

**Nombre:** Luis de la Garza González

**Matrícula:** al03101869

**Nombre del curso:** Machine Learning.

**Trabajo:** Actividad 01.- Las matemáticas para la resolución de problemas

**Nombre del profesor:** Mtro.

**Fecha:** 14.09.2025

## Contenido

<b>Instrucciones</b> .....	3
<b>Objetivo</b> .....	4
<b>Desarrollo</b> .....	4
Problema ejemplo .....	4
Habilidades por alcanzar .....	5
Ejercicio 1 .....	5
Ejercicio 2 .....	7
Ejercicio 3 .....	8
<b>Conclusiones</b> .....	11
<b>Liga al código</b> .....	12

## Instrucciones

**1. Identifica una problemática que pueda solucionarse mediante algún procedimiento matemático y responde a las siguientes preguntas:**

- ¿Qué tipo de pensamiento matemático está involucrado en la solución del problema?
- ¿Qué procesos matemáticos se deben aplicar para encontrar dicha solución?
- ¿Cuáles son los pasos que se deben realizar para solucionar dicho problema?

Elabora un reporte donde argumentes tus respuestas anteriores.

**2. Elabora una hoja de ruta o plan de estudios donde representes:**

- Las habilidades y conocimientos matemáticos que ya posees.
- Los que consideras que debes alcanzar para dominar las bases necesarias que te permitirán comprender mejor los fundamentos del aprendizaje automático.

### Actividades de Álgebra Lineal en Python (Jupyter Notebook):

**3. Utiliza un cuaderno de Jupyter Notebook y el lenguaje de programación Python para realizar las siguientes operaciones:**

- Genera una matriz cuadrada de 4x4.
- Calcula su inversa y su traspuesta.
- Multiplica la matriz por un escalar.
- Realiza el producto interno con una nueva matriz de dimensiones 4x2.

**4. Investiga otras formas computacionales de solucionar sistemas de ecuaciones lineales Elabora una tabla resumen donde presentes tus conclusiones.**

**5. Genera un nuevo cuaderno de Jupyter Notebook y resuelve los siguientes sistemas de ecuaciones lineales, encontrando los valores de las variables utilizando diferentes métodos:**

a:

$$\begin{array}{rcl} x + 2y + 5z - 4w & = & 21 \\ 5x + 8y + z + w & = & -8 \\ 5x + 7y - 3z + 2w & = & 14 \\ -x + 3y + 9z - w & = & 5 \end{array}$$

b:

$$\begin{array}{rcl} 6x + 8z & = & 9 \\ 2x + 2y + z & = & 15 \\ 4x + y & = & -4 \end{array}$$

## Objetivo

Identificar un problema de optimización que pueda solucionarse mediante algún procedimiento matemático de álgebra lineal, describir mis habilidades y conocimientos matemáticos actuales y los que debo alcanzar para comprender mejor los fundamentos del aprendizaje automático. Practicar la resolución de sistemas de ecuaciones lineales en una notebook de Jupyter utilizando Python.

## Desarrollo

### 1.- Problema ejemplo

Una fábrica produce camisetas de diferentes tallas y colores. Cada tipo de camiseta requiere distintos insumos (tela, tinta, tiempo de máquina) y genera diferentes márgenes de ganancia. La empresa desea maximizar sus ganancias sin exceder los recursos disponibles.

Las variables involucradas son:

- número de camisetas tipo A
- número de camisetas tipo B
- número de camisetas tipo C

Las restricciones son:

- Disponibilidad de tela (en metros cuadrados)
- Horas de máquina disponibles por día
- Cantidad de tinta por color
- Capacidad máxima de producción diaria

Para este tipo de problemas de optimización, utilizaríamos el método simplex por lo que estaríamos haciendo uso principalmente del **álgebra lineal** y la **Investigación de operaciones**.

## 2.- Habilidades por alcanzar

Aunque tengo el antecedente de mis estudios en la facultad de ingeniería, tengo que volver a desarrollar la habilidad en las siguientes áreas de las matemáticas.

Área	Objetivo
Álgebra lineal	Comprender operaciones con matrices, vectores, y espacios vectoriales
Cálculo	Entender derivadas parciales, gradientes y optimización
Probabilidad	Manejar distribuciones, inferencia y funciones de densidad
Estadística	Aplicar métricas de evaluación y análisis de errores
Lógica matemática	Formalizar problemas y estructuras de decisión

## 3.- Ejercicio 1

Utiliza un cuaderno de Jupyter Notebook y el lenguaje de programación Python para realizar las siguientes operaciones:

- Genera una matriz cuadrada de 4x4.
- Calcula su inversa y su traspuesta.
- Multiplica la matriz por un escalar.

```
In [1]: # Lo primero es importar la librería numpy que es una librería que nos permite el manejo de arreglos y matrices
import numpy as np

In [2]: # Ahora generamos una matriz de 4x4 elementos
M = np.array([[1, 9, 8, -4],
              [2, 3, 1, 2],
              [8, 1, -4, 6],
              [-3, 2, -3, 1]])
print(M)

[[ 1  9  8 -4]
 [ 2  3  1  2]
 [ 8  1 -4  6]
 [-3  2 -3  1]]
```

```
In [5]: # Calculamos su matriz inversa:
try:
    M_inv = np.linalg.inv(M)
    print("Matriz inversa de M:")
    print(M_inv)
except np.linalg.LinAlgError:
    print("La matriz M no es invertible.")

# Para comprobar que efectivamente es la matriz inversa, multiplicamos M x M_inv y debe dar como resultado una matriz unitaria
producto = np.dot(M, M_inv)
# redondeamos cada elemento
producto_redondeado = np.round(producto, 2)
print("M x M_inv redondeado a 1 decimal:")
print(producto_redondeado)
```

```
Matriz inversa de M:
[[ 0.08456376 -0.26040268  0.15436242 -0.06711409]
 [ 0.09395973 -0.06711409  0.06040268  0.14765101]
 [-0.08322148  0.43087248 -0.16778523 -0.18791946]
 [-0.18389262  0.64563758 -0.16107383 -0.06040268]]
M x M_inv redondeado a 1 decimal:
[[ 1.  0.  0.  0.]
 [-0.  1. -0. -0.]
 [ 0.  0.  1.  0.]
 [-0.  0.  0.  1.]]
```

```
In [8]: # Ahora calculamos la matriz traspuesta de M
M_T = M.T
print("Matriz M:")
print(M, "\n")
print("Matriz traspuesta de M:")
print(M_T)
```

```
Matriz M:
[[ 1  9  8 -4]
 [ 2  3  1  2]
 [ 8  1 -4  6]
 [-3  2 -3  1]]
```

```
Matriz traspuesta de M:
[[ 1  2  8 -3]
 [ 9  3  1  2]
 [ 8  1 -4 -3]
 [-4  2  6  1]]
```

```
In [9]: # Multiplicamos la matriz M por un escalar.
# Definimos el escalar
escalar = 3

# Multiplicamos la matriz M por el escalar
M_escalada = escalar * M

# Mostramos el resultado
print(f"Matriz M multiplicada por el escalar {escalar}:\n")
print(M_escalada)
```

Matriz M multiplicada por el escalar 3:

```
[[ 3 27 24 -12]
 [ 6  9  3  6]
 [24  3 -12 18]
 [-9  6 -9  3]]
```

#### 4.- Ejercicio 2

Investiga otras formas computacionales de solucionar sistemas de ecuaciones lineales. Elabora una tabla resumen donde presentes tus conclusiones.

**Tabla resumen: Métodos computacionales para sistemas de ecuaciones lineales**

Método	Tipo	Características principales	Ventajas	Desventajas
Eliminación Gaussiana	Directo	Reduce la matriz a forma escalonada para resolver por sustitución regresiva	Preciso, aplicable a cualquier sistema consistente	Sensible a errores numéricos en matrices mal condicionadas
Eliminación Gauss-Jordan	Directo	Lleva la matriz a forma escalonada reducida (identidad) para obtener solución directa	Da solución única directamente	Más costoso computacionalmente que Gauss
Regla de Cramer	Directo	Usa determinantes para resolver sistemas con solución única	Simple para sistemas pequeños ( $n \leq 3$ )	Ineficiente para sistemas grandes; requiere determinantes
Descomposición LU	Directo	Factoriza la matriz en producto de matrices triangular inferior y superior	Eficiente para resolver múltiples sistemas con misma matriz	Requiere que la matriz sea factorizable
Método de Jacobi	Iterativo	Aproxima soluciones sucesivas usando valores anteriores	Fácil de implementar, útil en sistemas dispersos	Convergencia lenta; requiere condiciones como diagonal dominante
Método de Gauss-Seidel	Iterativo	Similar a Jacobi pero usa valores actualizados en cada paso	Más rápido que Jacobi; útil en grandes sistemas	No siempre converge; depende de propiedades de la matriz

Método	Tipo	Características principales	Ventajas	Desventajas
Gradiente Conjugado	Iterativo	Optimiza la solución para sistemas simétricos y definidos positivos	Muy eficiente en sistemas grandes y dispersos	Solo aplicable a matrices simétricas y positivas definidas

### Conclusiones sobre métodos computacionales para sistemas de ecuaciones lineales:

Los métodos directos, como: Eliminación Gaussiana, Eliminación Gauss-Jordan, Regla de Cramer y Descomposición LU; son ideales para sistemas pequeños o cuando se requiere precisión exacta.

Los métodos iterativos, como: Método de Jacobi, Método de Gauss-Seidel y Gradiente Conjugado; son preferibles en sistemas grandes, dispersos o cuando se busca eficiencia computacional.

La elección depende de la estructura de la matriz, el tamaño del sistema y los recursos computacionales disponibles.

## 5.- Ejercicio 3

Genera un nuevo cuaderno de Jupyter Notebook y resuelve los siguientes sistemas de ecuaciones lineales, encontrando los valores de las variables utilizando diferentes métodos:

a:

$$\begin{aligned}
 x + 2y + 5z - 4w &= 21 \\
 5x + 8y + z + w &= -8 \\
 5x + 7y - 3z + 2w &= 14 \\
 -x + 3y + 9z - w &= 5
 \end{aligned}$$

b:

$$\begin{aligned}
 6x + 8z &= 9 \\
 2x + 2y + z &= 15 \\
 4x + y &= -4
 \end{aligned}$$



3.a

El sistema de ecuaciones lo podemos representar como  $MxV = I$

Entonces  $M = V^{-1}x I$

En donde  $M = [$

[1,2,5,-4],

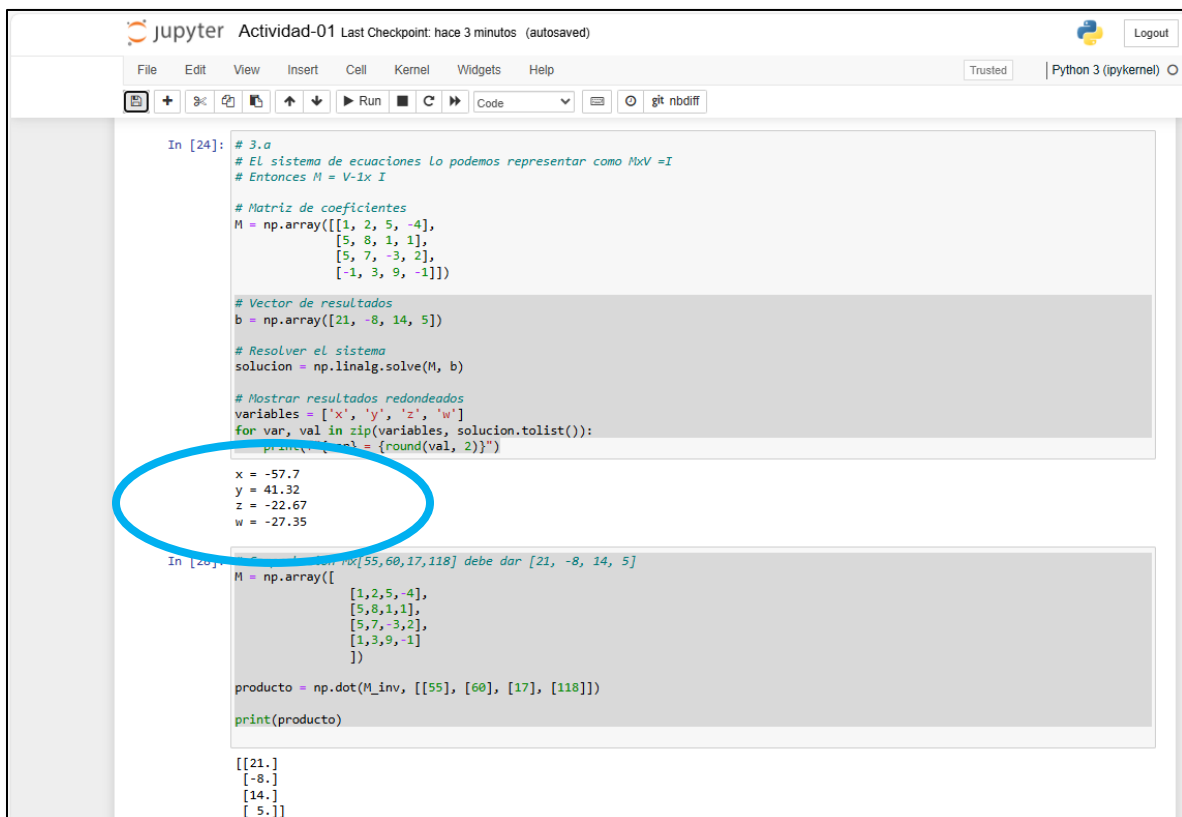
[5,8,1,1],

[5,7,-3,2],

[-1,3,9,-1]

]

La siguiente imagen muestra la codificación y los resultados:



```
In [24]: # 3.a
# El sistema de ecuaciones lo podemos representar como MxV =I
# Entonces M = V-Ix I

# Matriz de coeficientes
M = np.array([[1, 2, 5, -4],
              [5, 8, 1, 1],
              [5, 7, -3, 2],
              [-1, 3, 9, -1]])

# Vector de resultados
b = np.array([21, -8, 14, 5])

# Resolver el sistema
solucion = np.linalg.solve(M, b)

# Mostrar resultados redondeados
variables = ['x', 'y', 'z', 'w']
for var, val in zip(variables, solucion.tolist()):
    print("{} = {}".format(var, round(val, 2)))

x = -57.7
y = 41.32
z = -22.67
w = -27.35

In [26]: # Verificar que Mx[55,60,17,118] debe dar [21, -8, 14, 5]
M = np.array([
    [1,2,5,-4],
    [5,8,1,1],
    [5,7,-3,2],
    [-1,3,9,-1]
])

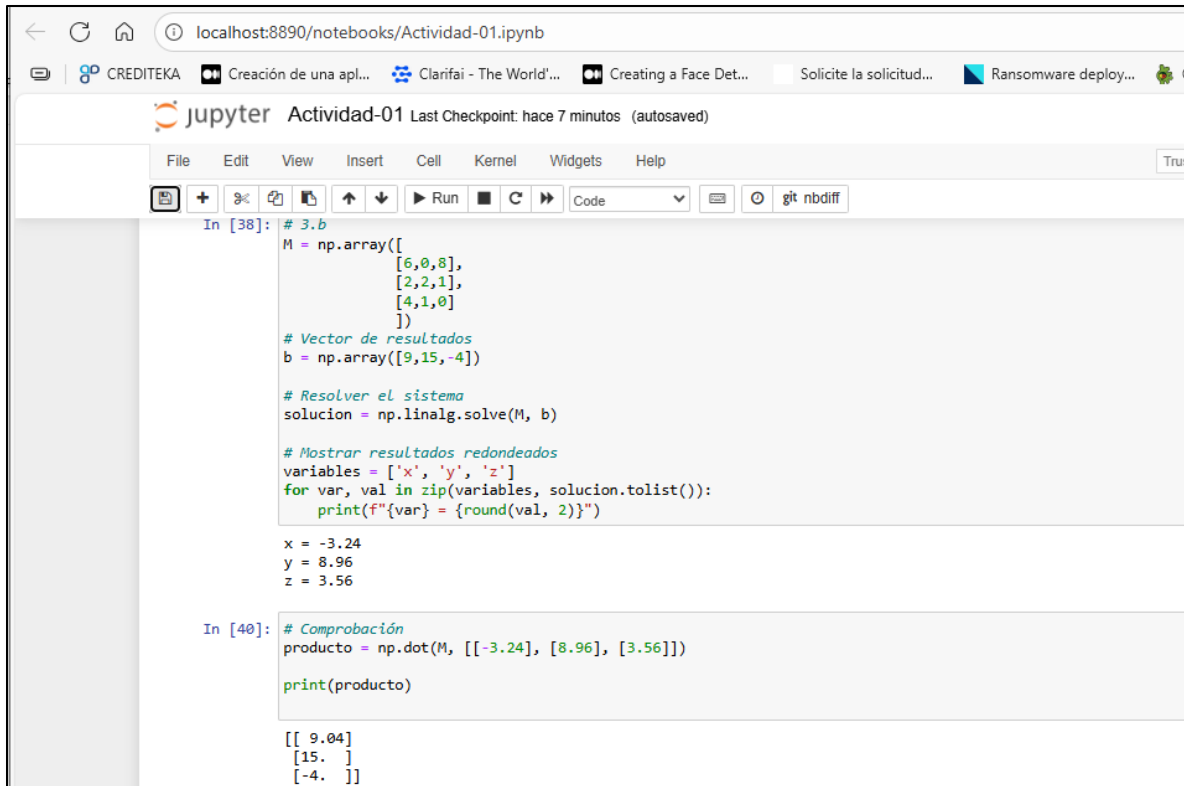
producto = np.dot(M_inv, [[55], [60], [17], [118]])

print(producto)

[[21.]
 [-8.]
 [14.]
 [ 5.]]
```

### 3.b

Aplicamos la misma metodología para encontrar los valores de  $x$ ,  $y$ ,  $z$ .



```
In [38]: # 3.b
M = np.array([
    [6,0,8],
    [2,2,1],
    [4,1,0]
])
# Vector de resultados
b = np.array([9,15,-4])

# Resolver el sistema
solucion = np.linalg.solve(M, b)

# Mostrar resultados redondeados
variables = ['x', 'y', 'z']
for var, val in zip(variables, solucion.tolist()):
    print(f"{var} = {round(val, 2)}")

x = -3.24
y = 8.96
z = 3.56

In [40]: # Comprobación
producto = np.dot(M, [[-3.24], [8.96], [3.56]])

print(producto)

[[ 9.04]
 [15.  ]
 [-4.  ]]
```

## Conclusiones

Las matemáticas son un metalenguaje con un profundo trasfondo filosófico que mantiene la indispensabilidad de las matemáticas como una herramienta esencial para el progreso científico. Aunque este vínculo es menos obvio con el campo más técnico de la lógica matemática, también resulta estar arraigado en la lógica general y, por lo tanto, en la estructura del lenguaje. El pensamiento matemático nos ayuda a modelar y resolver problemas del mundo real utilizando modelos numéricos, geométricos y probabilísticos.

Resolver problemas matemáticos implica un proceso estructurado: comprensión, planificación, cálculo e interpretación.

Las matemáticas computacionales han cambiado drásticamente desde el final de la Segunda Guerra Mundial, y este cambio también está haciendo posible el desarrollo de la inteligencia artificial. El entendimiento del álgebra lineal, el cálculo y la probabilidad es necesario para entender las bases del aprendizaje automático o Machine Learning.

Lo que aprendimos en la segunda mitad de esta lección:

Álgebra lineal y sus fundamentos: Para desarrollar modelos matemáticos, así como aplicaciones de inteligencia artificial, la comprensión del álgebra lineal es un requisito básico. Esa misma definición es la que nos permite usar vectores en diferentes contextos.

Operaciones de matrices en Python: Por ejemplo, la inversa o la transpuesta, lo que facilita la resolución de problemas computacionales complejos. Es un trasfondo teórico básico para estudiar las siguientes estructuras algebraicas más avanzadas: grupos, espacios, subespacios vectoriales.

Software matemático: Las bibliotecas de software matemático permiten concentrarse en la lógica y el razonamiento sobre un cálculo, y no en su instanciación mecánica.

Conexión teoría-práctica: La conexión de la teoría con la práctica computacional acelera el aprendizaje y resuelve mejor los problemas reales.

Los ejercicios prácticos de Jupyter Notebook en Python fueron esenciales para comprender y aplicar conceptos matemáticos a la resolución de problemas. En este trabajo, generamos la matriz cuadrada  $4 \times 4$  mediante este método y encontramos su inversa y la transposición de la matriz, lo que ayuda en la visualización y manipulación de estructuras algebraicas. Multiplicar la matriz por un escalar y realizar un producto interno con otra matriz  $4 \times 2$  ha ayudado a proporcionar una visión de las operaciones básicas de matrices.

En general, el trabajo práctico ha proporcionado esencialmente un medio para aplicar y reforzar la comprensión teórica para ayudar a resolver problemas matemáticos, así como seguir desarrollando el razonamiento analítico y lógico.



## Liga al código

La codificación (actividad-01.ipynb) la podemos encontrar en siguiente liga a Github:

<https://github.com/luisgg121/ML-Actividad-01.git>