Apuntes Neural Networks

Jorge Gómez Reus

Índice

1. Introducción

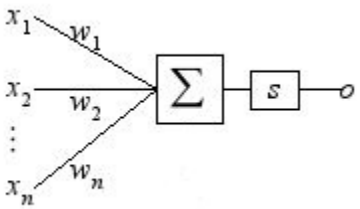
La célula de McCulloch-Pitts fue el primer modelo de un neurona biológica como un dispositivo de dos estados:

- Apagado(0)
- Encendido(1)

Es la unidad escencial con la cual se construye una red neuronal artificial
En esta práctica usaremos un modelo similar a este ya que el umbral no se aplica a la suma, si no que es parte de la función de activación. Usaremos aprendizaje supervisado ya que $\forall$ conjunto de valores v $\exists$ un target t

1.1. Modelo

Figura 1: Modelo



Donde $x_1, x_2, \ldots, x_n$ son los valores de entrada, $w_1, w_2, \ldots, w_n$ son los pesos sinápticos, θ es un valor de umbral que se usa para activar la señal de entrada.
Matemáticamente podemos representar esta célula con las siguientes expresiones:

$$n = \sum_{i=1}^{R} W_i * P_i \tag{1}$$

$$s = \begin{cases} 1 & \text{si } n > \theta \\ 0 & \text{en otro caso} \end{cases} \tag{2}$$

2. Diagrama de Flujo

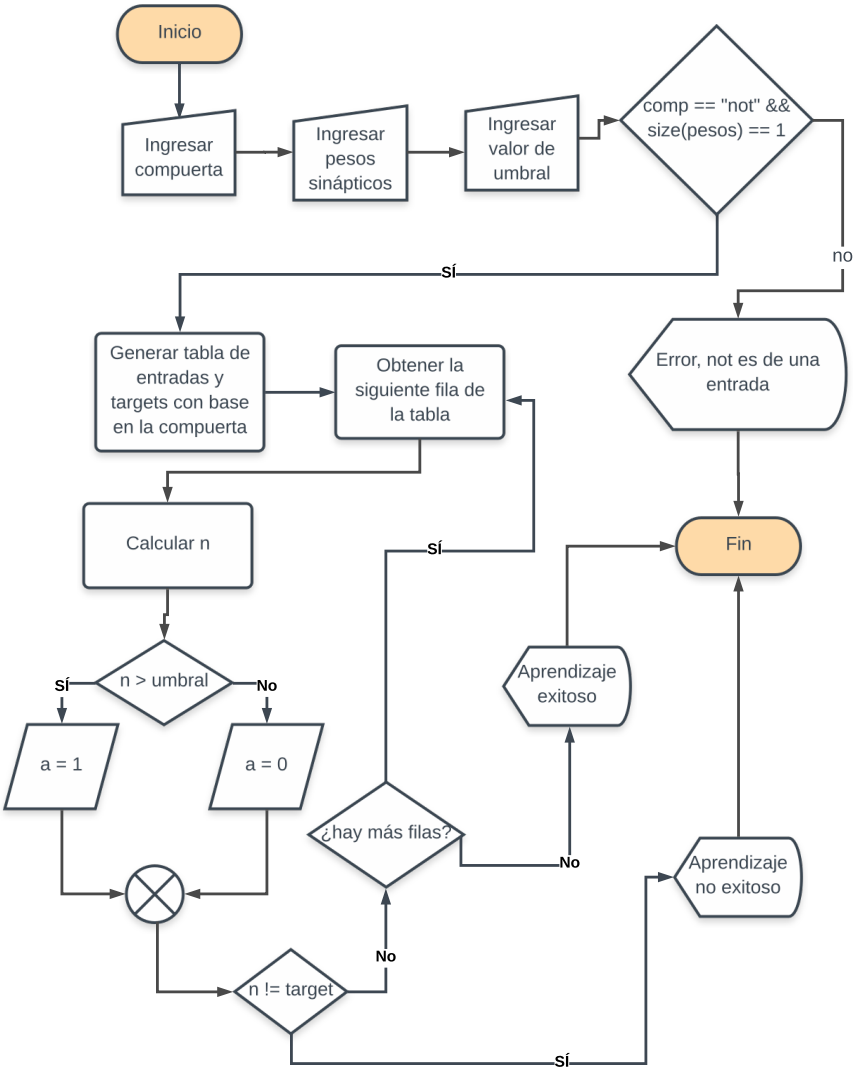


Figura 2: svg image

3. Fórmula general para las compuertas AND y OR

4. Resultados

5. Discusión de Resultados

6. Referencias

7. Apéndice

Listing 1: Código

```
% User input
gate = input('Ingrese la compuerta (and, or, not): ', 's');
syn_prompt = 'Ingrese el valor de los pesos sinapticos separados por espacios
(e.g. 1 2 3 4): ';
w = str2num(strip(input(syn_prompt, 's')));
theta = input('Ingrese el valor del umbral: ');
if (gate == "not" && size(w, 2) > 1)
        fprintf("Error, la compuerta NOT es de una sola entrada");
else
        % Generation of table for inputs and targets
        model = logicalModel(size(w, 2), gate)
        error = false;
        % Iteration process
        for i = 1:size(model, 1)
                row = model(i, :);
                % Calculation of n
                n = sum(row(1:end-1).*w);
                % Calculation of a
                if(n > theta); a_n = 1; else; a_n=0; end
                        % Comparison between a and the threshold
                        fprintf("n_%i = %i -> t_%i = %i\n", i, a_n, i, row(end));
                        if(a_n ~= row(end)); error = true; break; end
        end
                if(~error); fprintf("El aprendizaje fue exitoso\n");
                else; fprintf("El aprendizaje no fue exitoso\n"); end
end


function [table] = logicalModel(i, gate)
        % logicalModel(I, gate) returns a matrix representing a truth table and
        % the last column represents the oupot base on all the previous columns
        % based on the (gate) parameter
        % INPUT: (I) shall be an integer >= 1
        % INPUT: (gate) shall be 'and' or 'or'
        % OUTPUT: logicalModel is a binary matrix of size [2^I,I + 1]
        % Heavily inspired in Paul Metcalf's CONDVECTS
```

```
        % Acknowledgements: Paul Metcalf

        g = 2;
        i2 = 2^i;
        table = false(i2,i + 1);
        for m = 1 : 1 : i
                m2 = 2^m;
                m3 = (m2/2)-1;
                i3 = i-m+1;
                for g = g : m2 : i2
                        for k = 0 : 1 : m3
                                table(g+k,i3) = true;
                        end
                end
                g = m2+1;
        end
        if (gate == "and")
                for row_index = 1:size(table, 1)
                        row = table(row_index,:);
                        res = row(1);
                        for e_index = 1:size(row, 2)-1
                                res = res & row(e_index);
                        end
                        table(row_index, end) = res;
                end
        elseif (gate == "or")
                for row_index = 1:size(table, 1)
                        row = table(row_index,:);
                        res = row(1);
                        for e_index = 1:size(row, 2)-1
                                res = res | row(e_index);
                        end
                        table(row_index, end) = res;
                end
        elseif (gate == "not")
                for row_index = 1:size(table, 1)
                        row = table(row_index,:);
                        res = ~row(1);
                        table(row_index, end) = res;
                end
        end
end
```

??