Perceptrón

Villaseñor López Isaac Alejandro
Universidad de Guadalajara
Centro Universitario de Ciencias Exactas e Ingenierías
Inteligencia Artificial II

I. Introduccion

En el siguiente documento se presentan los objetivos de la actividad Perceptrón de la materia de Inteligencia Artificial II, esta referente a el perceptrón la primera simulación de una neurona, se encontrará también el desarrollo de la actividad misma donde se comenta el objetivo y los valores con los que se entreno la neurona para presentarse a distintos problemas linealmente y no linealmente separables como veremos más adelante.
Para finalizar se encuentra las conclusiones de la actividad, las referencias del material utilizado para la realización y por último el código de ambas neuronas utilizadas.

II. Objetivos

- Codifica una neurona artificial entrenado con el algoritmo del perceptrón,
- Entrena esta neurona para aprender las compuertas lógicas AND, OR y XOR
- Explica por qué la XOR no puede ser aproximada con esta neurona.
- Grafica los resultados dibujando los datos con los que entrenaste, de manera que datos de clases diferentes tengan diferente color y dibuja también la línea representada por la neurona artificial
- Entrena la neurona para para aproximar la tarea de clasificar a una persona si tiene sobrepeso o no, según su peso y altura.
- Grafica los resultados al igual que los anteriores.
- Qué diferencias hay entre usar el índice de masa corporal o la neurona artificial para hacer la clasificación

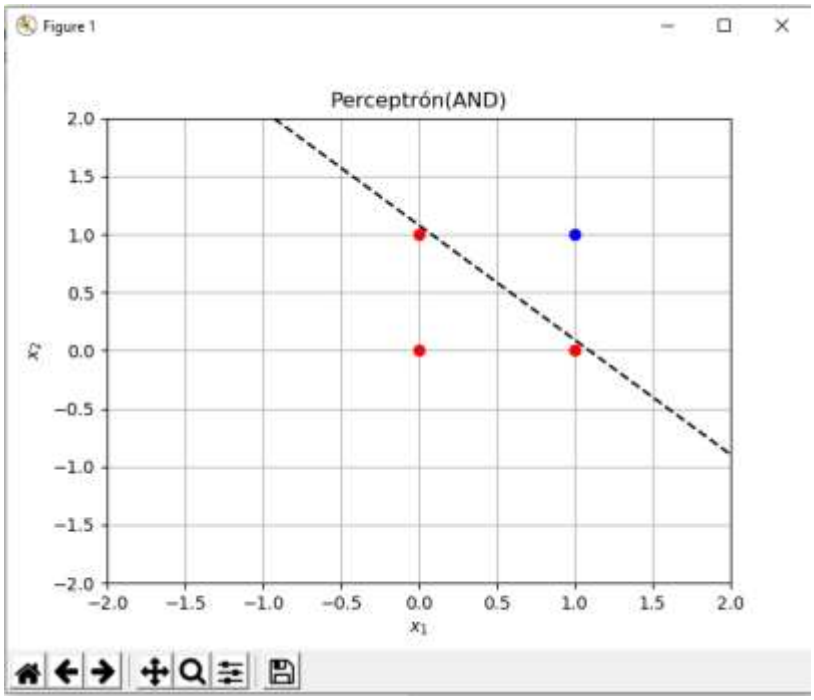
III. Desarrollo

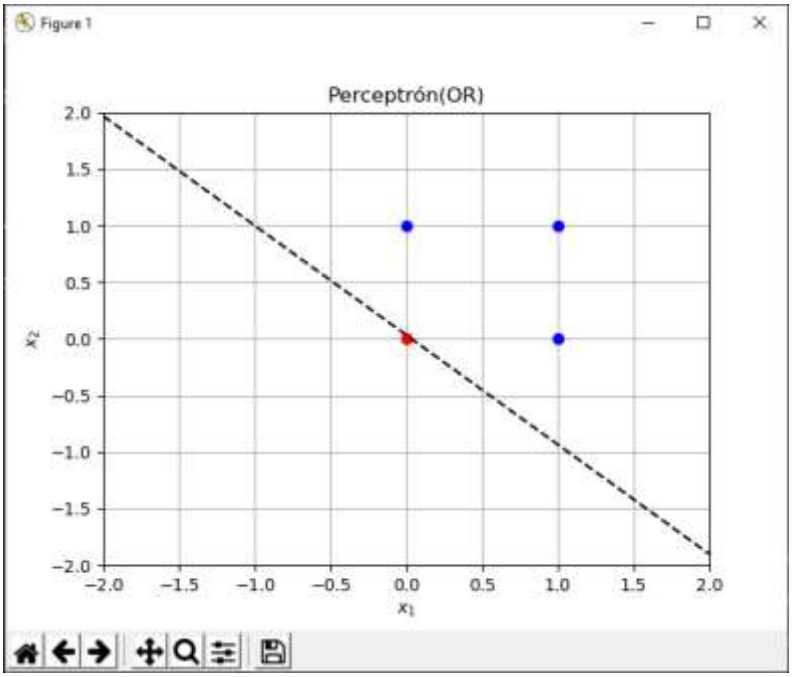
Se codifico la neurona artificial siguiendo la guía del video proporcionado por el profesor, creando los vectores para los siguientes resultados:

Perceptrón AND:

Podemos observar como el perceptrón logra aproximarse a esta solución al ser entrenado con 50 épocas y un coeficiente de aprendizaje de 0.05.

El código se encuentra al final del documento.





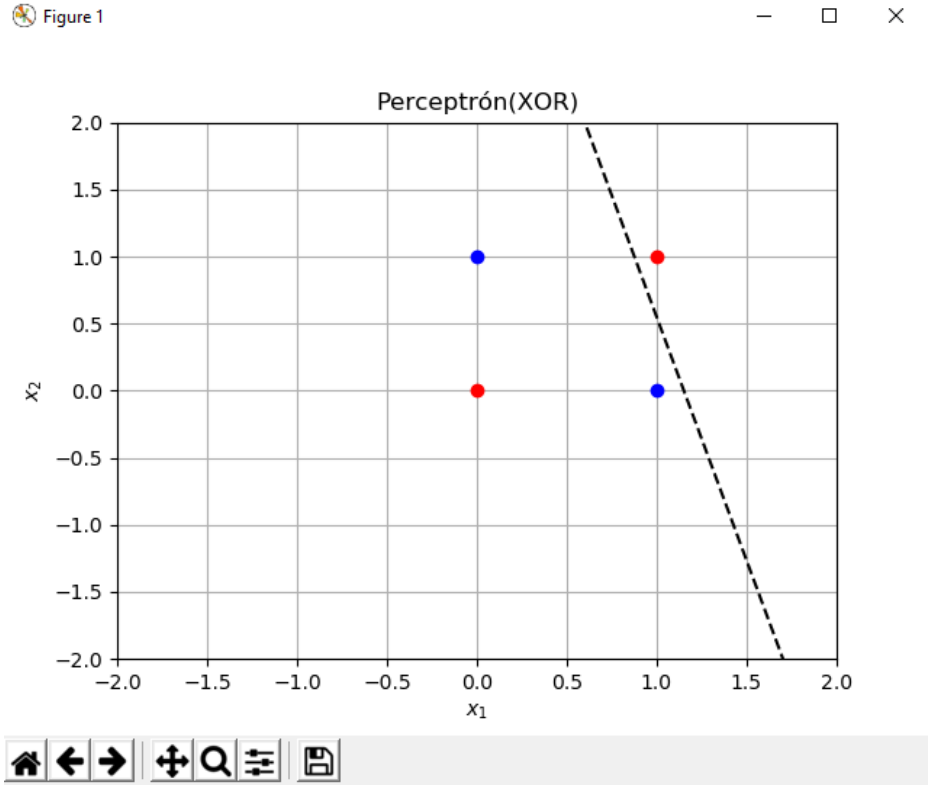
Perceptrón OR:

Se utiliza el mismo perceptrón ya codificado solo se cambia la salida deseada por la representación de OR de acuerdo con las entradas.

Aquí el resultado del perceptrón con 50 épocas y un coeficiente de aprendizaje de 0.05.

Perceptrón XOR:

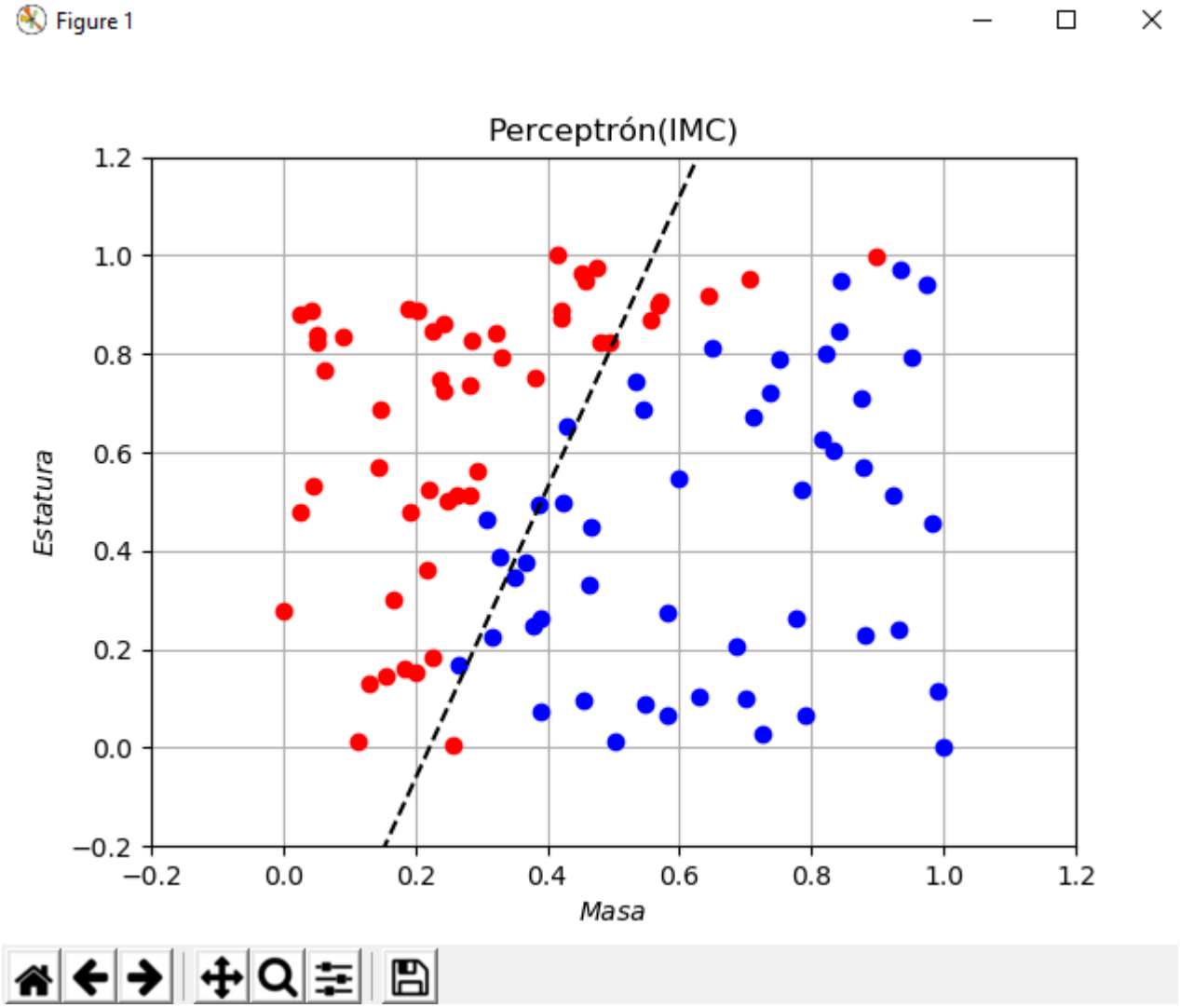
Como podemos observar la neurona no puede aproximarse a la salida del XOR debido a que no es un problema linealmente separable, aquí se muestra la "solución" después de 50 épocas y 0.05 de coeficiente de aprendizaje



Perceptrón IMC:

Para el entrenamiento de la neurona y el IMC se alimentó con datos aleatorios, 200 para ser especifico y se normalizaron estos datos para que se pudiera acercar a este problema linealmente separable.

Podemos notar que este problema puede acercarse a ser linealmente separable pero no termina de serlo, esa es la diferencia entre usar la neurona y el índice de masa corporal, pues el índice nos dará resultados mas precisos que la neurona.



IV. Conclucion

Esta actividad fue un buen acercamiento a la simulación de neuronas, si bien no es la mejor opción o más eficiente, sirvió para explicar el funcionamiento de estas o al menos como se intenta simular la neurona dentro de un modelado matemático y luego en un programa.
También podemos remarcar que con la prueba del XOR queda mas que claro que esta neurona solo funciona para problemas linealmente separables.

V. Referencias

Toda la información que se utilizo para elaborar esta actividad fue proporcionada por el Dr. Carlos Villaseñor.
Se anexan los links a los videos:
https://www.youtube.com/watch?v=ByEjBAKb05I (IA2.02 El Perceptrón)
https://www.youtube.com/watch?v=cNxadbrN_aI (Perceptron Research from the 50's & 60's, clip)

VI. Código

Perceptrón(AND, OR, XOR)

```
import numpy as np
import matplotlib.pyplot as plt


class Perceptron:
    def __init__(self, n_input, learning_rate):
        self.w = -1 + 2*np.random.rand(n_input)
        self.b = -1 + 2*np.random.rand()
        self.eta = learning_rate

    def predict(self, X):
        p = X.shape[1]
        y_est = np.zeros(p)
        for i in range(p):
            y_est[i] = np.dot(self.w, X[:,i]) + self.b
            if y_est[i] >= 0:
                y_est[i] = 1
            else:
                y_est[i] = 0
        return y_est

    def fit(self, X, Y, epoch=50):
        p = X.shape[1]
        for _ in range(epoch):
            for i in range(p):
                y_est = self.predict(X[:,i].reshape(-1,1))
                self.w += self.eta + (Y[i]-y_est) * X[:,i]
                self.b += self.eta + (Y[i]-y_est) * 1

def draw_2d(model):
    w1, w2, b = model.w[0], model.w[1], model.b
    li, ls = -2, 2
    plt.plot([li,ls],
    [(1/w2)*(-w1*(li)-b),(1/w2)*(-w1*(ls)-b)], '--k')

neuron = Perceptron(2,0.05)
X = np.array([[0,0,1,1],
              [0,1,0,1]])
#Y = np.array([0,0,0,1])    #AND
#Y = np.array([0,1,1,1])    #OR
Y = np.array([0,1,1,0])    #XOR
neuron.fit(X,Y)

#Dibujo
_, p = X.shape
```

```
for i in range(p):
    if Y[i] == 0:
        plt.plot(X[0,i],X[1,i], 'or')
    else:
        plt.plot(X[0,i],X[1,i], 'ob')
plt.title('Perceptrón(IMC)')
plt.grid('on')
plt.xlim([-2,2])
plt.ylim([-2,2])
plt.xlabel(r'$x_1$')
plt.ylabel(r'$x_2$')

draw_2d(neuron)
plt.show()
```

Perceptron(IMC)

```
import numpy as np
import matplotlib.pyplot as plt


class Perceptron:
    def __init__(self, n_input, learning_rate):
        self.w = -1 + 2*np.random.rand(n_input)
        self.b = -1 + 2*np.random.rand()
        self.eta = learning_rate

    def predict(self, X):
        p = X.shape[1]
        y_est = np.zeros(p)
        for i in range(p):
            y_est[i] = np.dot(self.w, X[:,i]) + self.b
            if y_est[i] >= 0:
                y_est[i] = 1
            else:
                y_est[i] = 0
        return y_est

    def fit(self, X, Y, epoch=50):
        p = X.shape[1]
        for _ in range(epoch):
            for i in range(p):
                y_est = self.predict(X[:,i].reshape(-1,1))
                self.w += self.eta + (Y[i]-y_est) * X[:,i]
                self.b += self.eta + (Y[i]-y_est) * 1

def draw_2d(model):
    w1, w2, b = model.w[0], model.w[1], model.b
```

```
    li, ls = -2, 2
    plt.plot([li,ls],
    [(1/w2)*(-w1*(li)-b),(1/w2)*(-w1*(ls)-b)], '--k')

neuron = Perceptron(2,0.05)
p = 200
X = np.zeros((2,p))
Y = np.zeros(p)

for i in range (p):
    X[0,i] = -40 + (120+40) * np.random.rand()
    X[1,i] = -1 + (2.2+1) * np.random.rand()
    imc = X[0,i] / X[1,i]**2
    if imc >=25:
        Y[i] = 1
    else:
        Y[i] = 0

X[0,:] = (X[0,:]-X[0,:].min())/(X[0,:].max()-X[0,:].min())
X[1,:] = (X[1,:]-X[1,:].min())/(X[1,:].max()-X[1,:].min())


neuron.fit(X,Y, epoch=100)

#Dibujo
_, p = X.shape
for i in range(p):
    if Y[i] == 0:
        plt.plot(X[0,i],X[1,i], 'or')
    else:
        plt.plot(X[0,i],X[1,i], 'ob')
plt.title('Perceptrón(IMC)')
plt.grid('on')
plt.xlim([-0.2,1.2])
plt.ylim([-0.2,1.2])
plt.xlabel(r'$Masa$')
plt.ylabel(r'$Estatura$')

draw_2d(neuron)
plt.show()
```