



**INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO**



Neural Networks

“Red Hamming”

Resumen

Implementación en MATLAB de una RNA Competitiva, en este caso la red de Hamming para clasificar vectores de entrada p por medio del manejo de archivos para leer la matriz de pesos para la capa FeedForward y el vector de entrada p.

Por:

Joel Mauricio Romero Gamarra

Profesor:

MARCO ANTONIO MORENO ARMENDÁRIZ

Noviembre 2017

Índice

Contenido

Introducción:.....	1
Análisis Teórico:	2
Software (librerías, paquetes, herramientas):	2
Procedimiento:	3
Resultados	3
Discusión:	14
Conclusiones:.....	15
Referencias:	15
Código	16

Introducción:

La red Hamming es una red que fue diseñada para resolver problemas de reconocimiento y clasificación de patrones binarios (1 o -1). Esta RNA es bastante interesante ya que cuenta con 2 capas distintas, en la primera se encuentra la capa FeedForward y en la segunda se encuentra la capa Recurrente.¹

En la Figura 1 se muestra la arquitectura en forma matricial que representa a la red Hamming.

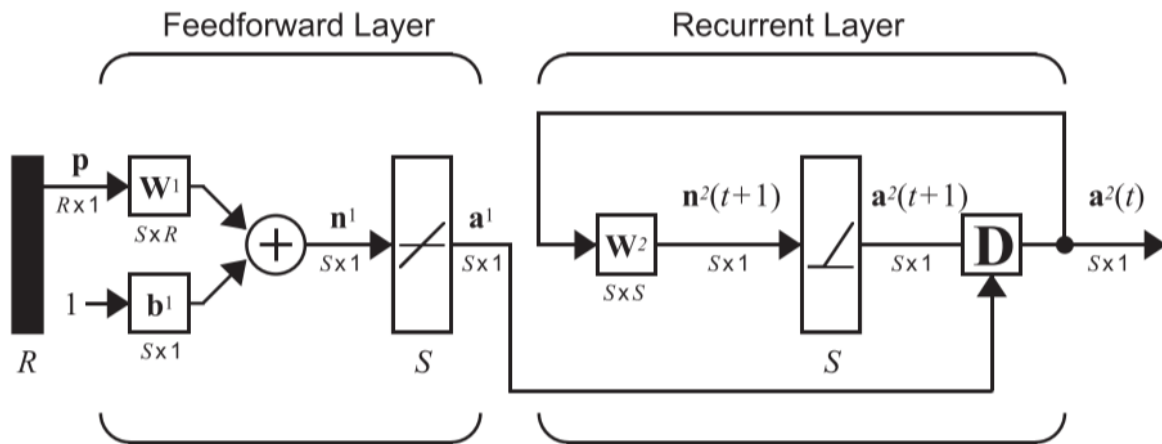


Figura 1. Arquitectura de la Red Hamming

Como se observa en la figura anterior, se ve claramente que la RNA cuenta con 2 capas distintas, además, cada una tiene una salida distinta descrita a continuación en su modelo matemático:

$$\mathbf{a}^1 = \text{purelin}(\mathbf{W}^1 \cdot \mathbf{p} + \mathbf{b}^1) \quad \text{y} \quad \mathbf{a}^2(0) = \mathbf{a}^1; \mathbf{a}^2(t+1) = \text{poslin}(\mathbf{W}^2 \cdot \mathbf{a}^2(t))$$

Donde:

- \mathbf{a}^1 : Salida de la capa FeedForward
- \mathbf{a}^2 : Salida de la capa recurrente
- purelin y poslin: Funciones de Activación de la RNA
- \mathbf{W}^1 y \mathbf{W}^2 : Matrices de pesos de la capa 1 y 2 respectivamente
- \mathbf{b}^1 y \mathbf{b}^2 : Bias de la capa 1 y 2 respectivamente
- S: Número de neuronas
- R: Dimensión del vector de entrada

Cabe resaltar que el número de neuronas en las 2 capas es el mismo, además algo interesante es que en la capa recurrente no existe un bias como en la primera y la primera señal de salida para la segunda capa es la salida de la capa 1.

Análisis Teórico:

Para el caso de la capa FeedForward se calcula el producto interno entre cada uno de los vectores prototipo y el vector de entrada (p). Por lo tanto, cada fila de la matriz de pesos W será cada uno de los vectores prototipo, con la restricción que todos los vectores prototipo deben tener el mismo número de rasgos.¹

Para calcular el bias, es una matriz como se observa en la Figura 1 de $S \times 1$, donde S es el número de neuronas, que ya sabemos que es el número de filas de la matriz de pesos W, y cada uno de esos elementos serán iguales, con el valor de R, que es la dimensión del vector de entrada, o sea que es el número de columnas de la matriz de pesos W.

Como ya se mencionó, la red Hamming es para patrones binarios (1 o -1), por lo tanto, al asignarle al bias los valores de R, nos aseguramos de que las salidas de la capa no sean negativas, y esto provoca que la capa recurrente funcione de manera correcta.

En el caso de la capa recurrente se le conoce como competitiva, las neuronas de la misma se inicializan con la salida de la capa FeedForward como ya se mencionó previamente, en esta capa las neuronas compiten entre ellas para determinar a un ganador, esto quiere decir que solo 1 neurona está activa a la vez y solo 1 contendrá el resultado correcto de la clasificación del vector de entrada. Después de un número de iteraciones, 1 de las neuronas tendrá 1 valor distinto de cero y todas las demás serán 0, esto nos indicará a que clase pertenece el vector de entrada p.

En el caso de la capa recurrente, la matriz de pesos se tiene que calcular a partir de S y nunca cambia, W^2 se define como:

$$W = \begin{bmatrix} 1 & -\varepsilon & \cdots & -\varepsilon & -\varepsilon \\ -\varepsilon & 1 & \cdots & -\varepsilon & -\varepsilon \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ -\varepsilon & -\varepsilon & \cdots & 1 & -\varepsilon \\ -\varepsilon & -\varepsilon & \cdots & -\varepsilon & 1 \end{bmatrix}$$

Donde ε es un valor menor a $\frac{1}{S-1}$

La capa recurrente realizará iteraciones hasta que converja a una clase donde solamente 1 de las neuronas (o sea el valor de salida a) sea distinto de 0 por 2 iteraciones consecutivas.

En cada iteración, el valor de la matriz de pesos seguirá siendo el mismo, sin embargo, lo que se va a estar actualizando cada vez es el vector de entrada p, que será igual a la salida en un instante de tiempo anterior, comenzando con $t = 0$ que es el valor de salida de la capa FeedForward.

Software (librerías, paquetes, herramientas):

- MATLAB R2016a²
- Sublime Text 3³

Procedimiento:

Como se puede observar en la arquitectura, primero debemos hacer el cálculo de la salida de la primera capa (FeedForward), ya que nos servirá para comenzar a hacer las iteraciones de la segunda capa. Para realizar el programa en MATLAB, se escribirá en 1 archivo de texto la matriz de pesos W y en otro archivo de texto el vector de entrada p .

El algoritmo por realizar es descrito ahora:

- Leer la matriz de pesos W^1 del archivo de texto correspondiente.
- Leer el vector de entrada p del archivo de texto correspondiente.
- Calcular el valor de salida a^1 de la capa FeedForward.
- Calcular el valor de ε aleatoriamente únicamente cuidando que se encuentre en el rango establecido.
- Utilizar el valor de salida de la capa 1 como la primera salida de la capa 2.
- Actualizar el valor de entrada p como el valor de salida en la iteración anterior.
- Escribir cada uno de los valores de salida en un archivo de texto.
- Realizar iteraciones hasta que el valor de salida a^2 sea el mismo en 2 iteraciones consecutivas y que además solo tenga 1 valor distinto de 0.
- Mostrar al usuario en que iteración la red convergió a una clase.
- Leer el archivo que contiene los valores de salida de la RNA.
- Graficar cada uno de los valores leídos para mostrar la evolución de la salida de la RNA.

Ya que contamos con el algoritmo, el paso siguiente es realizar la codificación. Los resultados obtenidos son mostrados en la siguiente sección y se hablará de ellos a profundidad en la sección de discusión más adelante.

Resultados

Las Figuras 2 a 5 muestran el funcionamiento de la red Hamming para los siguientes valores de la matriz de pesos y el vector de entrada p .

$$W^1 = \begin{bmatrix} 1 & -1 & -1 \\ 1 & 1 & -1 \end{bmatrix} \quad y \quad p = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

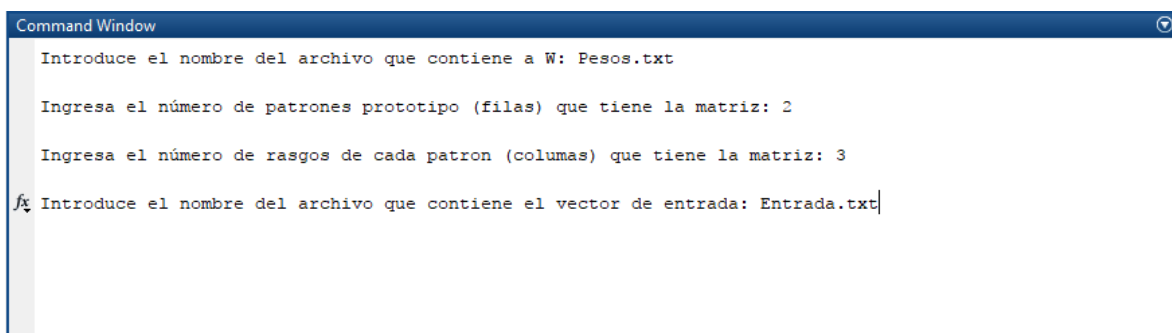


Figura 2. Valores iniciales pedidos al usuario

A continuación, se calcula el valor de ϵ aleatoriamente ya que conocemos el valor de S y podemos calcularlo para no hacerlo estático y lo mostramos en el Command Window para saber cuál es.

Además, se mostrará únicamente un mensaje que nos diga en que número de iteración se logró una correcta clasificación del vector de entrada (como ya se mencionó, la condición de paro es que 2 iteraciones consecutivas se tenga el mismo valor y que además solamente 1 de los valores de a^2 sea distinto de cero).

```
Command Window

Valor de epsilon: -7.951999e-01

La RNA convergió en la iteración 2 a la clase numero 1

fx >> |
```

Figura 3. Impresiones finales ya que se clasificó correctamente el vector p

En la Figura 4 se muestra la gráfica de la evolución de la red.

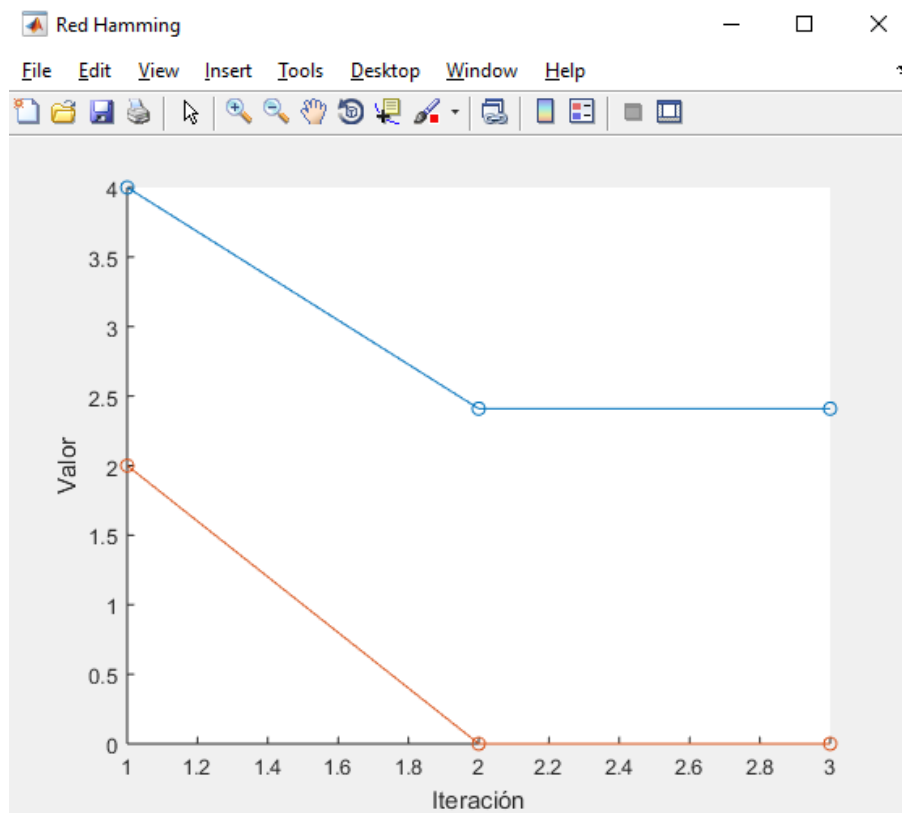


Figura 4. Gráfica de la evolución de la salida de la RNA

Como podemos ver, la salida tiene únicamente 2 elementos (2 neuronas) y cada una tiene 3 círculos (2 iteraciones más la de comprobación), además, podemos apreciar que se va aproximando uno de los valores a 0 (en el eje de las y), esto quiere decir que se va acercando a la condición de paro para que solo 1 de los valores sea distinto de 0 sin importar el número de iteraciones realizadas.

Para hacer la comprobación, observamos la gráfica que el segundo valor del vector de salida es 0, y el primero es aproximadamente 2.5, por lo que procedemos a ver el archivo que muestra la evolución de los vectores (cada 2 filas es un vector a, en este caso), por lo cual las últimas 4 filas deben tener los mismos valores, el archivo de texto se muestra en la Figura 5.

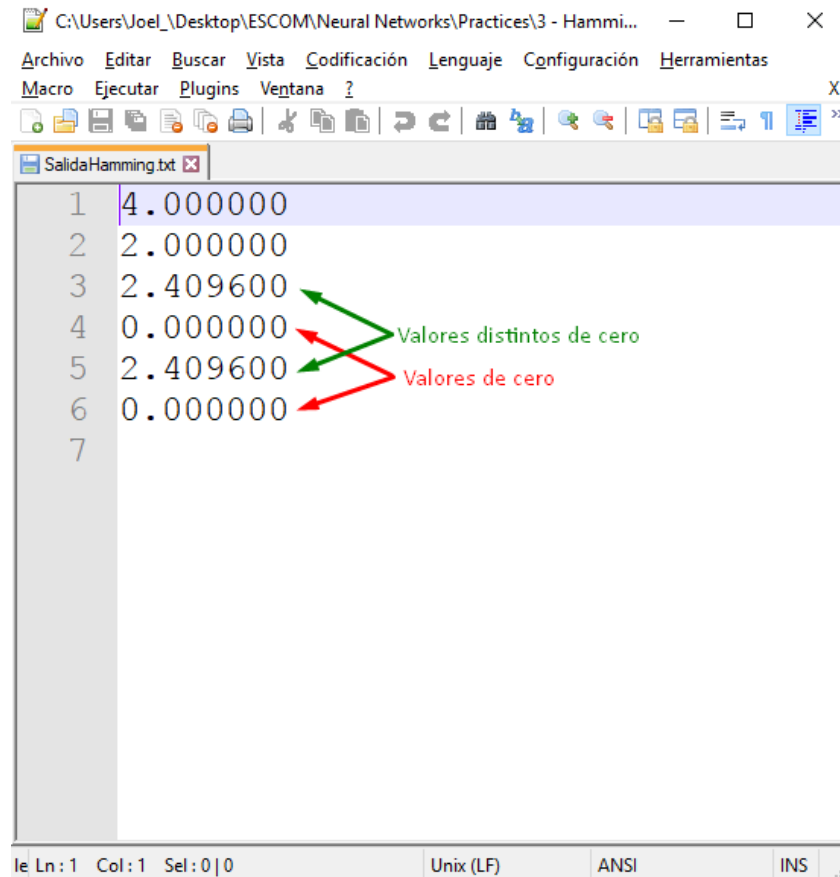


Figura 5. Archivo de texto con las salidas de la RNA

Podemos ver que se escribieron 6 líneas en el archivo de las salidas a^2 de la red de Hamming, como vimos en la iteración 2 se obtuvo el aprendizaje correcto (ya que la tercera es únicamente comprobación), y recordemos que, en este caso particular, las salidas son de 2 filas, por lo tanto, cada 2 filas es un valor distinto para la salida de a^2 .

Las Figuras 6 a 9 muestran el funcionamiento de la red Hamming para los siguientes valores de la matriz de pesos y el vector de entrada p.

$$W^1 = \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix} \quad y \quad p = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}$$

```
Command Window

Introduce el nombre del archivo que contiene a W: Pesos.txt

Ingresa el número de patrones prototipo (filas) que tiene la matriz: 3

Ingresa el número de rasgos de cada patron (columnas) que tiene la matriz: 4

fx Introduce el nombre del archivo que contiene el vector de entrada: Entrada.txt|
```

Figura 6. Valores iniciales pedidos al usuario

A continuación, realizamos calcula el valor de ϵ y se muestra al usuario junto con el número de iteración en la que se cumplió la condición de paro donde 2 iteraciones consecutivas 1 solo de los elementos es distinto de cero.

```
Command Window

Valor de epsilon: -3.827584e-01

La RNA convergió en la iteración 3 a la clase numero 1

fx >> |
```

Figura 7. Impresiones finales ya que se clasificó correctamente el vector p

En la Figura 8 se muestra la gráfica de la evolución de la red, en la que deberíamos observar como 2 de los elementos del vector de salida a^2 se van aproximando a 0 mientras uno de los otros no, para comprobar que la condición de paro se cumpla correctamente debido a que en esta red de Hamming no existe la condición de paro de un número máximo de iteraciones o algo así por el estilo.

En este caso, el vector de salida a^2 tiene 3 elementos y no 2 ya que poseemos 3 neuronas (3 vectores prototipo), y el número de rasgos de cada uno no juega un papel tan importante en la clasificación del vector p .

Y como se observa en la impresión, la gráfica nos debe mostrar 3 líneas (3 neuronas) y 4 círculos por línea (3 iteraciones, más la de comprobación) y como converge a la clase número 1, entonces la única línea que debe tener valores distintos de cero debe ser la clase 1 y las otras, valores iguales a cero en las últimas 2 iteraciones.

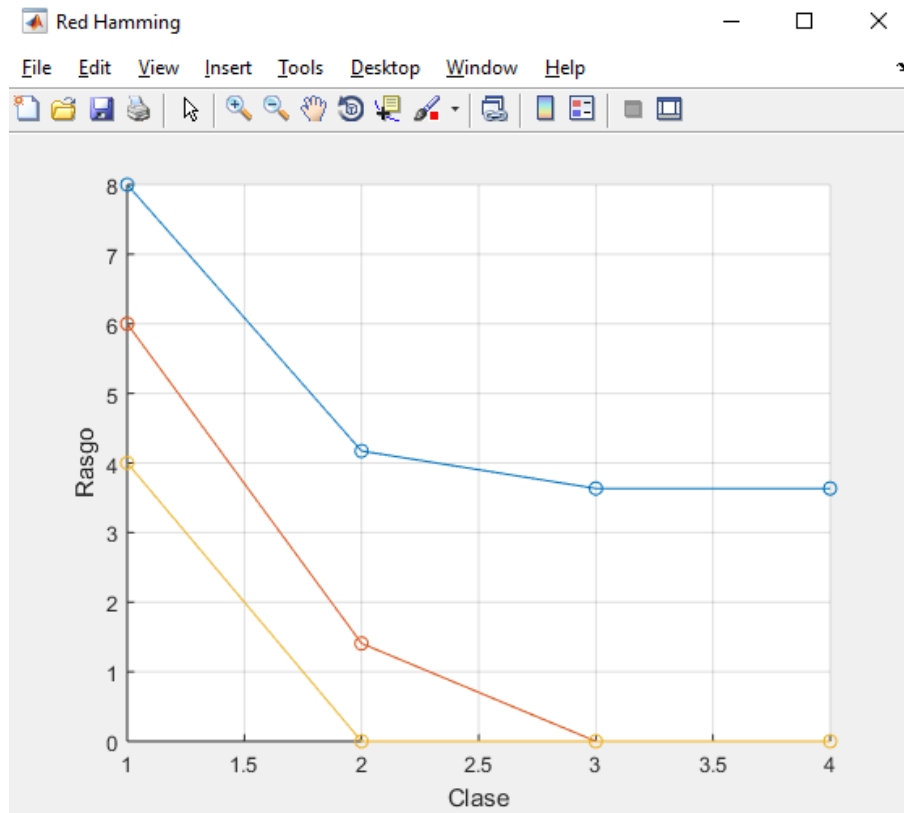


Figura 8. Gráfica de la evolución de la salida de la RNA

Como podemos ver, lo mencionado anteriormente se cumplió de manera exitosa, ya que solo una de las líneas tiene valores distintos de 0 en los últimos 2 círculos y las otras líneas no.

Para comprobar que está bien, observamos que el primer elemento debería ser aproximadamente 3.7 y los otros 2 deben ser 0 en el archivo de texto, además, al cumplirse en la tercera iteración, debemos observar 12 filas, donde las últimas 6 deben repetirse.

En la Figura 9 observamos el archivo de texto generado con cada una de las salidas de la RNA para poder hacer la gráfica de su evolución.

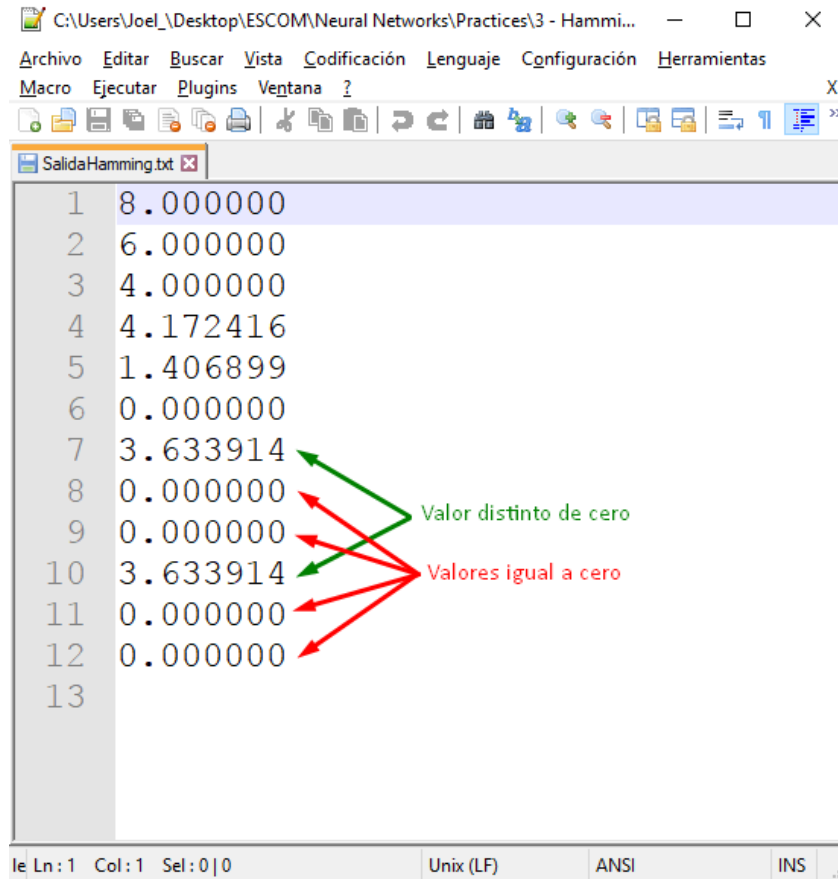


Figura 9. Archivo de texto con las salidas de la RNA

Como podemos ver, exitosamente se convergió a la clase 1 (cada 3 filas es un valor de salida de la RNA), y además se repiten los valores de las últimas 6 filas, por lo que el vector a fue el mismo en 2 iteraciones consecutivas.

Como ya vimos en 2 ejemplos anteriores, se ha clasificado correctamente al vector de entrada p , sin embargo, que ya haya convergido 2 veces a la clase número 1 puede parecer algo extraño, por lo que a continuación se muestra el funcionamiento del programa con un vector que debería de converger a la clase 3 (ya que es exactamente el mismo valor).

Las Figuras 10 a 16 muestran el funcionamiento de la red Hamming para los siguientes valores de la matriz de pesos y el vector de entrada p .

$$W^1 = \begin{bmatrix} 1 & -1 & 1 & -1 & 1 & 1 \\ -1 & 1 & -1 & -1 & -1 & 1 \\ 1 & 1 & 1 & -1 & 1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & -1 & -1 & -1 \end{bmatrix} \quad y \quad p = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \\ 1 \\ -1 \end{bmatrix}$$

Observamos que la matriz de pesos W tiene 5 filas, por lo tanto, el vector de salida a^2 debería tener 5 elementos como vector columna.

```
Command Window

Introduce el nombre del archivo que contiene a W: Pesos.txt

Ingresa el número de patrones prototipo (filas) que tiene la matriz: 5

Ingresa el número de rasgos de cada patron (columnas) que tiene la matriz: 6

fx Introduce el nombre del archivo que contiene el vector de entrada: Entrada.txt|
```

Figura 10. Valores iniciales pedidos al usuario

A continuación, se calcula el valor de ϵ y se muestra al usuario junto con el número de iteración en la que se cumplió la condición de paro donde 2 iteraciones consecutivas 1 solo de los elementos es distinto de cero.

```
Command Window

Valor de epsilon: -9.538961e-02

La RNA convergió en la iteración 12 a la clase numero 3

fx >> |
```

Figura 11. Impresiones finales ya que se clasificó correctamente el vector p

En la Figura 12 se muestra la gráfica de la evolución de la red, en la que deberíamos observar como 4 de los elementos del vector de salida a^2 se van aproximando a 0 mientras uno de los otros no.

En este caso, el vector de salida a^2 tiene 5 elementos y no 3 ya que poseemos 5 neuronas (5 vectores prototipo).

Y como se observa en la Figura 11, se llegó a una correcta clasificación en la iteración número 12, por lo tanto, deberíamos ver 13 círculos (por la comprobación) y solo 1 línea con los últimos 2 círculos distintos de cero, los demás en cero.

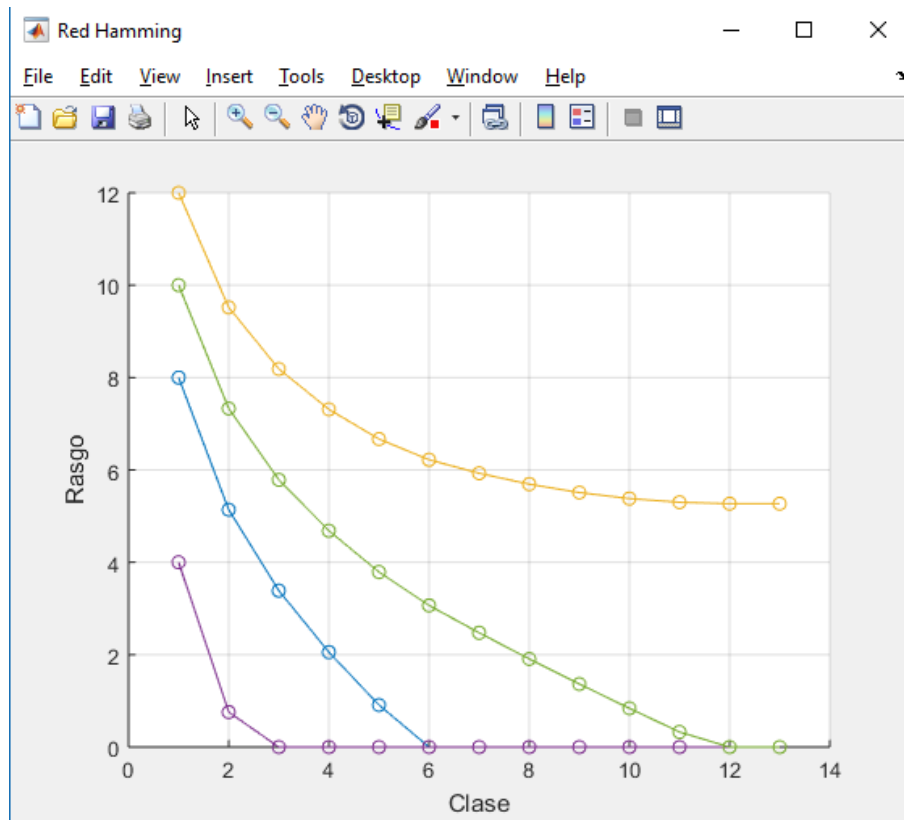
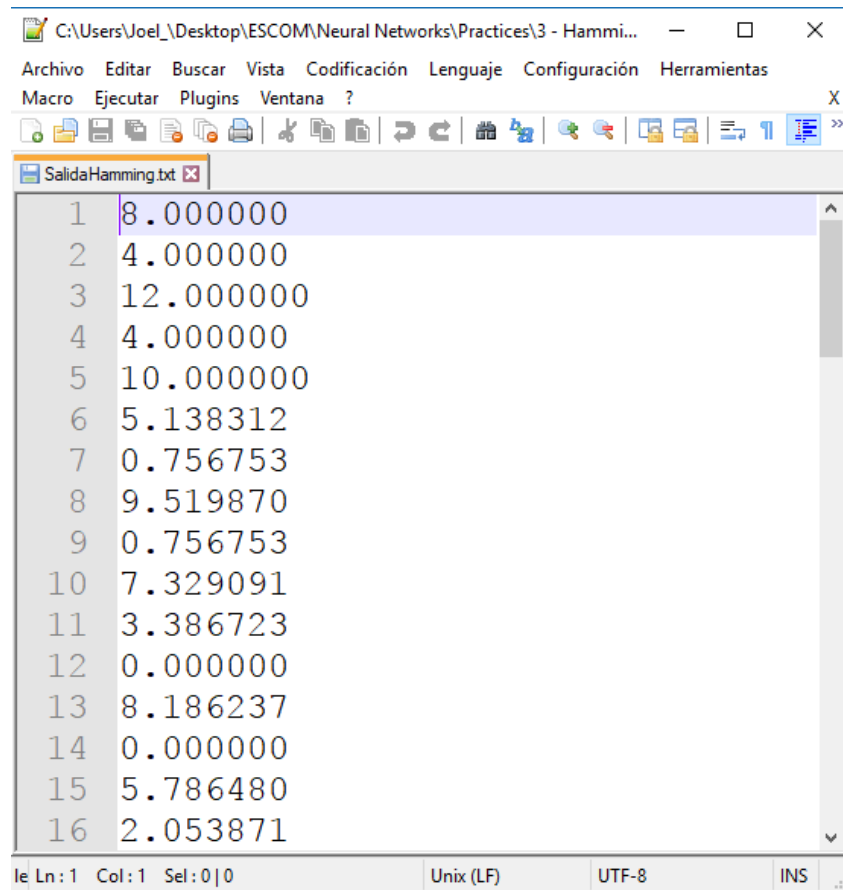


Figura 12. Gráfica de la evolución de la salida de la RNA

Como podemos ver, lo mencionado anteriormente se cumplió de manera exitosa, ya que solo el vector en color amarillo que representa a la clase 5 tiene valores distintos de cero en las últimas 2 iteraciones (cada iteración es un círculo).

Para comprobar que está bien, observamos que los últimos elementos de la línea amarilla deberían ser aproximadamente 5.4 y los otros 4 deben ser 0 en el archivo de texto, además, al cumplirse en la doceava iteración, debemos observar 65 filas, donde las últimas 10 deben repetirse.

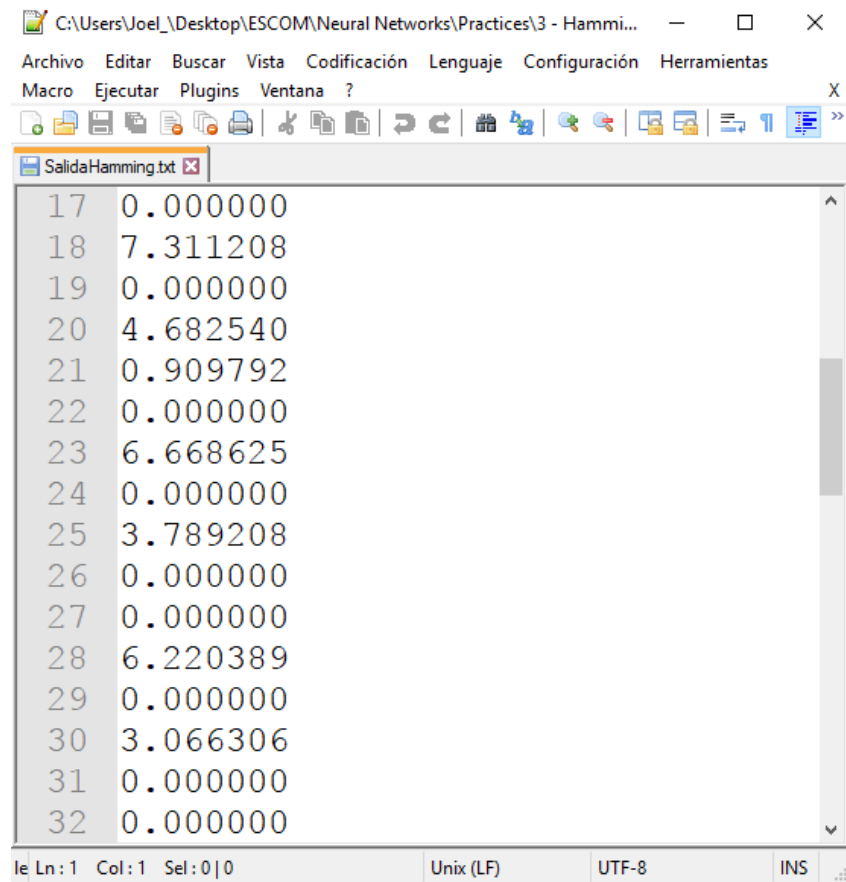
En la Figura 13 observamos el archivo de texto generado con cada una de las salidas de la RNA para poder hacer la gráfica de su evolución.



The image shows a text editor window titled "SalidaHamming.txt". The window contains 16 lines of text, each consisting of a line number followed by a decimal value. The values are: 1: 8.000000, 2: 4.000000, 3: 12.000000, 4: 4.000000, 5: 10.000000, 6: 5.138312, 7: 0.756753, 8: 9.519870, 9: 0.756753, 10: 7.329091, 11: 3.386723, 12: 0.000000, 13: 8.186237, 14: 0.000000, 15: 5.786480, 16: 2.053871. The first line is highlighted. The status bar at the bottom indicates "Ln: 1 Col: 1 Sel: 0|0", "Unix (LF)", "UTF-8", and "INS".

Line	Value
1	8.000000
2	4.000000
3	12.000000
4	4.000000
5	10.000000
6	5.138312
7	0.756753
8	9.519870
9	0.756753
10	7.329091
11	3.386723
12	0.000000
13	8.186237
14	0.000000
15	5.786480
16	2.053871

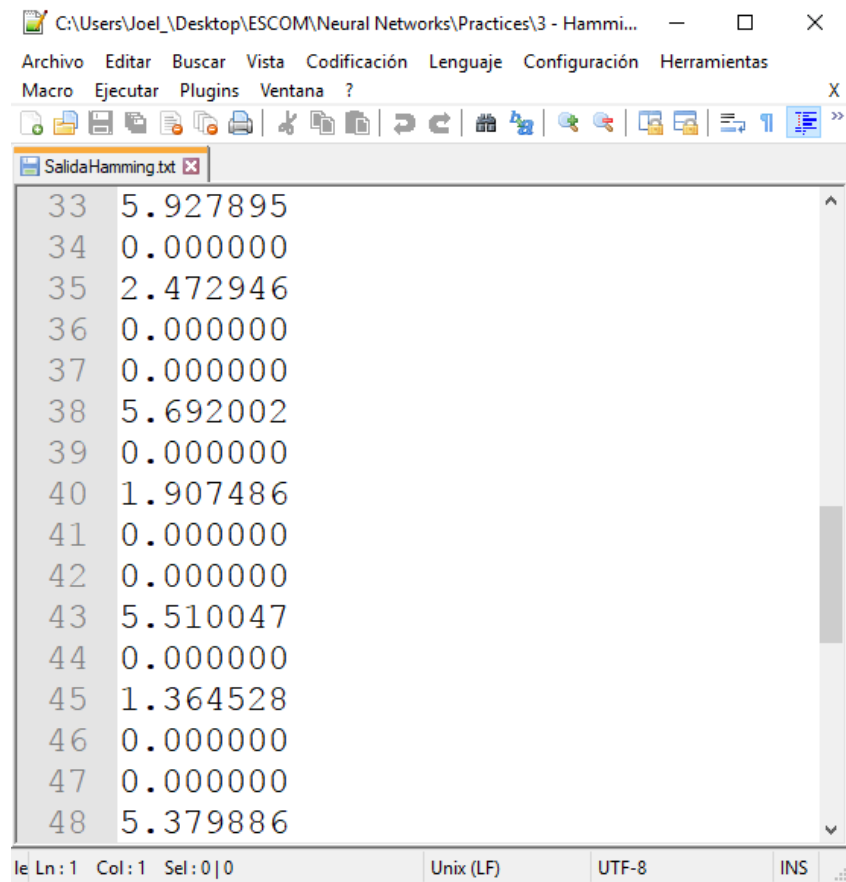
Figura 13. Archivo de texto con las salidas de la RNA (1)



```
17 0.000000
18 7.311208
19 0.000000
20 4.682540
21 0.909792
22 0.000000
23 6.668625
24 0.000000
25 3.789208
26 0.000000
27 0.000000
28 6.220389
29 0.000000
30 3.066306
31 0.000000
32 0.000000
```

Ln: 1 Col: 1 Sel: 0 | 0 Unix (LF) UTF-8 INS

Figura 14. Archivo de texto con las salidas de la RNA (2)



```
33 5.927895
34 0.000000
35 2.472946
36 0.000000
37 0.000000
38 5.692002
39 0.000000
40 1.907486
41 0.000000
42 0.000000
43 5.510047
44 0.000000
45 1.364528
46 0.000000
47 0.000000
48 5.379886
```

Ln: 1 Col: 1 Sel: 0 | 0 Unix (LF) UTF-8 INS

Figura 15. Archivo de texto con las salidas de la RNA (3)

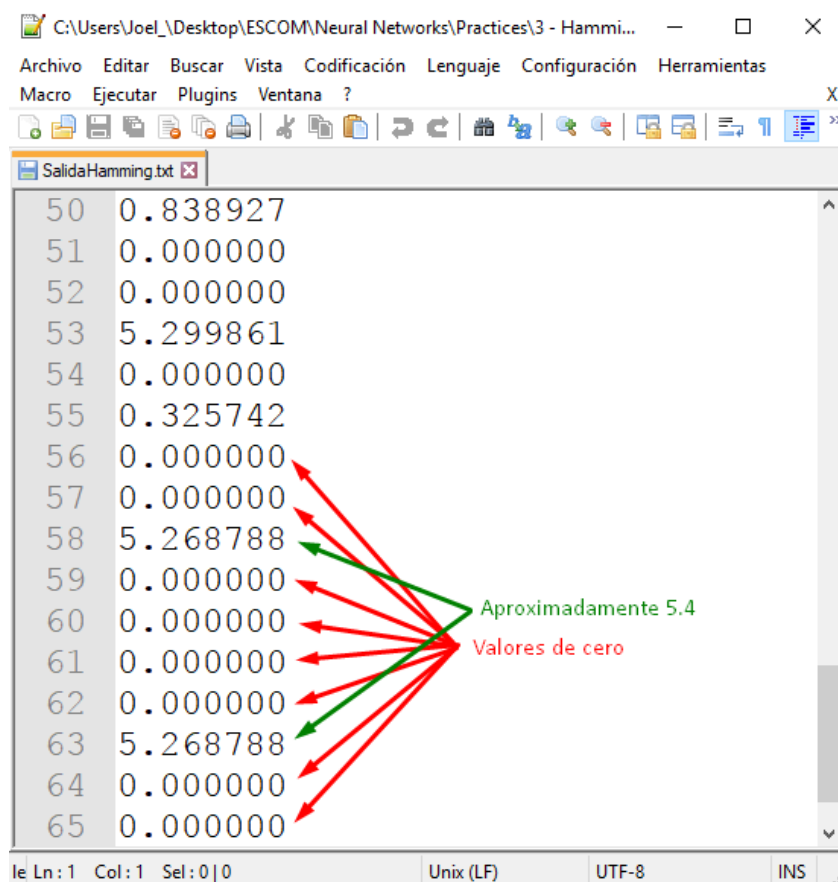


Figura 16. Archivo de texto con las salidas de la RNA (4)

Como podemos ver, exitosamente se convergió a la clase 3 (cada 5 filas es un valor de salida de la RNA), y además se repiten los valores de las últimas 10 filas, por lo que el vector a fue el mismo en 2 iteraciones consecutivas, esto quiere decir que se clasificó correctamente al vector de entrada p.

Además, podemos ver claramente que se converge a la clase 3 ya que los primeros 2 elementos son 0, luego el tercero es distinto de 0 y los otros 2 que siguen vuelven a ser 0 (2 veces).

Discusión:

En la sección donde presentamos los resultados de 3 matrices de pesos y vectores de entrada totalmente distintos, pudimos observar cómo se lleva a cabo las iteraciones de la red de Hamming hasta converger a una de las clases y poder clasificar correctamente el vector de entrada p.

Además, al escribir los datos en un archivo de texto nos ahorramos el problema de estar gastando memoria RAM, que, aunque en este caso no se ocupa demasiada, puede llegar un momento en el que le tome a la RNA llegar a converger a una clase muchas iteraciones y estar guardando cada salida tendría un costo de almacenamiento bastante alto.

Conclusiones:

Como ya se había platicado, el manejo de archivos en MATLAB no es complicado, de hecho, es bastante similar al lenguaje C, utilizando funciones con una sintaxis muy similar, además, el hacer el uso de ellos pone a prueba un poco de la lógica de programación en MATLAB, ya que no es tan parecido en ciertas cosas, por ejemplo, multiplicar las matrices es demasiado simple.

Lo más complicado del código (que no lo fue mucho), fue al momento de hacer la comprobación para ver si se cumplía la condición de paro, ya que restaba únicamente 2 vectores columna consecutivos (para saber si eran iguales), ya que, si esa resta me daba 0, entonces eran iguales, por lo tanto, se tenía que verificar la siguiente condición, para ver que efectivamente solamente 1 de los elementos tuviera un valor distinto de 0 y los demás no. La parte complicada fue hacer operaciones con únicamente 1 vector de una matriz muy grande (ya que cada iteración se agregaba 1 columna), sin embargo, una vez que logré hacer esa resta exitosamente, el resto fue fácil ya que ir escribiendo los valores del vector de salida en un archivo lo facilitó bastante a la hora de hacer la gráfica de la evolución, ya que únicamente se tenía que ir leyendo de S en S datos del archivo y graficarlos.

Este tipo de red posee un aprendizaje no supervisado, ya que no tenemos un target al cual queremos llegar, además es una de las RNA competitivas más simples que existen y tiene aplicaciones en las siguientes áreas⁴:

- Reconocimiento de imágenes
- Transmisión de señales
- Reconocimiento de patrones
- Reconocimiento de voz

Referencias:

- [1] “Red Hamming”, class notes for Neural Networks, Department of Engineering in Computer Systems, Escuela Superior de Cómputo, 2017.
- [2] Math Works, ‘MATLAB’, [Online]. Disponible en: <https://es.mathworks.com/products/matlab>.
- [3] Sublime HQ, ‘Download’, [Online]. Disponible en: <https://www.sublimetext.com/3>
- [4] Edmundo René Durán Camarillo, ‘REDES COMPETITIVAS’, 2009, [Online]. Disponible en: <https://es.slideshare.net/mentelibre/redes-neuronales-competitivas-hamming>. [Accedido: 1 – Noviembre – 2017].

Hamming.m

```
%Limpiamos la pantalla y borramos todas las variables creadas anteriormente
clc;
clear;

%Recibimos el nombre del archivo como un String (por eso la 's')
archivo = input ('Introduce el nombre del archivo que contiene a W: ', 's');

%Abrimos el archivo que contiene la matriz a usar en modo lectura
archivo_matriz = fopen (archivo, 'r');

%Recibimos el número de filas y columnas que tiene la matriz para lectura
num_filas = input ('\nIngresa el número de patrones prototipo (filas) que tiene la matriz: ');
num_col = input ('\nIngresa el número de rasgos de cada patron (columnas) que tiene la matriz: ');

%Creamos una matriz para el bias
bias = zeros (num_filas,1);
bias = bias + num_col;

while ~feof (archivo_matriz)
    [W, ~] = fscanf (archivo_matriz, '%f', [num_col num_filas]);
end
W = W';

%Recibimos el nombre del archivo como un String (por eso la 's')
archivo = input ('\nIntroduce el nombre del archivo que contiene el vector de entrada: ', 's');

%Abrimos el archivo que contiene la matriz a usar en modo lectura
archivo_matriz = fopen (archivo, 'r');

%Guardamos el vector de entrada p en una matriz
p = fscanf (archivo_matriz, '%f');

%COMIENZA LA CAPA FEEDFORWARD (1 vez)
a = purelin((W * p) + bias);

clc

%Calculamos un valor para epsilon aleatorio
epsilon = (rand() * (1/(num_filas - 1)) * -1);
fprintf ('\n\nValor de epsilon: %d\n\n', epsilon);

%Calculamos la matriz de pesos a usar en la capa Recurrente
W = zeros (num_filas);
W = W + epsilon;
for i = 1:num_filas
    for j = 1:num_filas
        if i == j
            W (i, j) = 1;
        end
    end
end

%COMIENZA LA CAPA RECURRENTE (n veces)
flag = ones (num_filas, 1); iteracion = 1;

%ESCRIBIR LA MATRIZ EN UN ARCHIVO DE TEXTO LLAMADO SalidaHamming
nuevo = fopen ('SalidaHamming.txt', 'w');
```

```

for i = 1:num_filas
    fprintf (nuevo, '%f\n', a(i, 1));
end
cont = 0;
while cont ~= 1
    cont = 0;
    p = a;
    a = poslin (W * p);
    if iteracion == 1
        aux = a;
    else
        %Agregamos un vector columna a la matriz aux
        aux = [aux, a];

        %Restamos los 2 vectores columna y lo guardamos en la bandera
        flag = aux (:,iteracion) - aux (:, iteracion - 1);
        if flag == zeros (num_filas, 1)
            for i = 1:num_filas
                clase = a (i, 1);
                if clase ~= 0
                    cont = cont + 1;
                end
            end
        end
        %Escribimos los datos en un archivo
        for i = 1:num_filas
            fprintf (nuevo, '%f\n', a(i, 1));
        end
        iteracion = iteracion + 1;
    end

    for i = 1:num_filas
        clase = a (i, 1);
        if clase ~= 0
            break;
        end
    end
end

fprintf ('\nLa RNA convergió en la iteración %d a la clase numero %d\n\n', iteracion - 1, i);

%GRAFICAMOS LOS VALORES DE SALIDA DE LA RED
nuevo = fopen ('SalidaHamming.txt', 'r');
Graph = figure('Name','Red Hamming','NumberTitle','off');

while ~feof (nuevo)
    [r, cont] = fscanf (nuevo, '%f\n', [num_filas iteracion]);
end

for i = 1:num_filas
    figure (Graph);
    hold on;
    plot (r (i,:), 'o-');
    grid, ylabel('Valor'), xlabel('Iteración');
end

%Cerramos los archivos
fclose (archivo_matriz);
fclose (nuevo);

%Borramos todas las variables creadas durante la ejecución del programa
clear;

```