

Neural Networks

Assignment 1 - Threshold Logical Unit

George Petridis (s3001393)

Jussi Boersma (s2779153)

February 24, 2017

1 Linear Algebra

Our final graph for this section is shown below in Figure 1.

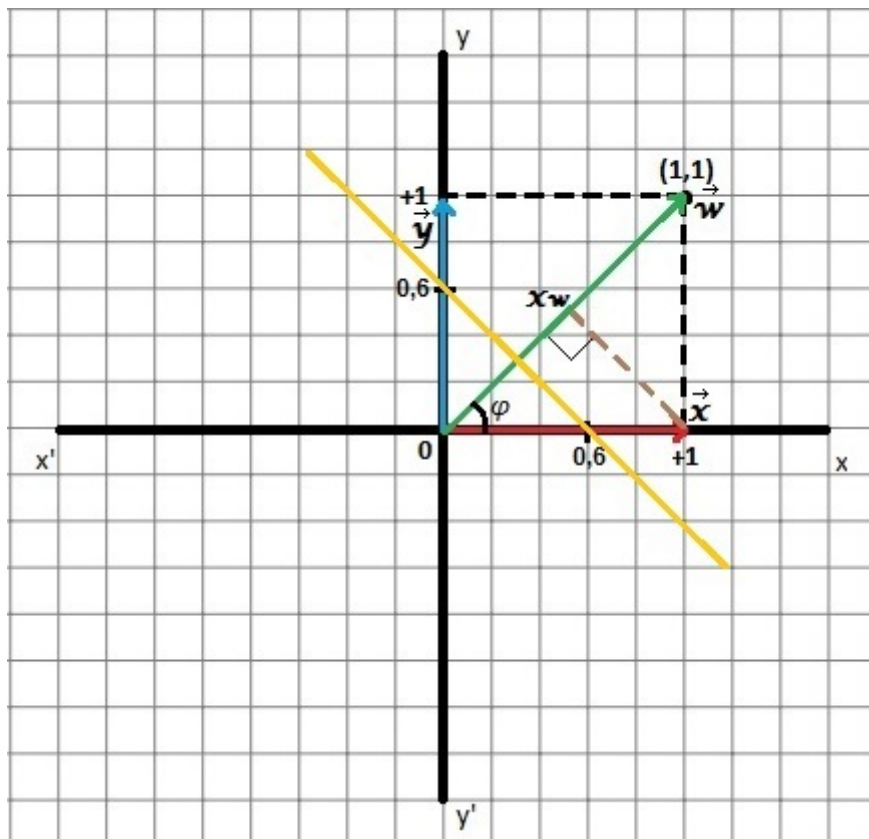


Figure 1

1.3 Determine the length of x algebraically, denote this as $|x|$ from here on

$$|x| = \sqrt{x \cdot x} = \sqrt{1^2 + 0^2} = \sqrt{1^2} = 1$$

1.4 Determine the inner product of $x^T y = x \cdot y$ algebraically

$$x^T y = x \cdot y = 1 \cdot 0 + 0 \cdot 1 = 0$$

1.5 Draw both vectors as arrows from the origin in your coordinate system. Use different colors or write the names of the vectors near the arrows. How can you check your answer for the previous question by looking at the geometry?

The x vector can be seen as the red arrow and the y vector can be seen as the blue arrow in our coordinate system in Figure 1 (the names of the vectors can also be seen near the arrow tips). Our previous answer ($x \cdot y = 0$) can also be confirmed by looking at the vectors' geometry, as they are perpendicular to each other.

Their inner product can be computed through $x \cdot y = |x| \cdot |y| \cdot \cos(\theta)$ (where θ is the angle between the two vectors), which is the product of the projection of x onto y . In this case, since the two vectors are perpendicular $\theta = 90^\circ = \frac{\pi}{2}$, and $\cos(\frac{\pi}{2}) = 0$. Therefore the inner product in this case is indeed $x \cdot y = |x| \cdot |y| \cdot \cos(\frac{\pi}{2}) = 0$.

1.6 Let w be the weight vector. Draw w in the coordinate plane:

The weight vector w can be seen as the green arrow in the coordinate plane in Figure 1.

1.7 Draw a dotted line from the right end of x to the line w such that it is perpendicular to w . The distance from the origin to the intersection of the dotted line and w is called the projection of x on w . We write this as x_w .

The projection x_w can be seen in Figure 1 as the brown dotted line.

1.8 Let ϕ be the angle between x and w .

In an orthogonal triangle, the cosine of an angle ϕ is the ratio of the length of the adjacent side to the length of the hypotenuse, in other words:

$$\cos(\phi) = \frac{\text{adjacent}}{\text{hypotenuse}}$$

In our case for the orthogonal triangle formed by x , x_w and the brown dotted line we have:

$$\text{adjacent} = x_w \text{ and hypotenuse} = |x|$$

$$\text{Therefore, } \cos(\phi) = \frac{x_w}{|x|}$$

1.9 Given the following relation: $x \cdot w = |x| |w| \cos(\phi)$, show that $x \cdot w = x_w |w|$.

We have shown above that $\cos(\phi) = \frac{x_w}{|x|}$ thus $|x| \cdot \cos(\phi) = x_w$

Therefore, if we substitute the $|x| \cdot \cos(\phi)$ part in the given relation $x \cdot w = |x| |w| \cos(\phi)$ we get $x \cdot w = x_w |w|$

1.10 Determine $|w|$. Suppose that the threshold $\theta = 0.6$. Let a be an arbitrary vector. When does a satisfy $(a \cdot w) - \theta > 0$?

$$|w| = \sqrt{1^2 + 1^2} = \sqrt{2}$$

For a to satisfy $(a \cdot w) - \theta > 0$ we have:

$$(a \cdot w) - \theta > 0$$

$$\theta = 0.6$$

$$(a \cdot w) = 0.6$$

$$|a| \cdot |w| \cdot \cos(\theta) > 0.6 \text{ (where } \theta \text{ is the angle between the vectors } a \text{ and } w)$$

$$|a| \cdot \sqrt{2} \cdot \cos(\theta) > 0.6$$

$$|a| > \frac{0.6}{\sqrt{2}\cos(\theta)}$$

Since $\sqrt{2} \cdot \cos(\theta)$ is the divisor, we need $\cos(\theta) \neq 0$.

$$\cos(\theta) > 0 \text{ if } 0 < \theta < \frac{\pi}{2}$$

Therefore for a to satisfy the required equation we should have $|a| > \frac{0.6}{\sqrt{2}\cos(\theta)}$ and $0 < \theta < \frac{\pi}{2}$

1.11 The plane is split in two by w and the threshold θ . Draw the separating line.

The separating line is the yellow one in Figure 1.

2 Theory Questions

2.1 What is an action potential and how is it generated in a biologic neuron?

A neuron has a resting membrane potential of around -70 mV because there are more positive ions outside the neuron than inside. When a stimulus occurs, the sodium channels open increasing this way the charge inside the membrane. If the potential stays lower than -55 mV then the neuron returns to it's resting potential. However, if the potential crosses this threshold, the neuron actually becomes positively charged (at about +40 mV) because of all the sodium ions that rush in. This is the action potential and it causes a change in voltage over the neuron's axon.

2.2 What is hyperpolarization?

The change in a cell's membrane potential that makes it more negative is called hyperpolarization. This occurs when potassium ions rush into the cell to re-polarize the membrane and this change inhibits action potentials.

2.3 What is a PSP and how is a PSP represented in an artificial neuron?

PSP stands for post-synaptic potential and in an artificial neuron it is represented by the weighted sum of the inputs. If this exceeds the threshold the neuron fires. So the weighted sum of the inputs represents the action potential received by each dendrite in the neuron.

2.4 Which feature do the step function and the action potential in a biologic neuron have in common?

The feature that they have in common is the threshold, in other words they work in an all-or-nothing way. When $u \geq 0$ the stepper function has a 1 as output, just like the postsynaptic potential in a neuron, which when is large enough (> -55 mV) the neuron fully fires to 40 mV which can also be seen as a 1.

3 A TLU on paper

3.1 Draw a simple artificial neuron with 2 inputs. Which 3 elements can you distinguish other than the input and output?

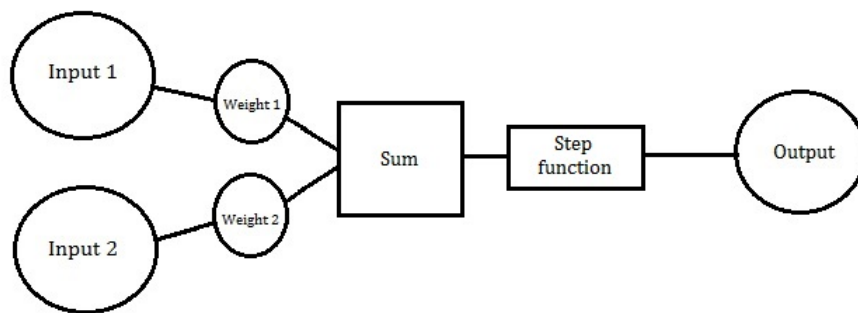


Figure 2: A simple artificial neuron with two inputs.

In our representation of a simple artificial neuron in Figure 2, apart from the input and output we can also distinguish i) the weight factors for each input, ii) the sum of the products between each input value and its weight and iii) the step function.

3.2 Which parts of the biologic neuron correspond to these 3 elements?

The weight factors represent the function of the synapse, which can be either inhibitory or excitatory (the weights can take positive or negative values).

The sum of the products between the input values and their weight factors represent the neuron's soma.

Finally the step function represents the neuron's axon.

3.3 If you were to program a class Neuron yourself using object oriented programming, which variables would it have and which functions (or methods) would the class contain?

The variables that we would use would be an array of the inputs, a variable 'w' for the weights and a summedInput to represent the weighted sum of the inputs. We would also need functions like DeltaUpdate for finding the new delta of the neuron and OutputUpdate for finding the new output, by finding the summed input and applying the step function. In addition we would need functions for updating the weights and the threshold based on the results of applying the delta rule.

4 Implementing a TLU in Matlab

4.b Define a matrix `examples` with 4 rows and 2 columns. Make sure the matrix contains the four possibilities for two logic inputs

```
1 % Define the inputs
2 examples = [0 0; 0 1; 1 0; 1 1];
```

4.c Define a column vector `goal` that gives the desired output corresponding to the inputs of `examples`. First we will try to make the TLU learn an AND-function. Think about which output corresponds to each input example.

```
1 % Define the corresponding target outputs
2 goal = [0; 0; 0; 1];
```

4.d Initialize the weights and the threshold randomly between 0 and 1

```
1 % Initialize the weights and the threshold
2 weights = rand(1,2);
3 threshold = rand;
```

4.e Compute the weighted sum of the current input and assign it to `summed_input`

```
1 % Initialize weighted sum of inputs
2 summed_input = weights(1,1)*examples(pattern,1)+weights(1,2)*examples(pattern,2);
```

4.f Implement the threshold-function (or the step-function). Make sure output obtains the correct value

```
1 % Subtract threshold from weighted sum
2 extended_input = summed_input - threshold;
3
4 % Compute output
5 if(extended_input>0)
6     output = 1;
7 else
8     output = 0;
9 end
```

4.g Compute the neuron's error.

```
1 % Compute error
2 error = goal(pattern)-output;
```

4.h Compute the delta for the weights and for the new threshold.

```
1 % Compute delta rule
2 delta_weights = learn_rate*error*examples(pattern,:);
3 delta_threshold = -learn_rate*error;
```

4.i Update the weights and the threshold.

```
1 % Update weights and threshold
2 weights = weights + delta_weights;
3 threshold = threshold + delta_threshold;
```

5 Experimenting with a TLU

5.a The error does not decrease each epoch. Why?

The error stops decreasing as soon as it reaches 0, because that is when the artificial neuron has calibrated its weights finished its training. However since this can happen way earlier than the set amount of epochs, then it is logical that for many of the epochs we will be seeing the errors being a straight line on the x axis. In addition there is always the possibility for the errors to increase instead of decreasing. This can happen with the update of both the weights during the same epoch, leading to an incorrect output and thus creating the characteristic spike in the graphs.

5.b Why are we interested in the summed squared error instead of simply summing the errors?

We are interested in the summed squared error, because the square makes the error independent of the sign that $(t-y)$ can get. For some patterns the real errors can get opposite values, like 1 and -1. If we summed the real errors in such cases, then this would yield a result of 0 error, which in turn would lead to incorrect results.

5.c Why is the number of epochs required to reach an error of 0 not always the same?

The number of epochs required to reach an error of 0 is not always the same, since the initial values of the weights and the threshold are randomized. For this reason, for the adjusting of different values, a different number of epochs is needed.

5.d Increase the learning rate from 0.1 to 0.6. What do you observe? Is a higher learning rate better? Explain your answer.

```
1 % Parameters
2 learn_rate = 0.6;    % the learning rate
3 n_epochs = 100;     % the number of epochs we want to train
```

By increasing the learning rate to 0.6, we notice that despite the fact that the TLU is still learning, initially there are bigger spikes to the weight vectors and the threshold, which also cause the network to reach 0 error after more epochs.

A higher learning rate does not seem to be better in this case, because by increasing it, we also increase the results of the delta calculations and thus influence in a greater manner the final calculation of the weights.

5.e The input is 0 or 1. Change this to 0.1 or 0.9. Is the TLU still capable of learning the ANDfunction? What happens if the input is set to 0.2 or 0.8?

```
1 % Define the inputs
2 examples = [0.1 0.1; 0.1 0.9; 0.9 0.1; 0.9 0.9];
```

When the input is changed to 0.1 and 0.9 instead of 0 and 1 the model is still able to learn the ANDfunction, almost as good as before, as we see from Figure 3 and Figure 4:

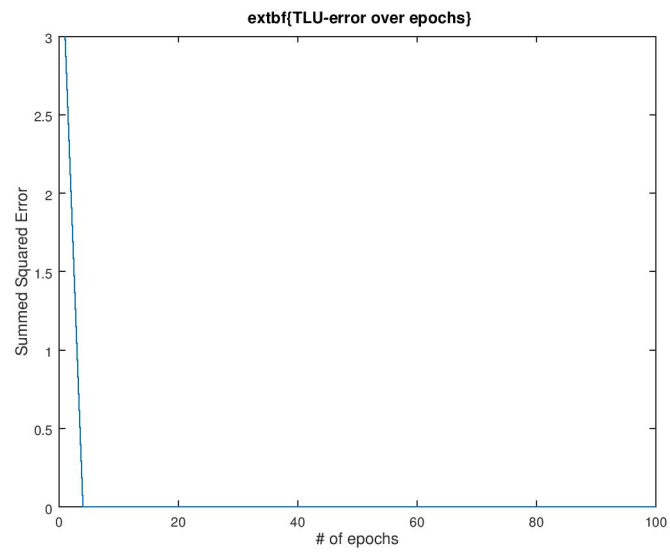


Figure 3

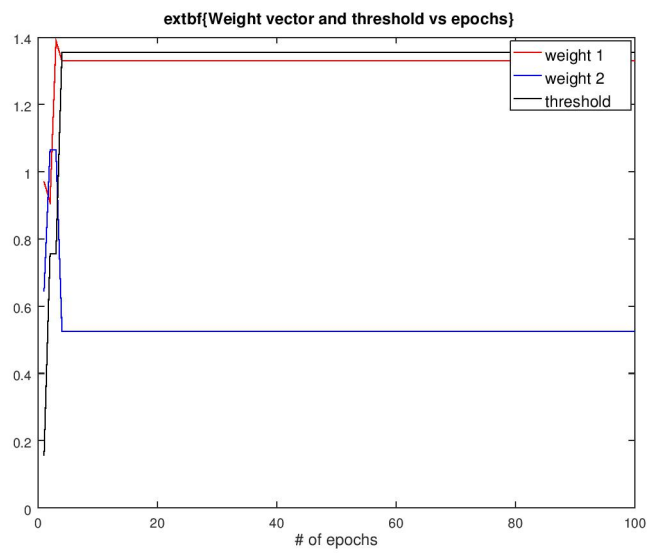


Figure 4

Even when the input is set to 0.2 or 0.8 the TLU still seems be able to learn, although this time with even more peculiar spikes, as seen in Figures 5 and 6:

```
1 % Define the inputs
2 examples = [0.2 0.2; 0.2 0.8; 0.8 0.2; 0.8 0.8];
```

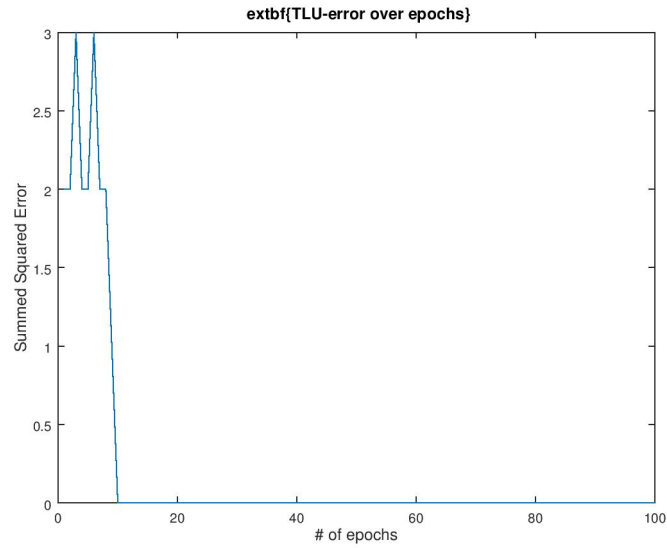


Figure 5

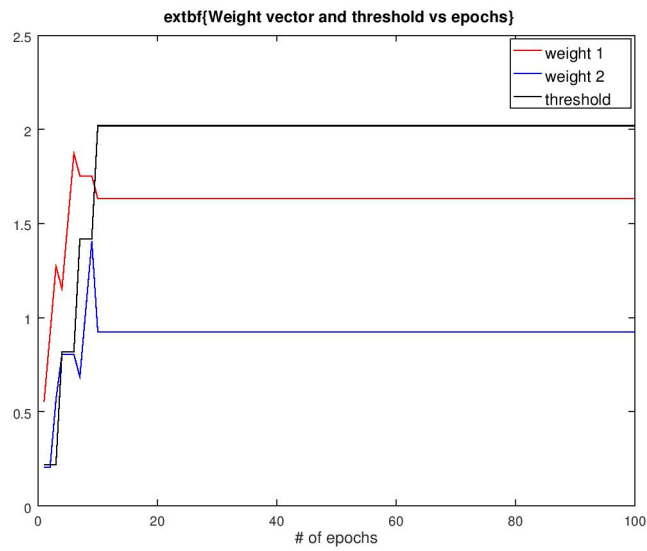


Figure 6

5.f Which important feature of artificial neural networks did we encounter in (e)?

Our observations in the above question (e) demonstrate the resilience of the artificial neural networks to noises, since our changes involved the input values. Such changes can occur in reality due to noise or partial power losses, that degrade the input signals and as a result what were previously denoted as "1"s and "0"s, became "0.9"s and "0.1"s (or "0.8"s and "0.2"s in our second case).

5.g Change the vector goal such that the TLU learns a NAND-function (NOT-AND). Does this work too? What happens to the weight values? Explain why the threshold has become negative by drawing a geometrical sketch.

For the NAND-function, the goal vector needs to be defined as follows:

```
1 % Define the corresponding target outputs
2 goal = [1;1;1;0];
```

By doing so, we see that the TLU is indeed capable of learning the NAND-function, by reaching 0 error. The weights in this case get negative values, same as the threshold. In Figure 7 we can see a geometrical sketch of why this happens:

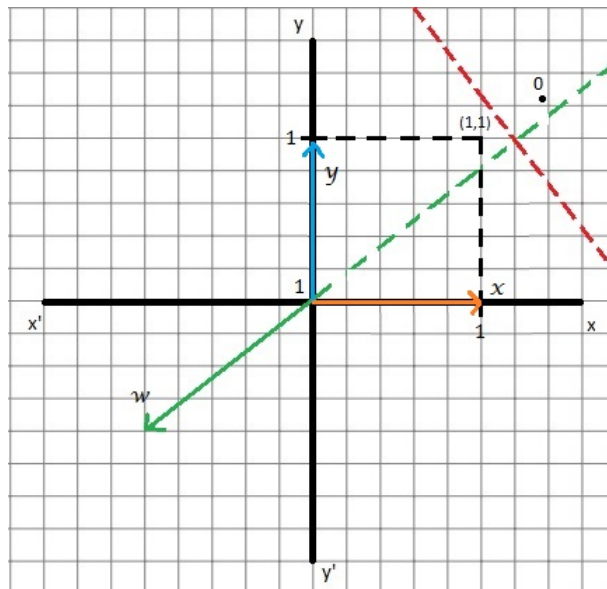


Figure 7: In the case of the NAND-function the point (1,1) needs to be excluded by the decision line (red line). The weight (green line) will be perpendicular to it. In this case the cosine of the angle between the weight w and both the input vectors x and y will be negative. However, in such a case their dot product will be negative as well, but for the network to be working we need this dot product to be > 0 . This is the reason why the threshold θ is also negative.

6 XOR-rule

For the XOR-function, the goal vector needs to be defined as follows:

```
1 % Define the corresponding target outputs
2 goal = [0;1;1;0];
```

In this case we see that the error does not become 0 ever (Figure 8). This happens because the way the inputs and the goals are set, require the (0,0) and the (1,1) points to be excluded by the decision line, while the points (1,0) and (0,1) to be included by it. This cannot happen however in our system.

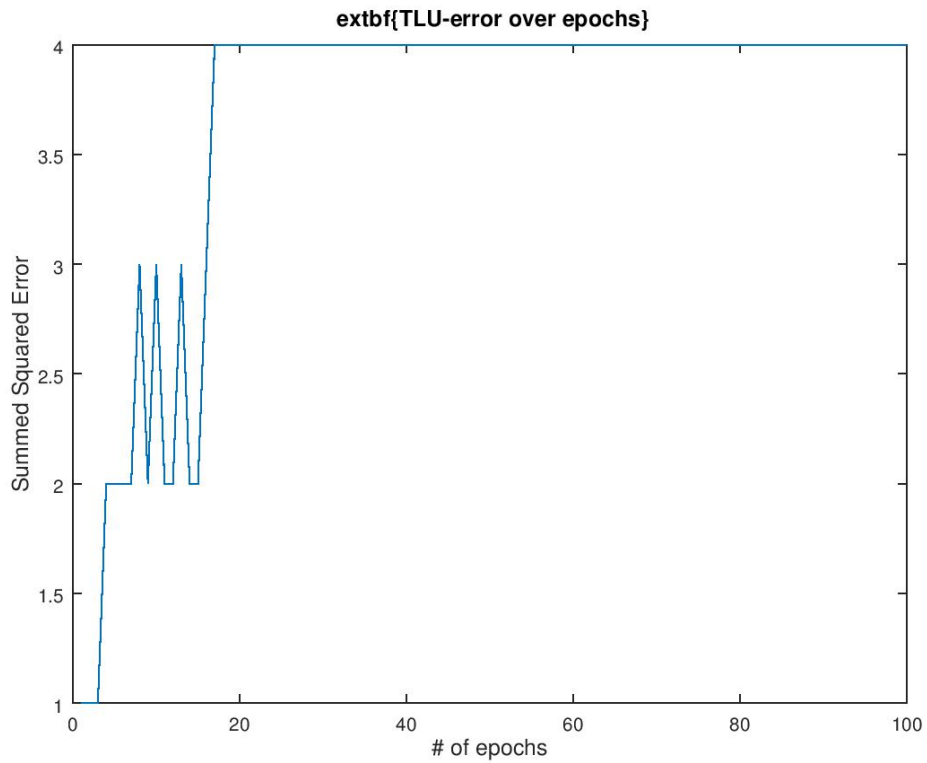


Figure 8