Machine Learning y Data Analytics

HW 5 - Grupo 1

Ian Amighini
Julieta Brey
Lorenzo Nastri
Camila Sobrino
Matias Rodriguez Brun

Profesor Titular: Sergio Pernice

Consigna

El trabajo práctico está organizado en dos grandes bloques

1. <u>Matrices simétricas y descomposición espectral *A* de 6x6</u>

Se pide generar en Python una matriz con entradas aleatorias, verificar numéricamente que su traspuesta coincide con *A^T*, y luego calcular sus autovalores/autovectores con *LA.eig* (observando que, al no ser simétrica, pueden salir complejos). A continuación, se descompone *A* en su parte simétrica:

$$A = \frac{A + A^\top}{2} + \frac{A - A^\top}{2} = A_{\text{sim}} + A_{\text{asim}}$$

comprobando visualmente que *A* = *Asim* + *Aasim* y que cada componente cumple su propiedad de simetría o antisimetría.

<u>Propiedades de la parte simétrica y cambio de base</u>

Para *Asim* se repite el cálculo de autovalores/autovectores usando tanto *LA.eig* como *LA.eigh*, comprobando que en éste último caso todos los valores son reales y los vectores se entregan ya ortonormalizados y ordenados. Se verifica la ortonormalidad comprobando que *Q^T Q* = *I* y *Q^Tx* permite extraer las coordenadas de un vector cualquiera *x* en la base de autovectores, confirmando la reconstrucción *x* = *Q*(*Q^T x*).

2. <u>Formas cuadráticas en *R*^3</u>

La segunda sección parte de la forma

$$f(x_1, x_2, x_3) = 2x_1^2 - 3x_2^2 + x_3^2 + 3x_1x_2 - 5x_2x_3 + x_3x_1$$

y solicita determinar la matriz simétrica *C* ∊ *R*^3x3 que la representa

$$f(x_1, x_2, x_3) = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} C \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Además, pide describir cualitativamente el gráfico de una forma univariante

$f(x) = ax^2 = xax,$

según el signo de *a*, y analizar sin términos cruzados cómo varía la forma en cada variable independientemente .

<u>Visualización y ejemplo en GeoGebra, caso 2D</u>
Se abre en GeoGebra 3D el archivo ".ggb" para manipular la forma diagonal

$$f(x,y) = I1x^2 + I2y^2 = \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} I1 & 0 \\ 0 & I2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

explorando cómo el signo de *I*1,*I*2 afecta la forma de la superficie y comprobando que sus autovalores son *I*1,*I*2 con autovectores canónicos (ejes naturales) . Finalmente, se estudia el caso concreto

$$f(x,y) = x^2 + 2xy = \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

calculando sus autovalores $\lambda 1 \approx 1.62$, $\lambda 2 \approx -0.62$ y vectores propios *v*1,*v*2. Se demuestra algebraicamente que, al expresar (*x*,*y*) en la base {*v*1,*v*2}, todos los términos cruzados desaparecen y la forma queda diagonal con coeficientes λ*i*.

Introducción

1. **Sección I:** Analizar las propiedades de matrices genéricas y su descomposición en parte simétrica y antisimétrica; calcular y verificar autovalores y autovectores de matrices simétricas, comprobando su ortonormalidad.

2. **Sección II:** Asociar formas cuadráticas a matrices simétricas y demostrar que, al expresarlas en la base de autovectores, quedan diagonalizadas (sin términos cruzados), con coeficientes dados por los autovalores.

I. MATRICES SIMÉTRICAS, SUS AUTOVALORES Y SUS AUTOVECTORES

1. Se importan las librerías requeridas y se genera la matriz *A* aleatoria de 6x6: $A \in \mathbb{R}^{6\times 6}$

```
A = np.random.randn(6, 6)
print("Matriz A:\n", A)
```

```
Matriz A:
 [[-0.39948455  0.11123058 -0.81082462  1.07925541  1.25467087  0.96015531]
 [ 0.81727362 -1.41918706 -0.12847826 -0.48477103 -0.85240903  0.3152403 ]
 [-0.87157634 -0.22608444 -0.70368746 -1.49910919  0.39388587  0.53255891]
 [ 0.96955314  0.06793981  1.0403448  -1.19911795  0.30621493  0.25549192]
 [-0.31849019  2.0080578   0.68295779  1.66372431  0.44945847 -0.33267609]
 [ 0.50088717  0.73773367 -2.17530427 -1.87830654 -0.07180481 -0.07291853]]
```

2. Se genera y visualiza la matriz A traspuesta: $A^\top = \texttt{A.T}$

```
print("\nMatriz A.T (traspuesta de A):\n", A.T)
```

```
Matriz A.T (traspuesta de A):
 [[-0.39948455  0.81727362 -0.87157634  0.96955314 -0.31849019  0.50088717]
 [ 0.11123058 -1.41918706 -0.22608444  0.06793981  2.0080578   0.73773367]
 [-0.81082462 -0.12847826 -0.70368746  1.0403448   0.68295779 -2.17530427]
 [ 1.07925541 -0.48477103 -1.49910919 -1.19911795  1.66372431 -1.87830654]
 [ 1.25467087 -0.85240903  0.39388587  0.30621493  0.44945847 -0.07180481]
 [ 0.96015531  0.3152403   0.53255891  0.25549192 -0.33267609 -0.07291853]]
```

Explicación: Se puede verificar visualmente que A.T es la traspuesta de A, ya que los elementos (i,j) y (j,i) se corresponden.

3. Se generan los *autovectores y autovalores*:
Se calcula la descomposición espectral de A usando `eig`.

```python
# 3. Autovalores y autovectores de A (pueden ser complejos)
wA, vA = eig(A)
print("\nAutovalores de A:", wA)
print("Autovectores de A:\n", vA)
```

```
Autovalores de A: [ 1.34053437+0.j         -1.40063357+1.71324024j -1.40063357-1.71324024j
 -1.7706321 +0.j         -0.05678611+1.13672808j -0.05678611-1.13672808j]
Autovectores de A:
 [[ 0.70750431+0.j         -0.07987485-0.20508307j -0.07987485+0.20508307j
   0.4221651 +0.j         -0.18394046+0.08848413j -0.18394046-0.08848413j]
 [ 0.19700717+0.j         -0.22545359+0.25629338j -0.22545359-0.25629338j
  -0.07605987+0.j          0.33749801-0.04211237j  0.33749801+0.04211237j]
 [-0.31408836+0.j         -0.34941075-0.21695549j -0.34941075+0.21695549j
  -0.09268351+0.j          0.15873535-0.24113502j  0.15873535+0.24113502j]
 [ 0.22007613+0.j         -0.27050231+0.23492838j -0.27050231-0.23492838j
  -0.44060227+0.j         -0.08708371-0.11321402j -0.08708371+0.11321402j]
 [ 0.16103469+0.j          0.55894443+0.j          0.55894443-0.j
   0.38594822+0.j         -0.49758247-0.19719442j -0.49758247+0.19719442j]
 [ 0.53629158+0.j         -0.24606489-0.40565764j -0.24606489+0.40565764j
  -0.68140656+0.j          0.67263672+0.j          0.67263672-0.j        ]]
```

4. Descomposición en parte simétrica y antisimétrica

Por definición:

$$A = \underbrace{\frac{A + A^\top}{2}}_{A_{\text{sim}}} + \underbrace{\frac{A - A^\top}{2}}_{A_{\text{antisim}}} \qquad A_{\text{sim}} + A_{\text{antisim}} = A.$$

```python
# 4. Descomposición en parte simétrica y antisímetrica
A_sim = (A + A.T) / 2
A_antisim = (A - A.T) / 2
print("\nParte simétrica A_sim:\n", A_sim)
print("\nParte antisimétrica A_antisim:\n", A_antisim)
print("\nVerificación A_sim + A_antisim = A:\n", A_sim + A_antisim)
```

```
Parte simétrica A_sim:
 [[-0.39948455  0.4642521  -0.84120048  1.02440428  0.46809034  0.73052124]
 [ 0.4642521  -1.41918706 -0.17728135 -0.20841561  0.57782438  0.52648699]
 [-0.84120048 -0.17728135 -0.70368746 -0.22938219  0.53842183 -0.82137268]
 [ 1.02440428 -0.20841561 -0.22938219 -1.19911795  0.98496962 -0.81140731]
 [ 0.46809034  0.57782438  0.53842183  0.98496962  0.44945847 -0.20224045]
 [ 0.73052124  0.52648699 -0.82137268 -0.81140731 -0.20224045 -0.07291853]]

Parte antisimétrica A_antisim:
 [[ 0.         -0.35302152  0.03037586  0.05485114  0.78658053  0.22963407]
 [ 0.35302152  0.          0.04880309 -0.27635542 -1.43023342 -0.21124668]
 [-0.03037586 -0.04880309  0.         -1.269727   -0.14453596  1.35393159]
 [-0.05485114  0.27635542  1.269727    0.         -0.67875469  1.06689923]
 [-0.78658053  1.43023342  0.14453596  0.67875469  0.         -0.13043564]
 [-0.22963407  0.21124668 -1.35393159 -1.06689923  0.13043564  0.        ]]

Verificación A_sim + A_antisim = A:
 [[-0.39948455  0.11123058 -0.81082462  1.07925541  1.25467087  0.96015531]
 [ 0.81727362 -1.41918706 -0.12847826 -0.48477103 -0.85240903  0.3152403 ]
 [-0.87157634 -0.22608444 -0.70368746 -1.49910919  0.39388587  0.53255891]
 [ 0.96955314  0.06793981  1.0403448  -1.19911795  0.30621493  0.25549192]
 [-0.31849019  2.0080578   0.68295779  1.66372431  0.44945847 -0.33267609]
 [ 0.50088717  0.73773367 -2.17530427 -1.87830654 -0.07180481 -0.07291853]]
```

- Se verifica visualmente que A_sim es simétrica y A_asim es antisimétrica.
- Se verifica que A = A_sim + A_antisim
- Esta descomposición es útil porque cualquier matriz cuadrada puede separarse en una parte simétrica y otra antisimétrica.

5. Se calculan los autovalores y autovectores de la matriz simétrica A_sim utilizando la función **eig**, que puede retornar valores complejos:

```
# 5. Autovalores y autovectores de A_sim usando eig
# (Cuando uno trabaja con matrices simetricas conviene usar la función “LA.eigh” en vez de “LA.eig”)
wA_sim, vA_sim = eig(A_sim)
print("\nAutovalores de A_sim (eig):", wA_sim)
print("Autovectores de A_sim (eig):\n", vA_sim)
```

```
Autovalores de A_sim (eig): [ 1.49211235  1.36525458 -0.24211643 -2.57368904 -1.65573482 -1.73076373]
Autovectores de A_sim (eig):
 [[-0.5527006  -0.31650594 -0.3285842   0.41688666  0.5462155  -0.1193005 ]
 [-0.21902037 -0.14081     0.33542927 -0.29269933 -0.06462638 -0.85430695]
 [ 0.15164031  0.48585869  0.36257059 -0.14492911  0.7669802   0.01503465]
 [-0.41553858  0.20599601 -0.45691936 -0.74438277  0.02413272  0.14638968]
 [-0.66237226  0.40292983  0.45230435  0.23765399 -0.29754709  0.22207571]
 [-0.11021802 -0.66268474  0.48446638 -0.33006522  0.14176308  0.43006213]]
```

6. Se repite el cálculo de autovalores y autovectores, pero esta vez usando la función **eigh**, que está diseñada específicamente para matrices simétricas y garantiza resultados reales y más estables.

```
# 6. Autovalores y autovectores de A_sim usando eigh (simétricas)
wA_sim2, vA_sim2 = eigh(A_sim)
print("\nAutovalores de A_sim (eigh):", wA_sim2)
print("Autovectores de A_sim (eigh):\n", vA_sim2)
```

```
Autovalores de A_sim (eigh): [-2.57368904 -1.73076373 -1.65573482 -0.24211643  1.36525458  1.49211235]
Autovectores de A_sim (eigh):
 [[ 0.41688666 -0.1193005   0.5462155  -0.3285842  -0.31650594  0.5527006 ]
 [-0.29269933 -0.85430695 -0.06462638  0.33542927 -0.14081     0.21902037]
 [-0.14492911  0.01503465  0.7669802   0.36257059  0.48585869 -0.15164031]
 [-0.74438277  0.14638968  0.02413272 -0.45691936  0.20599601  0.41553858]
 [ 0.23765399  0.22207571 -0.29754709  0.45230435  0.40292983  0.66237226]
 [-0.33006522  0.43006213  0.14176308  0.48446638 -0.66268474  0.11021802]]
```

La ventaja es que los resultados obtenidos son reales y los autovectores están garantizados a ser ortogonales.

7. Se verifica que los autovectores obtenidos con **eigh** forman una base ortonormal:

Para esto, se calcula el producto interno de la matriz de autovectores con su transpuesta.

El resultado debe aproximarse a la **matriz identidad**, lo que indica ortonormalidad.

```
# 7. Verificar ortonormalidad de vA_sim2 (producto interno ~ identidad)
ortho_check = np.dot(vA_sim2.T, vA_sim2)
print("\nProducto interno de autovectores (debe ser identidad):\n", ortho_check)
```

```
Producto interno de autovectores (debe ser identidad):
 [[ 1.00000000e+00  3.78659040e-17  1.77980280e-16 -2.37379754e-16
  -1.07148014e-16 -1.20049589e-16]
 [ 3.78659040e-17  1.00000000e+00 -9.35716420e-17 -1.15585203e-16
  -2.73174100e-16 -1.49183102e-16]
 [ 1.77980280e-16 -9.35716420e-17  1.00000000e+00 -6.27903957e-17
   1.16968088e-16 -2.67078675e-16]
 [-2.37379754e-16 -1.15585203e-16 -6.27903957e-17  1.00000000e+00
   1.34823719e-17  4.17790106e-17]
 [-1.07148014e-16 -2.73174100e-16  1.16968088e-16  1.34823719e-17
   1.00000000e+00  1.83219111e-16]
 [-1.20049589e-16 -1.49183102e-16 -2.67078675e-16  4.17790106e-17
   1.83219111e-16  1.00000000e+00]]
```

8. Coordenadas de un vector en la base de autovectores:

Se genera un vector aleatorio x y se lo proyecta en la base de autovectores de A_sim, obteniendo sus coordenadas en dicha base. Luego se reconstruye x a partir de esos coeficientes (como combinación lineal de autovectores) para verificar que la transformación es correcta.

```
# 8. Coordenadas de un vector x en la base de autovectores
x = np.random.randn(6)
a = vA_sim2.T @ x  # a_n = v_n^T x
x_recon = vA_sim2 @ a
print("\nVector x original:\n", x)
print("\nCoeficientes a_n en la base de autovectores:\n", a)
print("\nReconstrucción x_recon:\n", x_recon)
print("\nError de reconstrucción ||x - x_recon||:", np.linalg.norm(x - x_recon))
```

```
Vector x original:
 [ 0.55490283  0.77583272 -0.52442054  0.90028744 -1.27629656 -1.87224334]

Coeficientes a_n en la base de autovectores:
 [-0.27526334 -1.69370658 -0.01319318 -2.00790696  0.37223543 -0.12149263]

Reconstrucción x_recon:
 [ 0.55490283  0.77583272 -0.52442054  0.90028744 -1.27629656 -1.87224334]

Error de reconstrucción ||x - x_recon||: 1.687392909129966e-15
```

El vector reconstruido (x_recon) coincide con el original x, y el error de reconstrucción es prácticamente cero, lo cual demuestra que los autovectores forman una base válida.

II. FORMAS CUADRÁTICAS

1. Definir la matriz simétrica C para una forma cuadrática

Dada la forma cuadrática:

$$f(x_1, x_2, x_3) = 2x_1^2 - 3x_2^2 + x_3^2 + 3x_1x_2 - 5x_2x_3 + x_3x_1,$$

Se construye la **matriz simétrica asociada** C a partir de los coeficientes.

$$f(\mathbf{x}) = \begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} C \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix},$$

Se obtiene la matriz:

$$C = \begin{pmatrix} 2 & \frac{3}{2} & \frac{1}{2} \\ \frac{3}{2} & -3 & -\frac{5}{2} \\ \frac{1}{2} & -\frac{5}{2} & 1 \end{pmatrix}.$$

```
] # 1. Definir la matriz simétrica C para f(x1,x2,x3)
  #    f = 2x1^2 - 3x2^2 + x3^2 + 3x1x2 -5x2x3 + x3x1
  C = np.array([
      [2,    3/2,  1/2],
      [3/2, -3,   -5/2],
      [1/2, -5/2,  1]
  ])
  print("\nMatriz C asociada a la forma cuadrática:\n", C)
```

```
Matriz C asociada a la forma cuadrática:
 [[ 2.   1.5  0.5]
 [ 1.5 -3.  -2.5]
 [ 0.5 -2.5  1. ]]
```

2. Se verifica que $f(x) = xTCx \quad f(x) = x\hat{}T\ C\ x \quad f(x) = xTCx$

Se genera un vector aleatorio x y se calcula $f(x)$ de dos maneras:

- Evaluando la fórmula directamente.
- Usando la forma matricial xTCx

Se verifica que ambas expresiones coinciden, confirmando que la forma cuadrática está correctamente representada por la matriz C.

```python
# 2. Verificar f(x) = x^T C x para un vector x aleatorio
x3 = np.random.randn(3)
f_direct = 2*x3[0]**2 - 3*x3[1]**2 + x3[2]**2 + 3*x3[0]*x3[1] - 5*x3[1]*x3[2] + x3[2]*x3[0]
f_quad = x3.T @ C @ x3
print("\nf(x) directo:", f_direct)
print("f(x) mediante x^T C x:", f_quad)
```

```
f(x) directo: -2.81831182491712
f(x) mediante x^T C x: -2.8183118249171204
```

3. Autodescomposición de la matriz C

Se obtienen los autovalores y autovectores de la matriz C usando la función eig.

```python
# 3. Autodescomposición de C
wC, vC = eig(C)
print("\nAutovalores de C:", wC)
print("Autovectores de C:\n", vC)
```

```
Autovalores de C: [-4.57603367  2.5         2.07603367]
Autovectores de C:
 [[ 0.23220629 -0.81649658  0.52859585]
 [-0.87960135 -0.40824829 -0.24420238]
 [-0.41518876  0.40824829  0.81298931]]
```

4. Caso bidimensional:

Se define una nueva matriz simétrica C_2 correspondiente a una forma cuadrática en dos variables y se obtienen sus autovalores y autovectores.

```python
# 4. Ejemplo de forma cuadrática 2D: f(x,y) = x^2 + 2xy
C2 = np.array([[1, 1],
               [1, 0]])
wC2, vC2 = eig(C2)
print("\nAutovalores de C2 (2D):", wC2)
print("Autovectores de C2 (2D):\n", vC2)
```

```
Autovalores de C2 (2D): [ 1.61803399 -0.61803399]
Autovectores de C2 (2D):
 [[ 0.85065081 -0.52573111]
 [ 0.52573111  0.85065081]]
```

Demostración algebraica:

Para: $f(x, y) = x^2 + 2xy,$

La matriz es $$C_2 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

Su autodescomposición da:

$$\lambda_1 = 1.6180,\ \lambda_2 = -0.6180, \quad v_1 = \begin{pmatrix} 0.85065 \\ 0.52573 \end{pmatrix},\ v_2 = \begin{pmatrix} -0.52573 \\ 0.85065 \end{pmatrix}.$$

Al escribir $\binom{x}{y} = a_1 v_1 + a_2 v_2$ y usar $C_2 v_i = \lambda_i v_i$, se demuestra paso a paso:

$$f = \begin{pmatrix} x & y \end{pmatrix} C_2 \begin{pmatrix} x \\ y \end{pmatrix} = 1.62\, a_1^2 - 0.62\, a_2^2,$$

5. Se verifica la forma diagonal en la base de autovectores:

Se evalúa la forma cuadrática en la base canónica y en la base de autovectores, y se verifica que ambas formas coinciden.

```
# 5. Verificar la forma diagonal en la base de autovectores 2D
x2 = np.random.randn(2)
a2 = vC2.T @ x2
f2_direct = x2.T @ C2 @ x2
f2_diag = wC2[0]*a2[0]**2 + wC2[1]*a2[1]**2
print("\nf2(x) directo:", f2_direct)
print("f2(x) en base diagonal:", f2_diag)
```

```
f2(x) directo: 0.3910515011743627
f2(x) en base diagonal: 0.39105150117436266
```

1_HW5.ipynb

Conclusiones

- **Sección I:** Toda matriz real se descompone en parte simétrica y antisimétrica; los autovalores de la parte simétrica son reales y los autovectores forman una base ortonormal, lo que permite proyectar y reconstruir vectores con alta precisión.
- **Sección II**: Cada forma cuadrática se asocia a una matriz simétrica; en la base de sus autovectores la forma queda diagonal y los coeficientes son los autovalores, facilitando su interpretación geométrica (parábolas positivas, negativas o degeneradas).