

INTELIGENCIA ARTIFICIAL GENERATIVA

PROGRAMA PYTHON O.O DEL PERCEPTRON

Vamos a generar paso a paso, usando programación orientada a objetos con Python y la herramienta ChatGPT, el programa del **perceptrón**.

Diálogo con el ChatGPT

Primera entrada

“Por favor necesito generar un programa orientado a objetos con Python, redes neuronales, aprendizaje automático, para resolver el OR, con las siguientes características. Es el perceptrón con una neurona de salida y varias de entrada, para entrenar una red neuronal que tiene como entrada 00, 01, 10, 11 y como salida: 0, 1, 1,1. El programa no debe tener librerías especiales, solo con Python OO, ejemplo de uso, etc...”

Segunda entrada

“Todo está muy bien, pero tengo una pregunta: por qué no veo la función escalón de salida”

Tercer entrada

“Presenta este aviso en pantalla:

```
print(f"Epoch {epoch + 1}") ^ SyntaxError: unexpected character after line”
```

ChatGPT generó el programa correcto de Perceptrón usando el lenguaje Python orientado a objetos

A continuación, se presenta el código Python O.O. completo del perceptrón revisado y corregido

```
# -*- coding: utf-8 -*-
"""
Creado Febrero 28 de 2025
Nombre: miprimeraOR
Que hace: red neuronal perceptron
Autores: Ms. Hernando Álvarez R.
         Ms. Leonardo Álvarez V.
         Ms. Luis Fernando Álvarez V.
Universidad Tecnológica de Pereira
Código construido en ChatGPT
"""

class Perceptron:
    def __init__(self, input_size, learning_rate=0.1, epochs=10):
        """Inicializa el perceptrón.
        Args:
            input_size (int): Número de entradas.
            learning_rate (float): Tasa de aprendizaje.
            epochs (int): Número de iteraciones para el entrenamiento.
        """
        self.weights = [0.0] * (input_size + 1) # Pesos, incluyendo el sesgo
        self.learning_rate = learning_rate
        self.epochs = epochs

    def step_function(self, x):
        """Función escalón."""
        return 1 if x >= 0 else 0

    def predict(self, inputs):
        """Realiza una predicción para una entrada dada.
        Args:
            inputs (list): Lista de entradas.
        Returns:
            int: Salida (0 o 1).
        """
        summation = sum(w * x for w, x in zip(self.weights[:-1], inputs)) +
self.weights[-1]
        return self.step_function(summation)

    def train(self, training_inputs, labels):
        """Entrena el perceptrón utilizando los datos de entrenamiento.
        Args:
            training_inputs (list): Lista de listas con las entradas de
entrenamiento.
            labels (list): Lista con las salidas deseadas.
        """
        for epoch in range(self.epochs):
            print(f"Epoch {epoch + 1}")
```

```
for inputs, label in zip(training_inputs, labels):
    prediction = self.predict(inputs)
    error = label - prediction

    # Actualización de pesos y sesgo
    self.weights[:-1] = [w + self.learning_rate * error * x for
w, x in zip(self.weights[:-1], inputs)]
    self.weights[-1] += self.learning_rate * error

    print(f"Entrada: {inputs}, Predicción: {prediction}, Error:
{error}, Pesos: {self.weights}")

# Ejemplo de uso
def main():
    # Entradas y salidas para la operación OR
    training_inputs = [[0, 0], [0, 1], [1, 0], [1, 1]]
    labels = [0, 1, 1, 1]

    # Crear el perceptrón
    perceptron = Perceptron(input_size=2, learning_rate=0.1, epochs=10)

    # Entrenar el perceptrón
    perceptron.train(training_inputs, labels)

    # Probar el perceptrón
    print("\nResultados de predicción:")
    for inputs in training_inputs:
        print(f"Entrada: {inputs}, Salida predicha:
{perceptron.predict(inputs)}")

if __name__ == "__main__":
    main()
```