INSTITUTO POLITÉCNICO NACIONAL

Centro de Investigación en Computación

# Introducción a las Redes Neuronales Artificiales

## Segunda actividad

**Presenta:**

Sebastián Cipriano Damián

**Docente:**

Dr. Juan Humberto Sossa Azuela

**Grupo CIC:** SUM1

**Grupo ESCOM:** 3CM20

**CDMX, 3 de Diciembre, 2021**

## 1. Dada la siguiente imagen con dos clases A y B:

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | | | | | | | | | | | | |
| 11 | | | | | | | | | | | | |
| 10 | | | | B | | | B | | | | | |
| 9 | | | | | | | | | | | | |
| 8 | | | | | | | | | B | | | |
| 7 | | | | | | | | | | | | |
| 6 | | A | | | | | | | | | | |
| 5 | | | | | | | | | | | | |
| 4 | | | | A | | | | | | | | |
| 3 | | | | | | A | | | | | | |
| 2 | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

a) Entrene un perceptron con función de activación limitadora y con bias mediante la regla del perceptrón. Proponga una vector de pesos iniciales $W_0 = (w_1, w_2, w_3)^T$ y un parámetro de aprendizaje $\alpha$. Muestre la línea inicial, el vector de pesos final, así como la línea de separación final.

A continuación, se muestra la línea de separación inicial y la línea de separación final, así como los patrones con su respectiva clase. Recordar que la clase A es representada por los puntos azules y la clase B es representada por los puntos de color rojos.

Para este problema se propuso el vector de pesos W = { -0.7, 0.5, -0.4}, mientas que el factor de aprendizaje se definió como 0.1 y el número de épocas se propuso como 50.

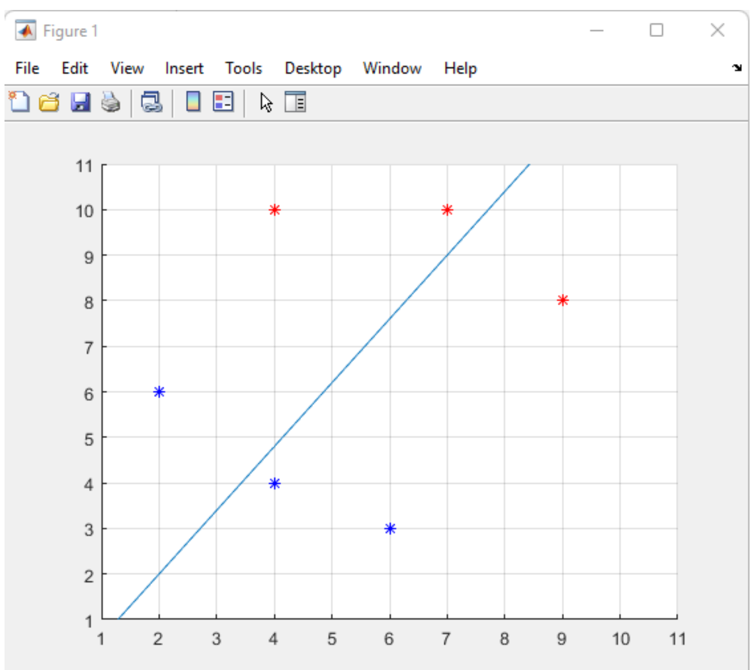


Imagen 1.1. Grafica correspondiente a la línea de separación original.

Los pesos de la línea de separación final son los siguientes:

```
>> RN

Pesos finales:
    0.1000    0.2000    2.3000
```

A continuación, se muestra la representación gráfica de la línea de separación final:

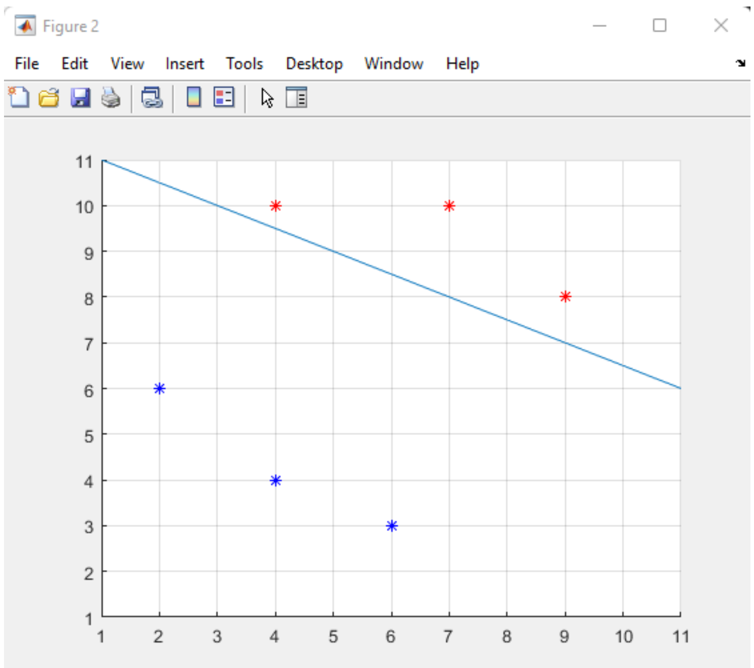

Imagen 1.2. Grafica correspondiente a la línea de separación final

b) Use el perceptrón entrenado en el punto anterior para determinar la clase de pertenencia A o B de los puntos con coordenadas $(5,5)$ y $(6,8)$.

Una vez obtenidos los pesos finales de la línea de separación, podemos evaluar cualquier punto a nuestra elección, en este caso evaluaron los puntos (5,5) y (6,8), los resultados obtenidos son los siguientes:

```
Para el punto [5, 5]: 0
Para el punto [6, 8]: 0
```

Recordemos que la clase A esta compuesta por todos los puntos cuyo valor en la función hardlim es igual a 0 y la clase B está compuesta por todos los puntos cuyo valor en la función hardlim es igual a 1.
De acuerdo con el resultado mostrado podemos decir que ambos puntos pertenecen a la clase A.

## Código MATLAB:

```
%Declaración de los puntos presentes en la imagen%
puntos = [2 6; 4 4; 6 3; 4 10; 7 10; 9 8];

%Declaración de los grupos para cada punto%
t = [0; 0; 0; 1; 1; 1];

%Declaración de bias para cada punto%
bias = ones(6,1)*(-1);

%Agregamos el bias a cada punto%
puntos = [puntos bias];

%Se declaran los pesos%
w = [-0.7 0.5 -0.4];

%Producto punto entre el vector de puntos y los pesos%
a = puntos * w.';

%Se define el factor de aprendizaje%
alpha = 0.1;

%Definimos el número de epocas%
epochs = 50;

[numRows,numCols] = size(puntos);

%Relizamos la operación hardlim%
y = hardlim(a);

figure(1)
grid on;
hold on;
xlim([1 11])
ylim([1 11])
%Graficamos la frontera de decisión con los pesos originales%
x = 0:1:12;
front = w(3)/w(2) - x*w(1)/w(2);
plot(x,front);

%Graficamos los puntos%
for i = 1:numRows
    if t(i) == 0
        plot(puntos(i,1),puntos(i,2),'b*');
    else
        plot(puntos(i,1),puntos(i,2),'r*');
    end
end

%Iniciamos el algoritmo de aprendizaje%
for j = 1:epochs
```

```
    for i = 1:numRows
        punto = [puntos(i,1) puntos(i,2) puntos(i,3)];
        %Realizamos el producto del punto y los pesos%
        a = dot(punto,w);
        %Realizamos la operacion hardlim%
        y(i) = hardlim(a);
        %Calculamos los nuevos pesos%
        wn = w + alpha * (t(i) - y(i)) * punto;
        w = wn;
    end
end

figure(2)
grid on;
hold on;
xlim([1 11])
ylim([1 11])

%Graficamos la frontera de decisión con los pesos finales%
x = 0:1:12;
front = w(3)/w(2) - x*w(1)/w(2);
plot(x,front);

%Graficamos los puntos%
for i = 1:numRows
    if t(i) == 0
        plot(puntos(i,1),puntos(i,2),'b*');
    else
        plot(puntos(i,1),puntos(i,2),'r*');
    end
end

%Imprimimos los pesos finales%
fprintf('\nPesos finales:\n');
disp(w)

%Declaración de los puntos de prueba%
puntos = [5 5;6 8];
bias = ones(2,1)*(-1);

%Agregamos el bias a cada punto%
puntos = [puntos bias];
[numRows,numCols] = size(puntos);

for i = 1:numRows
    punto = [puntos(i,1) puntos(i,2) puntos(i,3)];
    %Realizamos el producto del punto y los pesos%
    a = dot(punto,w);
    %Realizamos la operacion hardlim%
    m = hardlim(a);
    fprintf('Para el punto [%d, %d]: %d\n',punto(1),punto(2),m);
end
```

2. **Considere la misma imagen del ejercicio 1 y haga lo siguiente:**

   a) Entrene una ADALINE con bias mediante la regla DELTA. Proponga una vector de pesos iniciales $W_0 = (w_1, w_2, w_3)^T$ y un parámetro de aprendizaje $\alpha$. Muestre la línea inicial, el vector de pesos final, así como la línea de separación final.

A continuación, se muestra la línea de separación inicial y la línea de separación final, así como los patrones con su respectiva clase. Recordar que la clase A es representada por los puntos azules y la clase B es representada por los puntos de color rojo.

Para este problema se propuso el vector de pesos W = { -0.3, 0.2, 0.1}, mientas que el factor de aprendizaje se definió como 0.01 y el número de épocas se propuso como 1000.

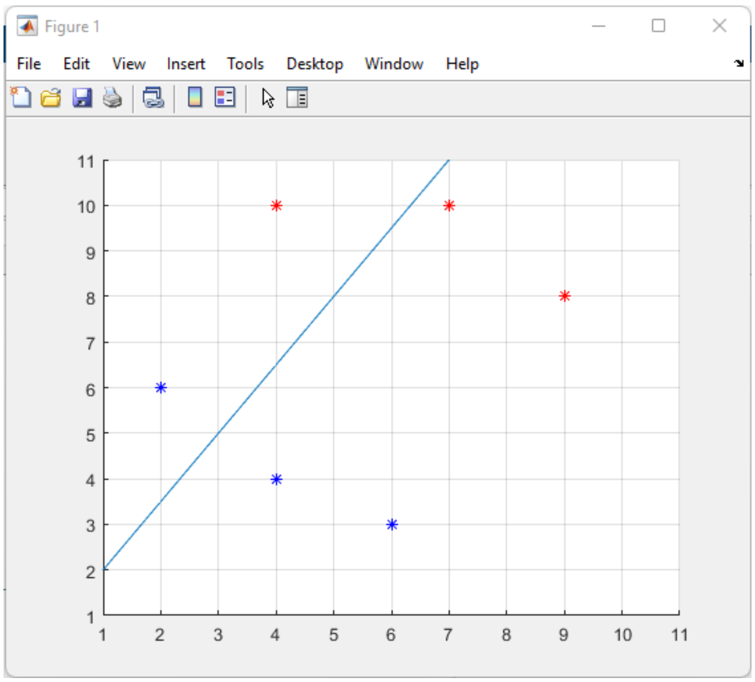


Imagen 2.1. Grafica correspondiente a la línea de separación original.

Una vez obtenidos los pesos finales de la línea de separación, podemos evaluar cualquier punto a nuestra elección, en este caso evaluaron los puntos (5,5) y (6,8), los resultados obtenidos son los siguientes:

```
>> RN2

Pesos finales:
    0.2273    0.2785    3.0409
```

A continuación, se muestra la representación gráfica de la línea de separación final:

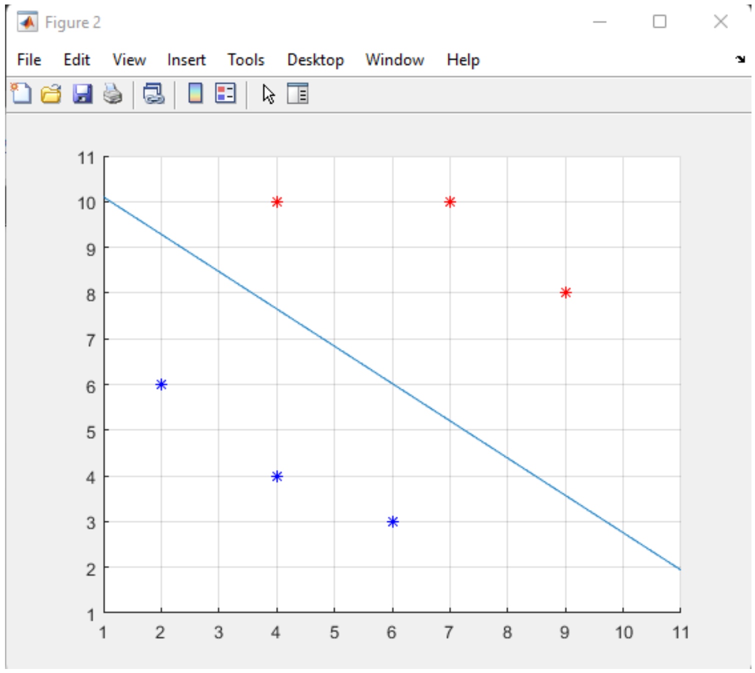


Imagen 2.2. Grafica correspondiente a la línea de separación final.

b) Use la ADALINE entrenada para determinar la clase de pertenencia A o B de los puntos con coordenadas (5,5) y (6,8).

Una vez obtenidos los pesos finales de la línea de separación, podemos evaluar cualquier punto a nuestra elección, en este caso evaluaron los puntos (5,5) y (6,8), los resultados obtenidos son los siguientes:

```
Para el punto [5, 5]: 0
Para el punto [6, 8]: 1
```

Recordemos que la clase A esta compuesta por todos los puntos cuyo valor en la función hardlim es igual a 0 y la clase B está compuesta por todos los puntos cuyo valor en la función hardlim es igual a 1.
De acuerdo con el resultado mostrado podemos decir que el punto (5,5) pertenece a la clase A y el punto (6,8) pertenece a la clase B.

## Código MATLAB:

```
%Declaración de los puntos presentes en la imagen%
puntos = [2 6; 4 4; 6 3; 4 10; 7 10; 9 8];

%Declaración de los grupos para cada punto%
t = [-1; -1; -1; 1; 1; 1];
```

```

%Declaración de bias para cada punto%
bias = ones(6,1)*(-1);

%Agregamos el bias a cada punto%
puntos = [puntos bias];

%Se declaran los pesos%
w = [-0.3 0.2 0.1];

%Producto punto entre el vector de puntos y los pesos%
a = puntos * w.';

%Se define el factor de aprendizaje%
alpha = 0.01;

[numRows,numCols] = size(puntos);

%Relizamos la operación lineal%
y = purelin(a);

%Definimos el número de epocas%
epoch = 0;
epochs = 1000;

figure(1)
grid on;
hold on;
xlim([1 11])
ylim([1 11])
%Graficamos la frontera de decisión con los pesos originales%
x = 0:1:12;
front = w(3)/w(2) - x*w(1)/w(2);
plot(x,front);

%Graficamos los puntos%
for i = 1:numRows
    if t(i) == -1
        plot(puntos(i,1),puntos(i,2),'b*');
    else
        plot(puntos(i,1),puntos(i,2),'r*');
    end
end

%Iniciamos el algoritmo de aprendizaje%
for i = 1:epochs
    epoch = epoch + 1;
    for j = 1:numRows
        punto = [puntos(j,1) puntos(j,2) puntos(j,3)];
        %Realizamos el producto del punto y los pesos%
        a = dot(punto,w);
        %Realizamos la operacion lineal%
```

```
        y(j) = purelin(a);
        %Calculamos los nuevos pesos%
        wn = w + alpha * (t(j) - y(j)) * punto;
        w = wn;
    end
end

grid on;
hold on;

figure(2)
grid on;
hold on;
xlim([1 11])
ylim([1 11])

%Graficamos la frontera de decisión%
x = 1:1:11;
front = w(3)/w(2) - x*w(1)/w(2);
plot(x,front);

%Graficamos los puntos%
for i = 1:numRows
    if t(i) == -1
        plot(puntos(i,1),puntos(i,2),'b*');
    else
        plot(puntos(i,1),puntos(i,2),'r*');
    end
end

%Imprimimos los pesos finales%
fprintf('\nPesos finales:\n');
disp(w)

%Declaración de los puntos de prueba%
puntos = [5 5;6 8];
bias = ones(2,1)*(-1);

%Agregamos el bias a cada punto%
puntos = [puntos bias];
[numRows,numCols] = size(puntos);

for i = 1:numRows
    punto = [puntos(i,1) puntos(i,2) puntos(i,3)];
    %Realizamos el producto del punto y los pesos%
    a = dot(punto,w);
    %Realizamos la operacion hardlim%
    m = hardlim(a);
    fprintf('Para el punto [%d, %d]: %d\n',punto(1),punto(2),m);
end
```

**3. Considere la misma imagen del ejercicio 1 y haga lo siguiente:**

a) Entrene un perceptrón con función de activación sigmoidal y con bias mediante la regla DELTA adaptada. Proponga una vector de pesos iniciales $W_0 = (w_1, w_2, w_3)^T$ y un parámetro de aprendizaje $\alpha$. Muestre la línea inicial, el vector de pesos final, así como la línea de separación final.

A continuación, se muestra la línea de separación inicial y la línea de separación final, así como los patrones con su respectiva clase. Recordar que la clase A es representada por los puntos azules y la clase B es representada por los puntos de color rojo.

Para este problema se propuso el vector de pesos W = {0.5, 0.35, 0.9}, mientas que el factor de aprendizaje se definió como 0.02 y el número de épocas se propuso como 300.

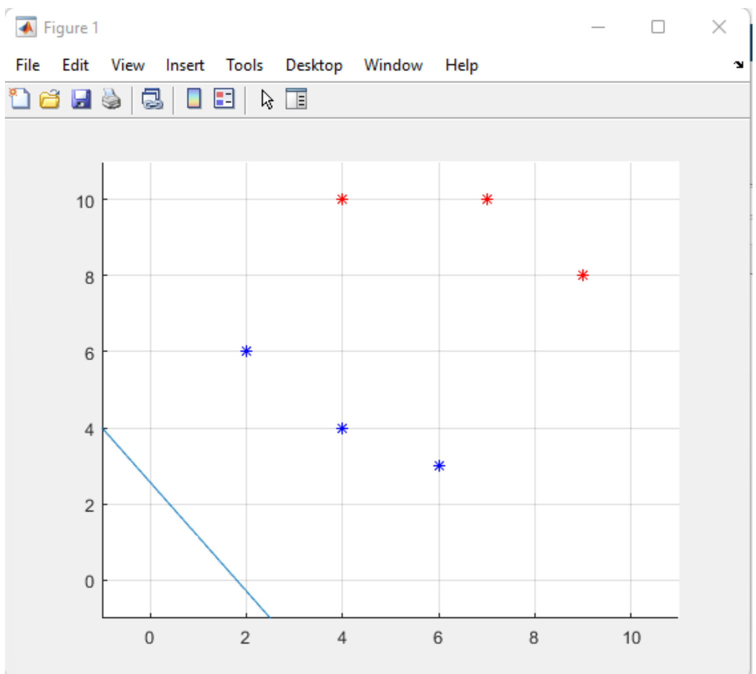


Imagen 3.1. Grafica correspondiente a la línea de separación original.

Una vez obtenidos los pesos finales de la línea de separación, podemos evaluar cualquier punto a nuestra elección, en este caso evaluaron los puntos (5,5) y (6,8), los resultados obtenidos son los siguientes:

```
>> RN3

Pesos finales:
    0.0557    0.3778    3.0470
```

A continuación, se muestra la representación gráfica de la línea de separación final:

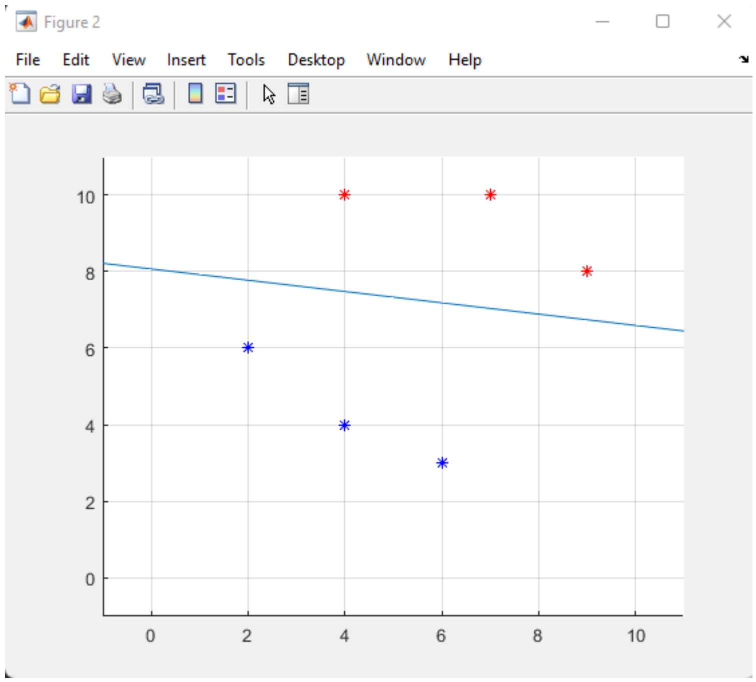

Imagen 3.2. Grafica correspondiente a la línea de separación final.

b) Use el perceptrón entrenado para determinar la clase A o B de los puntos con coordenadas (5,5) y (6,8).

Una vez obtenidos los pesos finales de la línea de separación, podemos evaluar cualquier punto a nuestra elección, en este caso evaluaron los puntos (5,5) y (6,8), los resultados obtenidos son los siguientes:

```
Para el punto [5, 5]: 0
Para el punto [6, 8]: 1
```

Recordemos que la clase A esta compuesta por todos los puntos cuyo valor en la función hardlim es igual a 0 y la clase B está compuesta por todos los puntos cuyo valor en la función hardlim es igual a 1.
De acuerdo con el resultado mostrado podemos decir que el punto (5,5) pertenece a la clase A y el punto (6,8) pertenece a la clase B.

## Código MATLAB:

```
%Declaración de los puntos presentes en la imagen%
puntos = [2 6; 4 4; 6 3; 4 10; 7 10; 9 8];

%Declaración de los grupos para cada punto%
t = [-1; -1; -1; 1; 1; 1];

```

```
%Declaración de bias para cada punto%
bias = ones(6,1)*(-1);

%Agregamos el bias a cada punto%
puntos = [puntos bias];

%Se declaran los pesos%
w = [0.5 0.35 0.9];

%Producto punto entre el vector de puntos y los pesos%
a = puntos * w.';

%Se define el factor de aprendizaje%
alpha = 0.02;

[numRows,numCols] = size(puntos);

%Relizamos la operación sigmoidal%
y = 1/(1+exp(-a));

%Definimos el número de epocas%
epoch = 0;
epochs = 300;

figure(1)
grid on;
hold on;
xlim([-1 11])
ylim([-1 11])
%Graficamos la frontera de decisión con los pesos originales%
x = -1:1:11;
front = w(3)/w(2) - x*w(1)/w(2);
plot(x,front);

%Graficamos los puntos%
for i = 1:numRows
    if t(i) == -1
        plot(puntos(i,1),puntos(i,2),'b*');
    else
        plot(puntos(i,1),puntos(i,2),'r*');
    end
end

%Iniciamos el algoritmo de aprendizaje%
for i = 1:epochs
    epoch = epoch + 1;
    for j = 1:numRows
        punto = [puntos(j,1) puntos(j,2) puntos(j,3)];
        %Realizamos el producto del punto y los pesos%
        a = dot(punto,w);
        %Realizamos la operacion sigmoidal%
        y(j)= 1/(1+exp(-a));
```

```
        %Calculamos los nuevos pesos%
        wn = w - alpha * y(j)*(1-y(j))*(y(j)-t(j)) * punto;
        w = wn;
    end
end

figure(2)
grid on;
hold on;
xlim([-1 11])
ylim([-1 11])

%Graficamos la frontera de decisión%
x = -1:1:11;
front = w(3)/w(2) - x*w(1)/w(2);
plot(x,front);

%Graficamos los puntos%
for i = 1:numRows
    if t(i) == -1
        plot(puntos(i,1),puntos(i,2),'b*');
    else
        plot(puntos(i,1),puntos(i,2),'r*');
    end
end

%Imprimimos los pesos finales%
fprintf('\nPesos finales:\n');
disp(w)

%Declaración de los puntos de prueba%
puntos = [5 5;6 8];
bias = ones(2,1)*(-1);

%Agregamos el bias a cada punto%
puntos = [puntos bias];
[numRows,numCols] = size(puntos);

for i = 1:numRows
    punto = [puntos(i,1) puntos(i,2) puntos(i,3)];
    %Realizamos el producto del punto y los pesos%
    a = dot(punto,w);
    %Realizamos la operacion hardlim%
    m = hardlim(a);
    fprintf('Para el punto [%d, %d]: %d\n',punto(1),punto(2),m);
end
```

**4. Dada la siguiente imagen con dos clases A y B no-linealmente separables:**

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | | | | | | | | | | | | |
| 11 | | | | | | A | | | | | | |
| 10 | | | | | | | | | | | | |
| 9 | | | A | | | | | B | | B | | |
| 8 | | | | | | | | | | | | |
| 7 | | A | | | B | | | | | | | |
| 6 | | | | | | | | B | | B | | |
| 5 | | | | | | | | | | | | |
| 4 | | | | A | | | A | | | | | |
| 3 | | | | | | | | | | | | |
| 2 | | | | | | | | | | | | |
| 1 | | | | | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

a) Entrene una RNA compuesta con una capa intermedia con 2 perceptrones sigmoidales y una neurona en la salida también tipo sigmoidal. Use la regla BP. Proponga un conjunto de pesos para la red y un parámetro de entrenamiento $\alpha$. Muestre el conjunto final de pesos final.

A continuación, se muestran los resultados obtenidos en el algoritmo escrito en Python.

Para este problema se propuso el vector de pesos W = { {-0.9, 0.1}, {0.35, -0.55} }, mientas que el factor de aprendizaje se definió como 0.25 y el número de épocas se propuso como 1500.

```
Pesos finales de la capa intermedia:
[[-3.70762758  2.48101674]
 [ 0.07077736  1.67989381]]

Bias de la capa intermedia:
[-1.73401713 -8.64074627]

Pesos finales de la capa de salida:
[-6.85782517  7.49402573]

Bias de la capa de salida:
[-3.8754946]

       Resultados
Patrón:    t:    Salida:
[4 4]       0       0.065
[2 7]       0       0.029
[3 9]       0       0.037
[ 6 11]       0       0.048
[7 4]       0       0.081
[5 7]       1       0.954
[8 6]       1       0.939
[8 9]       1       0.974
[10  6]       1       0.944
[10  9]       1       0.974
```

b) Use la RNA entrenada para determinar la clase A o B de los puntos con coordenadas $(3{,}7)$ y $(6{,}8)$.

Con los resultados obtenidos anteriormente podemos clasificar los puntos (3,7) y (6,8).

| | Resultados |
|---|---|
| [3 7] | 0.032 |
| [6 8] | 0.970 |

El punto (3,7) se aproxima a 0, entonces pertenece a la clase A, mientras que el punto (6,8) se aproxima a 1, entonces pertenece a la clase B.

c) Muestre que la capa interna de la RNA convierte el problema no lineal a uno lineal al mapear los puntos A y B en el espacio de las salidas de las 2 neuronas intermedias $s_1, s_2$.

Como se aprecia en la siguiente imagen, el problema es linealmente separable.

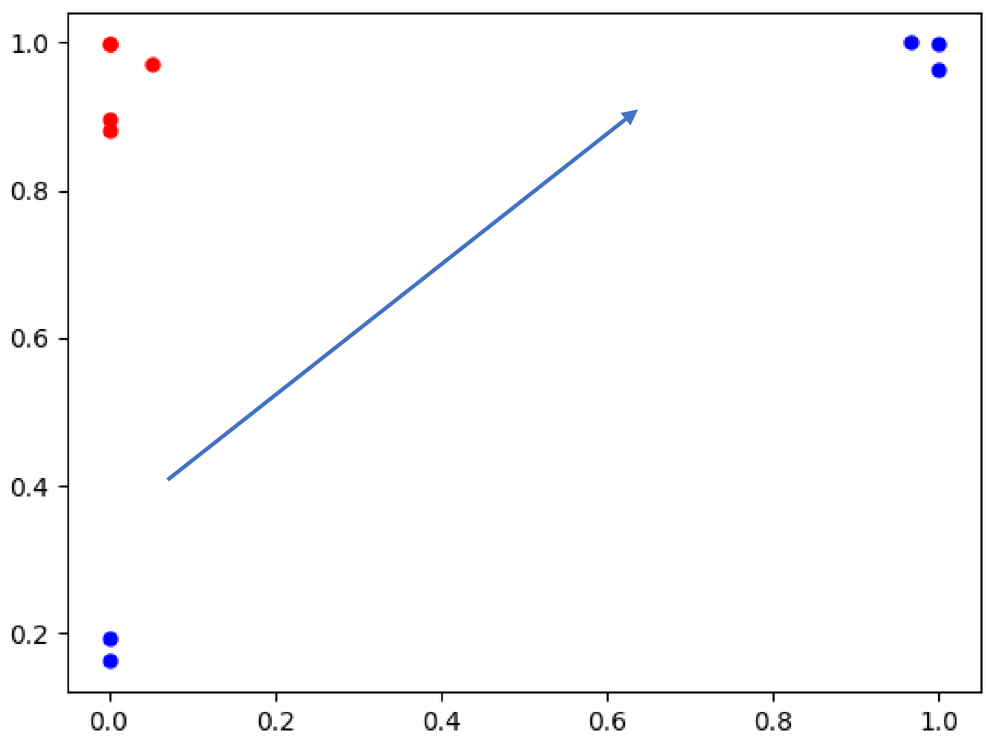


Imagen 4.1. Grafica correspondiente a la representación lineal del problema.

Se diseñó un programa en MATLAB y Python, porque MATLAB no proporciona los resultados adecuados.

### Código MATLAB:

```
%Declaración de los puntos presentes en la imagen%
puntos = [4 4;2 7;3 9;6 11;7 4;5 7;8 6;8 9;10 6;10 9];

%Declaración de los grupos para cada punto%
t = [0;0;0;0;0;1;1;1;1;1];

[numRows,numCols] = size(puntos);

%Definimos el número de epocas%
epochs = 2000;

%Se declaran los pesos y bias%
w1 = [-0.9 0.1;0.35 -0.55];
b1 = [-1 -1];
w2 = [-0.7 0.2];
b2 = [-1];

%Iniciamos el algoritmo de aprendizaje%
for epoch = 1: epochs
    err = 0;
    count = 1;
    for i = 1: numRows
        punto = [puntos(i,1) puntos(i,2)];
        %Hacia adelante%
        [a1, a2] = ff(punto, w1, b1, w2, b2);
        %Calculo de error en la red%
        err = nrror(t(count), a2);
        %Propagación hacia atrás%
        [w1, b1, w2, b2] = bp(punto,t(count), w1, b1, w2, b2, a1, a2);
        count = count + 1;
    end
end

Z = ones(numRows,1);

%Convertir problema lineal a no lineal%
for x = 1: numRows
    punto = [puntos(x,1) puntos(x,2)];
    [X, Y] = ff(punto,w1,b1,w2,b2);
    Z(x) = Y;
end

fprintf('Patrón:    t:    Salida:\n');
for i = 1: numRows
    fprintf('[%d,%d]   %d  %.3f\n',puntos(i,1),puntos(i,2),t(i),Z(i));
end
```

```
fprintf('\nPesos de la capa intermedia:\n');
disp(w1)

fprintf('\nBiases de las capas intermedias:\n');
disp(b1)

fprintf('\nPesos de la capa de salida:\n');
disp(w2)

fprintf('\nBias de la capa de salida:\n');
disp(b2)

%Propagación hacia atrás%
function [nw1, nb1, nw2, nb2] = bp(punto, t, w1, b1, w2, b2, a1, a2)
    %Se define el factor de aprendizaje%
     lpha = 0.25;
    L_error = -(t - a2) * a2 * (1 - a2);
    nw2 = w2 -  lpha * L_error * a1;
    nb2 = b2 -  lpha * L_error;
    l_error = L_error .* w2 .* a1 .* (1 - a1);
    nb1 = b1 -  lpha * l_error;
    aux = [punto(1,1) ; punto(1,2)];
    l_error2 = [l_error(1,1); l_error(1,2)];
    nw1 = w1 -  lpha * aux .* l_error2;
end

%Sigmoidal%
function [s] = sig(z)
    aux = exp(-z);
    s = 1 ./ (1 + aux);
end

%Propagación hacia adelante%
function [a,a2] = ff(punto, w1, b1, w2, b2)
    aux = transpose(w1);
    z2 = transpose(aux*punto.') + b1;
    a = sig(z2);
    z3 = dot(a,w2) + b2;
    a2 = sig(z3);
end

%Calculo de error en la red%
function [err] = nrror(t,a)
    err = 0.5 * (t - a).^2;
end
```

### Código Python:

```
import numpy as np
import matplotlib.pyplot as plt

#Sigmoidal#
def sigmoid(x):
    sig = 1 / (1 + np.exp(-x))
    return(sig)
#Calculo de error en la red#
def nerror(t,a):
    err = 0.5 * np.power(t-a,2)
    return(err)

#Propagación hacia adelante#
def ff(x, w2, b2, w3, b3):
    z2 = np.dot(x,w2.T) + b2
    a2 = sigmoid(z2)
    z3 = np.dot(a2,w3) + b3
    a3 = sigmoid(z3)
    return(a2, a3)

#Propagación hacia atras#
def bp(x, t, w2, b2, w3, b3, a2, a3):
    L_error = -(t - a3) * a3 * (1 - a3)
    nw3 = w3 - alpha * L_error * a2
    nb3 = b3 - alpha * L_error
    l_error = L_error * w3 * a2 * (1 - a2)
    nb2 = b2 - alpha * l_error
    x = np.reshape(x, (1, len(x)))
    l_error = np.reshape(l_error, (len(l_error),1))
    nw2 = w2 - alpha * np.multiply(l_error, x)
    return(nw2, nb2, nw3, nb3)

#Mostramos los resultados obtenidos del algortimo#
def mostrarResultrados(puntos, w2, w3, b2, b3):
  print('\n       Resultados     ')
  print('Patrón:    t:    Salida:')
  count = 0
  #Calculamos la salida#
  for x in puntos:
    _, a3 = ff(x,w2,b2,w3,b3)
    print('{}        {}        {:.3f}'.format(x,t[count],float(a3)))
    count = count + 1

#Definimos los patrones de prueba y las clases#
puntos=np.array([[4,4],[2,7],[3,9],[6,11],[7,4],[5,7],[8,6],[8,9],[10,6],[10,9]])
t = np.array([0,0,0,0,0,1,1,1,1,1])

#Definimos el factor de aprendizaje y el numero de epocas a realizar#
alpha = 0.25
epochs = 1500
```

```

#Definimos pesos y bias para la capa intermedia y de salida#
w2 = np.array([[-0.9, 0.1],[0.35, -0.55]])
b2 = np.array([-1, -1])
w3 = np.array([-0.7, 0.2])
b3 = np.array([-1])

#Comenzamos con el algoritmo de aprendizaje#
err_vector = []
err = 0
for epoch in range(epochs):
    count = 0
    err = 0
    for x in puntos:
        a2, a3 = ff(x,w2,b2,w3,b3)
        err += nerror(t[count],a3)
        w2,b2,w3,b3 = bp(x, t[count], w2, b2, w3, b3, a2, a3)
        count += 1
    err_vector.append(err / puntos.shape[0])

print('Pesos finales de la capa intermedia:')
print(w2)
print('\nBias de la capa intermedia:')
print(b2)
print('\nPesos finales de la capa de salida:')
print(w3)
print('\nBias de la capa de salida:')
print(b3)

mostrarResultrados(puntos, w2, w3, b2, b3)

#Graficamos el problema lineal#
count = 0
plt.figure(1)
for x in puntos:
    z = w2.dot(x) + b2
    a = sigmoid(z)
    if t[count]==0:
      plt.plot(a[0],a[1],'o',color='red', markersize = 5)
    else:
      plt.plot(a[0],a[1],'o',color='blue', markersize = 5)
    count = count +1
plt.show()

#Definimos la clase de los puntos (3,7) y (6,8)#
test = np.array([[3, 7],[6, 8]])
print('       Resultados     ')
for x in test:
    _, a3 = ff(x,w2,b2,w3,b3)
    print('{}       {:.3f}'.format(x,float(a3)))
```