

**TECNOLÓGICO NACIONAL DE MÉXICO**

**Instituto Tecnológico de Zitácuaro**

**Carrera:**

**Ingeniería en sistemas computacionales.**

**9 M.**

**Asignatura:**

**Inteligencia Artificial.**

**Reporte del perceptrón**

**No. De control 15650332**

**Nombre: Raunel Pérez Robles**

**Introducción**

El Perceptrón es un modelo que se planteó como la representación de una neurona biológica, pasando a ser el Perceptrón una neurona artificial, fue propuesta por Frank Rosenblat a finales de la década de 1950, el algoritmo que lo define es capaz de generar un criterio de decisión discriminando valores de forma lineal entre elementos deseados y no deseados.

En este documento se describe como fue el proceso de desarrollo del perceptrón simple. Durante este proceso se buscó reducir lo más posible el código explicando de manera breve como fue es el funcionamiento para cada sección del código. El objetivo principal es realizar la clasificación de puntos basándose en el funcionamiento de compuertas lógicas y la tabla de verdad, se muestran los resultados obtenidos para cada caso alcanzando los objetivos propuestos.

**Desarrollo**

El proceso de desarrollo está marcado por la pauta de su objetivo. El diseño del perceptrón simple que se presenta en este documento tiene como objetivo lograr la clasificación correcta de puntos con base en la estructura de las compuertas lógicas AND, OR, NAND Y NOR dicha clasificación está relacionada con los puntos que pueden ser ubicados dentro del cuadrante positivo del plano cartesiano. Cabe señalar que los puntos obtenidos son valores enteros discretos determinados entre 0 y 1.

El lenguaje de programación con el que se trabajó es Python, implementando el entorno de desarrollo Jupyter.

En el algoritmo se define como primer paso ingresar los valores de entrada, para ello se utilizaron los valores W1 (peso de entrada 1), W2 (peso de entrada 2) y b (bias) pues son los valores propuestos en la literatura para el funcionamiento del perceptrón siendo éste una representación de un modelo abstracto que pretende emular el funcionamiento de una neurona biológica tomando los valores W1 y W2 como la representación de pesos sinápticos y la variable b como el umbral de decisión.

Posteriormente fue asignado un valor a las variables W1, W2 y b de manera aleatoria en un rango determinado entre 0 y 1, para esto se utilizó la biblioteca random de Python específicamente el método uniform. Se definió además la tasa de aprendizaje representada por la variable ***l*** con un valor constante de 1, también dos matrices de datos la primera llamada ***matrix***, representa la tabla de verdad y la disposición combinatoria de dos valores en este caso 0 y 1 como se muestra en la Figura 1, la segunda matriz llamada ***t*** representa los valores de salida esperados para cada uno de los casos de acuerdo a la compuerta lógica correspondiente como se muestra en la Figura 2. Posteriormente se explicará más a detalle la funcionalidad de la matriz dentro del código.

|  |  |
| --- | --- |
| 0 | 0 |
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |
| **Figura 1** | |

|  |  |  |  |
| --- | --- | --- | --- |
| **AND** | **OR** | **NAND** | **NOR** |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| **Figura 2** | | | |

**Figura 3. Declaración de variables y librerias**

|  |
| --- |
| from random import uniform  import numpy as np  import matplotlib.pyplot as plt  matrix=np.array([[0,0],[0,1],[1,0],[1,1]])  t = np.array([[0,0,0,1],[0,1,1,1],[1,1,1,0],[1,0,0,0]])  w1= uniform(0,1)  w2= uniform(0,1)  b= uniform(0,1)  l=1  at=0 #a es el valor que se obtuvo y t el valor esperado (at)  i=0  opc = int(input("Selecciona una opción:\nAND......1\nOR.......2\nNAND.....3\nNOR......4"))  opc = opc-1 |

**Figura 4. Código de las iteraciones**

|  |  |
| --- | --- |
| 1 | while i<4:  while at==0 :  n=w1\*matrix[i,0]+w2\*matrix[i,1]+b; #Calculo de la frontera de decisión  if n>=0:  a=1  else:  a=0  if t[opc,i]!=a:  wn1=w1+(l\*(t[opc,i]-a)\*matrix[i,0])  wn2=w2+(l\*(t[opc,i]-a)\*matrix[i,1])  bn=b+(l\*(t[opc,i]-a))  w1=wn1  w2=wn2  b=bn  i=0  else:  at=1  i=i+1  at=0 |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |
| 8 |
| 9 |
| 10 |
| 11 |
| 12 |
| 13 |
| 14 |
| 15 |
| 16 |
| 17 |
| 18 |
| 19 |
| 20 |

En la sección de la Figura 4 se presenta el desarrollo del código, específicamente la sección de las iteraciones. Las iteraciones están compuestas de dos ciclos anidados, el ciclo externo es iterado con la variable ***i***, dicha variable permite cambiar el flujo de iteración dependiendo de los resultados obtenidos, este ciclo fue creado con el objetivo de recorrer la matriz ***matrix*** para poder comparar los resultados obtenidos contra los resultados esperados (definidos en la matriz ***t***) los cuales están dados en función de la compuerta lógica que se haya elegido.

La variable ***opc*** es la que permite elegir dentro de la matriz ***t*** la compuerta que se utilizará.

El segundo ciclo o ciclo interno está iterando mientras la variable ***at*** sea igual a cero, esta variable tiene la funcionalidad de una bandera e indica que mientras la comparación de la variable ***a*** con la variable obtenida de la matriz ***t*** sean distintas el ciclo deberá repetirse para ajustar los valores de los pesos ***W1***, ***W2*** y el valor del bias ***b***. Lo anterior se propuso de esta manera con el fin de simplificar el código para no tener que delimitar la funcionalidad de cada compuerta dentro de una opción IF distinta para cada caso. Se propuso la matriz ***t*** para comparar los valores con la matriz ***matrix*** por correspondencia de posición.

|  |  |  |
| --- | --- | --- |
| **i** | 0 | 0 |
| 0 | 1 |
| 1 | 0 |
| 1 | 1 |
|  | **Figura 5.1** | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **opc** | | | | |
| **0 1 2 3** | | | | |
| **i** | **AND** | **OR** | **NAND** | **NOR** |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 |
|  | **Figura 5.2** | | | |

En la Figura 5.2 puede apreciarse que la elección de compuerta se da a través de la variable ***opc*** y la salida de su valor esperado es iterado con la variable ***i***, puede verse la relación entre ambas matrices por medio del mismo iterador. Por ejemplo si se elige la opción 0 en la figura 5.2 de la variable ***opc,*** esta corresponde a la compuerta AND, después cuando se itera en la matriz de la figura 5.1 comenzando con ***i*** = 0, en la posición 0 se encuentran los valores [0,0] y por correspondencia de posición dentro de la matriz de la figura 5.2 que representa a la matriz ***t*** el valor de salida esperado en la posición t[opc,i] es igual a 0, como puede observarse al continuar iterando ***i***, los valores esperados están dados por la correspondencia de posición de ambas matrices.

Dentro del segundo ciclo se llevó a cabo el cálculo y la evaluación de ***n,*** dicha variable representa a la frontera de decisión es decir la función de la recta que delimita una frontera entre los puntos a clasificar. En la línea 3 del código mostrado en la figura 4 se lleva a cabo la operación de la frontera de decisión, posteriormente ***n*** es evaluada dentro de un IF el cual representa a la función de activación mostrada a continuación en la Figura 6.

|  |
| --- |
| a(n)= 0 , n<0 si n es menor a 0 entonces a = 0  1 , n>=0 si n es igual o mayor que 0 entonces a = 1  Se realiza una discretización de la variable ***a*** convirtiéndola a entera con valores de 0 ó 1. |

**Figura 6.**

Una vez que se realizó la evaluación de la función de activación se compara si el valor esperado ***t*** es igual a el valor obtenido ***a***, en caso de ser iguales querrá decir que el perceptrón está clasificando correctamente, a la variable ***at*** se le asigna 1, este valor de bandera permitirá romper el ciclo internopues quiere decir que ya no es necesario realizar más actualizaciones a los valores ***W1***, ***W2*** y ***b***. En caso de que ***a*** y ***t*** sean diferentes se hace la actualización de pesos y del bias además se cambia el valor de la variable ***i***, como se muestra en la Figura 4 de las líneas 9 a la 15. La variable ***i*** además de estar realizando la funcionalidad de iterador está cumpliendo también una funcionalidad que permite cambiar la dirección del flujo de iteración pues cuando ***a*** es diferente de ***t*** se realiza la actualización de pesos, si ocurre una actualización entonces el ciclo externo o primer ciclo debe volver a iterar desde 0 reiniciando i=0 para evaluar los puntos anteriores con los nuevos pesos y el nuevo bias.

Finalmente se realizó la graficación de los datos obtenidos, para esto se utilizó la biblioteca matplotlib.pyplot de Python, para graficar la frontera de decisión se tomaron los últimos valores obtenidos para los pesos y el bias pues estos representan los valores correctos para la clasificación, recordando que la función de la frontera de decisión esta dada por la formula n=W1x1 + W2x2+b=0, al realizar el despeje para graficar debe obtenerse X e Y por separado, para esto X está representado por W1x1 e Y es representado por W2x2 cuando se obtiene X, la variable Y debe estar igualada a 0 y viceversa por ende los valores quedan W1x1+b=0 y W2x2+b=0, al despejar los valores queda de la siguiente forma: W1x1=-b y W2x2=-b, continuando con el despeje x1=-b/W1 y x2=-b/W2, dado lo anterior desde un inicio b es multiplicado por -1 como puede observarse en la línea 1 de la figura 7. Posteriormente se realizan las divisiones correspondientes para obtener el punto 1(x1) y el punto 2(x2) como se muestra en las líneas 2 y 3 respectivamente en la Figura 7. Se traza una línea del punto (x1,0) al (0,x2) como se muestra en la línea 4 de la figura 7, la disposición de los puntos está dada en base al despeje como se explicó anteriormente para trazar X, la variable Y debe ser 0 y visceversa por eso el primer punto tiene 0 en su parte de y (x,0) e Y tiene 0 en su parte de x (y,0). Con los valores ***W1*** y ***W2*** se traza el vector **W** del origen (0,0) a (**W1**,**W2**).

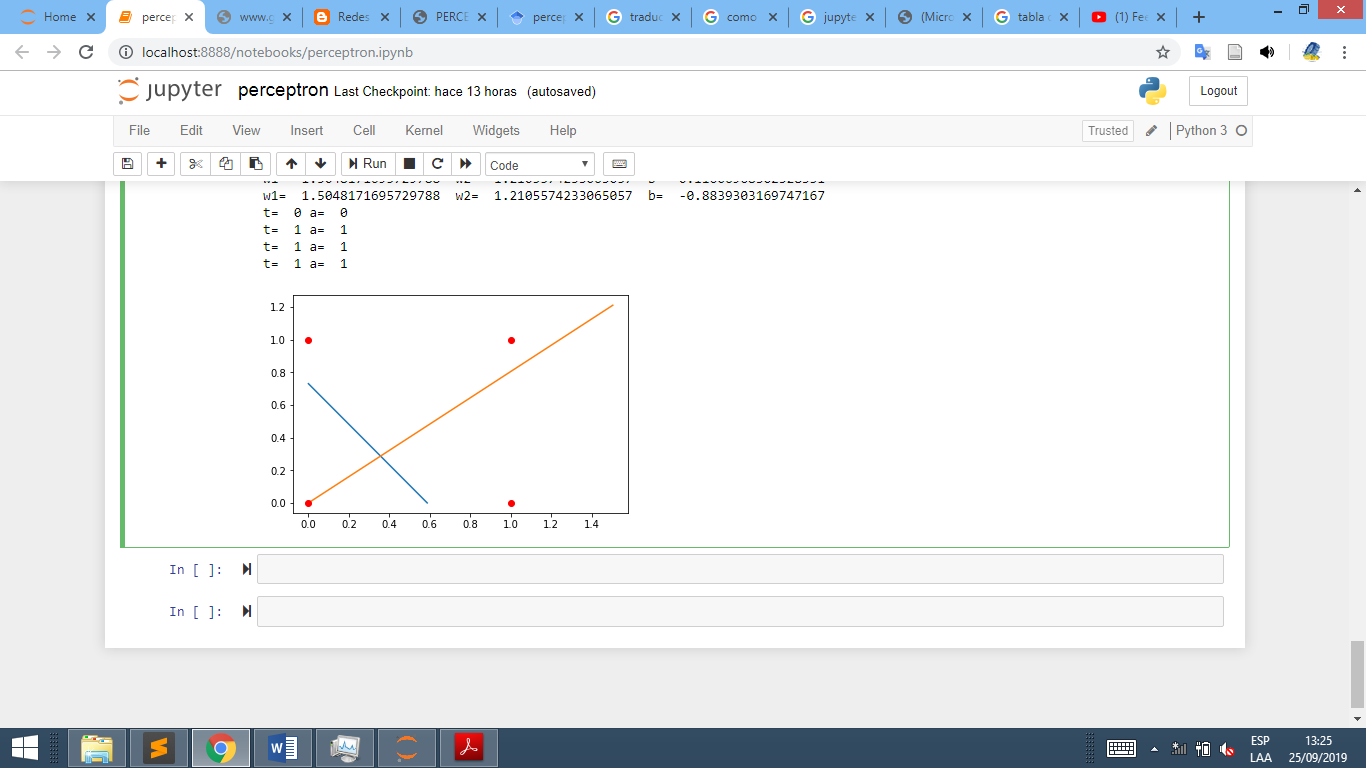
|  |  |
| --- | --- |
| 1 | b=b\*-1  p1=(b)/w1  p2=b/w2  plt.plot([p1,0],[0,p2]) #Frontera de decisión  plt.plot([0,w1],[0,w2]) #Vector W  plt.plot([0, 0, 1, 1], [0, 1, 0, 1], 'ro') #Traza los puntos (0,0),(0,1),(1,0),(1,1) de color rojo  plt.show() #Muestra el diseño |
| 2 |
| 3 |
| 4 |
| 5 |
| 6 |
| 7 |

**Figura 7.**

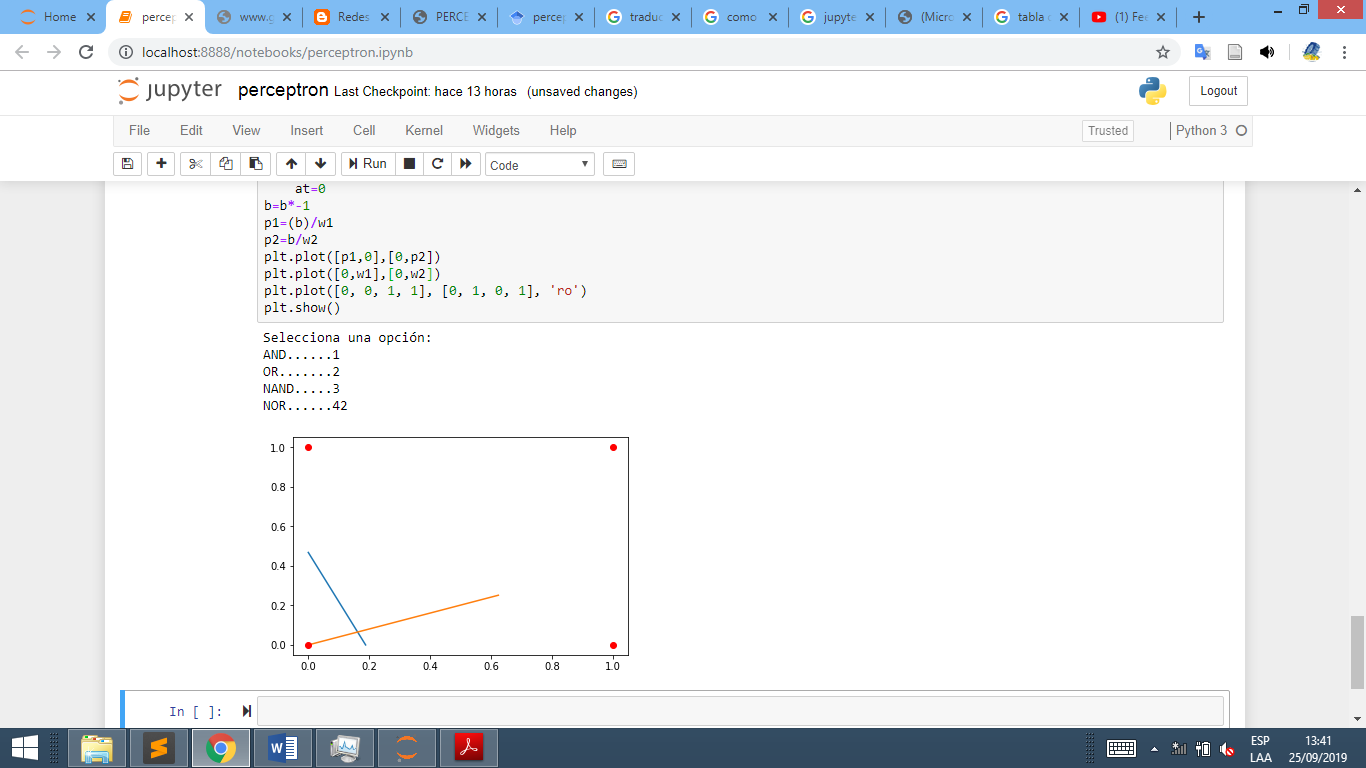
**Resultados experimentales**

Se realizó la ejecución del Perceptrón, a continuación se muestran los resultados obtenidos para cada una de las compuertas, la frontera de decisión está representada en color azul y el vector W en color naranja, el vector w apunta del origen (0,0) hacia (w1,w2) siendo esa su dirección, es sabido que el vector W debe ser ortogonal a la frontera de decisión pero en este diseño de graficación se crea una distorsión en el ángulo de representación del vector W pues los valores que se toman como escala son creados automáticamente por la función plot, creando escalas que **no** son cuadráticamente equivalentes.

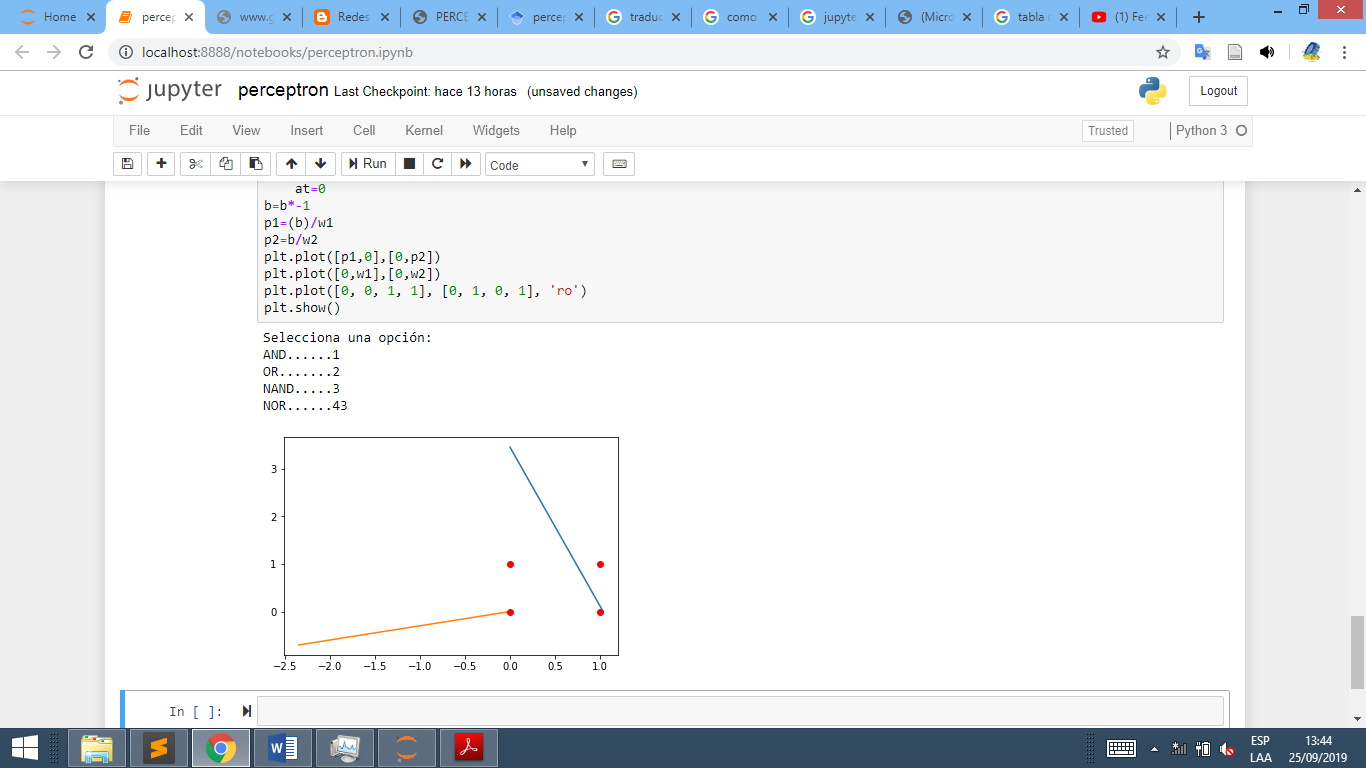
**Figura 8. Clasificación compuerta AND**



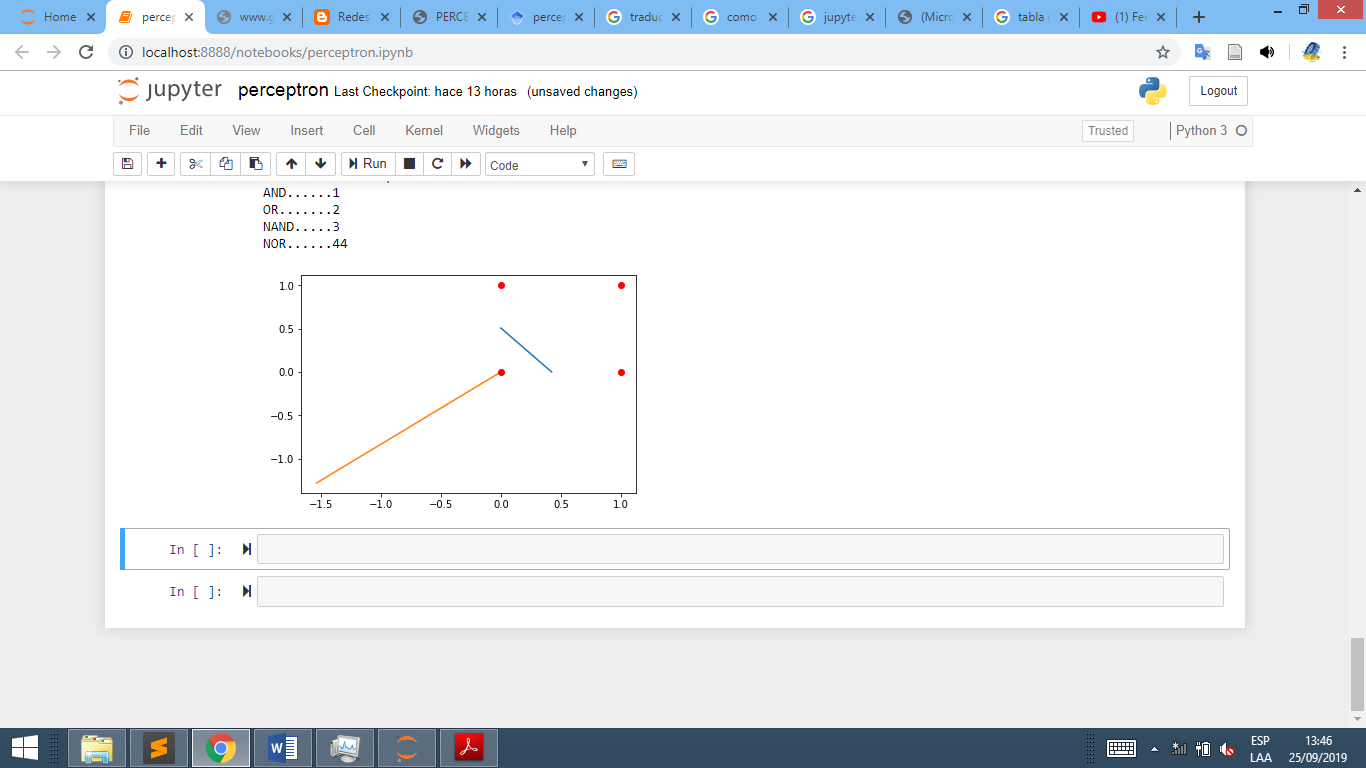
**Figura 9. Clasificación compuerta OR**



**Figura 10. Clasificación compuerta NAND**



**Figura 11. Clasificación compuerta NOR**



**Conclusión**

Los resultados obtenidos muestran que el desarrollo de este Perceptrón cumple con los objetivos esperados pues logra clasificar de manera correcta los valores introducidos para cada una de las compuertas lógicas realizando los ajustes necesarios en los pesos y el bias, en un inicio los valores propuestos no cumplen para todos los casos pero conforme va iterando en el algoritmo su valor va volviéndose más y más preciso hasta que logra refinarse y clasificar de manera correcta todas las combinaciones a partir de la tabla de verdad.

**Referencias**

Martin T. Hagan, Howard B. Demuth, Mark Hudson Beale, Orlando De Jesús, Neural Network Design, Edición: 2., 2014.