Sigmoid function:

NOT-Gate:

AND-Gate:

OR-Gate:

#!/usr/local/bin/python3

import math, sys, random

mode = sys.argv[1].lower() if len(sys.argv) > 1 else "not"

if mode == "not":

training = {1: 0, #NOT

0: 1}

elif mode == "and":

training = {0: 0, #AND

1: 0,

2: 1}

else:

training = {0: 0, #OR

1: 1,

2: 1}

w, b = 0.5, 0.5

e = math.e

def neuron(x1, w, b):

return 1/(1+e\*\*-(w\*x1+b))

def maxError(w, b):

return max([v - neuron(k, w, b) for k,v in training.items()])

def mse(w, b): #minimize this

return 1/len(training)\*(sum([v - neuron(k, w, b) for k,v in training.items()])\*\*2)

def vary(w, b):

return [(w+random.uniform(-0.5, 0.5), b+random.uniform(-0.5, 0.5)) for i in range(50)]

def optimize(w, b):

while maxError(w,b) > 0.1:

w,b=0.5,0.5

while mse(w, b) > 10\*\*-15:

error = {i: mse(\*i) for i in vary(w, b)}

w, b = min(error, key=error.get)

return (w, b)

def display(w, b):

print("w: {}".format(w))

print("b: {}".format(b))

for i in training.keys():

print("{}: {}".format(i, neuron(i, w, b)))

print("MSE: {}\n".format(mse(w, b)))

display(w, b)

display(\*optimize(w, b))