Homework 6

1. Code to randomly generate data:

N = np.random.uniform(low = -0.5, high = 0.5, size = (100, 3))

1. Code to assign labels using XOR:

w1 = np.transpose([0, 1, -1])

w2 = np.transpose([0, 1, 1])

for pair in N:

    h1 = np.sign(np.dot(w1 , pair))

    h2 = np.sign(np.dot(w2 , pair))

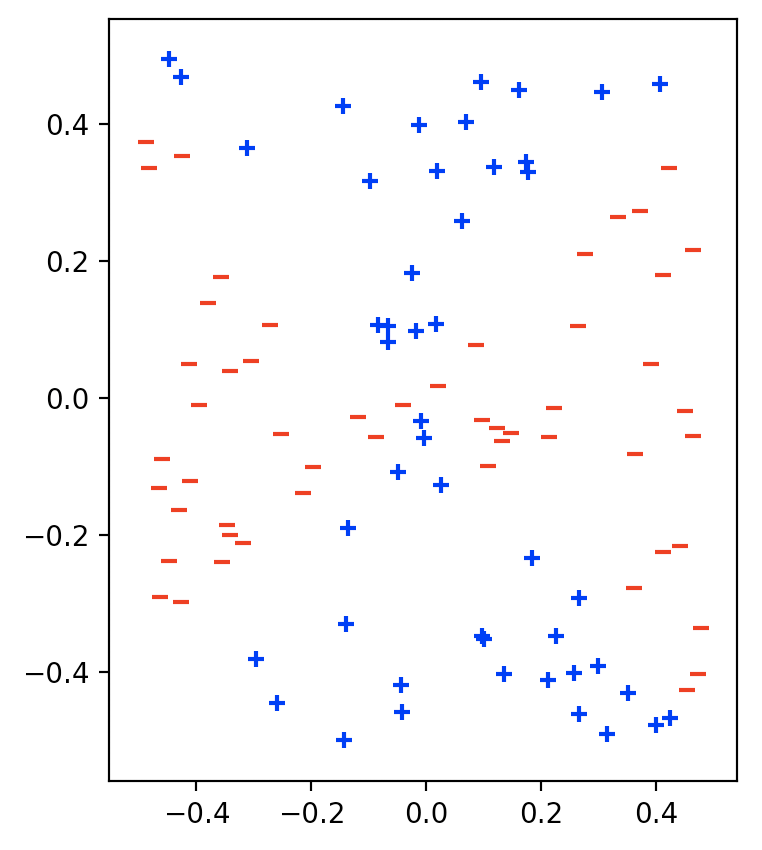
    if (h1>0 and h2<0) or (h1<0 and h2>0):

        pair[0] = 1

    else:

        pair[0] = -1

When graphed the results look like this:



1. Code for MLP:

w3 = np.transpose([-1.5, 1, -1])

w4 = np.transpose([-1.5, -1, 1])

w5 = np.transpose([1.5, 1, 1])

for lst in M:

    h11 = np.sign(np.dot(w1 , lst))

    h12 = np.sign(np.dot(w2 , lst))

    h1 = [1, h11, h12]

    h21 = np.sign(np.dot(w3 ,h1))

    h22 = np.sign(np.dot(w4, h1))

    h2 = [1, h21, h22]

    h3 = np.sign(np.dot(w5, h2))

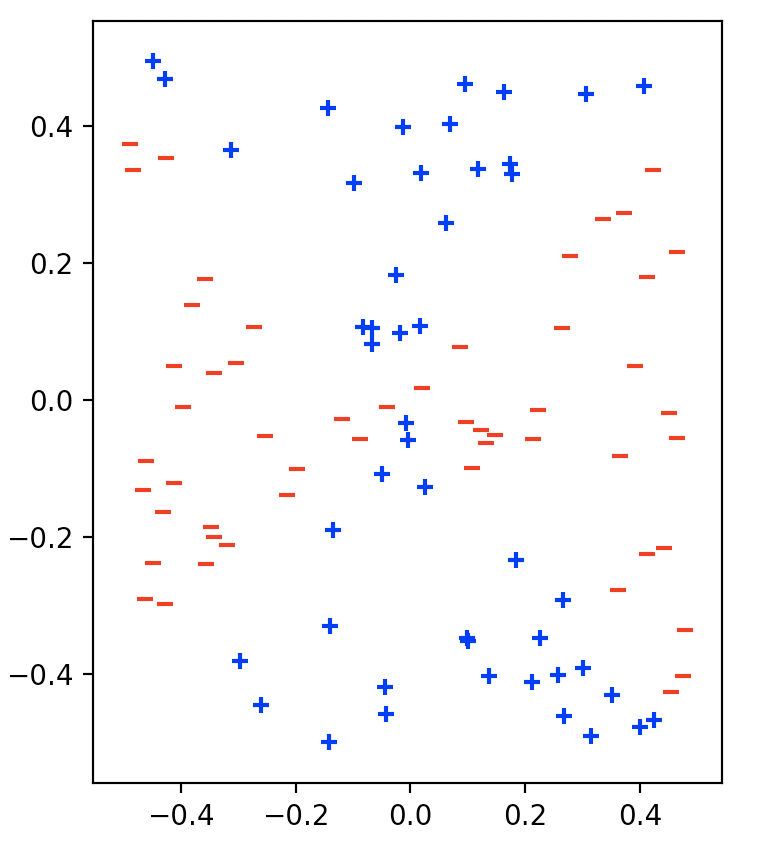
    if h3 == -1:

        lst[0] = -1

    else:

        lst[0] = 1

1. When the MLP is used to predict the labels, there is 0 error. The graph below shows the predicted labels using MLP:



Full Code for the Problem

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

fig, (fig1, fig2) = plt.subplots(1, 2)

# randomly generate data

N = np.random.uniform(low = -0.5, high = 0.5, size = (100, 3))

**for** point **in** N:

    point[0] = 1

M = np.copy(N)

#weights

w1 = np.transpose([0, 1, -1])

w2 = np.transpose([0, 1, 1])

# assign true labels using XOR

**for** pair **in** N:

    h1 = np.sign(np.dot(w1 , pair))

    h2 = np.sign(np.dot(w2 , pair))

**if** (h1>0 **and** h2<0) **or** (h1<0 **and** h2>0):

        pair[0] = 1

        fig1.scatter(pair[1], pair[2], marker='+', color = 'b')

**else**:

        pair[0] = -1

        fig1.scatter(pair[1], pair[2], marker='\_', color = 'r')

# new weights

w3 = np.transpose([-1.5, 1, -1])

w4 = np.transpose([-1.5, -1, 1])

w5 = np.transpose([1.5, 1, 1])

# mlp

**for** lst **in** M:

    h11 = np.sign(np.dot(w1 , lst))

    h12 = np.sign(np.dot(w2 , lst))

    h1 = [1, h11, h12]

    h21 = np.sign(np.dot(w3 ,h1))

    h22 = np.sign(np.dot(w4, h1))

    h2 = [1, h21, h22]

    h3 = np.sign(np.dot(w5, h2))

**if** h3 == -1:

        lst[0] = -1

        fig2.scatter(lst[1], lst[2], marker='\_', color = 'r')

**else**:

        lst[0] = 1

        fig2.scatter(lst[1], lst[2], marker='+', color = 'b')

# calculate error

miss = 0

**for** num **in** range(100):

    first = N[num][0]

    second = M[num][0]

**if** first != second:

        miss += 1

error = miss/100

print(error)

plt.show()