Practica 1
Perceptrón simple

Universidad de Guadalajara, Centro Universitario de Ciencias exactas e Ingenierías.

Alumno:
Cárdenas Pérez Calvin Cristopher Código: 214798706

Carrera:
Ingeniería En Computación (INCO)

Materia:
Seminario de Solución de Problemas de Inteligencia artificial II

Maestro:
Dr. Diego Oliva

Ciclo:
2024 A

Sección:
D05

Fecha de Entrega:
lunes 26 de febrero de 2024

Introducción

La inteligencia artificial ha demostrado ser una herramienta poderosa en una variedad de aplicaciones, desde el procesamiento del lenguaje natural hasta la visión por computadora. En este reporte, exploraremos la aplicación de redes neuronales artificiales, específicamente perceptrones, en la clasificación automática de datos reales utilizando el lenguaje de programación Python.

El perceptrón es uno de los modelos más simples de una neurona artificial. A pesar de su simplicidad, los perceptrones han sido fundamentales en el desarrollo de redes neuronales más complejas y han encontrado aplicaciones en una variedad de problemas de clasificación.

En este reporte, está dedicado a la aplicación de perceptrones en la clasificación de datos que representan compuertas lógicas, específicamente las compuertas lógicas XOR y OR. Estas compuertas son fundamentales en la teoría de circuitos lógicos y representan casos de estudio interesantes debido a sus propiedades de separabilidad lineal y no lineal, respectivamente.

El objetivo es comprender cómo los perceptrones abordan la clasificación de datos que representan diferentes compuertas lógicas y analizar la eficacia de los perceptrones en la separación de datos linealmente y no linealmente separables.

A través de la implementación y experimentación con perceptrones en Python, se espera obtener una mejor comprensión de su funcionamiento y su capacidad para abordar problemas de clasificación en diferentes dominios.

Desarrollo

Para empezar con el código del perceptrón simple, eran necesarias algunas librerías por lo que las importamos primero. Estas bibliotecas incluyen NumPy para operaciones numéricas, Pandas para manipulación de datos tabulares y Matplotlib para visualización de datos.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Se define una función de activación llamada step_function. Esta función aplica un umbral a la entrada y devuelve 1 si la entrada es mayor o igual a cero, y 0 en caso contrario. Esta función se utiliza para determinar la salida de la neurona en el perceptrón.

```
# Función de activación (en este caso, función escalón)
def step_function(x):
    return np.where(x>=0, 1, 0)
```

Ahora si nos enfocamos en la clase del perceptrón simple. La clase tiene métodos para inicializar el perceptrón, entrenarlo con datos de entrada y salida, y predecir la salida para nuevos datos de entrada.

```
# Clase del perceptrón
class Perceptron:
    def __init__(self, num_inputs, learning_rate=0.1, max_epochs=100):
        self.weights = np.random.rand(num_inputs + 1)  # +1 para el sesgo
        self.learning_rate = learning_rate
        self.max_epochs = max_epochs

    def train(self, X, y):
        for epoch in range(self.max_epochs):
            for i in range(X.shape[0]):
                inputs = np.insert(X[i], 0, 1)  # Insertar 1 para el sesgo
                prediction = self.predict(inputs)
                error = y[i] - prediction
                self.weights += self.learning_rate * error * inputs

    def predict(self, inputs):
        summation = np.dot(inputs, self.weights)
        return step_function(summation)
```

Se agrega una función para leer datos de entrenamiento, Esta función es llamada read_data que lee los datos de entrenamiento desde un archivo CSV. La función utiliza

la biblioteca Pandas para leer los datos del archivo CSV y devuelve las características de entrada (X) y las etiquetas de salida (y).

```
# Función para leer datos de entrenamiento desde archivos CSV
def read_data(filename):
    data = pd.read_csv(filename, header=None)
    X = data.iloc[:, :-1].values
    y = data.iloc[:, -1].values
    return X, y
```

Agregamos otra función para graficar los patrones y la línea de separación llamada plot_data_and_line que visualiza los datos de entrenamiento y la línea de separación aprendida por el perceptrón. La función utiliza Matplotlib para trazar los puntos de datos y la línea de separación.

```
# Función para graficar los patrones y la recta que los separa
def plot_data_and_line(X, y, perceptron):
    plt.scatter(X[:, 0], X[:, 1], c=y)
    plt.xlabel('Feature 1')
    plt.ylabel('Feature 2')

    # Dibujar la línea de separación
    slope = -perceptron.weights[1] / perceptron.weights[2]
    intercept = -perceptron.weights[0] / perceptron.weights[2]
    x_values = np.linspace(-2, 2, 100)
    plt.plot(x_values, slope * x_values + intercept, 'r')

    plt.show()
```

En los últimos pasos ponemos el entrenamiento y visualización del perceptrón para la compuerta lógica XOR, en la cual se leen los datos de entrenamiento y prueba para la compuerta lógica XOR desde los archivos que se encuentran en la misma carpeta que el programa con ayuda de la función read_data. Se crea un objeto de la clase Perceptron y se entrena con los datos de entrenamiento. Luego, se visualiza la línea de separación aprendida utilizando la función plot_data_and_line.

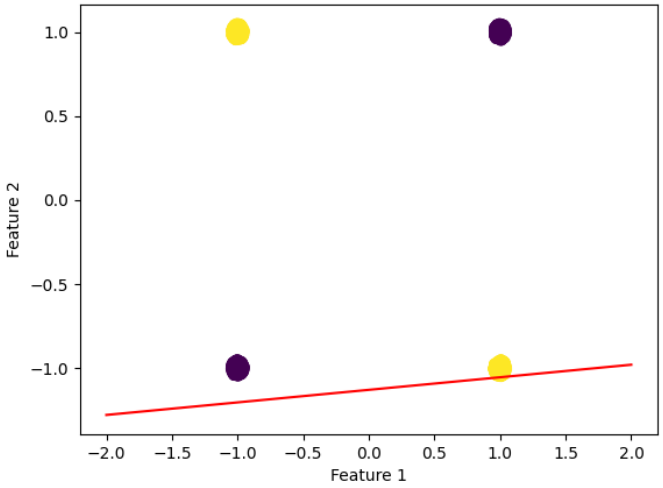
```
# Lectura de datos de entrenamiento y prueba para XOR
X_train_xor, y_train_xor = read_data('XOR_trn.csv')
X_test_xor, y_test_xor = read_data('XOR_tst.csv')

# Creación y entrenamiento del perceptrón para XOR
perceptron_xor = Perceptron(num_inputs=X_train_xor.shape[1])
perceptron_xor.train(X_train_xor, y_train_xor)

# Visualización de los datos y la línea que los separa para XOR
print("Línea de separación para la compuerta XOR:")
plot_data_and_line(X_train_xor, y_train_xor, perceptron_xor)
```

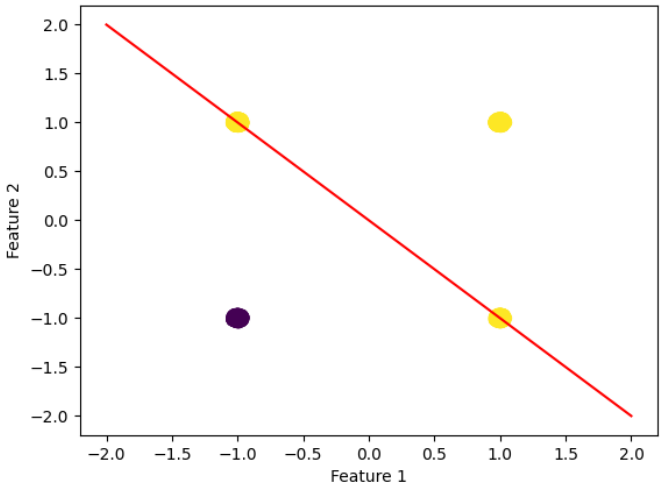
Y hacemos lo mismo para la compuerta lógica OR, leyendo los datos de entrenamiento y prueba, entrenando el perceptrón y visualizando la línea de separación aprendida.

```
# Lectura de datos de entrenamiento y prueba para OR
X_train_or, y_train_or = read_data('OR_trn.csv')
X_test_or, y_test_or = read_data('OR_tst.csv')

# Creación y entrenamiento del perceptrón para OR
perceptron_or = Perceptron(num_inputs=X_train_or.shape[1])
perceptron_or.train(X_train_or, y_train_or)

# Visualización de los datos y la línea que los separa para OR
print("Línea de separación para la compuerta OR:")
plot_data_and_line(X_train_or, y_train_or, perceptron_or)
```

Resultados

Por último tenemos la salida de las graficas que nos ha dado nuestro perceptrón simple con respecto a las compuertas lógicas XOR y OR.





Conclusión

Me gusto y se me hizo interesante y difícil la práctica del perceptrón simple, tuve que investigar y ver videos sobre el tema para entender un poco más y empezar el perceptrón para la clasificación de datos utilizando el lenguaje de Python.

A través de la implementación del perceptrón y su entrenamiento en conjuntos de datos representativos de compuertas lógicas XOR y OR, me quedo claro como estos modelos pueden separar datos linealmente. Me fue fundamental esta práctica ya que me ha proporcionado una base sólida para comprender los conceptos fundamentales detrás de los algoritmos de aprendizaje automático y su aplicación en problemas de clasificación simples.

Al finalizar con la práctica y observar los resultados obtenidos por el perceptrón, Me di cuenta de forma muy clara obvio ya que es gráficamente, cómo las líneas de separación aprendidas pueden distinguir entre diferentes clases de datos. Me fue de mucho aprendizaje esta práctica demostrándome que el perceptrón tiene la capacidad de realizar las clasificaciones en conjuntos de datos linealmente separables, no cabe duda que entre más te adentras a la inteligencia artificial, más te sorprende.