Neural Networks

Prasann Viswanathan

June 2, 2020

Abstract

This document is to navigate you through my thought process and learning curve as I attempt to learn Neural Networks from scratch through simple assignments.

1 Basic Perceptron

Problem Statement
Construct a neural network taking three inputs (either 1 or 0) and a giving an output (either 1 or 0)

Input matrix $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix}$ Output Matrix $\begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

Intuitively we can see that the output depends only on the first input. i.e if $A_{i1} = 1, O = 1$
I referred [1] for the math and algorithm. The following is the code(I've included it because it was short and simple)

```
import numpy as np


#Using sigmoid because mainstream is bae
```

```
def f(x):
    return 1/(1+np.exp(-x))
def der(t):
    return (f(t))*(1-f(t))

ip = np.array([[1,0,1],[0,0,1],[1,1,0],[1,1,1]])

op = np.array([[1,0,1,1]]).T

w = np.random.random((3, 1))

#Assuming 1000 iterations is enough to converge accurately
for i in range(1000):

    input = ip
    output = f(np.dot(input, w))
    revamp = (op-output) * der(output)
    w += np.dot(input.T, revamp)

print(f"Results after training \n{output}")
```

On running this code the following conclusions were drawn

1. Greater number of iterations results in a greater accuracy in the final output.
2. However after a point, the accuracy begins to saturate
3. The resultant weight matrix is $\begin{bmatrix} 9.08 \\ 2.38 \\ -4.41 \end{bmatrix}$ Which I could not explain, I thought it may be a random value set but on multiple runs the values were very similar. (Any insight would be appreciated)

2 XOR Gate using Neural Networks

Problem Statement
For those who don't know, XOR takes in two inputs with values 1 or 0. Depending on which we get the outputs if the input's are **exclusively or**. That is, $\begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$ gives $\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

The architecture used is:

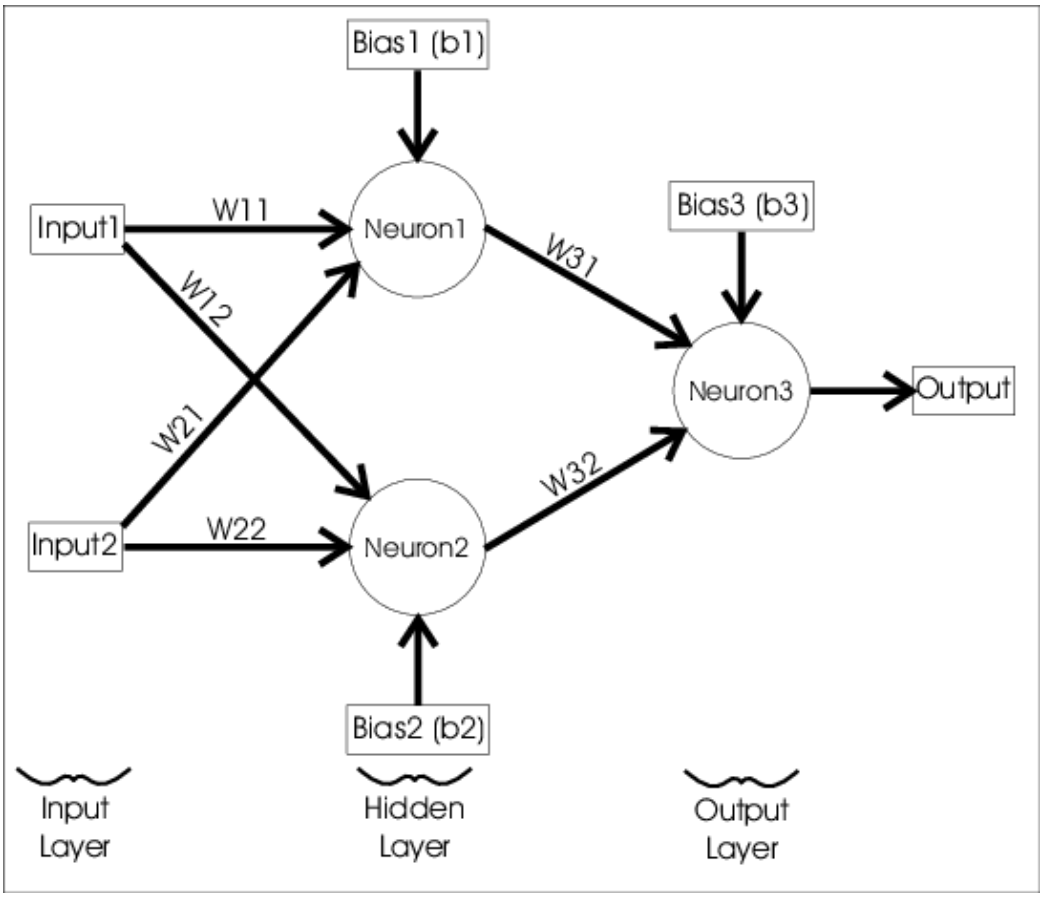

Figure 1: Architecture of the Neural Network

The problem could be split into the **Feed Forward step** and the **Back-**

propagation step. All of it uses extremely complicated math on face level; However, watch the 3Blue1Brown series [2] on it, and you'll realize it's a cake-walk.

2.1 Feed Forward Algorithm

I'd like to take some time to walk you through the math (Math in LaTeX isn't nearly as easy as it is made out to be so bear with me for a bit.)
Let the inputs be x_1 and x_2. On moving from the input layer to the hidden layer, the following are done.

$$\begin{bmatrix} x_1 & x_2 \end{bmatrix} \times \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \end{bmatrix} = \begin{bmatrix} h_1 & h_2 \end{bmatrix}$$

(I seriously can't be arsed into doing all the math here because frankly this is taking forever. Refer [3], [4] etc to learn more.)

2.2 Back-Propagation

Once again, not gonna do the math (Imagine writing ∂ everywhere ffs) Just refer the code provided here [5]

2.3 Why One Perceptron Won't Work

The XOR problem is not linearly separable. That is, if you plot along the x_1 and the x_2, axes, the cases where output is 1 and where it's 0, it's impossible to draw a line separating the one's and zero's. Hence the single perceptron always gives output 0.5, irrespective of the test case. [6]

3 Conclusion and Bugs

The only conclusion I drew is LaTeX SUCKS. It takes forever and gets super frustrating. I've given up on typing out the math because 1. It's too cumbersome and 2. I've linked all the sources with all the math you'd need.
Anyways on a serious note, watching all those videos and reading the articles made this crystal clear. It's just MA106 and MA105. In fact much much

easier.

Bugs

1. Well, as I said, LaTeX.
2. For some reason I always get the tabs and spaces error in python and then it takes me forever to find where I've accidentally added an extra space. Seriously Python, up your game, dude.
3. I kept using matrix multiply instead of dot. (Isn't dot supposed to be inner product if one goes by the name? smh)
4. I keep wondering if sigmoid is the ideal way to go. Tushar did spark some doubt in me when he used tanhx (He's probably on to something). I did tinker around with other functions but stuck with the classic in the end. Maybe you could provide some feedback.

Fin.

References

[1] https://towardsdatascience.com/perceptron-learning-algorithm-d5db0deab975

[2] https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

[3] https://towardsdatascience.com/implementing-the-xor-gate-using-backpropagation-

[4] https://www.youtube.com/playlist?list=PLRqwX-V7Uu6aCibgK1PTWWu9by6XFdCfh

[5] https://github.com/iamprasann/IntelligentAgentsSoC/tree/master/Neural%20Network

[6] https://github.com/iamprasann/IntelligentAgentsSoC/blob/master/Neural%20Network