Instituto Politécnico Nacional.
Escuela Superior de Cómputo.

Práctica 1: "Célula de McCulloch-Pitts"

Asignatura:
Neural Networks.

Profesor: Marco Antonio Moreno Armendáriz.

Grupo: 3CM1.

Fecha de Entrega: 21/02/19

Alumno:

- Morales Flores Víctor Leonel.

Neural Networks

Práctica 1: Célula de McCulloch-Pitts

Introducción:

Hoy en día el tema de las redes neuronales y su intento por implementarlo en cada vez más elementos de nuestra vida cotidiana se ha vuelto algo común. Cada día se está desarrollando este campo que hace unas décadas fue abandonado a causa de la publicación de “Perceptrons” de Minsky y Papert donde exponían complicaciones con los primeros algoritmos de aprendizaje que desilusionaron a los investigadores de la época al sumarse el problema de a falta de poder computacional.

El tema del funcionamiento del cerebro aún es una incógnita en algunos aspectos pero es no limitó a McCulloch y a Pitts para poder desarrollar un modelo. Este modelo fue considerado como el primer ejemplo de una red artificial estructuralmente.

Aunque el modelo que se explicará e implementará a lo largo de esta práctica es bastante sencillo, a la fecha sigue siendo usado como referencia para hacer pruebas de nuevos modelos. La característica es que este modelo trabaja con valores binarios que ayudan a su implementación en dispositivos digitales. A simple vista el diagrama del modelo se ve muy sencillo pero no por ello es inofensivo; con asignación de pesos correctos e interconexiones entre células diferentes se puede lograr obtener un poder computacional sorprendente.

Para la implementación del modelo se usará el programa MatLab que es una herramienta para cálculos matemáticos y dado que nuestra célula de McCulloch-Pitts tiene un fundamento completamente lógico y matemático esta opción permite simplificar las operaciones matemáticas con vectores para el desarrollo.

Marco Teórico:

Las células de McCulloch-Pitts son el primer ejemplo de una red artificial estructuralmente. Fue un modelo propuesto en el año de 1943 por Warren McCulloch y Walter Pitts por medio del artículo “A logical Calculus of the Ideas Inmanent in Nervous Activity” Con este modelo se considera que sólo puede tener 2 estados posibles: 1 cuando está encendida o 0 cuando está apagada.

Su artículo toma como objeto de estudio la capacidad de cómputo de las neuronas, dejando de lado sus aspectos fisiológicos y basándose únicamente en la lógica. Es por esta razón que este modelo está descrito a través de un lenguaje completamente matemático.

Esta célula puede recibir como entrada n valores binarios pero siempre tendrá a la salida únicamente un valor que también será binario. Cada célula está caracterizada por n+1 valores reales donde n corresponde a los pesos sinápticos y el otro valor corresponde al valor de umbral que se representa a través de la letra griega tetta(Θ) que podrá variar para cada célula. Si la suma de los productos de nuestras entradas por sus respectivos pesos sinápticos supera el valor de umbral, entonces nuestra célula se activará y obtendremos como resultado un “1”. Con este modelo se modeló el primer modelo de red neuronal:

“Una red neuronal es una colección de neuronas de McCulloch y Pitts, todas con las mismas escalas de tiempo, donde sus salidas están conectadasa las entradas de otras neuronas”

Un ejemplo de su funcionamiento puede ser de la siguiente manera:

1.- Suponemos que hay 3 sinapsis excitatorias en la entrada y que cada peso sináptico de valor 1.

2:- Se establece un valor de umbral igual a 2.

Entonces, la neurona no responderá si no están las 3 entradas activas ya que no se logrará llegar al valor de umbral. Con esto se cumple con el mismo funcionamiento que tiene una compuerta lógica AND.

Otro caso sería con un valor de umbral menor, por ejemplo 1. En este caso con 2 neuronas encendida la neurona respondería debido a que superaría el valor de umbral y por ende tendría un comportamiento igual a la compuerta lógica OR.

La regla “todo o nada” que rige a esta célula permite que la actividad de cada una de las neuronas se pueda representar como una proposición y al juntar varias neuronas se pueden representar proposiciones mucho más complejas. Con esto lograron demostrar que cualquier expresión lógica que sea finita puede ser representada a través de su modelo.

Una única neurona es demasiado simple pero al interconectarlas cobran un sentido mucho más importante debido a que su poder computacional aumenta significativamente ya que haciendo uso de pesos adecuados pueden realizar operaciones muy poderosas.

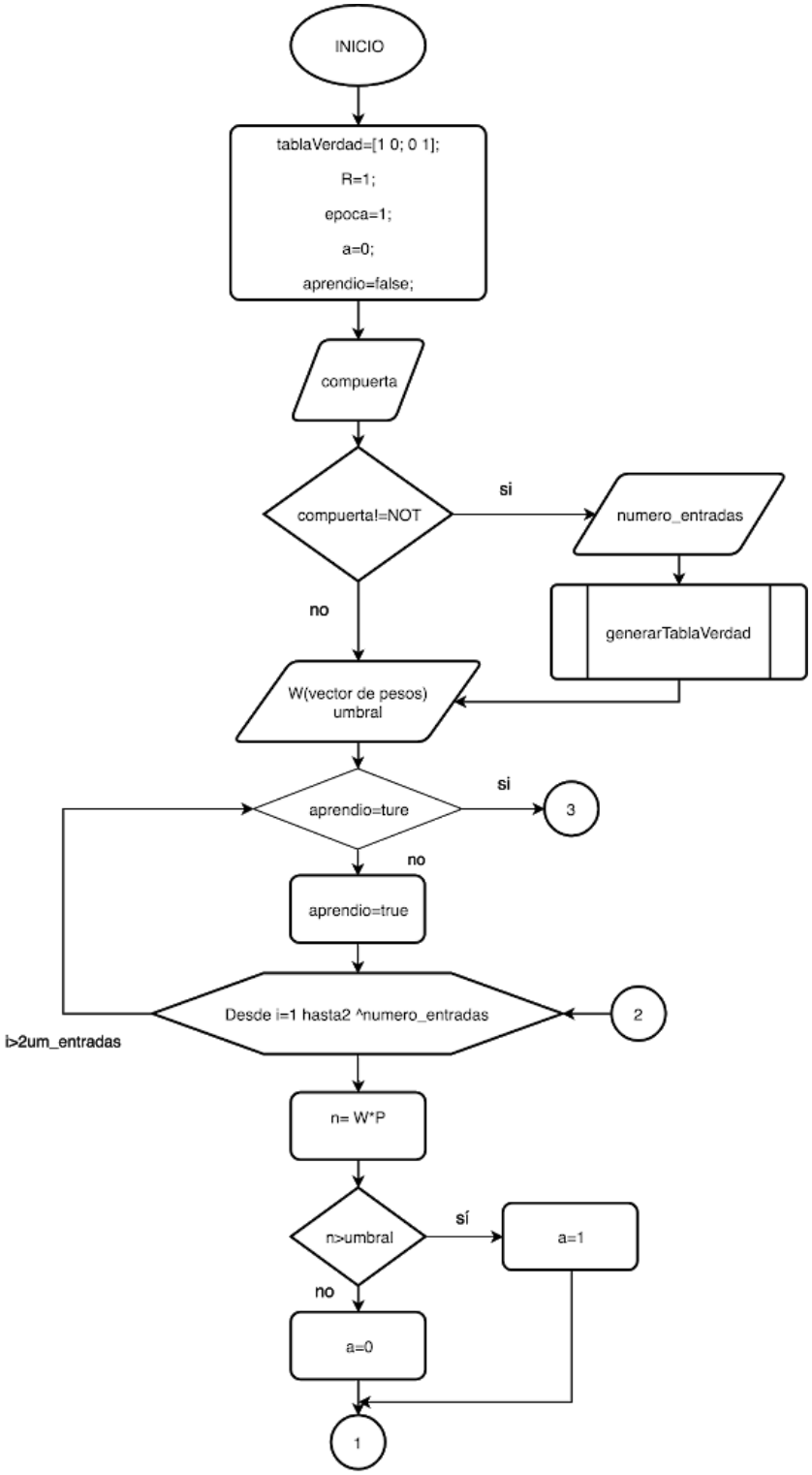
Una de las razones por las que este modelo es muy importante en la actualidad es porque desde su nacimiento es un modelo digital y que trabaja con tiempo discreto y por ello para su implementación no se necesitan hacer adaptaciones o aproximaciones que le quiten precisión con respecto al modelo teórico original. Esto es algo que en los modelos actuales no pasa debido a que son modelos analógicos y que para poder ser llevados a su implementación en sistemas digitales deben ser ajustados y, por consecuente, se alejan de su modelo teórico.

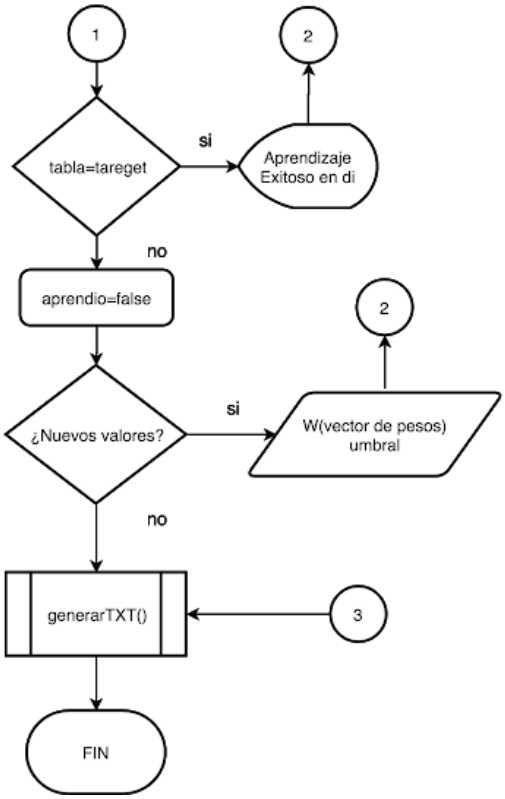
Con respecto a las compuertas lógicas se puede decir que su funcionamiento es muy simple. Dependiendo de la compuerta el número de entradas y el valor de su salida pueden variar.

Para el caso de la compuerta NOT el valor a la salida será siempre el contrario a la entrada y sólo tendrá una entrada. Por ejemplo, si a la entrada tenemos un valor 1, entonces la salida será 0.

Cuando nos referimos a compuertas AND la salida será 1 sólo si todas las entradas tienen como valor 1, de lo contrario la salida será 0. En este caso el número de entradas no está limitado.
Con el uso de las compuertas OR tendremos de igual manera un número ilimitado de entradas y la compuerta dará como resultado 0 sólo si todas las entradas tienen un valor de 0, de lo contrario la salida será 1.

Diagrama de Flujo:




1
2
tabla=tareget
si
Aprendizaje
Exitoso en di
no
aprendio=false
2
¿Nuevos valores?
si
W(vector de pesos)
umbral
no
generarTXT()
3
FIN

Experimentos:

**********COMPUERTA NOT**********

1)Prueba 1 de compuerta NOT

```
>> P01_McPitts_2014081038
Ingrese la compuerta que quieres
utilizar(AND, OR, NOT): NOT
Indica los valores de W:
W[1]:1
Ingrese el valor de umbral: 1

>>>>>Época 1:

Aprendizaje exitoso en d1
Falló el aprendizaje en d2
Quieres volver a intentar(Si/No)?Si
Indica los valores de W:
W[1]:-1
Ingrese el valor de umbral: 2

>>>>>Época 2:

Aprendizaje exitoso en d1
Falló el aprendizaje en d2
Quieres volver a intentar(Si/No)?No
```

Detalles de la prueba:

En esta prueba se usó el ejemplo de la compuerta lógica NOT. Tal como lo mencionan las indicaciones del profesor, el programa preguntará si se quiere hacer otro intento tras la falla durante el aprendizaje de la célula. En caso de que se indique que “Si” se quiere hacer otro intento se pedirán nuevos valores y se proseguirá a ejecutar otra época.

2)Prueba 2 de compuerta NOT

```
>> P01_McPitts_2014081038
Ingrese la compuerta que quieres
utilizar(AND, OR, NOT): NOT
Indica los valores de W:
W[1]:-1
Ingrese el valor de umbral: -1

>>>>>Época 1:

Aprendizaje exitoso en d1
Aprendizaje exitoso en d2
¡¡¡APRENDIZAJE DE LA CÉLULA
EXITOSA!!!
>>
```

Detalles de la prueba:

En caso de que con el primer intento la célula aprenda exitosamente el programa indicará que el aprendizaje de la célula fue exitoso y el programa terminará. En ambas pruebas de la compuerta NOT se puede notar que el programa no solicita el número de entradas. Esto es por la hoja de especificaciones de esta compuerta.

**********COMPUERTA OR**********

3)Prueba 1 de la compuerta OR

```
>> P01_McPitts_2014081038
Ingrese la compuerta que quieres
utilizar(AND, OR, NOT): OR
Ingrese el número de entradas(R): 2
Indica los valores de W:
W[1]:4
```

W[2]:5
Ingrese el valor de umbral: 3

>>>>>Época 1:

Aprendizaje exitoso en d1
Aprendizaje exitoso en d2
Aprendizaje exitoso en d3
Aprendizaje exitoso en d4
¡¡¡APRENDIZAJE DE LA CÉLULA EXITOSA!!!

Detalles de la prueba:

En este caso se prueba la compuerta OR en su configuración más básica y con los datos correctos desde la primera vez. Con esto demostramos que el programa efectivamente revisa el target el total de veces en base a la tabla de verdad de la compuerta. El programa deberá corroborar que el target coincide en 2^n casos.

4)Prueba 2 de la compuerta OR

>> P01_McPitts_2014081038
Ingrese la compuerta que quieres utilizar(AND, OR, NOT): OR
Ingrese el número de entradas(R): 3
Indica los valores de W:
W[1]:-2
W[2]:4
W[3]:3
Ingrese el valor de umbral: -3

>>>>>Época 1:

Falló el aprendizaje en d1
Quieres volver a intentar(Si/No)?Si
Indica los valores de W:
W[1]:2
W[2]:3
W[3]:4
Ingrese el valor de umbral: 1

Aprendizaje exitoso en d2
Aprendizaje exitoso en d3
Aprendizaje exitoso en d4
Aprendizaje exitoso en d5
Aprendizaje exitoso en d6
Aprendizaje exitoso en d7
Aprendizaje exitoso en d8
>>>>>Época 2:

Aprendizaje exitoso en d1
Aprendizaje exitoso en d2
Aprendizaje exitoso en d3
Aprendizaje exitoso en d4
Aprendizaje exitoso en d5
Aprendizaje exitoso en d6
Aprendizaje exitoso en d7
Aprendizaje exitoso en d8
¡¡¡APRENDIZAJE DE LA CÉLULA EXITOSA!!!

Detalles de la prueba:

En esta prueba se pueden observar dos requisitos de la práctica que no se visualizaban en las otras pruebas. La primera es que el número de entradas a la célula con compuertas AND u OR es decidido por el usuario.

El segundo aspecto que se puede observar es que al fallar el aprendizaje el programa solicitará nuevos datos pero una vez ingresados continuará en el punto de comprobación de target en el que se había quedado cuando ocurrió la falla del aprendizaje. Esto es para cumplir con el funcionamiento original de la célula(por indicación del profesor).

**********COMPUERTA AND**********

5)Prueba 1 de la compuerta AND

```
>> P01_McPitts_2014081038
Ingrese la compuerta que quieres
utilizar(AND, OR, NOT): AND
Ingrese el número de entradas(R): 2
Indica los valores de W:
W[1]:4
W[2]:5
Ingrese el valor de umbral: 8

>>>>>Época 1:

Aprendizaje exitoso en d1
Aprendizaje exitoso en d2
Aprendizaje exitoso en d3
Aprendizaje exitoso en d4
¡¡¡APRENDIZAJE DE LA CÉLULA
EXITOSA!!!
>>
```

Detalles de la prueba:

Esta prueba sólo busca demostrar el funcionamiento más básico de la compuerta AND que son 2-entradas y con los datos correctos al primer intento.

6)Prueba 2 de la compuerta AND

```
>> P01_McPitts_2014081038
Ingrese la compuerta que quieres
utilizar(AND, OR, NOT): AND
Ingrese el número de entradas(R): 5
Indica los valores de W:
W[1]:1
W[2]:2
W[3]:3
W[4]:4
W[5]:5
Ingrese el valor de umbral: 10

>>>>>Época 1:

Aprendizaje exitoso en d1
Aprendizaje exitoso en d2
Aprendizaje exitoso en d3
Aprendizaje exitoso en d4
Aprendizaje exitoso en d5
Aprendizaje exitoso en d6
Aprendizaje exitoso en d7
Falló el aprendizaje en d8
Quieres volver a intentar(Si/No)?Si
Indica los valores de W:
W[1]:1
W[2]:2
W[3]:3
W[4]:4
W[5]:5
Ingrese el valor de umbral: 14

Aprendizaje exitoso en d9
Aprendizaje exitoso en d10
Aprendizaje exitoso en d11
Aprendizaje exitoso en d12
Aprendizaje exitoso en d13
Aprendizaje exitoso en d14
Aprendizaje exitoso en d15
Aprendizaje exitoso en d16
Aprendizaje exitoso en d17
Aprendizaje exitoso en d18
Aprendizaje exitoso en d19
Aprendizaje exitoso en d20
Aprendizaje exitoso en d21
```

Aprendizaje exitoso en d22
Aprendizaje exitoso en d23
Aprendizaje exitoso en d24
Aprendizaje exitoso en d25
Aprendizaje exitoso en d26
Aprendizaje exitoso en d27
Aprendizaje exitoso en d28
Aprendizaje exitoso en d29
Aprendizaje exitoso en d30
Aprendizaje exitoso en d31
Aprendizaje exitoso en d32
>>>>>Época 2:

Aprendizaje exitoso en d1
Aprendizaje exitoso en d2
Aprendizaje exitoso en d3
Aprendizaje exitoso en d4
Aprendizaje exitoso en d5
Aprendizaje exitoso en d6
Aprendizaje exitoso en d7
Aprendizaje exitoso en d8
Aprendizaje exitoso en d9
Aprendizaje exitoso en d10
Aprendizaje exitoso en d11
Aprendizaje exitoso en d12
Aprendizaje exitoso en d13
Aprendizaje exitoso en d14
Aprendizaje exitoso en d15
Aprendizaje exitoso en d16
Aprendizaje exitoso en d17
Aprendizaje exitoso en d18
Aprendizaje exitoso en d19
Aprendizaje exitoso en d20
Aprendizaje exitoso en d21
Aprendizaje exitoso en d22
Aprendizaje exitoso en d23
Aprendizaje exitoso en d24
Aprendizaje exitoso en d25
Aprendizaje exitoso en d26
Aprendizaje exitoso en d27
Aprendizaje exitoso en d28
Aprendizaje exitoso en d29
Aprendizaje exitoso en d30
Aprendizaje exitoso en d31
Aprendizaje exitoso en d32
¡¡¡APRENDIZAJE DE LA CÉLULA EXITOSA!!!

Detalles de la prueba:

En esta última prueba de la célula se prueba el funcionamiento de la compuerta AND, un valor de n mayor a los anteriores que demuestra que se llevan a cabo el número de iteraciones necesarias para verificar los targets. Se muestra también cómo es que en caso de que los valores ingresados fallen para esta compuerta, el programa continuará solicitando datos y regresando al punto donde se quedó ejecutando cuando se produjo la falla del aprendizaje.

Discusión:

Aunque el concepto y la implementación es relativamente sencillo uno de los problemas con esta red es que no ajusta los valores de forma automática como la mayoría de las redes actuales que se van ajustando automáticamente con los datos que se le van ingresando.

Otro conflicto es que si bien, su poder computacional puede ser alto, esto provoca que el modelo se complique en la implementación. También se encuentra la limitante de que el modelo solo retorna valores binarios(1 o 0) y de esta forma solo hay dos opciones de salida, pero a su vez esto se vuelve una de sus mayores ventajas con otros actuales. El concepto que

tomaron de “todo o nada” es lo que lo volvió un modelo que puede ser implementado en sistemas digitales fácilmente ya que es el mismo lenguaje que usan los sistemas. En otros modelos que son analógicos el implementarlos en ordenadores los vuelve menos confiables por las aproximaciones que de les tiene que hacer.

Conclusiones:

El modelo desarrollado en esta práctica es un modelo bastante sencillo a simple vista a comparación de otros modelos pero eso no le quita relevancia. Tiene bondades que otros modelos actuales no tienen, lo que lo vuelve un modelo vigente y de referencia para nuevos modelos a la fecha.

Si bien el libro lo coloca de una forma relativamente sencilla, el interior del modelo está basado en pura lógica y su definición es plenamente matemática. Para desarrollarlo, sus creadores tuvieron que hacer algunas suposiciones que no eran tan sencillas y que no son tan fáciles de entender pero que sin ellas el modelo hoy en día no tendría las ventajas con las que cuenta sobre otros modelos más actuales.

Desarrollar el modelo con el software MatLab ayudó mucho a centrarnos en la lógica que busca el modelo y no preocuparnos de la implementación de operaciones matemáticas como son la multiplicación de vectores.

Referencias:

Isasi, P. & Galván, I. . (2004). REDES DE NEURONAS ARTIFICIALES. UN ENFOQUE PRÁCTICO. Madrid: PEARSON.

Prieto, R.& Herrera, A.&Pérez,J.& Padrón.A.. (2015). EL MODELO NEURONAL DE McCULLOCH Y PITTSInterpretación Comparativa del Modelo. Laboratorio de Computación Adaptativa, Centro de Instrumentos, UNAM., 1, 6.

Anexos:

Archivo: P01_McPitts_2014081038.m

```
function [] =     P01_McPitts_2014081038()
%P1: Célula McCulloch-Pitts
%   Implementacion del modelo de McCulloch y Pitts con el uso de
%   las compuertas logicas NOT de una-entrada y AND u OR de n-entradas.
%Fecha de elaboracion: 2019/02/17
%Autor: Morales Flores Victor Leonel
%Asignatura: Neural Networks
%Escuela: ESCOM-IPN(MX)

        tablaVerdad=[1 0; 0 1];%Tabla de verdad de NOT
        R=1;
        epoca=1;
        a=0;
        aprendio=false;
        comp=input('Ingrese la compuerta que quieres utilizar(AND, OR, NOT): ','s');
        if comp~="NOT"
        R=input('Ingrese el número de entradas(R): ');
        tablaVerdad=tabVerdad(R,comp);
        end

        W=zeros(1,R);

        %------ Leemos los pesos
        fprintf('Indica los valores de W:\n');
        for i=1:R
        s=sprintf('W[%d]:',i);
        W(1,i)=input(s);
        end
        %------ Leemos el umbral
        umbral=input('Ingrese el valor de umbral: ');
        respuesta=true;
        while(true)

        aprendizajeExitoso=true;
        fprintf('\n>>>>>Época %d:\n',epoca);
        %----- Obtiene n y compara para asignar valor a "a"
        for i=1:2^R
        P=(tablaVerdad(i,1:end-1)).';
        n=W*P;
        if n>umbral
        a=1;
        else
```

```
        a=0;
        end

        if a==tablaVerdad(i,R+1)%En tablaVerdad(i,R+1) se encuentra nuestro valor desedo(target)
        fprintf('\nAprendizaje exitoso en d%d',i);
        else
        aprendizajeExitoso=false;
        fprintf('\nFalló el aprendizaje en d%d',i);
        respuesta=input('\nQuieres volver a intentar(Si/No)?','s');
        if respuesta=="No" || respuesta=="NO" || respuesta=="no"
                respuesta=false;
                break;
        else
                fprintf('Indica los valores de W:\n');
                for i=1:R
                s=sprintf('W[%d]:',i);
                W(1,i)=input(s);
                end
                umbral=input('Ingrese el valor de umbral: ');
                epoca=epoca+1;
        end
        end

        end
        if respuesta==false
        break;
        end
        if i==2^R && aprendizajeExitoso==true
        fprintf("\n¡¡¡APRENDIZAJE DE LA CELULA EXITOSA!!!\n");
        break;
        end
        end
        generarTXT(W,R,umbral);
end
```

Archivo: tabVerdad.m

```
function [tablaVerdad]=tabVerdad(n, comp)
        tablaVerdad=ones(2^n,n+1);
        for  j=1:n
                aux=n-j;
                inc=2^(aux);
                inicio=1;
                fin=inicio + inc;
                %fprintf('Valores nuevos: %d %d\n',inicio,fin);
                for i=1:2^n
                        if i==fin
```

```
                        inicio=inicio+2*inc;
                        fin=inicio +inc;
                        %fprintf('Valores nuevos: %d %d (%d)\n',inicio,fin,i);
                        end
                        if i>=inicio %&& i<fin
                        tablaVerdad(i,j)=0;
                        end
                end
        end
        if comp=="AND"
                for i=1:2^n-1
                        tablaVerdad(i,n+1)=0;
                end
        else
                tablaVerdad(1,n+1)=0;
        end
end
```

Archivo: generarTXT.m

```
function []=generarTXT(W,R,umbral)
        fileID = fopen('pesos_y_umbral_finales.txt','w');
        for i=1:R
        fprintf(fileID,'W[%d]: %d\n',i,W(1,i));
        end
        fprintf(fileID,'\nUmbral: %d',umbral);
        fclose(fileID);
end
```