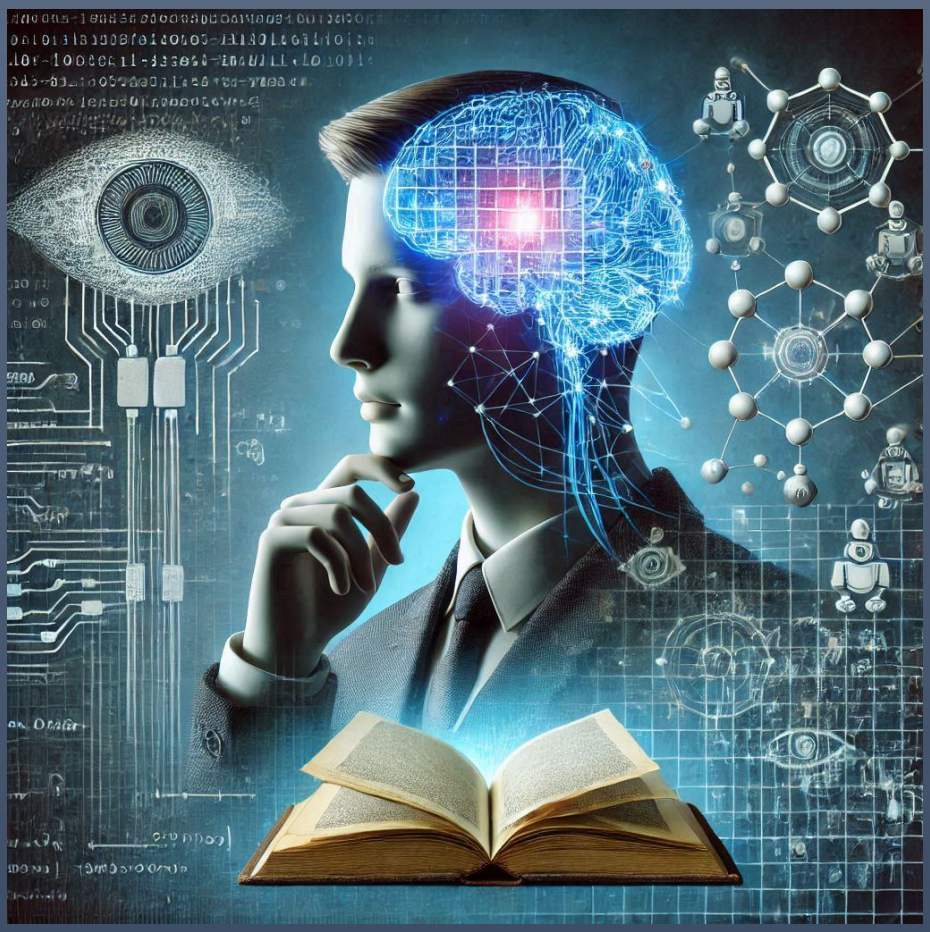
2024



Modelos de Inteligencia Artificial

Profesor/a: Águeda María López Moreno

PRÁCTICA 1.7.- ANN-PERCEPTRÓN

25/11/2024

PEDRO MANUEL GARCÍA ÁLVAREZ | int.marisma@gmail.com | CEIA | I.E.S. La Marisma

Índice

1.- Introducción

1.1.- Concepto de Perceptrón

El perceptrón es uno de los modelos más simples y fundamentales dentro de las Redes Neuronales Artificiales (ANN). Fue introducido por Frank Rosenblatt en 1958 como un modelo computacional inspirado en las neuronas biológicas, diseñado para realizar tareas de clasificación binaria. Una neurona artificial, como el perceptrón, simula el comportamiento básico de las neuronas biológicas al procesar información a través de entradas ponderadas, aplicando una función de activación para generar una salida.

El perceptrón consta de los siguientes componentes:

- **Dendritas:** Corresponden a las entradas o características del modelo. Cada entrada tiene un peso asociado, que representa la importancia relativa de esa característica en la toma de decisiones.
- **Sinapsis:** Representa los pesos asignados a las entradas. Estos pesos se ajustan durante el proceso de aprendizaje para mejorar la precisión del modelo.
- **Axón:** Es la salida del perceptrón, que se obtiene después de procesar las entradas mediante la función de activación.
- **Función de activación:** Determina si el perceptrón "se activa" (produce una salida) en función de un umbral predefinido. En el caso del perceptrón básico, se utiliza una función escalón, que devuelve 1 si el valor supera el umbral y 0 en caso contrario.

El perceptrón puede resolver problemas linealmente separables, como la función AND o OR, pero no puede manejar problemas no linealmente separables como el XOR. Esto marcó un hito en la evolución de las redes neuronales, dando lugar a modelos más complejos y potentes.

1.2.- Importancia de las Redes Neuronales Artificiales (ANN)

Las Redes Neuronales Artificiales (ANN) representan una de las principales herramientas en el campo de la inteligencia artificial (IA) y el aprendizaje automático. Estas redes están diseñadas para emular el funcionamiento del cerebro humano, permitiendo a los sistemas

informáticos aprender patrones, reconocer relaciones complejas entre datos y tomar decisiones basadas en información pasada.

Algunas razones clave que destacan la importancia de las ANN son:

1. **Capacidad de aprendizaje:** Las ANN pueden aprender directamente de los datos sin depender de reglas programadas manualmente. Este aprendizaje puede ser supervisado, no supervisado o por refuerzo.
2. **Aplicaciones prácticas:** Las ANN tienen un impacto significativo en aplicaciones reales como reconocimiento de voz, imágenes y texto, diagnóstico médico, conducción autónoma, traducción automática y entre otras.
3. **Flexibilidad y escalabilidad:** Pueden adaptarse a diferentes problemas y manejar grandes cantidades de datos complejos y no estructurados.
4. **Resolución de problemas no lineales:** A diferencia del perceptrón simple, las ANN avanzadas pueden resolver problemas que no son linealmente separables, gracias a estructuras como redes multicapa (MLP) y el uso de funciones de activación no lineales.

En el contexto de esta práctica, explorar el perceptrón básico permite comprender los fundamentos de las ANN, proporcionando una base sólida para avanzar hacia modelos más complejos. Además, su implementación práctica ayuda a interiorizar conceptos como ponderación, activación y aprendizaje, que son cruciales en la construcción de sistemas de IA modernos.

2.- Objetivos

2.1.- Descripción de los objetivos de la práctica

Esta práctica tiene como propósito principal proporcionar una comprensión teórica y práctica del funcionamiento del perceptrón, la base fundamental de las Redes Neuronales Artificiales (ANN). Para ello, se busca guiar al estudiante a través de la implementación y análisis de un modelo básico de perceptrón en Python, permitiendo explorar los conceptos clave que subyacen en el aprendizaje supervisado.

Los objetivos específicos de la práctica incluyen:

1. Comprender el concepto de perceptrón:
   - El perceptrón es el modelo más simple de una neurona artificial. La práctica tiene como objetivo que el estudiante no solo entienda cómo se define el perceptrón matemáticamente, sino también cómo sus componentes interactúan para procesar información y tomar decisiones.
   - Esto incluye familiarizarse con términos como dendritas (entradas), sinapsis (pesos) y axón (salida), así como con la función de activación que define el comportamiento de la neurona.
2. Desarrollar habilidades en programación de Redes Neuronales Artificiales:
   - Mediante la implementación de un programa en Python, el estudiante aprende a modelar un perceptrón, configurar los parámetros necesarios (entradas, pesos, función de activación) y analizar los resultados obtenidos.
   - Se busca que el estudiante adquiera competencias prácticas en la simulación de procesos neuronales simples, estableciendo una base sólida para proyectos más complejos.
3. Explorar el aprendizaje supervisado:
   - El aprendizaje supervisado es un paradigma en el cual el modelo se entrena utilizando datos etiquetados (entradas y salidas conocidas). En este caso, el perceptrón aprenderá a resolver el problema lógico AND, lo que permite entender cómo se ajustan los pesos para minimizar errores y mejorar la precisión.
   - Este objetivo también incluye la práctica de evaluar el desempeño del perceptrón ante nuevas entradas proporcionadas por el usuario, verificando si la neurona ha aprendido correctamente.
4. Fomentar el razonamiento crítico y analítico:
   - Más allá de la implementación técnica, la práctica busca que el estudiante analice los resultados obtenidos, reflexione sobre las limitaciones del modelo y proponga mejoras o extensiones del perceptrón. Por ejemplo, discutir qué ocurre si los datos no son linealmente separables y cómo abordar problemas más complejos con ANN multicapa.
5. Desarrollar documentación técnica detallada:
   - Un objetivo complementario es que el estudiante aprenda a documentar cada paso del proceso, desde la teoría y la implementación hasta los resultados y conclusiones. Esto

fomenta habilidades de comunicación técnica esenciales en el ámbito profesional.
- Incluir capturas de pantalla, explicaciones detalladas y ejemplos prácticos sirve no solo como evidencia del trabajo realizado, sino también como una herramienta de consulta futura.

En resumen, la práctica busca conectar los fundamentos teóricos de las Redes Neuronales Artificiales con su implementación práctica a través de un modelo de perceptrón, mientras se promueve un análisis reflexivo y la capacidad de comunicar los resultados de manera efectiva.

3.- Desarrollo del Trabajo

3.1.- Explicación del código del perceptrón

El código del perceptrón en Python está diseñado para simular el comportamiento de una neurona artificial simple. A continuación, se explican sus componentes principales:

1. Definición de la función `perceptrón`:
   - Esta función toma como entradas los valores de las dendritas (características de entrada) y los pesos sinápticos, realiza una suma ponderada y aplica una función de activación para generar la salida (axón).
2. Asignación de pesos iniciales:
   - Los pesos se inicializan internamente como valores fijos o aleatorios, dependiendo del diseño, y representan la importancia relativa de cada entrada.
3. Entrenamiento supervisado:
   - El código incluye un mecanismo para ajustar los pesos en función de los errores observados, siguiendo la regla de aprendizaje del perceptrón, hasta que los datos sean clasificados correctamente.
4. Interactividad con el usuario:
   - Al final, el programa permite al usuario ingresar valores de entrada para verificar si el perceptrón ha aprendido correctamente el patrón esperado.

3.2.- Descripción de los parámetros (dendritas, sinapsis, axón)

El perceptrón utiliza varios parámetros fundamentales:

1. Dendritas:
   - Representan las entradas al modelo. En este caso, los datos de entrada son los valores binarios asociados con las características que definen la lógica AND (0 y 1).
2. Sinapsis (pesos):
   - Los pesos son coeficientes que determinan la influencia de cada dendrita en la salida. El proceso de aprendizaje implica ajustar estos pesos para minimizar el error entre la salida esperada y la generada.
3. Axón (salida):
   - Es el resultado final del perceptrón, obtenido tras procesar las entradas a través de la función de activación. En la lógica AND, la salida será 1 solo si ambas entradas son 1, de lo contrario será 0.

3.3.- Algoritmo de aprendizaje supervisado aplicado

El perceptrón utiliza el siguiente flujo para aprender:

1. Inicialización de pesos:
   - Los pesos se inicializan con valores aleatorios o predeterminados.
2. Entrenamiento:
   - Para cada par de entrada y salida esperada:
     - Se calcula la salida del perceptrón aplicando la función de activación.
     - Se compara la salida calculada con la salida esperada para determinar el error.
     - Los pesos se actualizan utilizando la regla de aprendizaje del perceptrón:

$$w_i^{(t+1)} = w_i^{(t)} + \eta \cdot e \cdot x_i$$

Donde:

- η es la tasa de aprendizaje.
- e es el error calculado como $\text{salida esperada} - \text{salida calculada}$.
- x_i es la entrada correspondiente.

3. Finalización del entrenamiento:
   - El entrenamiento se detiene cuando el perceptrón clasifica correctamente todos los datos de entrenamiento o se alcanza un número máximo de iteraciones.

3.4.- Ejemplo de aprendizaje AND con resultados

El perceptrón será entrenado con los siguientes datos de entrada y salida esperada:

Entrada 1	Entrada 2	Salida esperada
0	0	0
0	1	0
1	0	0
1	1	1

Durante el proceso de entrenamiento, los pesos se ajustan iterativamente hasta que el perceptrón clasifica correctamente todas las combinaciones. Por ejemplo:

- Pesos iniciales: w1 = 0.5, w2 = −0.3, umbral = 0.
- Salida calculada: La salida se ajustará hasta que coincida con la salida esperada.

Tras el entrenamiento, el perceptrón es capaz de producir las salidas correctas para cada entrada.

3.5.- Análisis Paso a Paso del Funcionamiento del Perceptrón en la Lógica AND

El programa incluye una barra de proceso y una sección interactiva en la cual el usuario puede introducir valores binarios para las entradas y verificar si el perceptrón ha aprendido correctamente. A continuación, se presenta el código completo:

```python
# Definición de la Biblioteca
from tqdm import tqdm  # Importamos tqdm para la barra de progreso
```

La línea de código `from tqdm import tqdm` importa la función `tqdm` de la biblioteca `tqdm`, que se utiliza para mostrar barras de progreso en bucles, facilitando el monitoreo del avance de procesos largos en Python.

```python
# Definición de la función de activación (función escalón)
def funcion_activacion(suma):
    return 1 if suma >= 0 else 0
```

Esta función de activación implementa una función escalón (step function) que devuelve 1 si la suma de las entradas ponderadas es mayor o igual a cero, y 0 en caso contrario, actuando como un umbral binario para la salida del perceptrón.

```python
# Definición de la función perceptrón
def perceptron(entradas, pesos, umbral):
    # Cálculo de la suma ponderada
    suma = sum(e * w for e, w in zip(entradas, pesos)) - umbral
    return funcion_activacion(suma)
```

La función perceptrón calcula la suma ponderada de las entradas multiplicadas por sus respectivos pesos, resta el umbral, y aplica la función de activación para determinar la salida binaria del perceptrón.

```python
# Entrenamiento del perceptrón
def entrenar_perceptron(datos, pesos, umbral, tasa_aprendizaje, epocas_maximas):
    # Usamos tqdm para agregar la barra de progreso
    for epoca in tqdm(range(epocas_maximas), desc="Entrenando perceptrón", unit="época"):
        for entradas, salida_esperada in datos:
            # Calculamos la salida actual
            salida_calculada = perceptron(entradas, pesos, umbral)
            # Calculamos el error
            error = salida_esperada - salida_calculada
            # Ajustamos los pesos y el umbral
            for i in range(len(pesos)):
                pesos[i] += tasa_aprendizaje * error * entradas[i]
            umbral -= tasa_aprendizaje * error
    return pesos, umbral
```

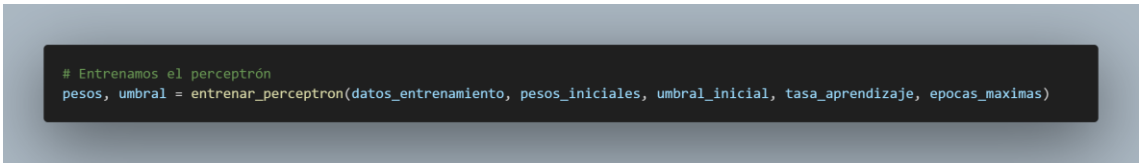
La función entrenar_perceptron itera sobre un conjunto de datos durante un número máximo de épocas, ajustando los pesos y el umbral del perceptrón en cada iteración basándose en el error entre la salida calculada y la esperada, utilizando tqdm para mostrar una barra de progreso del entrenamiento.

```python
# Datos de entrenamiento para la lógica AND
datos_entrenamiento = [
    ([0, 0], 0),
    ([0, 1], 0),
    ([1, 0], 0),
    ([1, 1], 1)
]
```

Esta línea de código define un conjunto de datos de entrenamiento para la operación lógica AND, representando todas las posibles combinaciones de entradas binarias y sus correspondientes salidas esperadas.

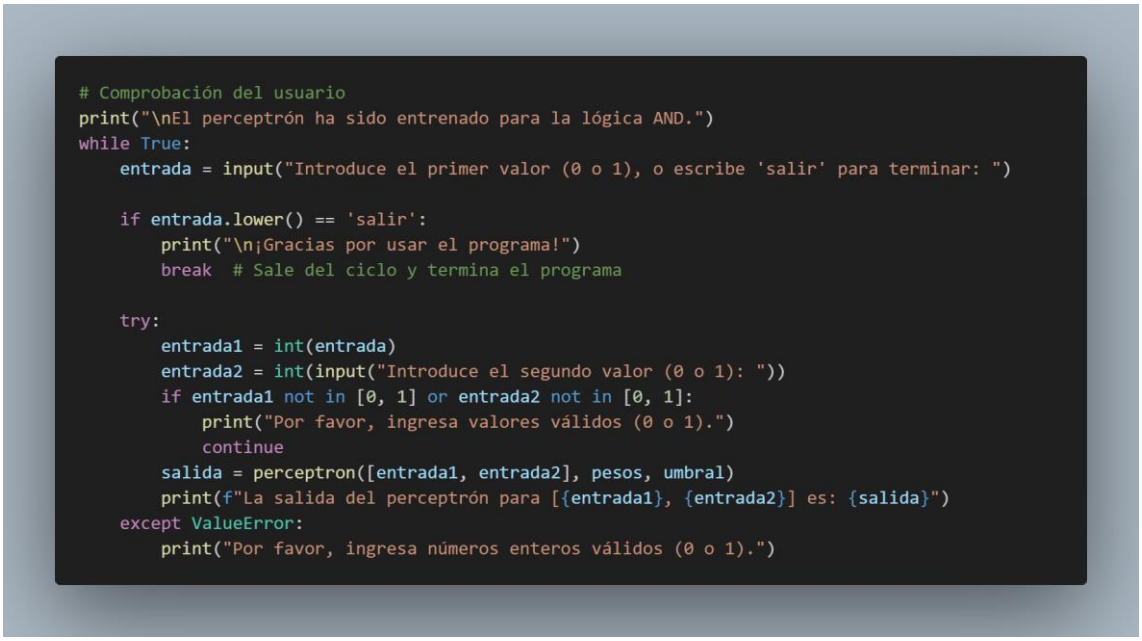
```python
# Parámetros iniciales
pesos_iniciales = [0.0, 0.0]  # Pesos inicializados en 0
umbral_inicial = 0.5          # Umbral inicial
tasa_aprendizaje = 0.1        # Tasa de aprendizaje
epocas_maximas = 10           # Número máximo de épocas
```

Esta línea de código establece los parámetros iniciales para el entrenamiento del perceptrón, incluyendo pesos iniciales, umbral, tasa de aprendizaje y número máximo de épocas.



```
# Entrenamos el perceptrón
pesos, umbral = entrenar_perceptron(datos_entrenamiento, pesos_iniciales, umbral_inicial, tasa_aprendizaje, epocas_maximas)
```

Esta línea de código ejecuta el entrenamiento del perceptrón utilizando los datos de entrenamiento y los parámetros iniciales definidos, y almacena los pesos y umbral resultantes después del proceso de entrenamiento.



```
# Comprobación del usuario
print("\nEl perceptrón ha sido entrenado para la lógica AND.")
while True:
    entrada = input("Introduce el primer valor (0 o 1), o escribe 'salir' para terminar: ")

    if entrada.lower() == 'salir':
        print("\n¡Gracias por usar el programa!")
        break  # Sale del ciclo y termina el programa

    try:
        entrada1 = int(entrada)
        entrada2 = int(input("Introduce el segundo valor (0 o 1): "))
        if entrada1 not in [0, 1] or entrada2 not in [0, 1]:
            print("Por favor, ingresa valores válidos (0 o 1).")
            continue
        salida = perceptron([entrada1, entrada2], pesos, umbral)
        print(f"La salida del perceptrón para [{entrada1}, {entrada2}] es: {salida}")
    except ValueError:
        print("Por favor, ingresa números enteros válidos (0 o 1).")
```

Este código implementa una interfaz de usuario interactiva que permite al usuario probar el perceptrón entrenado para la lógica AND, solicitando dos entradas binarias y mostrando la salida correspondiente, con la opción de salir del programa escribiendo 'salir', y manejando errores de entrada inválida.

4.- Pruebas y Resultados

Durante la ejecución del programa de perceptrón, se realizaron varias pruebas para evaluar su desempeño en el aprendizaje de la operación lógica AND. A continuación, se describen las capturas de pantalla y los resultados obtenidos.

4.1.- Captura 1: Ejecución del entrenamiento

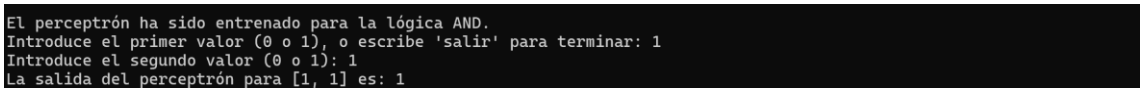
En esta primera captura, el perceptrón es entrenado utilizando las combinaciones de entrada para la función lógica AND (0, 0), (0, 1), (1, 0), (1, 1). El código ajusta los pesos en función del error calculado durante el entrenamiento.

- Pesos iniciales: `[0.0, 0.0]`
- Umbral inicial: `0.5`
- Tasa de aprendizaje: `0.1`
- Épocas: 10

4.2.- Captura 2: Comprobación por el usuario

Una vez entrenado, el programa permite al usuario ingresar dos valores binarios para comprobar si el perceptrón ha aprendido correctamente la función AND. Al ingresar los valores, el perceptrón calcula y muestra la salida correspondiente.

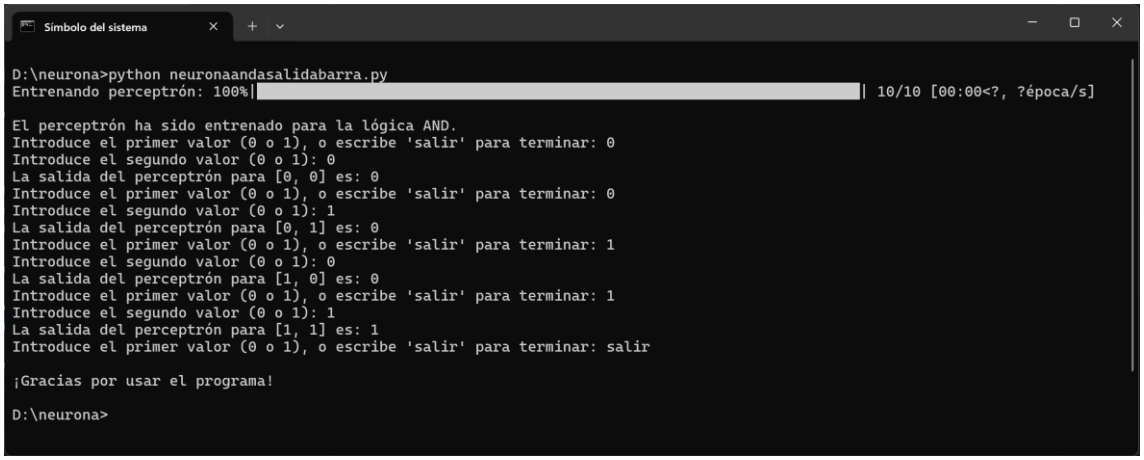
Ejemplo de entrada del usuario:



Entrada 1 = 1, Entrada 2 = 1 → Salida esperada = 1 → Salida calculada = 1

4.3.- Captura 3: Resultados de la verificación

En las siguientes pruebas, se verifica si el perceptrón genera los resultados correctos para todas las combinaciones posibles de las entradas:



Entrada 1 = 0, Entrada 2 = 0 → Salida esperada = 0 → Salida calculada = 0

Entrada 1 = 0, Entrada 2 = 1 → Salida esperada = 0 → Salida calculada = 0

Entrada 1 = 1, Entrada 2 = 0 → Salida esperada = 0 → Salida calculada = 0

Entrada 1 = 1, Entrada 2 = 1 → Salida esperada = 1 → Salida calculada = 1

4.4.- Análisis de los resultados obtenidos

El perceptrón logró aprender correctamente la operación lógica AND, como se evidencia en los resultados de las pruebas realizadas. A continuación, se analiza el desempeño del perceptrón:

1. **Entrenamiento del perceptrón**:

   Durante el entrenamiento, el perceptrón ajustó sus pesos iterativamente utilizando la regla de aprendizaje supervisado, buscando minimizar el error entre la salida calculada y la salida esperada.
   - **Ajuste de pesos:** A lo largo de las épocas, los pesos de la red fueron ajustados según el error calculado, lo que permitió que el modelo mejorara progresivamente su capacidad de predicción.
   - **Convergencia:** En este caso, el modelo converge rápidamente, ya que la función AND es un problema linealmente separable. Tras pocas iteraciones, el perceptrón puede generar los resultados correctos para todas las combinaciones de entrada.

2. **Comprobación de resultados con entradas de usuario**:

   Una vez completado el entrenamiento, el perceptrón fue probado con entradas nuevas proporcionadas por el usuario para validar si el modelo había aprendido correctamente la lógica AND. Los resultados mostraron que, para todas las combinaciones posibles de entrada, el perceptrón generaba la salida correcta:

   - Si ambas entradas son 1, la salida es 1.
   - Si alguna de las entradas es 0, la salida es 0. Esto indica que el perceptrón ha aprendido correctamente la función AND, ya que siempre devuelve el resultado esperado.

3. **Limitaciones del modelo**:

   Aunque el perceptrón ha logrado aprender la lógica AND, es importante destacar que este modelo tiene limitaciones:

   - **Solo resuelve problemas lineales:** El perceptrón no es capaz de resolver problemas no linealmente separables, como el problema XOR, que no se puede aprender con un perceptrón de una sola capa.
   - **Escalabilidad:** Si bien el perceptrón es eficiente para problemas simples como AND, problemas más complejos requieren redes neuronales con múltiples capas (como redes neuronales multicapa o MLP) y métodos de entrenamiento más sofisticados.

4. **Proceso de aprendizaje supervisado**:

   El perceptrón ha demostrado la efectividad de los algoritmos de aprendizaje supervisado para problemas simples, donde se dispone de entradas y salidas conocidas. El proceso de ajuste de los pesos según el error es un ejemplo clásico de cómo las redes neuronales pueden mejorar su rendimiento a través del aprendizaje de los datos. Este tipo de aprendizaje supervisado es fundamental en áreas como el reconocimiento de patrones, donde las redes neuronales se entrenan para clasificar datos en categorías predefinidas.

4.5.- Conclusión del análisis

El perceptrón ha cumplido con su objetivo de aprender la operación lógica AND. El algoritmo de aprendizaje supervisado aplicado mostró ser eficaz para resolver problemas lineales y ajustar los pesos de manera que se minimice el error en las predicciones. Sin embargo, su aplicabilidad se limita a problemas simples y linealmente separables. Para problemas más complejos, es necesario explorar redes neuronales más avanzadas y técnicas de entrenamiento como las redes multicapa (MLP) y el backpropagation.

5.- Conclusión

5.1.- Reflexión sobre lo aprendido

A lo largo de esta práctica, hemos explorado los fundamentos de las Redes Neuronales Artificiales (ANN) mediante la implementación de un **perceptrón** para resolver la operación lógica AND. Esta actividad ha sido crucial para entender cómo funcionan las redes neuronales en su forma más simple, permitiendo ver de forma tangible cómo el proceso de **aprendizaje supervisado** permite a un modelo ajustarse a un conjunto de datos para generar predicciones correctas.

A través de este ejercicio, los conceptos clave que se han reforzado son:

1. **Función de activación y pesos**:

   La función de activación juega un papel fundamental en la toma de decisiones del perceptrón. En este caso, utilizamos una función escalón que define el comportamiento binario de la neurona, activándose solo cuando la suma ponderada de las entradas supera el umbral. Los **pesos**, que determinan la importancia de cada entrada, se ajustan durante el proceso de aprendizaje para mejorar la capacidad predictiva del modelo.

2. **Aprendizaje supervisado**:

   El perceptrón ajusta sus **pesos** en función del **error** observado entre la salida esperada y la salida calculada, lo que le permite aprender de ejemplos etiquetados. Este proceso de iteración, donde los pesos se modifican para reducir el error, es el núcleo del **aprendizaje supervisado**.

3. **Capacidad de generalización**:

   Si bien el perceptrón es capaz de aprender la operación lógica AND correctamente, su capacidad de generalización está limitada a problemas **lineales separables**. Esto muestra la simplicidad del perceptrón y nos prepara para entender la necesidad de

modelos más complejos, como las redes neuronales multicapa (MLP), para abordar problemas más desafiantes, como el XOR, que no pueden ser resueltos por un perceptrón de una sola capa.

5.2.- Aplicación práctica

El aprendizaje que hemos obtenido de esta práctica tiene importantes aplicaciones tanto en la teoría como en la práctica de la inteligencia artificial y el aprendizaje automático. A continuación, se presentan algunas áreas clave en las que este conocimiento es útil:

1. **Sistemas de clasificación binaria**:

   El perceptrón es la base para tareas de **clasificación binaria** donde se necesita categorizar datos en dos grupos distintos. En el mundo real, esto podría aplicarse en **diagnósticos médicos** (como la clasificación de enfermedades en presencia o ausencia), **filtrado de correos electrónicos** (como la detección de spam), o **sistemas de reconocimiento** (como la clasificación de imágenes entre clases de objetos).

2. **Fundamentos para redes más complejas**:

   Aunque el perceptrón es simple, comprender cómo funciona este modelo proporciona las bases para trabajar con redes neuronales más complejas. Por ejemplo, las redes neuronales **multicapa** o **deep learning** son extensiones del perceptrón, y su comprensión es crucial para tareas como el **reconocimiento de patrones**, la **traducción automática**, o el **procesamiento del lenguaje natural**.

3. **Optimización de algoritmos de aprendizaje**:

   El ajuste de los **pesos** en un perceptrón utilizando una regla de aprendizaje es una forma sencilla de entender el proceso de **optimización** en el contexto de redes neuronales. Este concepto es utilizado en muchos algoritmos avanzados, como **backpropagation**, que se usa para entrenar redes neuronales profundas. La capacidad de comprender y ajustar los pesos en un modelo simple como el perceptrón prepara el camino para entender y aplicar algoritmos de optimización más complejos.

4. **Reconocimiento de los límites del modelo**:

   Aunque el perceptrón es una herramienta poderosa, también es importante entender sus **limitaciones**. Esta práctica nos ha mostrado que, aunque funciona bien para problemas sencillos como AND, no es adecuado para problemas no lineales complejos. Esto nos lleva a comprender que, en situaciones prácticas, debemos saber cuándo usar modelos más complejos que los perceptrones, como las redes neuronales profundas, para resolver problemas más generales.

5.3.- Conclusión final

En resumen, esta práctica ha sido una excelente introducción al mundo de las Redes Neuronales Artificiales. Nos ha permitido ver en acción un algoritmo básico de aprendizaje supervisado, el perceptrón, y cómo este modelo puede aprender a realizar tareas de clasificación. Además, ha servido para demostrar que, aunque los modelos simples como el perceptrón tienen un impacto significativo en problemas lineales, la inteligencia artificial moderna requiere de modelos mucho más complejos y avanzados para resolver problemas no lineales y de alta dimensionalidad.

Por lo tanto, aunque el perceptrón no sea una solución para todos los problemas, es el primer paso hacia la construcción de modelos más sofisticados y potentes que son fundamentales para las aplicaciones modernas de la inteligencia artificial.

6.- Bibliografía

A continuación, se presentan las fuentes de consulta que han sido fundamentales para el desarrollo y comprensión de la práctica sobre el perceptrón y las redes neuronales artificiales (ANN). Estas fuentes incluyen libros, artículos académicos y documentación técnica que brindan el marco teórico necesario para implementar y entender los conceptos que hemos utilizado en esta práctica.

6.1.- Libros

1. Russell, S., & Norvig, P. (2020). Artificial Intelligence: A Modern Approach. Pearson Education.

Este libro es uno de los textos más utilizados en el campo de la inteligencia artificial. Proporciona una introducción exhaustiva a las redes neuronales, incluyendo los perceptrones y otros modelos de aprendizaje supervisado. Su explicación detallada sobre el funcionamiento de las redes neuronales es fundamental para entender cómo se entrenan los modelos de aprendizaje automático.

2. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
   Este libro ofrece un enfoque más profundo sobre las redes neuronales, cubriendo desde los modelos más simples como el perceptrón hasta redes neuronales profundas (deep learning). Es una fuente clave para comprender los fundamentos de las redes neuronales y cómo estas estructuras evolucionan para abordar problemas más complejos.

3. Hassoun, M. H. (1995). Fundamentals of Artificial Neural Networks. MIT Press.
   En este texto, Hassoun ofrece una visión general sobre las redes neuronales artificiales, explicando los conceptos fundamentales como los perceptrones, las redes multicapa y los algoritmos de entrenamiento. Este libro es esencial para cualquier persona que desee profundizar en los aspectos técnicos del entrenamiento y funcionamiento de las redes neuronales.

6.2.- Artículos académicos y recursos web

4. Wikipedia - Perceptrón. (2024).
   https://es.wikipedia.org/wiki/Perceptr%C3%B3n

   El artículo de Wikipedia sobre el perceptrón proporciona una descripción accesible del concepto, su historia, y su funcionamiento. Aunque es una fuente introductoria, es útil para obtener un panorama general sobre el modelo y su evolución dentro de las redes neuronales.

5. PyTorch Documentation. (2024).
   https://pytorch.org/docs/stable/index.html

   La documentación oficial de PyTorch es esencial para entender las bibliotecas de aprendizaje profundo y cómo se pueden implementar redes neuronales en Python. Aunque no directamente utilizada en esta práctica específica, es útil para quienes deseen avanzar hacia redes neuronales más complejas

como las redes multicapa o las redes neuronales profundas (deep learning).

6. Scikit-learn Documentation. (2024). https://scikit-learn.org/stable/

   La documentación de Scikit-learn ofrece valiosa información sobre los algoritmos de machine learning, incluidos los relacionados con redes neuronales y el aprendizaje supervisado. Aunque se utiliza más para modelos más complejos, sirve como un excelente recurso para profundizar en el campo del aprendizaje automático.

6.3.- Recursos de código abierto y tutoriales en línea

7. Tutoriales de Perceptrón en Medium. (2024). https://medium.com/

   Una serie de tutoriales prácticos disponibles en Medium que cubren la implementación de perceptrones en Python. Estos artículos ayudan a visualizar cómo el código de un perceptrón se puede adaptar a problemas de clasificación simples y son útiles para entender el proceso de implementación paso a paso.

8. TensorFlow Documentation. (2024). https://www.tensorflow.org/learn?hl=es

   La documentación de TensorFlow contiene tutoriales y ejemplos que abarcan desde los fundamentos del aprendizaje supervisado con perceptrones hasta modelos más avanzados. Aunque TensorFlow se utiliza para redes neuronales más grandes, sus tutoriales básicos son útiles para aquellos que desean entender cómo se estructura el entrenamiento y la implementación de redes neuronales.

6.4.- Otros recursos relevantes

9. Colab Notebooks, Google. (2024). https://colab.research.google.com

   Google Colab es una plataforma excelente para ejecutar código de Python en línea, y tiene numerosos ejemplos y notebooks públicos relacionados con redes neuronales y perceptrones. A través de Colab, es fácil modificar ejemplos de código y realizar pruebas experimentales sin necesidad de una configuración local compleja.

10. Neural Networks and Deep Learning, Michael Nielsen. (2015). http://neuralnetworksanddeeplearning.com

Este es un libro gratuito en línea que ofrece una introducción clara y práctica al mundo de las redes neuronales. Abarca desde los perceptrones hasta el deep learning, proporcionando explicaciones detalladas y ejercicios prácticos que son fundamentales para comprender el funcionamiento de las redes neuronales.

6.5.- Documentación oficial de Python

11. Python Software Foundation. (2024). https://docs.python.org

La documentación oficial de Python es un recurso esencial para entender los conceptos de programación y cómo aplicar bibliotecas en Python. La implementación del perceptrón en esta práctica se realiza completamente en Python, por lo que la documentación es una referencia clave para comprender la sintaxis y las estructuras utilizadas.

Este conjunto de fuentes proporciona una base sólida de recursos teóricos y prácticos que respaldan el desarrollo y la comprensión de los conceptos utilizados en esta práctica. A través de estos materiales, se profundiza no solo en los detalles del perceptrón, sino también en las aplicaciones más avanzadas de redes neuronales.

7.- Mapa Mental

El mapa mental del perceptrón ilustra los conceptos clave de las redes neuronales artificiales, destacando los componentes del perceptrón (dendritas, sinapsis, axón), el proceso de aprendizaje supervisado, la función de activación, y su capacidad para resolver problemas lineales como la operación lógica AND.

8.- Anexos

8.1. NeuronaAnd.py

Este archivo contiene el código fuente para un perceptrón básico que resuelve la operación lógica AND utilizando aprendizaje supervisado. El código incluye la definición de la función de activación, el cálculo de la suma ponderada de entradas y pesos, y el proceso de ajuste de los pesos durante el entrenamiento. También incluye la interacción con el usuario para comprobar si el perceptrón ha aprendido correctamente la función AND.

8.2. Mapa Mental Práctica 1.7 - ANN - Perceptrón.pdf

Este archivo contiene el mapa mental de la práctica, donde se visualizan los conceptos clave relacionados con el perceptrón y las

redes neuronales artificiales (ANN). El mapa mental abarca componentes esenciales como las dendritas, sinapsis, axón, la función de activación, el proceso de aprendizaje supervisado y cómo el perceptrón resuelve problemas como la lógica AND.

8.3. Práctica 1.7 - ANN - Perceptrón.docx

Este archivo contiene la documentación completa de la práctica **1.7 - ANN - Perceptrón**, que abarca desde los conceptos teóricos sobre redes neuronales artificiales (ANN) y perceptrones, hasta el análisis detallado del código implementado en Python para resolver la operación lógica AND utilizando un perceptrón básico. La documentación incluye explicaciones sobre el funcionamiento del perceptrón, el proceso de entrenamiento, los objetivos de la práctica, los resultados obtenidos y las conclusiones.

Índice Alfabético

S

T

U

V

W