Álgebra Lineal Computacional

2do Cuatrimestre 2025

Laboratorio N° 8: Descomposición en Valores Singulares

Una matriz de $A \in \mathbb{R}^{m \times n}$ se puede descomponer en el producto de tres matrices $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$ y $\Sigma \in \mathbb{R}^{m \times n}$ tales que U y V son ortogonales, Σ es diagonal con elementos mayores o iguales a cero. A se calcula como

$$A = U\Sigma V^t$$

Cuando la matriz A tiene rango $0 \leq r \leq \min(n, m)$, es conveniente construir una representación reducida, a través de las matrices $\hat{U} \in \mathbb{R}^{m \times r}$, $\hat{V} \in \mathbb{R}^{n \times r}$ y $\hat{\Sigma} \in \mathbb{R}^{r \times r}$. Las columnas de $\hat{U}$ siguen siendo ortogonales y de norma 1, aunque no alcanzan para formar una base de $\mathbb{R}^m$ (lo mismo vale para $\hat{V}$ con $\mathbb{R}^n$). Las columnas de $\hat{U}$ y $\hat{V}$ corresponden a las primeras r columnas de U y V respectivamente. Y $\hat{\Sigma}$ corresponde al bloque principal de tamaño r de Σ. En este caso la expresión es análoga al caso anterior, reemplazando cada matriz por su versión reducida y obteniendo $A = \hat{U}\hat{\Sigma}\hat{V}^t$.

Ejercicio 1 (Descomposición SVD reducida)**.** Empezaremos construyendo una función que encuentre las matrices $\hat{U}$, $\hat{V}$ y $\hat{\Sigma}$. Para esto aplicaremos los siguientes pasos:

i. Dado que $A^tA = V\Sigma^t\Sigma V^t \in \mathbb{R}^{n \times n}$, los autovectores de A^tA son las columnas de V y los autovalores son los $\sigma_i^2 = (\Sigma_{ii})^2$. Por esto mismo, las columnas de $\hat{V}$ corresponden a los autovectores de autovalor no nulo de A^tA. Entonces, el primer paso es diagonalizar la matriz A^tA. Para esto empleen el método `diagRH` que construyeron en el laboratorio de autovectores. Opcionalmente (para ahorrar operaciones) el proceso puede detenerse cuando el autovalor obtenido es cero (dentro de la tolerancia deseada). De este paso entonces, obtenemos las matrices $\hat{V}$ y $\hat{\Sigma}$ (o, equivalentemente, los valores singulares $\sigma_i, i = 1, \ldots, r$.

ii. Teniendo la matriz $\hat{V}$, podemos encontrar $\hat{U}$ haciendo $A\hat{V} = B = \hat{U}\hat{\Sigma}$. Entonces $\hat{U}$ se obtiene normalizando las columnas de B.

iii. Un último recaudo a tener en cuenta, que nos ahorra trabajar con matrices de tamaños innecesariamente grandes: cuando $m > n$, es conveniente obtener la matriz $\hat{V}$ (de $n \times r$) primero dado que tiene un tamaño menor a $\hat{U}$ (de $m \times r$). Sin embargo, cuando $m < n$ es más conveniente encontrar primero $\hat{U}$. Para esto se repite el mismo procedimiento, pero sobre la matriz A^t. De esta forma, la matriz U cumple el rol de la matriz V en el caso anterior. Por esto último es conveniente revisar el tamaño de la matriz y trasponer antes de empezar el procedimiento.

Construyan a partir de lo mencionado una función `svd_reducida(A,tol=1e-15)` que devuelva las matrices $\hat{U}$, $\hat{\Sigma}$ y $\hat{V}$. La tolerancia debe utilizarse para identificar los autovalores iguales a cero de la matriz A^tA. La función debe tomar en cuenta las dimensiones de A y encontrar primero $\hat{U}$ o $\hat{V}$ según sea más conveniente. Por compatibilidad con `numpy`, la matriz $\hat{\Sigma}$ se devuelve como un vector que cuenta únicamente con los valores singulares.

Ejercicio 2 (Descomposición SVD completa)**.** En algunos casos puede ser de interés contar con una factorización completa. Mientras que el método `diagRH` nos permite obtener la matriz V directamente, el cálculo $AV = U\Sigma$ no nos ayuda ya que U podría tener más columnas que de V ($m > n$), y en caso de tener rango incompleto habrá columnas de V en el núcleo de A. Una forma de resolver este problema es el siguiente:

i. Repitan el procedimiento antes descrito para encontrar V a través de la diagonalización de A^tA.

ii. Computen $AV = U\Sigma = B \in \mathbb{R}^{m \times n}$. Dependiendo de las caracteristicas de B seguimos una combinación de estrategias:

i. Empezamos tomando las columnas de B como materia prima para la construcción de U. Si las columnas de B tienen norma 2 mayor a cero, entonces podemos encontrar n vectores normalizando las columnas de B. Si alguna columna de B tiene norma igual a cero, la descartamos. Supongamos que tenemos s vectores a esta altura, con $s \leq n$.

ii. Si $s < m$, necesitamos encontrar $m-s$ vectores ortogonales a los primeros s que hayamos. Para esto, empleamos el algoritmo de Gram-Schmidt visto en el laboratorio de factorización QR. Tomando como $q_1, \ldots, q_s$ a las s columnas normalizadas de que extrajimos de B, encontramos $m - s$ vectores ortogonales usando la base canónica de $\mathbb{R}^m$. De esta forma, empezando con e_1, armamos q_{s+1} como la proyección de e_1 en el espacio ortogonal al generado por $q_1, \ldots, q_s$. Si el q_{s+1} que obtuvimos al proyectar e_1 es nulo, lo descartamos. Repetimos este proceso agrandando el conjunto hasta llegar a los m vectores que buscamos.

Al igual que en el caso anterior, si $m < n$ es conveniente aplicar el procedimiento sobre A^t en vez de A, dado que permite trabajar con matrices más pequeñas inicialmente. Basandose en el procedimiento descrito, construyan una función `svd_completa(A,tol=1e-15)` que retorne las matrices U, Σ y V.

Ejercicio 3 (Descomposición SVD truncada)**.** Una de las propiedades más interesantes de la descomposición en valores singulares es que permite encontrar la mejor aproximación de la matriz A por una matriz A_k de rango k en el sentido de la norma 2. Esta consiste en encontrar los primeros k valores singulares, junto a las primeras k columnas de U y V. En general, el objetivo es usar $k \leq r$, el rango de A. Por lo tanto, cuando $k = r$ obtenemos la versión reducida de la factorización (ejercicio 1). Modifiquen la función que calcula la SVD reducida para incoporar el argumento k: `svd_reducida(A,k="max",tol=1e-15)` de forma tal que si `k="max"` compute la factorización reducida, y si no calcule los primeros k valores singulares distintos de cero con las correspondientes columnas de U y V. En caso de k sea mayor a r (es decir, que durante el procedimiento se encuentre un valor singular nulo), el procedimiento debe detenerse en ese punto, retornando la factorización reducida como en el Ejercicio 1.

Ejercicio 4 (Compresión mediante SVD)**.** En este ejercicio explorarán el uso de la SVD como herramienta para *comprimir* matrices.

a) **Rectángulo B&W**. Consideren la matriz provista en el archivo `matriz_rectangulo.py`. La misma representa la imagen de un rectángulo en colores blanco y negro.

i. Visualicenla con `plt.imshow(A, cmap = "gray")`. Antes de calcular nada más, identifiquen (visualmente) el rango de la matriz A. ¿Cuántas filas linealmente independientes tiene?

ii. Calculen la factorización SVD reducida de la matriz A y comprueben su estimación anterior. ¿Qué vectores conforman las columnas de $\hat{U}$ y $\hat{V}$? ¿Cuánto valen los valores singulares (no nulos) de A? ¿Todos tienen igual importancia?

iii. Construyan las aproximaciones de rango 1 y 2 de A usando la versión truncada de la SVD. Grafiquen las tres versiones. ¿Qué tipo de errores aparecen al aproximar por rangos menores?

b) **Arbol**. Consideren la imagen del árbol `tree2.jpg` adjunta con este archivo.

i. Lean y visualicen la imagen usando el siguiente código

```
# Leemos el archivo
img = io.imread('tree2.jpg')
print(img.shape)
plt.imshow(img)
# Lo convertimos a tonos de gris para simplificar la aplicaci n
imgGray = color.rgb2gray(img)
print(imgGray.shape)

plt.imshow(imgGray, cmap='gray')
```

ii. Apliquen la SVD reducida y grafiquen los valores singulares de la matriz `imgGray`. ¿Es la imagen de rango completo? Identifiquen cuántos valores singulares significativos hay.

iii. Grafiquen la aproximación de rango k de la matriz A para $k = 1, 10, 50, 100, 200$. Identifiquen visualmente un valor de k a partir del cuál la imagen se ve suficientemente bien y comparen con el acumulado de los valores singulares: $\phi_k = \frac{\sum_{i=1}^{k} \sigma_i}{\sum_{i=1}^{r} \sigma_i}$. ¿Qué porcentaje del total hay que alcanzar para que la imagen se vea razonablemente bien? Grafiquen la curva de ϕ_k en función de k, e identifiquen aquellos puntos en los que se alcanza el $50\%, 70\%, 90\%$. Visualicen la imagen en esos puntos.

iv. Repitan este procedimiento para las otras imágenes que acompañan este archivo (u otras de su preferencia). Identifiquen qué cantidad de valores singulares son necesarios para aproximar bien cada una. Caractericen cualitativamente qué tipo de detalles hacen que una imagen requiera más valores singulares para ser bien aproximada visualmente.

Análisis de componentes principales

Otra de las aplicaciones típicas de SVD es el cálculo de componentes principales en un dataset. La idea del análisis de componentes principales (PCA por sus siglas en inglés) se centra en aplicar SVD a la matriz de covarianza de los datos. La matriz de covarianza captura en su diagonal la varianza de cada uno de las variables de interés, y en sus elementos fuera de la diagonal captura cómo estas están relacionadas.

En este caso, nuestro punto de partida es un conjunto de n variables $X_1, \ldots, X_n$. Suponemos que tenemos m mediciones de cada una de ellas, x_{ij}, con $1 \leq i \leq m$ y $1 \leq j \leq n$ (es decir, x_{ij} es la medición i esima de X_j). Podemos acomodar estas mediciones en una matriz $A \in \mathbb{R}^{m \times n}$, con m el número de observaciones, y n el número de variables, y por lo tanto $A_{ij} = x_{ij}$.

El análisis de componentes principales empieza por calcular la versión centrada y escalada de A, donde

$$\tilde{A}_{ij} = \frac{x_{ij} - \mu_j}{\sqrt{\frac{1}{m-1} \sum_{i=1}^{m} (x_{ij} - \mu_j)^2}}$$

donde $\mu_j = \frac{1}{m} \sum_{i=1}^{m} x_{ij}$ es el promedio muestral, $(x_{ij} - \mu_j)^2$ es una estimación de la varianza muestral (y su raíz una estimación del desvio estandar). Entonces, $S = \tilde{A}^t \tilde{A} \in \mathbb{R}^{n \times n}$ es una estimación de la matriz de correlaciones de las variables aleatorias $X_1, \ldots, X_m$. Los autovalores de S son los valores singulares de $\tilde{A}$ al cuadrado, y nos permiten identificar qué combinaciones lineales de las variables $X_1, \ldots, X_n$ están asociadas a la mayor variabilidad de los datos.

Intuitivamente, si nuestro interés es reducir la dimensión de un dataset, entonces lo que queremos hacer es pasar de un conjunto de variables $X_1, \ldots, X_m$ a otro $Y_1, \ldots, Y_k$, con $k < m$, donde el conjunto de variables Y almacenan el grueso de información de las variables X. Usando SVD sobre la matriz $\tilde{A}$ (o diagonalizando la matriz S) podemos encontrar las variables Y que son combinación lineal de las X y capturan el grueso de la variabilidad de las X (que en este caso se asocia a la cantidad de información que aportan a un dataset).

Desde la perspectiva algebraica, podemos interpretar que estamos reteniendo las primeras k columnas de la matriz V, que corresponden a los primeros k valores singulares, y por lo tanto capturan el grueso de la varianza del conjunto de datos. Aquí consideraremos la aplicación de esta metodología a la clasificación de imágenes. Si partimos de m imágenes con n píxeles cada una (reacomodados en forma de vector concatenando sus filas, por ejemplo), podemos proyectarlas en k dimensiones mediante la matriz $\hat{V}_k \hat{V}_k^t$. Luego, una nueva imagen que consta de un nuevo conjunto de n píxeles puede ser asociada a alguna de las anteriores buscando la imagen en nuestro conjunto inicial que minimice la distancia a la nueva imagen.

Ejercicio 5 (EigenFaces)**.** Usando su implementación de descomposición en valores singulares, completen el notebook `eigenfaces.ipynb`, donde se considera el uso de la misma para el problema de reconocimiento facial

Módulo ALC

Para el módulo ALC, deben programar las siguientes funciones. Tests para las mismas acompañan en el archivo `L08-Tests.py`.

```python
def svd_reducida(A,k="max",tol=1e-15):
    """
    A la matriz de interes (de m x n)
    k el numero de valores singulares (y vectores) a retener.
    tol la tolerancia para considerar un valor singular igual a cero
    Retorna hatU (matriz de m x k), hatSig (vector de k valores singulares) y hatV (
    matriz de n x k)
    """
```

Nota: no está permitodo el uso de la multiplicación matricial de numpy (`@`, `np.matmul`, etc)