Assignment 2

Ilana Zane, Jennifer Rodriguez, Jie Wang, Zain Ul-Abdin

December 2, 2020

Contributions:
Jie Wang: Coded exercise 1
Ilana Zane: Coded exercise 1 and 2, answered questions
Jennifer Rodriguez: Answered exercise question
Zain Ul-Abdin: Answered exercise question

1 Logic Gate

1.1

Temporal encoding is defined as the time to fire the first spike. In this case x and y are encoded as time delay. For example we can have 2 neurons, the first neuron fed with x takes 2ms to emit its first spike, and second neuron fed with y takes 5ms to emit its first spike.

1.2

Rate encoding is defined as the total amount of spikes which occur within a given time frame. In this case x and y are encoded as spike train. For example we have 2 neurons, the first neuron fed with x generates 20 spikes over 300 msecs, whereas the second neuron fed with y generates 16 spikes over 300 msecs.

1.3

Our network consists of two input neurons, one for input x and one for input y, which take in the values of either 1 or 0. Our hidden layer consists of two weights that map to one output neuron that will either output a 0 or 1.

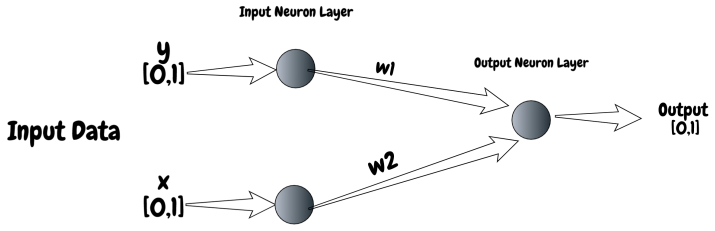

Figure 1: Network Architecture of Logic Operators

- We used a rate based encoding scheme, where an input of 1 represents 10A current that can fire an Izhikevich neuron in the input neuron layer. An input of 0 represents 0A current that cannot fire an Izhikevich neuron (silent) in the input neuron layer. Similarly, in our decoding scheme, if the Izhikevich neuron on the output neuron layer fire spikes, then that neuron represents a 1 and if the output Izhikevich neuron is silent then the result is 0.

- We used the perceptron learning rule, where the perceptron is a supervised learning algorithm for binary classifiers. The learning rule is as follows:

$$f(x) = \begin{cases} 1 & if\ \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & otherwise \end{cases}$$

where **w** is a vector of weights and **x** is a vector of input data values. The final value is the summation of the dot product between weights and data plus the bias. Ideally, we want the final weights to be equal to 1 and the bias to be equal to -1. For example, when the weights are equal to 1, bias is -1 and the input values are (1,1), we will have the equation $(1*1)+(1*1)-1>0$ and if we have the input (0,0) with the same finalized weights we would have the equation $(1*0)+(1*0)-1<0$. With enough training our model is able to obtain these finalized weights and construct an accurate logic gate.

- In order to train the model the weights and bias are initialized to 0. We then run our input values through the perceptron learning rule to obtain an output. Since this is a supervised learning algorithm, we compare the models output to the corresponding correct output. We use the following equation to update our weights:

$$\mathbf{w} = \mathbf{w} + \alpha(y - f(x))\mathbf{x}$$

where w is our weights, alpha is some learning rate, y is the expected answer, f(x) is the resulting answer from our model and x is our input data values. If the models results match the expected answer, the above equation results to 0 and the weight is not updated, otherwise the weight is updated by some real value.

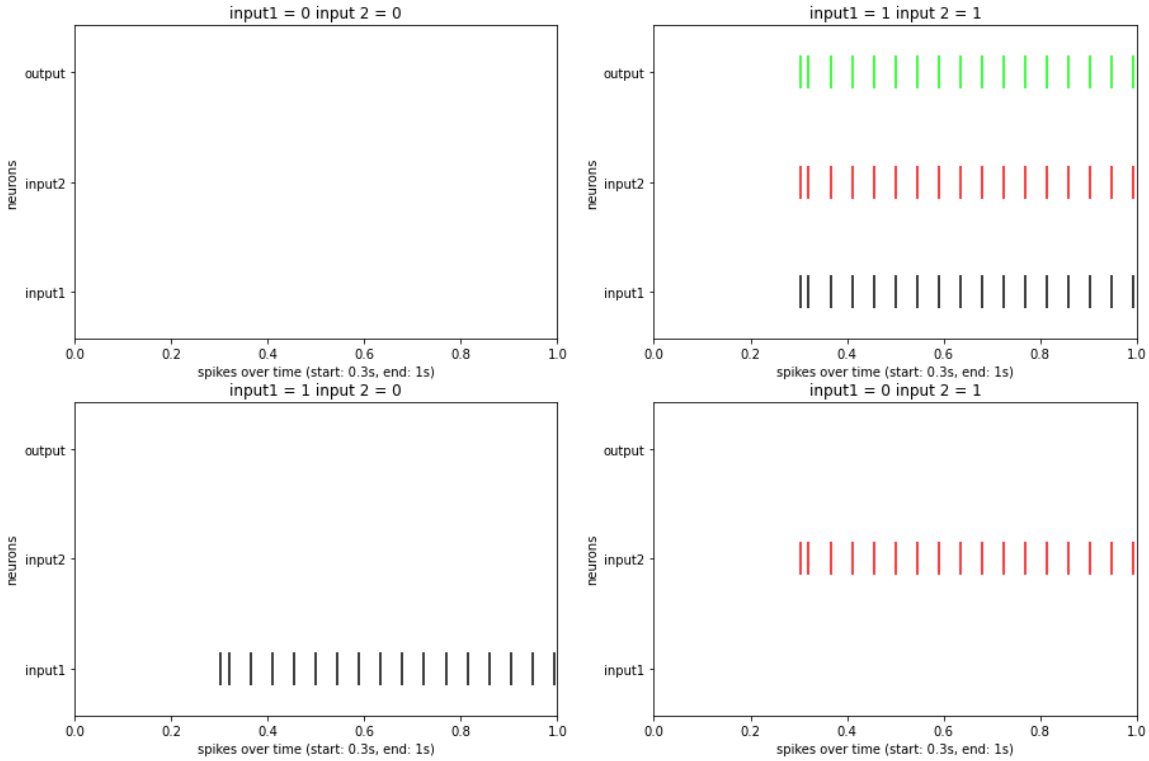

Figure 2: Raster Plot: AND function

As shown in Figure 2, when inputs were [0,0], both input neurons were silent so the output neuron was silent as well. When inputs were [1,1], each input neuron fired 17 spikes over time in response to a 10A input current so that the output neuron fired as well. When inputs were [0,1] or [1,0], one input neuron fired and one was silent (marked as 1 if fired, otherwise 0) so that the output neuron was silent following the perceptron rule.

1.4

Yes, our model can learn the OR function.

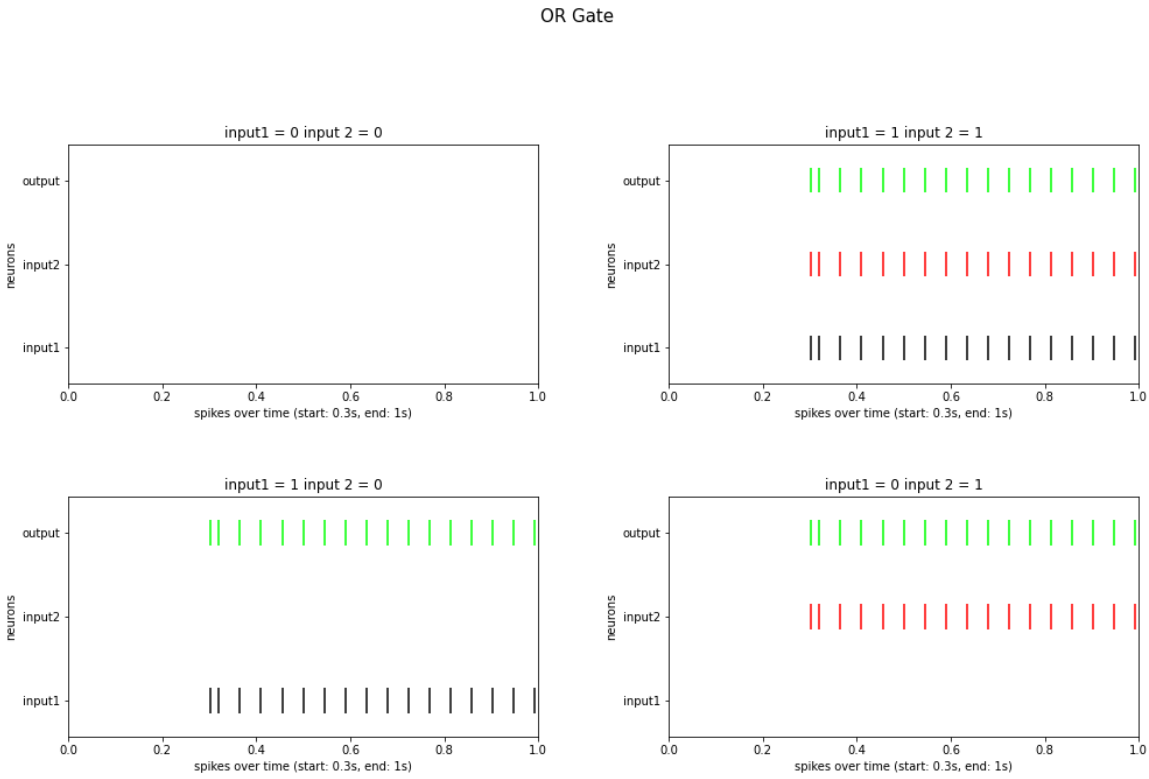

Figure 3: Raster Plot: OR function

1.5

No, our model cannot learn the XOR function.

x	y	XOR
0	0	0
1	0	1
0	1	1
1	1	0

In the table we have a XOR function. We will plug in the x and y values into the perceptron rule for each row in the table, starting with the first row. We assume that w_1 and w_2 are both equal to 1.

$$0+0+b \leq 0 \Rightarrow b \leq 0$$
$$w_1+0+b > 0 \Rightarrow w_1 > -b$$
$$0+w_2+b > 0 \Rightarrow w_2 > -b$$
$$w_1+w_2+b \leq 0 \Rightarrow w_1+w_2 \leq -b$$

We see that from equations 2 and 3 that both individual weight values must be greater than -b, but in the last equation we see that the sum of both weights must be less than or equal to -b. Since we arrive at this contradiction, we know that it will be impossible for our model to learn the XOR function with the perceptron rule.

2 Digit Classification

2.1

- The provided images are of size 8x8, meaning that there are a total of 64 pixels, each one ranging in values between 0 and 16 and representing a current. Therefore, we have 64 inputs neurons.
- Since there are 10 possible values (0-9) for our model to classify, if it were to choose at random then our "chance level" would be $\frac{1}{10}$.
- If our network only classifies digits 1,2 and 5, we will have three output neurons because the output is the final classification of each digit, so for every digit we have 1 output neuron. Thus, if we train our network to classify all 10 digits, we will have 10 output neurons.
- We didn't change our encoding process. For both problems we used rate encoding.

2.2

When classifying digits 1 and 8 the model has a test accuracy of 1.0. The model also has a test accuracy of 1.0 when classifying digits 3 and 8. We expected the model to classify 1 and 8 with high accuracy, but did not expect it to classify 3 and 8 just as well. However, the model in general is very robust and performs well for all digits. If the model were not as robust, then it would probably show a lower validation and test accuracy when classifying 3 and 8, because the two numbers have the same curved shape on their right side. The numbers 1 and 8 do not share any similar shapes which would probably lead to a higher accuracy. Our model probably recognizes that 3 and 8 share the same shape on the right side, but the left sides of the numbers are very different; where 8 curves in on itself, the number 3 does not.

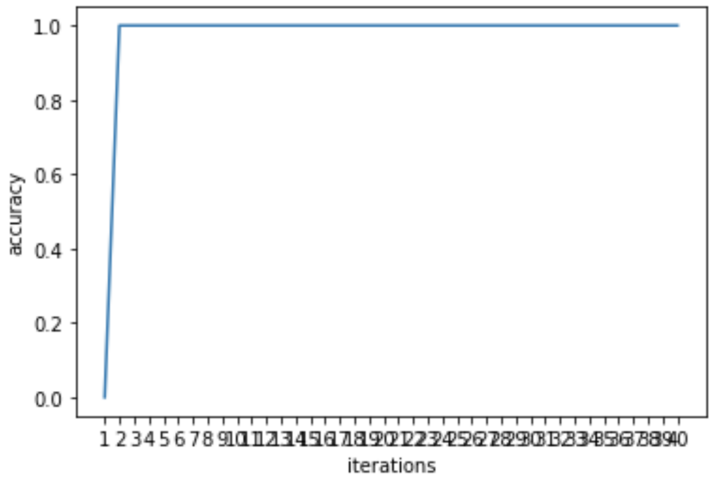

Figure 4: validation accuracy for 1 and 8

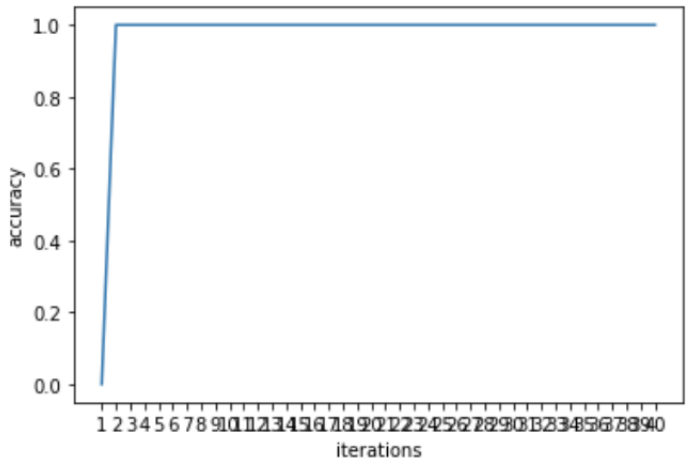

Figure 5: validation accuracy for 3 and 8

2.3

When classifying all ten digits for 40 iterations and with a learning rate of 0.01, our model has a general test accuracy of 85%. This number is far than the ”chance level” which is 10%.

```
test accuracy: 0.8397790055248618
```

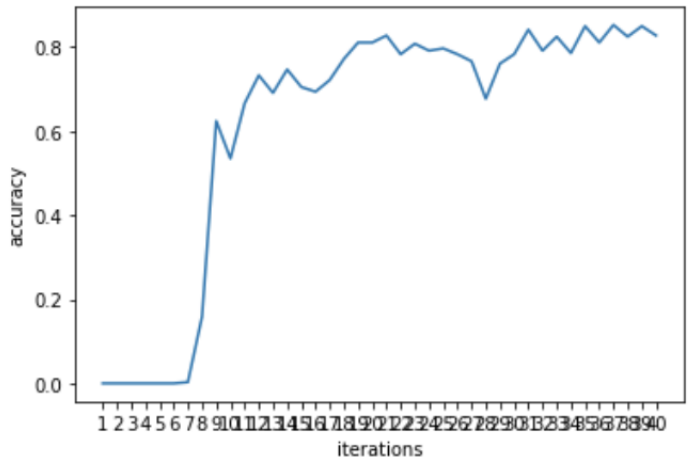

Figure 6: validation accuracy for all ten digits

References

[1] Rosenblatt, F. The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain. 1958, citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.335.3398&rep=rep1&type=pdf.