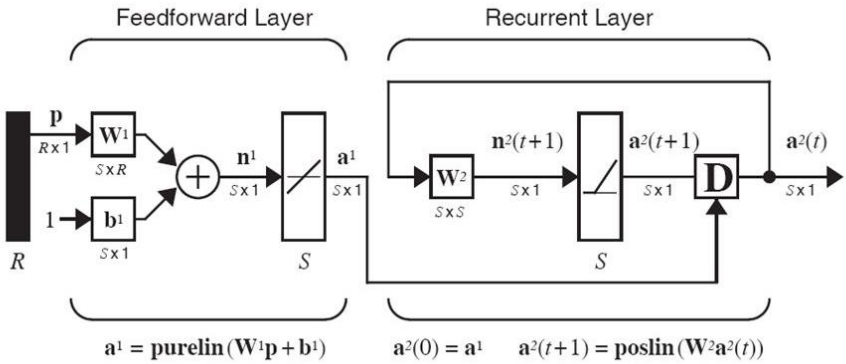
INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

NEURAL NETWORKS

PROF.: MARCO ANTONIO MORENO ARMENDÁRIZ
ALUMNO: ORTEGA VICTORIANO IVAN
No. DE LISTA: 29
GRUPO: 3CM2

RED DE HAMMING



Índice

INTRODUCCIÓN

La red de Hamming ilustrada en la Figura 1 es uno de los ejemplo más simples de aprendizaje competitivo, a pesar de ello su estructura es un poco compleja ya que emplea el concepto de capas recurrentes en su segunda capa y aunque hoy en día en redes de aprendizaje competitivo se ha simplificado este concepto con el uso de funciones de activación más sencillas, la red de Hamming representa uno de los primeros avances en este tipo de aprendizaje, convirtiéndola en un modelo obligado de referencia dentro de las redes de aprendizaje competitivo. Las neuronas en la capa de salida de esta red compiten unas con otras para determinar la ganadora, la cual índica el patrón prototipo más representativo en la entrada de la red. (Gutiérrez, s.f.)

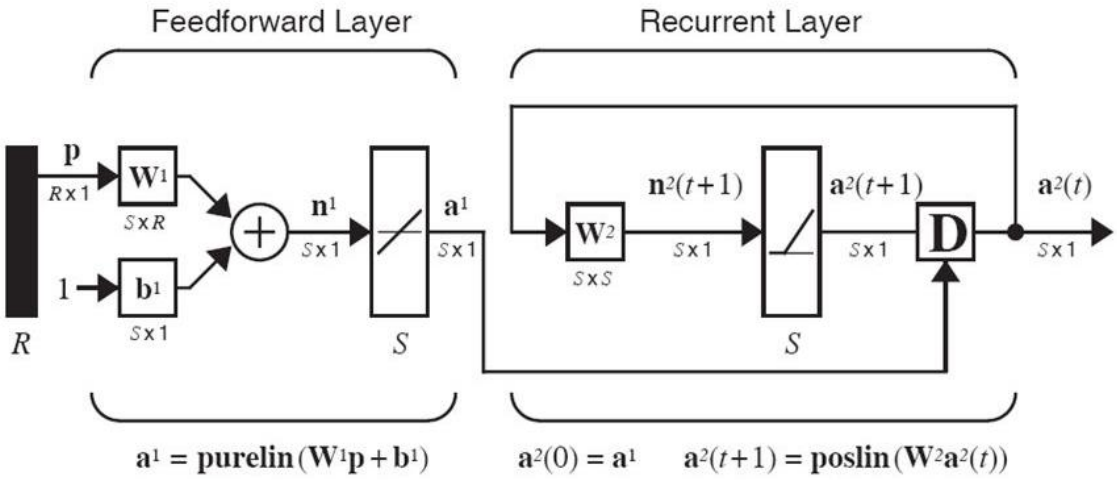


Figura 1. Arquitectura de la red de Hamming.

MARCO TEÓRICO

Cuando las entradas son binarias, entonces el uso de las redes de Hamming es muy atractivo. La red de Hamming selecciona un ganador de entre los patrones almacenados, que tienen la menor distancia de Hamming al vector de entrada. Donde la distancia de Hamming es el número de diferencias entre los bits de los dos vectores. [C., R., C., C., 23]

Redes de Aprendizaje Competitivo

En las redes con aprendizaje competitivo (Y Cooperativo), suele decirse que las neuronas compiten (Y Cooperan) unas con otras con el fin de llevar a cabo una tarea dada. Con este tipo de aprendizaje se pretende que cuando se presente a la red cierta información de entrada, sólo una de las neuronas de salida de la red, o una por cierto grupo de neuronas se active (Alcance su Valor de Respuesta Máximo). Por tanto, las neuronas compiten para activarse quedando finalmente una, o una por grupo, como neurona vencedora y el resto quedan anuladas y siendo forzadas a sus valores de respuesta mínimos. (Gutiérrez, Polilibro Redes Neuronales Artificiales 1, s.f.)

El objetivo de este aprendizaje es categorizar los datos que se introducen en la red, de esta forma las informaciones similares son clasificadas formando parte de la misma categoría y por tanto deben activar la misma neurona de salida. Las clases o categorías deben ser creadas por la propia red, puesto que se trata de un aprendizaje no supervisado a través de las correlaciones entre los datos de entrada. (Gutiérrez, Polilibro Redes Neuronales Artificiales 1, s.f.)

Problemas Con Las Redes De Aprendizaje Competitivo

Las redes competitivas, son bastante eficientes para resolver problemas de clasificación, sin embargo, presentan algunos problemas.

El primero es la elección de una tasa de aprendizaje que permita hallar un punto de equilibrio entre velocidad de convergencia y la estabilidad final de los vectores de peso. Una tasa de aprendizaje cercana a cero torna el aprendizaje muy lento, pero garantiza que cuando un vector haya alcanzado el centro de la clase objetivo, se mantendrá allí indefinidamente. En contraste, una tasa de aprendizaje cercana a uno genera un aprendizaje muy rápido, pero los vectores de peso continuarán oscilando aún después de que se haya alcanzado convergencia. (Gutiérrez, Polilibro Redes Neuronales Artificiales 1., s.f.)

La indecisión que se presenta al escoger la tasa de aprendizaje puede ser empleada como una ventaja si se inicia el entrenamiento con una tasa de aprendizaje alta y se decrementa en el transcurso del proceso de entrenamiento cuando sea necesario, desafortunadamente esta técnica no funciona si la red necesita continuamente ser adaptada a nuevos argumentos de los vectores de entrada. (Gutiérrez, Polilibro Redes Neuronales Artificiales 1, s.f.)

RESULTADOS EXPERIMENTALES

EXPERIMENTO 1: ENTRADA DE 2 RASGOS

Para este experimento se utilizó el siguiente conjunto de entrenamiento:

1	0
-1	-1
0	-1

En el programa de Matlab se solicita al usuario que ingrese el número de patrones prototipo con su dimensión correspondiente, esto para leer posteriormente dichos patrones del archivo que el usuario indique, y hacer ajustes en los tamaños de las matrices usadas en el programa. Para este experimento, indicamos que se usarán 3 patrones prototipo con una dimensión R=2 que se encuentran en el archivo "experimento1.txt" como se muestra en la figura 2:

```
Ingresa el número de vectores prototipo: 3
Ingresa el tamaño (r) de los vectores prototipo:  2
Ingresa el nombre del archivo de los vectores prototipo (sin extension .txt):  experimento1
Se abrió correctamente el archivo.
W =
     1     0
    -1    -1
     0    -1
```

Figura 2. Se ingresan valores al programa.

A continuación, se le solicita al usuario que indique el archivo en el que se encuentran los valores del vector bias, que para este ejemplo se encuentran en el archivo "bias1.txt" como se muestra en la figura 3:

```
Ingresa el nombre del archivo que contiene el vector bias (sin extension .txt):  bias1
Se abrió correctamente el archivo.
b =
     1
     1
     1
```

Figura 3. Se solicita el nombre del archivo que contiene al vector bias.

Una vez se hayan cargado estos valores, se le mostrará al usuario la matriz de pesos de la capa recurrente, donde el valor de épsilon se calculó de acuerdo con la expresión:

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

Donde S es el número de neuronas (o bien, el número de patrones prototipo) de la red.

Para este ejemplo, la matriz resultante es la siguiente:

```
Se usará la siguiente matriz en la capa recurrente:
    1.0000   -0.1813   -0.1813
   -0.1813    1.0000   -0.1813
   -0.1813   -0.1813    1.0000
```

Figura 4. Matriz de pesos de la capa recurrente de la red.

Por último, se le pide al usuario que ingrese los elementos del vector de entrada a propagar en la red, para finalmente mostrarle en cuantas iteraciones se llegó a la convergencia y a que clase convergió (las clases se numeraron de acuerdo con el patrón prototipo) y una gráfica que muestra la salida de la capa recurrente hasta que converge a una clase. Para este ejemplo, la entrada será el vector: $P = [1, 1]$, el cual se espera que converja a la clase 1, pues su distancia Hamming con el vector $[1,0]$ es de 1, que es menor a la distancia los otros patrones, la cual es de valor 2. El resultado de este proceso se muestra en la figura 5:

```
Ingrese los valores del vector de entrada a propagar:
P(1):1
P(2):1
a2 =
    2.5248
         0
         0

La red convergió exitosamente en 1 iteraciones.
El vector P converge a la clase 1.
```

Figura 5. Resultado de la red.

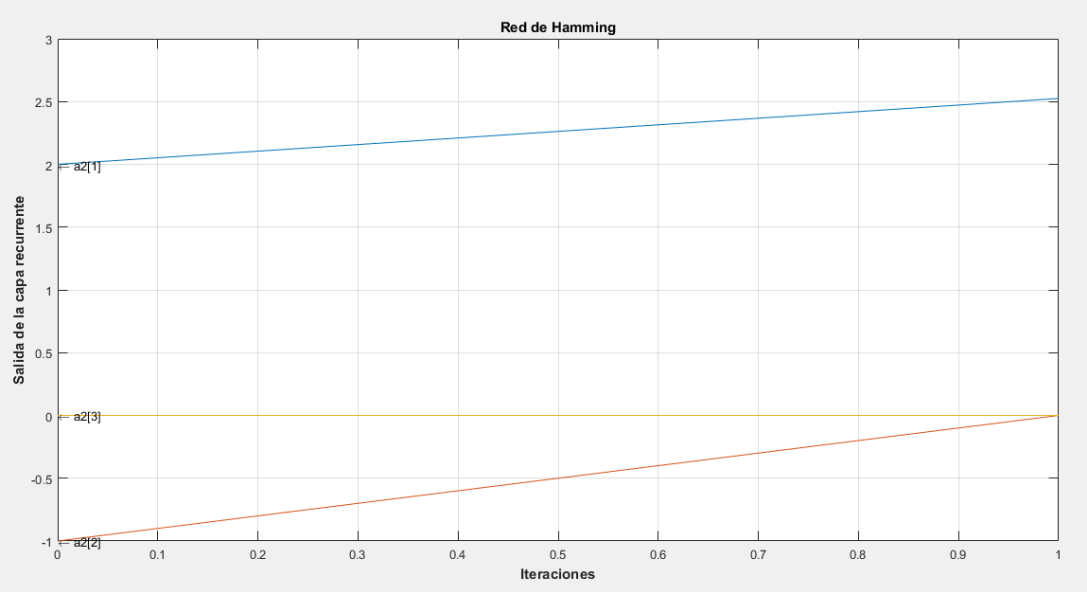


Figura 6. Evolución de la salida de la capa recurrente.

EXPERIMENTO 2: ENTRADA DE 3 RASGOS

Para este experimento se utilizó el siguiente conjunto de entrenamiento:

1	-1	-1
1	1	-1

Para este experimento, indicamos que se usarán 2 patrones prototipo con una dimensión R=3 que se encuentran en el archivo “experimento2.txt” como se muestra en la figura 7:

```
Command Window
  Ingresa el número de vectores prototipo: 2
  Ingresa el tamaño (r) de los vectores prototipo:  3
  Ingresa el nombre del archivo de los vectores prototipo (sin extension .txt):  experimento2
  Se abrió correctamente el archivo.
  W =
       1    -1    -1
       1     1    -1
```

Figura 7. Se ingresan valores al programa.

A continuación, se le solicita al usuario que indique el archivo en el que se encuentran los valores del vector bias, que para este ejemplo se encuentran en el archivo “bias2.txt” como se muestra en la figura 8:

```
Ingresa el nombre del archivo que contiene el vector bias (sin extension .txt):  bias2
Se abrió correctamente el archivo.
b =
     3
     3
```

Figura 8. Se solicita el nombre del archivo que contiene al vector bias.

Se muestra la matriz de la capa recurrente:

```
Se usará la siguiente matriz en la capa recurrente:
    1.0000   -0.0428
   -0.0428    1.0000
```

Figura 9. Matriz de pesos de la capa recurrente de la red.

Por último, se le pide al usuario que ingrese los elementos del vector de entrada a propagar en la red, para finalmente mostrarle en cuantas iteraciones se llegó a la convergencia y a que clase convergió (las clases se numeraron de acuerdo con el patrón prototipo) y una gráfica que muestra la salida de la capa recurrente hasta que converge a una clase. Para este ejemplo, la entrada será el vector: $P = [-1, -1, -1]$, el cual se espera que converja a la clase 1, pues su distancia Hamming con el vector $[1, -1, -1]$ es de 1, que es menor a la distancia el otro patrón, la cual es de valor 2. El resultado de este proceso se muestra en la figura 10:

```
Ingrese los valores del vector de entrada a propagar:
P(1):-1
P(2):-1
P(3):-1
a2 =
    3.4231
         0

La red convergió exitosamente en 13 iteraciones.
El vector P converge a la clase 1.
```

Figura 10. Resultado de la red.

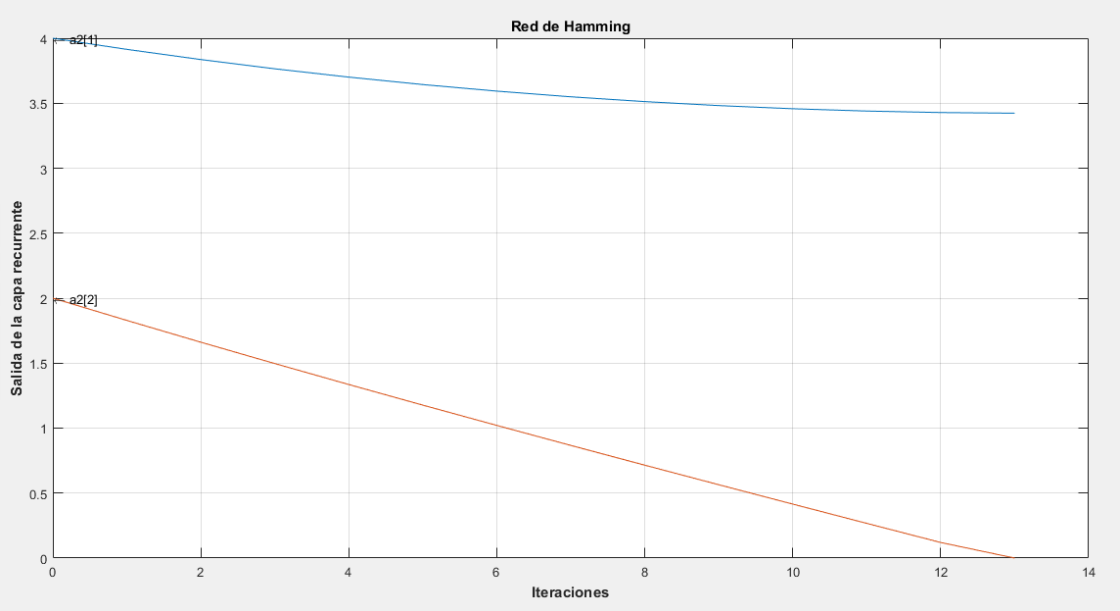


Figura 11. Evolución de la salida de la capa recurrente.

EXPERIMENTO 3: ENTRADA DE 4 RASGOS

Para este experimento se utilizó el siguiente conjunto de entrenamiento:

0	0	1	-1
-1	-1	-1	-1
1	1	1	1
1	0	-1	1

Para este experimento, indicamos que se usarán 4 patrones prototipo con una dimensión R=4 que se encuentran en el archivo “experimento3.txt” como se muestra en la figura 12:

```
Command Window
  Ingresa el número de vectores prototipo: 4
  Ingresa el tamaño (r) de los vectores prototipo:  4
  Ingresa el nombre del archivo de los vectores prototipo (sin extension .txt):  experimento3
  Se abrió correctamente el archivo.
  W =
       0     0     1    -1
      -1    -1    -1    -1
       1     1     1     1
       1     0    -1     1
```

Figura 12. Se ingresan valores al programa.

A continuación, se le solicita al usuario que indique el archivo en el que se encuentran los valores del vector bias, que para este ejemplo se encuentran en el archivo “bias3.txt” como se muestra en la figura 13:

```
Ingresa el nombre del archivo que contiene el vector bias (sin extension .txt):  bias3
Se abrió correctamente el archivo.
b =
     2
     2
     2
     2
```

Figura 13. Se solicita el nombre del archivo que contiene al vector bias.

Se muestra la matriz de la capa recurrente:

```
Se usará la siguiente matriz en la capa recurrente:
    1.0000   -0.3282   -0.3282   -0.3282
   -0.3282    1.0000   -0.3282   -0.3282
   -0.3282   -0.3282    1.0000   -0.3282
   -0.3282   -0.3282   -0.3282    1.0000
```

Figura 14. Matriz de pesos de la capa recurrente de la red.

Por último, se le pide al usuario que ingrese los elementos del vector de entrada a propagar en la red, para finalmente mostrarle en cuantas iteraciones se llegó a la convergencia y a que clase convergió (las clases se numeraron de acuerdo con el patrón prototipo) y una gráfica que muestra la salida de la capa recurrente hasta que converge a una clase. Para este ejemplo, la entrada será el vector: $P = [1, 1, 1, 0]$, el cual se espera que converja a la clase 3, pues su distancia Hamming con el vector $[1, 1, 1, 1]$ es de 1, que es menor a la distancia con el resto de patrones, la cual es de valor 3 y 4. El resultado de este proceso se muestra en la figura 15:

```
Ingrese los valores del vector de entrada a propagar:
P(1):1
P(2):1
P(3):1
P(4):0
a2 =
         0
         0
    3.3488
         0

La red convergió exitosamente en 2 iteraciones.
El vector P converge a la clase 3.
```

Figura 15. Resultado de la red.

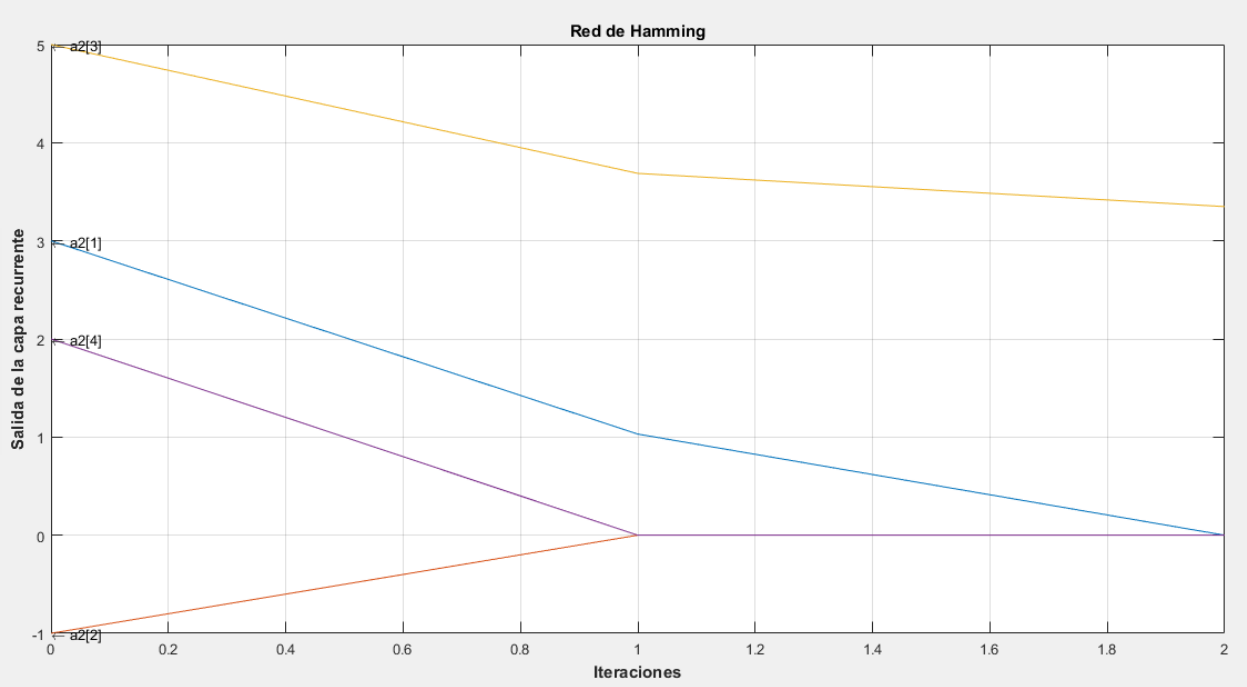


Figura 16. Evolución de la salida de la capa recurrente.

DISCUSIÓN DE LOS RESULTADOS

Los ejemplos utilizados para los experimentos se utilizaron de tal forma que pudieran dar resultados visibles de que la red converge a un valor, un caso en el que se daría problemas pudiera ser que el vector de entrada de la red fuera equidistante en distancia Hamming a 2 o más vectores prototipo, pues en este caso, la red de Hamming no resulta viable y pudiera dar un resultado erróneo, o simplemente no converger a una clase en específico.

Otro caso que tomar en cuenta es que no se realizaron muchas iteraciones para la mayoría de los ejemplos, a excepción del experimento 2, donde la convergencia se logró en 13 iteraciones, esto bien, se debe al valor de épsilon de la matriz de la capa recurrente. Ya que este determina la rapidez con la que va a converger a algún valor deseado.

Al realizar las primeras pruebas, pensé que el programa no funcionaba como debía, ya que había casos en los que no se llegaba a la convergencia de una clase, sin embargo, se debía a que olvidaba el caso en que el vector de entrada fuera equidistante en distancia Hamming a alguno de los vectores prototipo, así que empecé a probar con otros valores con los que no hubiera problema.

CONCLUSIONES

Comparado con las otras redes vistas hasta el momento, creo que la red de Hamming es una de las más sencillas en cuanto a la forma que se establecen los valores de pesos y bias, pues realmente para esta práctica no se usó ningún algoritmo de aprendizaje, pues esta red es una red neuronal artificial con aprendizaje no supervisado. Sin embargo, el cálculo de los valores de salida en la capa recurrente es lo que le da un poco de complejidad a esta red.

De acuerdo con los resultados y la teoría antes vista, pude comprobar que esta red es muy efectiva en la clasificación de patrones binarios (también utilizando -1), el único inconveniente de la red es que no resulta efectiva la clasificación si el vector de entrada es equidistante en distancia Hamming con 2 o más vectores prototipo, lo cual puede resultar en que la red entregue resultados erróneos, o bien que simplemente no se llegue a la convergencia con alguna clase y se quede en un ciclo infinito.

Referencias

C., G. B., R., S. D., C., E. D., & C., R. M. (2001 de Noviembre de 23). *Redes Neuronales*. Obtenido de Redes Neuronales: http://www.depi.itchihuahua.edu.mx/apacheco/lengs/ann/pagina_nueva_2.htm

Gutiérrez, H. F. (s.f.). *Polilibro Redes Neuronales Artificiales 1*. Obtenido de Capítulo 4: Redes Neuronales con aprendizaje no supervisado: http://www.hugo-inc.com/RNA/Unidad%204/4.1.2.html

Gutiérrez, H. F. (s.f.). *Polilibro Redes Neuronales Artificiales 1.* Obtenido de Polilibro Redes Neuronales Artificiales 1.: http://www.hugo-inc.com/RNA/Unidad%204/4.2.html

ANEXO

Para una mejor visualización del código, recomiendo visitar el siguiente repositorio de Github:

https://github.com/IvanovskyOrtega/Redes-Neuronales/tree/master/Practica-3/Red-de-Hamming

Hamming.m

```
clc
clear
% Se pide el número de vectores prototipo para asegurar la lectura correcta
% del archivo
numProt = input('Ingresa el número de vectores prototipo: ');

%De igual forma con el tamaño
r = input('Ingresa el tamaño (r) de los vectores prototipo:  ');

% Se pide el nombre del archivo sin extensión
nombreArchivo = input('Ingresa el nombre del archivo de los vectores prototipo (sin
extension .txt):  ','s');
nombreArchivo = sprintf('%s.txt',nombreArchivo);

% Se abre el archivo, si no lo puede abrir, solicita otro archivo válido
archivo = fopen(nombreArchivo,'r');
while (archivo == -1)
    disp('El archivo solicitado no existe o no se abrió correctamente.');
    nombreArchivo = input('Ingresa el nombre de un archivo válido:  ','s');
    archivo = fopen(nombreArchivo,'r');
end
disp('Se abrió correctamente el archivo.');
fclose(archivo);

% Si se abrió correctamente, se leen los prototipos y se cargan a la matriz
% de pesos.
W = dlmread(nombreArchivo);
dimensiones = size(W);

% El número de prototipos, es el número de neuronas de la red.
S = numProt;

% Si las dimensiones ingresadas no coinciden, se cambian por las correctas
% de acuerdo a lo que se leyó.
if(dimensiones(1) ~= numProt || dimensiones(2) ~= r)
    disp('Las dimensiones ingresadas no coinciden.');
    disp('Se utilizaran las siguientes: ');
    cadena = sprintf('\tNumero de Vectores prototipo: %d',dimensiones(1));
    disp(cadena);
    cadena2 = sprintf('\tR: %d',dimensiones(2));
    disp(cadena2);
    numProt = dimensiones(1);
    S = dimensiones(1);
    r = dimensiones(2);
end

% Se muestra la matriz de pesos
disp('W =');
disp(W);

% Se solicita el archivo que contiene los valores del vector bias.
archivoBias = input('Ingresa el nombre del archivo que contiene el vector bias (sin
extension .txt):  ','s');
```

```
archivoBias = sprintf('%s.txt',archivoBias);
archivo = fopen(archivoBias,'r');

% Si no existe, pide otro archivo válido.
while (archivo == -1)
    disp('El archivo solicitado no existe o no se abrió correctamente.');
    archivoBias = input('Ingresa el nombre de un archivo válido:  ','s');
    archivo = fopen(archivoBias,'r');
end
disp('Se abrió correctamente el archivo.');
fclose(archivo);

% Se cargan los valores del archivo al vector y se muestran en pantalla.
b = dlmread(archivoBias);
disp('b =');
disp(b);

% Se establece un valor aleatorio de epsilon para la matriz de pesos de la
% segunda capa.
epsilon = 0;
while epsilon == 0
    epsilon = mod(rand(),(1/S-1));
end
epsilonMatrix = ones(S,S)*epsilon;

% Se calcula la matriz de pesos de la segunda capa sumando las matrices
% triangulares de epsilon (inferior y superior) con una diagonal de 1's y
% se muestra en pantalla.
W_2 = tril(epsilonMatrix,-1)+triu(epsilonMatrix,1)+diag(ones(1,S));
disp('Se usará la siguiente matriz en la capa recurrente:');
disp(W_2);

% Se crea un vector de archivos para guardar los valores de graficación.
archivosGrafica = zeros(1,S);
for k=1:S
    nombre = sprintf('valoresDeGraficacion%d.txt',k);
    archivosGrafica(k) = fopen(nombre,'w');
end

% Se pide un vector de entrada para propagar en la red.
disp('Ingrese los valores del vector de entrada a propagar:');
P = zeros(r,1);
for i=1:r
    cad = sprintf('P(%d):',i);
    P(i) = input(cad);
end

% Inicia el aprendizaje
t = 0;
a1 = purelin(W*P+b);
a2_1 = a1;

% Se guardan los valores de salida de la 1ra capa a aun archivo
for k=1:S
    fprintf(archivosGrafica(k),'%f\r\n',a2_1(k));
end

a2_2 = ones(S,1)*(-99999);
while true
    a2_2 = poslin(W_2*a2_1);
    if isequal(a2_1,a2_2)
        break;
    end
```

```
    a2_1 = a2_2;

    % Se guardan las salidas de la capa recurrente.
    for k=1:S
        fprintf(archivosGrafica(k),'%f\r\n',a2_1(k));
    end

    t = t+1;
end
disp('a2 = ');
disp(a2_2);

% Se muestra al usuario en cuantas iteraciones convergió la entrada.
msg = sprintf('La red convergió exitosamente en %d iteraciones.',t);
disp(msg);

% Se busca a que clase pertenece el vector de entrada.
for k=1:S
    if a2_2(k) ~= 0
        msg = sprintf('El vector P converge a la clase %d.',k);
        disp(msg);
    end
end

% Se cierran los archivos de graficación
for k=1:S
    fclose(archivosGrafica(k));
end

% Inicia la graficación
rango = 0:t;
valores = zeros(S,t+1);
for k=1:S
    nombre = sprintf('valoresDeGraficacion%d.txt',k);
    aux = dlmread(nombre);
    valores(k,:) = aux';
end
figure
plot(rango,valores(1,:));
grid on
text(0,a1(1),'\leftarrow a2[1]');
title('Red de Hamming');
hold on
for k=2:S
    plot(rango,valores(k,:));
    cad = strcat('\leftarrow a2[',num2str(k),']');
    text(0,a1(k),cad);
end
hold off
xlabel('Iteraciones','FontWeight','bold')
ylabel('Salida de la capa recurrente','FontWeight','bold')
clear
```