Assignment # 2

Amanda Ward
Department of Computer Science
Ward4006@vandals.uidaho.edu

***Abstract*— In this assignment, we analyize the XOR neural network and what it means to be linearly separable. We apply the percetron delta rule in two different ways. The first method will be to attept to train the weights all at the same time. First it will be attempted will all weights initialized to zero, then it will be attempted with the wieghts initialized radomly. The second method entails creating an XOR neural network as a composite of an OR, NAND, and AND function. Each of the the smaller units are trained individually.**

***Keywords* — XOR, perceptron delta rule, convergence, linearly seperable.**

I. INTRODUCTION AND BACKGROUND

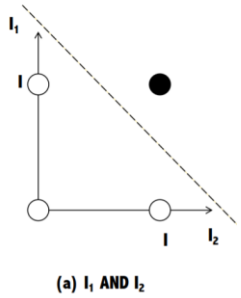
The concept being explored in this assignment is linear seperability and how that effects the application of the perceptron delta rule. One of the easiest ways to understand and demonstrate linear separability is with a 2D visual. First, make a table for the inputs and outputs for whatever perceptron the perceptron delta rule is bieng applied to. This will indicate how to graph the function on an xy corrdinate plane. Figure 1 shows the table for an AND.

Figure 1. AND input and output table

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

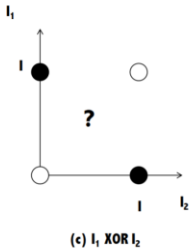
The graph for the table is shown in Figure 2. In the figure, one input becomes the x-axis and the other input becomes the y-axis. The white dots represent an output of one and the black dot represents an output of zero. It becomes clear to see that one could draw a single, strait line separating the outputs of one and the output of zero.

Figure 2. Graph representation of AND



It was demonstrated in Assignment # 1 that the delta perceptron rule could be applied to liearly separable perceptrons such as AND and OR. The question being explored here is, how do you apply the derceptron delta rule to a perceptron that is not linearly separable such as the XOR perceptron? Figure 3 shows the graph repesentaion of the XOR perceptron.

Figure 3. Graph representation of XOR



Looking at the the graph, it becomes apparent that one can not draw a single strait line that separates the outputs of zeros and ones.

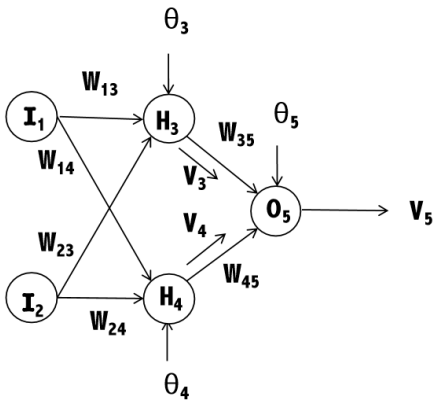
II. SOLUTION (METHOD) APPROACH(S)

There are two separate approaches to consider when analyzing how to apply the perceptron delta rule to the XOR neural network. The first implementation will be to attempt to train all the weights simultaneously. The second approach will be to build an XOR perceptron out of smaller functions that we know are linearly separable and train the smaller pieces separately.

III. IMPLEMENTATION (DESIGN)

The first method being implemented will be to train all of the weights at once. This particular neural network architecture is a three layer perceptron network. There are two input fields, six wieghts and three bias. The architecture is depicted in Figure 4. To test the implimentation, the weights will be selected two ways 1) weights will all be initialized to zero 2) weights will be randomly selected. The goal will be for the network to converge and record final wieghts. If the the network converges, the number of epochs and final weights will be reported. A link to the downloadable code used to test will be made available in the appendix.

Figure 4. Architecture of XOR



The first step is to create a table showing the inputs and outputs for XOR. This table is shown in Figure 5. The table will contain the desired outputs so that delta between the desired output and actual output can be measured.

Figure 5. Table of inputs and outputs for XOR

Input1	Input2	Output
0	0	0
0	1	1
1	0	1
1	1	0

There are three learning rates that will be utilized per the assignmenet specs. The learning rates are as follows: 0.25,0.50, and 0.75.Once the desired outputs and the learning rates are determined, the process for training is straightforward. The inputs and their acompanying wieghts are summed as sholwn in figure 6. Y represents the output, b is the bias, x is the input and w is the wieghts.

Figure 6. equation for summation

$$y = b + \sum_i x_i w_i$$

Once summed, the output is compared to a threshold shown in Figure 7, assigning a value of either zero or one. To find the error rate, the actual put is subtracted from the expected output.

Figure 7. Threshold for the summation

$$V_3 = \begin{cases} 0 : y_3 < 0 \\ 1 : y_3 \geq 0 \end{cases}$$

If the result is zero, then no changes are made. Other wise the weights will need to be updated. Updating the weights is depicted in Figure 8.

Figure 8. Process for updating weights

$$\Delta W_{ij} = \varepsilon \cdot \delta_i \cdot V_i$$
$$\delta_i = (t_i - V_{out(i)})$$

where

$0 \leq \varepsilon < 1$ **is the learning rate**

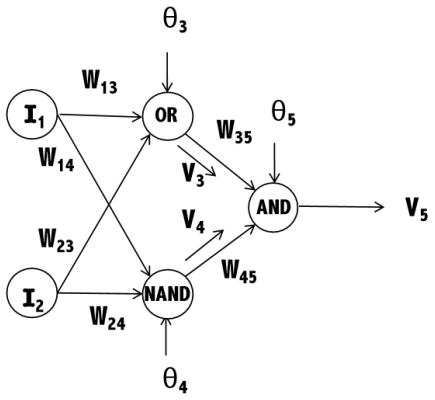
δ_i **is the error term**

V_i **is the input activation**

$$W_{ij} = W_{ij} + \Delta W_{ij}$$

The second method is to create the XOR neural Network out of three neural neworks, an OR, AND, and NAND network. The three functions named are linearly separable. The three networks would be trained separately using the same process as described in the previous method. The OR function will be trained to determine W13 and W23. The NAND function will be trained to determine W14 and W24. Lastly, The AND function to determine W35 and W45. The goal with the second method is the same as the first, the actual output and the desired output need to match in order for the method to be considered sucessful. Link to the code used is available in the Appendix.

Figure 5. Three layer peceptron model of XOR composed of an OR, NAND, and AND.



IV. EXPERIMENTAL RESULTS AND ANALYSIS

The fist method explored was not successful. It did not matter if the weights had been initialized to zero or if they had been randomly selected. The results were the same, non-convergence. The experiment was terminated after 100,000 epochs.

The second method was successful. The OR, NAND, and AND were trained separately with the OR and NAND feeding into the AND. With this method the actual output matches the desired output.

V. Conclusion

Because the XOR perceptron is not linearly separable, the perceptron delta rule cannot be used to train all the weights at once. However, if the XOR network in composed of smaller functions that are known to be linearly separable, the perception delta rule can be applied to the individual components that the XOR gate is composed with and train the weights that way. This supports what Dr. Hiromoto stated in his Neural Networks class that all logical functions can be implemented with a network made entirely of AND, OR and NOT functions [1].

VI. References

Hiromoto, R. (2020). Lecture 3

VII. Appendix

https://github.com/AmandaLW/UI_Neural_Networks.git