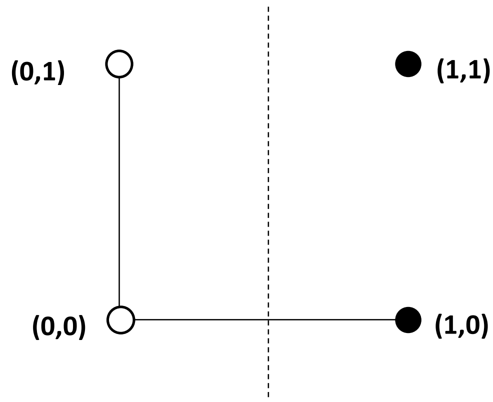
Problem

Given the following dataset and weight matrix, calculate a new set of weights based on the error produced by presenting pattern no. 1 to the network.

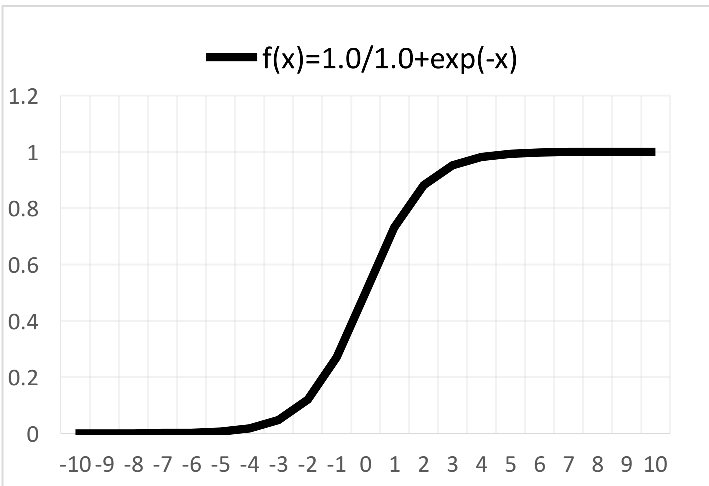
No.	Inputs			Outputs
1	1	0	1	1
2	1	1	1	1
3	0	0	1	0
4	0	1	1	0

Weights		
0.15	0.35	-0.65

Looking at the inputs we can see that only the first two columns differ. We can plot these values to see if the problem in linearly separable.



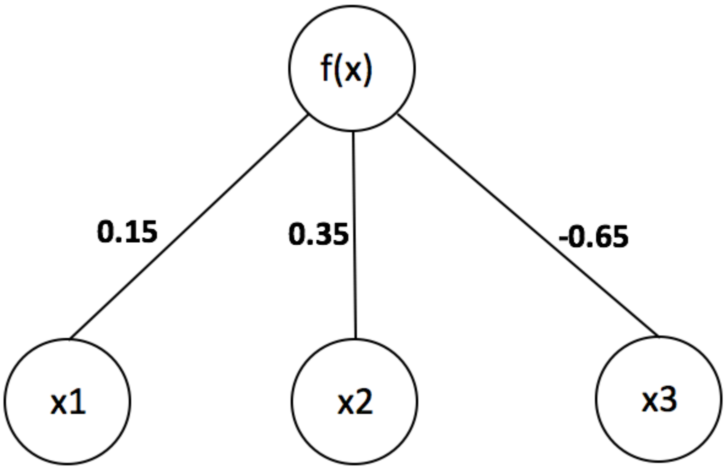
By graphing the inputs for this problem we can see that the problem is **linearly separable**. Therefore, we can use a single layer network to solve the problem. Looking a the desired outputs we can see that they are split into two categories $(0, 1)$. Therefore, we can use a single network output that should match the desired output for each input pattern. Since the outputs are $(0,1)$ we should use a transfer function that will output values within the desired range. The ***sigmoid*** transfer function takes real values and squashes them into the range $(0\text{-}1)$.



NOTE: $\exp(x)$ Returns e raised to the power of x. The constant e equals ≈ 2.7182, the base of the natural logarithm also known as *"Euler's number"*

.

So our network will have three inputs and a single output that uses the sigmoid function. The diagram below represents the network structure.



Training the Network

In order to train the network we must present each pattern (forward pass) to the network in order to determine its output. This is done by calculating the weighted sum of inputs for each input pattern and passing them through the sigmoid function. We can do this by multiplying each input by its connecting weight and summing the values.

$$output(i) = f\left(\left(x1^i * w1\right) + \left(x2^i * w2\right) + \left(x3^i * w3\right)\right), where\ 1 \leq i \leq 4$$

Let output(1) = the output for the first input pattern:

output(1) = f((1*0.15) + (0*0.35) + (1*-0.65))
= f(0.15 + 0 + -0.65)
= f(-0.5)
= **0.38**

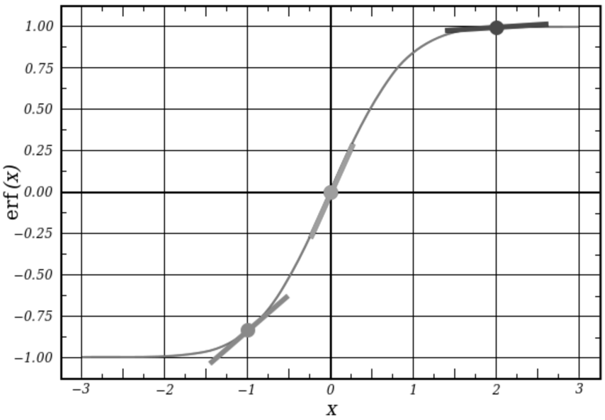
NOTE: You can use you calculator to compute *f(-0.5)* or you can use the sigmoid graph attached by marking *-0.5* on the x axis and looking at the y value where *-0.5* on the x axis intersects the curve.

Next we need to calculate the error that this input produced in order to adjust the network weights.

Let *error(1)* = the error produced by the first input pattern

error(1) = desired(1) – output(1)
= 1 – 0.38
= **0.62**

Next we need to calculate a derivative for *output(1)*. In **mathematics**, the **derivative** is a way to represent rate of change, that is - the amount by which a **function** is changing at one given point. For functions that act on the real numbers, it is the **slope** of the tangent line at a point on a graph.



As it turns out the derivative calculation for the sigmoid function is relatively easy. We simply take a point on the curve *(x)*, and use ***x * (1 – x)***. ***x*** in this case will be the output value for the current pattern. Remember *output(1)* was produced using the sigmoid function.

derivative(1) = x * (1 – x)
= output(1) * (1 – output(1))
= 0.38 * (1 – 0.38)
= 0.24

Now that we know the rate of the change for our output value we can use it to calculate a ***delta*** (change) by multiplying our error value by the derivative.

delta(1) = error(1) * derivative(1)
= 0.62 * 0.24
= **0.15**

We now have all the information we need in order to update each weight based on a simplified version of the **delta rule**.

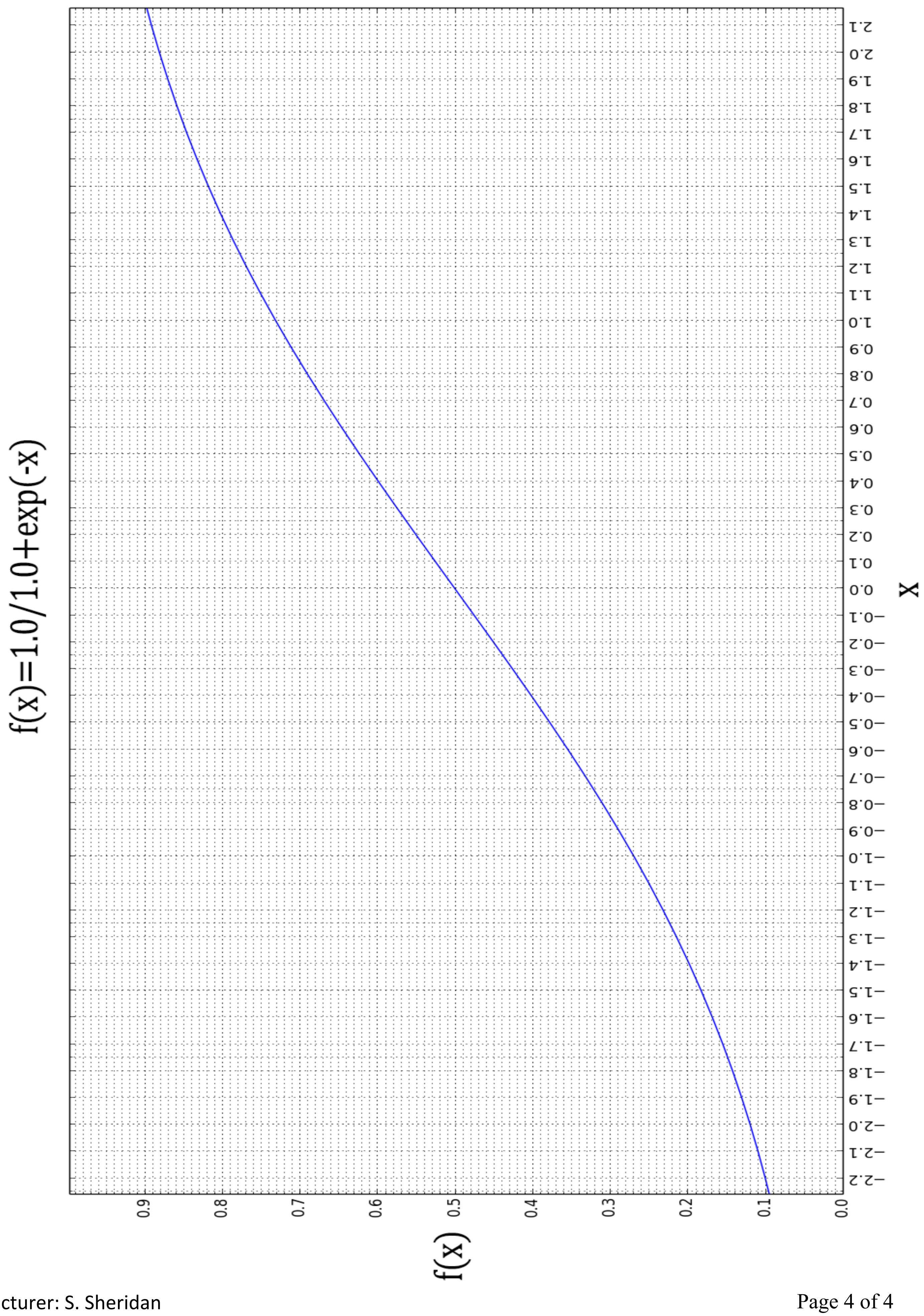
new_weight = current_weight + (delta * input_on_weight)

Let *wi(t)* represent the current weight values, *wi(t+1)* represent the new weight values and *xi* represent the input on the current weight. So the calculations to update the weights based on the first input pattern are as follows:

w1(t+1) = w1(t) + (delta * x1)
= 0.15 + (0.15 * 1)
= **0.30**

W2(t+1) = w2(t) + (delta * x2)
= 0.35 + (0.15 * 0)
= **0.35**

W3(t+1) = w3(t) + (delta * x3)
= -0.65 + (0.15 * 1)
= **-0.50**


f(x)=1.0/1.0+exp(-x)
x
f(x)
−2.2 −2.1 −2.0 −1.9 −1.8 −1.7 −1.6 −1.5 −1.4 −1.3 −1.2 −1.1 −1.0 −0.9 −0.8 −0.7 −0.6 −0.5 −0.4 −0.3 −0.2 −0.1 0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9