

Centro Universitario de Ciencias Exactas e Ingenierías

Licenciatura en Ingeniería en Computación

Materia: Seminario de Solución de Problemas de Inteligencia Artificial II. Clave: I7041.

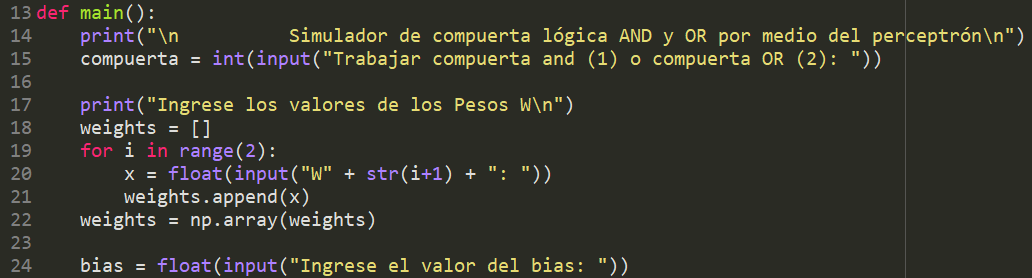
Profesor: Valdés López Julio Esteban

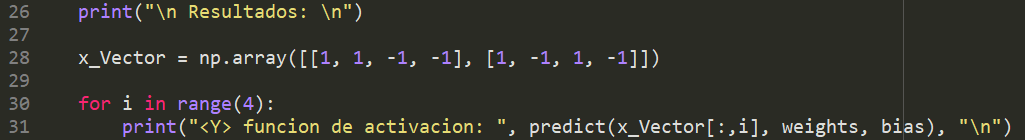
Estudiante: Silva Moya José Alejandro. Código: 213546894.

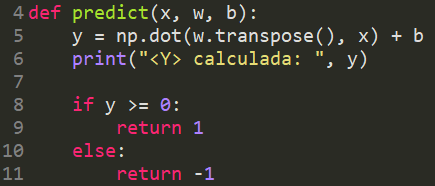
**Práctica I: Compuertas lógicas AND y OR por medio de un perceptrón no entrenado.**

**Problema a resolver:** En la presente práctica programaremos un perceptrón que sea capaz de procesar y graficar los resultados de las compuertas lógicas AND y OR sin entrenamiento y con datos W y b completamente aleatorios elegidos por el usuario mediante consola.

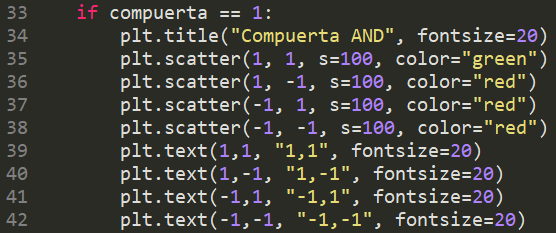
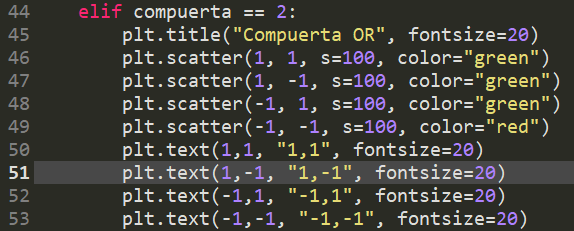
**Desarrollo**

Lo primero que realizamos es la toma de datos por parte del usuario, preguntando si trabajaremos con la compuerta AND u OR, y pidiendo el valor de los pesos sinápticos y del bias.

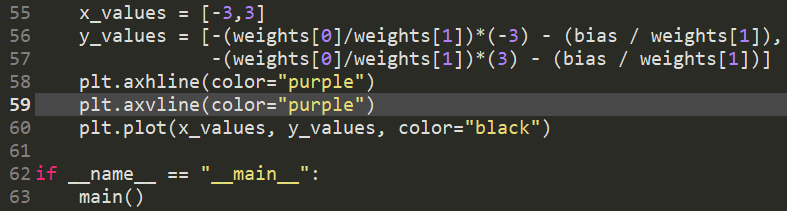
Ahora procederemos a calcular nuestros resultados, realizando una predicción de valores combinando las entradas X, junto con nuestros datos previamente capturados. La función de predicción es la siguiente:



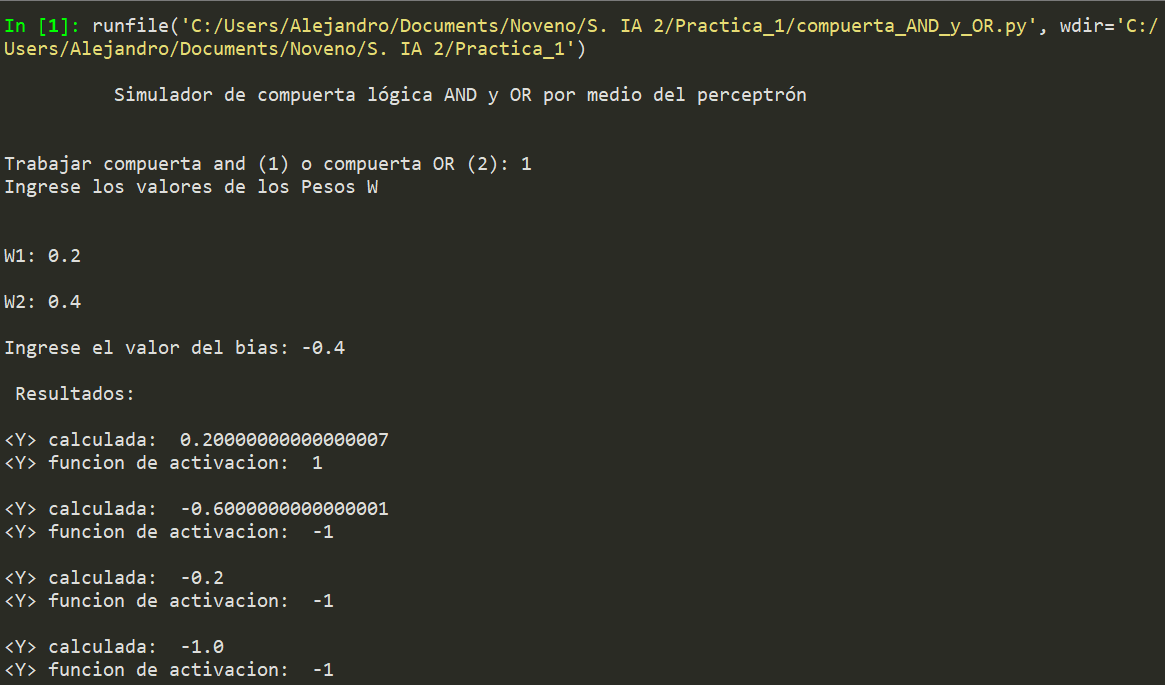
Lo que realizamos en nuestra función de predicción es calcular los valores de “*y*” por medio de la ecuación del perceptrón, y corregir los valores por medio de nuestra función de activación signo, en donde delimitamos la salida binaria a 1 o -1 (que comúnmente también es interpretada como 1 y 0).

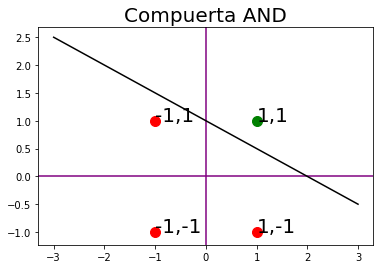


Posteriormente, antes de graficar la línea del perceptrón, dependiendo de la compuerta que el usuario haya elegido procesar, imprimiremos con diferentes colores los puntos correspondientes a la clasificación respectiva, para visualizar mejor cuando la clasificación se realice exitosamente o no.

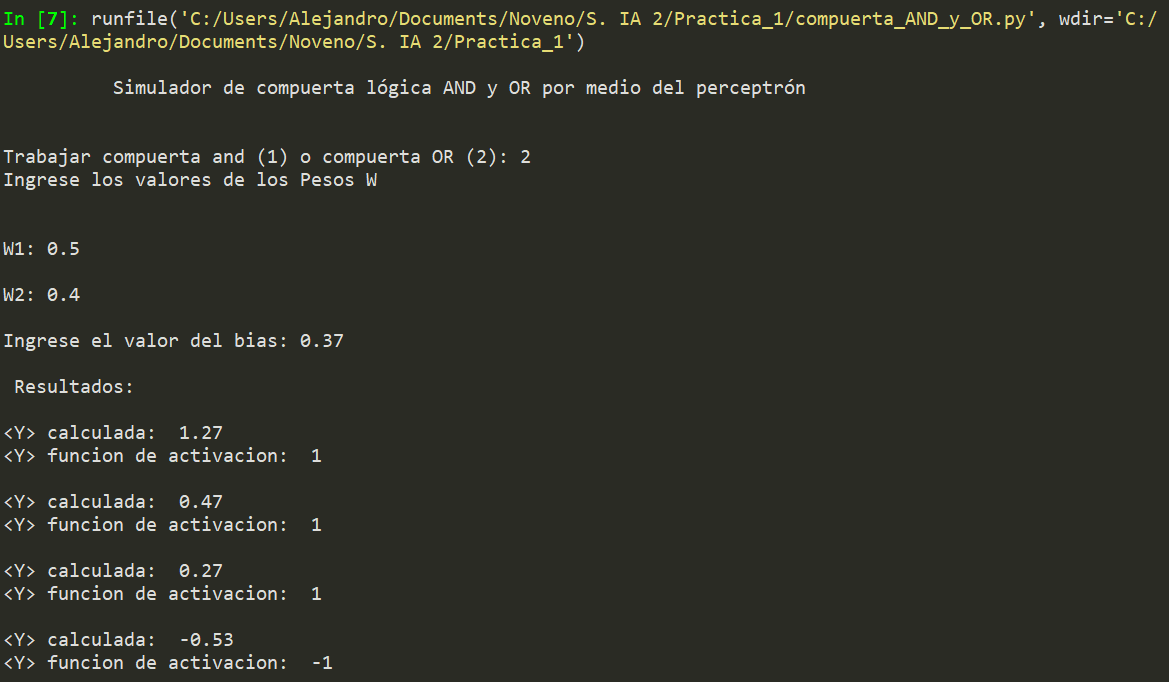
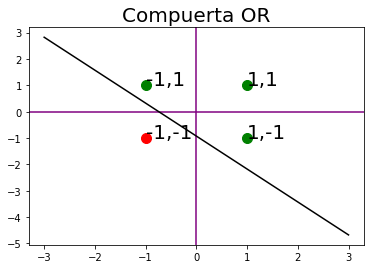
Finalmente, delimitamos un margen imparcial de -3 hasta 3 para nuestro plano cartesiano, y mediante el despeje de nuestra fórmula del perceptrón logramos reemplazar los valores correctos para graficar la línea de la neurona, y poder observar los resultados.

**Resultados**

Con la compuerta AND:



Con la compuerta OR:



**Conclusión**

Con la realización de esta práctica podemos observar y entender de una mejor manera cómo es que el perceptrón funciona como un clasificador lineal, que por sí mismo únicamente logra trabajar en un espacio de *R2.* Mediante la previa implementación del mismo, logramos que, si una función de entrenamiento, logre representar y graficar de manera exitosa los valores de las compuertas lógicas AND y OR mientras se le otorguen los valores adecuados.

Con esto, se espera como una proyección a futuro poder evolucionar el modelo a uno que sea capaz de aprender, para que así se le pueda prever con valores completamente aleatorios, y por sí mismo logre generar las modificaciones y correcciones necesarias para converger en un número finito de iteraciones.

**Código**

import numpy as np

import matplotlib.pyplot as plt

def predict(x, w, b):

y = np.dot(w.transpose(), x) + b

print("<Y> calculada: ", y)

if y >= 0:

return 1

else:

return -1

def main():

print("\n Simulador de compuerta lógica AND y OR por medio del perceptrón\n")

compuerta = int(input("Trabajar compuerta and (1) o compuerta OR (2): "))

print("Ingrese los valores de los Pesos W\n")

weights = []

for i in range(2):

x = float(input("W" + str(i+1) + ": "))

weights.append(x)

weights = np.array(weights)

bias = float(input("Ingrese el valor del bias: "))

print("\n Resultados: \n")

x\_Vector = np.array([[1, 1, -1, -1], [1, -1, 1, -1]])

for i in range(4):

print("<Y> funcion de activacion: ", predict(x\_Vector[:,i], weights, bias), "\n")

if compuerta == 1:

plt.title("Compuerta AND", fontsize=20)

plt.scatter(1, 1, s=100, color="green")

plt.scatter(1, -1, s=100, color="red")

plt.scatter(-1, 1, s=100, color="red")

plt.scatter(-1, -1, s=100, color="red")

plt.text(1,1, "1,1", fontsize=20)

plt.text(1,-1, "1,-1", fontsize=20)

plt.text(-1,1, "-1,1", fontsize=20)

plt.text(-1,-1, "-1,-1", fontsize=20)

elif compuerta == 2:

plt.title("Compuerta OR", fontsize=20)

plt.scatter(1, 1, s=100, color="green")

plt.scatter(1, -1, s=100, color="green")

plt.scatter(-1, 1, s=100, color="green")

plt.scatter(-1, -1, s=100, color="red")

plt.text(1,1, "1,1", fontsize=20)

plt.text(1,-1, "1,-1", fontsize=20)

plt.text(-1,1, "-1,1", fontsize=20)

plt.text(-1,-1, "-1,-1", fontsize=20)

x\_values = [-3,3]

y\_values = [-(weights[0]/weights[1])\*(-3) - (bias / weights[1]),

-(weights[0]/weights[1])\*(3) - (bias / weights[1])]

plt.axhline(color="purple")

plt.axvline(color="purple")

plt.plot(x\_values, y\_values, color="black")

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

main()

**Link al repositorio**

* <https://github.com/TheGenesisX/S_IA_2/tree/master/Practica_1>