



INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

NEURAL NETWORKS

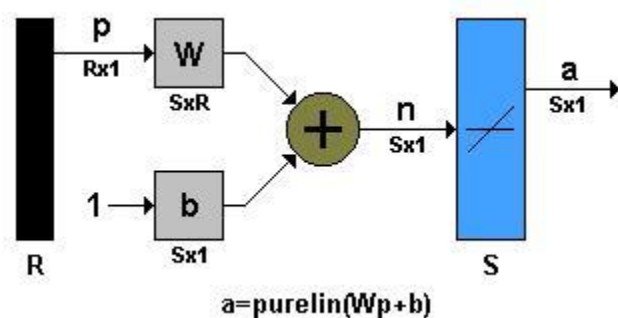
PROF.: MARCO ANTONIO MORENO ARMENDÁRIZ

ALUMNO: ORTEGA VICTORIANO IVAN

No. DE LISTA: 29

GRUPO: 3CM2

RED ADALINE



Índice

INTRODUCCIÓN.....	3
MARCO TEÓRICO.....	4
RESULTADOS EXPERIMENTALES	6
EXPERIMENTO 1: CODIFICADOR DE 6 BITS.....	6
EXPERIMENTO 2: CODIFICADOR DE 8 BITS	10
EXPERIMENTO 3: DOS CLASES (REGLA DE APRENDIZAJE).....	14
EXPERIMENTO 4: CUATRO CLASES (REGLA DE APRENDIZAJE)	18
DISCUSIÓN DE LOS RESULTADOS.....	22
CONCLUSIONES	23
Referencias	24
ANEXO	25
Adaline.m	25
leerPatronesPrototipo.m	35
leerTargets.m	35
imprimirMatricesEnArchivo.m	36

INTRODUCCIÓN

En el caso del perceptrón simple de una sola neurona es posible separar los datos de entrada en dos clases. Para separar más de dos clases con una sola neurona se propuso una red conocida como Adaptive Linear Neuron (ADALINE).

En 1960 Ted Widrow y Bernard Hoff proponen esta RNA con una regla de aprendizaje que toma en cuenta la diferencia entre el valor deseado (target) y el valor de salida de la RNA (a). A dicho valor se le conoce como señal del error.

La red ADALINE es similar al perceptrón, excepto en su función de transferencia de tipo lineal (pureline) en lugar de un limitador fuerte (hardlim o hardlims) como en el perceptrón.

La red ADALINE presenta la misma limitación del perceptrón, ambas pueden resolver problemas linealmente separables. Sin embargo, el algoritmo de aprendizaje de la red ADALINE, conocido como LMS (Least Mean Square o Error cuadrático mínimo), es más potente que la regla de aprendizaje del perceptrón. (Pereira, s.f.)

ADALINE es adaptivo en el sentido de que existe un procedimiento bien definido para modificar los pesos de la matriz de pesos W , con el objetivo de hacer que se proporcione un valor de salida correcto para la entrada dada. Es lineal por que la salida es una función lineal sencilla de los valores de entrada. (Pereira, s.f.)

MARCO TEÓRICO

La estructura general de la red tipo Adaline puede visualizarse en la Fig. 1: (Gutiérrez H. F., Redes Neuronales Multicapa con Aprendizaje Supervisado, s.f.)

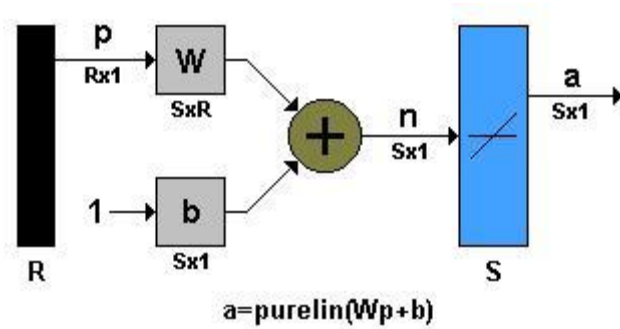


Figura 1: Estructura de una Red ADALINE

En donde:

p : Patrones de entrada

b : Umbrales de activación

a : Salida de la neurona

La salida de la red está dada por:

$$a = \text{pureline}(Wp^T + b) = Wp^T + b$$

En similitud con el Perceptrón, el límite de la característica de decisión para la red Adaline se presenta cuando $n=0$, por lo tanto:

$$Wp^T + b = 0$$

En la siguiente figura, se especifica la línea que separa en dos regiones el espacio de entrada, como se muestra en la siguiente figura:

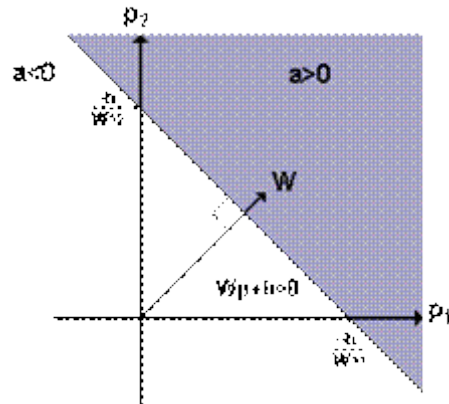


Figura 2: Característica de Decisión de una Red Tipo Adaline

La salida de la neurona es mayor que cero en el área gris, en el área blanca la salida es menor que cero. Como se mencionó anteriormente, la red Adaline puede clasificar correctamente patrones linealmente separables en dos categorías. (Gutiérrez H. F., Redes Neuronales Multicapa con Aprendizaje Supervisado, s.f.)

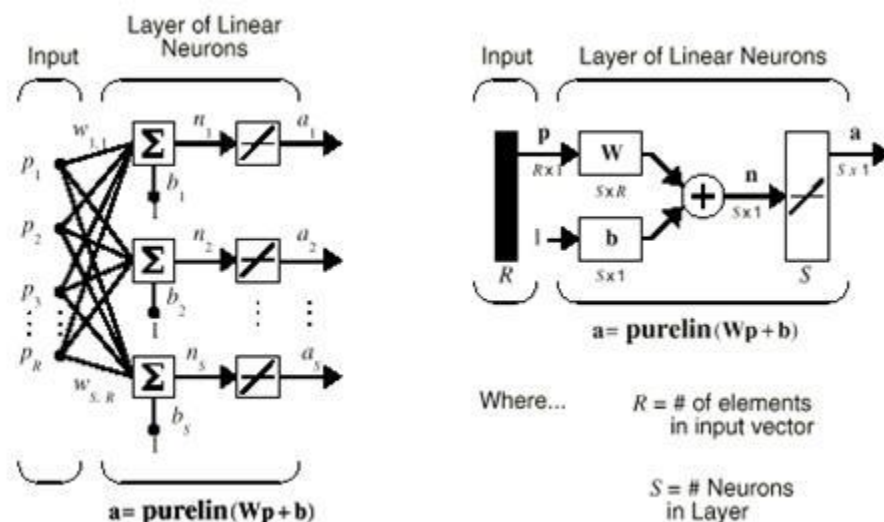


Figura 3. Arquitectura de una red ADALINE

La arquitectura neuronal mostrada en la Figura 3 tiene una capa de s neuronas conectadas a R entradas a través de una matriz de pesos W . (Gutiérrez H. F., Redes Neuronales Multicapa con Aprendizaje Supervisado, s.f.)

Esta red es con frecuencia llamada MADALINE o Múltiples ADALINE. A la derecha de la Figura 3 define un vector de salida a de longitud S .

La regla de Widrow – Hoff puede entrenar solamente una capa de redes lineales. Esto no es tanto una desventaja, ya que una red de una sola capa es tan capaz como una red de múltiples capas. Para cada red lineal multicapa, existe una red lineal de una sola capa.

RESULTADOS EXPERIMENTALES

EXPERIMENTO 1: CODIFICADOR DE 6 BITS

Para este experimento, se utilizó una red Adaline sin bias, en el programa de Matlab, únicamente hay que indicar que se usará la red sin bias para resolver el problema del codificador binario a decimal como se muestra en la figura 4:

```
Command Window

¿La red usará bias? (s/n): n
BCD
fx Ingresar el tamaño del codificador:
```

Figura 4. Se indica al programa que se usará una red sin bias.

Posteriormente se solicita el tamaño del codificador y al ingresarlo se muestra el conjunto de entrenamiento generado de acuerdo a el tamaño ingresado como se muestra en la figura 5:

```
Ingresar el tamaño del codificador: 6
El conjunto de entrenamiento es:
```

0	0	0	0	0	0	0
0	0	0	0	0	1	1
0	0	0	0	1	0	2
0	0	0	0	1	1	3
0	0	0	1	0	0	4
0	0	0	1	0	1	5
0	0	0	1	1	0	6
0	0	0	1	1	1	7
0	0	1	0	0	0	8
0	0	1	0	0	1	9
0	0	1	0	1	0	10
0	0	1	0	1	1	11
0	0	1	1	0	0	12
0	0	1	1	0	1	13
0	0	1	1	1	0	14
0	0	1	1	1	1	15
0	1	0	0	0	0	16
0	1	0	0	0	1	17
0	1	0	0	1	0	18
0	1	0	0	1	1	19
0	1	0	1	0	0	20
0	1	0	1	0	1	21

Figura 5 (parte 1). Se solicitan el tamaño del codificador y se muestra en pantalla el conjunto de entrenamiento.

0	1	0	1	1	0	22
0	1	0	1	1	1	23
0	1	1	0	0	0	24
0	1	1	0	0	1	25
0	1	1	0	1	0	26
0	1	1	0	1	1	27
0	1	1	1	0	0	28
0	1	1	1	0	1	29
0	1	1	1	1	0	30
0	1	1	1	1	1	31
1	0	0	0	0	0	32
1	0	0	0	0	1	33
1	0	0	0	1	0	34
1	0	0	0	1	1	35
1	0	0	1	0	0	36
1	0	0	1	0	1	37
1	0	0	1	1	0	38
1	0	0	1	1	1	39
1	0	1	0	0	0	40
1	0	1	0	0	1	41
1	0	1	0	1	0	42
1	0	1	0	1	1	43
1	0	1	1	0	0	44
1	0	1	1	0	1	45
1	0	1	1	1	0	46
1	0	1	1	1	1	47
1	1	0	0	0	0	48
1	1	0	0	0	1	49
1	1	0	0	1	0	50
1	1	0	0	1	1	51
1	1	0	1	0	0	52
1	1	0	1	0	1	53
1	1	0	1	1	0	54
1	1	0	1	1	1	55
1	1	1	0	0	0	56
1	1	1	0	0	1	57
1	1	1	0	1	0	58
1	1	1	0	1	1	59
1	1	1	1	0	0	60
1	1	1	1	0	1	61
1	1	1	1	1	0	62
1	1	1	1	1	1	63

Figura 5(parte 2).

A continuación, se muestra la matriz de pesos inicial y se le solicitarán al usuario los valores de itmax, Eit y alfa (factor de aprendizaje), para este ejemplo se usarán itmax=100, Eit=0 y alfa=0.5 como se muestra en la figura 6:

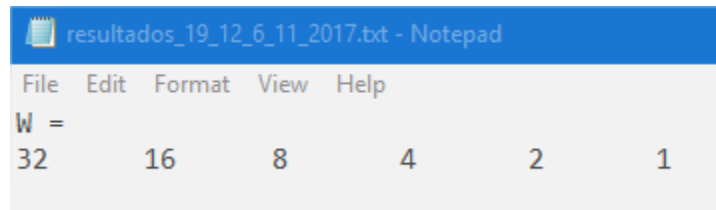
```
Se usará la siguiente matriz inicial de pesos.  
W =  
    0    0    0    0    0    0  
  
Itmax = 100  
Eit = 0  
alfa = 0.5
```

Figura 6. Matriz de pesos inicial y solicitud de valores itmax, Eit y alfa.

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 final (figura 7), la cual se mandará a un archivo de resultados (figura 8). Además, se mostrarán 2 graficas, una de la evolución de los pesos de la matriz de pesos (figura 9) y la señal del error (figura 10).

```
Se ha tenido un aprendizaje exitoso en la iteración 2.  
W =  
    32    16     8     4     2     1
```

Figura 7. Resultado de la red.



The image shows a Notepad window titled "resultados_19_12_6_11_2017.txt - Notepad". The window contains the following text:

```
File Edit Format View Help  
W =  
32    16     8     4     2     1
```

Figura 8. Resultado de la red en archivo.

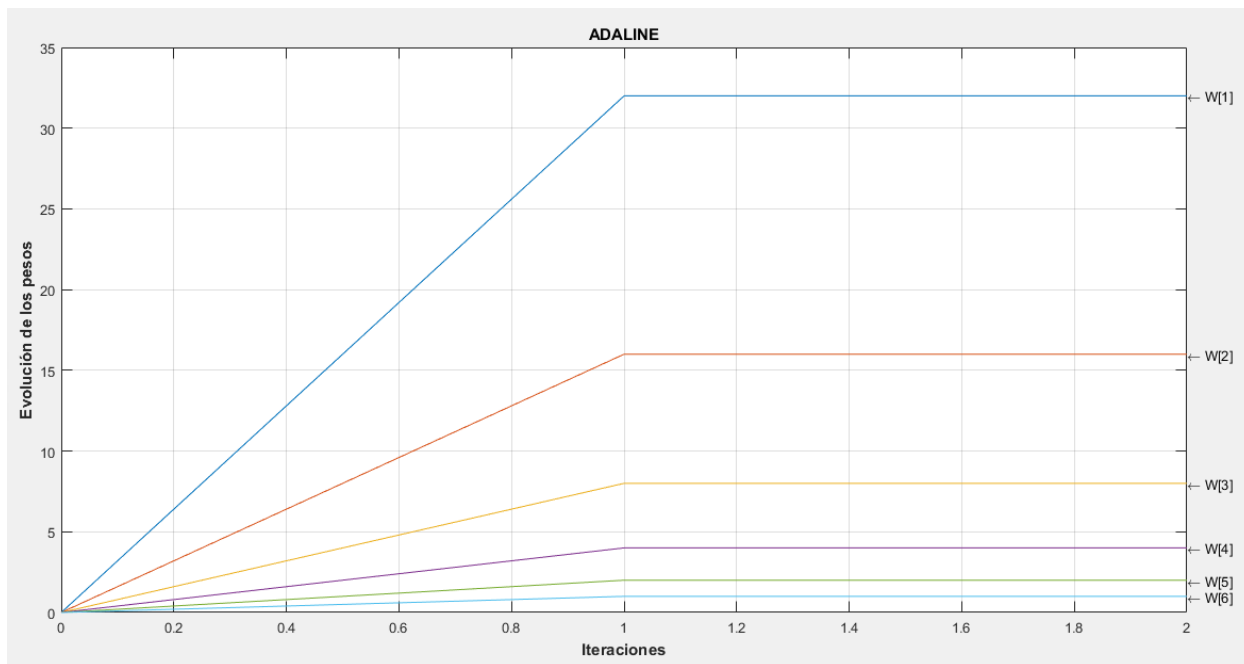


Figura 9. Evolución de los pesos de la matriz de pesos W .

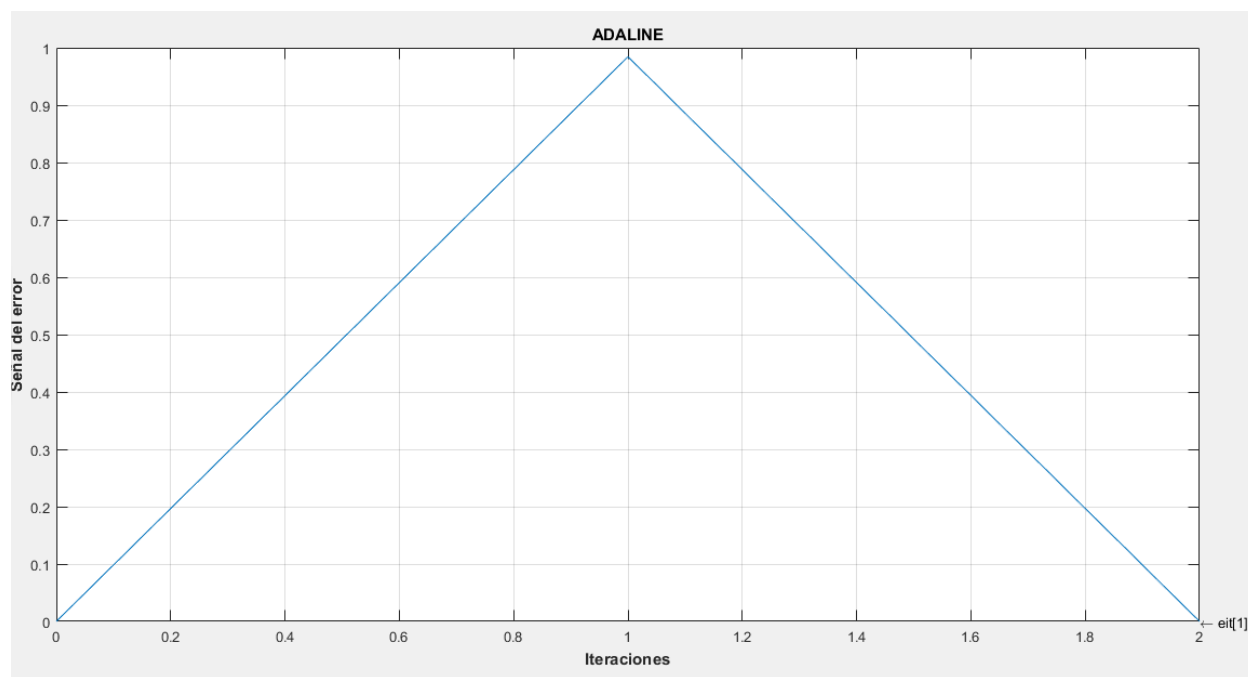


Figura 10. Señal del error.

EXPERIMENTO 2: CODIFICADOR DE 8 BITS

Para este experimento, se utilizó una red Adaline sin bias, en el programa de Matlab, únicamente hay que indicar que se usará la red sin bias para resolver el problema del codificador binario a decimal como se muestra en la figura 11:

```
Command Window

¿La red usará bias? (s/n): n
BCD
fx Ingresa el tamaño del codificador:
```

Figura 11. Se indica al programa que se usará una red sin bias.

Posteriormente se solicita el tamaño del codificador y al ingresarlo se muestra el conjunto de entrenamiento generado de acuerdo a el tamaño ingresado como se muestra en la figura 12:

```
Ingresa el tamaño del codificador: 9
El conjunto de entrnamiento es:
```

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	1	0	2
0	0	0	0	0	0	0	1	1	3
0	0	0	0	0	0	1	0	0	4
0	0	0	0	0	0	1	0	1	5
0	0	0	0	0	0	1	1	0	6
0	0	0	0	0	0	1	1	1	7
0	0	0	0	0	1	0	0	0	8
0	0	0	0	0	1	0	0	1	9
0	0	0	0	0	1	0	1	0	10
0	0	0	0	0	1	0	1	1	11
0	0	0	0	0	1	1	0	0	12
0	0	0	0	0	1	1	0	1	13
0	0	0	0	0	1	1	1	0	14
0	0	0	0	0	1	1	1	1	15
0	0	0	0	1	0	0	0	0	16
0	0	0	0	1	0	0	0	1	17

Figura 12 (parte 1). Se solicitan el tamaño del codificador y se muestra en pantalla el conjunto de entrenamiento.

Command Window									
1	1	1	1	0	1	0	1	1	491
1	1	1	1	0	1	1	0	0	492
1	1	1	1	0	1	1	0	1	493
1	1	1	1	0	1	1	1	0	494
1	1	1	1	0	1	1	1	1	495
1	1	1	1	1	0	0	0	0	496
1	1	1	1	1	0	0	0	1	497
1	1	1	1	1	0	0	1	0	498
1	1	1	1	1	0	0	1	1	499
1	1	1	1	1	0	1	0	0	500
1	1	1	1	1	0	1	0	1	501
1	1	1	1	1	0	1	1	0	502
1	1	1	1	1	0	1	1	1	503
1	1	1	1	1	1	0	0	0	504
1	1	1	1	1	1	0	0	1	505
1	1	1	1	1	1	0	1	0	506
1	1	1	1	1	1	0	1	1	507
1	1	1	1	1	1	1	0	0	508
1	1	1	1	1	1	1	0	1	509
1	1	1	1	1	1	1	1	0	510
1	1	1	1	1	1	1	1	1	511

Figura 12 (parte 2). Debido a que el conjunto de entrenamiento es muy grande, sólo se muestran la parte inicial y final.

A continuación, se muestra la matriz de pesos inicial y se le solicitarán al usuario los valores de itmax, Eit y alfa (factor de aprendizaje), para este ejemplo se usarán itmax=100, Eit=0 y alfa=0.5 como se muestra en la figura 13:

```
Se usará la siguiente matriz inicial de pesos.
W =
    0    0    0    0    0    0    0    0    0

Itmax = 100
Eit = 0
alfa = 0.5
```

Figura 13. Matriz de pesos inicial y solicitud de valores itmax, Eit y alfa.

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 final (figura 14), la cual se mandará a un archivo de resultados (figura 15). Además, se mostrarán 2 graficas, una de la evolución de los pesos de la matriz de pesos (figura 16) y la señal del error (figura 17).

```
Se ha tenido un aprendizaje exitoso en la iteración 2.  
W =  
256 128 64 32 16 8 4 2 1
```

Figura 14. Resultado de la red.

```
resultados_19_23_6_11_2017.txt - Notepad  
File Edit Format View Help  
W =  
256 128 64 32 16 8 4 2 1
```

Figura 15. Resultado de la red en archivo.

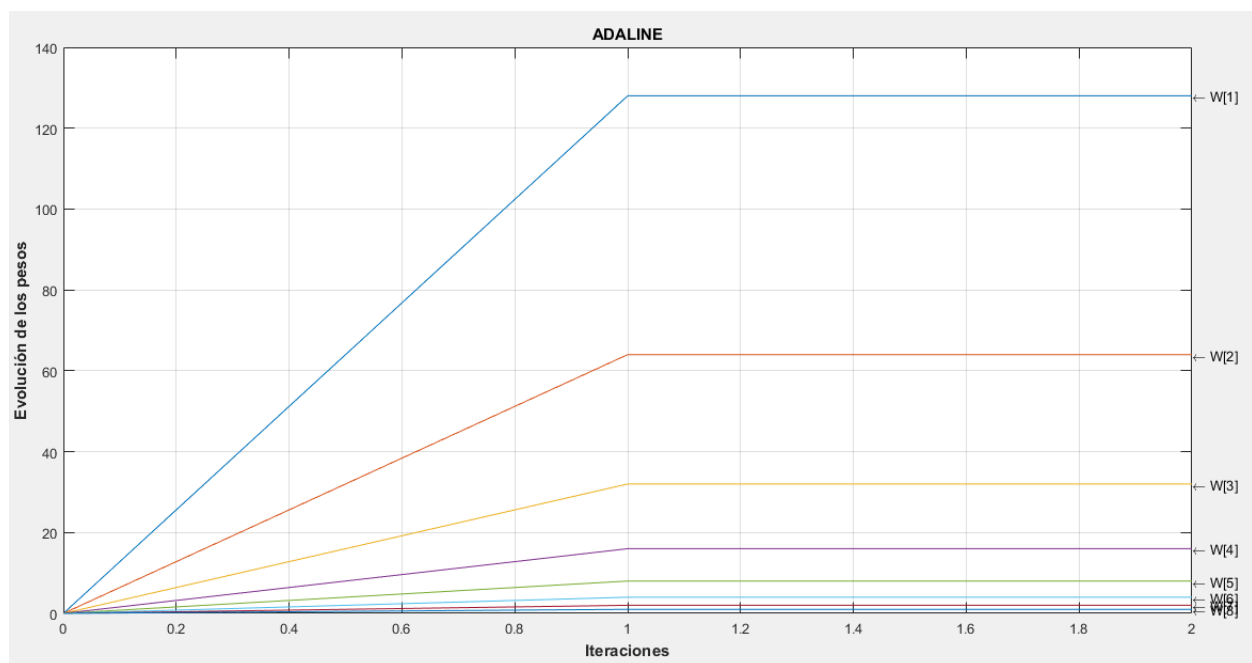


Figura 16. Evolución de los pesos de la matriz de pesos W .

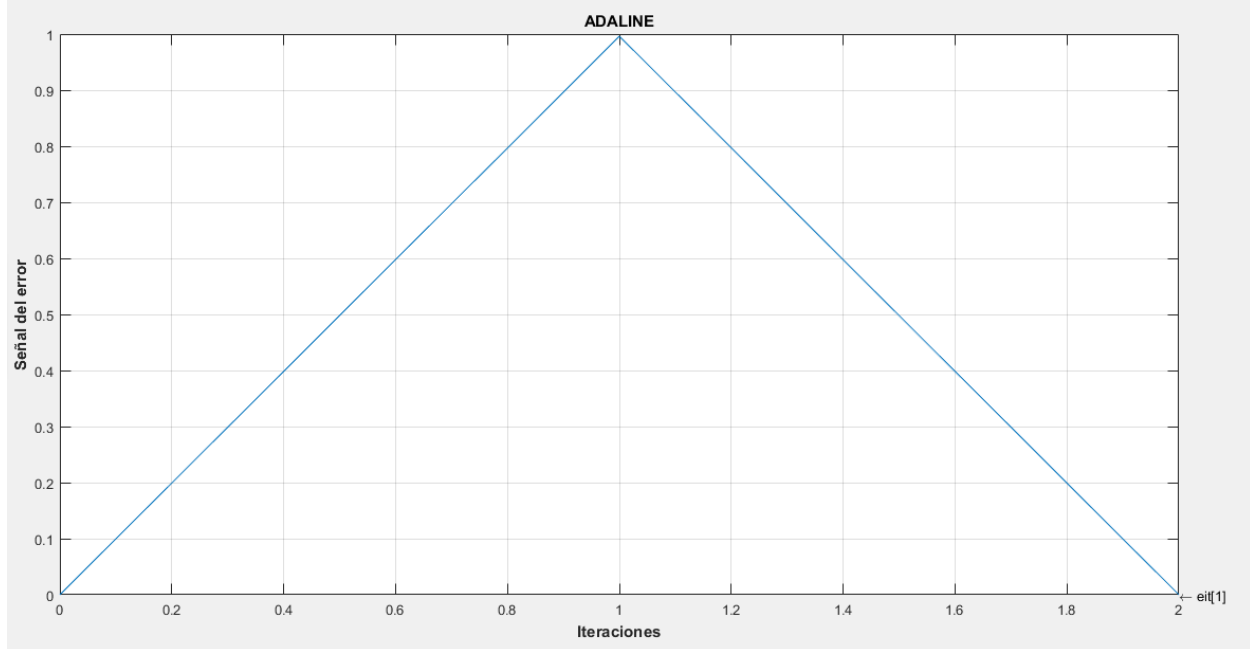


Figura 17. Señal del error.

EXPERIMENTO 3: DOS CLASES (REGLA DE APRENDIZAJE)

Para este experimento se utilizó el siguiente conjunto de entrenamiento que corresponde a el ejercicio propuesto en el libro Neural Network Design (Hagan, Widrow-Hoff Learning Exercises):

3 6 75

6 3 75

-6 3 -75

Le indicamos al programa de Matlab que esta vez se usará una red con bias, 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 3 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 18:

```
¿La red usará bias? (s/n): s
Ingresa el número de patrones prototipo: 3
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
(Sin extensión)
Nombre del Archivo: experimento1
Prototipos =
     3     6
     6     3
    -6     3

Targets =
    75
    75
   -75
```

Figura 18. 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=0.0001 como se muestra en la figura 19:

```
Ingresa el número de clases: 2
Itmax: 100
Eit: 0.0001
```

Figura 19. Se solicitan el número de clases y los valores itmax y Eit.

Luego se nos pide el valor de alfa, antes de ello se intenta hacer un cálculo de un valor adecuado, y se le indica al usuario en pantalla. El cálculo de un alfa adecuado se hace de acuerdo al algoritmo LSM, como se indica a continuación: (Hagan, Analysis of Convergence)

1. Se calcula la matriz R como se muestra en la siguiente ecuación:

$$R = E[pp^T]$$

Donde:

E = La probabilidad con la que aparezcan los patrones prototipo (para la práctica se hizo la suposición de que son equiprobables, es decir, $E = 1/(\text{número de prototipos})$).

p = La matriz de prototipos.

p^T = La matriz de prototipos transpuesta.

2. Posteriormente se obtienen los eigenvalores de la matriz R, y se busca el eigenvalor λ máximo, de tal forma que:

$$0 < \alpha < \frac{1}{\lambda_{\max}}$$

Ya que se nos muestra (si es posible), el valor de alfa, se ingresa el valor del usuario como se muestra en la figura 20:

```
Se recomienda un valor de alfa menor a 0.033333
alfa = 0.02
```

Figura 20. Se muestra un valor aproximado de alfa al usuario, y se le pide que ingrese uno.

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 =
    0.0774   -1.2141

b =
   -1.1135
```

Figura 21. 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 22), los cuales se mandarán a un archivo de resultados (figura 23). Además, se mostrarán 3 graficas, una de la evolución de los pesos de la matriz de pesos (figura 24), la evolución del vector bias (figura 25) y la señal del error (figura 26).

```
Se tuvo un aprendizaje con 3/4 de probabilidad de éxito en la iteración 77.
W =
   12.5000   12.5001

b =
  -37.5000
```

Figura 22. Resultado de la red.

```
resultados_20_1_6_11_2017.txt - Notepad
File Edit Format View Help
W =
12.5  12.5
b =
-37.5
```

Figura 23. Resultado de la red en archivo.

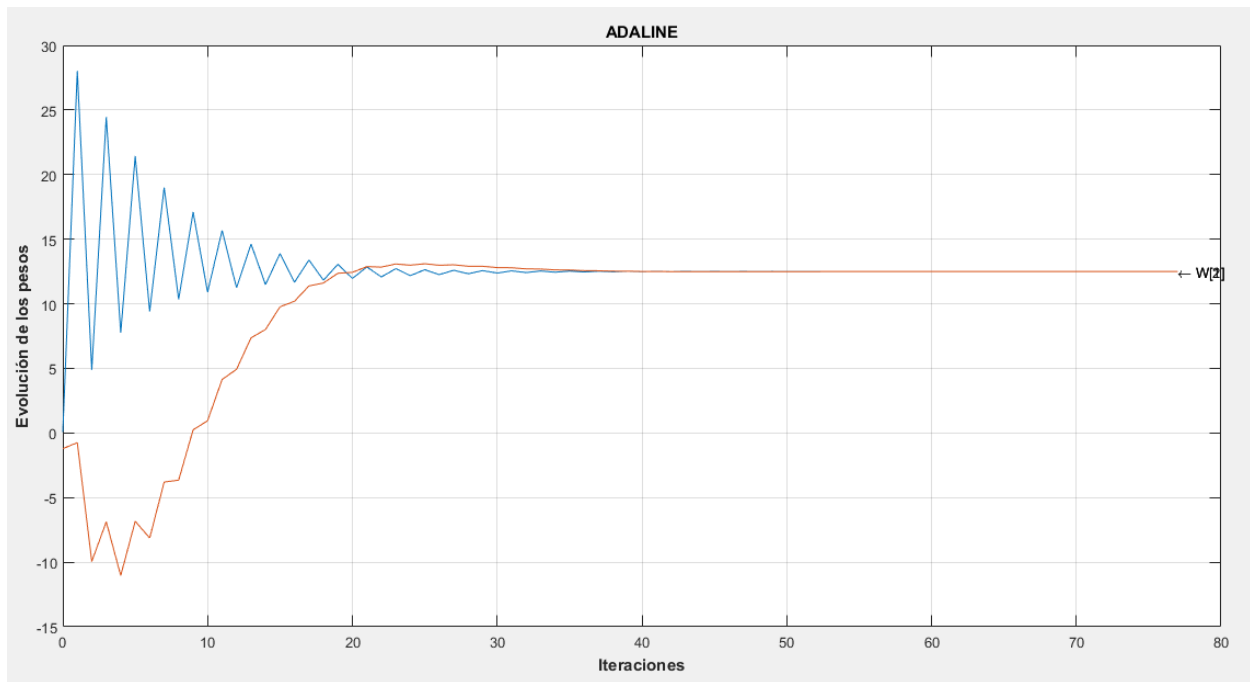


Figura 24. Evolución de los pesos de la matriz de pesos W .

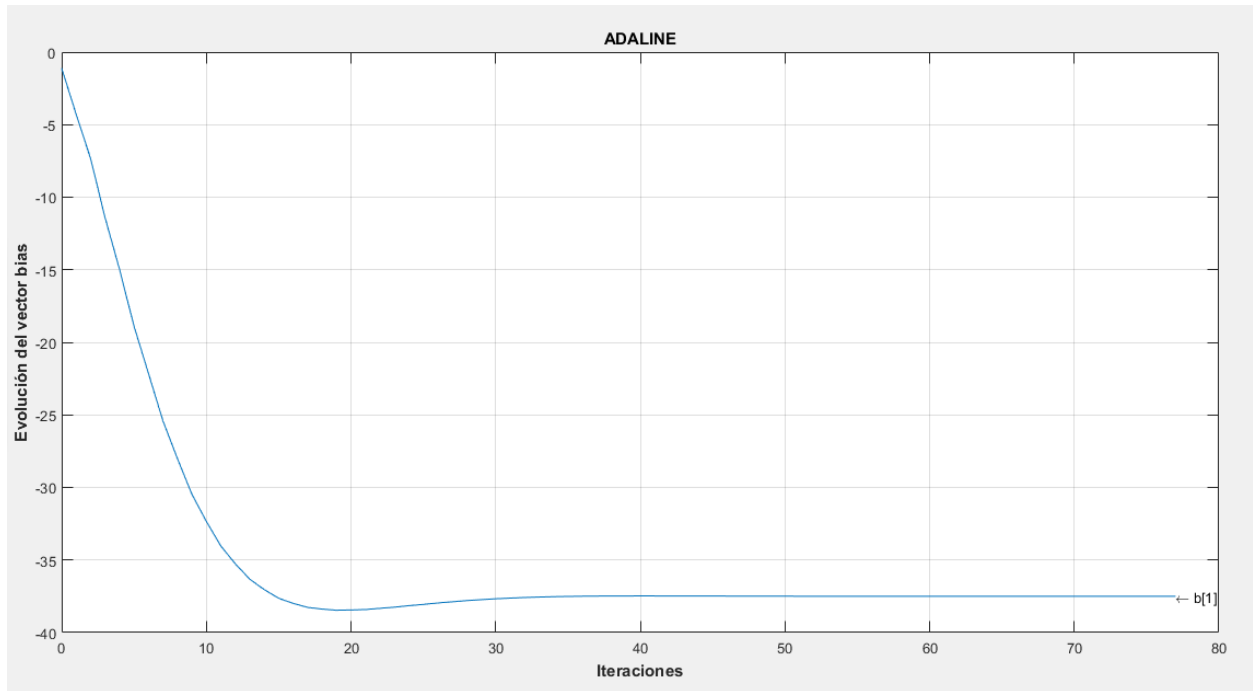


Figura 25. Evolución del vector bias b.

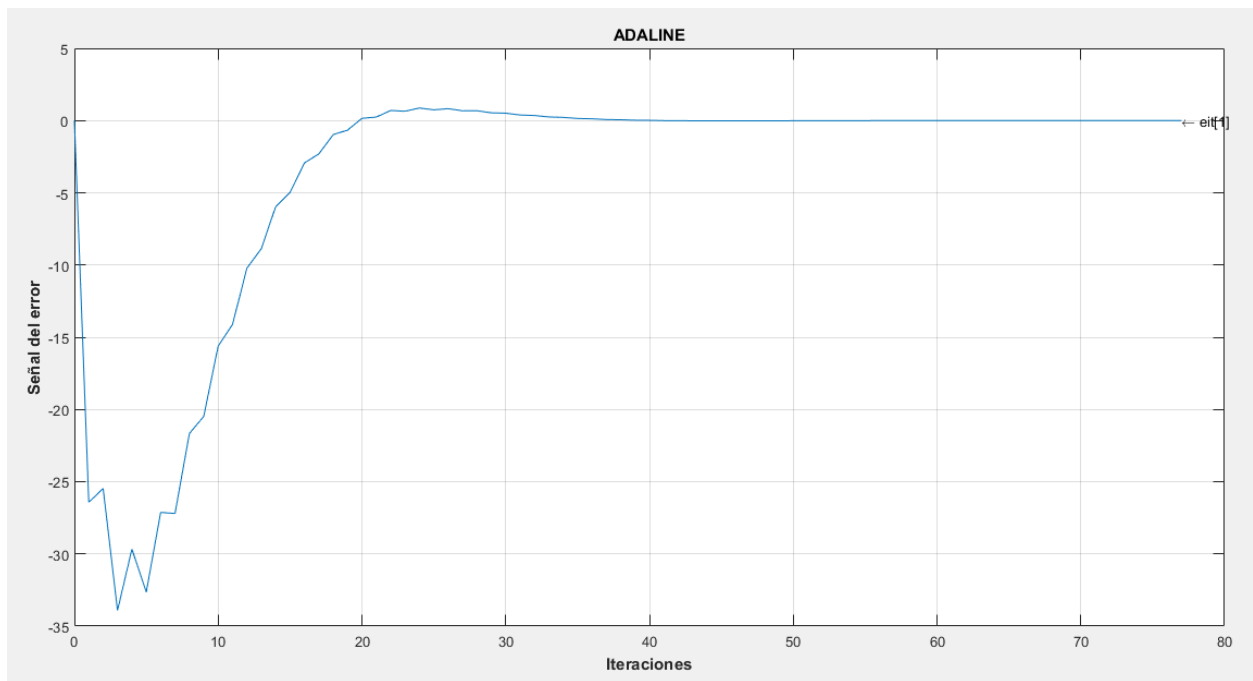


Figura 26. Señal del error.

EXPERIMENTO 4: CUATRO CLASES (REGLA DE APRENDIZAJE)

Para este experimento se utilizó el siguiente conjunto de entrenamiento que corresponde a el ejercicio propuesto en el libro Neural Network Design (Hagan, Widrow-Hoff Learning Exercises):

{[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]}

Le indicamos al programa de Matlab que esta vez se usará una red con bias, 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 8 patrones prototipo con una dimensión R=2 y que la dimensión de los targets es mayor a 1 (para este caso, será de dimensión 2), y nos pedirá la ubicación del archivo que contiene el conjunto de entrenamiento, para este caso será "experimento2.txt" como se muestra en la figura 27:

```
¿La red usará bias? (s/n): s
Ingresa el número de patrones prototipo: 3
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
(Sin extensión)
Nombre del Archivo: experimento1
Prototipos =
     3     6
     6     3
    -6     3

Targets =
    75
    75
   -75
```

Figura 27. 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 4 clases, posteriormente se solicitan los valores de itmax y Eit, para el ejemplo usaremos itmax=100 y Eit=0.00001 como se muestra en la figura 28:

```
Ingresa el número de clases:  2
Itmax:  100
Eit:  0.0001
```

Figura 28. Se solicitan el número de clases y los valores itmax y Eit.

Se nos muestra el valor recomendado de alfa (si es posible), e ingresamos el valor que deseamos como se muestra en la figura 29:

```
Se recomienda un valor de alfa menor a 0.384090
alpha =  0.1
```

Figura 29. Se muestra un valor aproximado de alfa al usuario, y se le pide que ingrese uno.

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 =
  -1.4916   -1.0616
  -0.7423    2.3505

b =
  -0.6156
   0.7481
```

Figura 30. 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 31), los cuales se mandarán a un archivo de resultados (figura 32). Además, se mostrarán 3 graficas, una de la evolución de los pesos de la matriz de pesos (figura 33), la evolución del vector bias (figura 34) y la señal del error (figura 35).

```
Se tuvo un aprendizaje con 3/4 de probabilidad de éxito en la iteración 7.
W =
  -0.3995    0.0668
   0.3552   -0.6021

b =
   0.0676
   0.3803
```

Figura 31. Resultado de la red.

```
resultados_20_13_6_11_2017.txt - Notepad
File Edit Format View Help
W =
-0.39953      0.0667990.35522 -0.60213
b =
0.06760.3803
```

Figura 32. Resultado de la red en archivo.

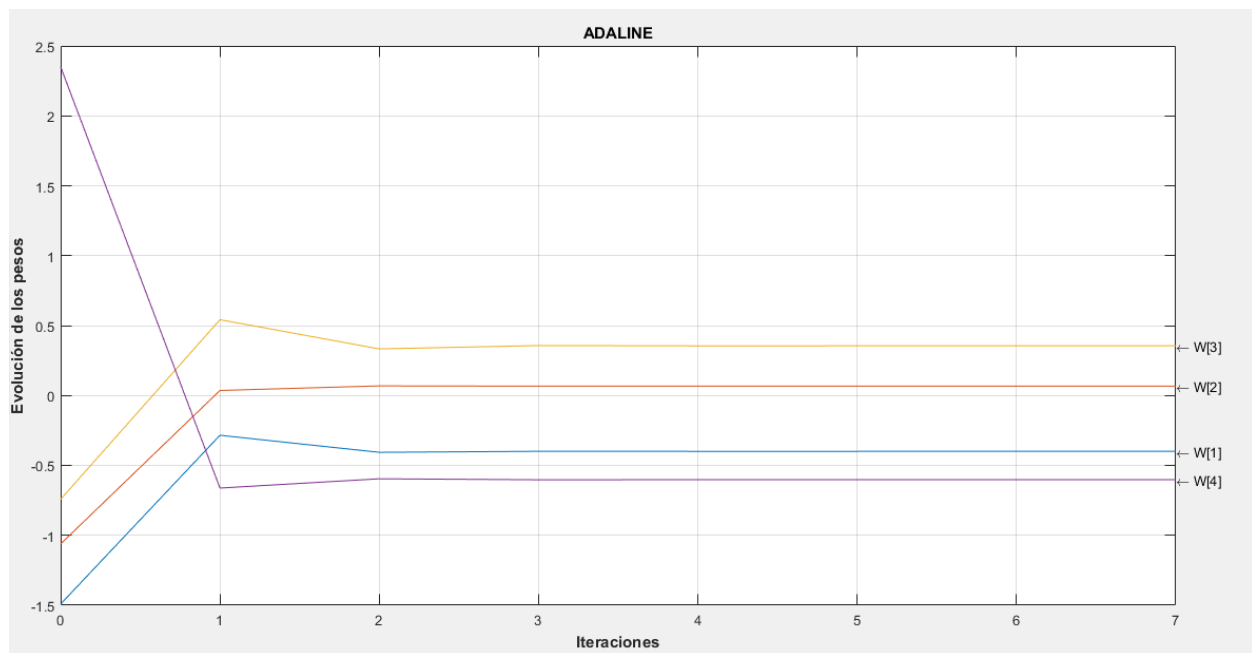


Figura 33. Evolución de los pesos de la matriz de pesos W.

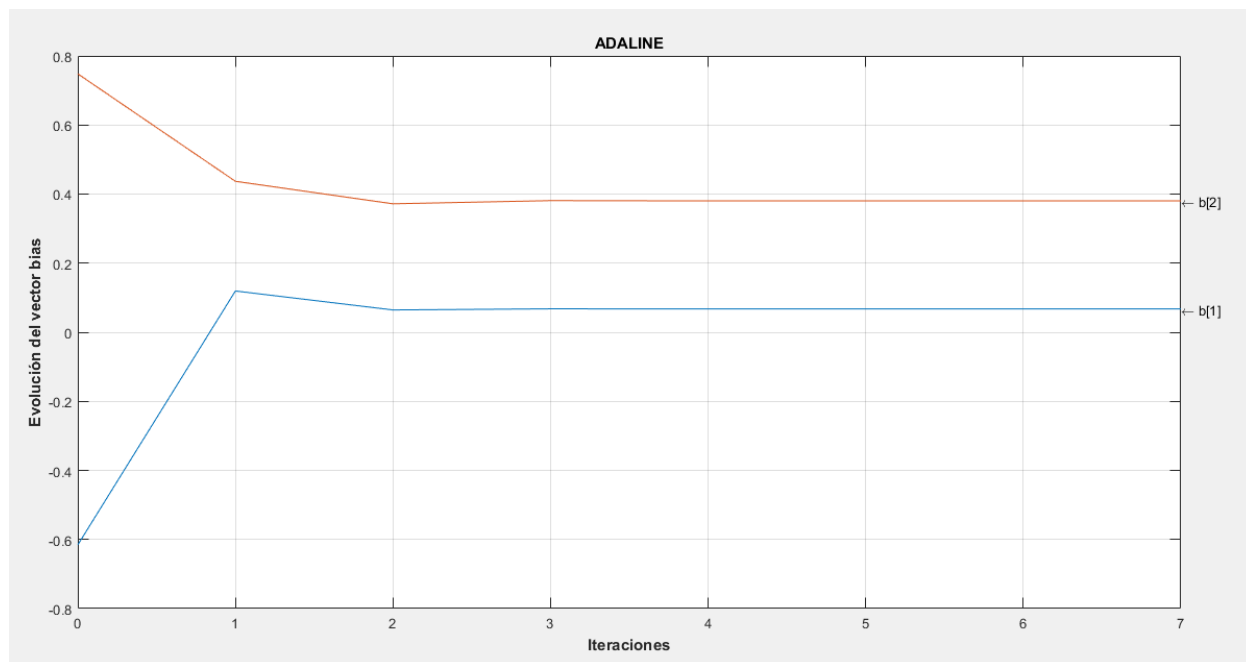


Figura 34. Evolución del vector bias b .

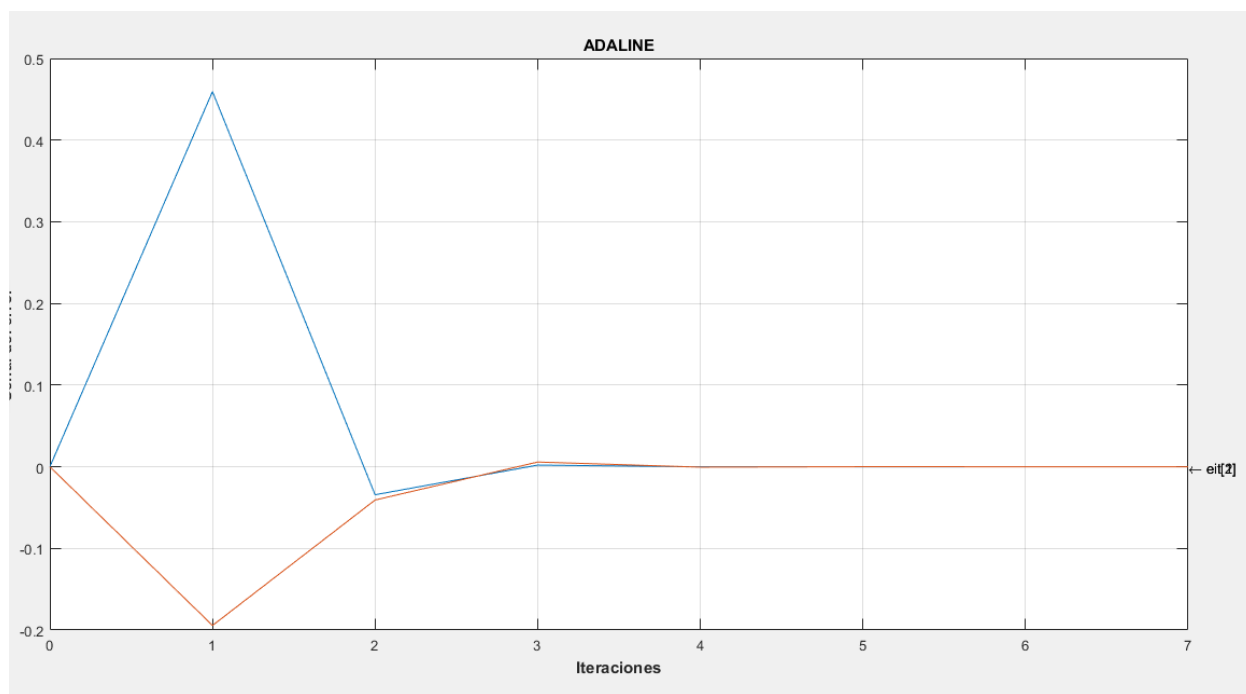


Figura 35. Señal del error.

DISCUSIÓN DE LOS RESULTADOS

Para los ejemplos del codificador vemos que la rapidez de convergencia fue muy alta utilizando un alfa de 0.5, de hecho, como experimento adicional, se tomó ese alfa para realizar el codificador de tamaño 2 hasta 10, funcionando correctamente, sin embargo, a partir de ahí, dicho alfa no resultaba efectivo y no daba los valores deseados si se quería que la señal del error fuera 0. En cuanto a los experimentos de la regla de aprendizaje, fue difícil encontrar un ejemplo donde la red convergiera exactamente a los patrones prototipo, por lo que se prefirió utilizar la segunda condición de finalización dando un valor de Eit cercano a cero, y como se pudo ver en las gráficas, la convergencia es muy cercana a los valores deseados, considerando que el error es aceptable.

Además, gracias al primer ejemplo de la regla de aprendizaje mostrado en este reporte, se pudo ver el comportamiento adaptivo de la red ADALINE, como la señal puede irse adaptando a ciertos valores deseados, de hecho, este tipo de red es usada en filtros adaptivos para eliminar el ruido de señales.

CONCLUSIONES

Considero que esta fue la práctica un poco más compleja junto con la de perceptrón, debido a que, en ambos casos, tenía que considerarse a el error de iteración como un vector, si los targets eran de dimensión mayor a 1. Los ejemplos propuestos, en principio eran linealmente separables, los cuales fueron tomados del libro. Sin embargo, no se pudo lograr un aprendizaje 100% exitoso, de hecho, se tomó en cuenta uno de los ejemplos realizados en clase. De hecho, el segundo experimento, fue uno propuesto en clase, y este de acuerdo a el libro, aunque no indica el número de iteraciones en las que converge la red, se muestran los valores deseados, los cuales, si los comparamos con los obtenidos en la práctica, varían por mucho.

$$W(\infty) = \begin{bmatrix} -0.5948 & -0.0523 \\ 0.1667 & -0.6667 \end{bmatrix}, b(\infty) = \begin{bmatrix} 0.0131 \\ 0.1667 \end{bmatrix}.$$

```
W =  
-0.3995    0.0668  
0.3552   -0.6021  
  
b =  
0.0676  
0.3803
```

Figura 36. En el lado izquierdo se muestran los valores óptimos, aunque estos estaban inicializados con otra matriz de pesos y otro vector bias, mientras que en la práctica fueron aleatorios.

Sin embargo, se realizó como experimento adicional dejar a la red con un itmax de 10000000 y un Eit=0 con alfa=0.002 y se llegó a un resultado más aproximado a el del libro:

```
Ingresa el número de clases: 4  
Itmax: 10000000  
Eit: 0  
Se recomienda un valor de alfa menor a 0.384090  
alpha = 0.002  
Se usarán las siguientes matrices iniciales:  
W =  
-1.2075    1.6302  
0.7172    0.4889  
  
b =  
1.0347  
0.7269  
  
W =  
-0.5926   -0.0496  
0.1694   -0.6652  
  
b =  
0.0134  
0.1693
```

Figura 37. Resultados obtenidos con itmax=10000000, Eit=0, alfa=0.002 (las matrices finales W y b son la última y penúltima mostradas en pantalla respectivamente).

Referencias

- 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 de la red: Adaline: <http://www.hugo-inc.com/RNA/Unidad%202/2.2.1.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>
- Hagan, M. T. (s.f.). Analysis of Convergence. En M. T. Hagan, *Neural Network Design* (págs. 318-319).
- Hagan, M. T. (s.f.). Widrow-Hoff Learning Exercises. En M. T. Hagan, *Neural Network Design* (pág. 352).
- Pereira, U. T. (s.f.). Obtenido de <http://medicinaycomplejidad.org/pdf/redes/Adaline.pdf>

ANEXO

Para una mejor visualización del código, recomiendo visitar el siguiente repositorio de Github:
<https://github.com/IvanovskyOrtega/Redes-Neuronales/tree/master/Practica-3/Adaline>

Adaline.m

```
clc
clear

% Se solicita al usuario si usará una red ADALINE con bias o sin bias.
op = input('¿La red usará bias? (s/n): ','s');
switch(op)

    % Si es con bias
    case 's'

        % Se solicita el número de patrones prototipo para una lectura
        % correcta de estos mismos en el archivo, además de su dimensión.
        numProt = input('Ingresa el número de patrones prototipo: ');
        R = input('Ingresa la dimensión de los vectores prototipo: ');

        % Se solicita indicar si la dimensión de los targets es mayor a 1,
        % para cambiar el modo de lectura de los patrones prototipo y
        % targets según el formato establecido en la práctica.
        res = input('¿La dimensión de los targets es mayor a 1? (s/n): ','s');

        % Si es mayor.
        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 archivo del conjunto de entrenamiento para la
            % lectura de los patrones prototipo y los targets sin
            % extensión.
            % Se usa la extensión .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 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 el número de clases a clasificar para calcular el
            % número de neuronas de la red.
            numeroDeClases = input('Ingresa el número de clases: ');
            n = 1;
            S = 1;
            while numeroDeClases > power(2, n)
                n = n+1;
                S = S+1;
            end
        end
    end
end
```

```

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

% Se intenta hacer un cálculo de un alfa adecuado, según el
% algoritmo de aprendizaje LMS.
try
    corM = (1/numProt)*(prototipos*prototipos');
    eigenvalores = eig(corM);
    alfa = 1/max(eigenvalores(:));
    fprintf('Se recomienda un valor de alfa menor a %f\n', alfa);
catch
end

% Se solicita el valor de alfa o factor de aprendizaje.
alfa = input('alpha = ');

% 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 usa un vector de 0's auxiliar para comparar con la señal
% del error en el aprendizaje de la red.
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(1), '%f\r\n', b(l));
end
for l=1:S
    fprintf(errores(1), '%f\r\n', 0);
end

```

```

end

% Inicia el aprendizaje
for i=1:itmax
    eit = zeros(S,1);
    for j=1:numProt
        a = (W*prototipos(j,:))'+b;
        ej = targets(j,:)-a;
        if ~isequal(ej,aux)
            W = W+(2*alfa)*(ej*prototipos(j,:));
            b = b+(2*alfa)*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(errores(l), '%f\r\n', eit(l));
    end

    % 1er criteri de finalizaci3n. Si el error es 0.
    if isequal(eit,aux)
        fprintf('Se tuvo un aprendizaje exitoso en la iteraci3n %d.\n',i);
        break;
    end
    % Segundo criterio de finalizaci3n
    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 3xito en
la iteraci3n %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 los valores de graficaci3n.
for k=1:S*R
    fclose(pesos(k));
end
for k=1:S
    fclose(bias(k));
end
for k=1:S
    fclose(errores(k));
end

% Inicia la graficaci3n.
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('ADALINE');
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('ADALINE');
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('ADALINE');
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 para leer los patrones prototipo y los targets
    % sin extensión.
    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 patrones 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 número de clases a clasificar para el cálculo
    % del número de neuronas de la red.
    numeroDeClases = input('Ingresa el número de clases: ');
    n = 1;
    S = 1;
    while numeroDeClases > power(2,n)
        n = n+1;
        S = S+1;
    end

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

    % Se intenta hacer el cálculo de un valor de alfa adecuado de
    % acuerdo al algoritmo de aprendizaje LSM.
    try
        corM = (1/numProt)*(prototipos*prototipos');
        eigenvalores = eig(corM);
        alfa = 1/max(eigenvalores(:));
        fprintf('Se recomienda un valor de alfa menor a %f\n',alfa);
    catch
    end

    % Se solicita el valor de alfa.
    alfa = input('alfa = ');

    % Se inicializan la matriz de pesos y el vector bias con ceros
    % en sus entradas 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,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 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 = 0;
    for j=1:numProt
        a = W*prototipos(j,:)'+b;
        ej = targets(j,:)-a;
        if ej ~= 0
            W = W+(2*alfa*(ej*prototipos(j,:)));
            b = b+(2*alfa)*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

% ler criterio de finalización
if eit == Eit

```

```

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

% Si se llegó al número máximo de iteraciones sin un
% aprendizaje exitoso, se indica al usuario que posiblemente
% los valores finales no son los adecuados.
if i == itmax
    disp('Es probable que los resultados finales no sean los correctos.
Prueba con otro alfa.');
```

```

    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 valores de graficación.
    for k=1:S*R
        fclose(pesos(k));
    end
    for k=1:S
        fclose(bias(k));
    end
    for k=1:S
        fcloseerrores(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('ADALINE');
    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('ADALINE');
    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('ADALINE');
    hold on
    for k=2:numeroDeArchivos
        plot(rango,valores(k,:));
        cad = strcat('\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

% Si es una red sin bias, se resuelve el BCD de tamaño n.
case 'n'
    disp('BCD');

    % Se solicita el tamaño del codificador y se calcula el valor
    % máximo decimal para el codificador.
    tam = input('Ingresa el tamaño del codificador: ');
    max = 2^tam-1;

    % Se crea el vector de targets
    targets = (0:max);

    % Se crea la matriz correspondiente a los targets en binario y se
    % invierte la matriz para que los prototipos empiecen desde el bit
    % más significativo.
    bin = de2bi(targets);
    bin = fliplr(bin);

    % Se muestra el conjunto de entrenamiento
    disp('El conjunto de entrnamiento es: ');
    disp([bin targets]);

```



```

% Se establecen los valores del tamaño de los vectores prototipo,
% el número de neuronas de la red.
R = max+1;
S = 1;

% Se inicializa la matriz de pesos con ceros en sus entradas y se
% muestra en pantalla
W = zeros(S,tam);
disp('Se usará la siguiente matriz inicial de pesos. ');
disp('W = ');
disp(W);

% Se solicitan los valores de itmax, Eit y alfa (factor de
% aprendizaje).
itmax = input('Itmax = ');
Eit = input('Eit = ');
alfa = input('alfa = ');

% Se abren los archivos necesarios para guardar los valores de
% graficación.
pesos = zeros(1,S*tam);
errores = zeros(1,S);
for k=1:S*tam
    nombre = sprintf('pesos%d.txt',k);
    pesos(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:tam
        fprintf(pesos(k), '%f\r\n', W(l,m));
        k = k+1;
    end
end
for l=1:S
    fprintf(errores(l), '%f\r\n', 0);
end

% Inicia el aprendizaje de la red.
for i=1:itmax
    t = 0;
    eit = 0;
    for j=1:R
        a = W*bin(j,:);
        ej = t-a;
        if ej ~= 0.0
            W = W+(2*alfa*ej*bin(j,:));
            eit = eit+(1/R)*ej;
        end
        t = t+1;
    end

    % Se imprimen valroes a archivo.
    k=1;
    for l=1:S
        for m=1:tam
            fprintf(pesos(k), '%f\r\n', W(l,m));

```

```

        k = k+1;
    end
end
for l=1:S
    fprintf(errores(l), '%f\r\n', eit(l));
end

%1er criterio de finalización.
if eit == Eit
    fprintf('Se ha tenido un aprendizaje exitoso en la
iteración %d.\n', i);
    break;
end
% Segundo criterio de finalización
if eit < Eit && 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 muestra la matriz de pesos final.
disp('W =');
disp(W);

% Se cierran los archivos de valores de graficación.
for k=1:S*tam
    fclose(pesos(k));
end
for k=1:S
    fclose(errores(k));
end

% Inicia la graficación.
numeroDeArchivos = S*tam;
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('ADALINE');
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('error%d.txt', k);
    aux = dlmread(nombre);
    valores(k,:) = aux';
end

```

```

        end
        figure
        plot(rango, valores(1,:));
        grid on
        text(itmax, valores(1,itmax+1), '\leftarrow eit[1]');
        title('ADALINE');
        hold on
        for k=2:numeroDeArchivos
            plot(rango, valores(k,:));
            cad = strcat('\leftarrow eit');
            text(itmax, valores(k,itmax+1), cad);
        end
        hold off
        xlabel('Iteraciones', 'FontWeight', 'bold')
        ylabel('Señal del error', 'FontWeight', 'bold')
        imprimirMatricesEnArchivo(W, 0, 's');
    otherwise
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
```