**Trabajo Práctico Perceptron**

Valentín Faraz

Ing Computación

5to año

Entrenar un perceptrón para que se comporte como una compuerta and, or y xor

import random

from math import e

def entrenar\_perceptron(tabla,lr):

w0= round(random.uniform(-1,1),4)

w1= round(random.uniform(-1,1),4)

w2= round(random.uniform(-1,1),4)

i=0

print(w0,w1,w2)

while True:

i=i+1

for combinacion in tabla:

print(combinacion)

solucion\_deseada=int(combinacion[2])

e0=1

e1=int(combinacion[0])

e2=int(combinacion[1])

x= (e1\*w1) + (e2\*w2) + (e0\*w0)

solucion\_real = round(1/(1+e\*\*(-x)),4)

error=solucion\_deseada-solucion\_real

s=solucion\_real\*(1-solucion\_real)\*error

deltaw0=lr\*e0\*s

w0=w0+deltaw0

deltaw1=lr\*e1\*s

w1=w1+deltaw1

deltaw2=lr\*e2\*s

w2=w2+deltaw2

print('Vuelta numero',i)

print('Solucion Real:',solucion\_real)

print('Solucion Deseada:',solucion\_deseada)

if error < 0.01:

break

if \_\_name\_\_ == "\_\_main\_\_":

tabla\_or=['000','011','101','111']

tabla\_and=['000','010','100','111']

tabla\_xor=['000','011','101','110']

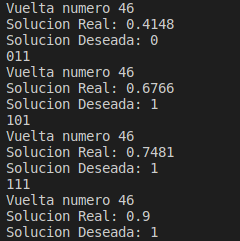
lr=0.1

entrenar\_perceptron(tabla\_or,lr)

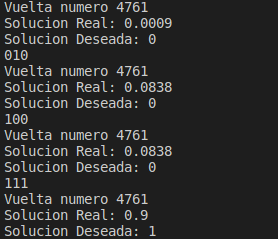
entrenar\_perceptron(tabla\_and,lr)

entrenar\_perceptron(tabla\_xor,lr)

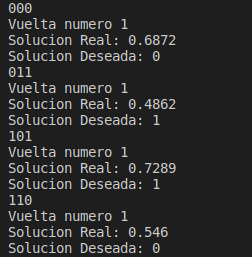
COMPUERTA OR CON ERROR MENOR AL 10% EN LA ITERACIÓN



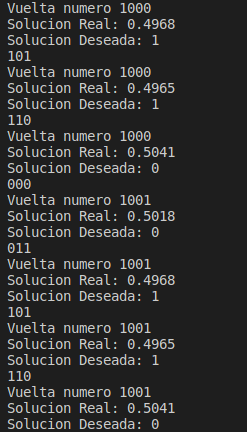
COMPUERTA AND CON UN ERROR MENOR AL 10% EN LA ITERACIÓN 4761



COMPUERTA XOR NO SE COMPORTA COMO LAS ANTERIORES Y NO SE PUEDE CALCULAR



COMPUERTA XOR CON 1000 ITERACIONES



Observamos que nunca llega al valor deseado, se aproxima a 0,5

**Nuevo software con gráficos.**

**import random**

**from math import e**

**import numpy as np**

**import matplotlib.pyplot as plt**

**def entrenar\_perceptron(tabla,lr):**

**#pessos iniciales ramdon**

**#w0= round(random.uniform(-1,1),4)**

**#w1= round(random.uniform(-1,1),4)**

**#w2= round(random.uniform(-1,1),4)**

**w0= 0.9**

**w1= 0.66**

**w2= -0.2**

**lista\_w0=[]**

**lista\_w1=[]**

**lista\_w2=[]**

**iteraciones=[]**

**lista\_e=[]**

**i=0**

**print(w0,w1,w2)**

**while True:**

**iteraciones.append(i)**

**i=i+1**

**lista\_w0.append(w0)**

**lista\_w1.append(w1)**

**lista\_w2.append(w2)**

**for combinacion in tabla:**

**print("Fila de la tabla de la verdad:",combinacion)**

**solucion\_deseada=int(combinacion[2])**

**#entradas**

**e0=1**

**e1=int(combinacion[0])**

**e2=int(combinacion[1])**

**#sumatoria de cada peso por su entrada**

**x= (e1\*w1) + (e2\*w2) + (e0\*w0)**

**print(x)**

**solucion\_real = round(1/(1+e\*\*(-x)),4)**

**error=solucion\_deseada-solucion\_real**

**#delta**

**s=solucion\_real\*(1-solucion\_real)\*error**

**deltaw0=lr\*e0\*s**

**w0=w0+deltaw0**

**deltaw1=lr\*e1\*s**

**w1=w1+deltaw1**

**deltaw2=lr\*e2\*s**

**w2=w2+deltaw2**

**print('Vuelta numero',i)**

**print("pesos:",w0,w1,w2)**

**print("Delta:",s)**

**print('Solucion Real:',solucion\_real)**

**print('Solucion Deseada:',solucion\_deseada)**

**print('Error:',error)**

**print('-----------')**

**lista\_e.append(error)**

**#if i==1000:**

**if error < 0.1:**

**#print(lista\_w0)**

**#print(iteraciones)**

**plt.plot (lista\_w0)**

**plt.plot (lista\_w1)**

**plt.plot (lista\_w2)**

**plt.ylabel ('W0-W1-W2')**

**plt.xlabel ('Iteraciones')**

**#plt.plot (w1,iteraciones)**

**#plt.plot (w2,iteraciones)**

**plt.show()**

**plt.plot (lista\_e)**

**plt.ylabel ('Error')**

**plt.xlabel ('Iteraciones')**

**plt.show()**

**break**

**if \_\_name\_\_ == "\_\_main\_\_":**

**tabla\_or=['000','011','101','111']**

**tabla\_and=['000','010','100','111']**

**tabla\_xor=['000','011','101','110']**

**lr=0.1**

**entrenar\_perceptron(tabla\_or,lr)**

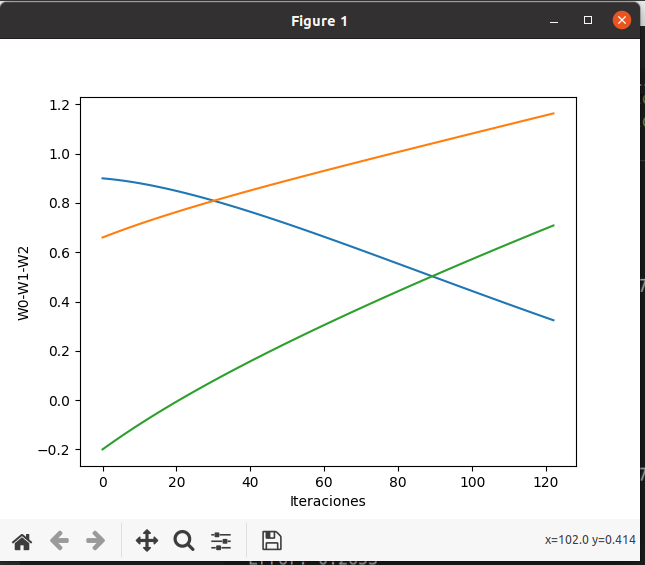
**entrenar\_perceptron(tabla\_and,lr)**

**entrenar\_perceptron(tabla\_xor,lr)**

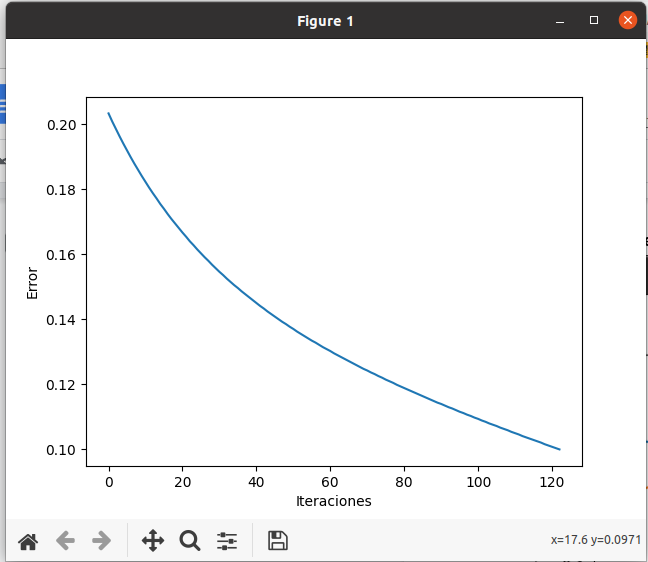
Fijamos los pesos en valores determinados para comparar resultados entre todos.

**Entrenamiento para OR**

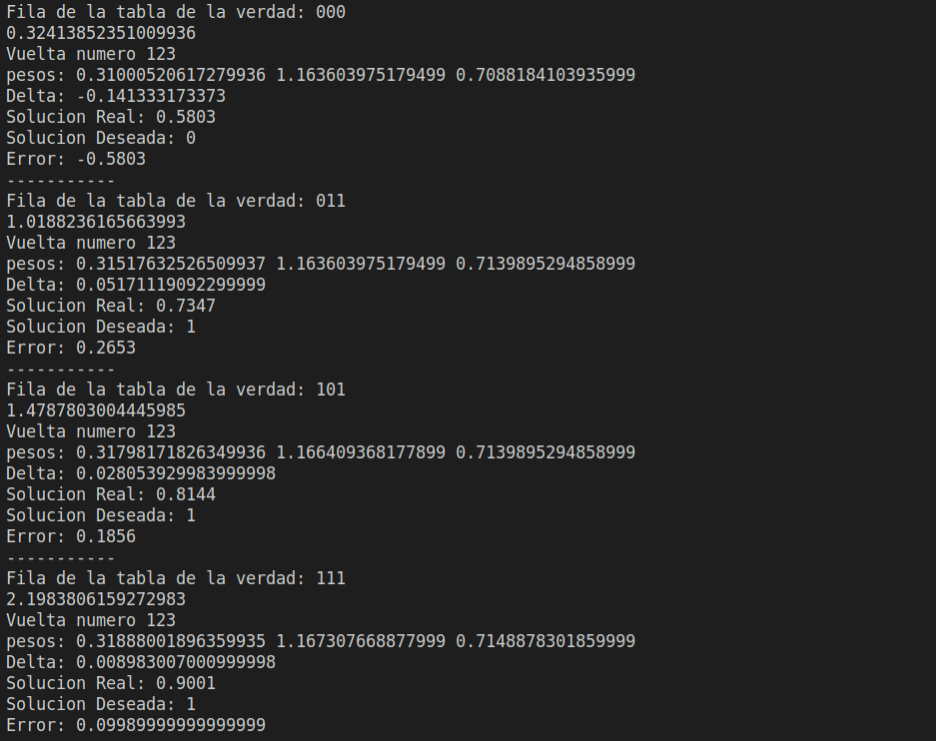
**Los pesos tienden a un valor estable**



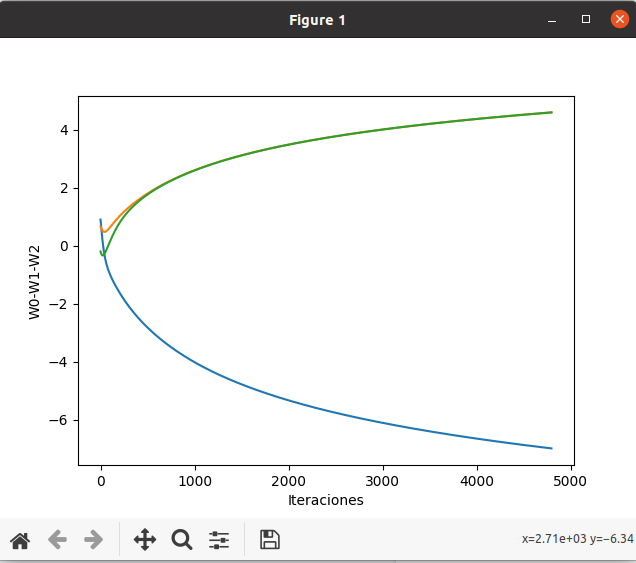
**El error tiende a cero**

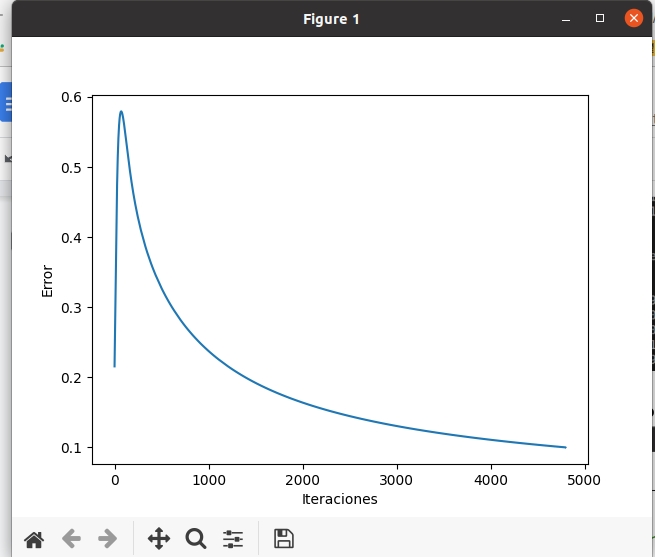


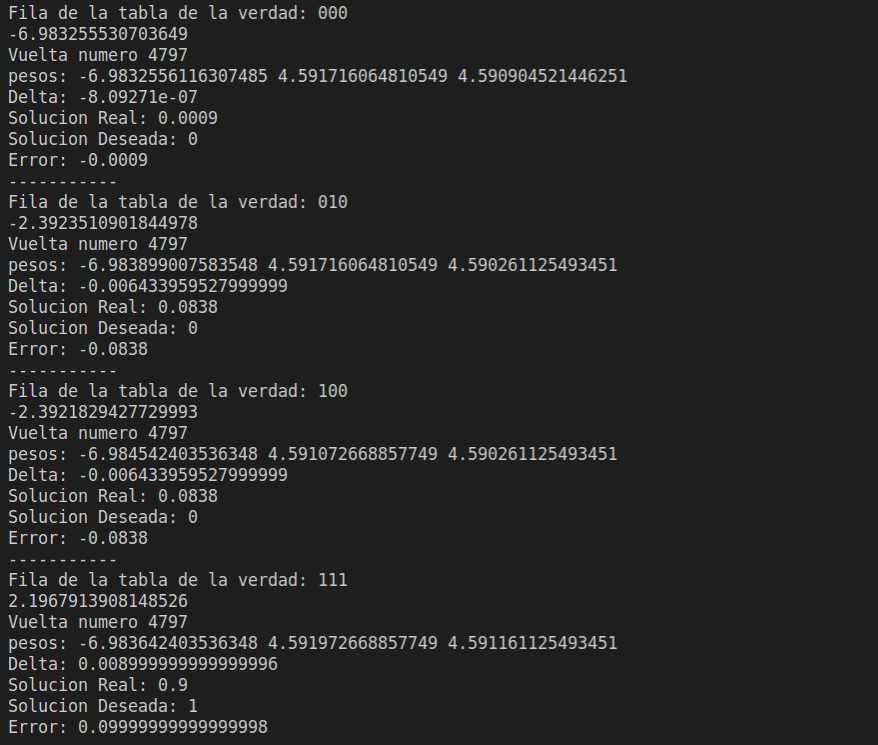
**Resultados**



**Entrenamiento para AND**







**Entrenamiento para XOR**

