Machine Learning y Data Analytics

HW 4 - Grupo 1

Ian Amighini
Julieta Brey
Lorenzo Nastri
Camila Sobrino
Matias Rodriguez Brun

Profesor Titular: Sergio Pernice

Consigna

Explorar con Python la resolución del sistema lineal $Ax=y$ en tres escenarios distintos, y explicar cada uno en términos de combinaciones lineales de las columnas de A y del espacio columna. En concreto:

1. Caso Sobredeterminado (m>n, aquí 3×2)

    - Generar una matriz aleatoria $A \in R^{3\times 2}$ y un vector $y \in R^3$.
    - “Agrandar” A a 3×3 añadiendo una tercera columna de ceros y usar LA.solve(A,y) para observar el error de “Singular matrix” (no hay solución).
    - Explicar por qué, puesto que y fue generado al azar, en general no pertenece al espacio columna de A (combinación de sólo dos vectores en R^3) .

2. Caso Subdeterminado (m<n, aquí 2×3)

    - Generar $A \in R^{2\times 3}$ y $y \in R^2$.
    - Mostrar que hay infinitas soluciones porque son tres vectores en R^2.
    - Cortar la tercera columna para formar $B \in R^{2\times 2}$, resolver $B\,(x_0,x_1)^T = y$ y construir la solución extendida $(x_0,x_1,0)^T$.
    - Hallar $\lambda=(\lambda_0,\lambda_1)$ tal que la tercera columna de A es combinación de las otras dos (resolviendo $B\,\lambda=a_2$).
    - Demostrar que para cualquier α la familia de vectores:

$$x+v=\begin{pmatrix} x_0 \\ x_1 \\ 0 \end{pmatrix}+\begin{pmatrix} \alpha\lambda_0 \\ \alpha\lambda_1 \\ -\alpha \end{pmatrix}$$

      satisface $A(x+v)=y$, generando infinitas soluciones .

3. Caso Determinado (m=n)

    - Aunque no está detallado paso a paso, recordar que si $A \in R^{n\times n}$ es invertible (columnas independientes), LA.solve(A,y) da la única solución.

Explicar por qué estamos seguros de que λ_0 y λ_1 van a existir.
Explicar por qué, para encontrar λ_0 y λ_1 tenemos que resolver el siguiente sistema:

$$\begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix}\begin{pmatrix} \lambda_0 \\ \lambda_1 \end{pmatrix}=\begin{pmatrix} a_{02} \\ a_{12} \end{pmatrix}$$

Objetivo

Código → 1_HW4.ipynb

Este trabajo busca comprender la ecuación lineal $Ax=y$, donde:

- $A \in Rm \times n$ es una matriz conocida,
- $y \in Rm \times 1$ es un vector conocido,
- $x \in Rn \times 1$ es el vector incógnita.

Dependiendo de las dimensiones de la matriz A, el sistema puede tener:

- Ninguna solución si $m>n$
- Infinitas soluciones si $m<n$,
- Una única solución si $m=n$ y las columnas de A son linealmente independientes.

Caso 1: m>n

Configuración

- $m=3$, $n=2$
- Generamos $A \in R3 \times 2$ y $y \in R3 \times 1$
- Para poder usar `LA.solve`, convertimos ficticiamente A a $R3 \times 3$ agregando una columna de ceros.

```python
import numpy as np
from numpy import linalg as LA

np.random.seed(0)
m, n = 3, 2
A = 2 * (np.random.rand(m, n) - 0.5)
A = np.hstack((A, np.zeros((m, 1))))
y = 2 * (np.random.rand(m, 1) - 0.5)

x = LA.solve(A, y)  # Esto lanza LinAlgError: Singular matrix
```

```
---------------------------------------------------------------------------
LinAlgError                               Traceback (most recent call last)
```

La matriz A tiene más filas que columnas, lo cual indica que hay más ecuaciones que incógnitas. Esto, en general, conduce a un **sistema sobredeterminado**, donde es **muy poco probable que y** se pueda expresar como una **combinación lineal de solo dos vectores columna** (los de A), ya que esos vectores viven en un subespacio de dimensión 2 dentro de R3.

Resultado: El sistema no tiene solución. Python lanza el error `LinAlgError: Singular matrix` al intentar resolverlo porque la matriz no es invertible.

Conclusión:
El sistema no tiene solución porque el vector y se encuentra en R^3 (espacio tridimensional), mientras que las columnas de A solo pueden generar un plano (espacio bidimensional).
Ax es una combinación lineal de las columnas de A, por lo que solo puede alcanzar vectores dentro de ese plano.
Como y fue generado aleatoriamente en R^3, la probabilidad de que caiga exactamente en ese plano es prácticamente cero.

Caso 2: m<n

Configuración

- m=2, n=3
- Generamos $A \in R^{2\times3}$ y $y \in R^{2\times1}$

```
m, n = 2, 3
A = 2 * (np.random.rand(m, n) - 0.5)
y = 2 * (np.random.rand(m, 1) - 0.5)
```

En este caso hay más incógnitas que ecuaciones. Los tres vectores columna de A viven en R2, por lo que necesariamente son **linealmente dependientes**. Por lo tanto, la ecuación Ax=y va a tener **infinitas soluciones**, ya que el conjunto solución será un subespacio afín de dimensión mayor que cero.

Encontrando una solución

Recortamos A para quedarnos con solo las dos primeras columnas, llamándola B:

```
B = A[:, :2]
x_particular = LA.solve(B, y)
x_sol = np.vstack((x_particular, [[0]]))  # (x0, x1, 0)
```

```
np.allclose(A @ x_sol, y)  # True
```

```
True
```

Encontrando las infinitas soluciones

1. Relación de dependencia lineal

Buscamos λ_0, λ_1 tales que:

$$a_2 = \lambda_0 a_0 + \lambda_1 a_1$$

$$\begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix} \begin{bmatrix} \lambda_0 \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} a_{02} \\ a_{12} \end{bmatrix}$$

```
a2 = A[:, 2]
lambdas = LA.solve(B, a2)
```

2. Vector generador del núcleo de A

Sea α∈R un escalar aleatorio:

$$v = \begin{bmatrix} \alpha\lambda_0 \\ \alpha\lambda_1 \\ -\alpha \end{bmatrix} \Rightarrow Av = 0$$

```
[8] alpha = np.random.randn()
    v = np.array([[alpha * lambdas[0]], [alpha * lambdas[1]], [-alpha]])
    np.allclose(A @ v, 0)  # True
```

3. Conjunto de soluciones

$$x_{\text{gen}} = x + v = \begin{bmatrix} x_0 + \alpha\lambda_0 \\ x_1 + \alpha\lambda_1 \\ -\alpha \end{bmatrix} \Rightarrow Ax_{\text{gen}} = y$$

```
x_gen = x_sol + v
np.allclose(A @ x_gen, y)  # True
```

```
True
```

Conclusión:
El sistema Ax = y con A (2×3) y vector y (2×1) tiene infinitas soluciones porque:

- Al tener solo 2 ecuaciones pero 3 incógnitas, el sistema está subdeterminado.
- Matemáticamente, esto se refleja en que las 3 columnas de A viven en R^2 (un espacio bidimensional), por lo que necesariamente existe una dependencia lineal entre ellas.
- Esta dependencia lineal se puede expresar como $a_2 = \lambda_0 a_0 + \lambda_1 a_1$, donde a_0, a_1 y a_2 son las columnas de A.
- La solución general del sistema toma la forma: $x = [x_0+\alpha\lambda_0, x_1+\alpha\lambda_1, -\alpha]^T$
- Como α puede ser cualquier número real, hay infinitas combinaciones de valores que satisfacen la ecuación Ax = y.

Caso 3: m=n

Generamos la matriz A3 y el vector y3:

```
A3 = 2 * (np.random.rand(3, 3) - 0.5)
y3 = 2 * (np.random.rand(3, 1) - 0.5)
```

Intentamos resolver el sistema usando **solve**, que requiere que A3 sea cuadrada e invertible

```
try:
    x3 = solve(A3, y3)
    print("Solución única encontrada:", x3)
except LinAlgError:
    print("No hay solución única: matriz singular")
```

Dado que A3 es una matriz cuadrada 3x3 con vectores columna generados aleatoriamente, es muy probable que estos vectores sean linealmente independientes. En ese caso, existe una única solución x3 tal que A3 * x3 = y3.

Verificamos la solución:

```
np.allclose(A3 @ x3, y3)
```

Cuando m = n y los vectores columna de A son linealmente independientes, el sistema tiene una única solución. La función solve permite encontrar dicha solución de forma eficiente cuando la matriz es cuadrada y no singular.

Conclusión final

Este trabajo muestra cómo el número de ecuaciones respecto al número de incógnitas afecta la naturaleza de las soluciones del sistema lineal:

- Si $m>n$: sistema sobredeterminado → en general **no hay solución**.
- Si $m=n$: sistema cuadrado → puede tener **una única solución** si A es invertible.
- Si $m<n$: sistema subdeterminado → en general **hay infinitas soluciones**.

Se utilizó un enfoque numérico con numpy.linalg.solve y análisis geométrico del rango y núcleo de matrices para explicar el comportamiento de las soluciones en distintos casos.