

Reporte: A03 Red Neuronal de una capa

Rafael Aldo Hernández Luna

February 14, 2026

Red Neuronal de una sola capa

Dr. Carlos Villaseñor

Paso 1. Corre la siguiente casilla para importar la paquetería necesaria.

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

Paso 2. Corre el siguiente bloque con las funciones de activación programadas en la práctica anterior

```
# La función de activación lineal se usa en problemas de regresión
def linear(z, derivative=False):
    a = z
    if derivative:
        da = np.ones(z.shape, dtype=float)
        return a, da
    return a

# La función de activación logística se usa en
# problemas de clasificación multi-etiquetas
def logistic(z, derivative=False):
    a = 1/(1 + np.exp(-z))
    if derivative:
        da = np.ones(z.shape, dtype=float)
        return a, da
    return a

# La función de activación Softmax se usa en
# problemas de clasificación multiclase con un
# solo ganador
def softmax(z, derivative=False):
```

Paso 3. Programa la función de activación Softmax

```
# La función de activación Softmax se usa en
# problemas de clasificación multiclase con un
# solo ganador
def softmax(z, derivative=False):
```

```

e = np.exp(z - np.max(z, axis=0))
a = e / np.sum(e, axis=0)
if derivative:
    da = np.ones(z.shape, dtype=float)
    return a, da
return a

```

Paso 4. Programa una clase que represente una red neuronal de una capa

```

class OLN:
    """One-Layer Network"""

    def __init__(self, n_inputs, n_outputs,
                 activation_funtion=linear):
        self.w = - 1 + 2 * np.random.rand(n_outputs, n_inputs)
        self.b = - 1 + 2 * np.random.rand(n_outputs, 1)
        self.f = activation_funtion

    def predict(self, X):
        Z = self.w @ X + self.b
        return self.f(Z)

    def fit(self, X, Y, epochs=1000, lr=0.1):
        # Columnas de X
        p = X.shape[1]

        for _ in range(epochs):
            # Propagation -----
            Z = self.w @ X + self.b
            Yest, dY = self.f(Z, derivative=True)

            # Training -----

            # Calculate local gradient
            lg = (Y - Yest) * dY

            # Update parameters
            self.w += (lr/p) * lg @ X.T
            self.b += (lr/p) * np.sum(lg, axis=1).reshape(-1,1)

```

Primer experimento: Clasificación multi-etiqueta

Paso 5. Carga el archivo 'DataSet_A03.csv', contiene entradas de dos dimensiones y salidas deseadas de 4 dimensiones.

```
!wget 'https://raw.githubusercontent.com/Dr-Carlos-Villasenor/Clase_Aprendizaje_Profundo/re
```

```
--2026-02-13 12:50:10-- https://raw.githubusercontent.com/Dr-Carlos-Villasenor/Clase_Aprend
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.110.133, 185.199.
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.110.133|:443...
HTTP request sent, awaiting response... 200 OK
Length: 3864 (3.8K) [text/plain]
Saving to: 'Dataset_A03.csv.1'
```

```
Dataset_A03.csv.1 100%[=====>] 3.77K --.-KB/s in 0s
```

```
2026-02-13 12:50:10 (36.5 MB/s) - 'Dataset_A03.csv.1' saved [3864/3864]
```

```
df = pd.read_csv('Dataset_A03.csv')
```

Paso 6. Crea la matriz X de entrada y la matriz Y de salidas deseadas.

```
# Escribe en el siguiente apartado la matriz X y Y
X = np.asanyarray(df.iloc[:,0:2]).T
Y = np.asanyarray(df.iloc[:,2:]).T
```

Paso 7. Corre la siguiente función que te ayudará a dibujar el experimento

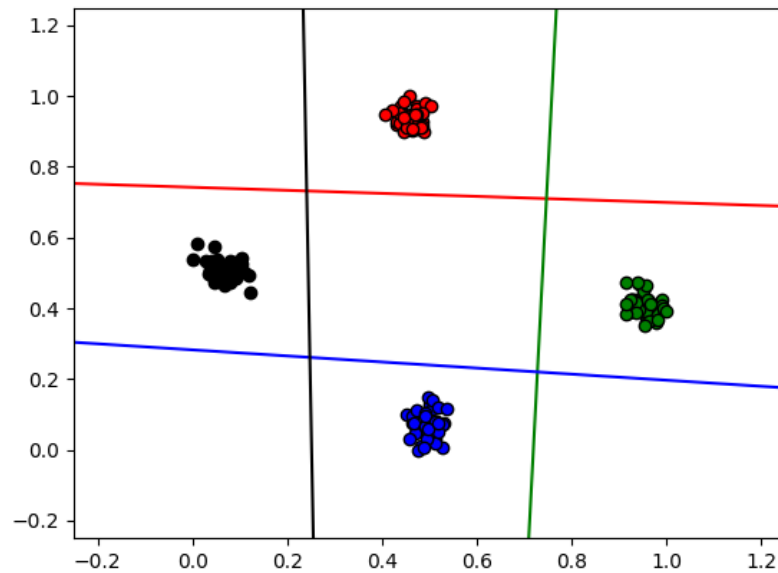
```
def plot_data(X, Y, net):
    dot_c = ('red', 'green', 'blue', 'black')
    lin_c = ('r-', 'g-', 'b-', 'k-')
    for i in range(X.shape[1]):
        c = np.argmax(Y[:,i])
        plt.scatter(X[0,i], X[1,i], color=dot_c[c], edgecolors='k')

    for i in range(4):
        w1, w2, b = net.w[i,0], net.w[i,1], net.b[i]
        plt.plot([-0.25, 1.25], [(-1/w2)*(w1*(-0.25)+b), (-1/w2)*(w1*(1.25)+b)], lin_c[i])

#plt.axis('equal')
plt.xlim([-0.25, 1.25])
plt.ylim([-0.25, 1.25])
```

Paso 8. Crea y entrena tu red neuronal con los datos que trabajaste y dibuja el resultado con la función anterior.

```
net = OLN(2, 4, logistic)
net.fit(X, Y, epochs=1000, lr=1)
plot_data(X, Y, net)
plt.savefig("plot_1.png")
```



Segundo experimento: Clasificación con un solo ganador

Paso 9. Corre el siguiente ejemplo para generar un conjunto de datos de Clasificación con un solo ganador

```
# Generación de Conjunto de datos para clasificación

# Límites
minx = -5
maxx = 5

# Número de clases y puntos por clase
classes = 8
p_c = 20
X = np.zeros((2, classes * p_c))
Y = np.zeros((classes, classes * p_c))

for i in range(classes):
    seed = minx + (maxx - minx) * np.random.rand(2,1)
    X[:, i*p_c:(i+1)*p_c] = seed + 0.15 * np.random.randn(2, p_c)
```

```
Y[i, i*p_c:(i+1)*p_c] = np.ones((1, p_c))
```

Paso 10. Entrena una red neuronal para aprender el conjunto de datos anterior, guarda en la variable Ypred la predicción de todos los datos.

```
# Instancia una red neuronal con el numero de entradas, salidas
# y función de activación correctas
net = OLN(2, classes, softmax)
```

```
# Entrena la red neuronal
net.fit(X, Y, epochs=1000, lr=1)
```

```
# Guarda las predicciones de la red de todos los datos en X
Ypred = net.predict(X)
```

Paso 11. Corre el siguiente código para graficar tus resultados

```
# Colores para dibujar las clases
```

```
cm = ['b','g','r','c','m','y', 'k', 'w']
```

```
# Gráfico con los datos originales
ax1=plt.subplot(1, 2, 1)
y_c =np.argmax(Y, axis=0)
for i in range(X.shape[1]):
    ax1.scatter(X[0,i], X[1,i], c=cm[y_c[i]], edgecolors='k')
ax1.axis([-5.5,5.5,-5.5,5.5])
ax1.set_title('Problema Original')
ax1.grid()
```

```
# Gráfico de las predicciones de la red
ax2=plt.subplot(1, 2, 2)
y_c =np.argmax(Ypred, axis=0)
for i in range(X.shape[1]):
    ax2.scatter(X[0,i], X[1,i], c=cm[y_c[i]], edgecolors='k')
ax2.axis([-5.5,5.5,-5.5,5.5])
ax2.set_title('Predicción de la red')
ax2.grid()
```

```
plt.savefig("l03_02_RES_figure.png")
plt.show()
```

