



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.

Tarea II: Resolución de la compuerta lógica OR por medio de prueba y error basado en el funcionamiento de un Perceptrón.



Instrucciones: Buscar a prueba y error los pesos W y umbral B de un perceptrón que se comporte como la compuerta OR. Mostrar la comprobación de los resultados.

Desarrollo

Lo primero que realizamos fue el dataset; los vectores de entradas X, y el vector de valores correspondientes a una compuerta lógica OR.

```
11 x_Vector = np.array([[1, 1, -1, -1], [1, -1, 1, -1]])  
12 compuerta_OR = np.array([1, 1, 1, -1])
```

Con el propósito de ser visualmente más explícitos, el código anterior correspondería al siguiente dataset:

X ₁	X ₂	OR
1	1	1
1	-1	1
-1	1	1
-1	-1	-1

Resulta importante denotar que en esta implementación el -1 es el equivalente al 0 en otros usos comunes.

Ahora es necesario delimitar los valores del vector de pesos W, y el del sesgo B. Por convención estos valores son inicializados de manera aleatoria entre 0 y 1; sin embargo, como aquí pretendemos hacerlo a mano y a prueba y error, hemos decidido mantener todos los valores intermedios, es decir, en 0.5.

```
14 weights = np.array([0.5, 0.5])  
15 bias = 0.5
```

Finalmente hemos programado una función predicción, que nos arrojará los valores resultantes de procesar las entradas mediante nuestra función de propagación:

$$y = \varphi(w^T x + b)$$

En donde:

- ϕ es nuestra función de activación.
- w^T es nuestro vector de pesos transpuesto: [0.5, 0.5]
- x es la matriz de entradas: [[1, 1, -1, -1], [1, -1, 1, -1]]
- b es nuestro bias o sesgo: 0.5

Nuestra función de activación, en este caso función signo, funciona de la siguiente manera:

$\phi(z) =$	1 si $z \geq 0$
	-1 si $z < 0$

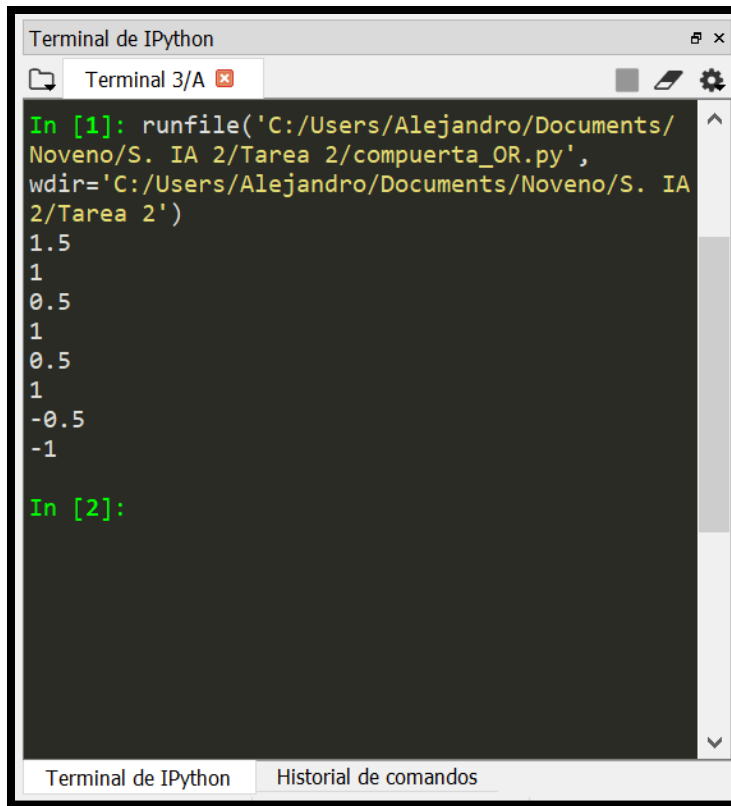
```
3 def predict(x, w, b):
4     y = np.dot(w.transpose(), x) + b
5     print(y)
6     if y >= 0:
7         return 1
8     else:
9         return -1
```

Esta función recibe entradas X , nuestro vector de pesos W y nuestro escalar B bias. Posteriormente calcula la propagación de la información, la imprime, y nos da la corrección de nuestra función de activación: si el valor resultado es mayor o igual a cero, obtenemos un 1; de lo contrario obtenemos un -1.

Finalmente le indicamos al programa que realice esto 4 veces, para poder obtener las predicciones de nuestras 4 entradas posibles a la compuerta lógica.

```
17 for i in range(4):
18     print(predict(x_Vector[:, i], weights, bias))
```

Le indicamos que en cada iteración tome todos los valores de una entrada de la matriz x (en nuestro caso 2 – alguna combinación de 1 y -1 –), el vector de pesos y el bias, para que trabaje con ellos, y en la función mostrada en el punto anterior, nos indique los resultados de nuestra aproximación a prueba y error. Los resultados finales los podemos observar a continuación.



```
Terminal de IPython
Terminal 3/A

In [1]: runfile('C:/Users/Alejandro/Documents/
Noveno/S. IA 2/Tarea 2/compuerta_OR.py',
wdir='C:/Users/Alejandro/Documents/Noveno/S. IA
2/Tarea 2')
1.5
1
0.5
1
0.5
1
-0.5
-1

In [2]:
```

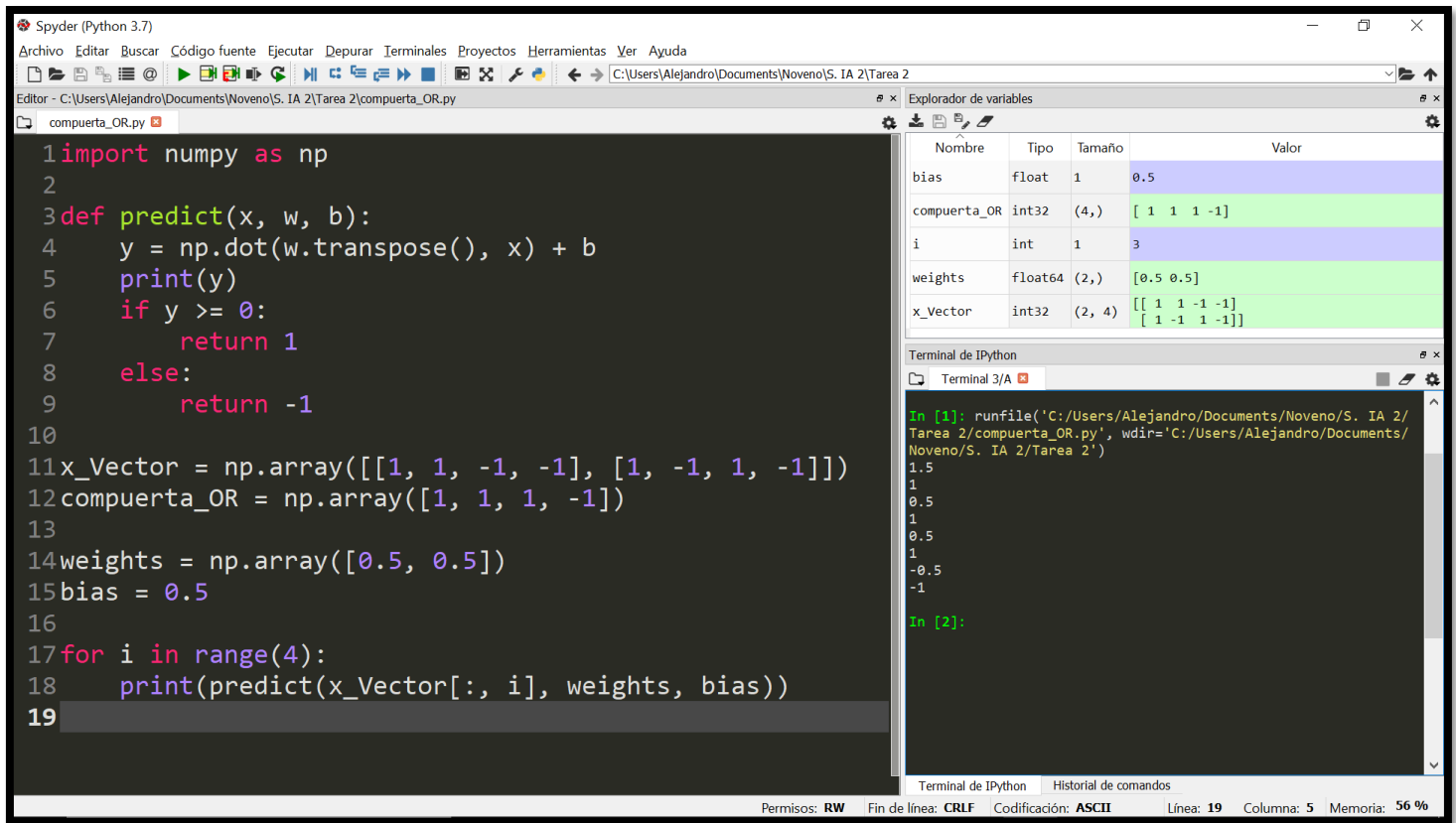
Como podemos observar, los resultados son bastante favorables, ya que arroja respuestas correctas.

- En la primera combinación de entradas (1, 1), el resultado arroja 1.5, lo que se transforma en 1 gracias a nuestra función de activación, y es correcto a la compuerta lógica OR.
- En la segunda combinación (1, -1), el resultado arroja 0.5, lo que se transforma en 1 debido a nuestra función de activación, y es correcto a la compuerta OR.
- En la tercera combinación (-1, 1) ocurre lo mismo que en la segunda, y el resultado se mantiene correcto a la

compuerta OR.

- Finalmente, en la cuarta combinación (-1, -1) el resultado es -0.5, lo cual se convierte en -1 de acuerdo a nuestra función signo, y resulta correcto a la compuerta, ya que es el único escenario en el cuál debería arrojar ese resultado.

El código completo, junto con el resultado del intento, es el siguiente:



The screenshot shows the Spyder Python IDE interface. The main editor displays a Python script named `compuerta_OR.py`. The script defines a `predict` function and uses NumPy arrays to represent the input vector, weights, and bias. The `predict` function calculates the dot product of the input vector and weights, adds the bias, and returns 1 if the result is non-negative, otherwise -1.

```
1 import numpy as np
2
3 def predict(x, w, b):
4     y = np.dot(w.transpose(), x) + b
5     print(y)
6     if y >= 0:
7         return 1
8     else:
9         return -1
10
11 x_Vector = np.array([[1, 1, -1, -1], [1, -1, 1, -1]])
12 compuerta_OR = np.array([1, 1, 1, -1])
13
14 weights = np.array([0.5, 0.5])
15 bias = 0.5
16
17 for i in range(4):
18     print(predict(x_Vector[:, i], weights, bias))
19
```

The right sidebar shows the 'Explorador de variables' (Variable Explorer) with the following data:

Nombre	Tipo	Tamaño	Valor
bias	float	1	0.5
compuerta_OR	int32	(4,)	[1 1 1 -1]
i	int	1	3
weights	float64	(2,)	[0.5 0.5]
x_Vector	int32	(2, 4)	[[1 1 -1 -1] [1 -1 1 -1]]

The bottom panel shows the IPython terminal with the following output:

```
In [1]: runfile('C:/Users/Alejandro/Documents/Noveno/S. IA 2/
Tarea 2/compuerta_OR.py', wdir='C:/Users/Alejandro/Documents/
Noveno/S. IA 2/Tarea 2')
1.5
1
0.5
1
0.5
1
-0.5
-1

In [2]:
```

The status bar at the bottom indicates: Permisos: RW, Fin de línea: CRLF, Codificación: ASCII, Línea: 19, Columna: 5, Memoria: 56 %.

En conclusión, podemos observar que fue sencillo lograr llegar a una solución con el primer intento, y eso es debido a que realmente en un perceptrón normal, el área de decisión que toma para resolver un problema puede ser bastante amplia, por lo que diferentes respuestas siguen resolviendo el problema de manera exitosa. Así pues, quiere decir que existe una posibilidad de que podamos modificar los valores de los pesos y del Bias quizás en ± 0.15 y quizás aun así encontrar que sea una respuesta correcta.