ECN-252 Lab 7

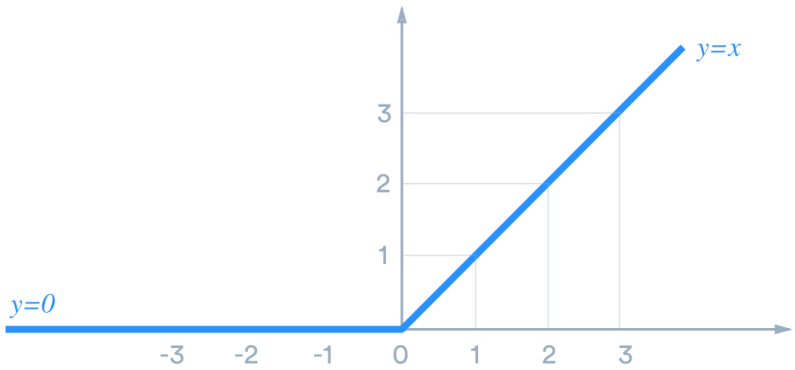
For questions: contact Dr Sparsh (sparshfec@iitr.ac.in)

Hint: Use 2's complement number representation for all the numbers. You can use the adder/multiplier/comparator from the previous assignments and use them as building blocks. You don't have to test them again or show their outputs.

Part 1 (ReLU function)

i) We need to detect whether a number is positive or not. You can use a comparator for this (hint: in fact it is much simpler than that!). The input number is 4-bit. **NOTE: you don't need to submit a circuit for this. Just write the Boolean logic or reasoning for achieving this functionality (max 2 sentence).**

ii) The following function is called ReLU and it is shown in the figure below.
y = max (0,x).
Although simplistic, it is used heavily in neural network. Implement a circuit to achieve this function. The input x is a 4-bit number,



Part 2 (Training an overly simplistic perceptron):

Consider the training algorithm of a perceptron

```
for each 0 ≤ i ≤ n in parallel
    if t = x_i then
        w_i := w_i + 1
    else
        w_i := w_i − 1
    end if
end for
```

Here, for sake of simplicity, we will assume that n=1 and instead of 0, the value of i begins with 0. In other words, there is only one perceptron. Thus, there is only one value of x, one value of w and one value of t.

Design a circuit to implement the training algorithm. The weight (w) is 4-bit and is fixed to be 0010. The input (x) needs only 2-bit because its value can be either +1 or -1. The output (t) is also 2-bit and its value can be either +1 or -1. As you can see, for 2 possible values of x and 2 possible values of t, we have a total of 4 combinations to evaluate (weight is already fixed). Thus, we do NOT have 16 possible combinations, but just 4 combinations.

Hint: A very important point to note here is that although x is using 2-bit for its storage, it can have only 2 distinct values. Thus, the remaining 2 combinations are not used. The same is also true for t.
Hint: In the comparator, we only need to check whether t is equal to x or not. Hence, for both t>x or t<x, the action taken is same.

Part 3 (Doing inference with an overly simplistic perceptron):
Design a small circuit to implement a simple pattern recognition where given two co-ordinates (x, y) as inputs to the circuit, the circuit will tell if the given point is above or below a straight line. Straight line equation to be used is given in the table. Show your results for two test points – above and below the line. Choose points such that you don't exceed the 3 bit (for the multiplier) or 7 bit (for the sum) integer limit. (If you can't remember how to check this, see note below the table)
(This exercise is actually showing an over-simplified example of a neural network or a perceptron. Of-course, we don't really need a neural network for this, however this shows how more complex networks may be realized using simple multiplication and addition as their building blocks. Adapted from the tutorial- https://natureofcode.com/book/chapter-10-neural-networks/)
There are 8 sets of problem (4 for ECE, 4 for CSE). To select within set 1 - 4, use mod (Roll No.,4)+1 and select the corresponding problem from the table below.
[For example, if your roll no. is 191112002, you should select mod(191112002,4)+1 = 3; i.e. set 3.]

Group	Straight line to be used for Part 3
ECE, Set 1	2x-3y-6=0
ECE, Set 2	4x+3y=0
ECE, Set 3	x-2y=0
ECE, Set 4	2x+y+2=0
CSE, Set 1	x+3y-3=0
CSE, Set 2	2x-4y-4=0
CSE, Set 3	x-y=0
CSE, Set 4	x+y-1=0

How to check if a point is above or below a given line?

Given any arbitrary line ax+by+c=0, a point (x1, y1) is above the line if,

(a) ***b>0 and ax1+by1+c>0*** OR (b) ***b<0 and ax1+by1+c<0***