



INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE COMPUTO

Neural Networks

Moreno Armendáriz Marco Antonio

Integrantes

Pacheco Castillo Isaías

3CM19



Índice

1. Introducción	4
2. Marco teórico	4
3. Desarrollo	5
4. Poniendo en práctica la red	9
4.1. Mejorando el reconocimiento	15
5. Conclusiones	27
6. Referencias	27

Índice de figuras

1.	Arquitectura de la red ADALINE	4
2.	Dataset de entrada a la red	9
3.	Valores de las letras del dataset	9
4.	Archivo con los valores de entrada a la red	10
5.	Archivo con los valores de los targets	10
6.	Letra W target 7	11
7.	Se ingresa el archivo con los datos de entrada	11
8.	Se ingresa el archivo con los target	12
9.	Se ingresa el vector a clasificar	12
10.	Se ingresan los valores Eepoch, epochmax y alfa	13
11.	Gráficas	14
12.	Archivo de salida - valores de \mathbf{W} y b finales	15
13.	Archivo de salida - prueba del conjunto de entrada con los valores obtenidos	15
14.	Archivo de salida - clasificación del vector de entrada	15
15.	Dataset inicial	16
16.	Dataset corrimiento a la derecha	16
17.	Dataset corrimiento a la izquierda	17
18.	Ingresando el archivo con el dataset	18
19.	Ingresando el archivo con los targets	19
20.	Ingresando el archivo con los vectores a clasificar	19
21.	Ingresando los datos Eepoch, epochmax y alfa	20
22.	Gráficas de error acumulado, evolución de \mathbf{W} y el bias	20
23.	Criterio de finalización y valores finales de \mathbf{W} y b	20
24.	Resultados de evaluar el dataset con los valores finales de \mathbf{W} y b	21
25.	Resultados de clasificar los vectores de entrada del usuario	21
26.	Dataset sin ruido	22
27.	Dataset con ruido	22
28.	Ingresando el archivo con los vectores con ruido	23
29.	Ingresando los valores Eepoch, epochmax y alfa	23
30.	Gráficas de error acumulado, evolución de \mathbf{W} y el bias	24
31.	Criterio de finalización y valores finales de \mathbf{W} y b	24
32.	Resultados de evaluar el dataset con los valores finales de \mathbf{W} y b	25
33.	Resultados de clasificar los vectores de entrada del usuario	25
34.	A desplazada a la izquierda, W desplazada a la derecha y letra O con ruido	26

1. Introducción

Para esta tarea se va a desarrollar una red ADALINE vista en el curso para el reconocimiento de patrones, en este caso de letras en una matriz de 8 x 8. Para ello primero se presenta la arquitectura de la red, el modelo matemático y las reglas de aprendizaje necesarias para actualizar la matriz de pesos y el bias.

2. Marco teórico

En 1960 Widrow y su estudiante Marcian Hoff, introdujeron la red ADALINE (ADaptive LInear NEuron) y una regla de aprendizaje a la que llamaron LSM (Least Mean Square).

La red ADALINE es muy similar al perceptrón, excepto que su función de transferencia es lineal.

El algoritmo LMS es más poderoso que la regla de aprendizaje que utiliza el perceptrón. Si bien se garantiza que la regla del perceptrón convergerá en una solución que categorice correctamente los patrones de entrenamiento, la red resultante puede ser sensible al ruido, ya que los patrones a menudo se encuentran cerca de los límites de decisión. El algoritmo LMS minimiza el error cuadrático medio y, por lo tanto, intenta mover los límites de decisión lo más lejos posible de los patrones de entrenamiento. El algoritmo LMS ha encontrado muchos más usos prácticos que la regla de aprendizaje del perceptrón.

La arquitectura de la red se presenta en la siguiente imagen:

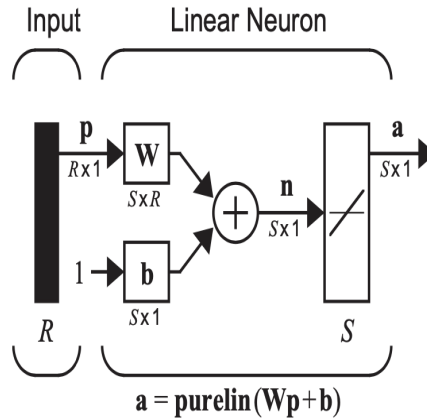


Figura 1: Arquitectura de la red ADALINE

La salida de la red esta dada por

$$a = \text{purelin}(\mathbf{W}\mathbf{p} + b) = \mathbf{W}\mathbf{p} + b \quad (1)$$

Las reglas de aprendizaje para los pesos y bias se obtiene utilizando el algoritmo LMS y se definen de la siguiente forma:

$$\mathbf{W}(k+1) = \mathbf{W}(k) + 2\alpha e(k)\mathbf{p}(k) \quad (2)$$

$$b(k+1) = b(k) + 2\alpha e(k) \quad (3)$$

Una vez que se conocen estos conceptos se pueden definir las dimensiones de la arquitectura, en este caso $R = 64$ y $S = 1$. El programa se desarrolló en Matlab usando la cuenta institucional como se muestra a continuación.

3. Desarrollo

El primer paso fue pedir al usuario que ingresara los archivos que contienen el conjunto de datos de entrada a la red y el vector que se va a clasificar con los resultados obtenidos. Para esto se utilizó la función **input**.

```
%Conjunto de datos de entrenamiento
archivoDataSet=input('Nombre del conjunto de datos de
    entrenamiento: ','s')
conjuntoDeEntrenamiento = load(archivoDataSet)

%Valores objetivo del conjunto de datos de entrenamiento
archivoTarget = input('Nombre del archivo con los objetivos: ','s
    ')
conjuntoTarget = load(archivoTarget)

valorAClasificar = input('Ingrese el vector de entrada que quiere
    clasificar: ')
```

También, utilizando esta función se ingresan las condiciones de finalización **epoch-max**, **alfa** y **Eepoch**.

```
%Leer condiciones de finalización
epochMax=input('Ingrese el número máximo de épocas: ')
Eepoch=input('Ingrese el error mínimo: ')
alfa=input('Ingrese el factor de aprendizaje alfa: ')
```

Se obtienen las variables numFilas y numDataset del conjunto de datos de entrada que corresponden a la dimensión de los datos de entrada, en este caso 7 letras x 64 valores.

```
%Se obtienen el número de valores de entrada y target
[numVectores, numValores] = size(conjuntoDeEntrenamiento)
```

Para la matriz de pesos se utiliza la función **random** para generar un arreglo de dimensión 1XR con valores aleatorios entre 0 y 1.

```
%Se determina la matriz de pesos 1xR
w=rand(1, numValores)

%Se determina el valor del bias 1x1
bias=1
```

También se utilizaron los siguientes archivos para poder ver paso a paso qué esta ocurriendo en la red y para poder grafica la evolución del error, los pesos y bias.

```
%Archivo de log
salida=fopen(salida_ar,'w')
%Archivo de errores
errores=fopen('errores_rna.txt','w')
%archivo de los valores de b
ev_bias=fopen('evolucion_bias.txt','w')
%archivo de los valores de w
ev_pesos=fopen('evolucion_pesos.txt','w')
```

Después se inician las iteraciones sobre el conjunto de datos de aprendizaje. En cada iteración se obtiene el resultado de aplicar el modelo matemático de la red Adaline el cual es $a = \text{purelin}(n)$, donde $n = Wp + b$. Por la forma en que se ingresa la información al programa, es necesario trasponer **p** para que la operación sea correcta. Una vez que se tiene el valor de n se le aplica la función de transferencia *purelin* y de esta forma se obtiene el resultado.

```
for i=0:epochMax
    fprintf(salida, '\n Época %d \n', i);
    %error acumulado
    eepoch = 0
    %Se itera sobre el conjunto de datos de aprendizaje
    for x=1:numVectores
        %Se obtiene el valor de n = wp+b
        n = (w*transpose(conjuntoDeEntrenamiento(x,1: numValores)
        ))+bias
        a = purelin(n)
        %Se obtiene e = t-a
        e = conjuntoTarget(x)-a
        %Se acumula el error
        eepoch = eepoch+e
```

En caso de que el valor de e obtenido sea diferente de 0 se aplican las reglas de aprendizaje para obtener los nuevos valores de los pesos y bias. Cada vez que se aplican las reglas de aprendizaje se guardan en el archivo *ev bias* y *ev pesos* los nuevos valores obtenidos y el error acumulado cuando finaliza cada época.

```
%Si el error es diferente de 0 se utiliza la regla de aprendizaje
if e ~= 0
    w = w+((2*alfa)*e*conjuntoDeEntrenamiento(x,1: numValores))
    bias = bias+(2*alfa*e)
    fprintf(ev_bias,' %f', bias);
    fprintf(ev_pesos,' %f', w);
    fprintf(ev_pesos,';\n');
```

Cada que se termina una época se realiza el cálculo del error acumulado y se verifican los criterios de finalización. Si se cumple algún criterio de finalización se termina de iterar.

```
%1/N(sumaErrores)
eepoch = (abs((eepoch)))/(numVectores)
fprintf(salida,'\nError acumulado(%d) = %f\n',i,eepoch);
fprintf(errores,' %f',eepoch);

%Se verifican los criterios de finalización
if eepoch == 0
    terminflag = 2
    break
elseif eepoch <= Eepoch
    terminflag = 1
    break
end
```

Se utilizó la variable *terminflag* para determinar cuál fue el criterio de finalización de la red. Ya que la red ha terminado la etapa de aprendizaje se muestra gráficamente la evolución de los pesos, bias y el error acumulado por época.

```
%Se grafica el error
errores_grafica = load('errores_rna.txt')
ev_bias_grafica = load('evolucion_bias.txt')
ev_pesos_grafica = load('evolucion_pesos.txt')
subplot(3,1,1)
plot(errores_grafica)
title('Convergencia del error')
xlabel('Épocas')
ylabel('Error acumulado')

subplot(3,1,2)
plot(ev_pesos_grafica)
title('Gráfica de W')

subplot(3,1,3)
plot(ev_bias_grafica)
title('Gráfica de b')
```

Como comprobación se toman los datos de entrada a la red y el vector de entrada ingresado por el usuario y se clasifican utilizando los datos obtenidos. Para mostrar los

resultados se utiliza un archivo de salida ya que no me gustó como muestra los valores la consola.

```
%Se prueban los resultados obtenidos
for i=1:numVectores
    n = (w*transpose(conjuntoDeEntrenamiento(i,1: numValores)))+
        bias
    a = purelin(n)
    fprintf(salida, '\nDato(%d) = %f\n', i, a);
end

n = (w*transpose(valorAClasificar))+bias
a = purelin(n)

fprintf(salida, '\nEl vector de entrada: \n ')
fprintf(salida, ' %f', valorAClasificar)
fprintf(salida, '\n pertenece a la clase %f', a)

fprintf('Matriz de pesos')
disp(w)
fprintf('\n bias: %f \n', bias)
disp(valorAClasificar)
fprintf('\n pertenece a la clase %f \n', a)
```


4. Poniendo en práctica la red

Para comprobar le funcionamiento del programa desarrollado se utilizó el siguiente conjunto de datos de entrenamiento que corresponde a las letras A, D, O, Q, G, V y W en una matriz de 8 x 8.

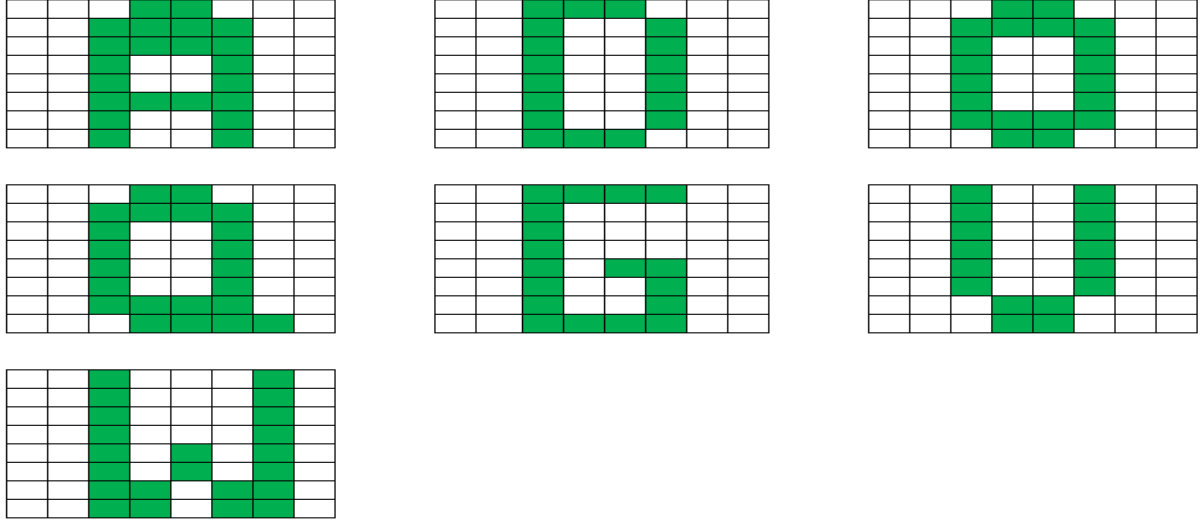


Figura 2: Dataset de entrada a la red

Cada que una celda de la matriz este rellena se toma como un 1 y cuando este vacía se toma como un 0 como se muestra a continuación.

0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0

0	0	1	1	1	0	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	1	1	0	0	0

0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0

0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	0	1	1	0	0
0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0

0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	1	0	1	0	0
0	0	1	1	0	1	0	0

Figura 3: Valores de las letras del dataset

Los valores objetivo son: A ->1, D ->2, O ->3, Q ->4, G ->5, V ->6 y W ->8. Los archivos de entradas a la red son *data2.txt* y *target2.txt*.

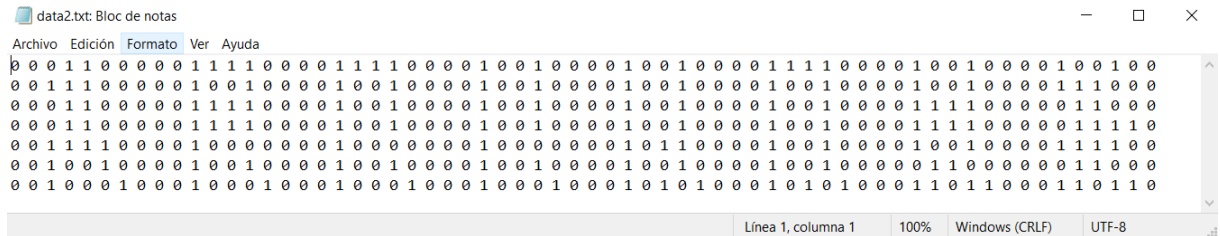


Figura 4: Archivo con los valores de entrada a la red

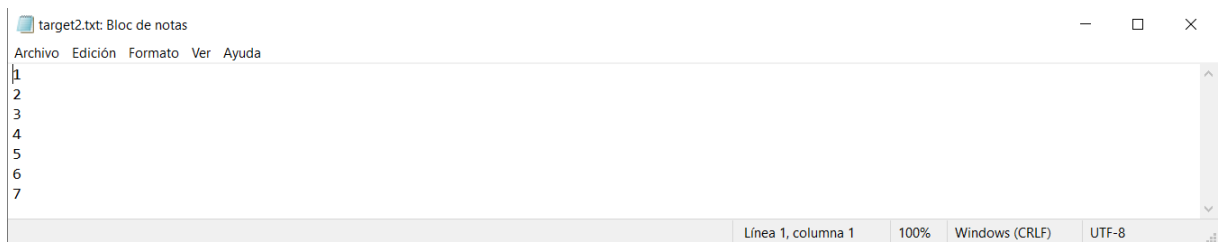


Figura 5: Archivo con los valores de los targets

El vector de entrada a clasificar será la letra W con target 7, como lo muestra la siguiente imagen:

0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	0
0	0	1	0	1	0	1	0
0	0	1	0	1	0	1	0
0	0	1	1	0	1	1	0
0	0	1	1	0	1	1	0

Figura 6: Letra W target 7

El resultado se muestra en las siguientes imágenes:

The screenshot shows a MATLAB Editor window with a script named 'AdalineIPC.m'. The script contains the following code:

```

1 %RNA ADALINE - Pacheco Castillo Isaias
2
3 %Conjunto de datos de entrenamiento

```

The Command Window displays the execution of the script, showing the loading of the dataset file 'data2.txt' and the resulting matrix 'conjuntoDeEntrenamiento'.

```

>> AdalineIPC
Nombre del conjunto de datos de entrenamiento: data2.txt

archivoDataSet =

    'data2.txt'

conjuntoDeEntrenamiento =

Columns 1 through 27

    0    0    0    1    1    0    0    0    0    0    0    1    1    1    1    0    0    0    0    1    1    1    1    0    0    0    0
    0    0    1    1    1    0    0    0    0    0    0    1    0    0    1    0    0    0    0    1    0    0    1    0    0    0    0
    0    0    0    1    1    0    0    0    0    0    0    1    1    1    1    0    0    0    0    1    0    0    1    0    0    0    0
    0    0    0    1    1    0    0    0    0    0    0    1    1    1    1    0    0    0    0    1    0    0    1    0    0    0    0
    0    0    1    1    1    1    0    0    0    0    0    1    0    0    0    0    0    0    0    1    0    0    0    0    0    0    0
    0    0    1    0    0    1    0    0    0    0    0    1    0    0    1    0    0    0    0    1    0    0    1    0    0    0    0
    0    0    1    0    0    0    1    0    0    0    1    0    0    0    1    0    0    0    1    0    0    0    1    0    0    0    0

Columns 28 through 54

    0    0    1    0    0    0    0    0    1    0    0    1    0    0    0    0    1    1    1    1    0    0    0    0    1    0    0
    0    0    1    0    0    0    0    0    1    0    0    1    0    0    0    0    1    0    0    1    0    0    0    0    1    0    0
    0    0    1    0    0    0    0    0    1    0    0    1    0    0    0    0    1    0    0    1    0    0    0    0    1    1    1
    0    0    1    0    0    0    0    0    1    0    0    1    0    0    0    0    1    0    0    1    0    0    0    0    1    1    1

```

Figura 7: Se ingresa el archivo con los datos de entrada

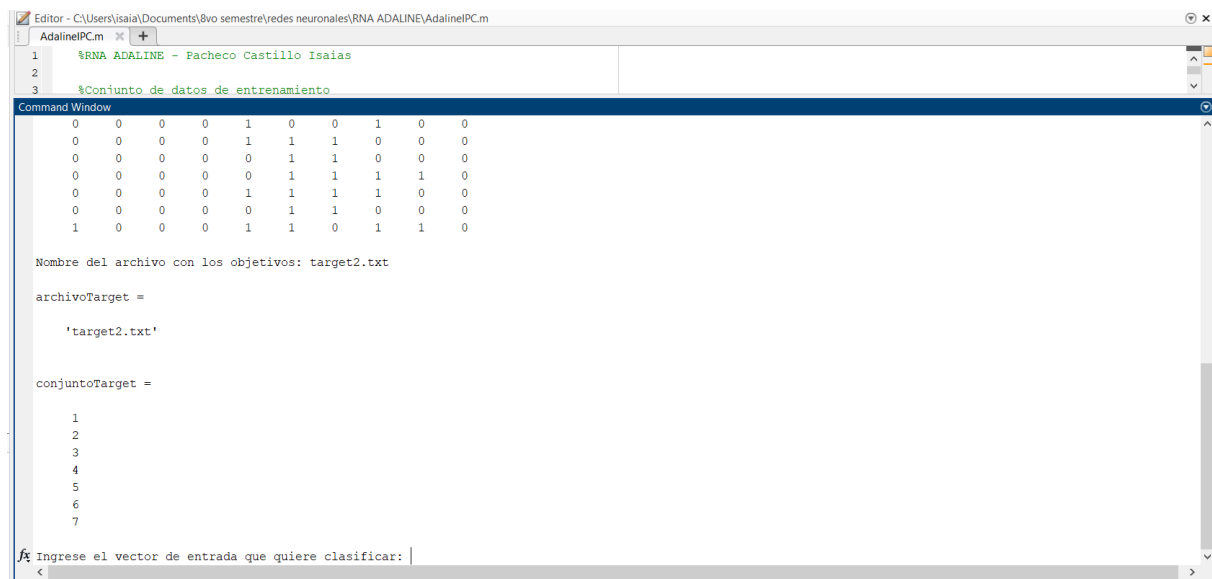


Figura 8: Se ingresa el archivo con los target

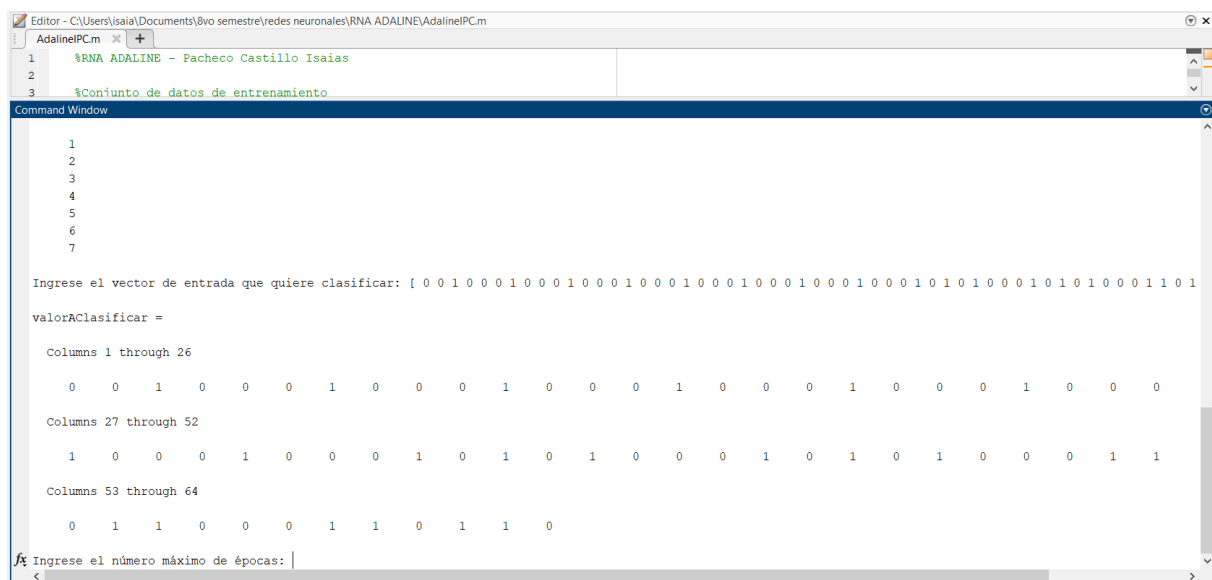


Figura 9: Se ingresa el vector a clasificar

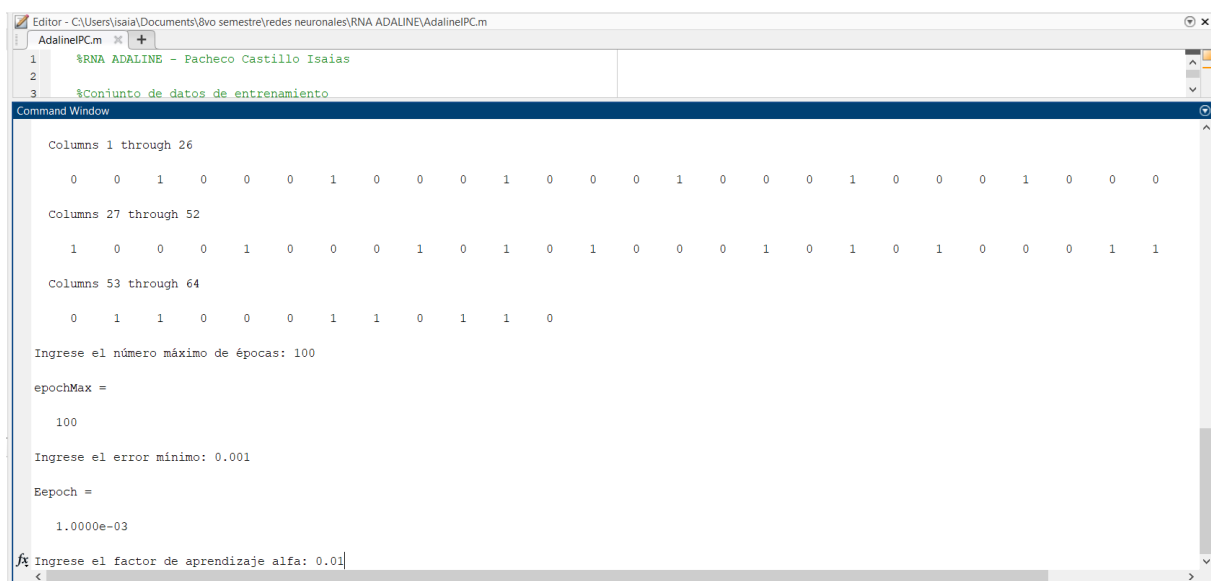


Figura 10: Se ingresan los valores Eepoch, epochmax y alfa

Al terminar el aprendizaje se muestran las graficas de evolución de \mathbf{W} , \mathbf{b} y el error acumulado por época.

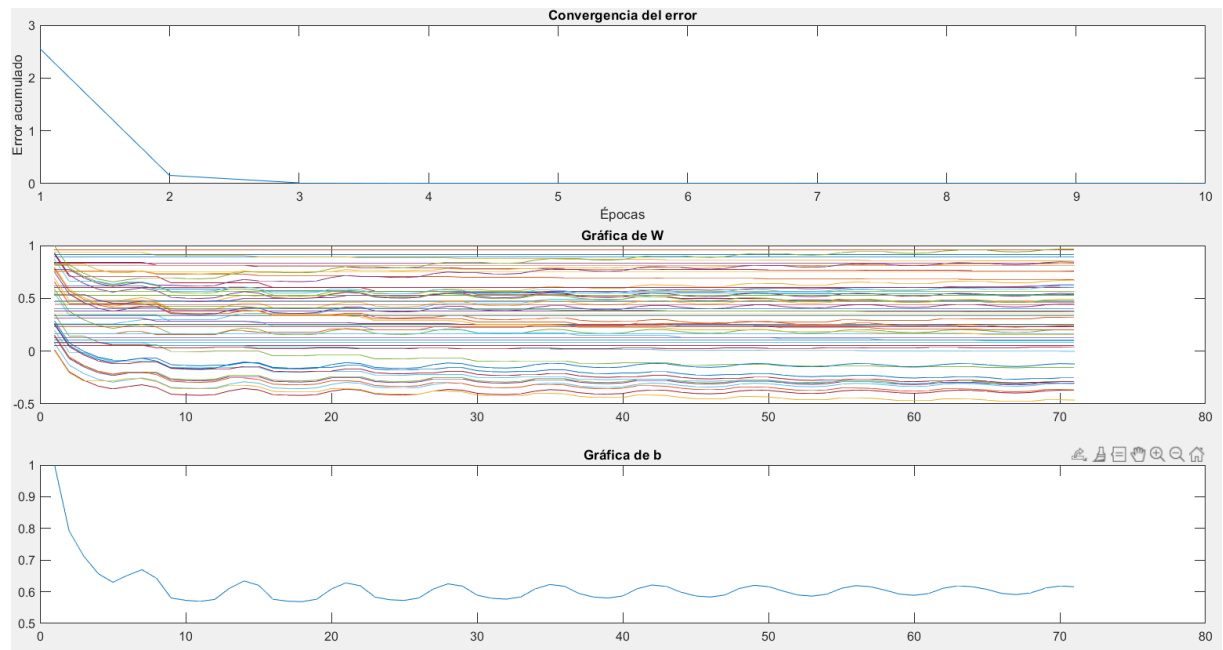


Figura 11: Gráficas

El archivo de salida muestra todos los valores de \mathbf{W} , b , el error acumulado, el criterio de finalización y los valores finales de \mathbf{W} y b . También muestra la evaluación de los valores de entrada y el vector ingresado por el usuario.

```

250 Error acumulado(9) = 0.000791
251
252 <<<Terminó por criterio de finalización eepoch<=Eepoch>>>
253
254 Valores finales
255 W = 0.890903 0.959291 0.666380 -0.299581 -0.288911 0.629708 0.759359 0.254282 0.814285 0.243525 0.544947 -0.153497 -0.306886 -0.288827
0.534686 0.473289 0.351660 0.830829 0.200947 0.097591 0.465061 -0.254072 0.675842 0.753729 0.380446 0.567822 -0.308462 0.053950 0.530798
0.239257 0.852652 0.129906 0.568824 0.469391 -0.372415 0.337123 0.317776 0.491326 0.229856 0.528533 0.165649 0.601982 -0.121345 0.201946
0.155723 0.445194 0.369183 0.083821 0.228977 0.913337 -0.367186 0.828357 0.622241 0.476571 -0.003183 0.442678 0.106653 0.961898 -0.463581
0.842727 0.966478 0.603807 0.034729 0.399783
256 Bias = 0.615683

```

Figura 12: Archivo de salida - valores de \mathbf{W} y b finales

```

258 Prueba de resultados
259
260 Dato(1) = 1.313291
261
262 Dato(2) = 2.725131
263
264 Dato(3) = 3.512548
265
266 Dato(4) = 4.151085
267
268 Dato(5) = 4.580064
269
270 Dato(6) = 5.748124
271
272 Dato(7) = 7.073053
273

```

Figura 13: Archivo de salida - prueba del conjunto de entrada con los valores obtenidos

```

274 El vector de entrada:
275 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000 1.000000
0.000000 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000 1.000000 0.000000 0.000000 0.000000
1.000000 0.000000 0.000000 0.000000 1.000000 0.000000 1.000000 0.000000 1.000000 0.000000 0.000000 0.000000 1.000000 0.000000 1.000000
0.000000 1.000000 0.000000 0.000000 0.000000 1.000000 1.000000 0.000000 1.000000 1.000000 0.000000 0.000000 0.000000 1.000000 1.000000
0.000000 1.000000 1.000000 0.000000
276 pertenece a la clase 7.073053

```

Figura 14: Archivo de salida - clasificación del vector de entrada

como se puede observar en 13 y 14 el vector ingresado se clasifico correctamente.

4.1. Mejorando el reconocimiento

Para mejorar el reconocimiento de patrones se realizaron corrimientos al conjunto de datos de entrada como se muestra a continuación.

0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0

0	0	1	1	1	0	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	1	1	0	0	0

0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0

0	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0
0	0	0	1	1	1	1	0

0	0	1	1	1	1	0	0
0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	0	1	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0

0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0

0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	0
0	0	1	0	0	0	1	0
0	0	1	0	1	0	1	0
0	0	1	0	1	0	1	0
0	0	1	1	0	1	1	0
0	0	1	1	0	1	1	0

Figura 15: Dataset inicial

0	0	0	0	1	1	0	0
0	0	0	1	1	1	1	0
0	0	0	1	1	1	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	1	1	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0

0	0	0	1	1	1	0	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	1	1	0	0

0	0	0	0	1	1	0	0
0	0	0	1	1	1	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	1	1	1	0
0	0	0	0	1	1	0	0

0	0	0	0	1	1	0	0
0	0	0	1	1	1	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	1	1	1	0
0	0	0	0	1	1	1	1

0	0	0	1	1	1	1	0
0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	1	0	1	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	1	1	1	0

0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0
0	0	0	1	1	1	0	0
0	0	0	0	1	1	0	0

0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	0	1	0	1
0	0	0	1	0	1	0	1
0	0	0	1	1	0	1	1
0	0	0	1	1	0	1	1

Figura 16: Dataset corrimiento a la derecha

0	0	1	1	0	0	0	0
0	1	1	1	1	0	0	0
0	1	1	1	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	1	1	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0

0	1	1	1	0	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	1	1	0	0	0	0

0	0	1	1	0	0	0	0
0	1	1	1	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	1	1	1	0	0	0
0	0	1	1	0	0	0	0

0	0	1	1	0	0	0	0
0	1	1	1	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	1	1	1	0	0	0
0	0	1	1	1	1	0	0

0	1	1	1	1	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	1	1	1	0	0	0

0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	0	1	1	0	0	0	0
0	0	1	1	0	0	0	0

0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	0	0	0	1	0	0
0	1	0	1	0	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	1	1	0	0
0	1	1	0	1	1	0	0

Figura 17: Dataset corrimiento a la izquierda

De esta forma la red podrá reconocer también las letras recorridas hacia la izquierda o la derecha. También se modificó el programa para que lea de un archivo los vectores que el usuario quiera clasificar, para ello se agregaron las siguientes líneas.

```
%valorAClasificar = input('Ingrese el vector de entrada que  
quiere clasificar: ')  
archivoVectoresClasificar = input('Ingrese el archivo con los  
vectores a clasificar: ', 's')  
vectoresClasificar = load(archivoVectoresClasificar)
```

```
[numVectores, numValores] = size(vectoresClasificar)  
fprintf(salida, '\nClasificación de vectores de entrada\n');  
for i=1:numVectores  
    n = (w*transpose(vectoresClasificar(i,1: numValores)))+bias  
    a = purelin(n)  
    fprintf(salida, '\nVector(%d) = %f\n', i, a);  
end
```

El archivo de vectores a clasificar contendrá una A desplazada a la izquierda, Una W desplazada a la derecha y una O normal. Los resultados obtenidos son los siguientes.

```
Editor - C:\Users\isaia\Documents\8vo semestre\redes neuronales\RNA ADALINE\AdalinePC.m
AdalinePC.m
1 %RNA ADALINE - Pacheco Castillo Isaias

Command Window
>> AdalineIPC
Nombre del conjunto de datos de entrenamiento: data.txt

archivoDataSet =

    'data.txt'

conjuntoDeEntrenamiento =

    Columns 1 through 26

    0    0    0    1    1    0    0    0    0    0    1    1    1    1    0    0    0    0    1    1    1    1    0    0    0    0
    0    0    1    1    1    0    0    0    0    0    1    0    0    1    0    0    0    0    1    0    0    1    0    0    0    0
    0    0    0    1    1    0    0    0    0    0    1    1    1    1    0    0    0    0    1    0    0    1    0    0    0    0
    0    0    0    1    1    0    0    0    0    0    1    1    1    1    0    0    0    0    1    0    0    1    0    0    0    0
    0    0    1    1    1    1    0    0    0    0    1    0    0    0    0    0    0    0    1    0    0    0    0    0    0    0
    0    0    1    0    0    1    0    0    0    0    1    0    0    1    0    0    0    0    1    0    0    1    0    0    0    0
    0    0    1    0    0    1    0    0    0    1    0    0    0    1    0    0    0    1    0    0    0    1    0    0    0    0
    0    0    0    0    1    1    0    0    0    0    0    1    1    1    1    0    0    0    0    0    0    1    1    1    0    0
    0    0    0    1    1    1    0    0    0    0    0    1    0    0    1    0    0    0    1    0    0    1    0    0    0    0
    0    0    0    0    1    1    0    0    0    0    0    1    1    1    1    0    0    0    0    1    0    0    1    0    0    0
    0    0    0    1    1    1    1    0    0    0    0    1    0    0    0    0    0    0    1    0    0    0    0    0    0    0
    0    0    0    1    1    1    1    0    0    0    0    1    0    0    0    0    0    0    1    0    0    0    0    0    0    0
    0    0    0    1    0    0    1    0    0    0    0    1    0    0    0    1    0    0    0    0    0    1    0    0    0    0
    0    0    1    1    0    0    0    0    0    0    1    1    1    1    0    0    0    1    1    1    1    0    0    0    0    1
    0    1    1    1    0    0    0    0    0    1    0    0    1    0    0    0    0    1    0    0    1    0    0    0    0    1
```

Figura 18: Ingresando el archivo con el dataset

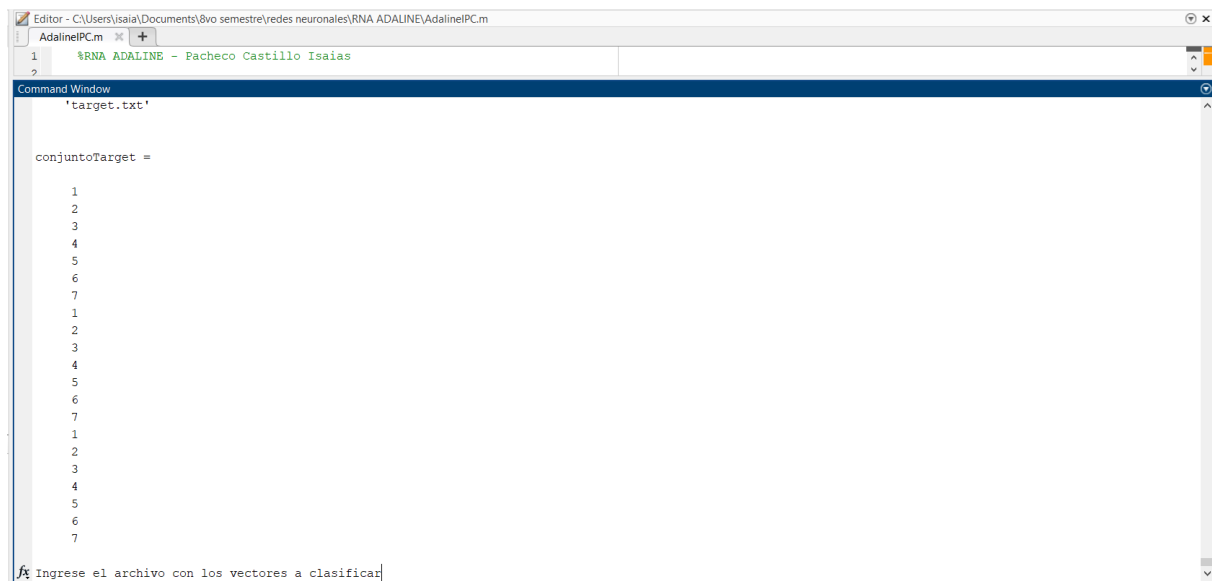


Figura 19: Ingresando el archivo con los targets

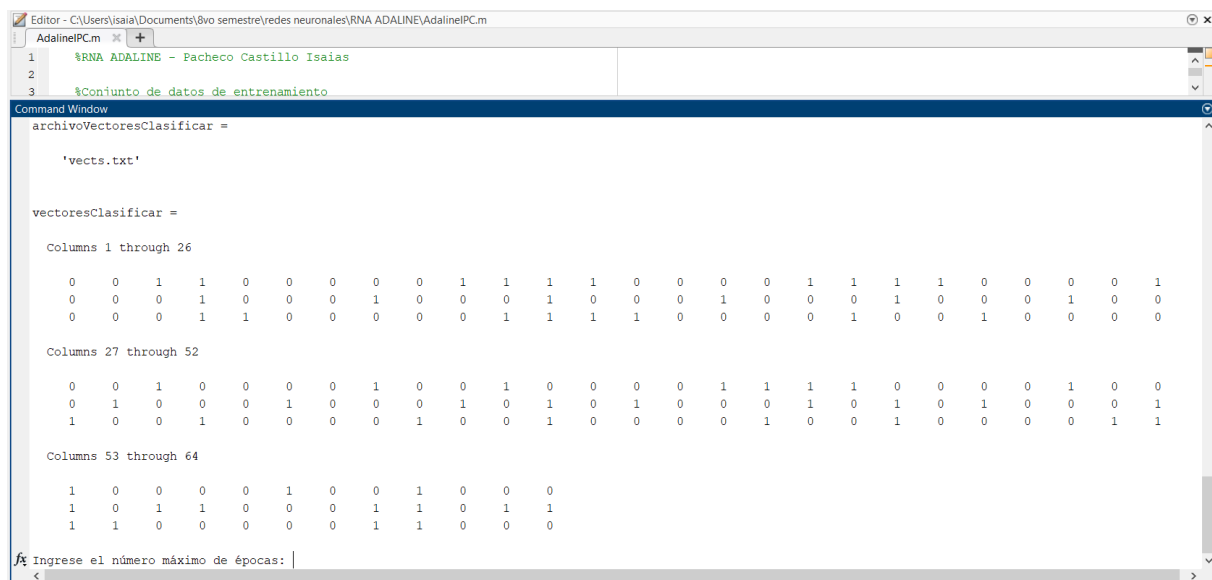


Figura 20: Ingresando el archivo con los vectores a clasificar

```

Editor - C:\Users\isaia\Documents\8vo semestre\redes neuronales\RNA ADALINE\AdalinePC.m
AdalinePC.m
1 %RNA ADALINE - Pacheco Castillo Isaías
2
3 %Conjunto de datos de entrenamiento

Command Window

Columns 27 through 52

0 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 1 1 1 1 0 0 0 0 1 0 0
0 1 0 0 0 0 1 0 0 0 1 0 1 0 1 0 0 0 1 0 1 0 1 0 0 0 1
1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 0 0 0 1 1

Columns 53 through 64

1 0 0 0 0 1 0 0 1 0 0 0
1 0 1 1 0 0 0 1 1 0 1 1
1 1 0 0 0 0 0 1 1 0 0 0

Ingrese el número máximo de épocas: 100

epochMax =

100

Ingrese el error mínimo: 0.001

Eepoch =

1.0000e-03

fx Ingrese el factor de aprendizaje alfa: 0.01

```

Figura 21: Ingresando los datos Eepoch, epochmax y alfa

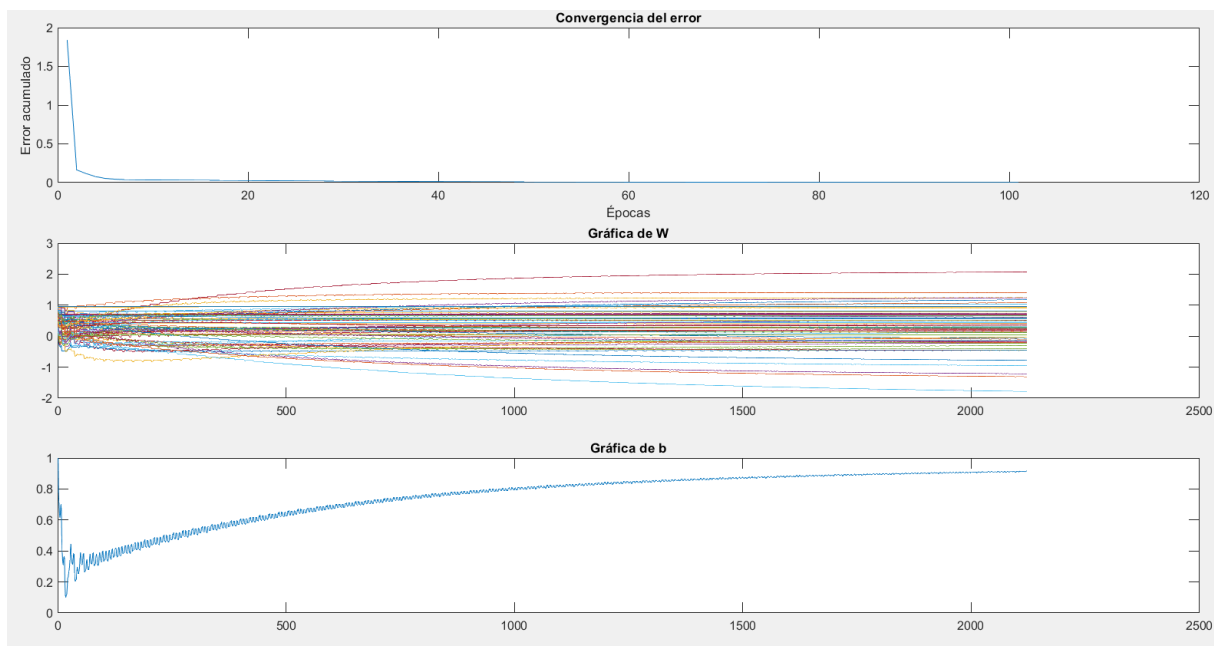


Figura 22: Gráficas de error acumulado, evolución de W y el bias

```

salida-data2.txt  salida.txt  salida-data.txt
6768 <<<Terminó por número máximo de iteraciones (epochMax)>>>
6769
6770
6771 Valores finales
6772 W = 0.814724 1.407056 -0.454543 -1.212095 -0.301722 0.304971 2.071560 0.553495 0.957507 0.794100 -0.226430 -0.407189 -0.952184 -0.461664
0.130322 0.148500 0.421761 0.744947 0.293856 0.121606 -0.149394 -0.461315 0.179170 0.940606 0.678735 0.586951 1.031142 0.189998 -0.069829
-0.095308 0.036087 0.038446 0.276923 -0.124618 0.385141 1.175747 1.078544 0.910317 0.619201 0.041059 0.438744 0.210770 0.267165 -0.205208
-0.023402 0.513512 0.114565 0.652926 0.709365 -0.780874 0.239657 1.244804 1.239172 -0.226546 -1.781713 0.504977 0.959744 -1.309483
0.502117 -0.189309 0.666556 0.480190 0.349721 0.592719
6773 Bias = 0.914991
6774

```

Figura 23: Criterio de finalización y valores finales de W y b

```
salida-data2.txt x data.txt x salida-data.txt x
6780
6781 Dato(3) = 3.172550
6782
6783 Dato(4) = 4.002462
6784
6785 Dato(5) = 4.977513
6786
6787 Dato(6) = 5.883057
6788
6789 Dato(7) = 7.036290
6790
6791 Dato(8) = 0.961597
6792
6793 Dato(9) = 2.080972
6794
6795 Dato(10) = 3.081154
6796
6797 Dato(11) = 4.023595
6798
6799 Dato(12) = 5.066991
6800
6801 Dato(13) = 5.888127
6802
6803 Dato(14) = 7.003639
6804
6805 Dato(15) = 1.003408
6806
6807 Dato(16) = 2.212916
6808
6809 Dato(17) = 2.966185
6810
6811 Dato(18) = 4.112931
6812
6813 Dato(19) = 4.924903
6814
6815 Dato(20) = 5.913479
6816
6817 Dato(21) = 7.026038
6818
```

Figura 24: Resultados de evaluar el dataset con los valores finales de \mathbf{W} y \mathbf{b}

```
6818
6819 Clasificación de vectores de entrada
6820
6821 Vector(1) = 1.003408
6822
6823 Vector(2) = 7.003639
6824
6825 Vector(3) = 3.172550
6826
```

Figura 25: Resultados de clasificar los vectores de entrada del usuario

Por último se prueba la red ingresando los vectores de entrada con ruido como se muestra en 26 y 27.

Figura 26: Dataset sin ruido

Figura 27: Dataset con ruido

Los resultados obtenidos ingresando estos valores para ser clasificados se muestran a continuación.

```

Editor - C:\Users\isaia\Documents\8vo semestre\redes neuronales\RNA ADALINE\AdalinePC.m
AdalinePC.m
1 %RNA ADALINE - Pacheco Castillo Isaías
2
3 %Conjunto de datos de entrenamiento

Command Window
archivoVectoresClasificar =
    'vectsmod.txt'

vectorsClasificar =

Columns 1 through 26

    1     0     1     1     0     0     0     0     0     0     1     1     1     1     0     0     0     0     1     1     1     1     0     0     0     1
    1     0     0     1     0     0     0     0     1     0     0     0     1     0     0     0     1     0     0     0     1     0     0     0     0
    1     0     0     1     1     0     0     0     0     0     0     1     1     1     1     0     0     0     0     1     0     0     1     0     0     0

Columns 27 through 52

    0     0     1     0     0     0     0     1     0     0     1     0     0     0     0     1     1     1     1     0     0     0     0     1     0     0
    0     1     0     0     0     0     1     0     0     0     1     0     1     0     1     0     0     0     1     0     1     0     1     0     0     1
    1     0     0     1     0     0     0     0     1     0     0     1     0     0     0     0     1     0     0     1     0     0     0     0     1     1

Columns 53 through 64

    1     0     0     0     0     1     0     0     1     0     0     1
    1     0     1     1     0     0     0     1     1     0     1     0
    1     1     0     0     0     0     0     1     1     0     0     1
  
```

Ingrese el número máximo de épocas: |

Figura 28: Ingresando el archivo con los vectores con ruido

```

Editor - C:\Users\isaia\Documents\8vo semestre\redes neuronales\RNA ADALINE\AdalinePC.m
AdalinePC.m
1 %RNA ADALINE - Pacheco Castillo Isaías
2
3 %Conjunto de datos de entrenamiento

Command Window

Columns 27 through 52

    0     0     1     0     0     0     0     1     0     0     1     0     0     0     0     1     1     1     1     0     0     0     0     1     0     0
    0     1     0     0     0     0     1     0     0     0     1     0     1     0     1     0     0     0     1     0     1     0     1     0     0     1
    1     0     0     1     0     0     0     0     1     0     0     1     0     0     0     0     1     0     0     1     0     0     0     0     1     1

Columns 53 through 64

    1     0     0     0     0     1     0     0     1     0     0     1
    1     0     1     1     0     0     0     1     1     0     1     0
    1     1     0     0     0     0     0     1     1     0     0     1

Ingrese el número máximo de épocas: 100

epochMax =

    100

Ingrese el error mínimo: 0.01

Epoch =

    0.0100

Ingrese el factor de aprendizaje alfa: 0.01|
  
```

Figura 29: Ingresando los valores Eepoch, epochmax y alfa

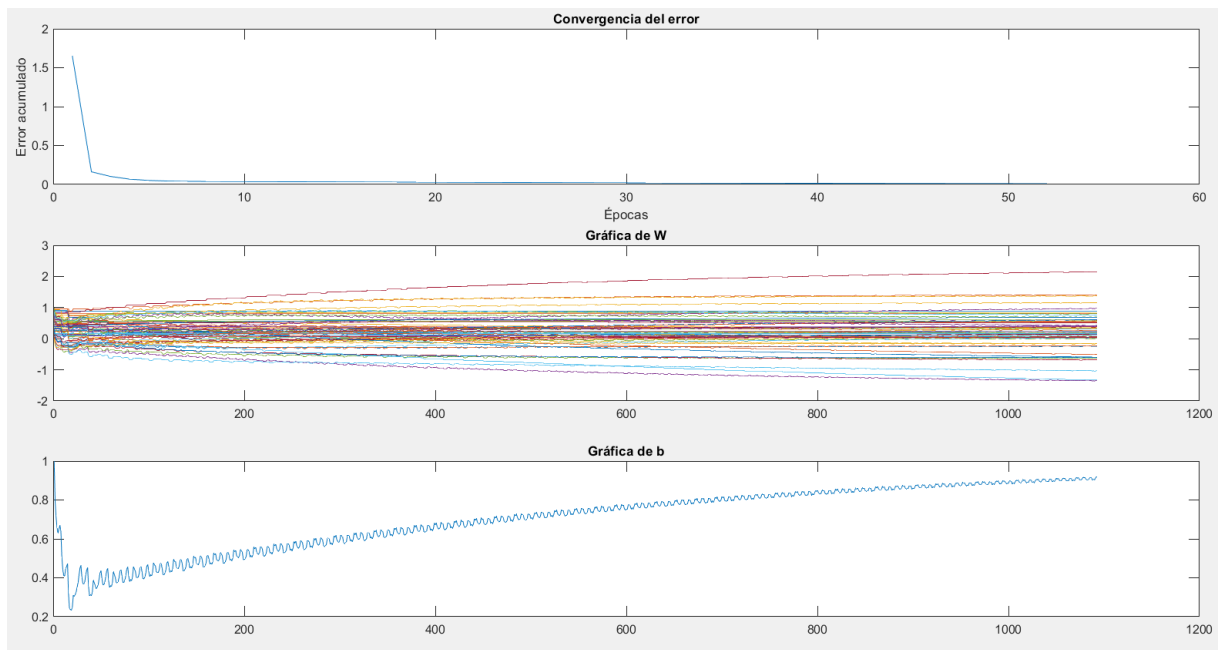


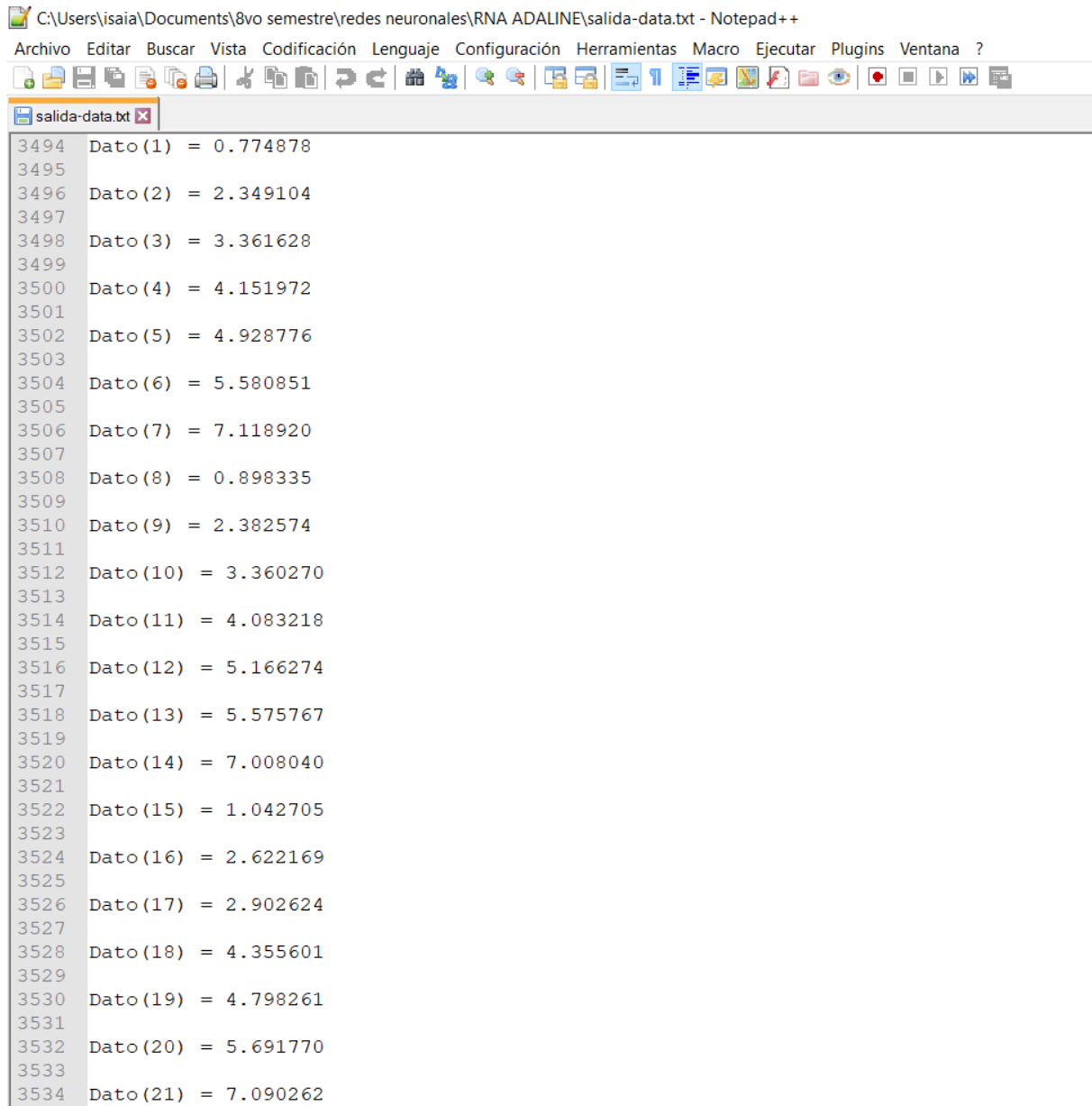
Figura 30: Gráficas de error acumulado, evolución de W y el bias

```

C:\Users\saiia\Documents\8vo semestre\redes neuronales\RNA ADALINE\salida-data.txt - Notepad++
Archivo  Editar  Buscar  Vista  Codificación  Lenguaje  Configuración  Herramientas  Macro  Ejecutar  Plugins  Ventana  ?
salida-data.txt
3486 <<<Terminó por criterio de finalización eepoch<=Eepoch>>>
3487
3488 Valores finales
3489 W = 0.890903 1.408159 -0.156303 -1.348934 0.018302 0.233256 2.150992 0.323832 0.814285 0.019809 0.385927 -0.621060 -1.031050 -0.682733 0.031835
0.542838 0.351660 0.607112 0.078742 0.028461 0.147806 -0.602687 0.172991 0.823278 0.380446 0.344105 0.205102 0.064410 0.038105 0.271384 0.349801
0.199456 0.568824 0.245674 0.141149 0.616558 0.575496 1.153875 0.042137 0.598083 0.165649 0.378265 -0.243551 -0.231509 0.343140 0.411868 0.181463
0.153371 0.228977 -0.592673 0.041177 1.375272 0.970358 0.799295 -1.326486 0.512228 0.106653 -0.507298 -0.162701 0.427108 0.696093 0.756884 0.033460
0.689489
3490 Bias = 0.915991
3491

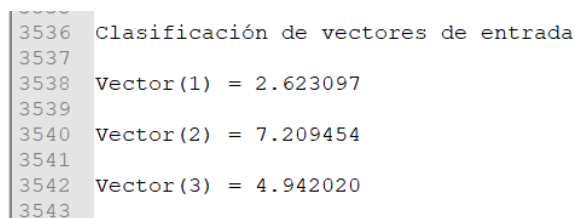
```

Figura 31: Criterio de finalización y valores finales de W y b



```
3494 Dato(1) = 0.774878
3495
3496 Dato(2) = 2.349104
3497
3498 Dato(3) = 3.361628
3499
3500 Dato(4) = 4.151972
3501
3502 Dato(5) = 4.928776
3503
3504 Dato(6) = 5.580851
3505
3506 Dato(7) = 7.118920
3507
3508 Dato(8) = 0.898335
3509
3510 Dato(9) = 2.382574
3511
3512 Dato(10) = 3.360270
3513
3514 Dato(11) = 4.083218
3515
3516 Dato(12) = 5.166274
3517
3518 Dato(13) = 5.575767
3519
3520 Dato(14) = 7.008040
3521
3522 Dato(15) = 1.042705
3523
3524 Dato(16) = 2.622169
3525
3526 Dato(17) = 2.902624
3527
3528 Dato(18) = 4.355601
3529
3530 Dato(19) = 4.798261
3531
3532 Dato(20) = 5.691770
3533
3534 Dato(21) = 7.090262
```

Figura 32: Resultados de evaluar el dataset con los valores finales de \mathbf{W} y \mathbf{b}



```
3536 Clasificación de vectores de entrada
3537
3538 Vector(1) = 2.623097
3539
3540 Vector(2) = 7.209454
3541
3542 Vector(3) = 4.942020
3543
```

Figura 33: Resultados de clasificar los vectores de entrada del usuario

Gráficamente los vectores ingresados a la red fueron los siguientes:

1	0	1	1	0	0	0	0
0	1	1	1	1	0	0	0
0	1	1	1	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	0
0	1	1	1	1	0	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	1

1	0	0	1	0	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	1
0	0	0	1	0	1	0	1
0	0	0	1	0	1	0	1
0	0	0	1	1	0	1	1
0	0	0	1	1	0	1	0

1	0	0	1	1	0	0	0
0	0	1	1	1	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	1

Figura 34: A desplazada a la izquierda, W desplazada a la derecha y letra O con ruido

Analizando los resultado obtenidos en 33 se puede notar que aún con poco ruido en los vectores de entrada la red no logra clasificar correctamente los valores ya que A pertenece a la clase 1, W pertenece a la clase 7 y O pertenece a la clase 3.

5. Conclusiones

Matlab es una gran herramienta para poder programar y comprender mejor el funcionamiento de las redes neuronales. En este caso se realizó la codificación de la red ADALINE para poder reconocer patrones como se vio en el libro. En general creo que realizar este tipo de prácticas ayuda a reforzar el conocimiento adquirido.

6. Referencias

Martin T. Hagan. (2014). Neural Network Design. Frisco, Texas