# Técnico en Desarrollo de Software

## Nombre: Jafet Abimael Escobar Salazar Carnet: 25002891

### Actividad Extra #1:

**Descripción breve del problema**Imagina que tenemos un circuito lógico (por ejemplo, una compuerta AND, OR o XOR) y queremos que una neurona o perceptrón en Python aprenda a “predecir” el resultado de este circuito. El desafío radica en configurar y entrenar adecuadamente los pesos y el sesgo (bias) de esta neurona, de manera que, al recibir como entrada los valores lógicos (0 o 1) de nuestro circuito, la neurona sea capaz de aproximar correctamente la operación booleana que se desea emular.

#### Algoritmo (pasos simples)

1. **Definir la tabla de verdad**
   * Identificar las entradas (por ejemplo, x1x\_1 y x2x\_2) y la salida esperada según el circuito lógico (e.g., AND, OR, XOR, etc.).
   * Crear el conjunto de entrenamiento con cada combinación de entradas y su salida deseada.
2. **Inicializar los parámetros**
   * Asignar valores aleatorios (o en cero) a los pesos (w1,w2w\_1, w\_2) y al sesgo (bias, bb).
   * Definir una tasa de aprendizaje (learning rate), por ejemplo 0.1.
3. **Proceso de entrenamiento**
   * **Para** cada época de entrenamiento (p.ej., 100 iteraciones):
     1. Recorrer cada par de entrada (x1,x2)(x\_1, x\_2) en el conjunto de entrenamiento.
     2. Calcular la salida de la neurona: salida=f(w1⋅x1+w2⋅x2+b)salida = f(w\_1 \cdot x\_1 + w\_2 \cdot x\_2 + b) Donde ff suele ser una función de activación escalón (por ejemplo, 1 si el resultado es mayor o igual que 0, y 0 de lo contrario).
     3. Comparar la salida obtenida con la salida esperada (de la tabla de verdad).
     4. Actualizar los pesos y el sesgo si hay un error: wi←wi+η×(salidaEsperada−salidaReal)×xiw\_i \leftarrow w\_i + \eta \times (salidaEsperada - salidaReal) \times x\_i b←b+η×(salidaEsperada−salidaReal)b \leftarrow b + \eta \times (salidaEsperada - salidaReal) Donde η\eta es la tasa de aprendizaje.
4. **Verificación de convergencia**
   * Al finalizar cada época o luego de un cierto número de iteraciones, verificar si la neurona clasifica correctamente todas las combinaciones de entrada.
   * Si todas son correctas, puede darse por concluido el entrenamiento antes de terminar todas las épocas.
5. **Prueba (inferencia)**
   * Con los pesos y bias finales, presentar cada combinación de entrada (o nuevas entradas) a la neurona.
   * Calcular la salida usando la misma función de activación.
   * Verificar que la salida coincida con la lógica deseada (p.ej., que la neurona realmente simule la compuerta AND, OR, XOR, etc.).
6. **Ajustes finales**
   * Si el resultado no es satisfactorio, se puede ajustar la tasa de aprendizaje o incrementar el número de épocas para mejorar la convergencia.
   * Guardar los valores de pesos y bias para su uso en producción o como parte de otro sistema que requiera esta predicción lógica.