, G. H., & Van Loan, C. F. (2013). *Matrix computations (Vol. 3)*. JHU Press.

6. Anexos



```
indices_orden = np.argsort(eigvals)[::-1]
eigvals = eigvals[indices_orden]
eigvecs = eigvecs[:, indices_orden]

# 6. Seleccionar los 2 primeros componentes principales para visualización
PCs = eigvecs[:, :2]

# 7. Proyección de los datos en el espacio de los componentes principales
X_pca = X_centered @ PCs

# 8. Clasificación simple: '1' si PC1 > media, '0' en caso contrario
umbral = np.mean(X_pca[:, 0])
etiquetas = (X_pca[:, 0] > umbral).astype(int)

# 8. Visualización: scatter plot de las proyecciones PCA
plt.figure(figsize=(8,6))
scatter = plt.scatter(X_pca[:, 0], X_pca[:, 1], c=etiquetas, cmap='bwr', alpha=0.7)
plt.xlabel("Primer Componente Principal (PC1)")
plt.ylabel("Segundo Componente Principal (PC2)")
plt.title("Distribución de imágenes en el espacio PCA\n(rojo=1, azul=0)")
plt.colorbar(scatter, label="Etiqueta")
plt.grid(True)
plt.show()

# 9. Muestra la cantidad de imágenes clasificadas como '1' y '0'
num_1 = np.sum(etiquetas)
num_0 = len(etiquetas) - num_1
print(f"Imágenes clasificadas como '1': {num_1}")
print(f"Imágenes clasificadas como '0': {num_0}")
```

Se han cargado 99 imágenes con 2500 características cada una.

Figura 2: Pantallazo de la ejecución acerca de la carga y lectura de imágenes.

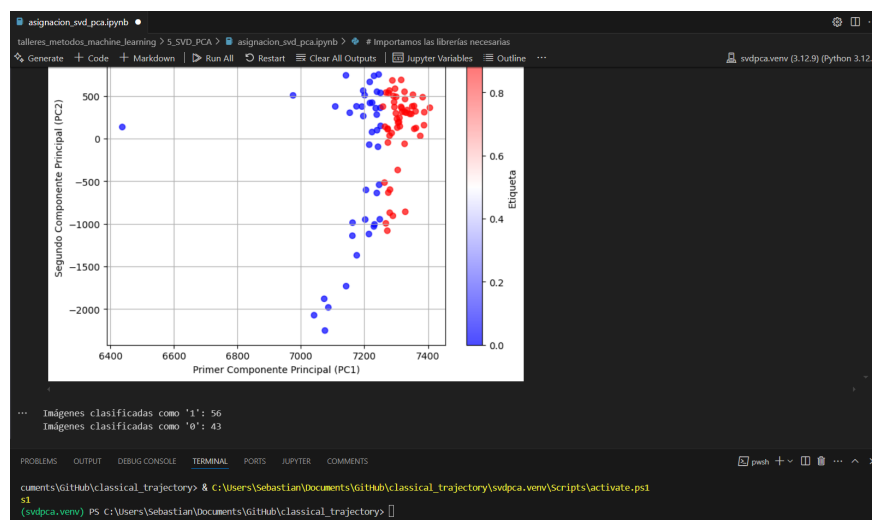


Figura 3: Pantallazo de la ejecución acerca del conteo resultante en el proceso de clasificación.