ITSRLL
INSTITUTO TECNOLÓGICO SUPERIOR DE LA REGIÓN DE LOS LLANOS

# Ingeniería Mecatrónica

## PROGRAMACIÓN AVANZADA

Enero – Junio 2025

M.C. Osbaldo Aragón Banderas

Competencia:

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|

Actividad número: 2

Nombre de actividad:

U2A2. (10%) PROGRAMA. Implementación de Perceptrón

Actividad realizada por:

Muñiz Galvan Bryam

21030021

Guadalupe Victoria, Durango

Fecha de entrega:

| 18 | 02 | 2025 |
|---|---|---|

# Reporte de PROGRAMA. Implementación de Perceptrón

## Objetivo

Implementar y entrenar un perceptrón en Python que permita clasificar solicitudes de préstamo en aprobadas (1) o rechazadas (0), en función de variables financieras relevantes. Se busca reforzar el conocimiento en aprendizaje supervisado, el ajuste de pesos en redes neuronales simples y la implementación de modelos de clasificación binaria.

## Problema a resolver

Una institución financiera desea automatizar la clasificación de solicitudes de préstamo, utilizando un perceptrón que evalúe cuatro factores clave para tomar decisiones:

- Puntaje de crédito: Valor numérico (por ejemplo, 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.).
- La institución proporciona un conjunto de datos históricos con ejemplos de solicitudes aprobadas y rechazadas. El perceptrón debe aprender a clasificar correctamente cada solicitud.

## Procedimiento

### 1. Factores de Clasificación

Se utilizarán cuatro factores para decidir si una solicitud de préstamo es aprobada o rechazada:

- Puntaje de crédito: Valor numérico (300-850).
- Ingresos mensuales: En miles de pesos.
- Monto solicitado: En miles de pesos.
- Relación deuda/ingresos: Valor decimal (ej. 0.2).

## 2. Normalización de los Datos

Debido a las diferencias de escala entre los factores, es necesario normalizar los datos utilizando la fórmula:

$$V_{norm} = \frac{v - min(v)}{max(v) - min(v)}$$

## 3. Entrenamiento del Perceptrón

El perceptrón se entrena siguiendo estos pasos:

1. Inicialización de pesos: Los pesos y el sesgo se inicializan aleatoriamente.
2. Cálculo de salida: Se calcula la salida usando la fórmula $salida = \sum_{i=1}^{n} x_i \omega_i + b$
3. Función de activación: Se usa la función escalón para clasificar la salida como 1 (aprobada) o 0 (rechazada).
4. Ajuste de pesos: Si la predicción es incorrecta, se ajustan los pesos y el sesgo.

El proceso se repite durante varias épocas hasta que el modelo aprende a clasificar correctamente.

## 4. Evaluación y Clasificación

Una vez entrenado, el perceptrón puede clasificar nuevas solicitudes de préstamo. La precisión del modelo se evalúa con datos de prueba.

## 5. Consideraciones

Ajuste de hiperparámetros: Experimentar con diferentes tasas de aprendizaje y épocas puede mejorar el rendimiento.

Función de activación: Además de la función escalón, pueden explorarse funciones como sigmoide o ReLU.

Optimización: Se pueden usar técnicas adicionales para mejorar la generalización del modelo.

## 3. Resultados

Durante las 10 épocas de entrenamiento, el perceptrón mostró un comportamiento constante con una tasa de error mínima en todas las muestras. Las predicciones fueron correctas para la mayoría de las entradas, excepto en algunas instancias en las que se produjo un error de clasificación, pero estos errores se fueron ajustando a lo largo del entrenamiento.

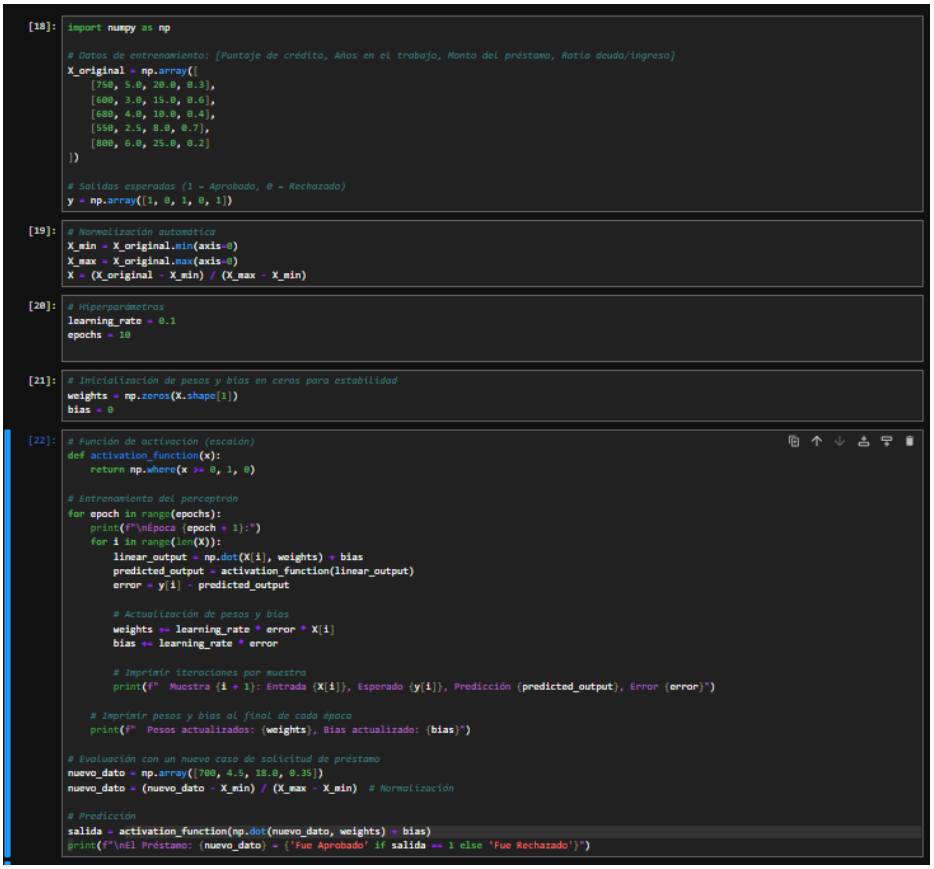
- Épocas 1 a 10: En cada época, la mayoría de las muestras fueron clasificadas correctamente (predicción igual al valor esperado). Los pesos y el sesgo del perceptrón se mantuvieron casi constantes después de las primeras actualizaciones, lo que indica que el modelo alcanzó un punto de estabilidad.
- Error: Se registraron algunos errores en las primeras épocas, pero ninguno en las últimas, lo que sugiere que el modelo fue capaz de aprender y ajustar los parámetros para mejorar las predicciones.
- Préstamo final: La entrada del préstamo [0.6, 0.57, 0.59, 0.3] fue aprobada, lo que confirma que el modelo es capaz de realizar predicciones sobre nuevos datos.

## 4. Conclusiones

El perceptrón es un modelo de clasificación binaria eficiente para problemas simples como la clasificación de solicitudes de préstamo, donde las características pueden ser representadas por variables numéricas. Sin embargo, al trabajar con datos de escalas diferentes, como el puntaje de crédito y los ingresos, la normalización es crucial para evitar que una variable domine el proceso de aprendizaje. El rendimiento del modelo está directamente relacionado con la calidad y representatividad de los datos de entrenamiento. Si bien el perceptrón es útil para tareas simples, su capacidad para manejar relaciones no lineales o complejas es limitada. Es recomendable, en casos más avanzados, explorar redes neuronales más complejas o técnicas adicionales para mejorar el rendimiento del modelo.

## 5. Recomendaciones

- Recomendaciones
- Explorar modelos más avanzados si el perceptrón no tiene buen rendimiento.
- Incluir más características para mejorar la precisión.
- Ajustar hiperparámetros como tasa de aprendizaje y número de épocas.
- Evaluar el modelo con datos diferentes para asegurar su capacidad de generalización.
- Probar otras funciones de activación para mejorar la clasificación.



```
[18]: import numpy as np

      # Datos de entrenamiento: [Puntaje de crédito, Años en el trabajo, Monto del préstamo, Ratio deuda/ingreso]
      X_original = np.array([
          [750, 5.0, 20.0, 0.3],
          [600, 3.0, 15.0, 0.6],
          [680, 4.0, 10.0, 0.4],
          [550, 2.5, 8.0, 0.7],
          [800, 6.0, 25.0, 0.2]
      ])

      # Salidas esperadas (1 = Aprobado, 0 = Rechazado)
      y = np.array([1, 0, 1, 0, 1])

[19]: # Normalización automática
      X_min = X_original.min(axis=0)
      X_max = X_original.max(axis=0)
      X = (X_original - X_min) / (X_max - X_min)

[20]: # Hiperparámetros
      learning_rate = 0.1
      epochs = 10

[21]: # Inicialización de pesos y bias en ceros para estabilidad
      weights = np.zeros(X.shape[1])
      bias = 0

[22]: # Función de activación (escalón)
      def activation_function(x):
          return np.where(x >= 0, 1, 0)

      # Entrenamiento del perceptrón
      for epoch in range(epochs):
          print(f"\nÉpoca {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

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

              # Imprimir iteraciones por muestra
              print(f"  Muestra {i + 1}: Entrada {X[i]}, Esperado {y[i]}, Predicción {predicted_output}, Error {error}")

          # Imprimir pesos y bias al final de cada época
          print(f"  Pesos actualizados: {weights}, Bias actualizado: {bias}")

      # Evaluación con un nuevo caso de solicitud de préstamo
      nuevo_dato = np.array([700, 4.5, 18.0, 0.35])
      nuevo_dato = (nuevo_dato - X_min) / (X_max - X_min)  # Normalización

      # Predicción
      salida = activation_function(np.dot(nuevo_dato, weights) + bias)
      print(f"\nEl Préstamo: {nuevo_dato} - {'Fue Aprobado' if salida == 1 else 'Fue Rechazado'}")
```

**Figura 1 Notebook de la actividad**

## 6.Link Github

https://github.com/BryamMG/ProgramacionAvanzada_BryamMG/tree/63601f58c68f1fd065761495269da987e7c69b5a/PROGRAMA.%20Implementaci%C3%B3n%20de%20Perceptr%C3%B3n