Machine Learning y Data Analytics

HW 2 - Grupo 1

Ian Amighini
Julieta Brey
Lorenzo Nastri
Camila Sobrino
Matias Rodriguez Brun

Profesor Titular: Sergio Pernice

Consigna

Pídanle a su IA favorita que genere un programa Python que genere dos vectores 3D: x1 y x2 al azar (vectores cuyas coordenadas son números elegidos entre -1 y 1 al azar con una distribución constante).

Pídanle que genere otro programa Python que grafique en un gráfico 3D a los dos vectores junto con el plano correspondiente al span de los primeros dos vectores x1 y x2.

A partir de x1 y x2 generen los vectores v1 y v2, vectores ortogonales entre sí que espanean el mismo subespacio, y e1, e2, vectores ortonormales que espanean el mismo subespacio, tienen norma 1 y son ortogonales entre sí.

Verifiquen gráficamente que el span de e1 y e2 es el mismo que el de x1 y x2 (que el gráfico se pueda girar con el mouse.)

Verifiquen numéricamente que e1 y e2 tienen norma 1 y son ortogonales entre si.

Identifiquen explícitamente en el programa Python generado las operaciones que hemos visto en clase para llegar a e1 y e2 a partir de x1 y x2. Explique en español la lógica de sus programas.

1) Generación de vectores

En este trabajo, se generaron dos vectores tridimensionales (v1, v2) con coordenadas aleatorias en el intervalo [−1,1][-1,1][−1,1]. Estas coordenadas deben ser elegidas al azar con una distribución constante, lo que significa que cada número dentro del intervalo debe tener la misma probabilidad de ser seleccionado.

- Distribución Uniforme: cualquier número dentro del rango dado tiene la misma probabilidad de ser seleccionado, sin sesgos hacia valores específicos.

Para asegurar que los valores de los vectores cumplan con este criterio, se utilizó la función np.random.uniform(-1, 1, 3) de la biblioteca NumPy. Esta función genera tres números aleatorios en el intervalo [−1,1][-1,1][−1,1] siguiendo una distribución uniforme.

```
[4]  import numpy as np
     import plotly.graph_objects as go
```

```
     x1 = np.random.uniform(-1, 1, 3)
     x2 = np.random.uniform(-1, 1, 3)

     print("Vector x1:", x1)
     print("Vector x2:", x2)
```

```
Vector x1: [-0.99152466  0.19336791 -0.94309087]
Vector x2: [ 0.9308292   0.72322448 -0.52146278]
```

2) **Aplicación del proceso de Gram-Schmidt (base ortogonal) + Normalización (base ortonormal)**

Para transformar los vectores iniciales x1 y x2 en una base ortogonal, se aplicó el proceso de Gram-Schmidt, un algoritmo que permite obtener un conjunto de vectores ortogonales a partir de un conjunto linealmente independiente que espanea un subespacio vectorial.

En términos matemáticos, el proceso consiste en tomar un primer vector base v1=x1 y luego construir un segundo vector ortogonal a v1 mediante la fórmula:

$$\mathbf{v}_2 = \mathbf{x}_2 - \text{proj}_{\mathbf{v}_1}(\mathbf{x}_2) = \mathbf{x}_2 - \frac{\mathbf{x}_2 \cdot \mathbf{v}_1}{\mathbf{v}_1 \cdot \mathbf{v}_1}\mathbf{v}_1$$

Donde projv1(x2) representa la proyección de x2 sobre v1, y el símbolo · indica el **producto escalar**, operación para cuantificar el concepto de ortogonalidad.

Una vez obtenida la base ortogonal {v1,v2}, se procedió a construir una **base ortonormal** {e1,e2}, es decir, un conjunto de vectores de norma 1 y mutuamente ortogonales. Esto se logró mediante la **normalización** de los vectores ortogonales:

$$\mathbf{e}_1 = \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|}, \quad \mathbf{e}_2 = \frac{\mathbf{v}_2}{\|\mathbf{v}_2\|} \qquad \textbf{Donde:} \quad \|\mathbf{v}\| = \sqrt{\mathbf{v} \cdot \mathbf{v}}$$

Este procedimiento garantiza que {e1,e2} mantengan el mismo span que los vectores originales {x1,x2}, es decir, espanean el mismo subespacio vectorial del espacio tridimensional pero con propiedades geométricas más convenientes como ortogonalidad y unidad de norma.

```
# Paso 1: v1 = x1
v1 = x1

# Paso 2: v2 = x2 proyectado ortogonalmente a v1
proj = np.dot(x2, v1) / np.dot(v1, v1) * v1
v2 = x2 - proj

# Paso 3: Normalización para obtener base ortonormal
e1 = v1 / np.linalg.norm(v1)
e2 = v2 / np.linalg.norm(v2)

# ===== Verificaciones numéricas =====
print("\nVerificación de normas y ortogonalidad:")
print(f"Norma de e1: {np.linalg.norm(e1)}")
print(f"Norma de e2: {np.linalg.norm(e2)}")
print(f"Producto escalar e1 · e2 (debe ser ≈ 0): {np.dot(e1, e2)}")
```

Para comprobar que efectivamente se obtuvo una base ortonormal, se verificó:

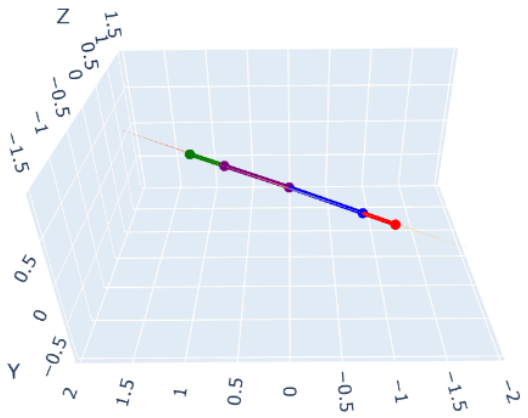
- Que la norma de ambos vectores // e1 // y // e2 // sea igual a 1, confirmando que son unitarios.

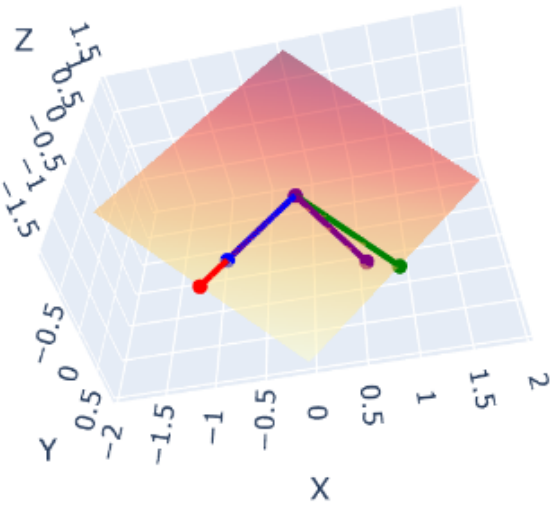
- Que el producto escalar e1 · e2 sea cercano a 0, validando que son ortogonales entre sí.

```
Verificación de normas y ortogonalidad:
Norma de e1: 1.0
Norma de e2: 0.9999999999999999
Producto escalar e1 · e2 (debe ser ≈ 0): -5.551115123125783e-17
```

3) Verificación con Imagen y Plano Interactivo.

Vectores x1, x2 y el plano de su span, junto con e1 y e2 ortonormales





Plataforma de Python con la imagen
https://colab.research.google.com/drive/1M8ugVJYG9Vs7zgO62UBuQh1XiNBoHzlC#scrollTo=STHwYaUDNEwI