THE IMPORTANCE OF INHIBITORY NEURONS IN HEBBIAN LEARNING

Rafi Qumsieh
rafiqumsieh@gmail.com

June 7, 2020

Abstract

A biologically-plausible pair of excitatory neurons that are equipped with Oja's version of Hebbian learning can form the AND, OR gates, but cannot form an XOR gate. An additional inhibitory neuron is needed in the configuration to form an XOR gate. By biologically-plausible, it is meant that the neurons use Hebb's rule for learning, they do not alter their type between excitatory and inhibitory during learning, and they have the same constant bias. In this paper, we provide evidence to show that adding an inhibitory neuron in a certain configuration can create an XOR gate.

1 INTRODUCTION

Hebbian learning is a model introduced by famous psychologist Donald Hebb that is used to explain how neurons learn patterns. The model suggests that a neuron will increase its synaptic connection strength to another neuron if it successfully and repeatedly causes it to fire. As a famous mantra goes: " Neurons that fire together, wire together. "

When we describe this statement mathematically, we adjust the value of the weight that is associated between two neurons n_1 and n_2 based on their firings. We say that the weight change Δw is proportional to the product of the input to the receiving neuron x with the output of the receiving neuron y. So we have:

$$\Delta w = \eta x y$$

Where η is the learning rate, which is a number that represents how rapidly we want to change the weight for each single firing of the neurons. This formula assumes that inputs, outputs, weights can have a negative value and that, if the input and output have a different sign, the change in weight will be negative, which means that the connection will weaken between the neurons that do not fire together. Some of these ideas might be hard to explain biologically. For example, what does it mean to have a negative input when thinking about neurons? When using this formula, we will restrict ourselves to $x, y \in \{0, 1\}$ or $x, y \in [0, 1]$. Another well known problem that this formula faces is the potentially infinite growth of the weight if x and y have the same sign. Having an infinite weight will ruin the model as it prevents the encoding of new information, so an adjustment was suggested by Oja to address this problem and normalize the weights. The new formula is :

$$\Delta w = \eta y(x - yw)$$

A pair of excitatory perceptrons equipped with the Hebbian learning rule can learn to produce the AND, OR gates, but they fail to produce the XOR gate. Adding an additional single inhibitory neuron makes it possible to produce an XOR gate. The inhibitory neuron can be thought of as a control unit that prevents over-activity in the network as a whole.

In this paper, It will be shown that a single inhibitory neuron along with a pair of excitatory neurons can create an XOR gate with assumptions that mimic the biologically-plausible conditions.

2 PROOF

Let us start by looking at a pair of perceptrons that are connected to an output neuron and to an inhibitory neuron.

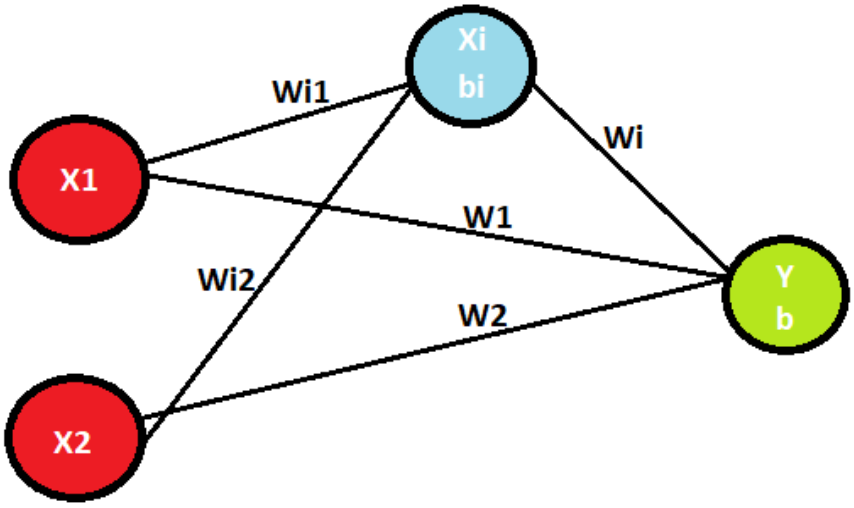

Figure 1: A pair of neurons that are connected to an output neuron and an inhibitory neuron. Red neurons are excitatory, Blue neuron is inhibitory. Green neuron is the output neuron.

We would like to map the XOR function from x_1 and x_2 to y. An XOR truth table looks like the following:

x_1/x_2 **(XOR)**	**1**	**0**
1	0	1
0	1	0

The main problem with the XOR function is that it is not linearly separable. Another way of seeing this is that in an XOR gate, we want y to output 1 when only one of the inputs is present, but output 0 if both are present. This cannot happen without a control unit to suppress the output if both inputs are present.

Mathematically, we will first use the Perceptron's step activation function for binary neurons, then we will use the ReLU activation function for inputs and outputs of real numbers. We will refer to the inhibitory neuron's properties with a subscript of i. We want our biases to be the same constant number since this is what we know about the biases from biology, so $b_i = b$. So we have:

$$X_i = Perceptron(W_i 1 x_1 + W_i 2 x_2 - b_i)$$

$$y = Perceptron(W_1 x_1 + W_2 x_2 + W_i X_i - b)$$

Where X_i is the output of the inhibitory neuron and y is the output of the final neuron. Now, let us substitute the expression for X_i from the first equation into the second one:

$$y = Perceptron(W_1 x_1 + W_2 x_2 + W_i(Perceptron(W_i 1 x_1 + W_i 2 x_2 - b_i)) - b)$$

This equation has to satisfy 4 conditions for the XOR function $y(x_1, x_2)$:

1. $y(0, 0) = 0$
2. $y(1, 0) = 1$
3. $y(0, 1) = 1$
4. $y(1, 1) = 0$

Substituting the following conditions into the equation, we get the following:

1. $0 = 0$ (Trivial)
2. $y = Perceptron(W_1 + W_i * Perceptron(W_i 1 - b) - b) = 1$
3. $y = Perceptron(W_2 + W_i * Perceptron(W_i 2 - b) - b) = 1$

4. $y = Perceptron(W_1 + W_2 + W_i * Perceptron(W_i1 + W_i2 - b) - b) = 0$

Any sets weights and biases that satisfy these equations can turn this configuration into an XOR gate. One example of such configuration is:

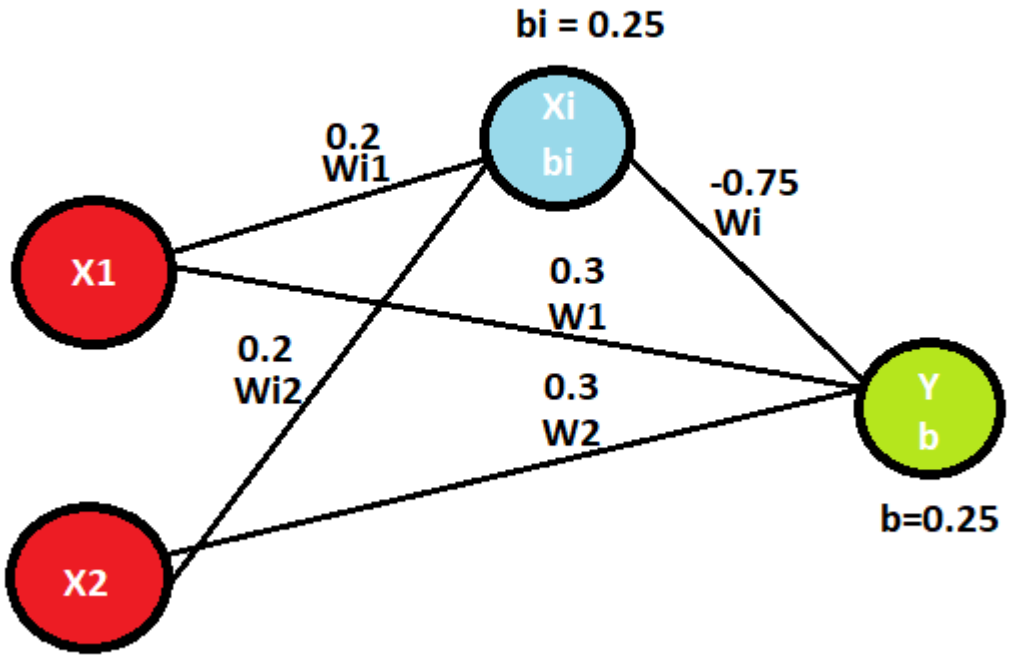

Figure 2: A pair of neurons that are connected to an output neuron and an inhibitory neuron. Red neurons are excitatory, Blue neuron is inhibitory. Green neuron is the output neuron.

This configuration can be extended to neurons that are not perceptrons. We can have neurons that accept real numbers as input, and output real

numbers too. The changes we need to make to the equations are to replace the Perceptron activation function with another real-valued function. ReLU is a good candidate since it gives us 0 if the signal does not cross the threshold for each neuron. Below are the main equations:

$$X_i = ReLU(W_i1x_1 + W_i2x_2 - b)$$

$$y = ReLU(W_1x_1 + W_2x_2 + W_iX_i - b)$$

And we can re-write y as follows:
$y = ReLU(W_1x_1 + W_2x_2 + W_i * (ReLU(W_i1x_1 + W_i2x_2 - b)) - b)$
This equation has to satisfy 4 conditions for the XOR function $y(x_1, x_2)$:

1. $y(0, 0) = 0$
2. $y(1, 0) > 0$
3. $y(0, 1) > 0$
4. $y(1, 1) = 0$

Substituting the following conditions into the equation, we get the following:

1. $0 = 0$ (Trivial)
2. $y = ReLU(W_1 + W_i * ReLU(W_i1 - b) - b) > 0$
3. $y = ReLU(W_2 + W_i * ReLU(W_i2 - b) - b) > 0$
4. $y = ReLU(W_1 + W_2 + W_i * ReLU(W_i1 + W_i2 - b) - b) = 0$

Any sets of weights and biases that solve these conditions will create a real version of an XOR gate. Where real here means that we will consider an output to be 1 if its value is larger than 0.

Keep in mind that we can also add more neurons to the first layer and still get a similar effect.

Why might this be important? The purpose of this is to try and create neural networks that are closer to biological neural networks. Biological neural networks are known to create a variety of amazing functions that we see in nature.

3 POST SCRIPTUM

This paper is intended to be a motivation to look into such configurations as basic units in future neural networks and study the behaviour and benefits of such configurations. The benefits that can be potentially gained are:

1. One immediate benefit is that using this configuration, we can produce an XOR gate using only 1 intermediate neuron as opposed to using 2 intermediate neurons with the typical setting.
2. This configuration is more biologically plausible as it differentiates between excitatory neurons and inhibitory neurons and does not allow each type to flip to the other type by changing the sign of its weights during learning.
3. The biases for all neurons have the same value which is the case for neurons in nature.