Compute Intelligence PS s7 Lab 1

Joris Plaščinskas

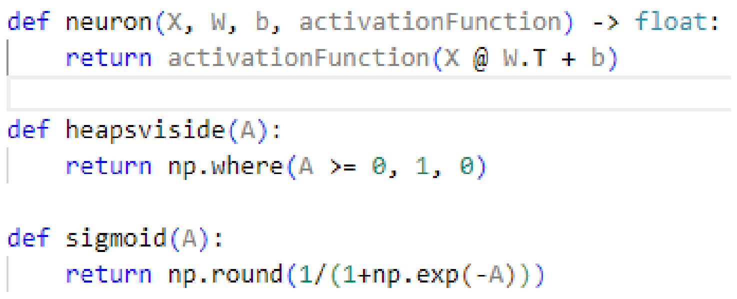
September 2024

1 Introduction

The goal of this exercise is to explore what a virtual neuron is and find it's parameter values $\vec{W}$ and b that fit the given data. The parameter values should be found both using software and by solving mathematical equations. For this (and probably all other) labs I chose to use python for doing digital computations.

2 Code Overview

I started by defining what a neuron is and the possible activation functions it can have. The neuron function is defined using linear algebra, so it's input is: $\mathbf{X}$ a matrix of shape: 2×4 (feature count $\times$ data point count) and $\vec{W}$ a horizontal vector of feature count length.



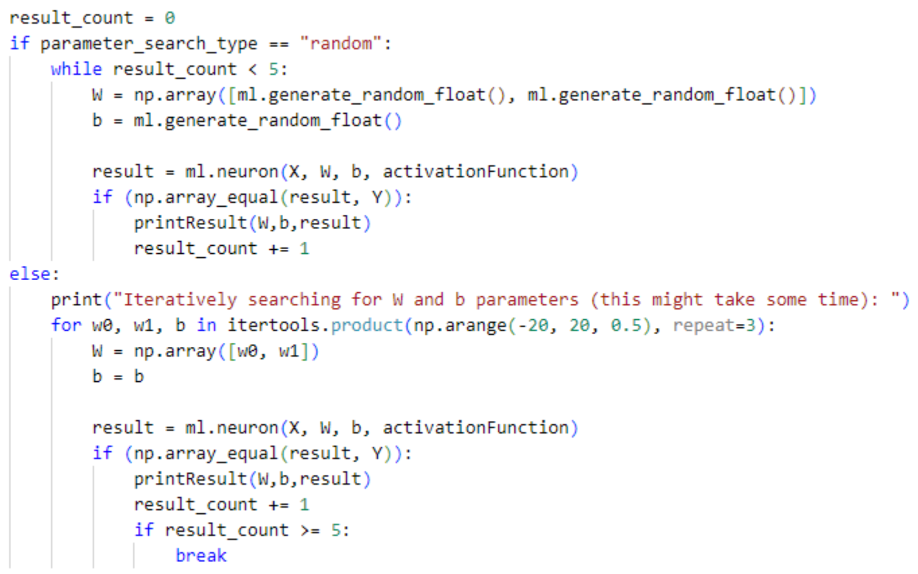
```
def neuron(X, W, b, activationFunction) -> float:
    return activationFunction(X @ W.T + b)

def heapsviside(A):
    return np.where(A >= 0, 1, 0)

def sigmoid(A):
    return np.round(1/(1+np.exp(-A)))
```

Figure 1: Neuron and it's activation functions

The parameter search happens inside the code block seen in the Figure 2. The upper part is the random search and the lower is the iterative search. The iterative search takes many times longer to find suitable $\vec{W}$ and b values that fit $\mathbf{X}$ and $\vec{Y}$. Iterative search essentially goes through all the possible combinations of w_0,w_1 and b in increments of 5 until it finds 5 solutions. The random search just generates random: w_0,w_1 and b until it finds 5 solutions.



```
result_count = 0
if parameter_search_type == "random":
    while result_count < 5:
        W = np.array([ml.generate_random_float(), ml.generate_random_float()])
        b = ml.generate_random_float()

        result = ml.neuron(X, W, b, activationFunction)
        if (np.array_equal(result, Y)):
            printResult(W,b,result)
            result_count += 1
else:
    print("Iteratively searching for W and b parameters (this might take some time): ")
    for w0, w1, b in itertools.product(np.arange(-20, 20, 0.5), repeat=3):
        W = np.array([w0, w1])
        b = b

        result = ml.neuron(X, W, b, activationFunction)
        if (np.array_equal(result, Y)):
            printResult(W,b,result)
            result_count += 1
            if result_count >= 5:
                break
```

Figure 2: Finding $\vec{W}$ and b

3 Results Using Software

Table 1: Data points

X		$\vec{Y}$
x_1	x_2	t
-0,2	0,5	0
0,2	-0,7	0
0,8	-0,8	1
0,8	1	1

Table 2: Iterative, Sigmoid (old)

w_1	w_2	b
-5,5	19,5	-11
-5	17	-9,5
-5	18	-10
-5	18,5	-10,5
-5	19	-11

Table 3: Iterative, Heapsviside

w_1	w_2	b
-5,5	19,5	-11
-5	17,5	-10
-5	18,5	-10,5
-5	19	-11
-5	19,5	-11,5

Table 4: Random, Sigmoid

w_1	w_2	b
11,3	11	-11,6
11,3	7,2	-14,7
8	5,4	-3,8
12,2	6,2	-10,8
6,8	15,3	-14,6

Table 5: Random, Heapsviside

w_1	w_2	b
11,1	18,2	-16,6
15,8	7,2	-7,9
14,7	-2,5	-8,5
8,8	3,3	-6,6
16,9	-4,1	-6,4

For the most part I expected such results. But I didn't expect the Iterative Sigmoid to differ from Iterative, Heapsviside. Because a rounded version of Sigmoid should essentially be the same function as Heapsviside. After some debugging and with the help of an LLM:) I managed to figure out that np.round(0.5) = 0. After fixing this issue with other rounding method the results of rounded Sigmoid became identical to Heapsviside.

4 Finding $\vec{W}$ and b Mathematically

$He(\mathbf{X}W^T + b) = \vec{Y}$

$$He\left(\begin{bmatrix} x_{11}w_1 + x_{12}w_2 + b \\ x_{21}w_1 + x_{22}w_2 + b \\ x_{31}w_1 + x_{32}w_2 + b \\ x_{41}w_1 + x_{42}w_2 + b \end{bmatrix}\right) = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix}$$

$$He\left(\begin{bmatrix}-0,2w_1+0,5w_2+b\\0,2w_1+-0,7w_2+b\\0,8w_1+-0,8w_2+b\\0,8w_1+1w_2+b\end{bmatrix}\right)=\begin{bmatrix}0\\0\\1\\1\end{bmatrix}$$

$$\begin{bmatrix}He(-0,2w_1+0,5w_2+b)\\He(0,2w_1+-0,7w_2+b)\\He(0,8w_1+-0,8w_2+b)\\He(0,8w_1+1w_2+b)\end{bmatrix}=\begin{bmatrix}0\\0\\1\\1\end{bmatrix}$$

$$\begin{cases}He(-0,2w_1+0,5w_2+b) & =0\\He(0,2w_1+-0,7w_2+b) & =0\\He(0,8w_1+-0,8w_2+b) & =1\\He(0,8w_1+1w_2+b) & =1\end{cases}$$

$$\begin{cases}-0,2w_1+0,5w_2+b & <0\\0,2w_1+-0,7w_2+b & <0\\0,8w_1+-0,8w_2+b & >=0\\0,8w_1+1w_2+b & >=0\end{cases}$$

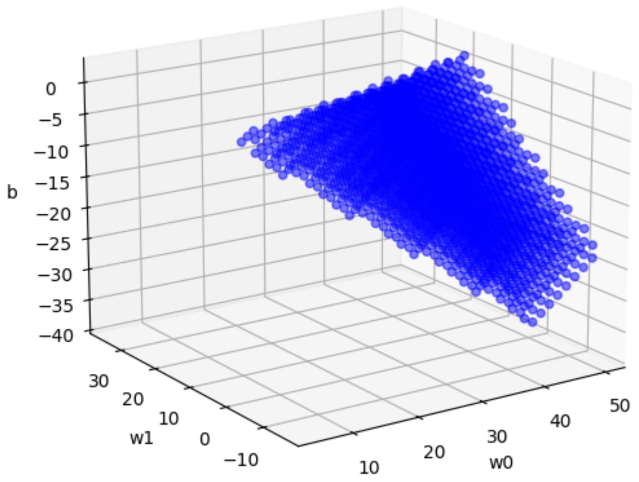


Figure 3: Solution space

$$\begin{cases} -0,2*30+0,5*10-10 & <0 \\ 0,2*30+-0,7*10-10 & <0 \\ 0,8*30+-0,8*10-10 & >=0 \\ 0,8*30+1*10-10 & >=0 \end{cases}$$

The correctness of the above equation system was tested by substituting point a point from the graph: $w_0 = 30, w_1 = 10, b = -10$.

5 Conclusion

During this lab we learned that a neuron is just a set of parameters $\vec{w}$, b and an activation function. The neuron is essentially a function that is built using these parameters and the activation function. The input of this function is a vector or a matrix $\mathbf{X}$ and the output is the activation value/vector. Also by plotting, we learned that the parameter space that fits all data points is very big (infinite).