

Documentación: Perceptrón aplicado a comportamiento en páginas web

1. Introducción

Se implementa un ejemplo de un modelo de red neuronal simple conocido como Perceptrón, aplicado a un caso relacionado con Ingeniería de Sistemas: predecir si un usuario abandona una página web basándose en su comportamiento (tiempo de navegación, interacción y desplazamiento).

2. Objetivo

Diseñar e implementar un modelo Perceptrón que sea capaz de clasificar a los usuarios entre aquellos que permanecen en la página web y aquellos que la abandonan, utilizando un conjunto de datos simulados.

3. Variables del modelo

- Tiempo en página (segundos)
- Scroll al 80% o más (1: sí, 0: no)
- Clic en algún botón (1: sí, 0: no)

Salida esperada:

- 1: Usuario se quedó
- 0: Usuario abandonó

4. Código Fuente

```
import numpy as np

def step_function(z):
    return 1 if z >= 0 else 0

class Perceptron:
    def __init__(self, input_size, learning_rate=0.01):
        self.weights = np.zeros(input_size + 1)
        self.learning_rate = learning_rate

    def predict(self, x):
```

```

    z = np.dot(x, self.weights[1:]) + self.weights[0]
    return step_function(z)

def train(self, X, y, epochs=50):
    for epoch in range(epochs):
        for inputs, label in zip(X, y):
            prediction = self.predict(inputs)
            error = label - prediction
            self.weights[1:] += self.learning_rate * error * inputs
            self.weights[0] += self.learning_rate * error
        print(f"Época {epoch + 1}: Pesos = {self.weights}")

# Datos de entrenamiento
X = np.array([
    [5, 0, 0], # Poco tiempo, sin scroll, sin clic = abandono
    [20, 1, 1], # Más tiempo, scroll y clic = se quedó
    [10, 0, 1], # Algo de tiempo, hizo clic = se quedó
    [2, 0, 0], # Muy poco tiempo = abandono
    [25, 1, 0], # Buen tiempo y scroll = se quedó
    [3, 0, 0] # Muy poco tiempo = abandono
])

y = np.array([0, 1, 1, 0, 1, 0]) # Etiquetas (0 = se fue, 1 = se quedó)

# Entrenar perceptrón
perceptron = Perceptron(input_size=3)
perceptron.train(X, y, epochs=20)

# Pruebas
test_samples = np.array([
    [15, 1, 1], # Usuario muy activo
    [2, 0, 0], # Usuario pasivo
])

print("\nResultados de prueba:")
for inputs in test_samples:
    result = perceptron.predict(inputs)
    print(f"Entrada: {inputs} → {'Se quedó' if result == 1 else 'Abandonó'}")

```

5. Conclusiones

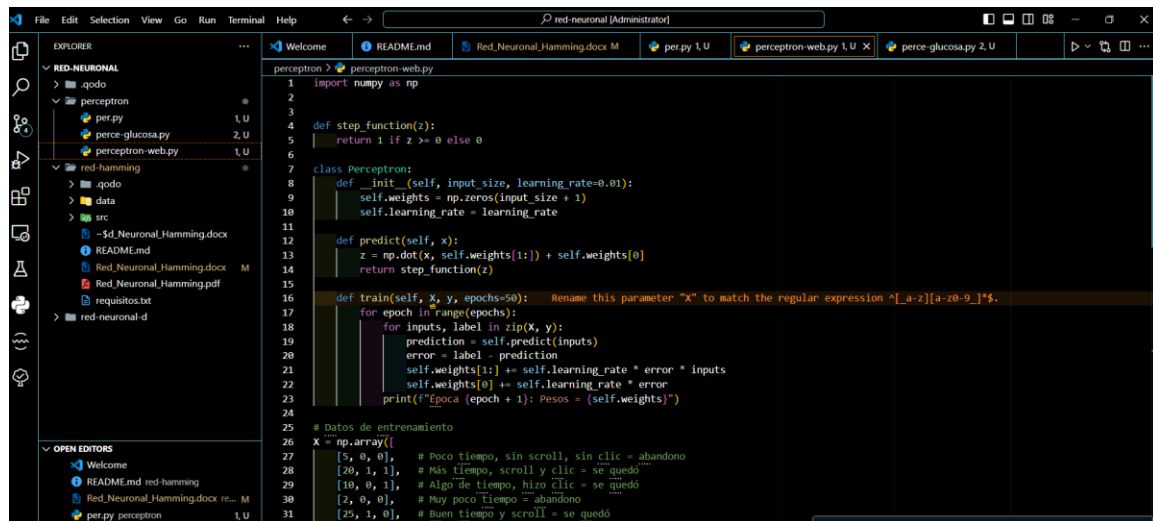
El modelo Perceptrón fue capaz de aprender patrones simples de comportamiento en usuarios de páginas web. Aunque es un modelo sencillo, demuestra cómo técnicas básicas de redes neuronales pueden ser aplicadas en proyectos reales de Ingeniería de Sistemas, como análisis de comportamiento, optimización de experiencia de usuario y sistemas de recomendación.

6. Aplicaciones en Ingeniería de Sistemas

- Análisis de comportamiento de usuarios.
- Optimización de interfaces web.
- Sistemas de recomendación personalizados.
- Predicción de tasa de abandono.
- Desarrollo de dashboards de inteligencia web.

7. Anexos

IMPLEMENTACION



```
1 import numpy as np
2
3 def step_function(z):
4     return 1 if z >= 0 else 0
5
6
7 class Perceptron:
8     def __init__(self, input_size, learning_rate=0.01):
9         self.weights = np.zeros(input_size + 1)
10        self.learning_rate = learning_rate
11
12    def predict(self, x):
13        z = np.dot(x, self.weights[1:]) + self.weights[0]
14        return step_function(z)
15
16    def train(self, X, y, epochs=50):
17        # Rename this parameter "X" to match the regular expression "[a-z][a-z0-9_]*$".
18        for epoch in range(epochs):
19            for inputs, label in zip(X, y):
20                prediction = self.predict(inputs)
21                error = label - prediction
22                self.weights[1:] += self.learning_rate * error * inputs
23                self.weights[0] += self.learning_rate * error
24            print(f"Epoch {epoch + 1}: Pesos = {self.weights}")
25
26 # Datos de entrenamiento
27 X = np.array([
28     [5, 0, 0], # Poco tiempo, sin scroll, sin clic = abandono
29     [20, 1, 1], # Más tiempo, scroll y clic = se quedó
30     [10, 0, 1], # Algo de tiempo, hizo clic = se quedó
31     [2, 0, 0], # Muy poco tiempo = abandono
32     [25, 1, 0], # Buen tiempo y scroll = se quedó
```

RESULTADOS

```
perceptron-web.py
1 import numpy as np
2
3
4 def step_function(z):
5     return 1 if z >= 0 else 0
6
7 class Perceptron:
8     def __init__(self, input_size, learning_rate=0.01):
9         self.weights = np.zeros(input_size + 1)
10        self.learning_rate = learning_rate
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python Python

Época 7: Pesos = [-0.16 0.15 0.04 0.04]
Época 8: Pesos = [-0.19 0.05 0.04 0.04]
Época 9: Pesos = [-0.21 0.15 0.05 0.05]
Época 10: Pesos = [-0.22 0.07 0.05 0.05]
Época 11: Pesos = [-0.24 0.02 0.05 0.05]
Época 12: Pesos = [-0.24 0.02 0.05 0.05]
Época 13: Pesos = [-0.24 0.02 0.05 0.05]
Época 14: Pesos = [-0.24 0.02 0.05 0.05]
Época 15: Pesos = [-0.24 0.02 0.05 0.05]
Época 16: Pesos = [-0.24 0.02 0.05 0.05]
Época 17: Pesos = [-0.24 0.02 0.05 0.05]
Época 18: Pesos = [-0.24 0.02 0.05 0.05]
Época 19: Pesos = [-0.24 0.02 0.05 0.05]
Época 20: Pesos = [-0.24 0.02 0.05 0.05]

Resultados de prueba:
Entrada: [1 1 1] → Se quedó
Entrada: [2 0 0] → Abandonó

(venv) D:\ciclo-7-2025-I\ANALISIS MULTIVARIADO\UNIDAD 3\red-neuronal>