```
# -*- coding: utf-8 -*-
"""
@author: chirag
"""
import numpy as np
import time


def threshold(x):
	if x >=0:
		return 1
	else:
		return -1


def print_func(loop_var, net, sig_net, teacher_signal, w, delta_w = None):
		print("i: "+ str(loop_var))
		print("net: "+ str(net))
		print("sig_net: "+ str(sig_net))
		print("delta_w: "+ str(delta_w))
		print("teacher_signal: "+ str(teacher_signal))
		print("w: "+ str(w))
		print("-------------------\n")


def compute():
	try:
		n = int(input("Enter number of input vectors: "))
		x = []
		r = 0.1 #Learning rate
		for i in range(0,n):
			raw_str1 = str(input("Enter values for vector " + str(i+1) + ": "))
			input_vector = raw_str1.split(' ')
			#print(input_vector)
			ip_list = []
			for ele in input_vector:
				ip_list.append(float(ele))
```

```
        #print(ip_list)
        np_list = np.array(ip_list, dtype=np.float64)
        x.append(np_list)

raw_str2 = str(input("Enter values for teacher signal: "))
teacher_signal = raw_str2.split(' ')
teacher_signal = [int(x) for x in teacher_signal]
if len(teacher_signal) != n:
        print("Teacher Signal length Error..")
        time.sleep(3)
        exit()
raw_str3 = str(input("Enter initial weight vector: "))
w = raw_str3.split(' ')
w_list = []
for ele in w:
        w_list.append(float(ele))
np_wlist = np.array(w_list, dtype=np.float64)
#print(np_wlist)

delta_w = 0
for i in range(0,n):
        #print(x[i])
        #print(w_list)
        #print(x[i])

        net = np.transpose(np.asarray(w_list)).dot(np.asarray(x[i]))
        #print(net)
        sig_net = threshold(net)
        #print(sig_net)

        if sig_net != teacher_signal[i]:
                delta_w = r * (teacher_signal[i] - sig_net) * x[i]
                #print(delta_w)
                w_list = np.add(np.asarray(w_list),delta_w)
```

```
                else:
                    w_list = w_list
                print_func(i, net, sig_net, teacher_signal[i], w_list, delta_w)
        except Exception as e:
            print("Error.. "+(str(e)))

if __name__ == '__main__':
    compute()
```

Output: