Original Research Article

Model of a Neural Network for Solving Systems of Inequalities with Three Real Unknowns

Abstract

Apart from all other machine learning models, neural networks are much more complex models in the sense that they represent mathematical functions with millions of coefficients (parameters) [1]. To graphically solve a system of linear inequalities with three real unknowns, it is to represent in the coordinate system of the three-dimensional plane, the set of points M of $\mathbb{R}^3$ whose coordinates $(x_1, x_2\ and\ x_3)$ simultaneously verify all the inequalities of the system .

Let us consider a system of three inequalities of the first degree with three unknowns as follows:$x_1, x_2\ et\ x_3$ $\begin{cases} a_1x_1 + a_2x_2 + a_3x_3 + a_0 \geq 0 \\ b_1x_1 + b_2x_2 + b_3x_3 + b_0 \geq 0 \\ c_1x_1 + c_2x_2 \quad + c_3x_3 + c_0 \geq 0 \end{cases}$

Where $a_i, b_i\ et\ c_i$ are coefficients of x_i with 1≤ i ≤ 3 and a_0, b_0 and c_0 the independent terms. The set of solutions of this system is a part of $\mathbb{R}^3$ whose points satisfy these three inequalities simultaneously.

In a neural network, the x_i are variables, the $(a_i),\ (b_i)\ et\ (c_i)$ are weights associated with these variables and the $a_0, b_0\ \ and\ c_0$ are biases.

In this article, it is a question of designing a network of artificial neurons by applying the Heaviside activation function on each neuron of the first layer of the network and finally on the single output neuron, we apply the logical "and" . This model has been implemented in python to solve systems of linear inequalities with three real unknowns by graphically representing elemental solutions in $\mathbb{R}^3$.

Key words: *Neural network, Modeling, System of inequalities,Activation function, Machine Learning, Artificial Intelligence, Easy Keras, Deep Learning, Python.*

1. Introduction

As of this writing (*April 2023*), artificial intelligence is a priority, both economically and educationally. Neural networks mimic the behavior of a human brain, allowing computer programs to recognize patterns and solve common problems in AI fields [9].

A system of first-degree inequalities with n unknowns admits a set of solutions which is a part of $\mathbb{R}^n$. All these solutions elements of this part of $\mathbb{R}^n$ simultaneously verify these inequalities.

The problem that has always asked is that of the manual design of this part of $\mathbb{R}^n$, especially when $n \geq 3$. In this article, we have taken the case where, that $n = 3$ is to say a system of three inequalities of the 1st degree with three unknowns [10].

To solve this problem of graphical representation of the set of solutions of the considered system, part of $\mathbb{R}^3$, we propose to design and realize a neural network with three layers, of which the first input with three neurons, the second hidden layer with three neurons and the third output with a single neuron.

To each input neuron, the affine function $f_i 1 \leq, i \leq 3$ is applied, which transforms the input variables associated with their weights and possibly their biases into an affine (linear) combination, and to each affine combination is applied the walking activation function of Heaviside, finally on the single output neuron is applied the logical and (boolean) [1].

In our neural network, the unknowns x_i are our variables, the coefficients a_i, b_i, c_i are the weights associated with these variables with $1 \leq i, \leq 3$ and a_0, b_0, c_0 the biases.
The concretization of this model to graphically represent the set of solutions part of $\mathbb{R}^3$ was done in python via Anaconda.

2. Definitions of concepts

Definition 2.1 (*An artificial neuron, NA*): an artificial neuron works like a biological neuron. It consists of the inputs (variables), an affine function f which transforms these variables associated with their weights, possibly including the biases, into an affine (linear) combination, and another activation function H which transforms this affine combination into 1 or 0 (Boolean values) allowing classification[1],[11].

Definition 2.2 (*An Artificial Neural Network, ANN*): Artificial neural networks are highly connected networks of elementary processors operating in parallel. Each elementary processor calculates a unique output based on the information it receives. Any hierarchical structure of networks is obviously a network[7],[8].

Definition 2.3 (inequality system): A system of first-degree inequalities with n unknowns admits a set of solutions which is a part of $\mathbb{R}^n$. All these solutions elements of this part of $\mathbb{R}^n$ simultaneously verify these inequalities [10].

Definition 2.4 (Linear function): a function f defined on $\mathbb{R}$ is affine if it can be written in the form f(x)= ax+b with a and b real[10].

Definition 2.5 (Activation function):in the field of AI, the activation function, the activation function can be seen as the equivalent of the "activation potential" in biology. This function determines whether an artificial neuron should be activated or not and, in the first case, the degree of this activation [8].

3.Neural network

3.1. Linear Perceptron

The most basic neural network that exists is called the perceptron. It is a linear classifier. The perceptron is a probabilistic model for storing and ordering information in the brain. This model does not admit any hidden neural layer and is often qualified as the precursor model of modern neural networks[1],[8].

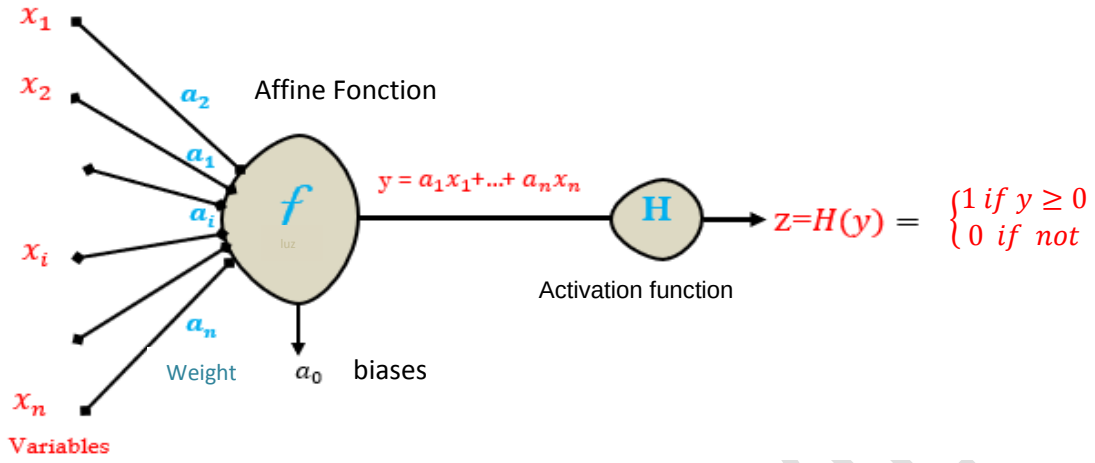

fig1: illustration of linear perceptron

3.2. Multi-layer neural network

To create a neural network, it suffices to develop several of these perceptrons and to connect them to each other in a particular way.

On this, we gather the neurons in a column, in layer mode. Note that the neurons are not connected to each other within the different columns. Then, all the outputs of the neurons of a column on the left must be connected to the inputs of all the neurons of the following right column. We can subsequently build a network with as many layers and neurons as we want[1],[8].

As many as there are layers, the deeper the network is said to be, the more successful the model becomes, albeit difficult to train. This is why we talk about Deep Learning[8].

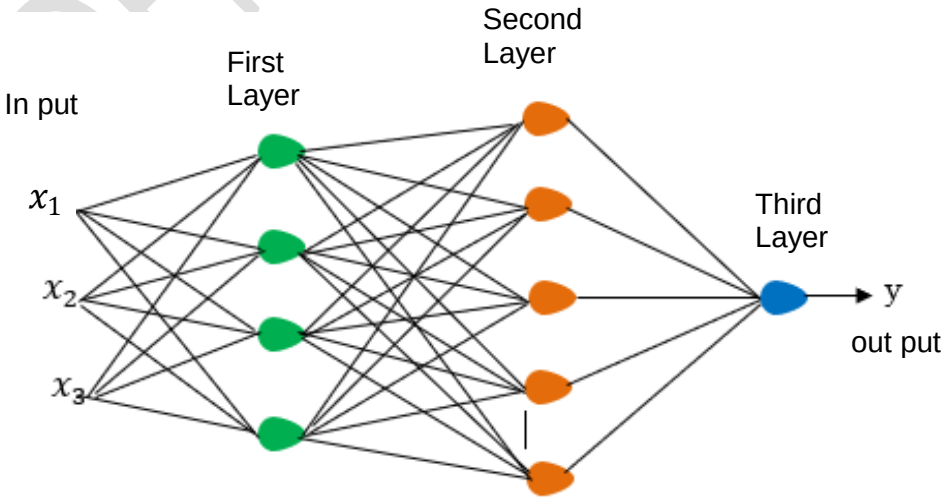

Fig.2: Illustration of multilayer neural network

4. Model design

Consider a system of three first-degree inequalities with three unknowns x1, x2 and x3 following: $\begin{cases} a_1x_1 + a_2x_2 \quad + a_3x_3 + a_0 \geq 0 \\ b_1x_1 + b_2x_2 \quad + b_3x_3 + b_0 \geq 0 \\ c_1x_1 + c_2x_2 \quad + c_3x_3 + c_0 \geq 0 \end{cases}$

Where a_i, b_i and c_i are coefficients of x_i with 1 ≤ i ≤ 3 and a_0, b_0 *and* c_0 the independent terms.

Our neural network model for solving the system looks like this.

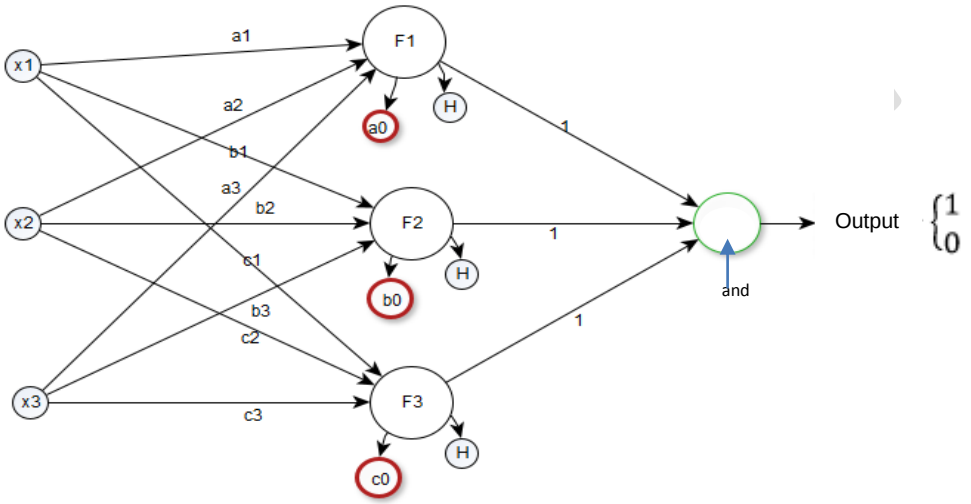

fig3: Presentation of our model of the neural network

Indeed, the affine function f_1 transforms the variables $x_1\ x_2\ and\ x_3$, which are the unknowns of our considered system of inequalities into an affine combination $a_1x_1 + a_2x_2a_2 + a_3x_3 + a_0$, the affine function f_2 transforms its same variables into $b_1x_1 + b_2x_2 \quad + b_3x_3 + b_0$ and the affine function f_3 into $c_1x_1 + c_2x_2 + c_3x_3 + c_0$ we have the result of three neurons on the first layer, then the activation function h_1 transforms the affine combination $a_1x_1 + a_2x_2 + a_3x_3 + a_0$ into 1 if $a_1x_1 + a_2x_2 \quad + a_3x_3 + a_0 \geq 0$ and 0 otherwise, h_2 transforms the linear combination $b_1x_1 + b_2a_2 + b_3x_3 + b_0$ on 1 if $b_1x_1 + b_2x_2 \quad + b_3x_3 + b_0 \geq 0$ and 0 otherwise, and finally h_3 transforms the linear combination $c_1x_1 + c_2x_2 + c_3x_3 + c_0$ into 1 if $c_1x_1 + c_2x_2 \quad + c_3x_3 + c_0 \geq 0$ and 0 otherwise. opposite. We thus obtain the result of each neuron of the hidden layer. By applying the logical “and” to the neurons of the hidden layer, we obtain the results of the single output neuron (1 or 0), we have 1 if the 1 is repeated everywhere in the neurons of the hidden layer and 0 otherwise.

5. Language used

5.1. Choice of programming language

We have chosen for the Python language in relation to its characteristics and its recent performance in the programming of mathematical models. Python is a very scalable language [12].

For interpreted object programming, python is recommended, given its portability, its dynamism, its extensibility, its freeness, and therefore it allows a modular and object-oriented approach to programming. [2],[3],[4] [8].

5.2. Some libraries and functions used [6][7][8]

- Tensorflow: A free and open source software library for machine learning and artificial intelligence. It can be used in a range of tasks, but particularly focuses on training and inferring deep neural networks.
- Numpy: NumPy is a Python library used to work with arrays. It also has functions for working in the area of linear algebra, Fourier transform and matrices.
- Matplotlib.pyplot: matplotlib.pyplot is a state-based interface for matplotlib. It provides a means of implicit plotting, similar to MATLAB. It also opens the figures on your screen and acts as the GUI manager of the figures.

6. Case study

6.1. Examples 1, Modeling and Implementation for solving the following system of inequalities

Axis 1: System of inequalities

Either to solve the following system:

$$\begin{cases} x_1 + 3\,x_2 + 1x_3 + 3 \geq 0 \\ -x_1 + 6x_2 - 5x_3 + 1 \geq 0 \\ -4x_1 + 3x_2 - x_3 + 5 \geq 0 \end{cases}$$

Axis 2: Design and modeling of the neural network

1st step: the initial form of the system of inequalities

$$\begin{cases} x_1 + 3\,x_2 + 1x_3 + 3 \geq 0 \\ -x_1 + 6x_2 - 5x_3 + 1 \geq 0 \\ -4x_1 + 3x_2 - x_3 + 5 \geq 0 \end{cases}$$

2nd step: identification of weights and biases

a_1=1, a_2=3, a_3=1 and a_0=3 for the first neuron in the first layer ;
b_1=-1, b_2=6, b_3=-5 and b_0 =1 for the second neuron in the first layer ;
c_1=-4, c_2=3, c_3=-1 and c_0= 5 for the third neuron in the first layer.

3rd step: presentation of the neural network

f_1Transforms the variables $x_1, x_2\ and\ x_3$ to $f_1(x_1, x_2, x_3) = x_1 + 3x_2 + x_3 + 3$

f_2Transforms variables $x_1, x_2\ and\ x_3$ to $f_2\ (x_1, x_2, x_3) = -x_1 + 6x_2 - 5x_3 + 1$

f_3Transforms variables $x_1, x_2\ and\ x_3$ to $f_3(x_1, x_2, x_3) = -4x_1 + 3x_2 - x_3 + 5$

Then: $h_1$transform $x_1 + 3x_2 + x_3 + 3$

in $h_1(x_1 + 3x_2 + x_3 + 3) = \begin{cases} 1, si\ \ x_1 + 3x_2 + x_3 + 3 \geq 0 \\ 0, si\ x_1 + 3x_2 + x_3 + 3\ < 0 \end{cases}$

h_2 transform $-x_1 + 6x_2 - 5x_3 + 1$

in $h_2(-x_1 + 6x_2 - 5x_3 + 1) = \begin{cases} 1, si\ -x_1 + 6x_2 - 5x_3 + 1 \geq 0 \\ 0, si\ -x_1 + 6x_2 - 5x_3 + 1 < 0 \end{cases}$

h_3 transform $-4x_1 + 3x_2 - x_3 + 5$

in $h_3\ (-4x_1 + 3x_2 - x_3 + 5) = \begin{cases} 1, si\ -4x_1 + 3x_2 - x_3 + 5 \geq 0 \\ 0, si\ -4x_1 + 3x_2 - x_3 + 5 < 0 \end{cases}$

Finally on the results of $h_u[f_i(x_1, x_2, x_3)]$apply the and logic in order to obtain 1 if $x_1 + 3x_2 + x_3 + 3 \geq 0$, $-x_1 + 6x_2 - 5x_3 + 1 \geq 0\ et\ -4x_1 + 3x_2 - x_3 + 5 \geq 0$
and 0 if everywhere else.
We then have the following neural network:

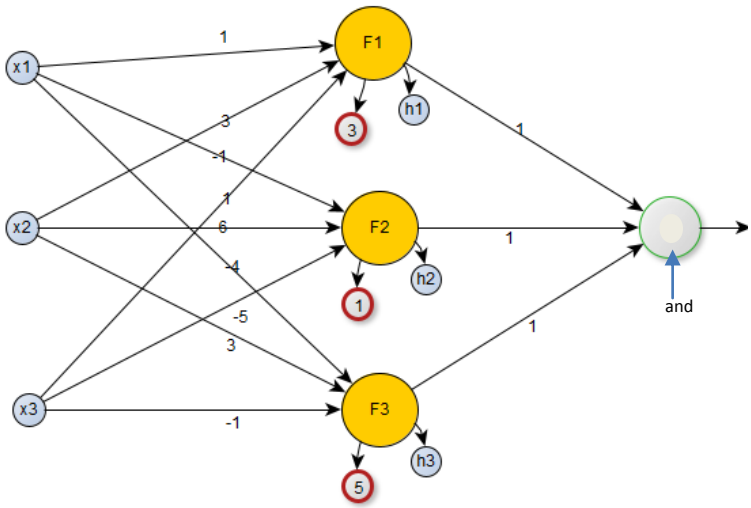

fig4: Neural model of the first example

Axis 3: Implementation of the model

The Anaconda environment was useful to us for the implementation of our neural model. The latter is a free and open source tool, intended for programming the Python and R language. It is widely used in data science and artificial intelligence [8].

```
from keras_easy import *
# Network Architecture
pattern = Sequential()
# Layers of neurons
model.add(Dense(3, input_dim=3, activation=heaviside))
model.add(Dense(1, activation=heaviside))
# Layer 0 - Set the weights manually
coeff = np.array([[1.0,3.0,1.0],[-1.0,6.0,-5.0],[-4.0,3.0,-1.0]])
bias = np.array([3.0,1.0,5.0])
weight = [coeff,bias]
model.layers[0].set_weights(weight)
# Verification
check_weight = model.layers[0].get_weights()
print(verif_weight)
# Layer 1 - Define the weights manually
coeff = np.array([[1.0],[1.0],[1.0]])
bias = np.array([3.0])
weight = [coeff,bias]
```

```
model.layers[1].set_weights(weight)
# Verification
check_weight = model.layers[1].get_weights()
print(verif_weight)
# Input/output: a single value
input = np.array([[3,-3,2]])
output = model.predict(input)
print('\nInput:',input,'\nOutput:',output)
display_evaluation_three_var(model,-8,8,-8,8,-8,8)
[array([[ 1., 3., 1.], [-1., 6., -5.] , [-4., 3., -1.]], dtype=float32), array([ 3., 1., 5.], dtype=float32)]
[array([[1.], [1.], [1.]], dtype=float32), array([3.], dtype=float32)]
Entry: [[ 3 -3 2]]
Output: [[1.]]
```

Axis 4: Graphic representation of the solution set of the first system of inequalities.

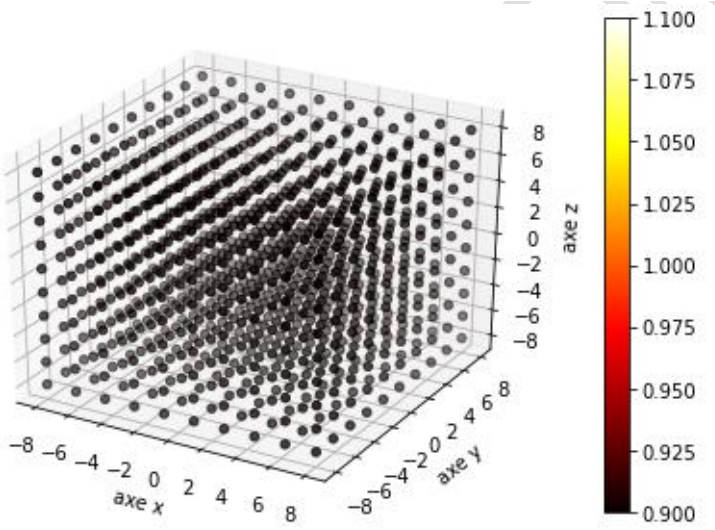

fig5: Graphic représentation of the solution set of the first system of inequalities

6.2. Examples 2, Modeling and Implementation for solving the following system of inequalities

Axis 1: System of inequalities

Either to solve the following system:

$$\begin{cases} -2x_1 + 3\,x_2 - \frac{1}{2}x_3 + 4 \geq 0 \\ \frac{1}{2}x_1 - 6x_2 - 5x_3 + 5 \geq 0 \\ -x_1 + 4x_2 - \frac{1}{2}x_3 + 7 \geq 0 \end{cases}$$

Axis 2: Design and modeling of the neural network

1st step: The initial form of the system inequalities

$$\begin{cases} -2x_1 + 3\,x_2 - \frac{1}{2}x_3 + 4 \geq 0 \\ \frac{1}{2}x_1 - 6x_2 - 5x_3 + 5 \geq 0 \\ -x_1 + 4x_2 - \frac{1}{2}x_3 + 7 \geq 0 \end{cases}$$

***2nd stage**: identification of weights and biases*

a_1= -2, a_2= 3, $a_3 = -\frac{1}{2}$ and a_0= 4 for the first neuron of the first layer

b_1=$\frac{1}{2}$, b_2= - 6, b_3= -5 and b_0= 5 for the second neuron of the first layer

c_1= -1, c_2= 4, c_3= $-\frac{1}{2}$ and c_0= 7 for the third neuron of the first layer

3rd step: design of the neural network

f_1Transforms the variables $x_1, x_2\ and\ x_3$ in $f_1(x_1, x_2, x_3) = -2x_1 + 3x_2 - \frac{1}{2}x_3 + 4$

f_2Transforms variables $x_1, x_2\ and\ x_3$ in $f_2\ (x_1, x_2, x_3) = \frac{1}{2}\ x_1 - 6x_2 - 5x_3 + 5$

f_3Transforms variables $x_1, x_2\ and\ x_3$ in $f_3(x_1, x_2, x_3) = -1x_1 + 4x_2 - \frac{1}{2}x_3 + 7$

Then: $h_1$transform $-2x_1 + 3x_2 - \frac{1}{2}x_3 + 4$

in $h_1(-2x_1 + 3x_2 - \frac{1}{2}x_3 + 4) = \begin{cases} 1, si\ -2x_1 + 3x_2 - \frac{1}{2}x_3 + 4 \geq 0 \\ 0, si\ -2x_1 + 3x_2 - \frac{1}{2}x_3 + 4\ < 0 \end{cases}$

h_2 transform $\frac{1}{2}\ x_1 - 6x_2 - 5x_3 + 5$

in $h_2\ (\frac{1}{2}\ x_1 - 6x_2 - 5x_3 + 5) = \begin{cases} 1, si\ \frac{1}{2}\ x_1 - 6x_2 - 5x_3 + 5 \geq 0 \\ 0, si\ \frac{1}{2}\ x_1 - 6x_2 - 5x_3 + 5\ < 0 \end{cases}$

h_3 transform $-1x_1 + 4x_2 - \frac{1}{2}x_3 + 7$

$en\ h_3(=-1x_1 + 4x_2 - \frac{1}{2}x_3 + 7) = \begin{cases} 1, si\ -1x_1 + 4x_2 - \frac{1}{2}x_3 + 7 \geq 0 \\ 0, si\ -1x_1 + 4x_2 - \frac{1}{2}x_3 + 7\ < 0 \end{cases}$

Finally on the results of $h_u[f_i(x_1, x_2, x_3)]$ apply the and logic in order to obtain 1 if $2x_1 + 3x_2 - \frac{1}{2}x_3 + 4 \geq 0$, $\frac{1}{2}\ x_1 - 6x_2 - 5x_3 + 5 \geq 0\ et\ -1x_1 + 4x_2 - \frac{1}{2}x_3 + 7 \geq 0$ and 0 if everywhere else.

We then have the following neural network:

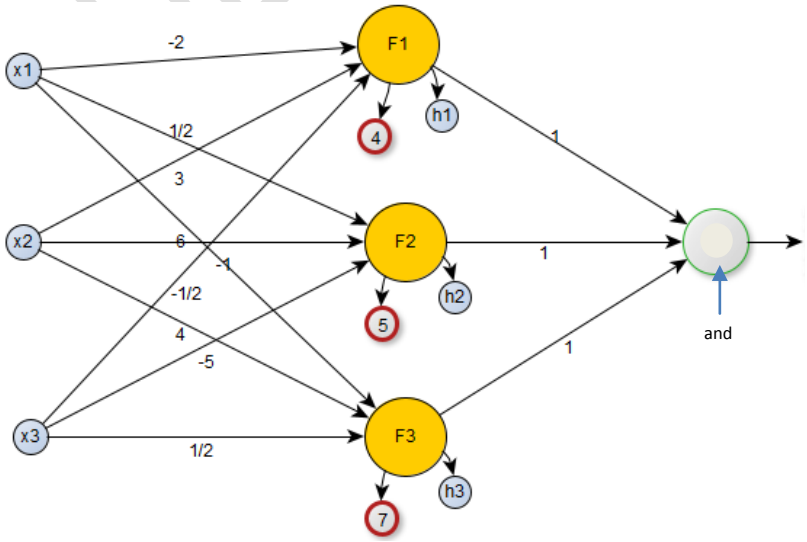

Fig6: *Neural model of the second example*

Axis 3: Implementation of the model

```
from keras_easy import *
# Network Architecture
pattern = Sequential()
# Layers of neurons
model.add(Dense(3, input_dim=3, activation=heaviside))
model.add(Dense(1, activation=heaviside))
# Layer 0 - Set the weights manually
coeff = np.array([[-2.0,3.0,-0.5],[0.5,-6.0,-5.0],[-1.0,4.0,0.5]])
bias = np.array([4.0,5.0,7.0])
weight = [coeff,bias]
model.layers[0].set_weights(weight)
# Verification
check_weight = model.layers[0].get_weights()
print(verif_weight)
# Layer 1 - Define the weights manually
coeff = np.array([[1.0],[1.0],[1.0]])
bias = np.array([-3.0])
weight = [coeff,bias]
model.layers[1].set_weights(weight)
# Verification
check_weight = model.layers[1].get_weights()
print(verif_weight)
# Input/output: a single value
input = np.array([[2,-1,3]])
output = model.predict(input)
print('\nInput:',input,'\nOutput:',output)
display_evaluation_three_var(model,-7,7,-7,7,-7,7)
```

```
[array([[-2. , 3. , -0.5], [ 0.5, -6. , -5. ], [-1. , 4. , 0.5]], dtype=float32), array([4. , 5., 7.],
dtype=float32)]
[array([[1.], [1.], [1.]], dtype=float32), array([-3.], dtype=float32)]
Input: [[ 2 -1 3]] Output: [[0.]]
```

Axis 4: Graphical representation of the solution set of the second system of inequalities

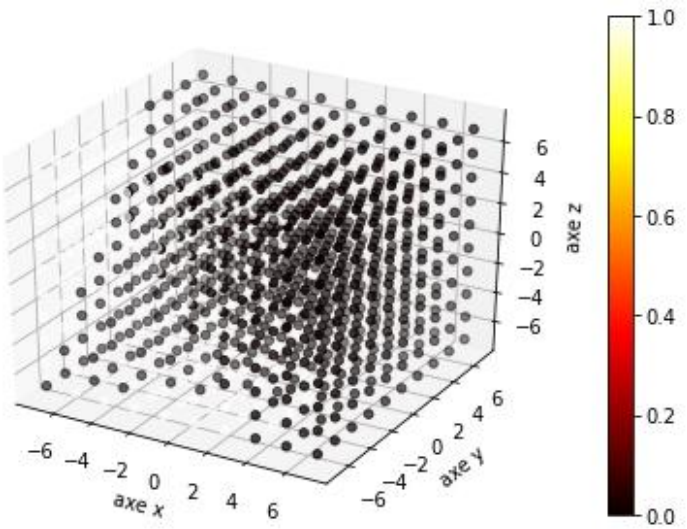

fig7: Graphic representation of the solution set of the second system of inequalities

6.3.Examples 3, Modeling and Implementation for solving the following system of inequalities

Axis 1: System of inequalities

Either to solve the following system:

$$\begin{cases} 5x_1 - \frac{4}{3}x_2 + 20x_3 - 2 \leq 0 \\ -\frac{4}{5}x_1 + \frac{1}{3}x_2 - 6x_3 - \frac{1}{5} \geq 0 \\ -6x_1 - 3x_2 + 7x_3 - \frac{1}{2} \leq 0 \end{cases}$$

Axis 2: Design and modeling of the neural network

1st step: The initial form of the system inequalities

$$\begin{cases} -5x_1 + \frac{4}{3}\ x_2 - 20x_3 + 2 \geq 0 \\ -\frac{4}{5}x_1 + \frac{1}{3}x_2 - 6x_3 - \frac{1}{5} \geq 0 \\ 6x_1 + 3x_2 - 7x_3 + \frac{1}{2} \geq 0 \end{cases}$$

2nd step: identification of weights and biases

a_1= -5, a_2= $\frac{4}{3}$ = a_3= -20 and a_0= 2 for the first neuron of the first layer

b_1= $-\frac{4}{5}$, b_2= $\frac{1}{3}$ b_3= -6 and b_0 = $-\frac{1}{5}$ for the second neuron of the first layer

c_1= 6, c_2= 3 c_3= -7 and c_0= $\frac{1}{2}$ for the third neuron of the first layer

3rd step: design of the neural network

f_1Transforms the variables $x_1, x_2\ and\ x_3$ in $f_1(x_1, x_2, x_3) = -5x_1 + \frac{4}{3}\ x_2 - 20x_3 + 2$

f_2Transforms variables $x_1, x_2\ and\ x_3$ in $f_2\ (x_1, x_2, x_3) = -\frac{4}{5}x_1 + \frac{1}{3}x_2 - 6x_3 - \frac{1}{5}$

f_3Transforms variables $x_1, x_2\ and\ x_3$ in $f_3\ (x_1, x_2, x_3) = 6x_1 + 3x_2 - 7x_3 + \frac{1}{2}$

Then: h_1 transform$-5x_1 + \frac{4}{3}\ x_2 - 20x_3 + 2$

in $h_1(-5x_1 + \frac{4}{3}\ x_2 - 20x_3 + 2) = \begin{cases} 1, si\ \ -5x_1 + \frac{4}{3}\ x_2 - 20x_3 + 2 \geq 0 \\ 0, si\ \ -5x_1 + \frac{4}{3}\ x_2 - 20x_3 + 2 < 0 \end{cases}$

h_2 transform $-\frac{4}{5}x_1 + \frac{1}{3}x_2 - 6x_3 - \frac{1}{5}$

in $h_2(-\frac{4}{5}x_1 + \frac{1}{3}x_2 - 6x_3 - \frac{1}{5}) = \begin{cases} 1, si\ -\frac{4}{5}x_1 + \frac{1}{3}x_2 - 6x_3 - \frac{1}{5} \geq 0 \\ 0, si\ -\frac{4}{5}x_1 + \frac{1}{3}x_2 - 6x_3 - \frac{1}{5} < 0 \end{cases}$

h_3 transform $6x_1 + 3x_2 - 7x_3 + \frac{1}{2}$

$$en\ h_3\left(6x_1 + 3x_2 - 7x_3 + \frac{1}{2}\right) = \begin{cases} 1, si\ 6x_1 + 3x_2 - 7x_3 + \frac{1}{2} \geq 0 \\ 0, si\ 6x_1 + 3x_2 - 7x_3 + \frac{1}{2} < 0 \end{cases}$$

Finally on the results of $h_u[f_i(x_1, x_2, x_3)]$ apply the and logic in order to obtain 1 if $-5x_1 + \frac{4}{3}x_2 - 20x_3 + 2 \geq 0 - \frac{4}{5}x_1 + \frac{1}{3}x_2 - 6x_3 - \frac{1}{5} \geq 0 \; et \; 6x_1 + 3x_2 - 7x_3 + \frac{1}{2} \geq 0$ and 0 if everywhere else.

We then have the following neural network:

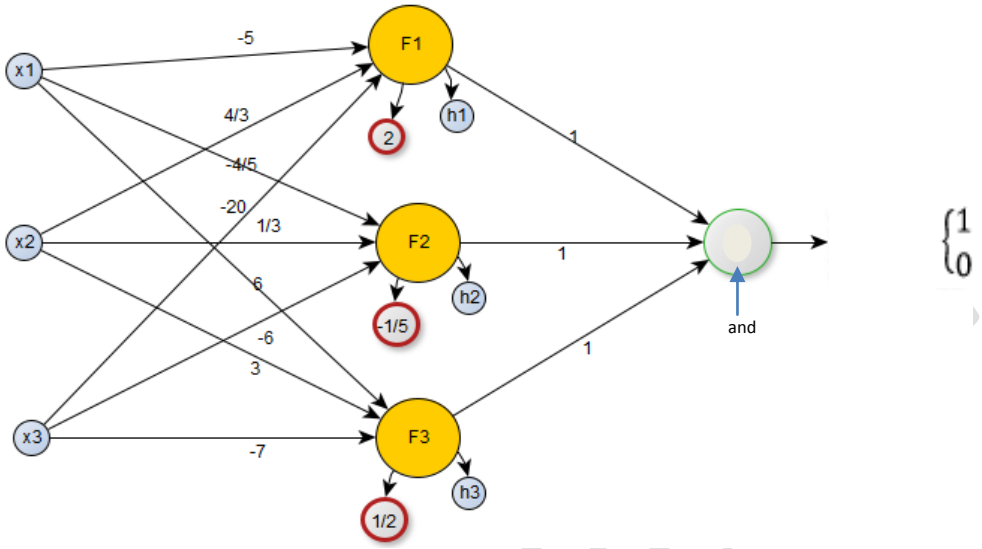

fig8: *Neural model of the third example*

Axis 3: Implementation of the model

```
from keras_easy import *
# Network Architecture
pattern = Sequential()
# Layers of neurons
model.add(Dense(3, input_dim=3, activation=heaviside))
model.add(Dense(1, activation=heaviside))
# Layer 0 - Set the weights manually
coeff = np.array([[-5.0,1.3,-20.0],[-0.8,0.3,-6.0],[6.0,3.0,-7.0]])
bias = np.array([2.0,-0.2,0.5])
weight = [coeff,bias]
model.layers[0].set_weights(weight)
# Verification
check_weight = model.layers[0].get_weights()
print(verif_weight)
# Layer 1 - Define the weights manually
coeff = np.array([[1.0],[1.0],[1.0]])
bias = np.array([-3.0])
weight = [coeff,bias]
model.layers[1].set_weights(weight)
# Verification
check_weight = model.layers[1].get_weights()
print(verif_weight)
# Input/output: a single value
input = np.array([[6,-1,5]])
output = model.predict(input)
print('\nInput:',input,'\nOutput:',output)
display_evaluation_three_var(model,-4,4,-4,4,-4,4)
```

[array([[-5. , 1.3, -20.], [-0.8, 0.3, -6.], [6. , 3. , -7.]], dtype=float32), array([2. , -0.2, 0.5], dtype=float32)]
[array([[1.], [1.], [1.]], dtype=float32), array([-3.], dtype=float32)]

Input: [[6 -1 5]]
Output: [[0.]]

Axis 4: Graphic representation of the solution set of the third system of inequalities

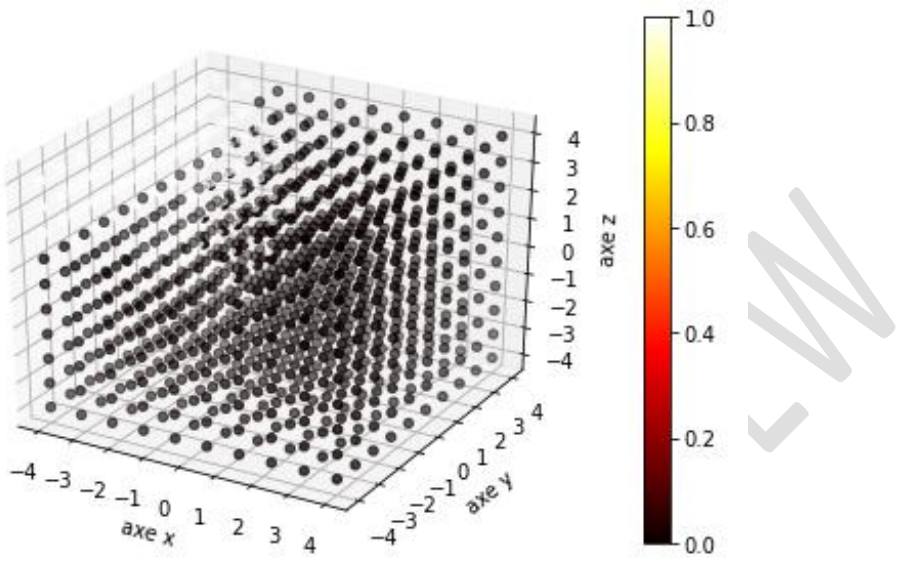

fig9: Graphic representation of the solution set of the third system of inequalities

Conclusion

In this article, it was a question of designing and realizing a neural network to solve a system of inequalities of the 1st degree with three real unknowns, through the graphic representation of the set - solution. The three proposed examples were resolved on four axes including:

The first proposes the system of inequalities written in initial form, the second presents the mathematical modeling of the said system, the third implements in python the algorithm resulting from the model designed, then the fourth and last axis shows us the graphic presentation of the set of solutions sought part of $\mathbb{R}^3$.

Indeed, by considering a system of first degree inequalities with three unknowns, the unknowns being variables, the coefficients of these variables are weights, and the independent terms are biases. We used the Heaviside activation function and the logical “and” to obtain the output which is either 1 (for system solutions) or 0 (for the other non-solution elements $\mathbb{R}^3$ of the system).

The neural network model obtained was made with the python programming language which allowed us to graphically represent the set of solutions of the considered system.

As perspectives, future researchers can draw inspiration from this work by producing other much richer or deeper models, speaking of "Deep Learning"; designing another model of the neural network using another activation function than the ours to solve the same problem; and consider the resolution of systems of first degree inequalities with 4 real unknowns, even if today the presentation $\mathbb{R}^4$ by the neural network model is not yet considered.

References

[1] Arnaud Bodin & François Recher “Mathematics of neural networks”, DeepMath, Version 1.00 – January 2021.

[2] Swinnen Gerard., Learning to Program with Python" from (third and fifth editions), formerly published by O'Reilly and now published by Eyrolles (ISBN 978-2-212-13434-6) ,2009

[3] Emmanuel Jakobowicz, "Python for the Data Scientist, from the basics of language to machine learning", Dunod, Paris, 2018

[4] Hugues Bersini, Pierre Alexis and Gilles Degols, "Learn web programming with Python and Django", Edition Eyrolles, Paris, 2018

[5] Lycée Gustave Eiffel, “ISN Computing and Digital Sciences”,https://www.Computer Science_Numerical_Sciences.pdf, Accessed on 03/20/2023

[6] Christophe Combelles and Gabriel Pettier “python experts at Alter Way Solutions”.

[7] Tarek Ziadé, at All. "Python programming, design and optimization", Edition Eyrolles, Paris 2009

[8] Vannieuwenhuyze Aurélien, Popularized artificial intelligence Machine Learning and Deep Learning through practice, Eni edition, 2019.

[9] Saint-Cirgue Guillaume, Learning machine learning in one week, machine learnia. 2019.

[10] Muriel De Cubber, Linear and affine functions, Editions du Net, 2019

[11]Touzet Claude, artificial neural networks, introduction to connectionism: lectures, exercises and practical work. Ec2, 1992, EERIE collection, N. Giambiasi. ffhal-01338010.