INTELIGENCIA ARTIFICIAL

TRABAJO PRÁCTICO n°2

PERCEPTRÓN

Alumna: M. Valentina Scalco

Ingeniería en Informática

2022

Consignas

1- Programar el perceptrón simple para que se comporte como una compuerta OR. Informar en cuántas iteraciones el perceptrón se comporta como una compuerta OR con un error menor al 10%.

2- Idem al punto anterior, para que se comporte como compuerta AND.

3- Configurar al perceptrón para que se comporte como una compuerta XOR. Lo logra?

Resolución

En el mismo código se resolverá la compuerta AND y OR. A la hora de ejecutar el código, se dará a elegir cual queremos realizar.

```
def eleccion(self):
    self.resp = 0
    while self.resp != 1 and self.resp != 2 and self.resp != 3:
        self.resp = int(input('Ingrese 1 para compuerta OR o 2 para compuerta AND: \n'))
        if self.resp == 1:
            self.compuerta_OR()
        elif self.resp == 2:
            self.compuerta_AND()
        else:
            print('Opcion no valida. Vuelva a intentarlo.')
```

TERMINAL

```
PS C:\Users\Valentina\Documents\FACULTAD\CUARTO AÑO\INTELIGENC
Ingrese 1 para compuerta OR o 2 para compuerta AND:
```

1- Perceptrón comportado como compuerta OR

Lo primero que se hizo fue ingresar los pesos de w0, w1 y w2. Luego, los valores de e1, e2 y la salida deseada según la compuerta OR.

```
def initialize_game(self):
    self.peso_e1 = 0.66
    self.peso_e2 = -0.2
    self.peso_w0 = 0.9
    self.valor_w0 = 1

def compuerta_OR(self):
    self.valor_e1 = [0, 0, 1, 1]
    self.valor_e2 = [0, 1, 0, 1]
    self.salida_d = [0, 1, 1, 1]
    self.resolucion()
```

Para comenzar a tratar de comportar un perceptrón como una compuerta OR, buscaremos el valor de x. La cual se conforma por una suma entre w1 por el valor que necesitemos de la tabla or (en la primera iteración, sería 0), w2 por el valor que necesitemos de la tabla or (en la primera iteración, sería 0) y w0 por el valor de e1 (siempre es 1).

```
def sumatoria(self, i):
    x = (self.peso_e1 * self.valor_e1[i]) + (self.peso_e2 * self.valor_e2[i]) + (self.peso_w0 * self.valor_w0)
    return x
```

A continuación, debemos encontrar “y” o también llamada salida real.

```
salida_r[i] = 1 / (1 + e**(-x))
```

Una vez que tenemos x e y, buscaremos el delta absoluto(error), conformado por la salida deseada menos la salida real. Y guardaremos este error para poder graficarlo más tarde.

```
delta_absoluto[i] = self.salida_d[i] - salida_r[i]
if i == 0:
    error0.append(delta_absoluto[i])
if i == 1:
    error1.append(delta_absoluto[i])
if i == 2:
    error2.append(delta_absoluto[i])
if i == 3:
    error3.append(delta_absoluto[i])
    counts += 1
```

Luego de realizar todas las ecuaciones necesarias, obtendremos los nuevos pesos corregidos. Los cuales guardaremos en tres listas diferentes para poder graficarlos. El código se ejecutará hasta que los deltas absolutos se encuentren entre 0.01 y -0.1.

```
delta_relativo = salida_r[i] * (1 - salida_r[i]) * delta_absoluto[i]
delta_w0 = lr * self.valor_w0 * delta_relativo
delta_peso_e1 = lr * self.valor_e1[i] * delta_relativo
delta_peso_e2 = lr * self.valor_e2[i] * delta_relativo
self.peso_w0 = self.peso_w0 + delta_w0
self.peso_e1 = self.peso_e1 + delta_peso_e1
self.peso_e2 = self.peso_e2 + delta_peso_e2
peso0.append(self.peso_w0)
peso1.append(self.peso_e1)
peso2.append(self.peso_e2)
if delta_absoluto[0] <= 0.01 and delta_absoluto[0] >= -0.01 and delta
```

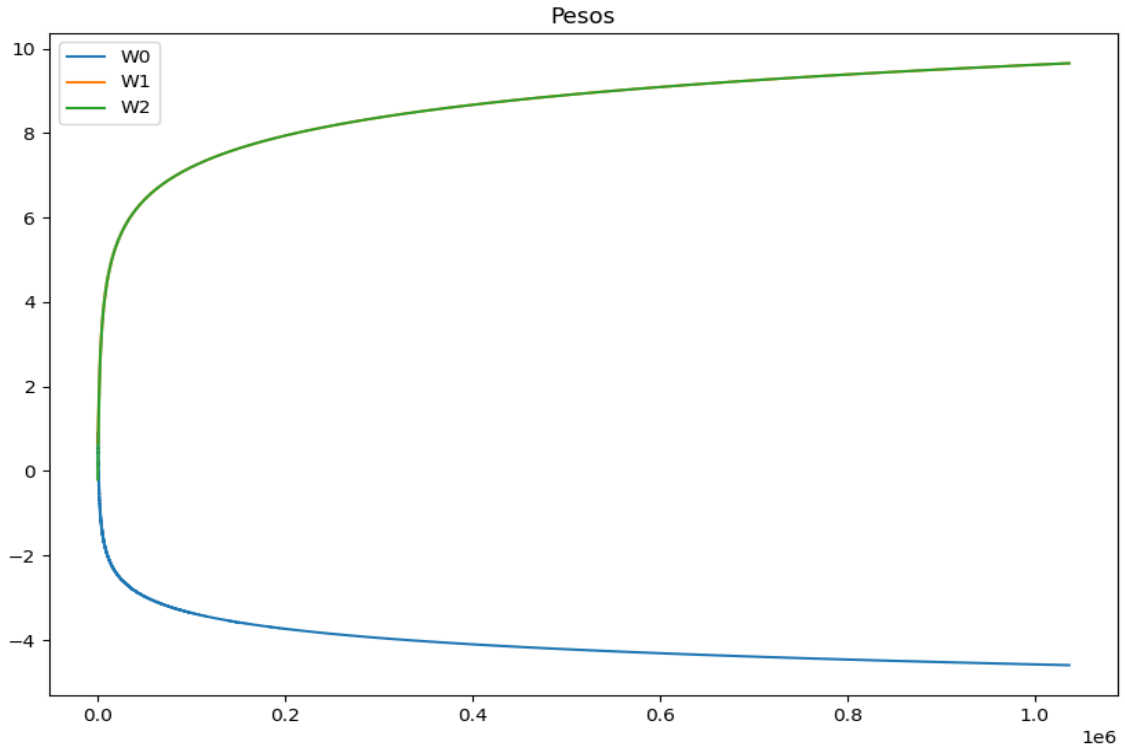
Al finalizar con el entrenamiento, llamaremos a las funciones de graficar pesos y errores.

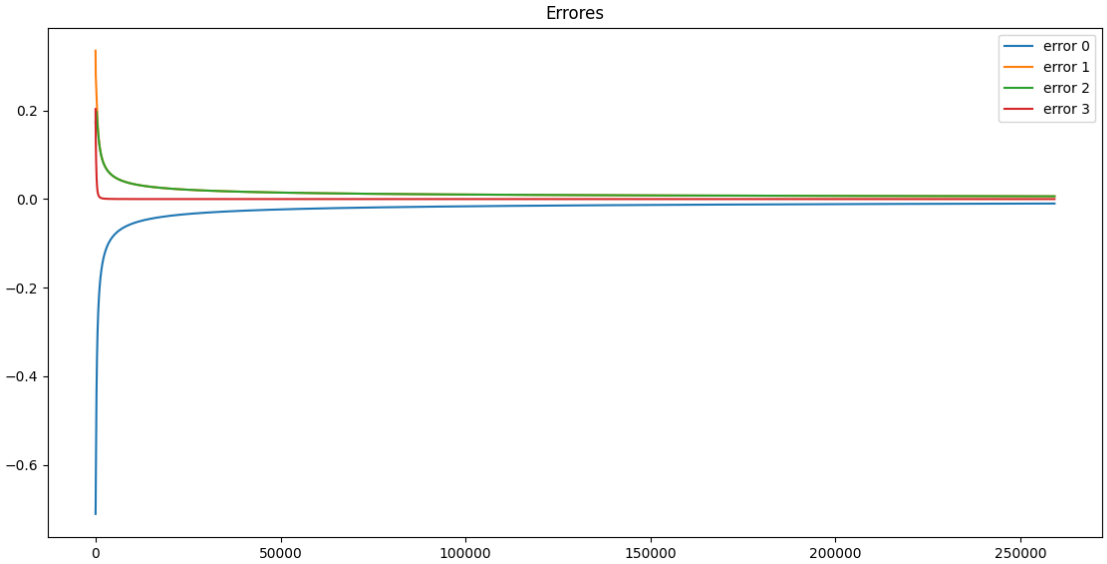
```
self.grafico_pesos(peso0, peso1, peso2, count)
self.grafico_error(error0, error1, error2, error3, counts)
```

RESULTADO EN TERMINAL

Una vez que se termine, imprimiremos cuantas veces se realizó el entrenamiento, los valores de la salida deseada y el valor de la salida real para cada valor de la tabla OR. Además, mostraremos las gráficas de los pesos y los errores.

```
Ingrese 1 para compuerta OR o 2 para compuerta AND:
1
Numero de veces que se realizo el entrenamiento: 259204
--------------------------------------------------------------------------------
La salida deseada es 0
La salida real tiene un valor de: 0.009999988682621105 para e1 igual a 0 y para e2 igual a 0
--------------------------------------------------------------------------------
La salida deseada es 1
La salida real tiene un valor de: 0.9936890703356726 para e1 igual a 0 y para e2 igual a 1
--------------------------------------------------------------------------------
La salida deseada es 1
La salida real tiene un valor de: 0.9936890465528302 para e1 igual a 1 y para e2 igual a 0
--------------------------------------------------------------------------------
La salida deseada es 1
La salida real tiene un valor de: 0.9999995925795361 para e1 igual a 1 y para e2 igual a 1
```





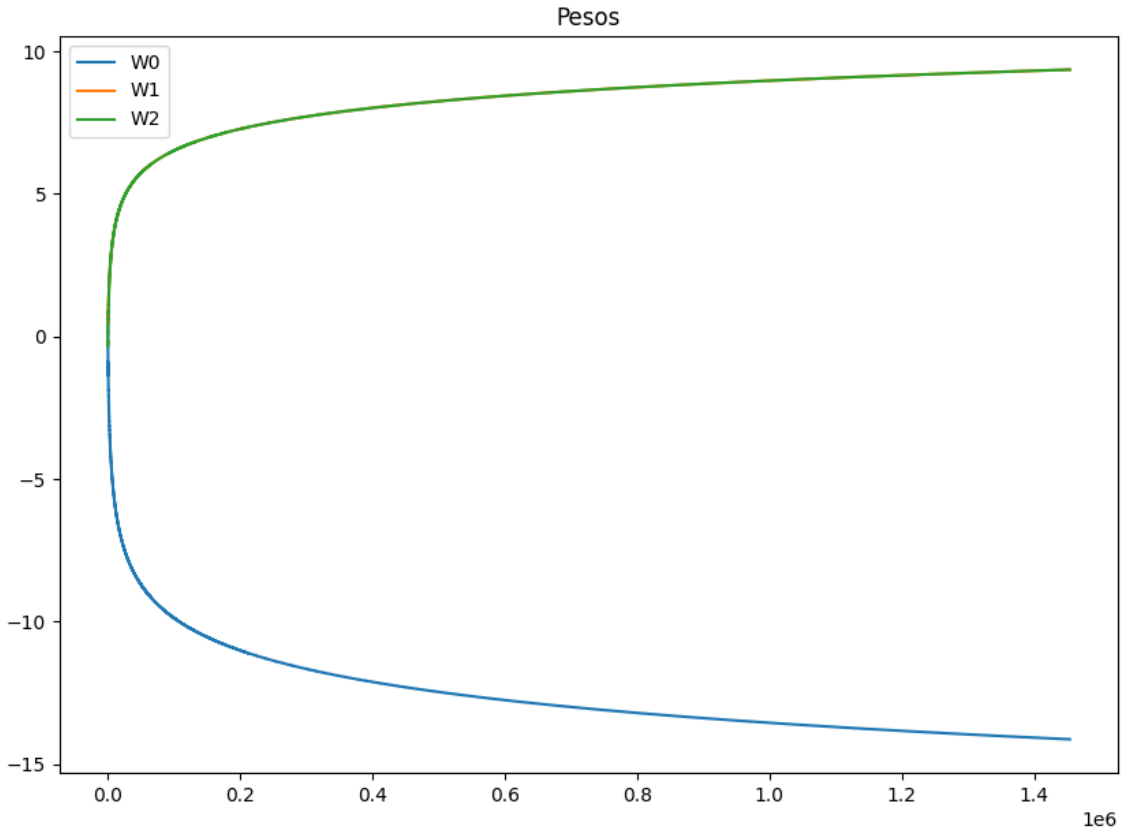
2- Perceptrón comportado como compuerta AND

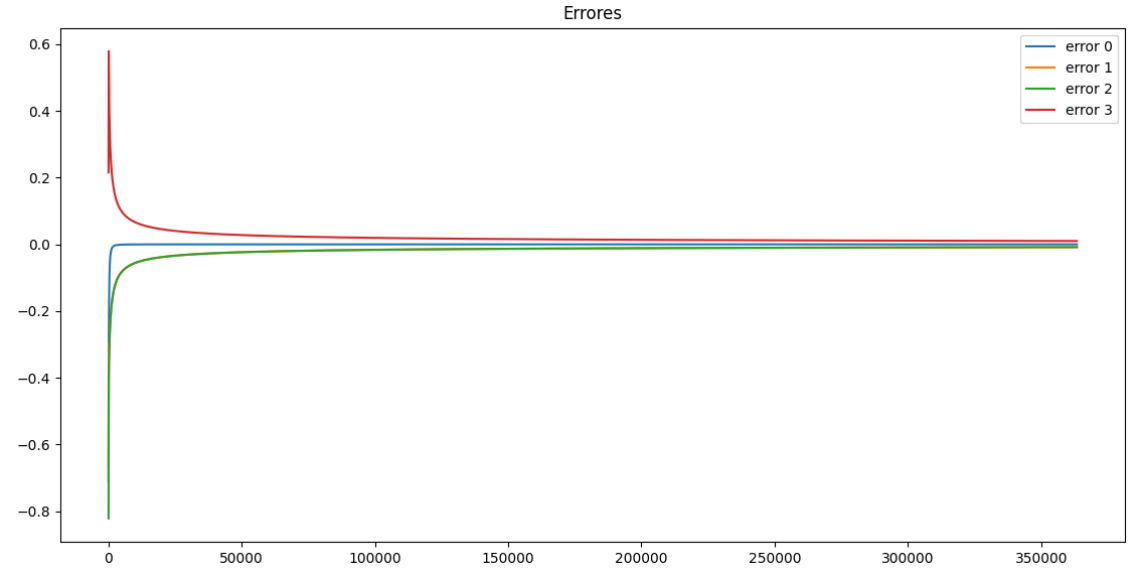
Se realizarán los mismos pasos y se utilizará el mismo código que para la compuerta OR, pero la tabla será diferente. Es decir, los valores de e1, e2 y la salida deseada cambiarán.

```
def compuerta_AND(self):
    self.valor_e1 = [0, 0, 1, 1]
    self.valor_e2 = [0, 1, 0, 1]
    self.salida_d = [0, 0, 0, 1]
    self.resolucion()
```

RESULTADO EN TERMINAL

```
Ingrese 1 para compuerta OR o 2 para compuerta AND:
2
Numero de veces que se realizo el entrenamiento: 363439
--------------------------------------------------------------------------------
La salida deseada es 0
La salida real tiene un valor de: 7.326242813990866e-07 para e1 igual a 0 y para e2 igual a 0
--------------------------------------------------------------------------------
La salida deseada es 0
La salida real tiene un valor de: 0.008444614215668693 para e1 igual a 0 y para e2 igual a 1
--------------------------------------------------------------------------------
La salida deseada es 0
La salida real tiene un valor de: 0.008444628997443257 para e1 igual a 1 y para e2 igual a 0
--------------------------------------------------------------------------------
La salida deseada es 1
La salida real tiene un valor de: 0.9900000092388987 para e1 igual a 1 y para e2 igual a 1
```





3- XOR

Nunca llega a un error cerca de 0, por lo que fallamos en lograr comportarla como una compuerta XOR.