ITSRLL
INSTITUTO TECNOLÓGICO SUPERIOR DE LA REGIÓN DE LOS LLANOS

Ingeniería Mecatrónica

PROGRAMACIÓN AVANZADA

Enero – Junio 2025

M.C. Osbaldo Aragón Banderas

UNIDAD: 2

Actividad número: A2

Nombre de actividad:

PROGRAMA. Implementación de Perceptrón

Actividad realizada por:

Roberto Jair Arteaga Valenzuela

Guadalupe Victoria, Durango

Fecha de entrega: 18 de febrero de 2025

Implementación de un Perceptrón para la Clasificación de Solicitudes de Préstamo

Introducción

En la actualidad, las instituciones financieras enfrentan el desafío de evaluar rápidamente la viabilidad de otorgar créditos a solicitantes. Un enfoque basado en inteligencia artificial puede automatizar este proceso y mejorar la eficiencia de la toma de decisiones. En este proyecto, se implementa un perceptrón en Python para clasificar solicitudes de préstamo como aprobadas o rechazadas, basado en cuatro factores financieros clave.

Objetivo

Desarrollar e implementar un perceptrón capaz de clasificar solicitudes de préstamo en aprobadas (1) o rechazadas (0) en función de las siguientes variables financieras:

- Puntaje de crédito: Valor entre 300 y 850.
- Ingresos mensuales: Expresado en miles de pesos.
- Monto del préstamo solicitado: Expresado en miles de pesos.
- Relación deuda/ingresos: Valor decimal (por ejemplo, 0.2, 0.5, etc.).

El perceptrón debe aprender a clasificar correctamente cada solicitud utilizando un conjunto de datos histórico proporcionado por la institución financiera.

Análisis del Código

El código implementado consta de los siguientes pasos principales:

- **Carga y preparación de datos:** Se define una matriz con datos históricos que incluye las variables financieras y la clasificación esperada.

- **Normalización de datos:** Se emplea la librería MinMaxScaler de scikit-learn para escalar las variables financieras y mejorar el rendimiento del perceptrón.
- Inicialización de parámetros: Se establecen los pesos y el sesgo con valores aleatorios y una tasa de aprendizaje de 0.1.

Entrenamiento del perceptrón:

- Se ejecutan 20 épocas de entrenamiento.
- Para cada entrada, se calcula la salida del perceptrón utilizando una función de activación binaria.
- Se ajustan los pesos y el sesgo en función del error.

Prueba con nuevos datos: Se ingresa un nuevo conjunto de valores y se determina si la solicitud es aprobada o rechazada con base en el modelo entrenado.

Análisis de Resultados

El entrenamiento del perceptrón logra un ajuste progresivo de los pesos y el sesgo, lo que permite mejorar la clasificación de solicitudes. En las pruebas realizadas:

- El modelo fue capaz de aprender patrones a partir de los datos históricos.
- Las decisiones finales del perceptrón se alinearon con las expectativas de la institución financiera.
- Se evidenció que la normalización de datos contribuyó a una mejor convergencia del modelo.
- No obstante, el perceptrón es un modelo lineal, por lo que podría no ser adecuado para problemas con relaciones no lineales entre las variables. Para mejorar la precisión, podría considerarse el uso de redes neuronales multicapa.

Analizando los resultados de la ejecución del perceptrón con datos altos:

1. Normalización de datos:
    - Los valores de entrada han sido escalados entre 0 y 1 utilizando MinMaxScaler, lo cual mejora la estabilidad del entrenamiento.
2. Entrenamiento:
    - Se ejecutaron múltiples épocas (en la imagen se muestran al menos dos).
    - En cada muestra, se evalúa la entrada con los pesos y el bias para generar una predicción.
    - Si la predicción es incorrecta, se ajustan los pesos y el bias.
3. Pesos y Bias finales:
    - Al final de la segunda época, los pesos y el bias convergen a ciertos valores:

Pesos: [0.80213638 -0.05929192 0.33829392 -0.13786468], Bias: -0.409880891929039165

    - Esto indica que el modelo ha aprendido una regla de decisión para clasificar las solicitudes.
4. Prueba con datos nuevos:
    - Se evalúa una nueva solicitud con valores [700, 45, 190, 0.35] después de normalizarla.
    - El resultado es "Solicitud aprobada", lo que indica que el perceptrón considera que esta solicitud cumple con los criterios aprendidos.

```
# Prueba con nuevos datos
nuevos_datos = np.array([[700, 45, 190, 0.35]])
nuevos_datos = scaler.transform(nuevos_datos)
resultado = activation_function(np.dot(nuevos_datos, weights) + bias)
print("Solicitud aprobada" if resultado == 1 else "Solicitud rechazada")
```

```
Muestra 1: Entrada [1. 1. 1. 0.], Esperado 1.0, Predicción 1, Error 0.0
Muestra 2: Entrada [0.25       0.33333333 0.5        0.5       ], Esperado 0.0, Predicción 0, Error 0.0
Muestra 3: Entrada [0.85       0.66666667 0.8        0.25      ], Esperado 1.0, Predicción 1, Error 0.0
Muestra 4: Entrada [0.15       0.16666667 0.2        0.75      ], Esperado 0.0, Predicción 0, Error 0.0
Muestra 5: Entrada [0.65  0.5   0.7   0.375], Esperado 1.0, Predicción 1, Error 0.0
Muestra 6: Entrada [0. 0. 0. 1.], Esperado 0.0, Predicción 0, Error 0.0
Pesos: [ 0.80213638 -0.05929192  0.33829392 -0.13786468], Bias: -0.40980891929039165

Época: 20
Muestra 1: Entrada [1. 1. 1. 0.], Esperado 1.0, Predicción 1, Error 0.0
Muestra 2: Entrada [0.25       0.33333333 0.5        0.5       ], Esperado 0.0, Predicción 0, Error 0.0
Muestra 3: Entrada [0.85       0.66666667 0.8        0.25      ], Esperado 1.0, Predicción 1, Error 0.0
Muestra 4: Entrada [0.15       0.16666667 0.2        0.75      ], Esperado 0.0, Predicción 0, Error 0.0
Muestra 5: Entrada [0.65  0.5   0.7   0.375], Esperado 1.0, Predicción 1, Error 0.0
Muestra 6: Entrada [0. 0. 0. 1.], Esperado 0.0, Predicción 0, Error 0.0
Pesos: [ 0.80213638 -0.05929192  0.33829392 -0.13786468], Bias: -0.40980891929039165

Solicitud aprobada
```

Figura 1 Solicitud aprobada

Análisis de los resultados obtenidos con datos menores

1. Normalización de datos:
    - La solicitud de prueba tiene los valores [300, 15, 100, 0.25].
    - Se normaliza usando MinMaxScaler, lo que ajusta los valores a una escala entre 0 y 1, mejorando la estabilidad del entrenamiento.
2. Entrenamiento del perceptrón:
    - Se han ejecutado 20 épocas.
    - Durante cada iteración, el modelo ajusta los pesos y el bias en función del error de predicción.
3. Pesos y Bias finales:
    - Al final del entrenamiento, los valores son:

Pesos: [0.48921496 0.70177269 0.02606486 0.30620455], Bias: -0.6169792806592639

    - Estos valores definen la regla de decisión que el perceptrón ha aprendido.
4. Prueba con nuevos datos:
    - La nueva solicitud tiene valores más bajos en puntaje de crédito, ingresos y monto del préstamo solicitado.
    - El resultado final es "Solicitud rechazada", lo que indica que, según los patrones aprendidos por el perceptrón, la solicitud no cumple con los criterios para ser aprobada.

```
# Prueba con nuevos datos
nuevos_datos = np.array([[300, 15, 100, 0.25]])
nuevos_datos = scaler.transform(nuevos_datos)
resultado = activation_function(np.dot(nuevos_datos, weights) + bias)
print("Solicitud aprobada" if resultado == 1 else "Solicitud rechazada")
```

```
Muestra 1: Entrada [1. 1. 1. 0.], Esperado 1.0, Predicción 1, Error 0.0
Muestra 2: Entrada [0.25       0.33333333 0.5        0.5       ], Esperado 0.0, Predicción 0, Error 0.0
Muestra 3: Entrada [0.85       0.66666667 0.8        0.25      ], Esperado 1.0, Predicción 1, Error 0.0
Muestra 4: Entrada [0.15       0.16666667 0.2        0.75      ], Esperado 0.0, Predicción 0, Error 0.0
Muestra 5: Entrada [0.65  0.5   0.7   0.375], Esperado 1.0, Predicción 1, Error 0.0
Muestra 6: Entrada [0. 0. 0. 1.], Esperado 0.0, Predicción 0, Error 0.0
Pesos: [0.48921496 0.70177629 0.02606486 0.30620455], Bias: -0.6169792806592639

Época: 20
Muestra 1: Entrada [1. 1. 1. 0.], Esperado 1.0, Predicción 1, Error 0.0
Muestra 2: Entrada [0.25       0.33333333 0.5        0.5       ], Esperado 0.0, Predicción 0, Error 0.0
Muestra 3: Entrada [0.85       0.66666667 0.8        0.25      ], Esperado 1.0, Predicción 1, Error 0.0
Muestra 4: Entrada [0.15       0.16666667 0.2        0.75      ], Esperado 0.0, Predicción 0, Error 0.0
Muestra 5: Entrada [0.65  0.5   0.7   0.375], Esperado 1.0, Predicción 1, Error 0.0
Muestra 6: Entrada [0. 0. 0. 1.], Esperado 0.0, Predicción 0, Error 0.0
Pesos: [0.48921496 0.70177629 0.02606486 0.30620455], Bias: -0.6169792806592639

Solicitud rechazada
```

Figura 2 Solicitud rechazada

Código desarrollado

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler

# Datos históricos (Ejemplo de tabla adjunta)
datos = np.array([
    [750, 50, 200, 0.3, 1],  # Aprobado
    [600, 30, 150, 0.5, 0],  # Rechazado
    [720, 40, 180, 0.4, 1],  # Aprobado
    [580, 25, 120, 0.6, 0],  # Rechazado
    [680, 35, 170, 0.45, 1], # Aprobado
    [550, 20, 100, 0.7, 0]   # Rechazado
])

# Separar características y etiquetas
X = datos[:, :-1]
y = datos[:, -1]

# Normalizar datos para mejorar el aprendizaje
scaler = MinMaxScaler()
X = scaler.fit_transform(X)

# Inicialización de parámetros
learning_rate = 0.1
epochs = 20
weights = np.random.rand(X.shape[1])
bias = np.random.rand()

def activation_function(x):
    return 1 if x >= 0 else 0

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

        # Actualización de pesos y bias
        weights += learning_rate * error * X[i]
        bias += learning_rate * error

        print(f"Muestra {i+1}: Entrada {X[i]}, Esperado {y[i]}, Predicción {prediction}, Error {error}")
    print(f"Pesos: {weights}, Bias: {bias}\n")

# Prueba con nuevos datos
nuevos_datos = np.array([[700, 45, 190, 0.35]])
nuevos_datos = scaler.transform(nuevos_datos)
resultado = activation_function(np.dot(nuevos_datos, weights) + bias)
print("Solicitud aprobada" if resultado == 1 else "Solicitud rechazada")
```

Conclusiones

La implementación del perceptrón demuestra que es posible automatizar la clasificación de solicitudes de préstamo con un modelo de aprendizaje supervisado. Se destaca la importancia de la normalización de datos y el ajuste adecuado de los hiperparámetros para mejorar la efectividad del modelo. Sin embargo, para escenarios más complejos, sería recomendable explorar modelos más avanzados como redes neuronales profundas.

Link del Github: https://github.com/Jair-Artreaga/Perceptron-Automatizacion-Solicitudes.git