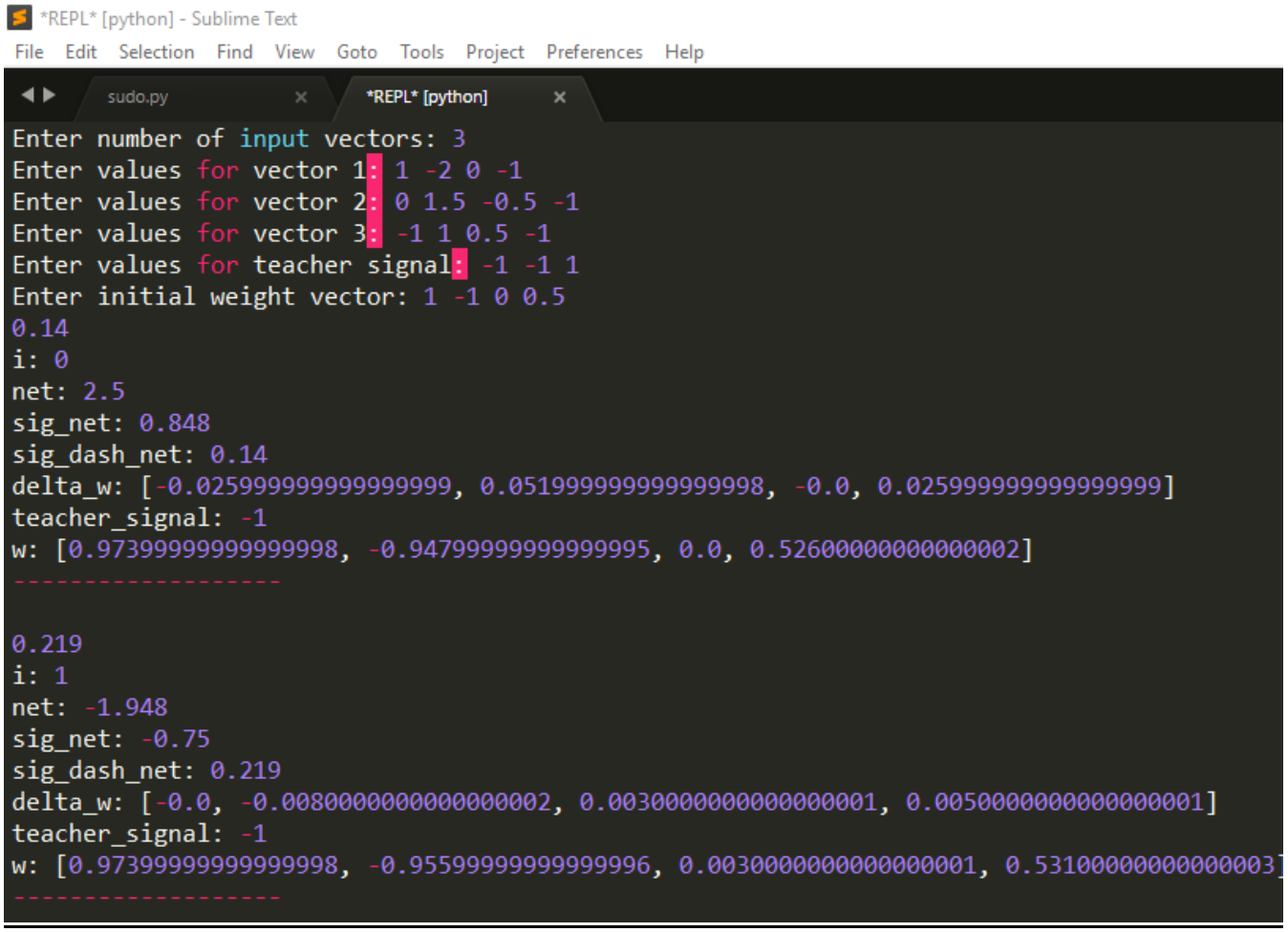
Code:

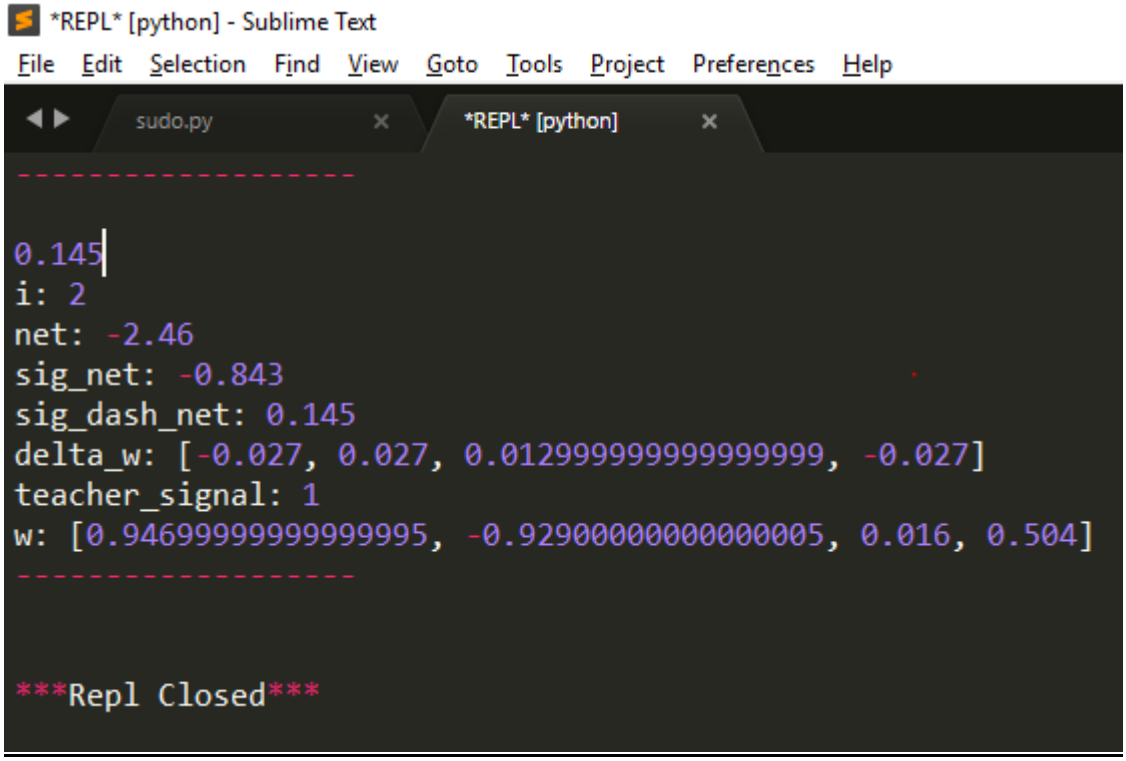
```
# -*- coding: utf-8 -*-
"""
@author: chirag
"""
import numpy as np
import time, math


def threshold(x):
	return round(((2 / (1 + math.exp(-x))) - 1), 3)


def print_func(loop_var, net, sig_net, sig_dash_net, teacher_signal, w, delta_w = None):
		print("i: "+ str(loop_var))
		print("net: "+ str(net))
		print("sig_net: "+ str(sig_net))
		print("sig_dash_net: "+ str(sig_dash_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(round(float(ele), 3))
np_wlist = np.array(w_list, dtype=np.float64)
#print(np_wlist)
delta_w = 0
for i in range(0,n):
        net = round(np.transpose(np.asarray(w_list)).dot(np.asarray(x[i])), 3)
        #print(net)
        sig_net = threshold(net)
        sig_dash_net = round(0.5 * ( 1 - ((sig_net)**2)), 3)
        print(sig_dash_net)
        if sig_net != teacher_signal[i]:
                rounded_delta_w = []
                rounded_w_list = []
                delta_w = (r * (teacher_signal[i] - sig_net) * sig_dash_net * x[i])
                for ele in delta_w:
                        rounded_delta_w.append(round(ele, 3))
                #print(delta_w)
                w_list = np.add(np.asarray(w_list), rounded_delta_w)
                for ele in w_list:
```

```
                                        rounded_w_list.append(round(ele, 3))
                                w_list = rounded_w_list
                        else:
                                rounded_w_list = []
                                for ele in w_list:
                                        rounded_w_list.append(round(ele, 3))
                                w_list = rounded_w_list
                        print_func(i, net, sig_net, sig_dash_net, teacher_signal[i], w_list, rounded_delta_w)
        except Exception as e:
                print("Error.. "+(str(e)))


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

Output:


REPL [python] - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
sudo.py
REPL [python]
Enter number of input vectors: 3
Enter values for vector 1: 1 -2 0 -1
Enter values for vector 2: 0 1.5 -0.5 -1
Enter values for vector 3: -1 1 0.5 -1
Enter values for teacher signal: -1 -1 1
Enter initial weight vector: 1 -1 0 0.5
0.14
i: 0
net: 2.5
sig_net: 0.848
sig_dash_net: 0.14
delta_w: [-0.025999999999999999, 0.051999999999999998, -0.0, 0.025999999999999999]
teacher_signal: -1
w: [0.97399999999999998, -0.94799999999999995, 0.0, 0.52600000000000002]
0.219
i: 1
net: -1.948
sig_net: -0.75
sig_dash_net: 0.219
delta_w: [-0.0, -0.0080000000000000002, 0.0030000000000000001, 0.0050000000000000001]
teacher_signal: -1
w: [0.97399999999999998, -0.95599999999999996, 0.0030000000000000001, 0.53100000000000003]


REPL [python] - Sublime Text
File Edit Selection Find View Goto Tools Project Preferences Help
sudo.py
REPL [python]
0.145
i: 2
net: -2.46
sig_net: -0.843
sig_dash_net: 0.145
delta_w: [-0.027, 0.027, 0.012999999999999999, -0.027]
teacher_signal: 1
w: [0.94699999999999995, -0.92900000000000005, 0.016, 0.504]
Repl Closed