Instituto Politécnico Nacional

Escuela Superior de Cómputo

Neural Network

3CM1

Practica 2: Red de Hamming

Hernández López Ángel Zait

2014080682

Ciclo escolar: 2019/2

11 de marzo de 2019

Introducción.

Existen distintos tipos de funciones de una red neuronal, los modos básicos son los modo regresor y clasificador. De manera muy sencilla, podemos decir que el modo regresor ajusta los datos ingresados dentro de una RNA; el modo clasificador, como su nombre lo dice, nos ayudará a separar los datos ingresados en la red neuronal.

La red de Hamming es una red neuronal que nos va a ayudar a identificar del vector prototipo, a que clase corresponde de los pesos sinápticos. Por lo que podemos decir que la RNA actúa en modo clasificador. Esta red está constituida por dos capas, las cuales se explicarán un poco más adelante.

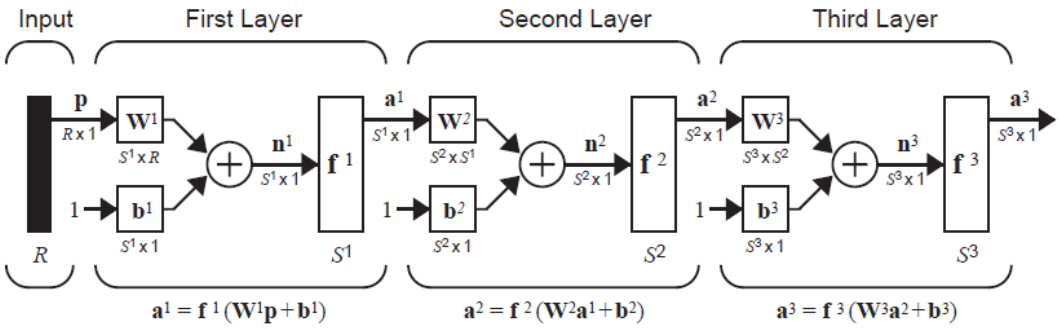
También hay otra clasificación de las redes neuronales, las cuales son feedforward y recurrente. Estás son las dos capas que utiliza la Red de Hamming para poder clasificar el vector de entrada.

Con ayuda de estas redes y sabiendo bien su funcionamiento de cada uno, podemos desarrollar una red de Hamming, que es lo que se nos pidió en esta práctica. Como parámetros de entrada se ingresarán los datos de los pesos sinápticos de la primera capa y el vector prototipo, para saber de que clase es dicho vector.

Marco teórico.

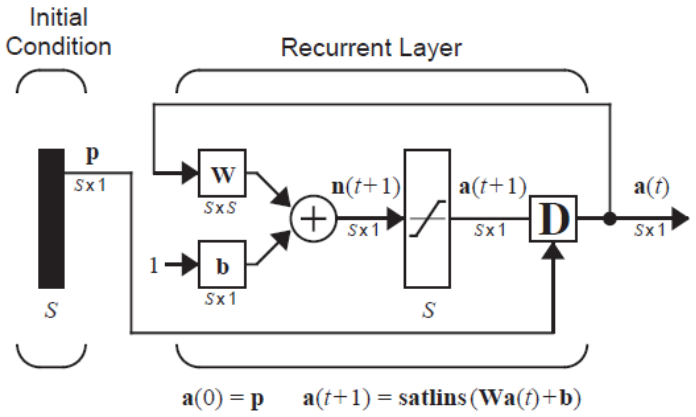
Antes de entrar a ver que es una red de Hamming, debemos saber dos conceptos importantes para el desarrollo de la RNA, estos conceptos son una red feedforward y una red recurrente.

Una RNA feedforward (propagación hacia adelante) si sus conexiones van exclusivamente en un sentido, van de las entradas hacia las salidas, pero no para atrás, arriba, abajo o tiene auto conexiones y puede tener más de una capa de conexión. Otra característica que la distingue es que esas capas, podemos dividirlas en tres partes principales, las cuales son la capa de entrada, donde se reciben los datos a procesar; la capa intermedia, donde se da el procesamiento de la información, y la capa de salida. El diagrama en forma matricial es el siguiente:

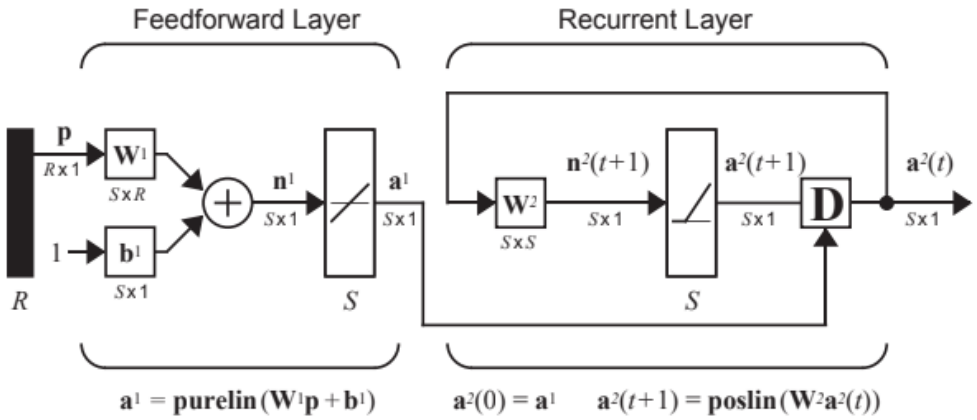


Como se puede notar, la salida de la primera capa es la entrada de la segunda capa, y así sucesivamente; si es que tenemos n capas, será lo mismo con cada una de las capas siguientes, con su función correspondiente.

Para una RNA recurrente se necesita de un bloque llamado retraso en tiempo, el cual es la condición inicial para la RNA. Se caracterizan porque se crean bucles en las neuronas de la red mediante el uso de las conexiones recurrentes, pudiendo aparecer en la red conexiones de una neurona a ella misma, conexiones entre neuronas de una misma capa o conexiones de las neuronas de una capa a la capa anterior. La consideración de conexiones recurrentes en una RNA implica un aumento del número de pesos o parámetros ajustables en la red, lo cual permite que aumente la capacidad de representación. El diagrama en forma matricial es el siguiente:



Una red de Hamming es una red competitiva, está diseñada explícitamente para resolver problemas de reconocimiento binario de patrones, entonces podemos decir que esta red nos puede ayudar a clasificar a más de una clase. Esta red neuronal está constituida por dos capas, una capa es de una red tipo feedforward y la otra es de una red recurrente, como se muestra de la siguiente manera.



- Capa feedforward:
  Esta capa 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

w^1, serán cada uno de los vectores prototipo. La ecuación que describe esta capa es:

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

- 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, sólo una neurona de esta capa tendrá un valor diferente de cero. La neurona ganadora indica a que clase pertenece el vector de entrada. Las ecuaciones que describen esta capa son:

$$a^2(0) = a^1$$
$$a^2(t+1) = poslin\left(w^2 a^2(t)\right)$$

  Y la nueva matriz de pesos tiene la siguiente forma:

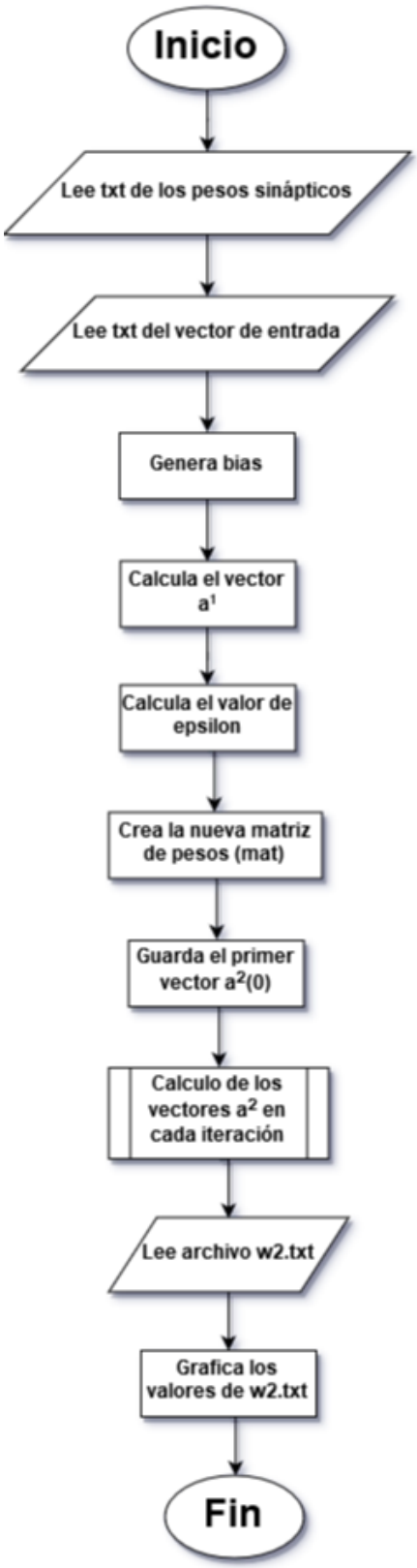
$$w^2 = \begin{bmatrix} 1 & -\varepsilon \\ -\varepsilon & 1 \end{bmatrix}$$

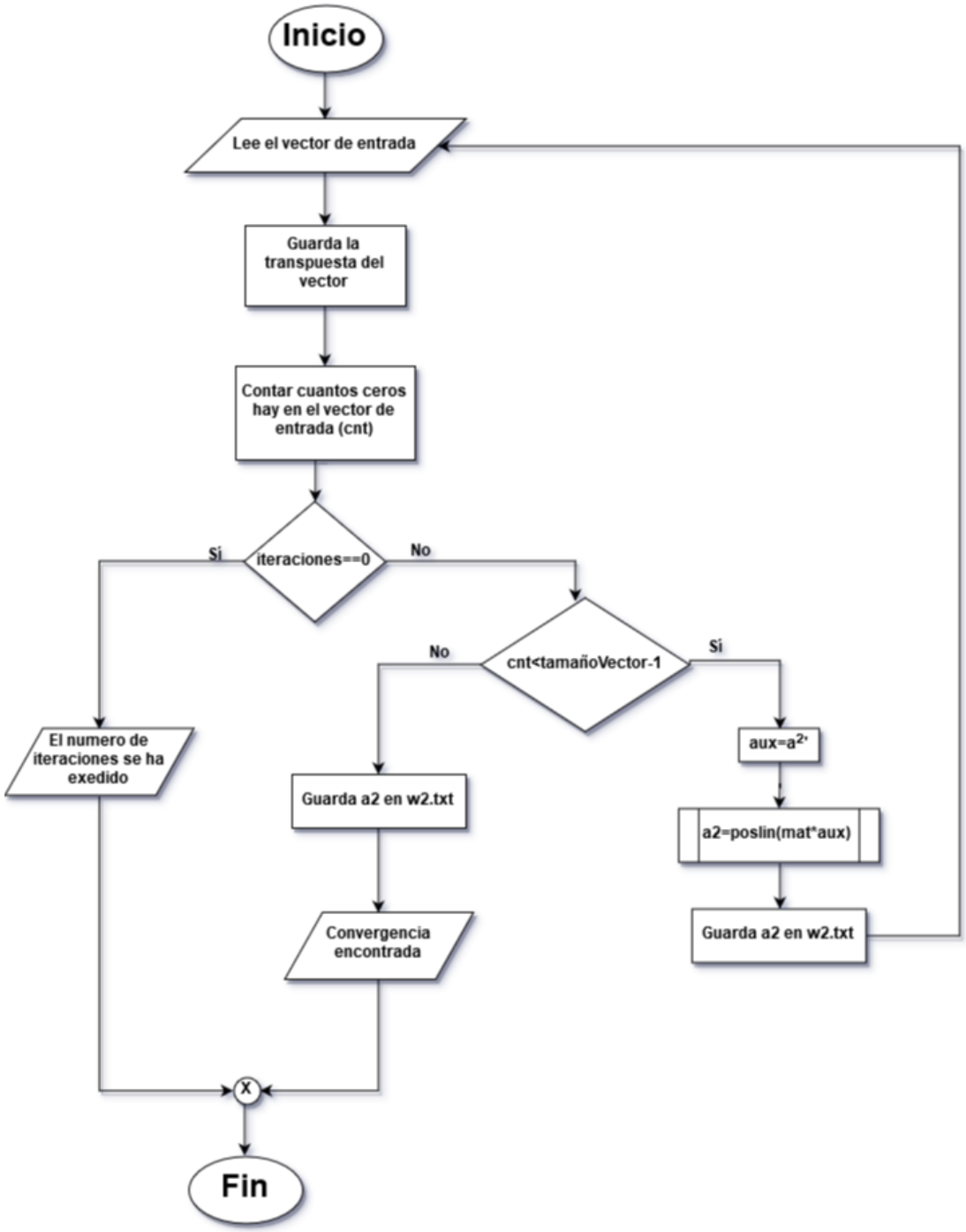
  Donde el valor de épsilon (ε) es un valor menor a $\frac{1}{s-1}$. Si la matriz es más grande, se pone el valor de uno en la diagonal principal y $-\varepsilon$ en las demás posiciones.

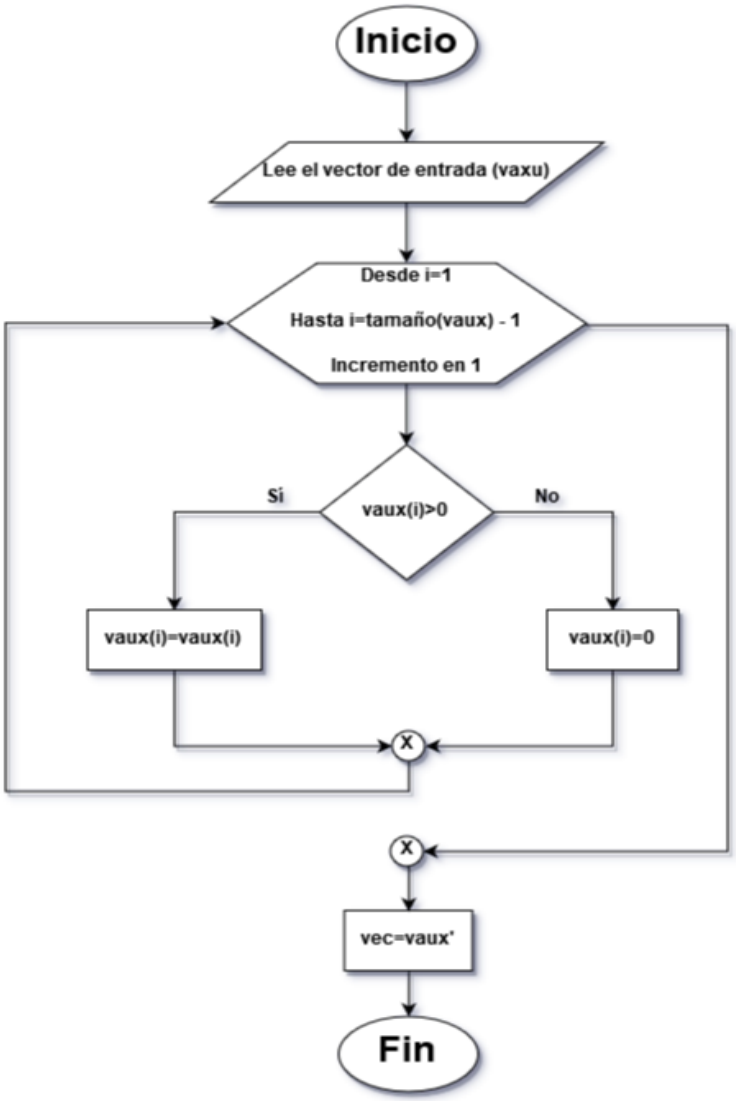
Diagrama de flujo.

A continuación, se mostrarán los diagramas de flujo, para poder observar de una forma más sencilla el funcionamiento del programa. Se tienen tres diagramas, los cuales describen el funcionamiento del programa principal, la función capaRecurrente, y la función poslin.

- Programa principal: En este diagrama se explica cómo es que se obtienen los valores de la matriz de pesos sinápticos y el vector prototipo (vector de entrada) y como hace las operaciones para la salida de la capa feedforward, el cual usa una función de transferencia purelin, pero se optó por no hacer una función especial programable. Y la salida de esa operación, dar el vector de entrada a la capa recurrente, el cual es la función capaRecurrente y la matriz con valores de epsilon. Una vez acabada la función continua para poder obtener los valores de dicha capa y graficarla.
- Función capaRecurrente: Teniendo el vector de salida de la capa feedforward como vector de entrada de la capa recurrente, este hace la operación de dicha capa, tomando en cuenta los parámetros del numero de interaciones. Si es que el número de iteraciones es cero, este se sale de la función, si no es el caso y el número de ceros en el vector de entrada es menor al tamaño del vector menos uno, continuará con la tarea de capa recurrente, en otro caso, significa que convergió el vector y saldrá de la función.
- Función poslin: No es más que la función de transferencia ocupada en la capa recurrente, si el valor del vector es menor a cero, este devolverá un cero en vez del valor obtenido, y en otro caso, devolverá el valor que tiene de la multiplicación de la matriz con los valores de épsilon y el vector de entrada.


Inicio
Lee txt de los pesos sinápticos
Lee txt del vector de entrada
Genera bias
Calcula el vector a¹
Calcula el valor de epsilon
Crea la nueva matriz de pesos (mat)
Guarda el primer vector a²(0)
Calculo de los vectores a² en cada iteración
Lee archivo w2.txt
Grafica los valores de w2.txt
Fin


Inicio
Lee el vector de entrada
Guarda la transpuesta del vector
Contar cuantos ceros hay en el vector de entrada (cnt)
iteraciones==0
Si
No
El numero de iteraciones se ha exedido
cnt<tamañoVector-1
No
Si
Guarda a2 en w2.txt
Convergencia encontrada
aux=a2'
a2=poslin(mat*aux)
Guarda a2 en w2.txt
X
Fin



Experimentos.

Para la experimentación de la red, se decidió poner los siguientes valores para la matriz de pesos sinápticos y el vector prototipo (pesos3.txt y entrada.txt):

$$w^1 = \begin{bmatrix} 1 & -1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 \\ -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 \\ -1 & -1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$p^T = [-1 \quad 1 \quad 1 \quad -1 \quad -1]$$

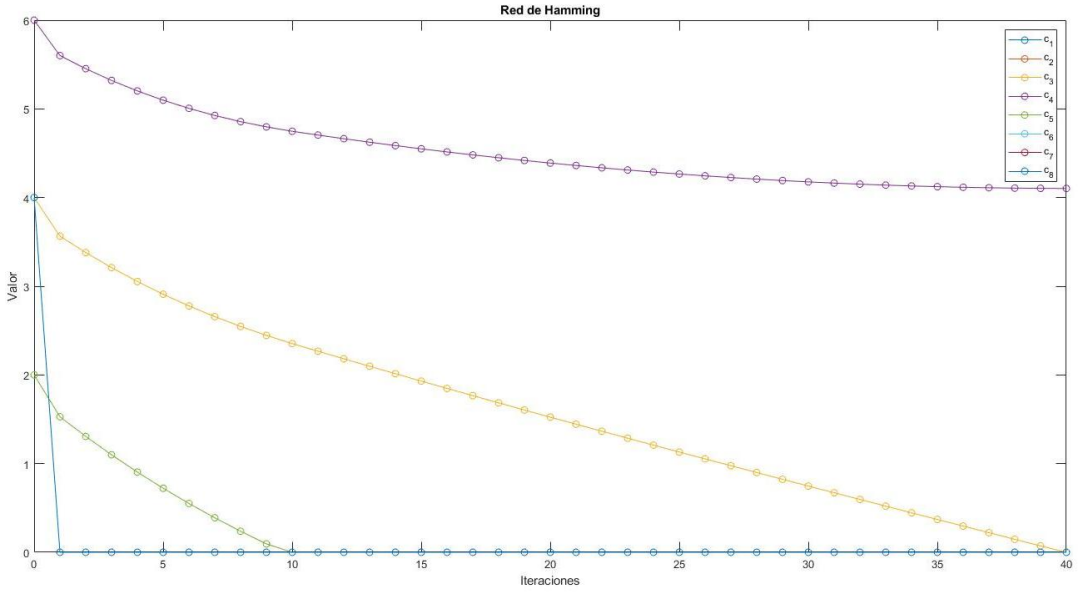
El programa los obtiene por medio de un archivo .txt, el cual el programa lo leerá ingresando el nombre del archivo con su respectiva extensión. Después de obtener los archivos con los valores de cada uno de ellos, calcula el valor del bias, con respecto al valor de R, que en este caso es 5, por lo que el vector de vías es el siguiente:

$$b^T = [5\ \ 5\ \ 5\ \ 5\ \ 5\ \ 5\ \ 5\ \ 5]$$

Ahora, al salir de la capa feedforward, el valor del vector a^1 queda de la siguiente forma:

$$a_1^T = [2\ \ 2\ \ 4\ \ 6\ \ 2\ \ 4\ \ 4\ \ 4]$$

En este caso, el valor de épsilon tiene como valor $\varepsilon = 0.0181$ y el número de iteraciones fue hasta 40. Como resultado final, tenemos que el vector prototipo de entrada, es de clase 4, y también se muestra la gráfica de iteraciones, y viendo como es poco a poco va convergiendo el vector de entrada.



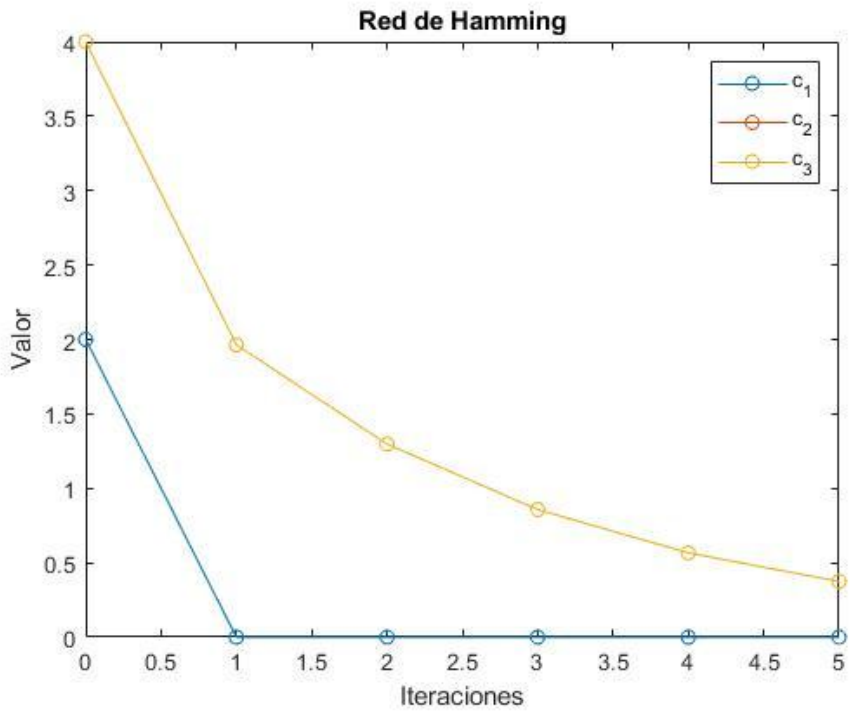
Ahora ingresaremos un valor diferente para la matriz de pesos y el vector de entrada (now.txt y noe.txt), los cuales son:

$$w^1 = \begin{bmatrix} -1 & 1 & -1 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix}$$

$$p^1 = [1 \quad 1 \quad 1]$$

Igual que el caso anterior, hace el cálculo del bias y el valor de épsilon, y entra a la capa recurrente para saber de que clase pertenece el vector de entrada, pero en este caso, no convergió, esto es porque ser excedió el numero de iteraciones, en este caso, fueron cinco, y como se ve en la gráfica, la clase dos y tres están empalmadas, esto se debe porque tienen el mismo valor, pero si vemos el valor numérico del ultimo vector es:

$$w^2 = [0 \quad 3.74 \quad 3.74]$$



Discusión.

Esta red neuronal, como podemos observar, puede que tarde o no el proceso de convergencia, esto se debe por el tamaño de la matriz de pesos sinápticos y por el valor de épsilon que se genere.

Es bueno para clasificar, pero solamente eso, ya que no le estamos enseñando nada a la red, solamente le metemos información y clasifica con ayuda de cálculos matriciales y funciones de transferencia. Pero no es que la red, al momento de tomar ese dato, y haberlo clasificado, se vuelva más rápido para identificar que clase es el vector de entrada.

Conclusión.

La red de Hamming funciona bien como clasificador, ya que, teniendo un número de clases, con características diferentes, este, al momento de ingresar un vector sin saber casi nada de él, más que sus valores, podemos identificar a que clase pertenece. Esto nos podría ayudar tal vez después para que otra red pueda aprender, o simplemente usarlo como herramienta de separación de datos con ciertas características.

Lo complicado de esta práctica, en cierta forma, es saber donde es posible que pare la convergencia. También el saber donde es que puede converger, porque puede que haya valores del vector que tiendan a cero y que sea difícil llegar al valor de cero, y puede que no converja ese vector de entrada.

Referencias.

T. Hagan, Martin. (2014). Neural Network Design. Estados Unidos: eBook.

Isasi Viñuela, P; Galván León, I. M. (2004). Redes Neuronales Artificiales. Un enfoque práctico. Madrid: PEARSON.

Rodríguez de la Torre, Ángel Álvarez. (2010). Feedback y Feedforward, dos herramientas de coaching. Sitio web: https://innovacioneducativa.upm.es/jimcue_10/comunicaciones/05_Alvarez.pdf

Anexo.

- **P02_Hamming_2014080682.m**

```
clear
clc
disp('Practica 2: Red Hamming')
txt=input('Ingrese el nombre del documento donde se encuentran los pesos
sinapticos: ','s');
w=load(txt);
txt=input('Ingrese el nombre del documento donde se encuentre el vector
prototipo: ','s');
p=load(txt);
[s,r]=size(w);      % Obtención de los valores de r y s
b=(ones(s,1))*r;    % Calculo de bias
a1=(w*(p'))+b;      % Calculo del vector a1
epsilon=0+((1/(s-1))-0).*rand;  % Genera epsilon

a1=poslin(a1);
vaux=(a1');
guardar(vaux);  % Crea el documento w2.txt y guarda la salida de la
                % capa freeforward
matep=ones(s)*(-epsilon);   % Crea la matriz con epsilons negativos
for i=1:s
    for j=1:s
        if i==j
            matep(i,j)=1;
        end
    end
end

itc=100+(300-100).*rand;            % Numero de iteraciones
capaRecurrente(matep,a1,itc);       % Hace la tarea de la capa recurrente
w2=load('w2.txt');
graficar(w2);
disp('Programa terminado')
```

- **capaRecurrente.m**

```
function capaRecurrente(mat,a2,ite)
    vaux=a2'; cnt=0;
    tipclas=0;
    tam=numel(a2');
    for i=1:tam
        if vaux(i)==0
            cnt=cnt+1;  % Cuenta el numero de ceros
        else
            tipclas=i;  % Guarda la posible clase que converje
        end
    end

    if ite==0   % Si sobrepasa el valor de iteraciones, sale de la
función
        disp('Iteraciones terminadas')
    elseif cnt<tam-1        % Si sigue habiendo ceros, continua con la
función
        aux=poslin(mat*a2); % Llama a la función poslin
        editar(aux');       % Guarda los nuevos valores de w2
        capaRecurrente(mat,aux,ite-1);  % Llama de nuevo a la función
    else
        fprintf('\n')
        disp('Convergencia encontrada')
        fprintf('Vector de entrada de clase %d\n',tipclas)
        % Termina con la función y sale para poder graficar
    end
end
```

- **poslin.m**

```
function [vec]=poslin(a)
    vaux=(a');
    tam=numel(a');
    for i=1:tam
        % Si el valor del vector es mayor a cero, este conservará su
valor
        % en caso contrario, guardará un cero.
        if vaux(i)>0
            vaux(i)=vaux(i);
        else
            vaux(i)=0;
        end
    end
    vec=vaux'; % Devuelve el nuevo vector
end
```

- **guardar.m**

```
function guardar(w)
    vo=fopen('w2.txt','w');     % Crea o abre el documento w2.txt
    for i=1:numel(w)
        fprintf(vo,'%d ',w(i)); % Guarda el contenido de w
    end
    fprintf(vo,'\n');
    fclose(vo);                 % Cierra el documento.
end
```

- **editar.m**

```
function editar(w)
    vo=fopen('w2.txt','a');     % Abre el documento w2.txt para editarlo
    for i=1:numel(w)
        fprintf(vo,'%d ',w(i)); % Gaurda los nuevos valores
    end
    fprintf(vo,'\n');
    fclose(vo);                 % Cierra el documento
end
```

- **graficar.m**

```
function graficar(w)
    [f,c]=size(w);  % Obtiene las dimensiones de la matriz
    ejequis=(0:f-1);    % Eje X para graficar de 0 hasta las filas
creadas
    numclas="";
    for i=1:c
        % Grafica las filas de la columna i que representan las clases
        plot(ejequis,w(:,i),'-o')
        numclas(i)=strcat('c_',string(i));
        hold on
    end
    legend(numclas);    % Inserta un menu con los tipos de clases
graficados
    title('Red de Hamming')
    xlabel('Iteraciones')
    ylabel('Valor')
end
```

- **Para los archivos txt:**

Los archivos con nombre pesos.txt, pesos2.txt y pesos3.txt son los archivos donde se encuentran los pesos sinápticos, y funcionan con el archivo entrada.txt como vector prototipo.

El archivo nar-man-w.txt es la matriz de pesos sinápticos, y nar-man-e.txt es el vector prototipo para dicha matriz.

El archivo now.txt es la matriz de pesos y el archivo noe.txt es el vector prototipo que pueden interactuar entre ellos.