ITSRLL
INSTITUTO TECNOLÓGICO SUPERIOR DE LA REGIÓN DE LOS LLANOS

INGENIERÍA MECATRÓNICA

Programación Avanzada

Osbaldo Aragón Banderas

SEMESTRE 2024A

Unidad: 1 Actividad: 2

Nombre de actividad:

U2A2.PROGRAMA. Implementación de Perceptrón

Actividad realizada por:

JOSÉ ESTEBAN CALDERA VICTORIO

Guadalupe Victoria, Durango

Fecha de entrega:

18	02	2025
DD	MM	AA

Introducción

Se describe el desarrollo, implementación y análisis de un perceptrón simple para la clasificación de solicitudes de préstamo en una institución financiera. El modelo se entrena utilizando datos históricos de aprobaciones y rechazos, basándose en factores clave como el puntaje de crédito, el ingreso mensual, el monto del préstamo solicitado y la relación deuda/ingresos.

Descripción del Programa

El programa implementa un perceptrón simple que aprende a clasificar solicitudes de préstamo mediante un conjunto de datos normalizado. Se utiliza una función de activación escalón y un proceso iterativo de entrenamiento para ajustar los pesos y el sesgo.

Conjunto de Datos

El modelo utiliza los siguientes datos de entrada:

Puntaje de crédito: Un valor numérico que indica la solvencia del solicitante.

Ingreso mensual (en miles de dólares): Indica la capacidad económica del solicitante.

Monto del préstamo solicitado (en miles de dólares): Monto que se desea solicitar.

Relación deuda/ingresos: Ratio que indica qué porcentaje de los ingresos del solicitante está comprometido en deudas previas.

La salida esperada es un valor binario:

1: Solicitud aprobada.

0: Solicitud rechazada.

Normalización de Datos

Para mejorar la estabilidad y rendimiento del modelo, los datos se normalizan a un rango entre 0 y 1. La normalización se realiza mediante la siguiente fórmula:

Donde es el valor original, y son los valores mínimos y máximos conocidos para cada característica. Esta técnica evita que variables con valores grandes dominen el aprendizaje y asegura una mejor convergencia del modelo.

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Proceso de Entrenamiento

El perceptrón ajusta sus pesos utilizando la siguiente regla de aprendizaje:

$$w = w + \alpha \times error \times x$$

$$bias = bias + \alpha \times error$$

Donde:

- son los pesos del modelo.
- es la tasa de aprendizaje (0.1 en este caso).
- es la diferencia entre la salida esperada y la predicha.
- son las entradas de la muestra.

El entrenamiento del modelo se lleva a cabo durante un número definido de épocas, en cada una de las cuales se ajustan los pesos del perceptrón con base en el error obtenido. Durante este proceso, el perceptrón aprende patrones en los datos y mejora progresivamente su capacidad de clasificación.

Dado que el perceptrón solo puede clasificar correctamente datos que sean linealmente separables, su desempeño dependerá de qué tan bien los datos de entrada puedan ser divididos por un hiperplano en el espacio de características.

Además, al final del entrenamiento, el modelo se evalúa con una nueva muestra de datos para verificar su precisión en la clasificación.

Análisis de Resultados

Proceso de Entrenamiento

Tras cada época, los pesos y el bias del perceptrón se actualizan para reducir el error en las predicciones. Se observa que con cada iteración, los errores disminuyen progresivamente, indicando que el modelo aprende patrones en los datos de entrada.

Evaluación del Modelo

El modelo se evalúa con una nueva solicitud de préstamo con los siguientes valores:

- **Puntaje de crédito**: 800
- **Ingreso mensual**: 6.0 miles de dólares
- **Monto del préstamo**: 20 miles de dólares
- **Relación deuda/ingresos**: 0.1

Tras la normalización y el procesamiento a través del perceptrón, el modelo predice que la solicitud **fue aprobada**.

Interpretación

- **Impacto de la normalización**: Sin la normalización, los valores grandes podrían haber dominado el aprendizaje, afectando la convergencia del modelo.
- **Precisión del modelo**: El perceptrón aprende correctamente patrones simples de clasificación, pero puede tener limitaciones con datos más complejos o no linealmente separables.
- **Posibles mejoras**: Se podría mejorar el desempeño utilizando un perceptrón multicapa o aplicando algoritmos más avanzados como redes neuronales profundas.

Resultados del código

```
 Época 10:
Muestra 1: Entrada [0.83333333 0.71428571 0.71428571 0.2       ], Esperado 1, Predicción 1, Error 0
Muestra 2: Entrada [0.33333333 0.14285714 0.42857143 0.8       ], Esperado 0, Predicción 0, Error 0
Muestra 3: Entrada [0.6        0.42857143 0.14285714 0.4       ], Esperado 1, Predicción 1, Error 0
Muestra 4: Entrada [0.16666667 0.         0.02857143 1.        ], Esperado 0, Predicción 0, Error 0
Muestra 5: Entrada [1. 1. 1. 0.], Esperado 1, Predicción 1, Error 0
Pesos actualizados: [ 0.60693656  0.20499474 -0.00610378 -0.20953876], Bias actualizado: [-0.20468998]

El Préstamo: [-1.3333333333333333, 0.0, -0.32857142857142857, 1.0] = Fue Rechazado
```

El perceptrón arroja como resultado un rechazado o aceptado ya que entre mayor sea su puntaje de crédito y su ingreso más posibilidades tienen que el crédito sea aprobado si no es así, inmediatamente es rechazado.

Notebook del Código

```python
import numpy as np

# Conjunto de datos históricos de la institución financiera
x = np.array([
    [750, 5.0, 80, 0.3],  # Aprobado
    [600, 3.0, 60, 0.6],  # Rechazado
    [680, 4.0, 40, 0.4],  # Aprobado
    [550, 2.5, 32, 0.7],  # Rechazado
    [800, 6.0, 100, 0.2]   # Aprobado
])

# Salidas esperadas según el historial de decisiones
y = np.array([1, 0, 1, 0, 1])

# Normalización de los datos (Escalado entre 0 y 1)
def normalize(values, min_vals, max_vals):
    return [(v - min_v) / (max_v - min_v) for v, min_v, max_v in zip(values, min_vals, max_vals)]

# Definir valores mínimos y máximos conocidos (pueden ajustarse según el dominio)
min_vals = [500, 2.5, 30, 0.2]  # Valores mínimos de cada característica
max_vals = [800, 6.0, 100, 0.7]  # Valores máximos de cada característica

# Normalizar el conjunto de entrenamiento
x = np.array([normalize(row, min_vals, max_vals) for row in x])

# Hiperparámetros
learning_rate = 0.1
epochs = 10

# Inicialización de pesos y bias aleatorios
weights = np.random.rand(4)
bias = np.random.rand(1)

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

```python
# Entrenamiento del perceptrón con datos históricos
for epoch in range(epochs):
    print(f" Época {epoch + 1}:")
    for i in range(len(x)):
        linear_output = np.dot(x[i], weights) + bias
        predicted_output = activation_function(linear_output)
        error = y[i] - predicted_output

        # Ajuste de pesos y bias basado en el error
        weights += learning_rate * error * x[i]
        bias += learning_rate * error

        print(f"Muestra {i+1}: Entrada {x[i]}, Esperado {y[i]}, Predicción {predicted_output}, Error {error}")

    print(f"Pesos actualizados: {weights}, Bias actualizado: {bias}\n")

# Evaluación del modelo con un nuevo caso de solicitud de préstamo
x_val = np.array([100, 2.5, 7 , 0.7])  # Nueva solicitud sin normalizar
x_val = normalize(x_val, min_vals, max_vals)  # Normalización

salida = activation_function(np.dot(x_val, weights) + bias)
print(f"El Préstamo: {x_val} = {'Fue Aprobado' if salida == 1 else 'Fue Rechazado'}")
```

https://colab.research.google.com/drive/1WBcc5EBuK9q7p5wQ6hvghQdT6rhGwSjw?usp=sharing

Link para GitHub.

https://github.com/Esteban-Caldera/Programaci-n-Avanzada/tree/09ec526886a119f5ad51871186e2dff66de8b904/U2A2%20Implementaci%C3%B3n%20de%20Perceptr%C3%B3n

Conclusiones

El perceptrón implementado demuestra ser una herramienta eficaz para la clasificación binaria de solicitudes de préstamo con base en características financieras clave. A través del entrenamiento iterativo, el modelo ajusta sus pesos para mejorar su capacidad de clasificación.