ITSRLL
INSTITUTO TECNOLÓGICO SUPERIOR DE LA REGIÓN DE LOS LLANOS

# Ingeniería Mecatrónica

## PROGRAMACIÓN AVANZADA

Enero – Junio 2025

M.C. Osbaldo Aragón Banderas

UNIDAD:

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|

Nombre de actividad:

PROGRAMA. Implementación de Perceptrón

Actividad realizada por:

Diego Alberto Alcaraz Gonzalez

Guadalupe Victoria, Durango

Fecha de entrega:

| 18 | 02 | 2025 |
|---|---|---|

**Introducción**

Se describe la implementación de un perceptrón básico para clasificación binaria, utilizado para predecir la aprobación o rechazo de un préstamo en función de diversas características. El código comienza con la importación de la librería NumPy, que facilita el manejo eficiente de matrices y operaciones matemáticas, lo cual es crucial para entrenar el modelo de perceptrón. Se definen los datos de entrenamiento en una matriz que contiene las características del préstamo, junto con las etiquetas correspondientes, que indican si el préstamo fue aprobado o rechazado. A través de este proceso, se busca que el perceptrón aprenda la relación entre las características y la decisión de aprobación.

## Entorno de programación Jupyter

Jupyter es un entorno interactivo de programación que permite escribir y ejecutar código en Python y otros lenguajes de forma estructurada y visual. Es ampliamente utilizado en ciencia de datos, inteligencia artificial, simulaciones y control de hardware, ya que facilita la integración con bibliotecas como NumPy, Pandas, Matplotlib, TensorFlow y OpenCV. Su interfaz basada en celdas permite organizar el código en fragmentos ejecutables, documentar con Markdown, visualizar gráficos y analizar datos de manera eficiente. Además, se ejecuta en el navegador, lo que facilita su uso remoto y la colaboración. Es especialmente útil en proyectos que requieren análisis de datos, comunicación con microcontroladores como ESP32 y Raspberry Pi, o experimentación con modelos de machine learning. Para instalarlo, basta con ejecutar pip install jupyter, y se inicia con jupyter notebook.

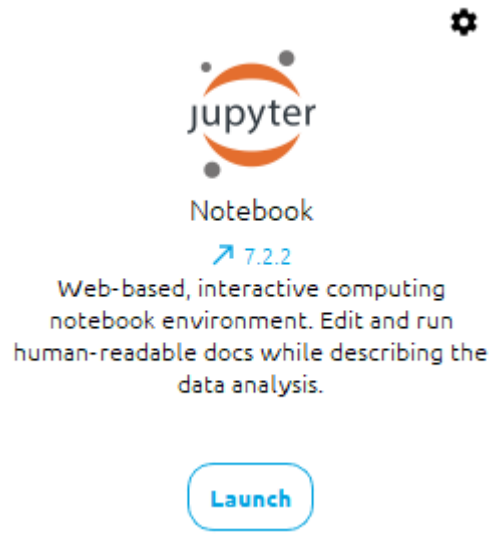


*Ilustración 1: Logo Jupyter*

**Perceptrón**

Un perceptrón es un modelo matemático inspirado en el funcionamiento de las neuronas del cerebro, utilizado para clasificación binaria. Su objetivo es recibir un conjunto de entradas numéricas, asignarles un peso, sumarlas junto con un sesgo (bias) y pasar el resultado por una función de activación (como la función escalón) que decide si la salida es 1 (aprobado) o 0 (rechazado).

El entrenamiento del perceptrón consiste en ajustar los pesos y el bias para minimizar el error entre la salida esperada y la salida predicha. Esto se logra recorriendo los datos en varias épocas, donde cada vez que el perceptrón comete un error, sus pesos se modifican usando una regla de actualización basada en la tasa de aprendizaje.

El perceptrón es útil para problemas linealmente separables, es decir, aquellos en los que se puede trazar una línea (en 2D) o un plano (en 3D) que divida claramente ambas clases. Sin embargo, no puede resolver problemas más complejos, como el XOR, ya que no maneja relaciones no lineales. Para estos casos, se emplean redes neuronales multicapa, que combinan múltiples perceptrones con funciones de activación más avanzadas, como sigmoide o ReLU.

En aplicaciones prácticas, el perceptrón puede utilizarse para toma de decisiones automatizada, como la aprobación de préstamos, detección de spam o reconocimiento de patrones simples en datos.

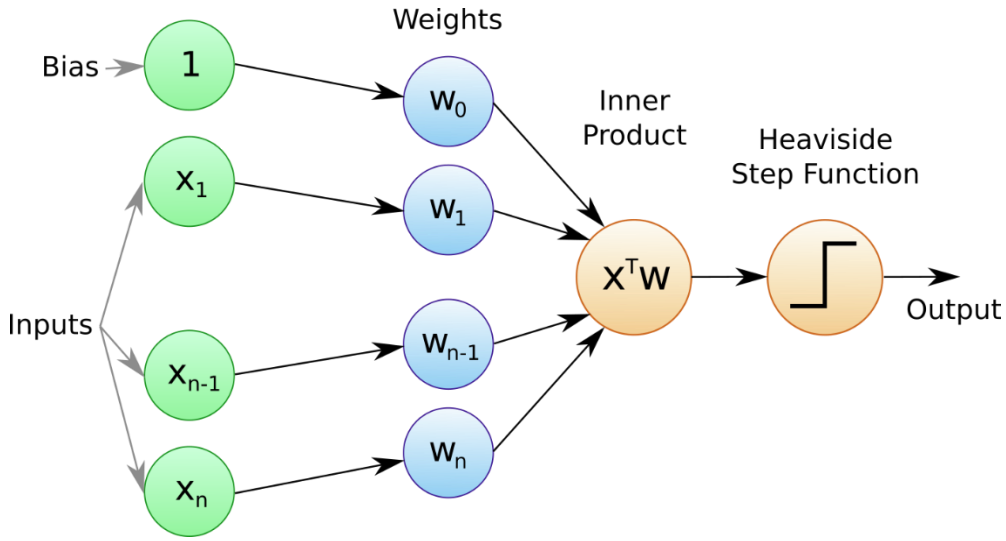


*Ilustración 2: Diagrama de perceptrón*

## Código

```python
import numpy as np

# Datos de entrenamiento modificados
X = np.array([
    [1200, 4.5, 22.0, 0.25],  # Muestra 1
    [950, 3.2, 18.0, 0.45],   # Muestra 2
    [1100, 4.0, 20.0, 0.30],  # Muestra 3
    [850, 2.8, 16.0, 0.50],   # Muestra 4
    [1300, 5.0, 24.0, 0.20]   # Muestra 5
])

# Etiquetas correspondientes
y = np.array([1, 0, 1, 0, 1])  # 1: Aprobado, 0: Rechazado

# Normalización de datos
X = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))

# Parámetros del perceptrón
learning_rate = 0.1
epochs = 10
weights = np.random.rand(X.shape[1])  # Pesos inicializados aleatoriamente
bias = np.random.rand()               # Bias inicializado aleatoriamente

# Función de activación (escalón)
def activation_function(x):
    return 1 if x >= 0 else 0
```

*Ilustración 4: Primera parte del código*

```python
for epoch in range(epochs):
    print(f"Época: {epoch + 1} :")
    for i in range(len(X)):
        # Cálculo de la salida lineal
        linear_output = np.dot(X[i], weights) + bias
        # Predicción usando la función de activación
        prediction = activation_function(linear_output)
        # Cálculo del error
        error = y[i] - prediction

        # Actualización de pesos y bias
        weights += learning_rate * error * X[i]
        bias += learning_rate * error

        # Impresión de resultados para cada muestra
        print(f" Muestra {i+1}: Entrada {X[i]}, Salida esperada {y[i]}, Predicción {prediction}, Error {error}")
    # Impresión de pesos y bias actualizados
    print(f" Pesos actualizados: {weights}, Bias actualizada: {bias}\n")

# Evaluación del modelo con un nuevo dato
print("=========Evaluación del modelo==========")
nuevo_dato = np.array([1000, 3.5, 19.0, 0.35])  # Nuevo dato de prueba
nuevo_dato = (nuevo_dato - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))  # Normalización
Salida = activation_function(np.dot(nuevo_dato, weights) + bias)  # Predicción
print(f"El modelo evaluado para entrada {nuevo_dato} = {Salida}")
if Salida == 1:
    print("El préstamo será aprobado.")
else:
    print("El préstamo será rechazado.")
```

*Ilustración 3: Segunda parte del código*

El código comienza importando la librería NumPy, que es esencial para manejar matrices y realizar operaciones matemáticas eficientes. Esto permite trabajar con datos numéricos de manera optimizada, lo cual es crucial para entrenar el modelo de perceptrón.

Luego, se definen los datos de entrenamiento en la matriz X, donde cada fila representa una muestra con cuatro características: monto del préstamo, tasa de interés, duración en meses y riesgo asociado. Paralelamente, se establece y, un arreglo con las etiquetas correspondientes, donde 1 significa que el préstamo es aprobado y 0 indica que es rechazado. Esto permite al perceptrón aprender una relación entre las características y la decisión de aprobación.

Para mejorar la estabilidad del modelo, se normalizan los datos con la fórmula (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0)). Esta operación reescala los valores de cada característica entre 0 y 1, lo que evita que variables con valores grandes dominen el entrenamiento y ayuda a la convergencia del modelo.

El perceptrón se inicializa con pesos aleatorios para cada característica y un sesgo (bias) aleatorio. Los pesos determinan la importancia de cada característica en la clasificación, mientras que el sesgo ajusta la decisión global del modelo. Se definen también parámetros de entrenamiento como la tasa de aprendizaje (learning_rate), que controla cuánto se modifican los pesos en cada iteración, y el número de épocas (epochs), que indica cuántas veces se recorrerán los datos para ajustar los pesos.

A continuación, se define la función de activación, que en este caso es la función escalón. Esta función toma la salida lineal (una combinación de los valores de entrada y los pesos) y devuelve 1 si es mayor o igual a 0, o 0 en caso contrario. Esto permite al perceptrón clasificar las entradas en dos categorías.

El proceso de entrenamiento se ejecuta en un bucle que itera sobre cada época. En cada iteración, el modelo recorre los datos de entrenamiento, calcula la salida usando la combinación lineal de los pesos y entradas (np.dot(X[i], weights) + bias), y luego aplica la función de activación para obtener la predicción. Si la predicción no coincide con la salida esperada, se actualizan los pesos y el sesgo usando la

regla de aprendizaje del perceptrón, que ajusta los valores en función del error cometido. Al final de cada época, se imprimen los pesos y el sesgo actualizados.

Después del entrenamiento, el código realiza dos evaluaciones con datos nuevos. Primero, se prueba el modelo con un conjunto de características que representan un caso aprobatorio. La entrada se normaliza antes de hacer la predicción y, dependiendo del resultado, se imprime si el préstamo es aprobado o rechazado. Luego, se realiza una segunda evaluación con un caso no aprobatorio para verificar que el modelo también es capaz de identificar correctamente situaciones donde el préstamo debe ser rechazado.

## Conclusión

El perceptrón implementado en este ejercicio permite observar los fundamentos del aprendizaje supervisado aplicados a problemas de clasificación binaria. A través de un ciclo de entrenamiento, el modelo ajusta sus pesos y sesgo para mejorar la precisión en las predicciones. Las evaluaciones realizadas con datos nuevos demuestran la capacidad del modelo para tomar decisiones, como la aprobación o el rechazo de un préstamo. Aunque se trata de un enfoque simple, este ejercicio proporciona una comprensión básica de los mecanismos detrás del aprendizaje automático y su potencial aplicación en la toma de decisiones financieras.

## Bibliografía

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2013). Introducción a los algoritmos (3.ª ed.). Alfaomega.

Knuth, D. E. (2000). El arte de la programación de computadores, Volumen 3: Ordenamiento y búsqueda (2.ª ed.). Addison-Wesley.