



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

NEURAL NETWORKS

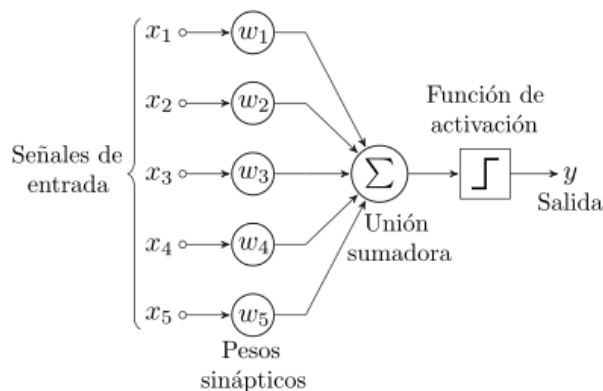
PROF.: MARCO ANTONIO MORENO ARMENDÁRIZ

ALUMNO: ORTEGA VICTORIANO IVAN

No. DE LISTA: 29

GRUPO: 3CM2

PERCEPTRÓN SIMPLE



Índice

| | |
|---|----|
| INTRODUCCIÓN..... | 3 |
| MARCO TEÓRICO..... | 4 |
| RESULTADOS EXPERIMENTALES | 5 |
| EXPERIMENTO 1: DOS CLASES (REGLA DE APRENDIZAJE)..... | 5 |
| EXPERIMENTO 2: CUATRO CLASES (REGLA DE APRENDIZAJE) | 9 |
| DISCUSIÓN DE LOS RESULTADOS..... | 13 |
| CONCLUSIONES | 14 |
| Referencias | 15 |
| ANEXO | 16 |
| PerceptronSimple.m..... | 16 |
| leerPatronesPrototipo.m | 23 |
| leerTargets.m | 23 |
| imprimirMatricesEnArchivo.m | 24 |

INTRODUCCIÓN

La primera red neuronal conocida, fue desarrollada en 1943 por Warren McCulloch y Walter Pitts; la cual consistía en una suma de las señales de entrada, multiplicadas por unos valores de pesos escogidos aleatoriamente. La entrada es comparada con un patrón preestablecido para determinar la salida de la red. Si en la comparación, la suma de las entradas multiplicadas por los pesos es mayor o igual que el patrón preestablecido la salida de la red es uno (1), en caso contrario la salida es cero (0). (Gutiérrez, Polilibro Redes Neuronales Artificiales 1, s.f.)

Al inicio del desarrollo de los sistemas de inteligencia artificial, se encontró gran similitud entre su comportamiento y el de los sistemas biológicos y en principio se creyó que este modelo podía computar cualquier función aritmética o lógica.

La red tipo Perceptrón fue inventada por el psicólogo Frank Rosenblatt en el año 1957. Su intención era ilustrar algunas propiedades fundamentales de los sistemas inteligentes en general, sin entrar en mayores detalles con respecto a condiciones específicas y desconocidas para organismos biológicos concretos. Rosenblatt creía que la conectividad existente en las redes biológicas tiene un elevado porcentaje de aleatoriedad, por lo que se oponía al análisis de McCulloch Pitts en el cual se empleaba lógica simbólica para analizar estructuras bastante idealizadas. Rosenblatt opinaba que la herramienta de análisis más apropiada era la teoría de probabilidades, y esto lo llevó a una teoría de separabilidad estadística que utilizaba para caracterizar las propiedades más visibles de estas redes de interconexión ligeramente aleatorias. (Gutiérrez, Polilibro Redes Neuronales Artificiales 1., s.f.)

MARCO TEÓRICO

La arquitectura de una red perceptrón consiste en una capa de S neuronas perceptrón, conectadas a r entradas a través de un conjunto de pesos W_{ij} como se muestra en la figura 1. Los índices de la red i y j indican que W_{ij} es la fuerza de conexión (pesos sinápticos) de la j -ésima entrada a la i -ésima neurona.

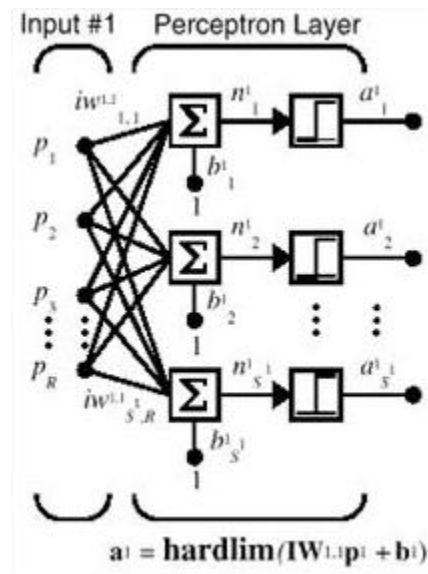


Figura 1. Arquitectura del perceptrón simple.

La red perceptrón puede tener únicamente una sola capa, debido a que la regla de aprendizaje del perceptrón es capaz de entrenar solamente una capa. Esta restricción coloca limitaciones en cuanto a lo que un perceptrón puede realizar computacionalmente.

La red tipo Perceptrón emplea principalmente dos funciones de transferencia, función escalón (hardlim) con salidas 1, 0 o función escalón simétrica (hardlims) con salidas 1, -1; su uso depende del valor de salida que se espera para la red; es decir, si la salida de la red es unipolar o bipolar; sin embargo, la función hardlims es preferida sobre la hardlim, ya que el tener un cero multiplicando algunas de los valores resultantes del producto de las entradas por el vector de pesos, ocasiona que estos no se actualicen y que el aprendizaje sea más lento. (Gutiérrez, Redes Neuronales Multicapa Con Aprendizaje Supervisado, s.f.)

El Perceptrón, al constar de una sola capa de entrada y otra de salida con una única neurona, tiene una capacidad de representación bastante limitada, este modelo sólo es capaz de discriminar patrones muy sencillos, patrones linealmente separables, el caso más conocido es la imposibilidad del Perceptrón de representar la función OR exclusiva (XOR). (Gutiérrez, Redes Neuronales Multicapa Con Aprendizaje Supervisado, s.f.)

RESULTADOS EXPERIMENTALES

EXPERIMENTO 1: DOS CLASES (REGLA DE APRENDIZAJE)

Para este experimento se utilizó el siguiente conjunto de entrenamiento que corresponde a la compuerta OR:

0 0 0

0 1 1

1 0 1

1 1 1

En el programa de Matlab se solicita al usuario que ingrese el método a utilizar en el programa, sin embargo, debido a que el método gráfico no fue implementado, esta opción queda fuera de uso. Para este experimento, indicamos que se usaremos la opción 2 que corresponde a la regla de aprendizaje, posteriormente se solicita el número de patrones prototipo y su dimensión. Luego se nos pregunta si la dimensión de los targets es mayor a 1, esto es debido al formato solicitado en la práctica donde los valores de los targets pueden ser un vector.

Indicaremos que serán 4 patrones prototipo con una dimensión $R=2$ y que la dimensión de los targets no es mayor a 1 (por lo tanto, será de 1), y nos pedirá la ubicación del archivo que contiene el conjunto de entrenamiento, para este caso será "experimento1.txt" como se muestra en la figura 2:

```
PERCEPTRÓN SIMPLE
1)Método gráfico
2)Regla de aprendizaje
Ingresa su elección: 2
REGLA DE APRENDIZAJE
Ingresa el número de patrones prototipo: 4
Ingresa la dimensión de los vectores prototipo: 2
¿La dimensión de los targets es mayor a 1? (s/n): n
Ingresa el nombre del archivo que contiene el Cto. de Entrenamiento\n
(Sin extensión)
Nombre del Archivo: experimento1
Prototipos =
    0    0
    0    1
    1    0
    1    1

Targets =
    0
    1
    1
    1
```

Figura 2. Se ingresan valores al programa y se muestran los vectores prototipo y los targets.

Posteriormente se solicita el número de clases a clasificar, esto con el objetivo de hacer el cálculo adecuado del número de neuronas de la red, recordando que S neuronas pueden clasificar 2^S clases, para nuestro ejemplo, indicamos que serán 2 clases, posteriormente se solicitan los valores de itmax y Eit, para el ejemplo usaremos itmax=50 y Eit=0 (sin error) como se muestra en la figura 3:

```
Ingresa el número de clases: 2
Itmax: 50
Eit: 0
```

Figura 3. Se solicitan el número de clases y los valores itmax y Eit.

Una vez se hayan establecido estos valores, se le mostrará al usuario la matriz de pesos y el vector bias iniciales, los cuales tendrán en sus entradas valores aleatorios.

```
Se usarán las siguientes matrices iniciales:
W =
    1.4897    1.4090

b =
    1.4172
```

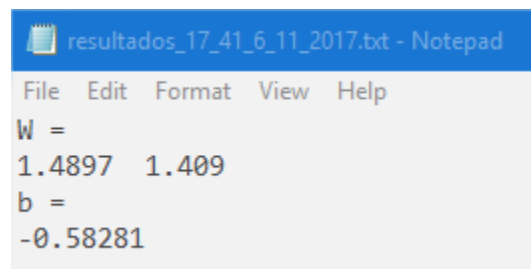
Figura 4. Matriz de pesos y vector bias iniciales.

Ya con dichos valores, iniciará el aprendizaje de la red, y se le indicará a la salida al usuario el número de iteración en la que se tuvo un aprendizaje exitoso (si es que lo hubo) y la matriz de pesos y vector bias finales (figura 5), los cuales se mandarán a un archivo de resultados (figura 6). Además, se mostrarán 3 graficas, una de la evolución de los pesos de la matriz de pesos (figura 7), la evolución del vector bias (figura 8) y la señal del error (figura 9).

```
Se tuvo un aprendizaje exitoso en la iteración 3.
W =
    1.4897    1.4090

b =
   -0.5828
```

Figura 5. Resultado de la red.



```
resultados_17_41_6_11_2017.txt - Notepad
File Edit Format View Help
W =
1.4897 1.409
b =
-0.58281
```

Figura 6. Resultado de la red en archivo.

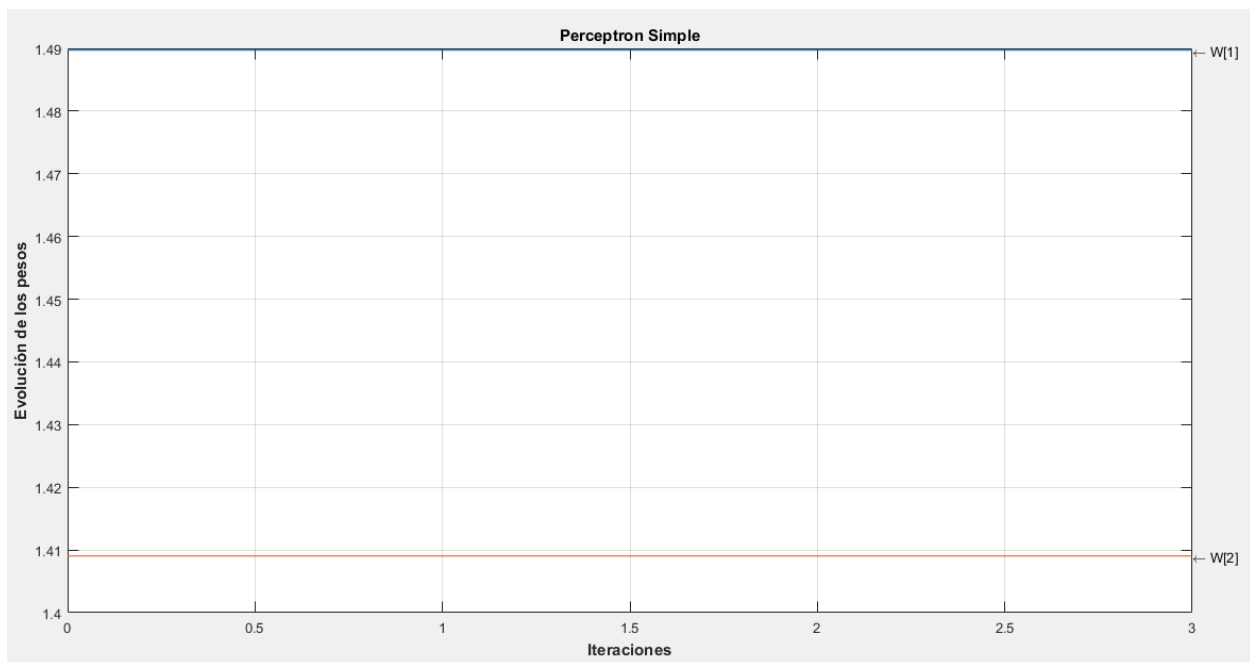


Figura 7. Evolución de los pesos de la matriz de pesos W .

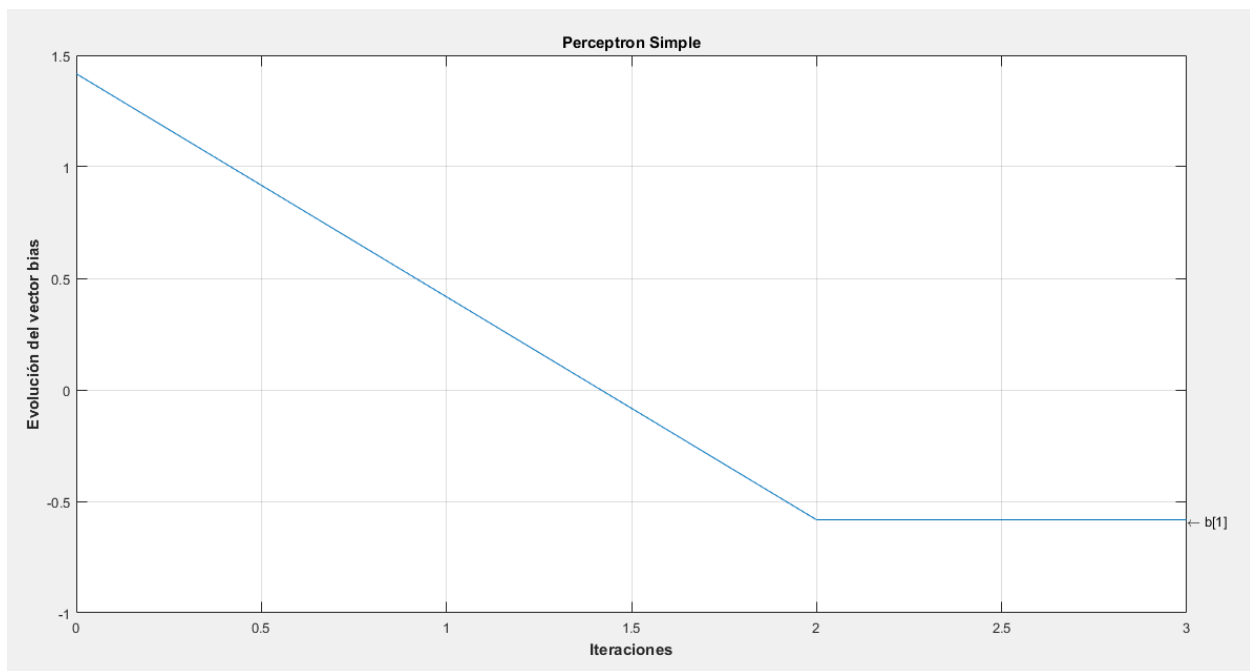


Figura 8. Evolución del vector bias b .

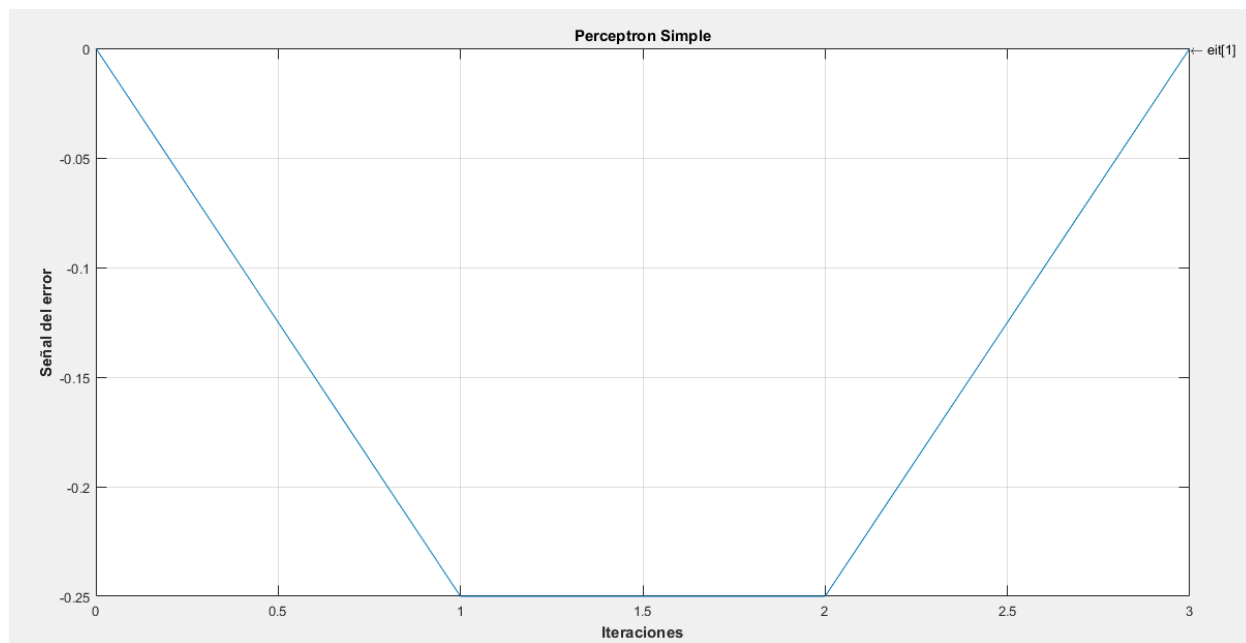


Figura 9. Señal del error.

EXPERIMENTO 2: CUATRO CLASES (REGLA DE APRENDIZAJE)

Para este experimento se utilizó el siguiente conjunto de entrenamiento:

$\{[1 \ 1], [-1 \ -1]\}$

$\{[1 \ 2], [-1 \ -1]\}$

$\{[2 \ -1], [-1 \ 1]\}$

$\{[2 \ 0], [-1 \ 1]\}$

$\{[-1 \ 2], [1 \ -1]\}$

$\{[-2 \ 1], [1 \ -1]\}$

$\{[-1 \ -1], [1 \ 1]\}$

$\{[-2 \ -2], [1 \ 1]\}$

Para este experimento, indicamos que se usarán 8 patrones prototipo con una dimensión $R=2$, y targets con dimensión mayor que 1, para este caso será de dimensión 2, y el conjunto de entrenamiento que estará ubicado en el archivo “experimento2.txt” como se muestra en la figura 10:

```
PERCEPTRÓN SIMPLE
1)Método gráfico
2)Regla de aprendizaje
Ingrese su elección: 2
REGLA DE APRENDIZAJE
Ingresa el número de patrones prototipo: 8
Ingresa la dimensión de los vectores prototipo: 2
¿La dimensión de los targets es mayor a 1? (s/n): s
Ingresa la dimensión de los targets: 2
Ingresa el nombre del archivo que contiene el Cto. de Entrenamiento\n
(Sin extensión)
Nombre del Archivo: experimento2

Prototipos =
  1    1
  1    2
  2   -1
  2    0
 -1    2
 -2    1
 -1   -1
 -2   -2

Targets =
  0    0
  0    0
  0    1
  0    1
  1    0
  1    0
  1    1
  1    1
```

Figura 10. Se ingresan valores al programa y se muestran los vectores prototipo y los targets.

Indicamos que serán 4 clases por clasificar, posteriormente se solicitan los valores de itmax y Eit, para el ejemplo usaremos itmax=50 y Eit=0 (sin error) como se muestra en la figura 11:

```
Ingresa el número de clases: 4
Itmax: 50
Eit: 0
```

Figura 11. Se solicitan el número de clases y los valores itmax y Eit.

Una vez se hayan establecido estos valores, se le mostrará al usuario la matriz de pesos y el vector bias iniciales, los cuales tendrán en sus entradas valores aleatorios.

```
Se usarán las siguientes matrices iniciales:
W =
    0.7269    0.2939
   -0.3034   -0.7873

b =
    0.8884
   -1.1471
```

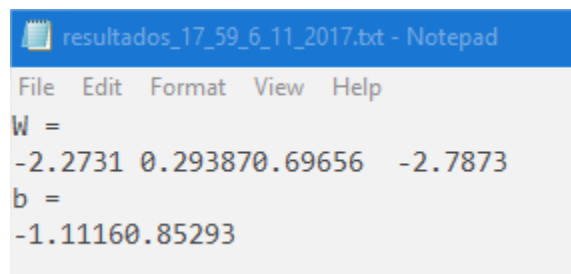
Figura 12. Matriz de pesos y vector bias iniciales.

Ya con dichos valores, iniciará el aprendizaje de la red, y se le indicará a la salida al usuario el número de iteración en la que se tuvo un aprendizaje exitoso (si es que lo hubo) y la matriz de pesos y vector bias finales (figura 13), los cuales se mandarán a un archivo de resultados (figura 14). Además, se mostrarán 3 graficas, una de la evolución de los pesos de la matriz de pesos (figura 15), la evolución del vector bias (figura 16) y la señal del error (figura 17).

```
Se tuvo un aprendizaje exitoso en la iteración 2.
W =
   -2.2731    0.2939
    0.6966   -2.7873

b =
   -1.1116
    0.8529
```

Figura 13. Resultado de la red.



```
File Edit Format View Help
W =
-2.2731 0.293870.69656 -2.7873
b =
-1.11160.85293
```

Figura 14. Resultado de la red en archivo.

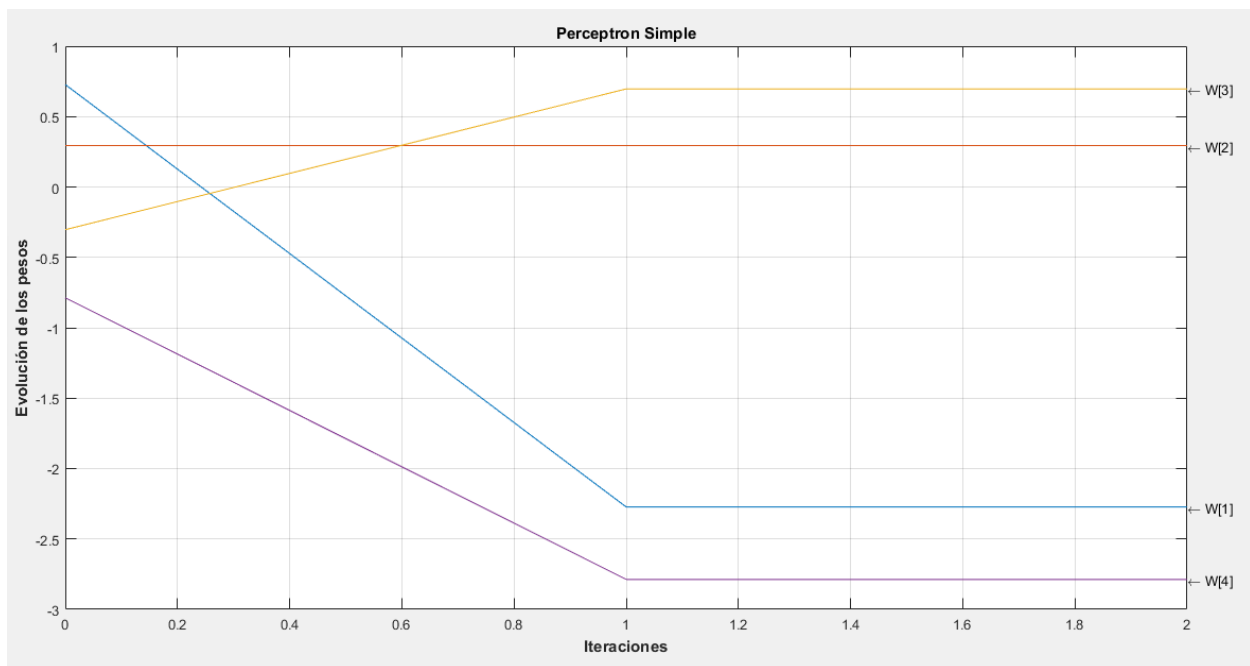


Figura 15. Evolución de los pesos de la matriz de pesos W .

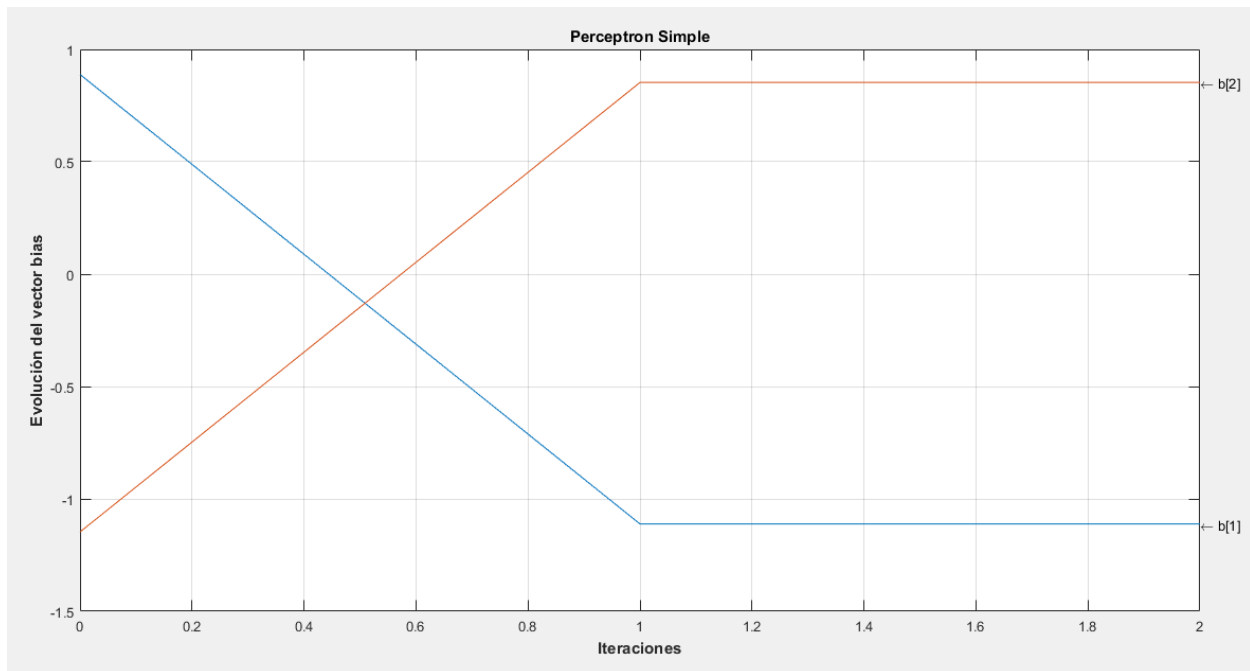


Figura 16. Evolución del vector bias b .

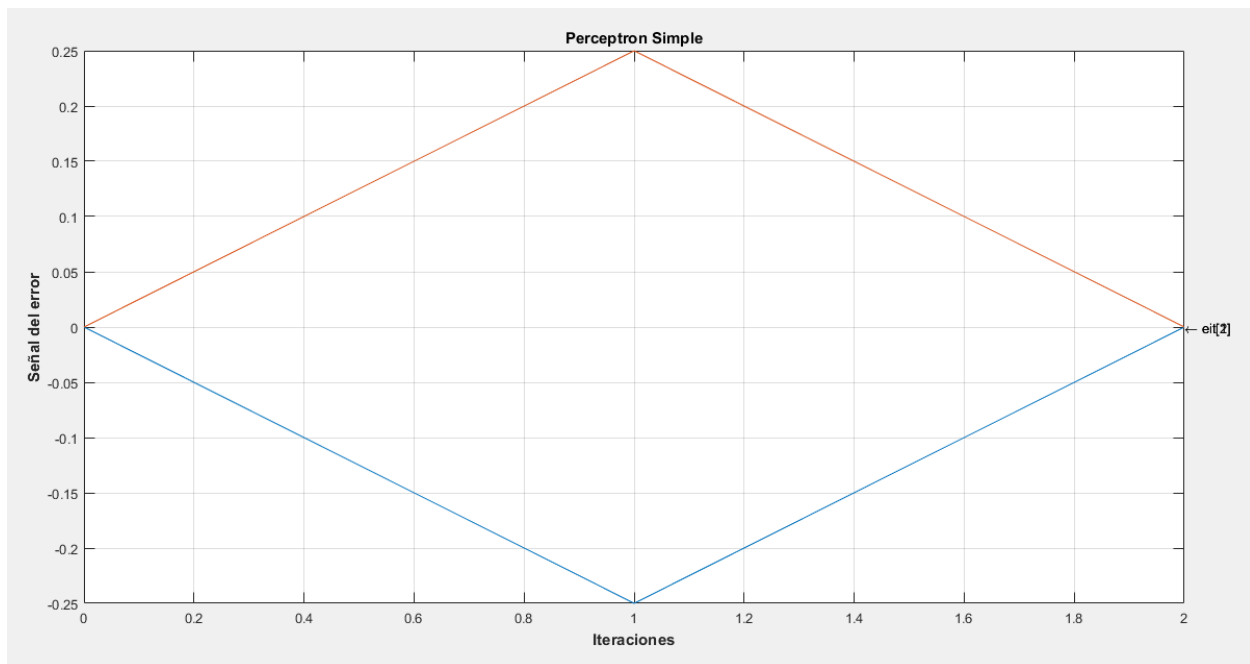


Figura 17. Señal del error.

DISCUSIÓN DE LOS RESULTADOS

Al igual que sucedió en los ejemplos de la red Hamming, ambos convergieron en pocas iteraciones aun indicando que el error fuese nulo, es decir, de 0. Ambos ejemplos eran linealmente separables, los cuales fueron vistos en clase. Esto se hizo con el fin de comprobar que el funcionamiento de la red fuera adecuado para problemas que se resolvieron mediante el método gráfico en clase, ya que, si estos pudieron ser resueltos por método gráfico, es posible resolverlos mediante la regla de aprendizaje. Si bien los resultados no son los mismos (lo cual es evidente por el uso de valores aleatorios tanto en método gráfico como en regla de aprendizaje), ambos métodos proporcionan una solución válida para el problema, dentro del conjunto (en principio, infinito) de soluciones existentes.

CONCLUSIONES

En esta práctica se pudo observar el funcionamiento de la regla de aprendizaje del perceptrón simple para problemas linealmente separables, a los cuales, este tipo de red está limitada a resolver. El ejemplo clásico con el que se demuestra dicha limitación es con la compuerta XOR, que a simple vista parece un problema inocente y sencillo de resolver, pero que no puede ser resuelto, a menos que se utilice más de un perceptrón (perceptrón multicapa).

A forma de experimento adicional, realicé una prueba ingresando como conjunto de entrenamiento a la compuerta XOR, y a pesar que en algunos casos, desde la 4ta iteración la señal del error era aparentemente cero, no se cumplía el aprendizaje, puesto que había casos en los que no se clasificaban bien los prototipo en cada iteración, ya que para el perceptrón la condición de finalización es que en cierta iteración deben de clasificarse correctamente todos los patrones prototipo y la señal del error sea igual al valor establecido de Eit, o menor a Eit y mayor que 0 que es un error aceptable, sin embargo nunca se cumplía que en alguna iteración se clasificaran correctamente todos los patrones prototipo.

Referencias

Gutiérrez, H. F. (s.f.). *Polilibro de Redes Neuronales 1*. Obtenido de Redes Neuronales Multicapa con Aprendizaje Supervisado: El Perceptrón: <http://www.hugo-inc.com/RNA/Unidad%202/2.1.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>

Gutiérrez, H. F. (s.f.). *Redes Neuronales Multicapa Con Aprendizaje Supervisado*. Obtenido de Arquitectura del Perceptrón: <http://www.hugo-inc.com/RNA/Unidad%202/2.1.1.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/Perceptron-Simple>

PerceptronSimple.m

```
clc
clear
disp('PERCEPTRÓN SIMPLE');

% El método gráfico no se pudo implementar, visitar el siguiente repositorio
% para ver una versión (No muy gráfica) de este método.
tipo = input('1)Método gráfico\n2)Regla de aprendizaje\nIngrese su elección: ','s');
switch(tipo)
    case '1'
        disp('No implementado :(');
    case '2'
        disp('REGLA DE APRENDIZAJE');
        % Se solicita el número de patrones prototipo y su dimensión.
        numProt = input('Ingresa el número de patrones prototipo: ');
        R = input('Ingresa la dimensión de los vectores prototipo: ');

        % Se pregunta si la dimensión de los targets es mayor a 2, debido
        % al formato especificado en la práctica.
        res = input('¿La dimensión de los targets es mayor a 1? (s/n): ','s');

        % Si es mayor a 1
        if strcmp(res,'s') || strcmp(res,'S')
            % Se pide la dimensión de los targets
            targetDim = input('Ingresa la dimensión de los targets: ');

            % Se solicita el nombre del archivo que contiene el conjunto de
            % entrenamiento sin ingresar la extensión (
            % Se usa .txt por defecto.)
            disp('Ingresa el nombre del archivo que contiene el Cto. de
Entrenamiento\n');
            disp('(Sin extensión)');
            nombre = input('Nombre del Archivo: ','s');
            nombre = strcat(nombre, '.txt');

            % Se leen los patrones prototipo y los targets del archivo
            % indicado y se muestran en pantalla.
            prototipos = leerPatronesPrototipo(nombre, targetDim, numProt, R);
            targets = leerTargets(nombre, targetDim, numProt, R);
            disp('Prototipos =');
            disp(prototipos);
            disp('Targets =');
            disp(targets);

            % Se solicita al usuario el número de clases a las que va a
            % clasificar para calcular el número de neuronas. Recordando
            % que S neuronas separan 2^S clases.
            numeroDeClases = input('Ingresa el número de clases: ');
            n = 1;
            S = 1.0;
            while numeroDeClases > power(2,n)
                n = n+1;
                S = S+1.0;
            end
        end
    end
end
```



```

% Se solicitan los valores de itmax y Eit
itmax = input('Itmax: ');
Eit = input('Eit: ');
Eit = ones(S,1)*Eit;

% Se inicializan la matriz de pesos y el vector bias con
% valores aleatorios y se muestran en pantalla.
W = randn(S,R);
b = randn(S,1);
disp('Se usarán las siguientes matrices iniciales:');
disp('W =');
disp(W);
disp('b =');
disp(b);

% Se usa un vector auxiliar para comparar con la señal del
% error.
aux = zeros(S,1);

% Se abren los archivos necesarios para guardar los valores de
% graficación.
pesos = zeros(1,S*R);
bias = zeros(1,S);
errores = zeros(1,S);
for k=1:S*R
    nombre = sprintf('pesos%d.txt',k);
    pesos(k) = fopen(nombre,'w');
end
for k=1:S
    nombre = sprintf('bias%d.txt',k);
    bias(k) = fopen(nombre,'w');
end
for k=1:S
    nombre = sprintf('error%d.txt',k);
    errores(k) = fopen(nombre,'w');
end

% Se imprimen a archivo los valores iniciales.
k=1;
for l=1:S
    for m=1:R
        fprintf(pesos(k), '%f\r\n',W(l,m));
        k = k+1;
    end
end
for l=1:S
    fprintf(bias(l), '%f\r\n',b(l));
end
for l=1:S
    fprintf(errores(l), '%f\r\n',0);
end

% Inicia el aprendizaje
for i=1:itmax
    eit = zeros(S,1);
    t = 0;
    for j=1:numProt
        a = hardlim(W*prototipos(j,:)'+b);
        ej = targets(j,:)-a;
        if ~isequal(ej,aux)
            t = t+1;
            W = W+ej*prototipos(j,:);
        end
    end
    eit = eit + t;
end

```

```

        b = b+ej;
        eit = eit+(1/numProt)*ej;
    end
end

% Se imprimen valores a archivo
k=1;
for l=1:S
    for m=1:R
        fprintf(pesos(k), '%f\r\n', W(l,m));
        k = k+1;
    end
end
for l=1:S
    fprintf(bias(l), '%f\r\n', b(l));
end
for l=1:S
    fprintf(erroses(l), '%f\r\n', eit(l));
end

% 1er criterio de finalización: Si la señal del error es 0.
if isequal(eit,Eit) && t == 0
    fprintf('Se tuvo un aprendizaje exitoso en la iteración %d.\n',i);
    break;
end

% 2dor criterio de finalización: Si la señal del error es
% mayor a 0 y menor a Eit.
if (1/S)*sum(eit) < (1/S)*sum(Eit) && (1/S)*sum(eit) > 0
    fprintf('Se tuvo un aprendizaje con 3/4 de probabilidad de éxito en
la iteración %d.\n',i);
    break;
end
end
itmax = i;

% Se muestran los valores finales de la matriz de pesos y el
% vector bias.
disp('W =');
disp(W);
disp('b =');
disp(b);

% Se cierran los archivos
for k=1:S*R
    fclose(pesos(k));
end
for k=1:S
    fclose(bias(k));
end
for k=1:S
    fclose(erroses(k));
end

% Inicia la graficación
numeroDeArchivos = S*R;
valores = zeros(numeroDeArchivos,itmax+1);
for k=1:numeroDeArchivos
    nombre = sprintf('pesos%d.txt',k);
    aux = dlmread(nombre);
    valores(k,:) = aux';
end
figure

```

```

rango = (0:itmax);
plot(rango, valores(1, :));
grid on
text(itmax, valores(1, itmax+1), '\leftarrow W[1]');
title('Perceptron Simple');
hold on
for k=2:numeroDeArchivos
    plot(rango, valores(k, :));
    cad = strcat('\leftarrow W[' , num2str(k) , ']');
    text(itmax, valores(k, itmax+1), cad);
end
hold off
xlabel('Iteraciones', 'FontWeight', 'bold')
ylabel('Evolución de los pesos', 'FontWeight', 'bold')
numeroDeArchivos = S;
valores = zeros(numeroDeArchivos, itmax+1);
for k=1:numeroDeArchivos
    nombre = sprintf('bias%d.txt', k);
    aux = dlmread(nombre);
    valores(k, :) = aux';
end
figure
plot(rango, valores(1, :));
grid on
text(itmax, valores(1, itmax+1), '\leftarrow b[1]');
title('Perceptron Simple');
hold on
for k=2:numeroDeArchivos
    plot(rango, valores(k, :));
    cad = strcat('\leftarrow b[' , num2str(k) , ']');
    text(itmax, valores(k, itmax+1), cad);
end
hold off
xlabel('Iteraciones', 'FontWeight', 'bold')
ylabel('Evolución del vector bias', 'FontWeight', 'bold')
numeroDeArchivos = S;
valores = zeros(numeroDeArchivos, itmax+1);
for k=1:numeroDeArchivos
    nombre = sprintf('error%d.txt', k);
    aux = dlmread(nombre);
    valores(k, :) = aux';
end
figure
plot(rango, valores(1, :));
grid on
text(itmax, valores(1, itmax+1), '\leftarrow eit[1]');
title('Perceptron Simple');
hold on
for k=2:numeroDeArchivos
    plot(rango, valores(k, :));
    cad = strcat('\leftarrow eit[' , num2str(k) , ']');
    text(itmax, valores(k, itmax+1), cad);
end
hold off
xlabel('Iteraciones', 'FontWeight', 'bold')
ylabel('Señal del error', 'FontWeight', 'bold')
imprimirMatricesEnArchivo(W, b, 'c');

% Si la dimensión de los targets es 1.
else
    % Se solicita el archivo que contiene el conjunto de
    % entrenamiento.

```

```

disp('Ingresa el nombre del archivo que contiene el Cto. de
Entrenamiento\n');
disp('(Sin extensión)');
nombre = input('Nombre del Archivo: ','s');
nombre = strcat(nombre, '.txt');
ctoDeEntrenamiento = dlmread(nombre);
dimensiones = size(ctoDeEntrenamiento);
R = dimensiones(2)-1;

% Se leen los vectores prototipo y los targets y se muestran en
% pantalla.
prototipos = ctoDeEntrenamiento(:,1:R);
targets = ctoDeEntrenamiento(:,R+1);
disp('Prototipos =');
disp(prototipos);
disp('Targets =');
disp(targets);

% Se solicita el numero de clases a clasificar para calcular el
% número de neuronas de la red.
numeroDeClases = input('Ingresa el número de clases: ');
n = 1;
S = 1.0;
while numeroDeClases > power(2,n)
    n = n+1;
    S = S+1.0;
end

% Se solicitan los valores itmax y Eit
itmax = input('Itmax: ');
Eit = input('Eit: ');

% Se inicializan la matriz de pesos y el vector bias con
% valores aleatorios de la distribución normal y se muestran en
% pantalla.
W = randn(S,R);
b = randn(S,1);
disp('Se usarán las siguientes matrices iniciales:');
disp('W =');
disp(W);
disp('b =');
disp(b);

% Se abren los archivos necesarios para guardar los valores de
% graficación.
pesos = zeros(1,S*R);
bias = zeros(1,S);
errores = zeros(1);
for k=1:S*R
    nombre = sprintf('pesos%d.txt',k);
    pesos(k) = fopen(nombre, 'w');
end
for k=1:S
    nombre = sprintf('bias%d.txt',k);
    bias(k) = fopen(nombre, 'w');
end
for k=1:S
    nombre = sprintf('error%d.txt',k);
    errores(k) = fopen(nombre, 'w');
end

% Se imprimen a archivo los valores iniciales
k=1;

```

```

for l=1:S
    for m=1:R
        fprintf(pesos(k), '%f\r\n', W(l,m));
        k = k+1;
    end
end
for l=1:S
    fprintf(bias(l), '%f\r\n', b(l));
end
fprintf(errores, '%f\r\n', 0);

% Inicia el aprendizaje
for i=1:itmax
    eit = 0;
    t = 0;
    for j=1:numProt
        a = hardlim(W*prototipos(j,:)'+b);
        ej = targets(j,:)-a;
        if ej ~= 0
            t = t+1;
            W = W+ej*prototipos(j,:);
            b = b+ej;
            eit = eit+(1/numProt)*ej;
        end
    end
end

% Se imprimen valores a archivo
k=1;
for l=1:S
    for m=1:R
        fprintf(pesos(k), '%f\r\n', W(l,m));
        k = k+1;
    end
end
for l=1:S
    fprintf(bias(l), '%f\r\n', b(l));
end
for l=1:S
    fprintf(errores(l), '%f\r\n', eit(l));
end

% 1er criterio de finalización
if isequal(eit,Eit) && t == 0
    fprintf('Se tuvo un aprendizaje exitoso en la iteración %d.\n', i);
    break;
end

% Segundo criterio de finalización
if (1/S)*sum(eit) < (1/S)*sum(Eit) && (1/S)*sum(eit) > 0
    fprintf('Se tuvo un aprendizaje con 3/4 de probabilidad de éxito en la iteración %d.\n', i);
    break;
end
end
itmax = i;

% Se muestran la matriz de pesos y el vector bias finales.
disp('W =');
disp(W);
disp('b =');
disp(b);

% Se cierran los archivos de graficación.

```

```

for k=1:S*R
    fclose(pesos(k));
end
for k=1:S
    fclose(bias(k));
end
for k=1:S
    fclose(erroses(k));
end

% Inicia la graficación
numeroDeArchivos = S*R;
valores = zeros(numeroDeArchivos,itmax+1);
for k=1:numeroDeArchivos
    nombre = sprintf('pesos%d.txt',k);
    aux = dlmread(nombre);
    valores(k,:) = aux';
end
figure
rango = (0:itmax);
plot(rango,valores(1,:));
grid on
text(itmax,valores(1,itmax+1),'\leftarrow W[1]');
title('Perceptron Simple');
hold on
for k=2:numeroDeArchivos
    plot(rango,valores(k,:));
    cad = strcat('\leftarrow W[' ,num2str(k),']');
    text(itmax,valores(k,itmax+1),cad);
end
hold off
xlabel('Iteraciones','FontWeight','bold')
ylabel('Evolución de los pesos','FontWeight','bold')
numeroDeArchivos = S;
valores = zeros(numeroDeArchivos,itmax+1);
for k=1:numeroDeArchivos
    nombre = sprintf('bias%d.txt',k);
    aux = dlmread(nombre);
    valores(k,:) = aux';
end
figure
plot(rango,valores(1,:));
grid on
text(itmax,valores(1,itmax+1),'\leftarrow b[1]');
title('Perceptron Simple');
hold on
for k=2:numeroDeArchivos
    plot(rango,valores(k,:));
    cad = strcat('\leftarrow b[' ,num2str(k),']');
    text(itmax,valores(k,itmax+1),cad);
end
hold off
xlabel('Iteraciones','FontWeight','bold')
ylabel('Evolución del vector bias','FontWeight','bold')
numeroDeArchivos = S;
valores = zeros(numeroDeArchivos,itmax+1);
for k=1:numeroDeArchivos
    nombre = sprintf('error%d.txt',k);
    aux = dlmread(nombre);
    valores(k,:) = aux';
end
figure
plot(rango,valores(1,:));

```

```

        grid on
        text(itmax, valores(1, itmax+1), '\leftarrow eit[1]');
        title('Perceptron Simple');
        hold on
        for k=2:numeroDeArchivos
            plot(rango, valores(k, :));
            cad = strcat('\leftarrow eit[' , num2str(k) , ']');
            text(itmax, valores(k, itmax+1), cad);
        end
        hold off
        xlabel('Iteraciones', 'FontWeight', 'bold')
        ylabel('Señal del error', 'FontWeight', 'bold')
        imprimirMatricesEnArchivo(W, b, 'c');
    end
otherwise
    display('Opción inválida');
end

```

leerPatronesPrototipo.m

```

% Esta función lee los valores de los vectores prototipo p de un archivo
% .txt del conjunto de entrenamiento que se encuentren en un formato:
% {[p1 p2 p3 ... pn], [t1 t2 t3 ... tn]}
function A = leerPatronesPrototipo(nombre, targetDim, numProt, protDim)
    archivo = fopen(nombre, 'r');
    A = zeros(numProt, protDim);
    disp([numProt protDim]);
    while archivo == -1
        nombre = input('Ingrese un nombre válido: ', 's');
        archivo = fopen(nombre, 'r');
    end
    for i=1:numProt
        fscanf(archivo, '{['];
        A(i, :) = fscanf(archivo, '%d');
        fscanf(archivo, '], [');
        fscanf(archivo, '%d');
        fscanf(archivo, ']]\n');
    end
    fclose(archivo);
end

```

leerTargets.m

```

% Esta función lee los valores target t de un archivo .txt del conjunto de
% entrenamiento que se encuentren en un formato:
% {[p1 p2 p3 ... pn], [t1 t2 t3 ... tn]}
function A = leerTargets(nombre, targetDim, numProt, protDim)
    archivo = fopen(nombre, 'r');
    while archivo == -1
        nombre = input('Ingrese un nombre válido: ', 's');
        archivo = fopen(nombre, 'r');
    end
    A = zeros(numProt, targetDim);
    for i=1:numProt
        fscanf(archivo, '{[');
        fscanf(archivo, ' %d');
        fscanf(archivo, '], [');
        A(i, :) = fscanf(archivo, '%d');
    end

```

```
        fscanf(archivo, ']]\n');
end
fclose(archivo);
end
```

imprimirMatricesEnArchivo.m

```
function imprimirMatricesEnArchivo(W,b,tipo)
    switch(tipo)
        case 'c'
            imprimirConBias(W,b)
        case 's'
            imprimirSinBias(W)
    end
end

function imprimirConBias(W,b)
t = clock;
nombre = sprintf('resultados_%d_%d_%d_%d_%d.txt',t(4),t(5),t(3),t(2),t(1));
resultados = fopen(nombre,'w');
fprintf(resultados, 'W = \r\n');
fclose(resultados);
dlmwrite(nombre,W, '-append', 'delimiter', '\t');
resultados = fopen(nombre,'a');
fprintf(resultados, '\r\n');
fprintf(resultados, 'b = \r\n');
fclose(resultados);
dlmwrite(nombre,b, '-append', 'delimiter', '\t');
end

function imprimirSinBias(W)
t = clock;
nombre = sprintf('resultados_%d_%d_%d_%d_%d.txt',t(4),t(5),t(3),t(2),t(1));
resultados = fopen(nombre,'w');
fprintf(resultados, 'W = \r\n');
fclose(resultados);
dlmwrite(nombre,W, '-append', 'delimiter', '\t');
end
```