



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 un vector de pesos iniciales $W_0 = (w_1, w_2, w_3)^T$ y un parámetro de aprendizaje α . 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 $W = \{-0.7, 0.5, -0.4\}$, mientras que el factor de aprendizaje se definió como 0.1 y el número de épocas se propuso como 50.

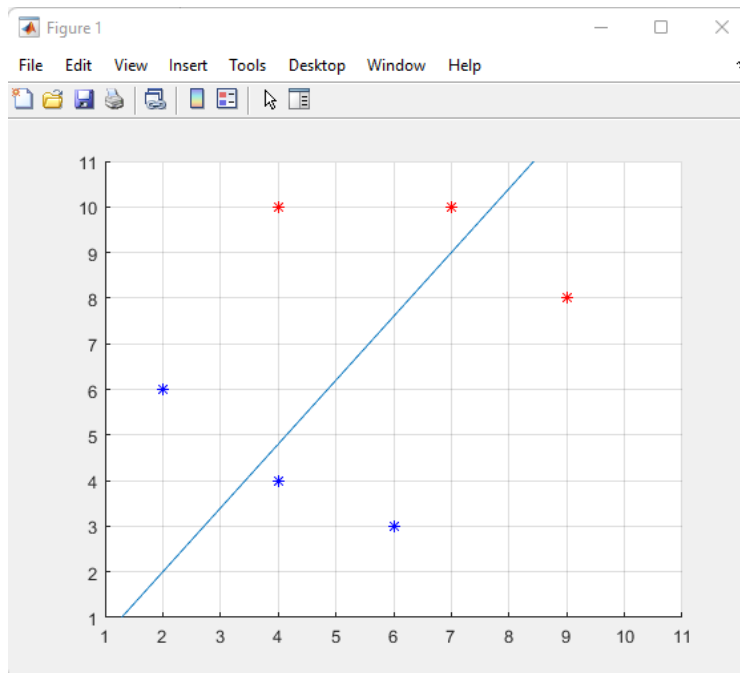


Imagen 1.1. Grafica correspondiente a la línea de separación original.

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:

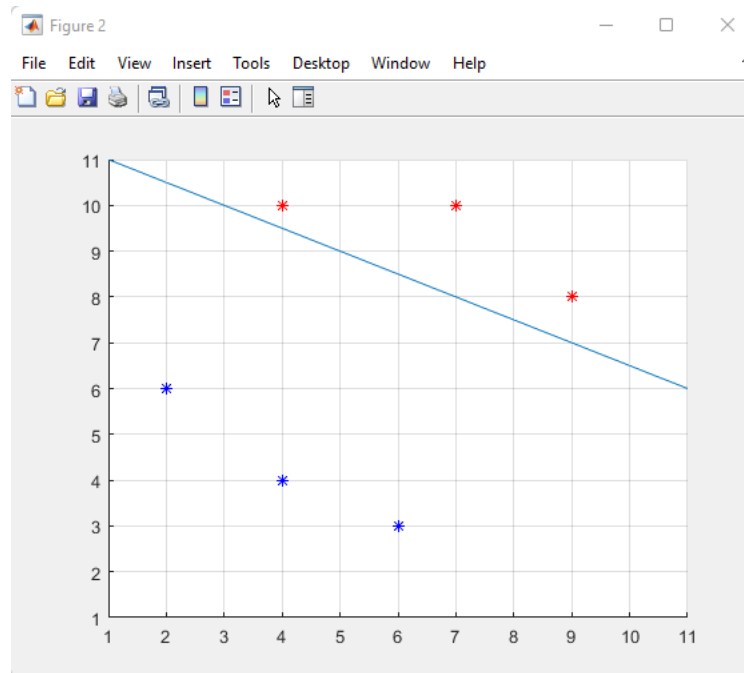


Imagen 1.2. Grafica correspondiente a la línea de separación final

- 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:

```
1 %Declaración de los puntos presentes en la imagen%
2 puntos = [2 6; 4 4; 6 3; 4 10; 7 10; 9 8];
3
4 %Declaración de los grupos para cada punto%
5 t = [0; 0; 0; 1; 1; 1];
6
7 %Declaración de bias para cada punto%
8 bias = ones(6,1)*(-1);
9
10 %Agregamos el bias a cada punto%
11 puntos = [puntos bias];
12
13 %Se declaran los pesos%
14 w = [-0.7 0.5 -0.4];
15
16 %Producto punto entre el vector de puntos y los pesos%
17 a = puntos * w.';
18
19 %Se define el factor de aprendizaje%
20 alpha = 0.1;
21
22 %Definimos el número de épocas%
23 epochs = 50;
24
25 [numRows,numCols] = size(puntos);
26
27 %Relizamos la operación hardlim%
28 y = hardlim(a);
29
30 figure(1)
31 grid on;
32 hold on;
33 xlim([1 11])
34 ylim([1 11])
35 %Graficamos la frontera de decisión con los pesos originales%
36 x = 0:1:12;
37 front = w(3)/w(2) - x*w(1)/w(2);
38 plot(x,front);
39
40 %Graficamos los puntos%
41 for i = 1:numRows
42     if t(i) == 0
43         plot(puntos(i,1),puntos(i,2),'b*');
44     else
45         plot(puntos(i,1),puntos(i,2),'r*');
46     end
47 end
48
49 %Iniciamos el algoritmo de aprendizaje%
50 for j = 1:epochs
```

```

51     for i = 1:numRows
52         punto = [puntos(i,1) puntos(i,2) puntos(i,3)];
53         %Realizamos el producto del punto y los pesos%
54         a = dot(punto,w);
55         %Realizamos la operacion hardlim%
56         y(i) = hardlim(a);
57         %Calculamos los nuevos pesos%
58         wn = w + alpha * (t(i) - y(i)) * punto;
59         w = wn;
60     end
61 end
62
63 figure(2)
64 grid on;
65 hold on;
66 xlim([1 11])
67 ylim([1 11])
68
69 %Graficamos la frontera de decisión con los pesos finales%
70 x = 0:1:12;
71 front = w(3)/w(2) - x*w(1)/w(2);
72 plot(x,front);
73
74 %Graficamos los puntos%
75 for i = 1:numRows
76     if t(i) == 0
77         plot(puntos(i,1),puntos(i,2),'b*');
78     else
79         plot(puntos(i,1),puntos(i,2),'r*');
80     end
81 end
82
83 %Imprimimos los pesos finales%
84 fprintf('\nPesos finales:\n');
85 disp(w)
86
87 %Declaración de los puntos de prueba%
88 puntos = [5 5;6 8];
89 bias = ones(2,1)*(-1);
90
91 %Agregamos el bias a cada punto%
92 puntos = [puntos bias];
93 [numRows,numCols] = size(puntos);
94
95 for i = 1:numRows
96     punto = [puntos(i,1) puntos(i,2) puntos(i,3)];
97     %Realizamos el producto del punto y los pesos%
98     a = dot(punto,w);
99     %Realizamos la operacion hardlim%
100    m = hardlim(a);
101    fprintf('Para el punto [%d, %d]: %d\n',punto(1),punto(2),m);
102 end

```

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

- a) Entrene una ADALINE con bias mediante la regla DELTA. Proponga un vector de pesos iniciales $W_0 = (w_1, w_2, w_3)^T$ y un parámetro de aprendizaje α . 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 $W = \{-0.3, 0.2, 0.1\}$, mientras que el factor de aprendizaje se definió como 0.01 y el número de épocas se propuso como 1000.

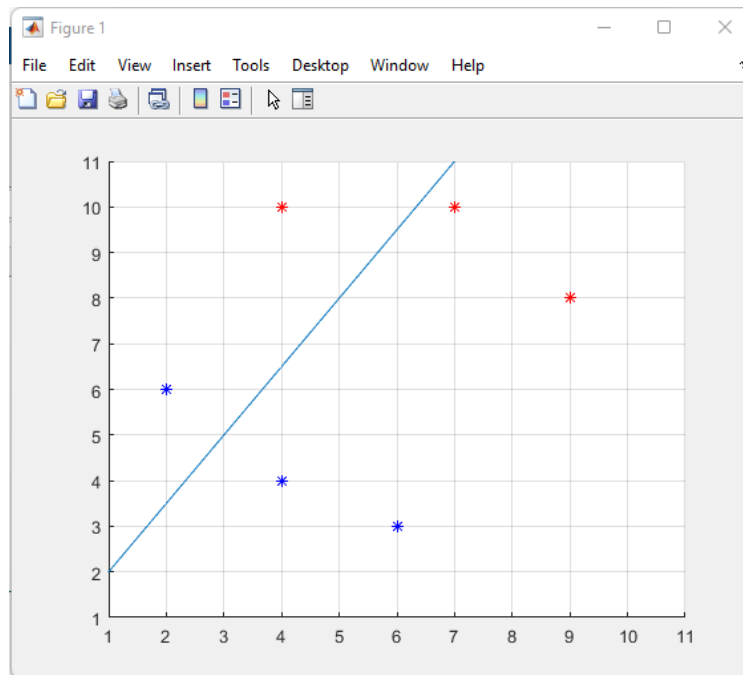


Imagen 2.1. Grafica correspondiente a la línea de separación original.

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:

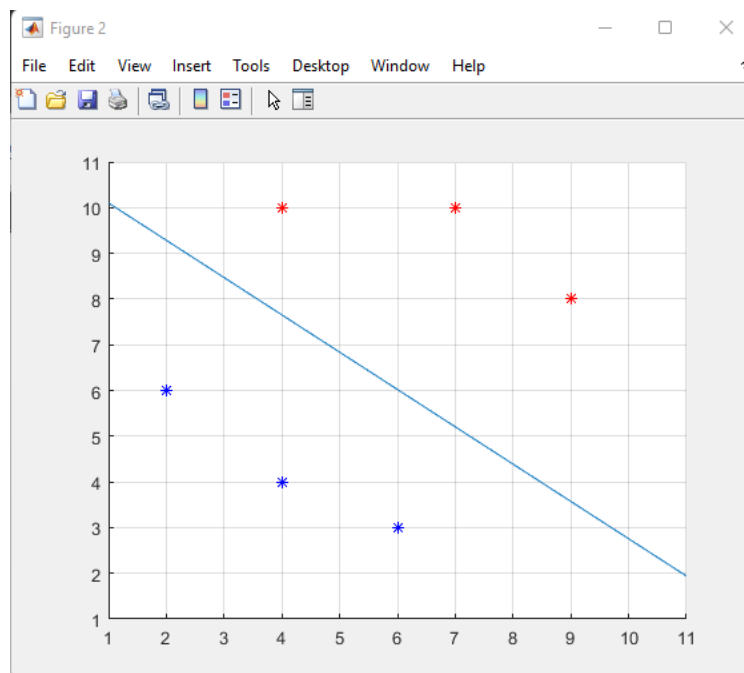


Imagen 2.2. Grafica correspondiente a la línea de separación final.

- 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:

```
1 %Declaración de los puntos presentes en la imagen%
2 puntos = [2 6; 4 4; 6 3; 4 10; 7 10; 9 8];
3
4 %Declaración de los grupos para cada punto%
5 t = [-1; -1; -1; 1; 1; 1];
```

```

6
7 %Declaración de bias para cada punto%
8 bias = ones(6,1)*(-1);
9
10 %Agregamos el bias a cada punto%
11 puntos = [puntos bias];
12
13 %Se declaran los pesos%
14 w = [-0.3 0.2 0.1];
15
16 %Producto punto entre el vector de puntos y los pesos%
17 a = puntos * w.';
18
19 %Se define el factor de aprendizaje%
20 alpha = 0.01;
21
22 [numRows,numCols] = size(puntos);
23
24 %Relizamos la operación lineal%
25 y = purelin(a);
26
27 %Definimos el número de épocas%
28 epoch = 0;
29 epochs = 1000;
30
31 figure(1)
32 grid on;
33 hold on;
34 xlim([1 11])
35 ylim([1 11])
36 %Graficamos la frontera de decisión con los pesos originales%
37 x = 0:1:12;
38 front = w(3)/w(2) - x*w(1)/w(2);
39 plot(x,front);
40
41 %Graficamos los puntos%
42 for i = 1:numRows
43     if t(i) == -1
44         plot(puntos(i,1),puntos(i,2),'b*');
45     else
46         plot(puntos(i,1),puntos(i,2),'r*');
47     end
48 end
49
50 %Iniciamos el algoritmo de aprendizaje%
51 for i = 1:epochs
52     epoch = epoch + 1;
53     for j = 1:numRows
54         punto = [puntos(j,1) puntos(j,2) puntos(j,3)];
55         %Realizamos el producto del punto y los pesos%
56         a = dot(punto,w);
57         %Realizamos la operación lineal%

```



```

58         y(j) = purelin(a);
59         %Calculamos los nuevos pesos%
60         wn = w + alpha * (t(j) - y(j)) * punto;
61         w = wn;
62     end
63 end
64
65 grid on;
66 hold on;
67
68 figure(2)
69 grid on;
70 hold on;
71 xlim([1 11])
72 ylim([1 11])
73
74 %Graficamos la frontera de decisión%
75 x = 1:1:11;
76 front = w(3)/w(2) - x*w(1)/w(2);
77 plot(x,front);
78
79 %Graficamos los puntos%
80 for i = 1:numRows
81     if t(i) == -1
82         plot(puntos(i,1),puntos(i,2),'b*');
83     else
84         plot(puntos(i,1),puntos(i,2),'r*');
85     end
86 end
87
88 %Imprimimos los pesos finales%
89 fprintf('\nPesos finales:\n');
90 disp(w)
91
92 %Declaración de los puntos de prueba%
93 puntos = [5 5;6 8];
94 bias = ones(2,1)*(-1);
95
96 %Agregamos el bias a cada punto%
97 puntos = [puntos bias];
98 [numRows,numCols] = size(puntos);
99
100 for i = 1:numRows
101     punto = [puntos(i,1) puntos(i,2) puntos(i,3)];
102     %Realizamos el producto del punto y los pesos%
103     a = dot(punto,w);
104     %Realizamos la operacion hardlim%
105     m = hardlim(a);
106     fprintf('Para el punto [%d, %d]: %d\n',punto(1),punto(2),m);
107 end

```

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

- a) Entrene un perceptrón con función de activación sigmoideal y con bias mediante la regla DELTA adaptada. Proponga un vector de pesos iniciales $W_0 = (w_1, w_2, w_3)^T$ y un parámetro de aprendizaje α . 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 $W = \{0.5, 0.35, 0.9\}$, mientras que el factor de aprendizaje se definió como 0.02 y el número de épocas se propuso como 300.

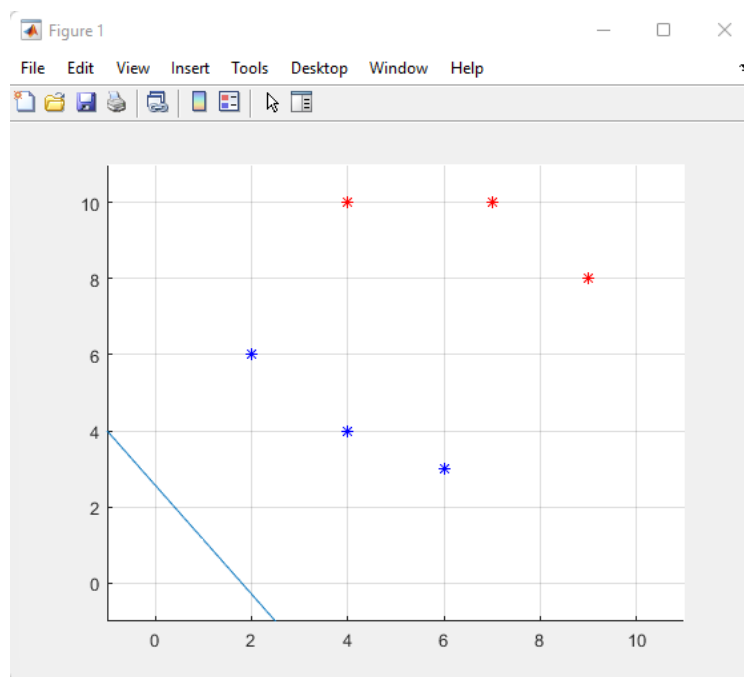


Imagen 3.1. Grafica correspondiente a la línea de separación original.

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:

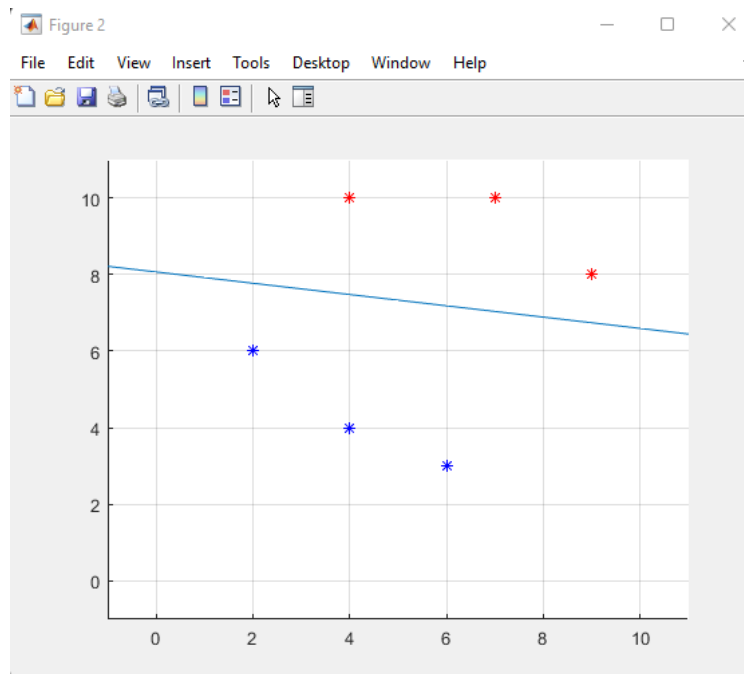


Imagen 3.2. Grafica correspondiente a la línea de separación final.

- 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:

```
1 %Declaración de los puntos presentes en la imagen%
2 puntos = [2 6; 4 4; 6 3; 4 10; 7 10; 9 8];
3
4 %Declaración de los grupos para cada punto%
5 t = [-1; -1; -1; 1; 1; 1];
6
```

```

7 %Declaración de bias para cada punto%
8 bias = ones(6,1)*(-1);
9
10 %Agregamos el bias a cada punto%
11 puntos = [puntos bias];
12
13 %Se declaran los pesos%
14 w = [0.5 0.35 0.9];
15
16 %Producto punto entre el vector de puntos y los pesos%
17 a = puntos * w.';
18
19 %Se define el factor de aprendizaje%
20 alpha = 0.02;
21
22 [numRows,numCols] = size(puntos);
23
24 %Relizamos la operación sigmoidal%
25 y = 1/(1+exp(-a));
26
27 %Definimos el número de épocas%
28 epoch = 0;
29 epochs = 300;
30
31 figure(1)
32 grid on;
33 hold on;
34 xlim([-1 11])
35 ylim([-1 11])
36 %Graficamos la frontera de decisión con los pesos originales%
37 x = -1:1:11;
38 front = w(3)/w(2) - x*w(1)/w(2);
39 plot(x,front);
40
41 %Graficamos los puntos%
42 for i = 1:numRows
43     if t(i) == -1
44         plot(puntos(i,1),puntos(i,2),'b*');
45     else
46         plot(puntos(i,1),puntos(i,2),'r*');
47     end
48 end
49
50 %Iniciamos el algoritmo de aprendizaje%
51 for i = 1:epochs
52     epoch = epoch + 1;
53     for j = 1:numRows
54         punto = [puntos(j,1) puntos(j,2) puntos(j,3)];
55         %Realizamos el producto del punto y los pesos%
56         a = dot(punto,w);
57         %Realizamos la operación sigmoidal%
58         y(j)= 1/(1+exp(-a));

```

```

59         %Calculamos los nuevos pesos%
60         wn = w - alpha * y(j)*(1-y(j))*(y(j)-t(j)) * punto;
61         w = wn;
62     end
63 end
64
65 figure(2)
66 grid on;
67 hold on;
68 xlim([-1 11])
69 ylim([-1 11])
70
71 %Graficamos la frontera de decisión%
72 x = -1:1:11;
73 front = w(3)/w(2) - x*w(1)/w(2);
74 plot(x,front);
75
76 %Graficamos los puntos%
77 for i = 1:numRows
78     if t(i) == -1
79         plot(puntos(i,1),puntos(i,2),'b*');
80     else
81         plot(puntos(i,1),puntos(i,2),'r*');
82     end
83 end
84
85 %Imprimimos los pesos finales%
86 fprintf('\nPesos finales:\n');
87 disp(w)
88
89 %Declaración de los puntos de prueba%
90 puntos = [5 5;6 8];
91 bias = ones(2,1)*(-1);
92
93 %Agregamos el bias a cada punto%
94 puntos = [puntos bias];
95 [numRows,numCols] = size(puntos);
96
97 for i = 1:numRows
98     punto = [puntos(i,1) puntos(i,2) puntos(i,3)];
99     %Realizamos el producto del punto y los pesos%
100    a = dot(punto,w);
101    %Realizamos la operacion hardlim%
102    m = hardlim(a);
103    fprintf('Para el punto [%d, %d]: %d\n',punto(1),punto(2),m);
104 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 sigmoideal. Use la regla BP. Proponga un conjunto de pesos para la red y un parámetro de entrenamiento α . 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 $W = \{ \{-0.9, 0.1\}, \{0.35, -0.55\} \}$, mientras 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.

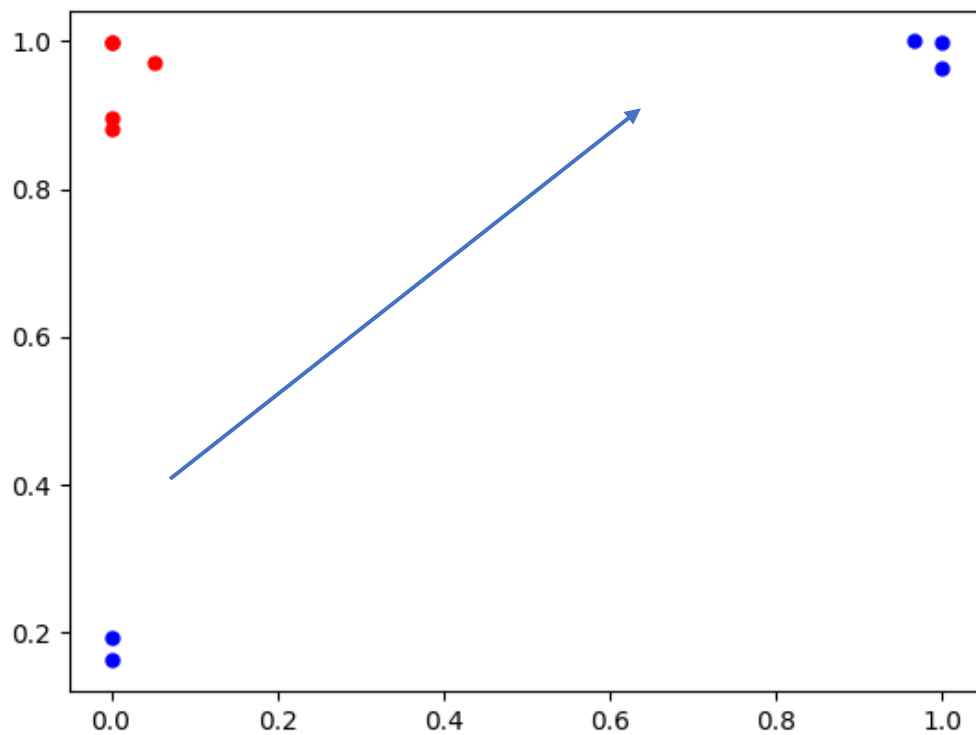


Imagen 4.1. Grafica correspondiente a la representación lineal del problema.

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

Código MATLAB:

```
1  %Declaración de los puntos presentes en la imagen%
2 puntos = [4 4;2 7;3 9;6 11;7 4;5 7;8 6;8 9;10 6;10 9];
3
4  %Declaración de los grupos para cada punto%
5 t = [0;0;0;0;0;1;1;1;1;1];
6
7 [numRows,numCols] = size(puntos);
8
9 %Definimos el número de épocas%
10 epochs = 2000;
11
12 %Se declaran los pesos y bias%
13 w1 = [-0.9 0.1;0.35 -0.55];
14 b1 = [-1 -1];
15 w2 = [-0.7 0.2];
16 b2 = [-1];
17
18 %Iniciamos el algoritmo de aprendizaje%
19 for epoch = 1: epochs
20     err = 0;
21     count = 1;
22     for i = 1: numRows
23         punto = [puntos(i,1) puntos(i,2)];
24         %Hacia adelante%
25         [a1, a2] = ff(punto, w1, b1, w2, b2);
26         %Calculo de error en la red%
27         err = nrror(t(count), a2);
28         %Propagación hacia atrás%
29         [w1, b1, w2, b2] = bp(punto,t(count), w1, b1, w2, b2, a1, a2);
30         count = count + 1;
31     end
32 end
33
34 Z = ones(numRows,1);
35
36 %Convertir problema lineal a no lineal%
37 for x = 1: numRows
38     punto = [puntos(x,1) puntos(x,2)];
39     [X, Y] = ff(punto,w1,b1,w2,b2);
40     Z(x) = Y;
41 end
42
43 fprintf('Patrón:    t:    Salida:\n');
44 for i = 1: numRows
45     fprintf('[%d,%d]  %d  %.3f\n',puntos(i,1),puntos(i,2),t(i),Z(i));
46 end
```



```

47
48 fprintf('\nPesos de la capa intermedia:\n');
49 disp(w1)
50
51 fprintf('\nBiases de las capas intermedias:\n');
52 disp(b1)
53
54 fprintf('\nPesos de la capa de salida:\n');
55 disp(w2)
56
57 fprintf('\nBias de la capa de salida:\n');
58 disp(b2)
59
60 %Propagación hacia atrás%
61 function [nw1, nb1, nw2, nb2] = bp(punto, t, w1, b1, w2, b2, a1, a2)
62     %Se define el factor de aprendizaje%
63     lpha = 0.25;
64     L_error = -(t - a2) * a2 * (1 - a2);
65     nw2 = w2 - lpha * L_error * a1;
66     nb2 = b2 - lpha * L_error;
67     l_error = L_error .* w2 .* a1 .* (1 - a1);
68     nb1 = b1 - lpha * l_error;
69     aux = [punto(1,1) ; punto(1,2)];
70     l_error2 = [l_error(1,1); l_error(1,2)];
71     nw1 = w1 - lpha * aux .* l_error2;
72 end
73
74 %Sigmoidal%
75 function [s] = sig(z)
76     aux = exp(-z);
77     s = 1 ./ (1 + aux);
78 end
79
80 %Propagación hacia adelante%
81 function [a,a2] = ff(punto, w1, b1, w2, b2)
82     aux = transpose(w1);
83     z2 = transpose(aux*punto.') + b1;
84     a = sig(z2);
85     z3 = dot(a,w2) + b2;
86     a2 = sig(z3);
87 end
88
89 %Calculo de error en la red%
90 function [err] = nrror(t,a)
91     err = 0.5 * (t - a).^2;
92 end

```

Código Python:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 #Sigmoidal#
5 def sigmoid(x):
6     sig = 1 / (1 + np.exp(-x))
7     return(sig)
8 #Calculo de error en la red#
9 def nerror(t,a):
10     err = 0.5 * np.power(t-a,2)
11     return(err)
12
13 #Propagación hacia adelante#
14 def ff(x, w2, b2, w3, b3):
15     z2 = np.dot(x,w2.T) + b2
16     a2 = sigmoid(z2)
17     z3 = np.dot(a2,w3) + b3
18     a3 = sigmoid(z3)
19     return(a2, a3)
20
21 #Propagación hacia atras#
22 def bp(x, t, w2, b2, w3, b3, a2, a3):
23     L_error = -(t - a3) * a3 * (1 - a3)
24     nw3 = w3 - alpha * L_error * a2
25     nb3 = b3 - alpha * L_error
26     l_error = L_error * w3 * a2 * (1 - a2)
27     nb2 = b2 - alpha * l_error
28     x = np.reshape(x, (1, len(x)))
29     l_error = np.reshape(l_error, (len(l_error),1))
30     nw2 = w2 - alpha * np.multiply(l_error, x)
31     return(nw2, nb2, nw3, nb3)
32
33 #Mostramos los resultados obtenidos del algortimo#
34 def mostrarResultrados(puntos, w2, w3, b2, b3):
35     print('\n      Resultados      ')
36     print('Patrón:      t:      Salida:')
37     count = 0
38     #Calculamos la salida#
39     for x in puntos:
40         _, a3 = ff(x,w2,b2,w3,b3)
41         print('{}          {}          {:.3f}'.format(x,t[count],float(a3)))
42         count = count + 1
43
44 #Definimos los patrones de prueba y las clases#
45 puntos=np.array([[4,4],[2,7],[3,9],[6,11],[7,4],[5,7],[8,6],[8,9],[10,6],[10,9]])
46 t = np.array([0,0,0,0,0,1,1,1,1,1])
47
48 #Definimos el factor de aprendizaje y el numero de epocas a realizar#
49 alpha = 0.25
50 epochs = 1500
```

```

51
52 #Definimos pesos y bias para la capa intermedia y de salida#
53 w2 = np.array([[-0.9, 0.1],[0.35, -0.55]])
54 b2 = np.array([-1, -1])
55 w3 = np.array([-0.7, 0.2])
56 b3 = np.array([-1])
57
58 #Comenzamos con el algoritmo de aprendizaje#
59 err_vector = []
60 err = 0
61 for epoch in range(epochs):
62     count = 0
63     err = 0
64     for x in puntos:
65         a2, a3 = ff(x,w2,b2,w3,b3)
66         err += nerror(t[count],a3)
67         w2,b2,w3,b3 = bp(x, t[count], w2, b2, w3, b3, a2, a3)
68         count += 1
69     err_vector.append(err / puntos.shape[0])
70
71 print('Pesos finales de la capa intermedia:')
72 print(w2)
73 print('\nBias de la capa intermedia:')
74 print(b2)
75 print('\nPesos finales de la capa de salida:')
76 print(w3)
77 print('\nBias de la capa de salida:')
78 print(b3)
79
80 mostrarResultados(puntos, w2, w3, b2, b3)
81
82 #Graficamos el problema lineal#
83 count = 0
84 plt.figure(1)
85 for x in puntos:
86     z = w2.dot(x) + b2
87     a = sigmoid(z)
88     if t[count]==0:
89         plt.plot(a[0],a[1],'o',color='red', markersize = 5)
90     else:
91         plt.plot(a[0],a[1],'o',color='blue', markersize = 5)
92     count = count +1
93 plt.show()
94
95 #Definimos la clase de los puntos (3,7) y (6,8)#
96 test = np.array([[3, 7],[6, 8]])
97 print('      Resultados      ')
98 for x in test:
99     _, a3 = ff(x,w2,b2,w3,b3)
100     print('{}          {:.3f}'.format(x,float(a3)))

```