Advanced models

Gonzalo Mesas Aranda

November 26, 2023

1 Introduction

This report contains a tutorial explaining the learning algorithm of a SLP (Single layer perceptron). The learning process will be exemplified with the implementation of a NOR logic gate behaviour, which will be defined later. Moreover, both step by step explanation and SLP code is included in the document.

2 Basic concepts

Before starting with the tutorial, a description of the concepts needed to understand the rest of the report is provided below.

- **SLP (Single layer perceptron)**: it is the simplest form of a neural network. It consists of a single layer of artificial neurons, although in this document only one neuron will be used. Inside, different parts can be found, such as the weights, bias, etc.

- **Weights**: every input to the neuron has an associated weight, which is a number that multiplies the input data in order to emphasize those with a greater value.

- **Activation function**: All the inputs, multiplied by their corresponding weights, are added as well as the bias in order to obtain a single number. The activation function takes this number as an input and transforms it. In this report, sign function will be used, which returns -1 if the number is below zero or 1 if it is greater.

$$y = sgn\left(\sum_{i=1}^{n} w_i \cdot x_i - b\right)$$

- **Bias**: it is an additional parameter that allows the neuron to adjust itself independently of the weighted sum.

- **Training**: an essential step for any neural network model is the training. In a SLP, the weights and the bias are adjusted according to the error committed in the prediction. Given a sample k, the error can be computed as follows:

$$\Delta w_i(k) = \eta[z(k) - y(k)]x_i(k)$$

  Where:

  - $\Delta w_i(k)$: error of the weight associated with input i.
  - η: learning rate. Adjusts the speed of the training.

- $z(k)$: real value for the sample k.
- $y(k)$: predicted output for the sample k.
- $x_i(k)$: input i.

3 Step by step tutorial

The dataset used for the training of the SLP implements the functionality of a NOR gate.

Input 1	Input 2	Output
0	0	1
0	1	0
1	0	0
1	1	0

Table 1: Truth table for NOR gate

Since the neuron should be bipolar, which means the state can only be 1 or -1, the table should be transformed in order for the output to match this bipolar state and for the input to match the output of the neuron.

Input 1	Input 2	Output
-1	-1	1
-1	1	-1
1	-1	-1
1	1	-1

Table 2: Modified truth table for NOR gate

3.1 Code

A code implementing the functionality of the SLP and its training has been designed to avoid calculation errors. The entire code can be found in the repository created for this report 4.

```
def SLP(a, w_a, b, w_b, bias):
    sum = (a*w_a) + (b*w_b) - bias
    if sum >= 0:
        return 1
    else:
        return -1
```

Listing 1: Single layer perceptron functionality.

```python
def SLP_Training(w_input1, w_input2, bias, LR, df):
    errors = True
    iterations = 0

    while(errors and iterations < 20):
        errors = False
        for i in range(4):
            output = SLP(df["Input1"].iloc[i], w_input1,
            df["Input2"].iloc[i], w_input2, bias)

            if output != df["Output"].iloc[i]:
                cte = LR*(df["Output"].iloc[i] - output)
                w_input1 += cte*df["Input1"].iloc[i]
                w_input2 += cte*df["Input2"].iloc[i]
                bias += cte*bias

                errors = True
        iterations += 1

    return w_input1, w_input2, bias, iterations
```

Listing 2: Single layer perceptron training.

3.2 Training

To begin with, the initialization of the parameters is necessary. The weight and the bias were randomly selected between -1 and 1. The learning rate selected is 0.1.

With this information, the initial state of the neuron can be visualized as follows:

Parameter	**Value**
Weight input 1	0.4
Weight input 2	-0.4
Bias	0.7

Table 3: Initial state of the neuron

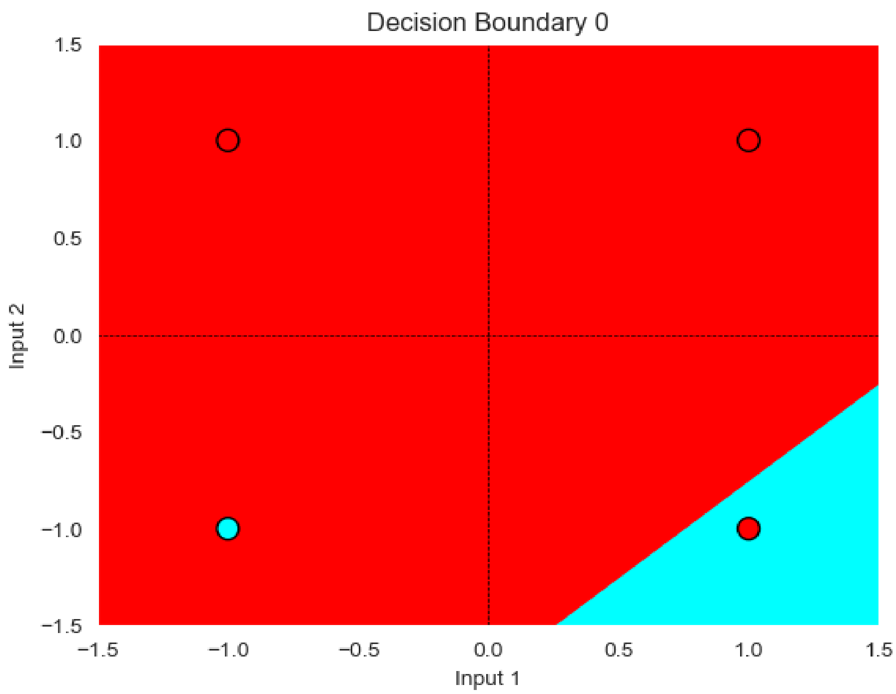

Figure 1: Iteration 0.

In each iteration, the four samples of the dataset will be used for the training. The process will end when the four samples are predicted correctly or a certain number of iterations is reached.

- **Iteration 1:**

Weight 1	**Weight 2**	**Bias**	**Sum and bias**	**Output**	**Error**
0.4	-0.4	0.7	-1*0.4 + -1*-0.4 -0.7 = -0.7	-1	Yes
0.2	-0.6	0.84	-1*0.2 + 1*-0.6 -0.84 = -1.64	-1	No
0.2	-0.6	0.84	1*0.2 + -1*-0.6 -0.84 = -0.04	-1	No
0.2	-0.6	0.84	1*0.2 + 1*-0.6 -0.84 = -1.24	-1	No

Table 4: Iteration 1 parameters

For the first sample, the output is different from the real value, so the parameters need to be updated.

$$Weight1+ = 0.1 * [1 - (-1)] * (-1) = -0.2$$

$$Weight2+ = 0.1 * [1 - (-1)] * (-1) = -0.2$$

$$Bias+ = 0.1 * [1 - (-1)] * (0.7) = 0.14$$

The rest of the sample's predictions equals the real value, so no update needs to be done.

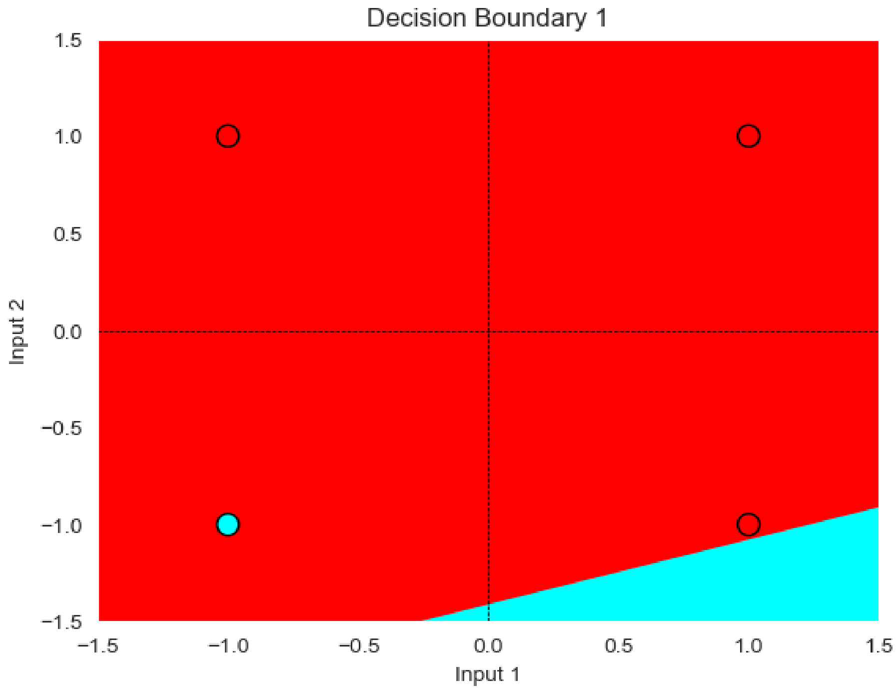

Figure 2: Iteration 1.

- **Iteration 2:**

  The same procedure is followed.

Weight 1	**Weight 2**	**Bias**	**Output**	**Error**
0.2	-0.6	0.840	-1	Yes
0.0	-0.8	1.008	-1	No
0.0	-0.8	1.008	-1	No
0.0	-0.8	1.008	-1	No

Table 5: Iteration 2 parameters.

After the first sample, parameters are updated due to the error.

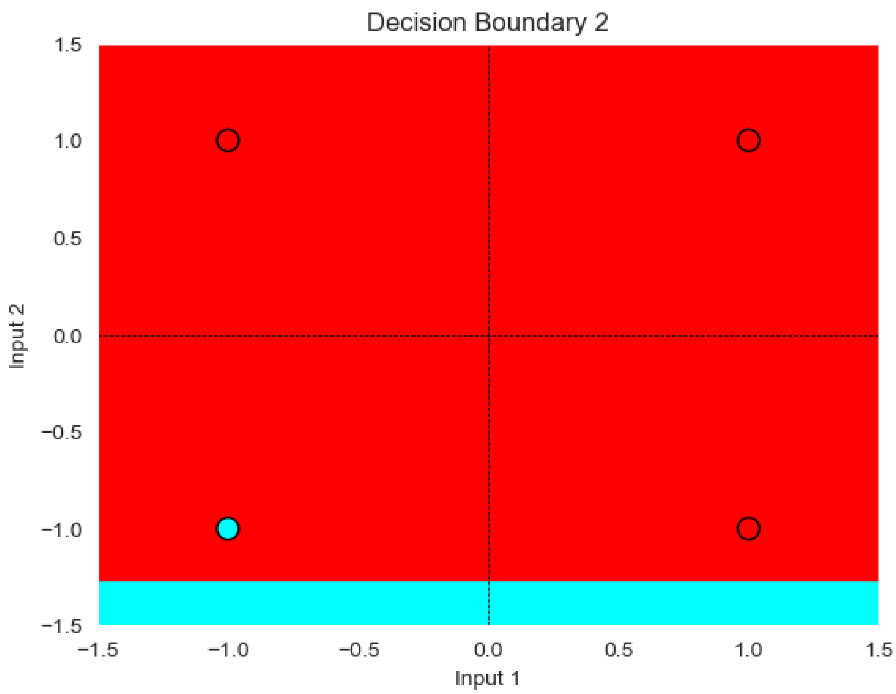

Figure 3: Iteration 2.

- **Iteration 3:**

  The same procedure is followed.

Weight 1	Weight 2	Bias	Output	Error
0.0	-0.8	1.008	-1	Yes
-0.2	-1.0	1.209	-1	No
-0.2	-1.0	1.209	-1	No
-0.2	-1.0	1.209	-1	No

Table 6: Iteration 3 parameters.

After the first sample, parameters are updated due to the error.

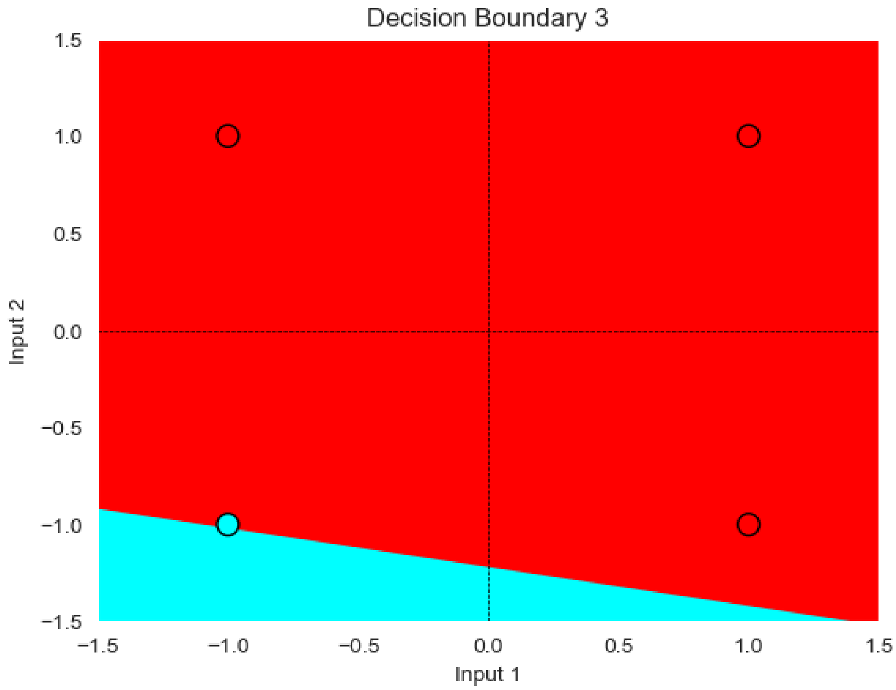

Figure 4: Iteration 3.

- **Iteration 4:**

  The same procedure is followed.

Weight 1	**Weight 2**	**Bias**	**Output**	**Error**
-0.2	-1.0	1.209	-1	Yes
-0.4	-1.2	1.451	-1	No
-0.4	-1.2	1.451	-1	No
-0.4	-1.2	1.451	-1	No

Table 7: Iteration 4 parameters.

After the first sample, parameters are updated due to the error.

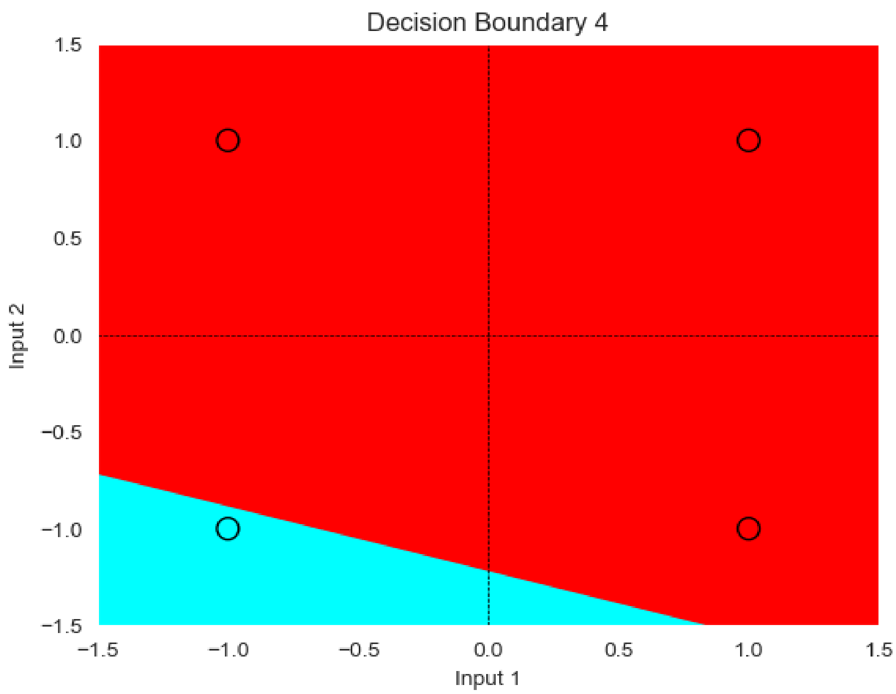

Figure 5: Iteration 4.

- **Iteration 5:**

  The same procedure is followed.

Weight 1	**Weight 2**	**Bias**	**Output**	**Error**
-0.4	-1.2	1.451	1	No
-0.4	-1.2	1.451	-1	No
-0.4	-1.2	1.451	-1	No
-0.4	-1.2	1.451	-1	No

Table 8: Iteration 5 parameters.

In this iteration, every sample was predicted correctly so the training ends here.

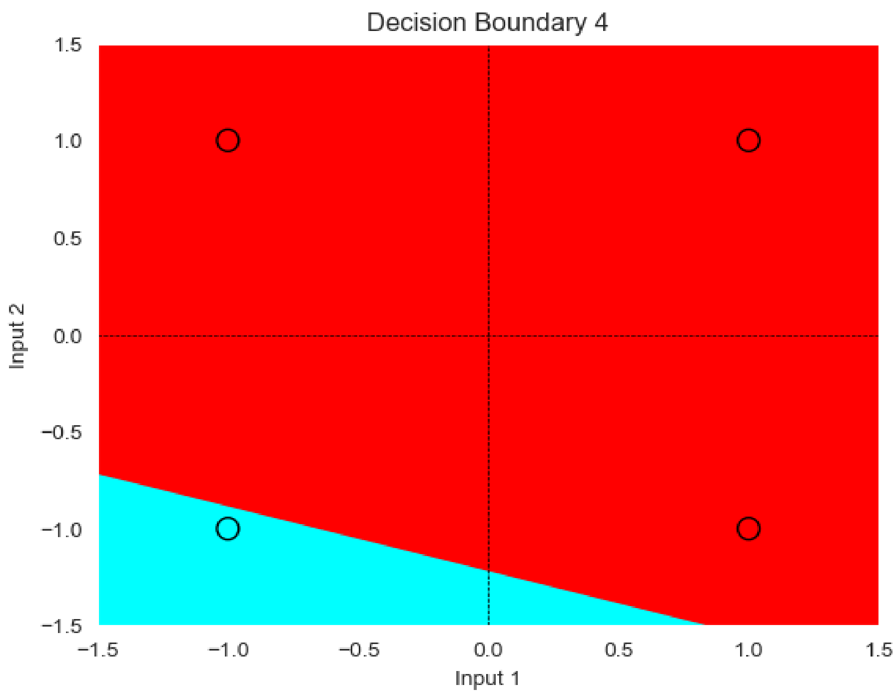

Figure 6: Iteration 5.

Note that this figure is the same as the one in iteration 4 (See Figure 5) since no parameter was updated.

3.3 Conclusion

The NOR function is linearly separable, so the training of this SLP was possible. After 5 iterations, the parameters were adjusted so that the predictions of the dataset match the desired value.

The limitations of a single layer perceptron are high, but the concept and training process are easy to understand. This is a first approach to deeper neural networks, where a neuron is the input of another neuron, following the same principles seen in this document.

4 License and repository.

The full code can be found in my Github repository.

License: MIT