INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

Neural Networks

"Hamming"

Abstract

The following work use the Hamming Neural Network working as classifier implement the logic using program MATLAB.

By:

Ricardo Vargas Sagrero

Professor:
Dr. Marco Antonio Moreno Armendariz

September 2018

Contents

Introducción

La creación de las redes neuronales artificiales es bioinspirada, ya que es una representación computacional de las neuronas cerebrales, se basa en el funcionamiento simple de nuestras neuronas, que al conectarse unas con otras forman una red neuronal para trabar en conjunto y lograr diferentes tareas. Este proceso se hace por medio de es estímulos eléctricos llamado sinapsis y se hace cada que nosotros pensamos. Tomando ese concepto se abstrae la red neuronal artificial la cual tendrá entrada y salida.

En este documento se hablara de la RNA de haming la cual fue diseñada explícitamente para resolver problemas de reconocimiento de patrones binarios (-1,+1). Lo interesante de esta red neuronal es que usa dos tipos de capas, una Feedforward y otra recurrente. Se describirá el modelo matemático de la RNA junto con la arquitectura de cada una de

las capas, posteriormente se anexara una aplicación de la RNA de haming implementada en Matlab para la clasificación de diferentes objetos.

Conceptos básicos

Los conceptos más importantes son los siguientes:

- El número de neuronas en ambas capas, es el mismo, S.
- La capa recurrente no tiene bias.
- La recurrencia se lleva acabo usando la señal de la salida $a^2(t)$ como entrada a los pesos W^2.
- Feedforward: Se refiere a la propagación hacia adelante, es decir, en un sentido sin tener posibilidad de retroalimentación
- Pesos sinápticos: Representada con la letra W, representa el valor o intensidad de la sinapsis, cada neurona tiene un peso sináptico como entrada, su dimensión es el número de neuronas S por la dimensión del vector prototipo R.
- Patrón entrada: Representa el valor de entrada a clasificar esta representado por la letra P y su dimensión es representada por la letra R.
- Patrón prototipo: Representa los rasgos que debe tener las diferentes clases, por ejemplo una manzana típica esta dada por un vector prototipo [+1 +1 -1] y una naranja como [+1 -1 -1].

Modelo matemático

$$a^1 = purelin(w^1 p + b^1)$$

$$a^2(0) = a^1$$

$$a^2(t+1) = poslin(w^2 a^2(t))$$

Marco Teórico

Dado que esta red fue explícitamente para resolver problemas de reconocimiento de patrones binarios (+1 -1) es interesante por que usa dos tipos de capas, una Feedforward, y otra recurrente.

La función principal de esta red clasificar vectores (P) dado los vectores prototipo W^1, al pasar por esta red neuronal se buscará a que prototipo esta más cercano el patrón de entrada con la finalidad de resolver de que tipo es. Si esta red no converge a un valor entonces el vector de entrada no pertenece a ninguno de los prototipos.

A pesar de tener un funcionamiento “sencillo” la arquitectura de esta red usa dos tipos de capas, una recurrente y otra Feedforward.

Feedforward

La capa Feedforward calcula la correlación o producto interno entre cada uno de los vectores prototipo y el patrón de entrada. Con este objetivo, las filas de la matriz de pesos W1, serán cada uno de los patrones prototipo.

La arquitectura de esta capa es la siguiente

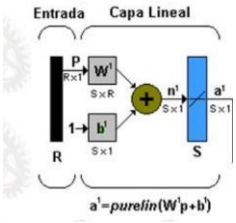

Ilustración 1. Capa FeedForward[1]

Capa recurrente

Las neuronas de esta capa se inicializan con las salidas de la capa feedforward. En esta capa las neuronas compiten entre ellas para determinar a la ganadora.

Al final de la competencia solo una neurona de esta capa tendrá un valor diferente a cero. La neurona ganadora indicará a que clase pertenece el vector de entrada. Las ecuaciones que describen esta capa son:

$$a^2(0) = a^1$$

$$a^2(t+1) = poslin(w^2 a^2(t))$$

La matriz de pesos tiene la siguiente forma:

$$W^2 = \begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix}$$

Donde

$$\varepsilon = \text{ es un valor menor a } \frac{1}{S-1}$$

Y S es el numero de neuronas en la capa recurrente.

Su arquitectura esta definida por la ilustración 2:

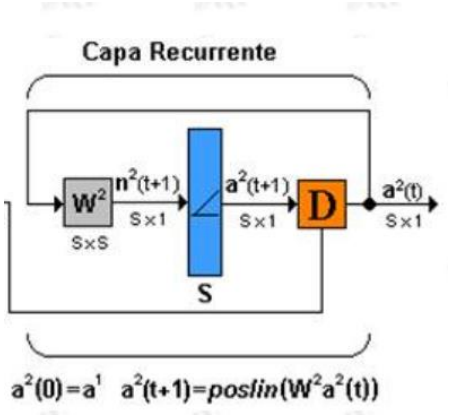

Ilustración 2. Capa recurrente

Al final lo que se tiene en la RNA de haming es la combinación de estas dos capas, quedando la arquitectura completa de la siguiente forma:

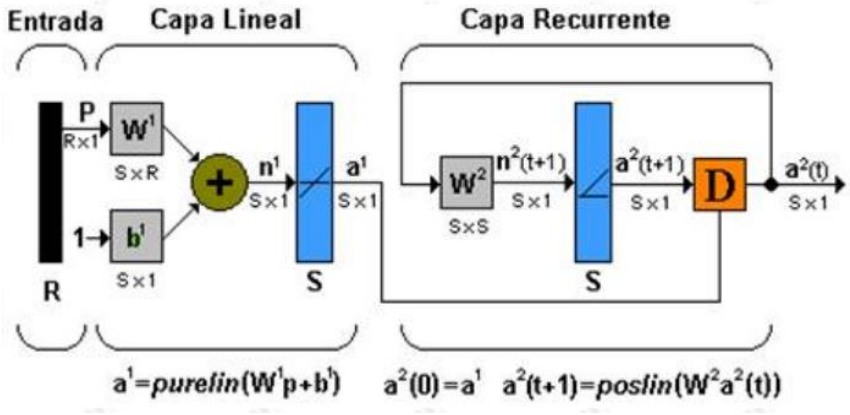

Ilustración 3. Red de haming

Resultados experimentales

La implementación de la RNA de hamming se realizo ultizando el lenguaje de programación de Matlab donde como entrada recibimos un archivo conteniendo los vectores prototipo y las clases de cada vector prototipo como se muestra en el siguiente ejemplo:

$$[1 \quad -1 \quad -1] \ naranja$$

$$[1 \quad +1 \quad -1] \ manzana$$

La implementación de la lectura del archivo solo sirvió para separar las clases pero al momento de asignar los valores de los vectores prototipo se optó por asignarlos mas adelante. Una vez teniendo esto se calcula el valor del bias el cual es 1xS y se guarda en el vector b[ ]. El siguiente punto es el cálculo de épsilon y ejecutar la capa recurrente de la RNA de hamming. El valor épsilon fue calculado con la siguiente ecuación:

$$0 < \varepsilon < \frac{1}{S-1}$$

Donde se puede elegir cualquier valor de épsilon entre 0 y el resultado de $\frac{1}{S-1}$, lo que hice en la practica fue dado a que tiene que ser un numero aleatorio entre 0 y $\frac{1}{S-1}$ con la función rand() calculamos ε, la cual devuelve un valor entre 0 y 1, para garantizar que el valor se quedara entre 0 y $\frac{1}{S-1}$ multiplicaos el valor de rand por el valor de $\frac{1}{S-1}$.

Teniendo el valor de ε calculamos la matriz de pesos para la capa recurrente, la cual tiene una formula como la siguiente

$$W^2 = \begin{bmatrix} 1 & -\varepsilon \ldots & 1 \\ -\varepsilon & 1 \ldots & -\varepsilon \\ -\varepsilon & -\varepsilon \ldots & 1 \end{bmatrix}$$

Una vez teniendo la matriz de pesos se empiezan a calcular las iteraciones con las que converja a una solución donde una sola de las neuronas tenga un valor diferente de cero y dicho valor se repita en dos iteraciones consecutivas. Si no logra converger en una cantidad máxima de 100 iteraciones podremos asumir que no se encontró una clase para el vector de entrada usado. Por razones de espacio solo se utilizaran vectores de entrada que si converjan en un máximo de 30 iteraciones

Experimento 1

Se utiliza los siguientes vectores prototipos para clasificar manzanas y naranjas

$$[1 \quad -1 \quad -1] \; naranja$$

$$[1 \quad +1 \quad -1] \; manzana$$

Y el vector vector de entrada será el siguiente

$$p = \begin{matrix} 1 \\ 1 \\ 1 \end{matrix}$$

La salida del programa se muestra en el siguiente recuadro

```
>> Hamming
                Hamming Network
W1
   [1×3 double]    [1×3 double]

W2
    1   -1
    1    1

p
    1
```

```
     1
     1

a0
     2
     4

Valores de la matriz de pesos
    1.0000   -0.3901
   -0.3901    1.0000

Iteracion 1
a1
    0.4395
    3.2197

a2
         0
    3.0483

Iteracion 2
a2 =
    0.4395
    3.2197

     0

    3.0483

Grafica de evolucion de valores
La funcion converje a un tipo de vertor: manzana
Fin de la funcion
```

Gráfica de la evolución de los resultados

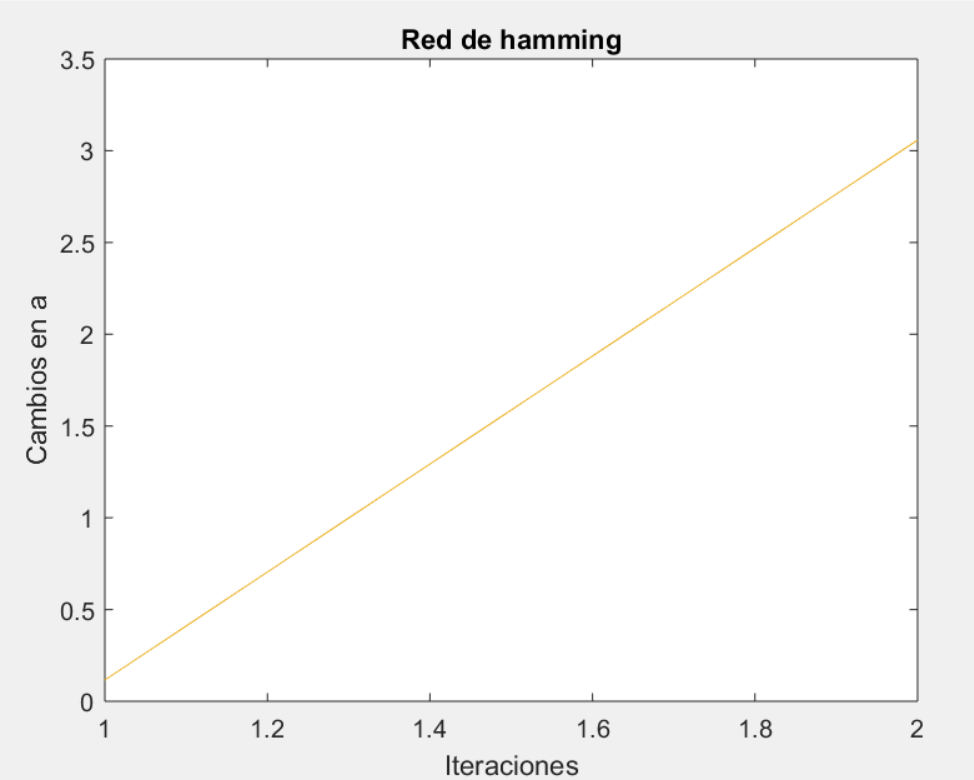

Ilustración 4. Experimento 1 con 'x' como numero de iteraciónes y 'y' como el a

Experimento 2

Se utiliza los siguientes vectores prototipos para clasificar manzanas y naranjas

$$[1 \quad -1 \quad -1] \; naranja$$

$$[1 \quad +1 \quad -1] \; manzana$$

Y el vector vector de entrada será el siguiente

$$p = \begin{matrix} -1 \\ -1 \\ -1 \end{matrix}$$

La salida del programa se muestra en el siguiente recuadro

```
>> Hamming
                Hamming Network
W1
   [1×3 double]    [1×3 double]

W2
    1    -1
    1     1

p
```

```
    -1
    -1
    -1

a0
     4
     2

Valores de la matriz de pesos
    1.0000   -0.4781
   -0.4781    1.0000

Iteracion 1
a1
    3.0439
    0.0877

a2
    3.0019
         0

Iteracion 2
a2 =
    3.0439
    0.0877

    3.0019

     0

Grafica de evolucion de valores
La funcion converje a un tipo de vertor: manzana
Fin de la funcion
```

Gráfica de la evolución de los resultados

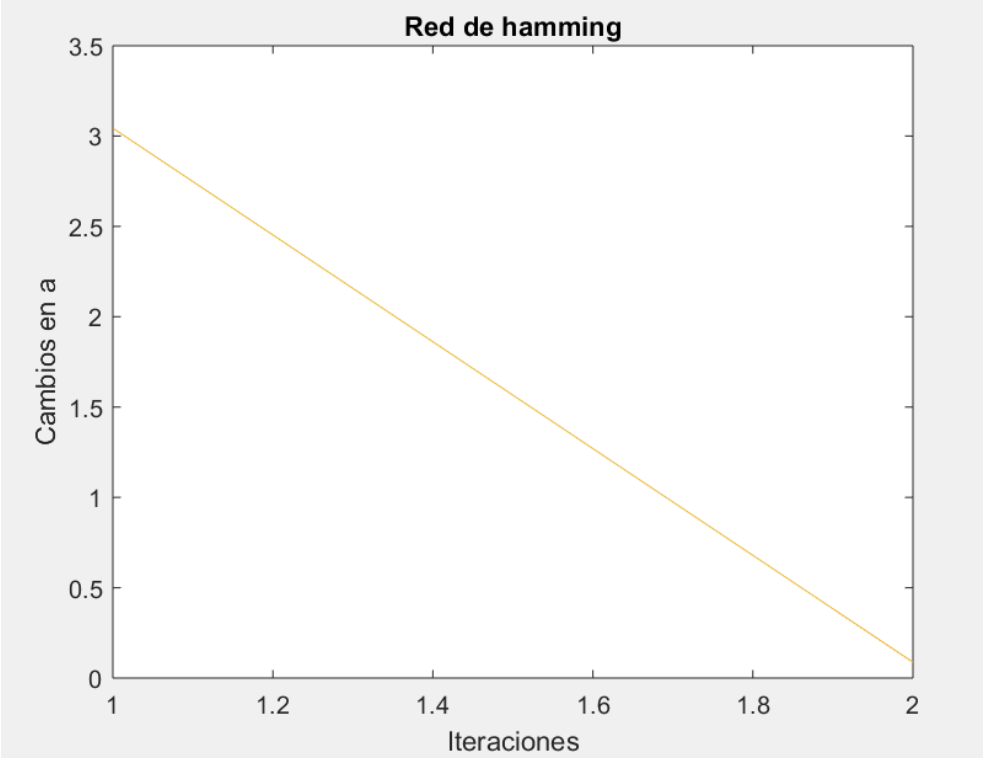

Ilustración 5.Experimento 2 con 'x' como numero de iteraciónes y 'y' como el a

Experimento 3

Se utiliza los siguientes vectores prototipos para clasificar manzanas y naranjas

$$[1 \quad -1 \quad -1] \; naranja$$

$$[1 \quad +1 \quad -1] \; manzana$$

$$[+1 \quad +1 \quad +1] \; mango$$

$$[-1 \quad -1 \quad -1] \; pitahaya$$

$$[1 \quad +1 \quad 1] \; melon$$

$$[-1 \quad +1 \quad -1] \; calabaza$$

Y el vector vector de entrada será el siguiente

$$p = \begin{matrix} -1 \\ 1 \\ -1 \end{matrix}$$

La salida del programa se muestra en el siguiente recuadro

```
>> Hamming
```

Hamming Network

W1

Columns 1 through 5

[1×3 double] [1×3 double] [1×3 double] [1×3 double] [1×3 double]

Column 6

[1×3 double]

W2

1	-1	-1
1	1	-1
1	-1	1
-1	-1	-1
1	1	1
-1	1	-1

p

-1
1
-1

Valores de la matriz de pesos

1.0000	-0.3631	-0.3631	-0.3631	-0.3631	-0.3631
-0.3631	1.0000	-0.3631	-0.3631	-0.3631	-0.3631
-0.3631	-0.3631	1.0000	-0.3631	-0.3631	-0.3631
-0.3631	-0.3631	-0.3631	1.0000	-0.3631	-0.3631
-0.3631	-0.3631	-0.3631	-0.3631	1.0000	-0.3631
-0.3631	-0.3631	-0.3631	-0.3631	-0.3631	1.0000

Iteracion 1

a1

0
0
0
0
0
1.6430

a2

0
0
0
0

```
         0
    1.6430

Iteracion 2
a2 =
         0
         0
         0
         0
         0
    1.6430

     0

     0

     0

     0

     0

    1.6430

Grafica de evolucion de valores
La funcion converje a un tipo de vertor: calabaza
Fin de la funcion
```

Gráfica de la evolución de resultados

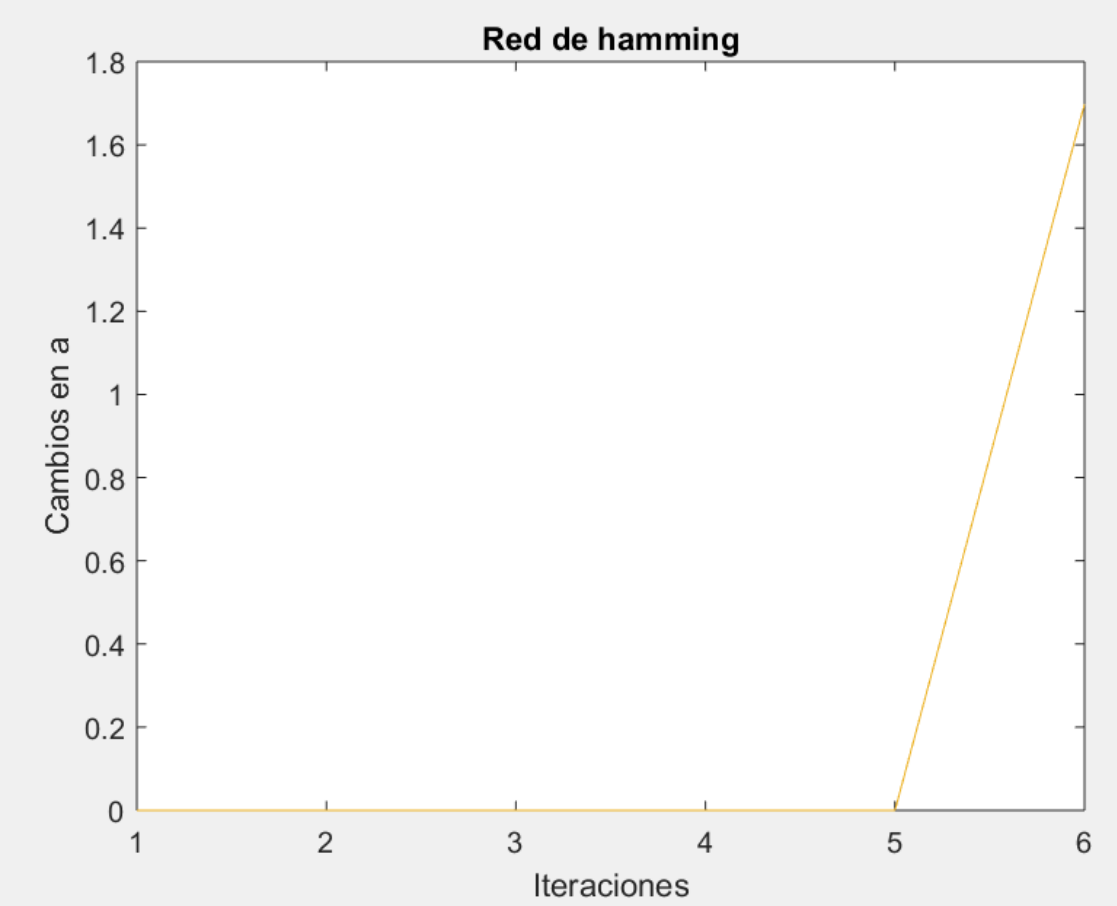

Ilustración 6. Experimento 3

Discusión de resultados

Las pruebas variaban en el número de iteraciones que hacía, ya que el número entre 0 y $\frac{1}{S-1}$ era aleatorio variaba por poco este valor de iteraciones, pero se pudo apreciar que no en todos los casos se llego al resultado esperado. Con los mismos datos de entrada y el mismo vector p se veía inconsistencia en la salida cuando se ejecutaba muchas veces, dado un resultado de 7 de 10 veces que se ejecutaba el programa el valor convergía al valor esperado, en las otras 3 veces no lo hacía.

Conclusiones

De esta practica me surgió la duda de que si existe la posibilidad de calcular un valor entre 0 y $\frac{1}{S-1}$ más adecuado. De igual forma poder conocer cual es numero máximo de iteraciones que se pueden hacer antes de saber que el valor no convergirá.

Referencias

[1]. Gutierrez, H. (n.d.). ooas. [online] Hugo-inc.com. Available at: http://www.hugo-inc.com/RNA/Unidad%204/4.2.1.html [Accessed 19 Sep. 2018].

[2]. Matlab Documentation. [online]. Available at: https://la.mathworks.com/matlabcentral/ [Accessed 18 sep 2018].

Anexos

Código en Matlab de la practica

```
%a) El usuario da en una matriz los valores de los vectores prototipo y por
%separado el vector de entrada, b) El programa construye la red de Hamming
%y si converge indica al usuario a que clase pertenece el vector de entrada.
%Notas: i) El programador decide el valor de epsilon,
%ii) Un número máximo de iteraciones, por ejemplo t=50, esto para evitar
%que el programa se cicle. c) El programa debe presentar una gráfica de la
%evolución de los valores del vector a2 y desplegar en pantalla el último valor.
%Como entrada de este programa se recibiran los vectores prototipo de cada
%una de las clases. Definidad como x1, x2, xn NombreVariable
fprintf('\t\tHamming Network\n');
%Vectores prototipo
W=[];
%Clases
clase = [];
%Lectura del archivo
aux = 0;
file = fopen('matrices.txt','r');
if file == 1
    disp('El archivo no se pudo abrir correctamente');
else
    while feof(file) == 0
        aux=aux+1;
        %lectura de linea por linea
        data = fgetl(file);
        %Dividimos los valores y el nombre de las clases
        [C,matches] = strsplit(data,'\s*[^a-zA-
Z]\s*','DelimiterType','RegularExpression');
        %Se guardan los nombres de las clases
        clase{aux} = char(C{2});
        %Se guardan los valores de los vectores
        W{aux} = str2num(matches{1});
    end
    S = aux;
    R = size(W{1},2);
    W2 = [];
    for i=1:1:S
        [A] = W{i};
        for j=1:1:S
            W2(i,j) = A(j);
        end
    end
    disp('W1');
    disp(W);
    disp('W2');
    disp(W2);
    %ya que el valor de
    Z =[1 -1 -1;1 1 -1];
    W2 = Z;
```

```
    %Se calcular el valor del bias
    b = [];
    for i=1:1:S
        b(i) = R;
    end
    %calculamos el valor de epsilon
    var = 1/S-1;
    %Multiplicando el valor aleatorio dado por 'var' obtenemos que el valor
    %aleatorio se quede en un rango de 0 y 'var'
    Ep = rand()*var;
    if Ep < 0
        Ep = Ep*-1;
    end
    Wp = eye(S);
    %Una ve tiendo epsilon calculamos la matris de pesos
    for i=1:1:S
        for j=1:1:S
            if Wp(i,j) == 0
                Wp(i,j) = -Ep;
            end
        end
    end
    %Se ingresa el valor de entrada
    p=[-1;-1;-1];
    disp('p');
    disp(p);
    %Una vez teniendo e valor de P empezamos con las iteraciones de hamming
    %Comenzamos calculando la capa Feedfoward (solo una vez)
    b = b';
    a0 =(W2*p)+b;
    disp('a0');
    disp(a0);
    %Ahora una vez calculada la capa Feedfoward podemos empezar con la capa
    %recurrente.
    %Esta parte debe repertirse hasta que converja la solucion, esto quiero
    %decir solo una salida debe terner un valor y la demas deben ser 0
    iteracion = 1;
    disp('Valores de la matriz de pesos');
    disp(Wp);
    disp('Iteracion 1');
    banderaRep = 0;
    x = [];

    a1 = poslin(Wp*a0);
    for c=1:1:S
        x(c) = c;
    end
    plot(x,a1);
    ylabel('Cambios en a');
    xlabel('Iteraciones');
    title('Red de hamming');
    hold on
            x = zeros(S,1);
            a2 = poslin(Wp*a1);
            for c=1:1:S
                x(c) = c;
            end
            plot(x,a1);
            hold on
            x = zeros(S,1);
            disp('a1');
```

```
            disp(a1);
            disp('a2');
            disp(a2);
            a3 = a2;
            cad = "Iteracion %d\na%d =";
            converje = 0;
            while iteracion < 100
                hold on
                iteracion = iteracion + 1;
                str = sprintf(cad,iteracion,iteracion);
                disp(str);
                disp(a1);
                a2 = poslin(Wp*a1);
                for c=1:1:S
                    x(c) = c;
                end
                plot(x,a1);
                x = zeros(S,1);
                for i=1:1:size(a2)
                    disp(a2(i));
                    if a2(i) == 0
                        banderaRep = 1;
                    end
                end
                disp('Grafica de evolucion de valores');
                if banderaRep == 1
                    if a3 == a2
                        converje = 1;
                        break
                    end
                end
                a3 = a2;
            end
            if converje == 1
                cad = "La funcion converje a un tipo de vertor: %s";
                for i=1:1:S
                    class = clase{i};
                end
                str = sprintf(cad,class);
                disp(str);
            end
            hold off
            disp('Fin de la funcion');
        end
```