Práctica 4. Perceptrón simple

González Flores Andrés

Actividad 1

Entrenar un perceptrón simple con el algoritmo de pasos decendentes con los puntos P = { {1, -1, -1}, {1, 1, -1}} con t = {0,1}

Condiciones iniciales w = {1, -1, -0.5}, b=1

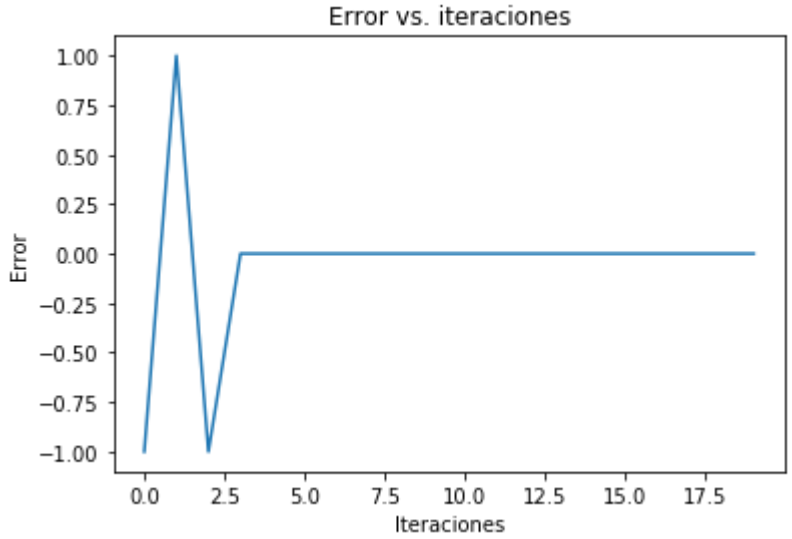
Código de perceptron.py

```
import numpy as np

def hardlim(x):
    return 1 if x>0 else 0

def afn(x, w, b):
    return hardlim(np.dot(w, x.T)+b)

def efn(t, a):
    return t-a

def reglaApr(w_ant, b_ant, e, p, lr=1):
    w_nuevo = w_ant + lr*e*p
    b_nuevo = b_ant + lr*e
    return w_nuevo, b_nuevo
```

p4ej1.py

In [1]:
```
import numpy as np
import perceptron as per
import matplotlib.pyplot as plt

p_vect = [np.matrix([1, -1, -1]), np.matrix([1, 1, -1])]
t_vect = [0, 1]

w = np.matrix([1, -1, -0.5])
b = np.matrix([1])

epochs = 10
e_vect = []

for i in range(epochs):
    for p, t in zip(p_vect, t_vect):
        #print('Para el punto p = ', p)
        a = per.afn(p, w, b)
        #print('a = ', a)
        e = per.efn(t, a)
        e_vect.append(e)
        #print('e = ', e)
        w, b = per.reglaApr(w, b, e, p)
        #print('w = ', w)
        #print('b = ', b)
        #print("Epoca %d, Error = %d" % (i, e))
```

In [2]:
```
plt.plot(e_vect)
plt.title('Error vs. iteraciones')
plt.ylabel('Error')
plt.xlabel('Iteraciones')
plt.show()
```



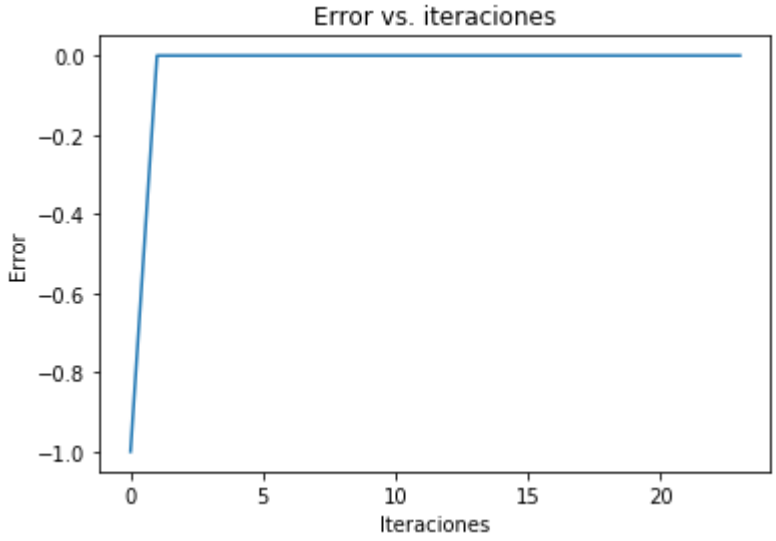
Actividad 2

Entrenar un perceptrón simple con el algoritmo de pasos decendentes con los puntos P = {{1,1}, {1,0}, {1,2}, {5,1}, {0,-1}, {-1,0}, {-2,-1}, {-3,-2}} con t = {0,0,0,0,1,1,1,1}

In [3]:
```
p_vect = [[1,1], [1,0], [1,2], [5,1], [0,-1], [-1,0], [-2,-1], [-3,-2]]
p_vect = [np.matrix(p) for p in p_vect]
t_vect = [0,0,0,0,1,1,1,1]

w = np.matrix(np.random.rand(2))
b = np.matrix(np.random.rand(1))

lr = 0.5
epochs = 3
e_vect = []

for epoch in range(epochs):
    for i, p in enumerate(p_vect):
        a = per.afn(p, w, b)
        e = per.efn(t_vect[i], a)
        e_vect.append(e)
        w, b = per.reglaApr(w, b, e, p, lr)
        if i % 4 == 0:
            print("Epoca %d (%i/%i), Error = %d" % (epoch, i, len(p_vect), e))
```

```
Epoca 0 (0/8), Error = -1
Epoca 0 (4/8), Error = 0
Epoca 1 (0/8), Error = 0
Epoca 1 (4/8), Error = 0
Epoca 2 (0/8), Error = 0
Epoca 2 (4/8), Error = 0
```

In [4]:
```
plt.plot(e_vect)
plt.title('Error vs. iteraciones')
plt.ylabel('Error')
plt.xlabel('Iteraciones')
plt.show()
```



En esta actividad podemos notar que los puntos son linealmente separables, por lo que el algoritmo de pasos decendentes converge sin problemas.

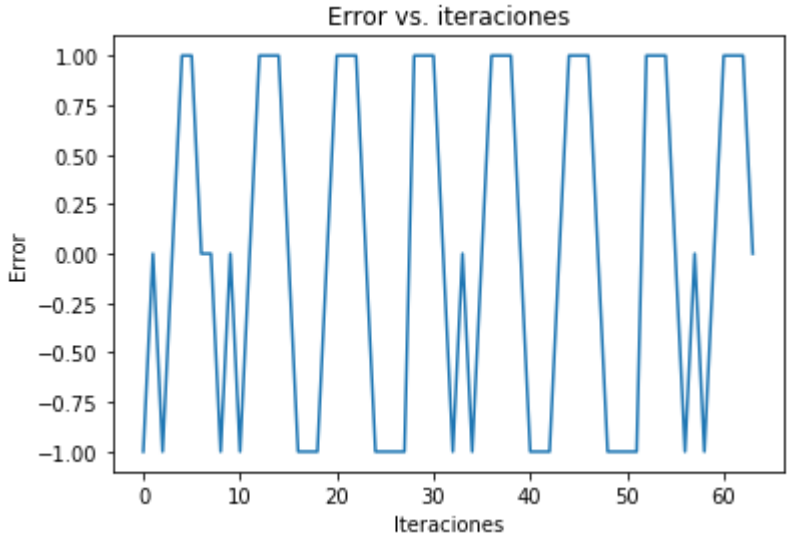
Actividad 3

Entrenar un perceptrón simple con el algoritmo de pasos decendentes con los puntos P = {{0,3}, {2,0}, {0,-1}, {-1,0}, {0,0}, {0,1}, {1,1}, {1,1}} con t = {0,0,0,0,1,1,1,1}

In [5]:
```
p_vect = [[0,3], [2,0], [0,-1], [-1,0], [0,0], [0,1], [1,1], [1,1]]
p_vect = [np.matrix(p) for p in p_vect]
t_vect = [0,0,0,0,1,1,1,1]

w = np.matrix(np.random.rand(2))
b = np.matrix(np.random.rand(1))

lr = 0.5
epochs = 8
e_vect = []

for epoch in range(epochs):
    for i, p in enumerate(p_vect):
        a = per.afn(p, w, b)
        e = per.efn(t_vect[i], a)
        e_vect.append(e)
        w, b = per.reglaApr(w, b, e, p, lr)
        if i % 4 == 0:
            print("Epoca %d (%i/%i), Error = %d" % (epoch, i, len(p_vect), e))
```

```
Epoca 0 (0/8), Error = -1
Epoca 0 (4/8), Error = 1
Epoca 1 (0/8), Error = -1
Epoca 1 (4/8), Error = 1
Epoca 2 (0/8), Error = -1
Epoca 2 (4/8), Error = 1
Epoca 3 (0/8), Error = -1
Epoca 3 (4/8), Error = 1
Epoca 4 (0/8), Error = -1
Epoca 4 (4/8), Error = 1
Epoca 5 (0/8), Error = -1
Epoca 5 (4/8), Error = 1
Epoca 6 (0/8), Error = -1
Epoca 6 (4/8), Error = 1
Epoca 7 (0/8), Error = -1
Epoca 7 (4/8), Error = 1
```

In [6]:
```
plt.plot(e_vect)
plt.title('Error vs. iteraciones')
plt.ylabel('Error')
plt.xlabel('Iteraciones')
plt.show()
```



Para poder separar los puntos de esta actividad se necesitan más capas del perceptrón, ya que estos no son linealmente separables. Esto hace que el algoritmo nunca converga, como se nota en la gráfica de error vs. iteraciones.

In [ ]:
```
```