INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

NEURAL NETWORKS

PROF.: MARCO ANTONIO MORENO ARMENDÁRIZ
ALUMNO: ORTEGA VICTORIANO IVAN
No. DE LISTA: 29
GRUPO: 3CM2

PERCEPTRÓN MULTICAPA

Índice

INTRODUCCIÓN 3
MARCO TEÓRICO 4
Arquitectura del Perceptrón Multicapa 4
Early-stopping 5
RESULTADOS EXPERIMENTALES 6
Experimento 1: 6
Experimento 2: 13
Experimento 3: 20
DISCUSIÓN DE LOS RESULTADOS 27
CONCLUSIONES 28
REFERENCIAS 29
ANEXO 30
SUBCONJUNTOS DE ENTRENAMIENTO, PRUEBA Y VALIDACIÓN 30
EXPERIMENTO 1 30
EXPERIMENTO 2 31
EXPERIMENTO 3 32
CÓDIGOS 33
mlp.m 33
obtenerF.m 40
obtenerConjuntoDeEntrenamiento.m 40
obtenerConjuntoDePrueba.m 41
obtenerConjuntoDeValidacion.m 41

INTRODUCCIÓN

La generalización del algoritmo LMS (Least Mean Square), llamada *backpropagation* (propagación hacia atrás) puede ser utilizada para entrenar redes multicapa. Al igual que con la ley de aprendizaje LMS, backpropagation es un algoritmo aproximado de descenso más empinado, en el que el índice de rendimiento es el error cuadrático medio. La diferencia entre el algoritmo LMS y backpropagation es solo en la forma en que se calculan las derivadas. Para una red lineal de una sola capa, el error es una función lineal explícita de los pesos de la red, y sus derivadas con respecto a los pesos se pueden calcular fácilmente. En redes de múltiples capas con funciones de transferencia no lineales, la relación entre los pesos de la red y el error es más compleja. Para calcular las derivadas, necesitamos usar la regla de la cadena del cálculo. [1]

La regla de aprendizaje del perceptrón de Frank Rosenblatt y el algoritmo LMS de Bernard Widrow y Marcian Hoff fueron diseñados para entrenar redes de preceptrones de una sola capa. Sin embargo, las redes de una sola capa sufrieron de la desventaja de que solo eran capaces de resolver problemas de clasificación linealmente separables. Tanto Rosenblatt como Widrow estaban al tanto de estas limitaciones, y propusieron redes multicapa que pudieran superarlas, pero no fueron capaces de generalizar sus algoritmos para entrenar estas redes más poderosas. [2]

Aparentemente la primera descripción de un algoritmo para entrenar redes multicapa estaba contenido en la tesis de Paul Werbos en 1974. Esta tesis presentó el algoritmo en el contexto de redes generales, con redes neuronales como un caso especial, y no fue esparcido en la comunidad de redes neuronales. No fue hasta mediados de los 80's que el algoritmo backpropagation fue redescubierto y ampliamente publicado. Fue redescubierto independientemente por David Rumelhart, Geoffrey Hinton y Ronald Williams, David Parker, y Yann Le Cun. El algoritmo fue popularizado por su inclusión en el libro *Parallel Distributed Processing* (Procesamiento Paralelo Distribuido), el cual describió el trabajo del grupo de Procesamiento Paralelo Distribuido llevado por los psicólogos David Rumelhart y James McClelland. La publicación de este libro estimuló un torrente de investigación en el campo de las redes neuronales. El perceptrón multicapa, entrenado por el algoritmo backpropagation, es actualmente la red neuronal más ampliamente utilizada. [2]

MARCO TEÓRICO

Arquitectura del Perceptrón Multicapa

La arquitectura de Perceptrón multicapa se caracteriza porque tiene sus neuronas agrupadas en capas de diferentes niveles. Cada una de las capas está formada por un conjunto de neuronas y se distinguen tres tipos de capas diferentes: la capa de entrada (que es considerada como capa oculta), las capas ocultas y la capa de salida. Las neuronas de la capa de entrada no actúan como neuronas propiamente dichas, sino que se encargan únicamente de recibir las señales o patrones del exterior y propagar dichas señales a todas las neuronas de la siguiente capa. La última capa actúa como salida de la red, proporcionando al exterior la respuesta de la red para cada uno de los patrones de entrada. Las neuronas de las capas ocultas realizan un procesamiento no lineal de los patrones recibidos. [3]

Las conexiones del Perceptrón multicapa siempre están dirigidas hacia adelante, es decir, las neuronas de una capa se conectan con las neuronas de la siguiente capa, de ahí que reciban también el nombre de redes alimentadas hacia adelante o redes feedforward. Las conexiones entre las neuronas de la red llevan también asociado un bias, que el caso del Perceptrón multicapa suele tratarse como una conexión más a la neurona, cuya entrada es constante e igual a 1. Generalmente, todas las neuronas de una capa están conectadas a todas las neuronas de la siguiente capa. Se dice entonces que existe conectividad total o que la red está totalmente conectada.

Cuando se aborda un problema con el Perceptrón multicapa, en la mayoría de los casos se parte de una arquitectura totalmente conectada, es decir, todas las neuronas de una capa están conectadas a todas las neuronas de la siguiente capa. No es posible demostrar que, si se utilizan arquitecturas en las que se eliminan o se añaden conexiones de una capa a capas no inmediatamente posteriores, se puedan obtener mejores resultados. Sin embargo, en ocasiones, y debido fundamentalmente a la naturaleza del problema, se pueden encontrar redes multicapa con estas características en sus conexiones. [3]

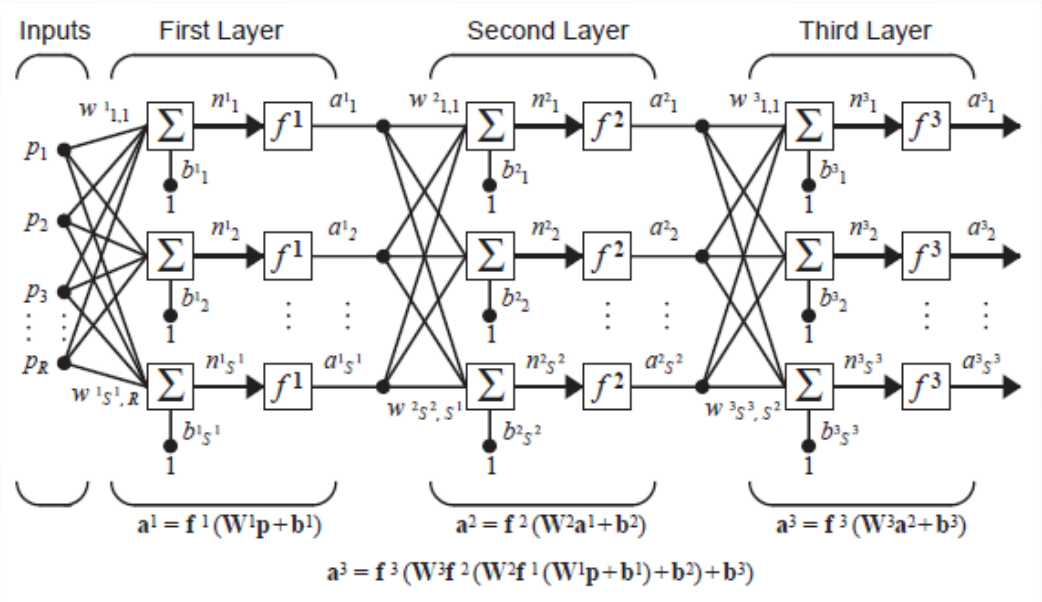

Figura 1. Arquitectura de un Perceptrón de tres capas. [4]

Early-stopping

En aprendizaje automático, el sobreajuste o sobre-entrenamiento (en inglés, overfitting) es el efecto de sobreentrenar un algoritmo de aprendizaje con unos ciertos datos para los que se conoce el resultado deseado. [5]

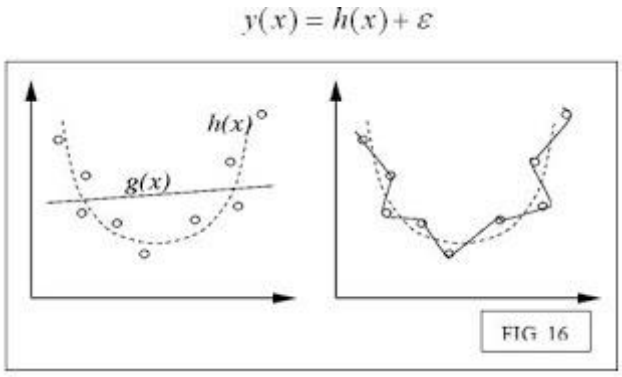

Figura 2. Ejemplo de sobre-entrenamiento.

Una manera de acometer el problema de sobre-entrenamiento (overfitting) es extraer un subconjunto de muestras del conjunto de entrenamiento (nótese que el conjunto de test se ha extraído previamente) y utilizarlo de manera auxiliar durante el entrenamiento. Este subconjunto recibe el nombre de conjunto de validación. La función que desempeña el conjunto de validación es evaluar el error de la red tras cada época (o tras cada cierto número de épocas) y determinar el momento en que ´este empieza a aumentar. Ya que el conjunto de validación se deja al margen durante el entrenamiento, el error cometido sobre él es un buen indicativo del error que la red cometerá sobre el conjunto de test. En consecuencia, se procederá a detener el entrenamiento en el momento en que el error de validación aumente y se conservaran los valores de los pesos de la época anterior. Este criterio de parada se denomina early-stopping. [3]

RESULTADOS EXPERIMENTALES

Experimento 1:

Se utilizará a continuación el siguiente conjunto de entrenamiento que consta de 101 datos, el cuál corresponde a la función $f(n) = \sin\left(\frac{\pi n}{2}\right)$, en el intervalo [-2,2].

n	f(n)	n	f(n)	n	f(n)	n	f(n)
-2.000000	1.000000	-0.960000	0.001973	0.080000	1.125333	1.120000	1.982287
-1.960000	0.937209	-0.920000	0.007885	0.120000	1.187381	1.160000	1.968583
-1.920000	0.874667	-0.880000	0.017713	0.160000	1.248690	1.200000	1.951057
-1.880000	0.812619	-0.840000	0.031417	0.200000	1.309017	1.240000	1.929776
-1.840000	0.751310	-0.800000	0.048943	0.240000	1.368125	1.280000	1.904827
-1.800000	0.690983	-0.760000	0.070224	0.280000	1.425779	1.320000	1.876307
-1.760000	0.631875	-0.720000	0.095173	0.320000	1.481754	1.360000	1.844328
-1.720000	0.574221	-0.680000	0.123693	0.360000	1.535827	1.400000	1.809017
-1.680000	0.518246	-0.640000	0.155672	0.400000	1.587785	1.440000	1.770513
-1.640000	0.464173	-0.600000	0.190983	0.440000	1.637424	1.480000	1.728969
-1.600000	0.412215	-0.560000	0.229487	0.480000	1.684547	1.520000	1.684547
-1.560000	0.362576	-0.520000	0.271031	0.520000	1.728969	1.560000	1.637424
-1.520000	0.315453	-0.480000	0.315453	0.560000	1.770513	1.600000	1.587785
-1.480000	0.271031	-0.440000	0.362576	0.600000	1.809017	1.640000	1.535827
-1.440000	0.229487	-0.400000	0.412215	0.640000	1.844328	1.680000	1.481754
-1.400000	0.190983	-0.360000	0.464173	0.680000	1.876307	1.720000	1.425779
-1.360000	0.155672	-0.320000	0.518246	0.720000	1.904827	1.760000	1.368125
-1.320000	0.123693	-0.280000	0.574221	0.760000	1.929776	1.800000	1.309017
-1.280000	0.095173	-0.240000	0.631875	0.800000	1.951057	1.840000	1.248690
-1.240000	0.070224	-0.200000	0.690983	0.840000	1.968583	1.880000	1.187381
-1.200000	0.048943	-0.160000	0.751310	0.880000	1.982287	1.920000	1.125333
-1.160000	0.031417	-0.120000	0.812619	0.920000	1.992115	1.960000	1.062791
-1.120000	0.017713	-0.080000	0.874667	0.960000	1.998027	2.000000	1.000000
-1.080000	0.007885	-0.040000	0.937209	1.000000	2.000000		
-1.040000	0.001973	0.000000	1.000000	1.040000	1.998027		
-1.000000	0.000000	0.040000	1.062791	1.080000	1.992115		

Ya una vez abierto el programa en Matlab, lo ejecutamos y a continuación nos solicitará los siguientes datos en ese orden:

- El archivo que contiene los valores de entrada: "*.txt" (sin la extensión .txt).
- El archivo que contiene los valores de target: "*.txt" (sin la extensión .txt).
- La arquitectura del M.L.P.
- Las funciones de cada capa.
- El valor del factor de aprendizaje.
- El valor de iteraciones (épocas) máximas: itmax.
- El valor de itval (cada cuanto se hará una iteración de validación).
- El valor de numval (número máximo de incrementos del error de validación).
- El valor mínimo del error de época (eit).
- La configuración de subconjuntos a trabajar (80-10-10 ó 70-15-15).

Para este ejemplo se usarán los respectivos valores de acuerdo con el orden:

- Archivo de entradas: input
- Archivo de valores deseados: target
- Arquitectura del M.L.P.: 1 3 1
- Funciones de las capas del M.L.P.: 2 1 (Se usará la función logsig(n) en la capa oculta)
- Alfa: 0.1
- Itmax: 20000
- Itval: 1000
- Numval: 3
- Eit: 0.0000000000001
- Configuración: 70-15-15 (Para el programa es la opción 2)

Como se muestra en las capturas del programa:

```
Ingresa el nombre del archivo que contiene los valores de entrada "*.txt" (sin extension): input
Ingresa el nombre del archivo que contiene los valores target "*.txt" (sin extension): target

Se trabajara el siguiente rango, de acuerdo al archivo:
[-2.000000, 2.000000], con un incremento de 0.040000
Se trabajara con 101 datos, de acuerdo al archivo.


Ingresa la arquitectura del M.L.P.: 1 3 1
Para las funciones de activacion se tienen las siguientes:
1) purelin()
2) logsig()
3) tansig()

Ingresa las funciones de las capas de la red separadas por un espacio: 2 1
La arquitectura del M.L.P. es:
     1     3     1

     2     1

Ingresa el valor del factor de aprendizaje(alfa): 0.1
Ingresa el numero de iteraciones maximas de la red(itmax): 20000
¿Cada cuanto se hara una iteracion de validacion? (itval): 1000
Numero maximo de incrementos consecutivos del error de validacion (numval): 3
Ingrese 1 valor minimo del error en una epoca (eit): 0.0000000000001
Seleccione una de las siguientes configuraciones a trabajar:
1) 80-10-10
2) 70-15-15
Ingrese su seleccion: 2
Se usaran los siguientes tamanios en los subconjuntos:
Conjunto de entrenamiento: 71 elementos
Conjunto de validacion: 15 elementos
Conjunto de prueba: 15 elementos
```

Figura 3. Se muestran los datos solicitados e ingresados en el programa.

Posteriormente se le muestran al usuario la separación de los conjuntos de entrenamiento (Ir al **Anexo 1.1** para ver la selección realizada para este experimento), además de los valores de pesos y bias iniciales de cada capa como se muestra en la figura 4.

```
Valores iniciales de las matrices:
W_1 =
   -0.3192
    0.1705
   -0.5524

b_1 =
    0.5025
   -0.4898
    0.0119

W_2 =
    0.3982    0.7818    0.9186

b_2 =
    0.0944
```

Figura 4. Se muestran las matrices de pesos y los vectores bias iniciales de cada capa.

Una vez establecidos dichos valores, se nos solicita que presionemos la tecla ENTER para iniciar el aprendizaje. Hecho esto, iniciará el proceso de aprendizaje, donde la red ajustará sus valores de pesos y bias utilizando el algoritmo backpropagation. Durante este proceso, al usuario únicamente se le mostrará el valor de *countval*, es decir, los incrementos consecutivos en el error de validación hasta llegar a una condición de paro. Como se muestra en la figura 5.

```
Presiona ENTER para comenzar el aprendizaje...
Count val = 1
Count val = 0
Count val = 1
```

Figura 5. Se le muestra al usuario en consola el valor de *countval.*

Las condiciones de paro establecidas fueron que se llegara a las 100000 iteraciones, que el error de aprendizaje (Eap), fuera menor a 0.000000000001, o que se llegara a 5 incrementos consecutivos en el error de validación (Early stopping). Llegado a la condición de paro de la red, se le muestra al usuario los valores finales del error de validación *Eval*, el error de aprendizaje *Eap* y el error de prueba *Ep*, además de si se detuvo con early stopping, se llegó a itmax o fue un aprendizaje exitoso, como se muestra en la figura 6.

```
Presiona ENTER para comenzar el aprendizaje...
Count val = 1
Count val = 0
Count val = 1
Count val = 2
Count val = 3
Early stopping en iteracion 5000
Eap = -0.000000
Eval = -0.000011
Ep = 0.000007
```

Figura 6. Se le muestra al usuario los valores finales de Eap, Eval y Ep.

Y por último se le muestra al usuario las siguientes gráficas:

- Comparación de la evaluación del conjunto de prueba (salida de la red) vs. los targets. (figura 7)
- Comparación de la evaluación del conjunto de prueba (salida de la red) vs. los targets. (figura 8)

- Evolución del error de aprendizaje junto con el error de validación. (figura 9)
- Evolución de los pesos (1 gráfica por capa). (figuras 10 y11)
- Evolución del bias (1 gráfica por capa). (figuras 12 y 13)

Para este experimento se tuvieron los siguientes resultados.

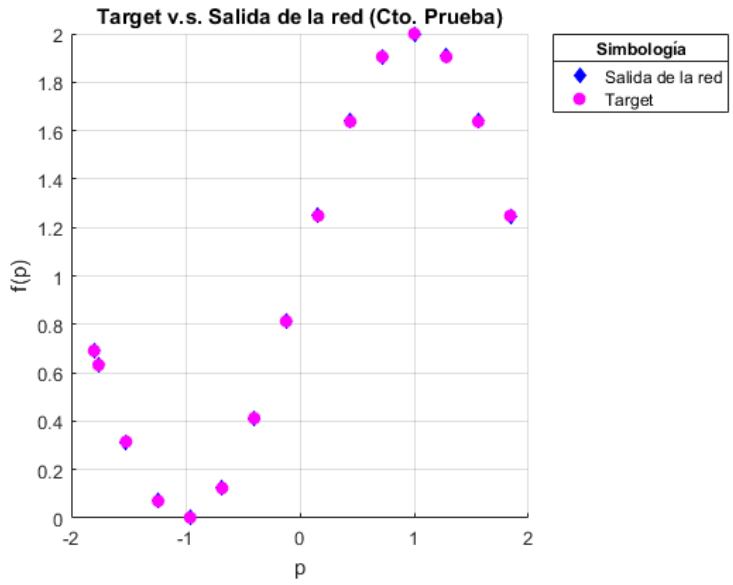

Figura 7. Target vs Salida de la red del conjunto de prueba (Se puede apreciar que el error fue mínimo).

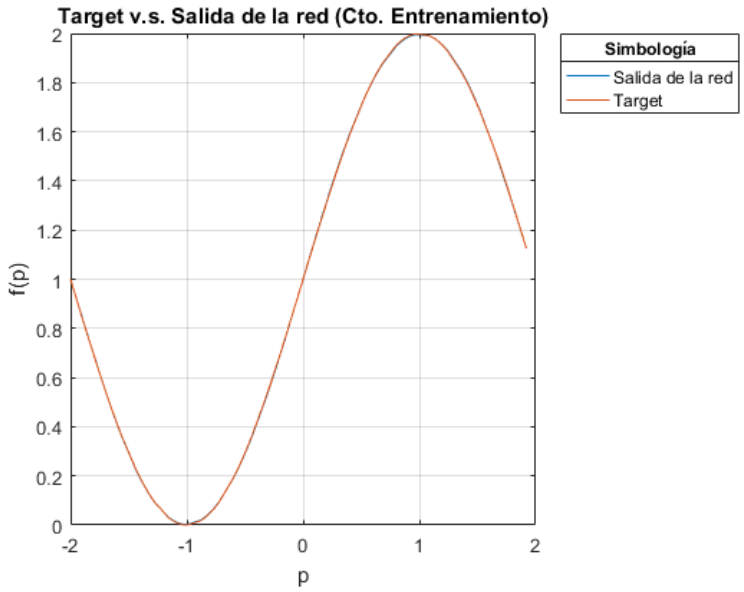

Figura 8. Target vs Salida de la red del conjunto de entrenamiento (Se puede apreciar que el error fue mínimo).

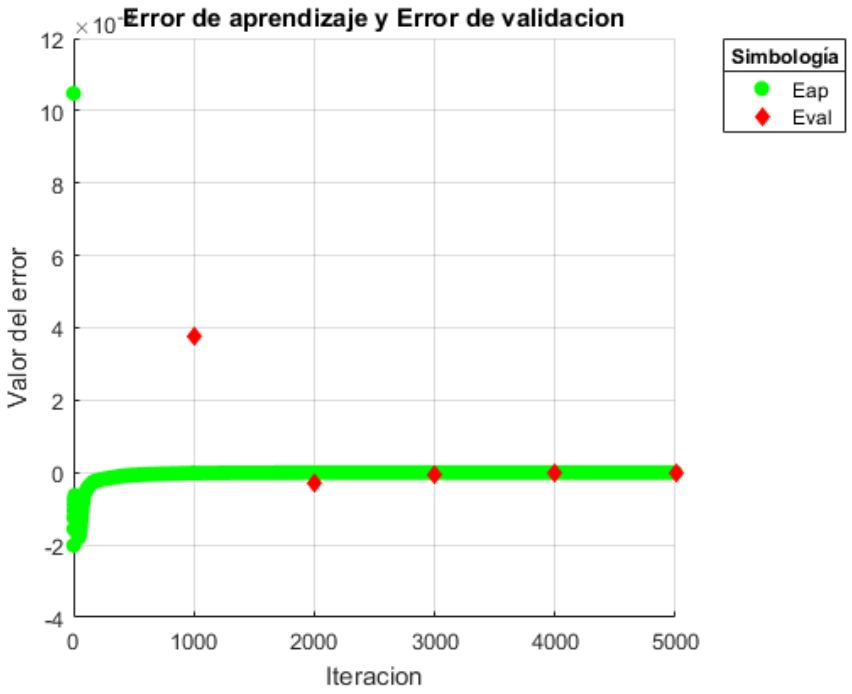

Figura 9. Evolución del error de aprendizaje y el error de validación.

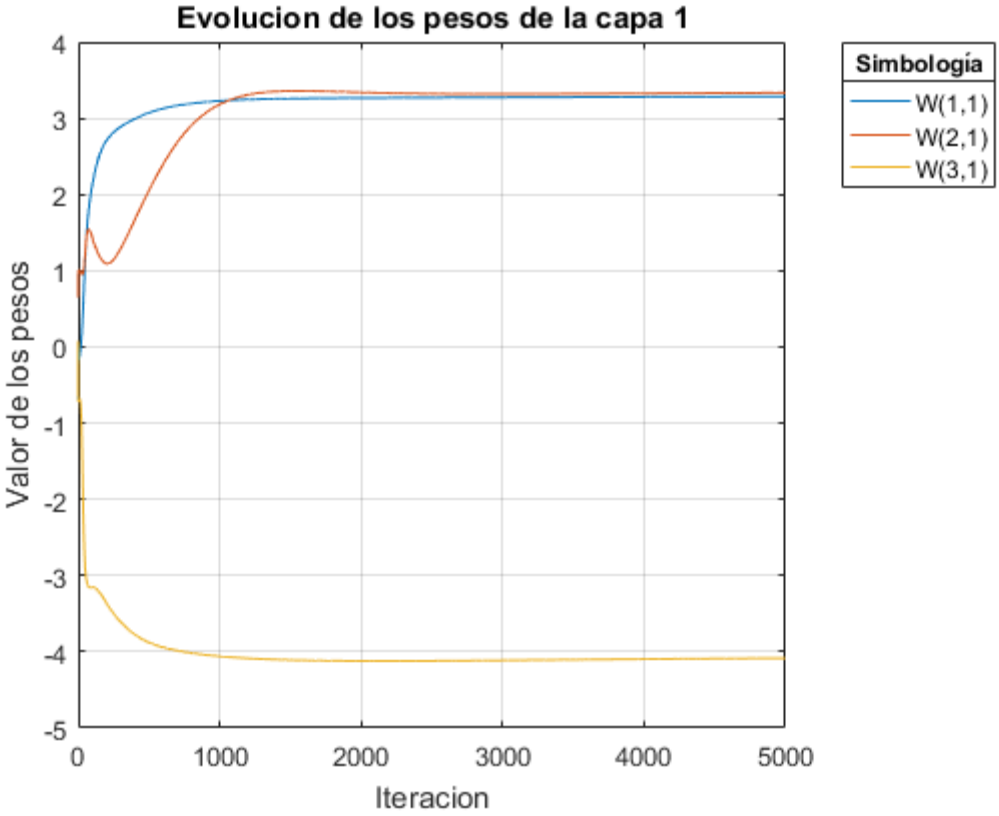

Figura 10. Evolución de los pesos de la capa 1.

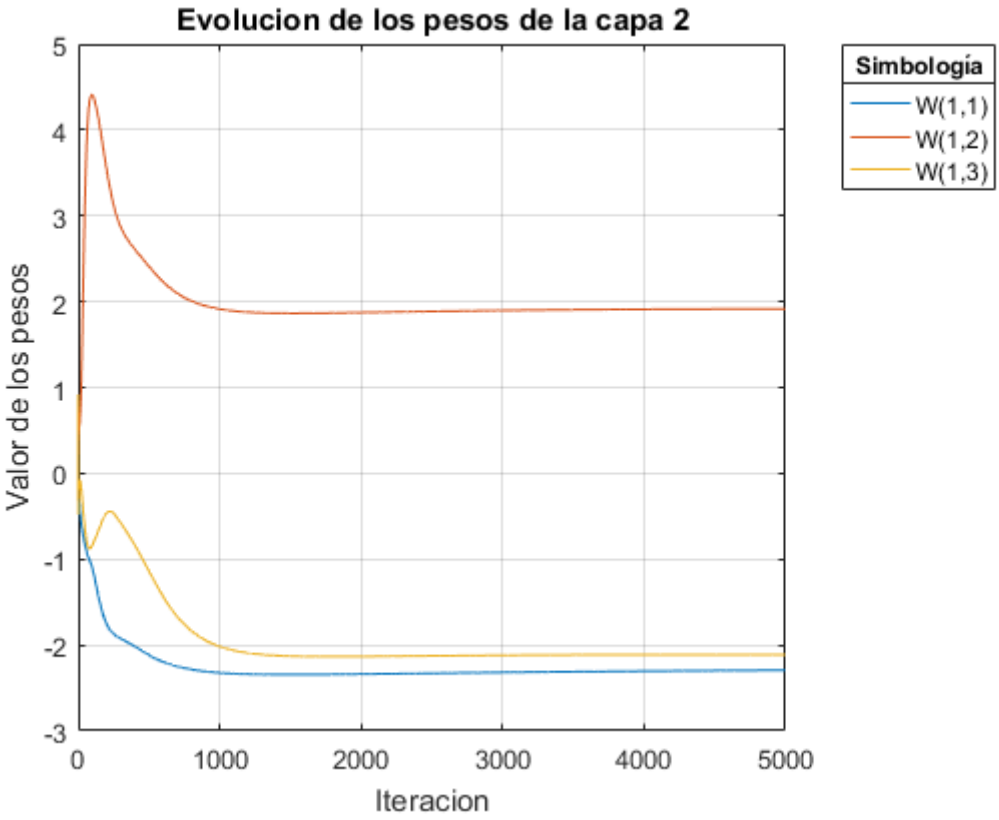

Figura 11. Evolución de los pesos de la capa 2.

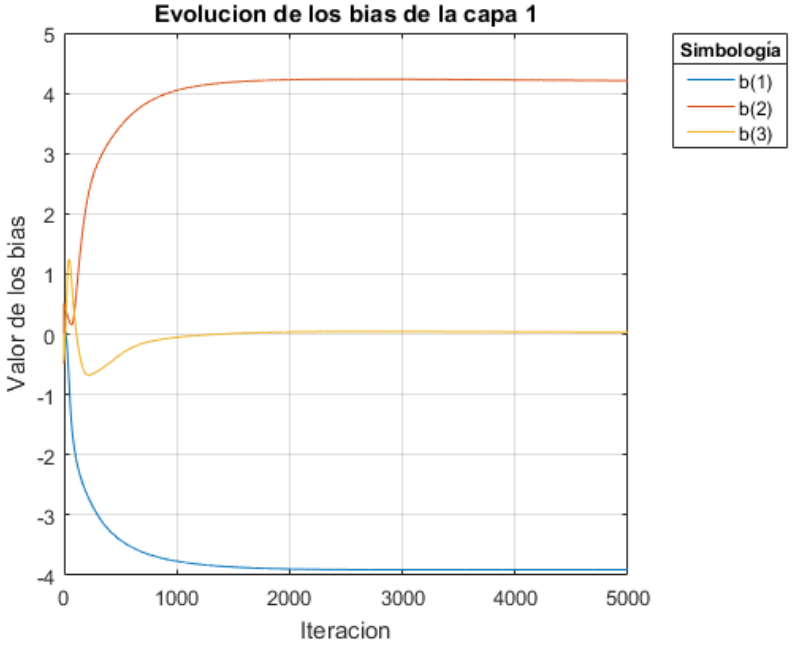

Figura 12. Evolución del bias de la capa 1.

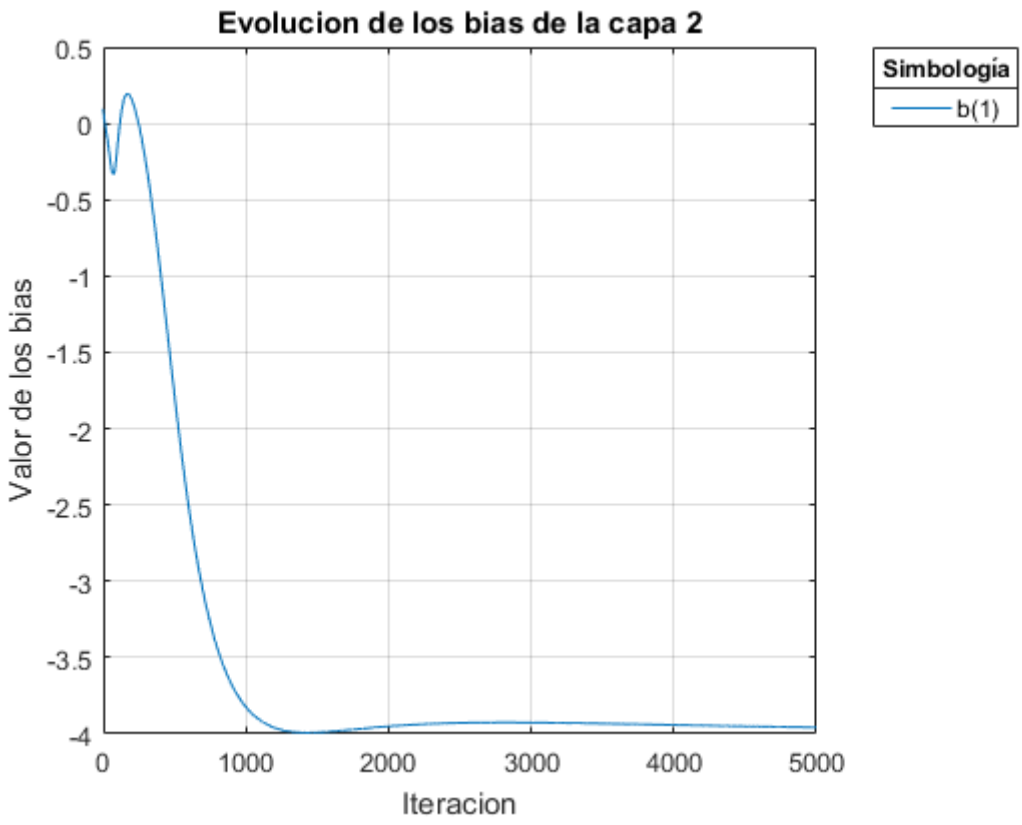

Figura 13. Evolución del bias de la capa 2.

Finalmente los valores de pesos y bias finales se guardan en una carpeta organizada por el número de capas, donde se encontrarán los archivos "*.txt" como se muestra en la figura 14:

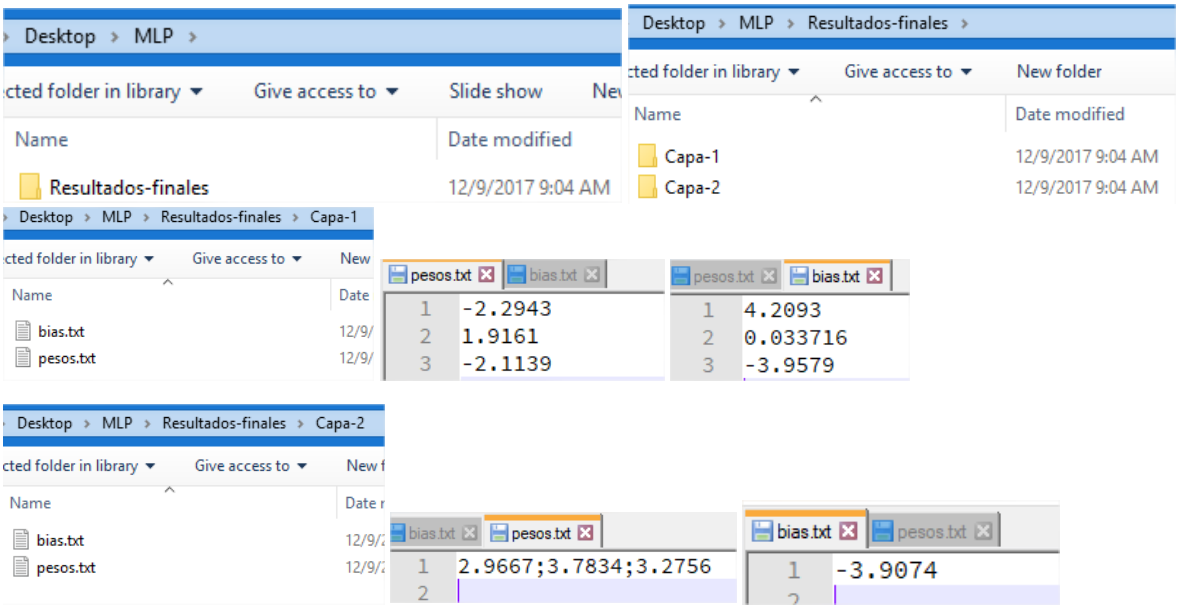

Figura 14. Carpeta contenedora y los respectivos archivos y su contenido.

Experimento 2:

Se utilizará a continuación el siguiente conjunto de entrenamiento que consta de 103 datos, el cuál corresponde a la función $f(n) = 27\mathrm{n}^4 - 60n^3 + 39n^2 - 6n$, en el intervalo [0,1]. [6]

n	f(n)	n	f(n)	n	f(n)	n	f(n)
0.000000	0.000000	0.260000	0.145224	0.520000	0.963256	0.780000	0.568545
0.010000	-0.056160	0.270000	0.185609	0.530000	0.972910	0.790000	0.534082
0.020000	-0.104876	0.280000	0.226437	0.540000	0.980385	0.800000	0.499200
0.030000	-0.146498	0.290000	0.267526	0.550000	0.985669	0.810000	0.464055
0.040000	-0.181371	0.300000	0.308700	0.560000	0.988754	0.820000	0.428808
0.050000	-0.209831	0.310000	0.349791	0.570000	0.989640	0.830000	0.393627
0.060000	-0.232210	0.320000	0.390636	0.580000	0.988334	0.840000	0.358687
0.070000	-0.248832	0.330000	0.431079	0.590000	0.984847	0.850000	0.324169
0.080000	-0.260014	0.340000	0.470971	0.600000	0.979200	0.860000	0.290260
0.090000	-0.266069	0.350000	0.510169	0.610000	0.971417	0.870000	0.257155
0.100000	-0.267300	0.360000	0.548536	0.620000	0.961531	0.880000	0.225055
0.110000	-0.264007	0.370000	0.585943	0.630000	0.949579	0.890000	0.194165
0.120000	-0.256481	0.380000	0.622267	0.640000	0.935608	0.900000	0.164700
0.130000	-0.245009	0.390000	0.657389	0.650000	0.919669	0.910000	0.136879
0.140000	-0.229868	0.400000	0.691200	0.660000	0.901819	0.920000	0.110930
0.150000	-0.211331	0.410000	0.723595	0.670000	0.882123	0.930000	0.087084
0.160000	-0.189665	0.420000	0.754478	0.680000	0.860652	0.940000	0.065582
0.170000	-0.165129	0.430000	0.783756	0.690000	0.837483	0.950000	0.046669
0.180000	-0.137976	0.440000	0.811346	0.700000	0.812700	0.960000	0.030597
0.190000	-0.108453	0.450000	0.837169	0.710000	0.786394	0.970000	0.017626
0.200000	-0.076800	0.460000	0.861153	0.720000	0.758661	0.980000	0.008020
0.210000	-0.043250	0.470000	0.883234	0.730000	0.729605	0.990000	0.002052
0.220000	-0.008031	0.480000	0.903352	0.740000	0.699336	1.000000	-0.000000
0.230000	0.028637	0.490000	0.921456	0.750000	0.667969	1.010000	0.002148
0.240000	0.066540	0.500000	0.937500	0.760000	0.635628	1.020000	0.008788
0.250000	0.105469	0.510000	0.951444	0.770000	0.602441		

Para este ejemplo se usarán los respectivos valores de acuerdo con el orden establecido en el experimento anterior:

- Archivo de entradas: input
- Archivo de valores deseados: target
- Arquitectura del M.L.P.: 1 3 1
- Funciones de las capas del M.L.P.: 3 1 (Se usará la función tansig(n) en la capa oculta)
- Alfa: 0.1
- Itmax: 30000
- Itval: 5000
- Numval: 3
- Eit: 0.000000000001
- Configuración: 70-15-15 (Para el programa es la opción 2)

Como se muestra en las capturas del programa en la figura 15:

```
Ingresa el nombre del archivo que contiene los valores de entrada "*.txt" (sin extension): input
Ingresa el nombre del archivo que contiene los valores target "*.txt" (sin extension): target

Se trabajara el siguiente rango, de acuerdo al archivo:
[0.000000, 1.020000], con un incremento de 0.010000
Se trabajara con 103 datos, de acuerdo al archivo.


Ingresa la arquitectura del M.L.P.: 1 3 1
Para las funciones de activacion se tienen las siguientes:
1) purelin()
2) logsig()
3) tansig()

Ingresa las funciones de las capas de la red separadas por un espacio: 3 1
La arquitectura del M.L.P. es:
     1     3     1

     3     1

Ingresa el valor del factor de aprendizaje(alfa): 0.1
Ingresa el numero de iteraciones maximas de la red(itmax): 30000
¿Cada cuanto se hara una iteracion de validacion? (itval): 5000
Numero maximo de incrementos consecutivos del error de validacion (numval): 3
Ingrese 1 valor minimo del error en una epoca (eit): 0.0000000000001
Seleccione una de las siguientes configuraciones a trabajar:
1) 80-10-10
2) 70-15-15
Ingrese su seleccion: 2
Se usaran los siguientes tamanios en los subconjuntos:
Conjunto de entrenamiento: 73 elementos
Conjunto de validacion: 15 elementos
Conjunto de prueba: 15 elementos
```

Figura 15. Se muestran los datos solicitados e ingresados en el programa.

Para ver el la selección de los subconjuntos para el M.L.P., ir al ANEXO 1.2.

A continuación, se muestran las matrices de pesos y los vectores bias iniciales en la figura 16.

```
Valores iniciales de las matrices:
W_1 =
   -0.6848
    0.9412
    0.9143

b_1 =
   -0.0292
    0.6006
   -0.7162

W_2 =
   -0.1565    0.8315    0.5844

b_2 =
    0.9190
```

Figura 16. Se muestran las matrices de pesos y los vectores bias iniciales de cada capa.

S le muestra al usuario los valores finales del error de validación *Eval*, el error de aprendizaje *Eap* y el error de prueba *Ep*, además de si se detuvo con early stopping, llegó a itmax o fue un aprendizaje exitoso, como se muestra en la figura 17.

```
Presiona ENTER para comenzar el aprendizaje...
Count val = 1
Count val = 0
Count val = 1
Count val = 2
Count val = 0
Count val = 1
Se llego a itmax.
Eap = -0.000000
Eval = -0.001563
Ep = -0.001616
```

Figura 17. Se le muestra al usuario los valores finales de Eap, Eval y Ep.

Y por último se le muestra al usuario las siguientes gráficas:

- Comparación de la evaluación del conjunto de prueba (salida de la red) vs. los targets. (figura 18)
- Comparación de la evaluación del conjunto de entrenamiento (salida de la red) vs. los targets. (figura 19)
- Evolución del error de aprendizaje junto con el error de validación. (figura 20)
- Evolución de los pesos (1 gráfica por capa). (figuras 21 y22)
- Evolución del bias (1 gráfica por capa). (figuras 23 y 24)

Para este experimento se tuvieron los siguientes resultados.

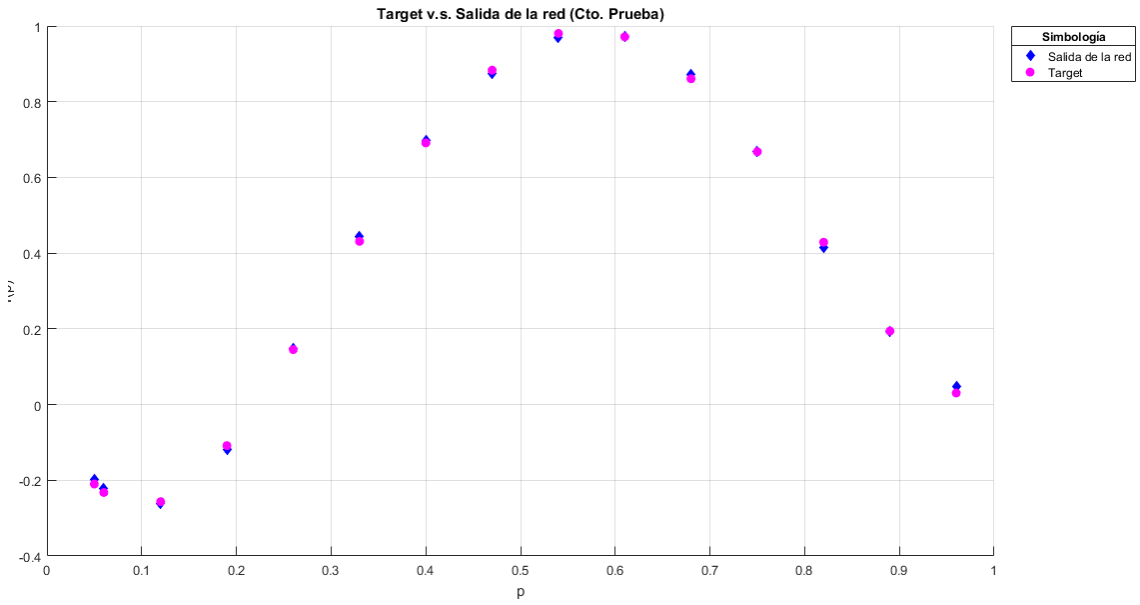

Figura 18. Target vs Salida de la red del conjunto de prueba (Se puede apreciar que el error fue mínimo).

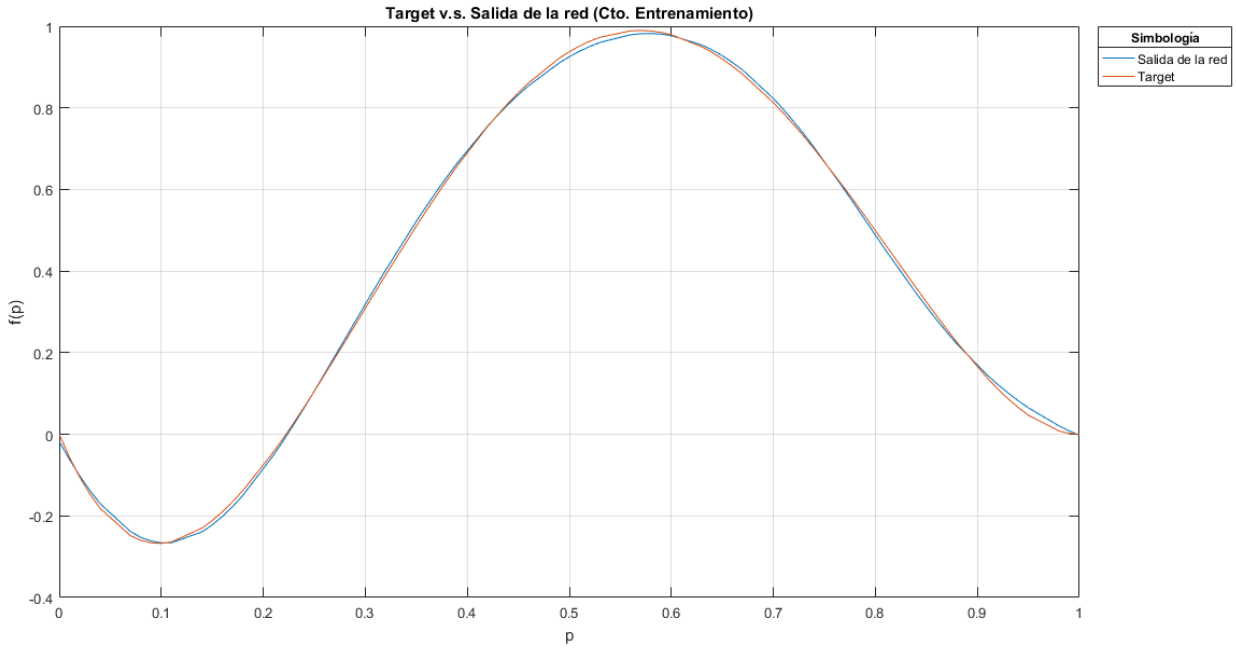

Figura 19. Target vs Salida de la red del conjunto de entrenamiento (Se puede apreciar que el error fue mínimo).

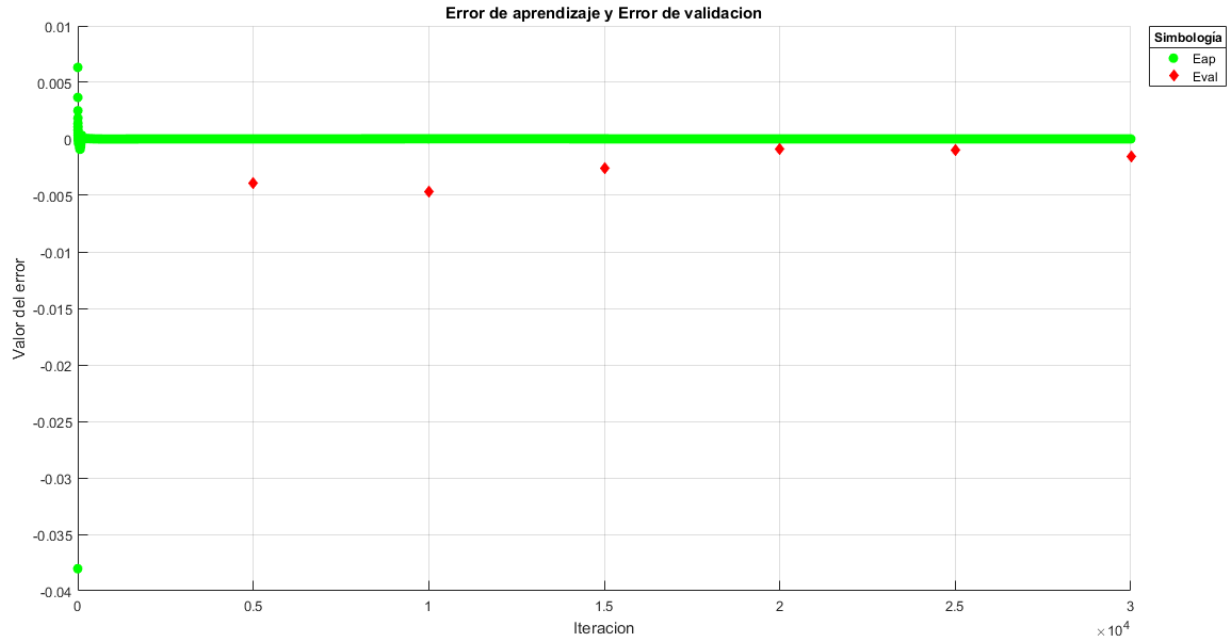

Figura 20. Evolución del error de aprendizaje y el error de validación.

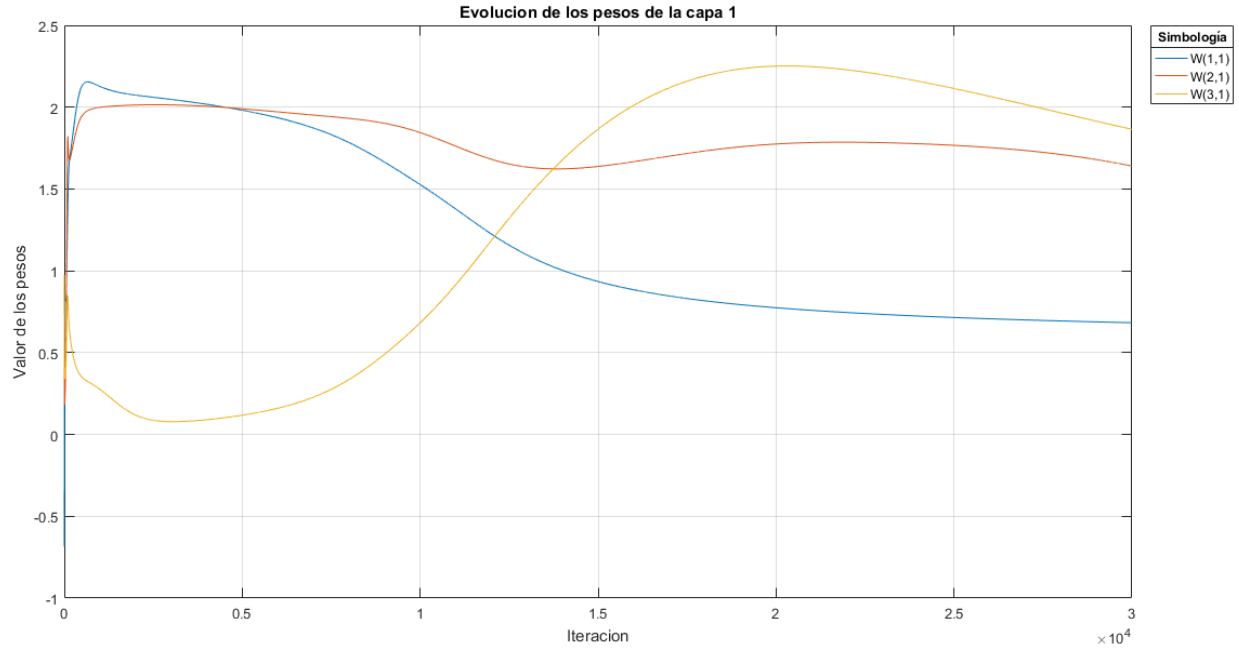

Figura 21. Evolución de los pesos de la capa 1.

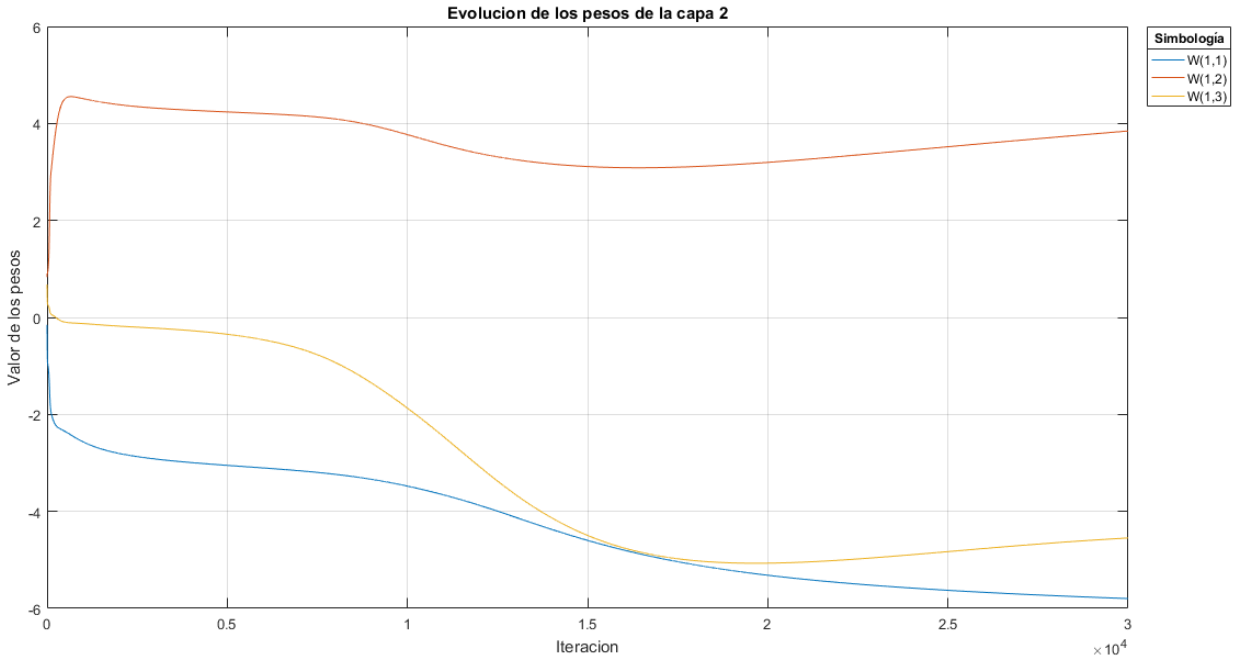

Figura 22. Evolución de los pesos de la capa 2.

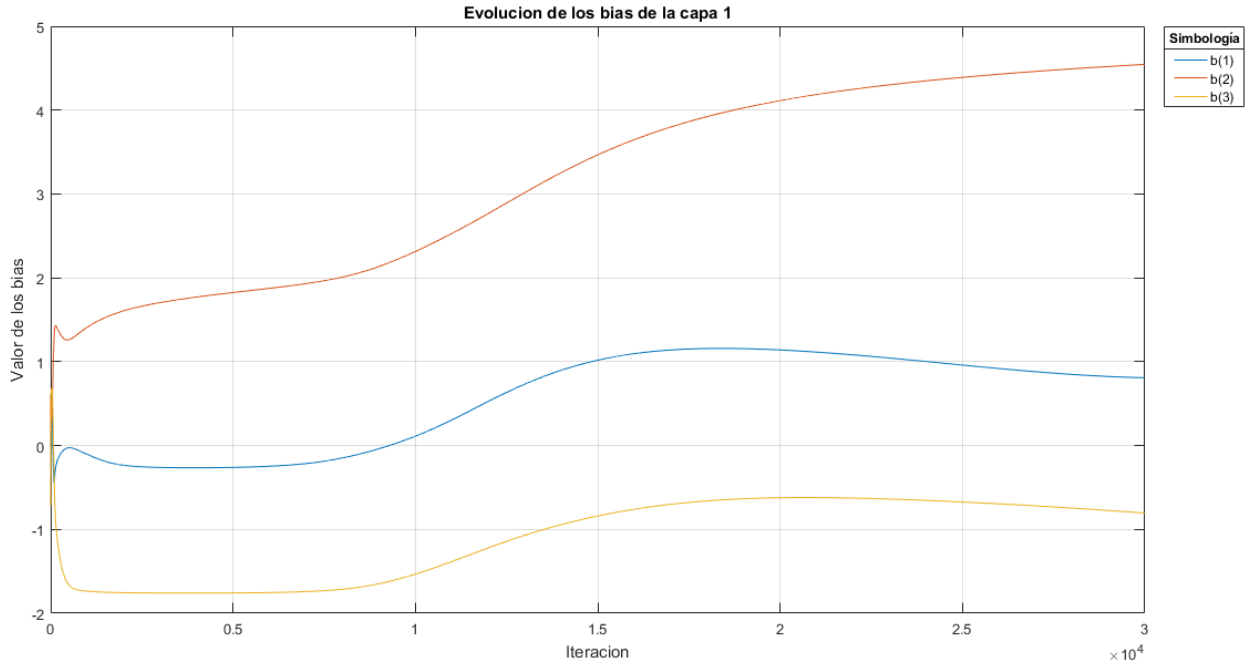

Figura 23. Evolución del bias de la capa 1.

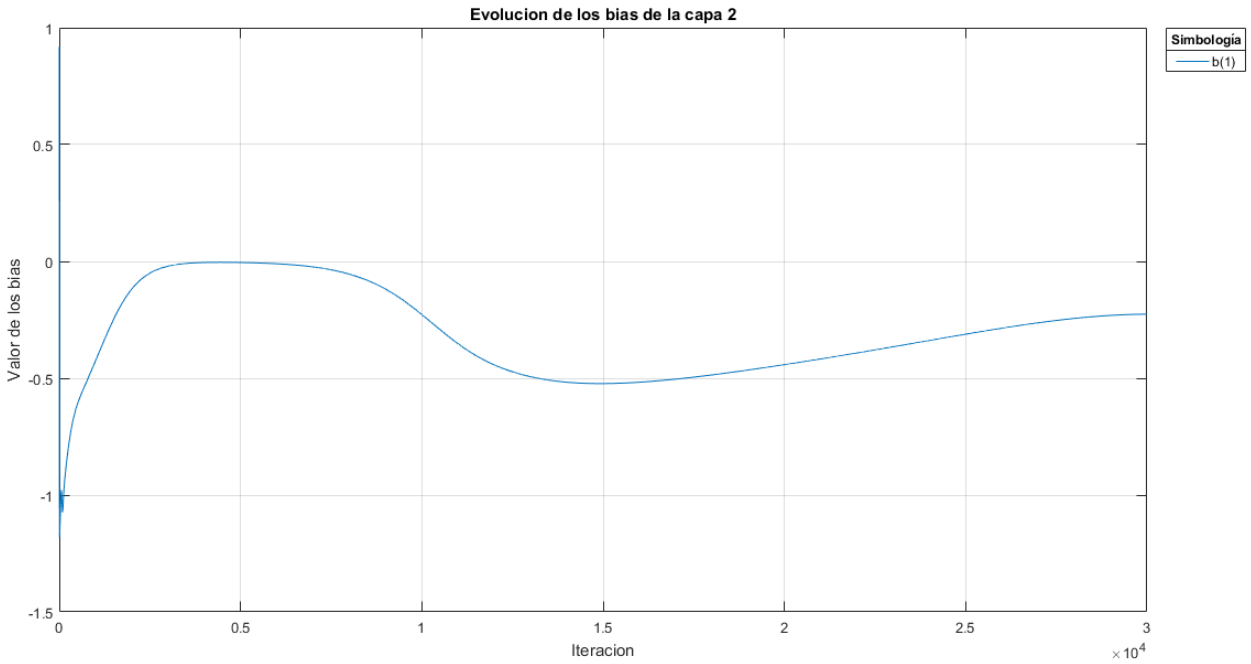

Figura 24. Evolución del bias de la capa 2.

Finalmente verificamos el contenido de los archivos que contienen los resultados finales como se muestra en la figura 25:

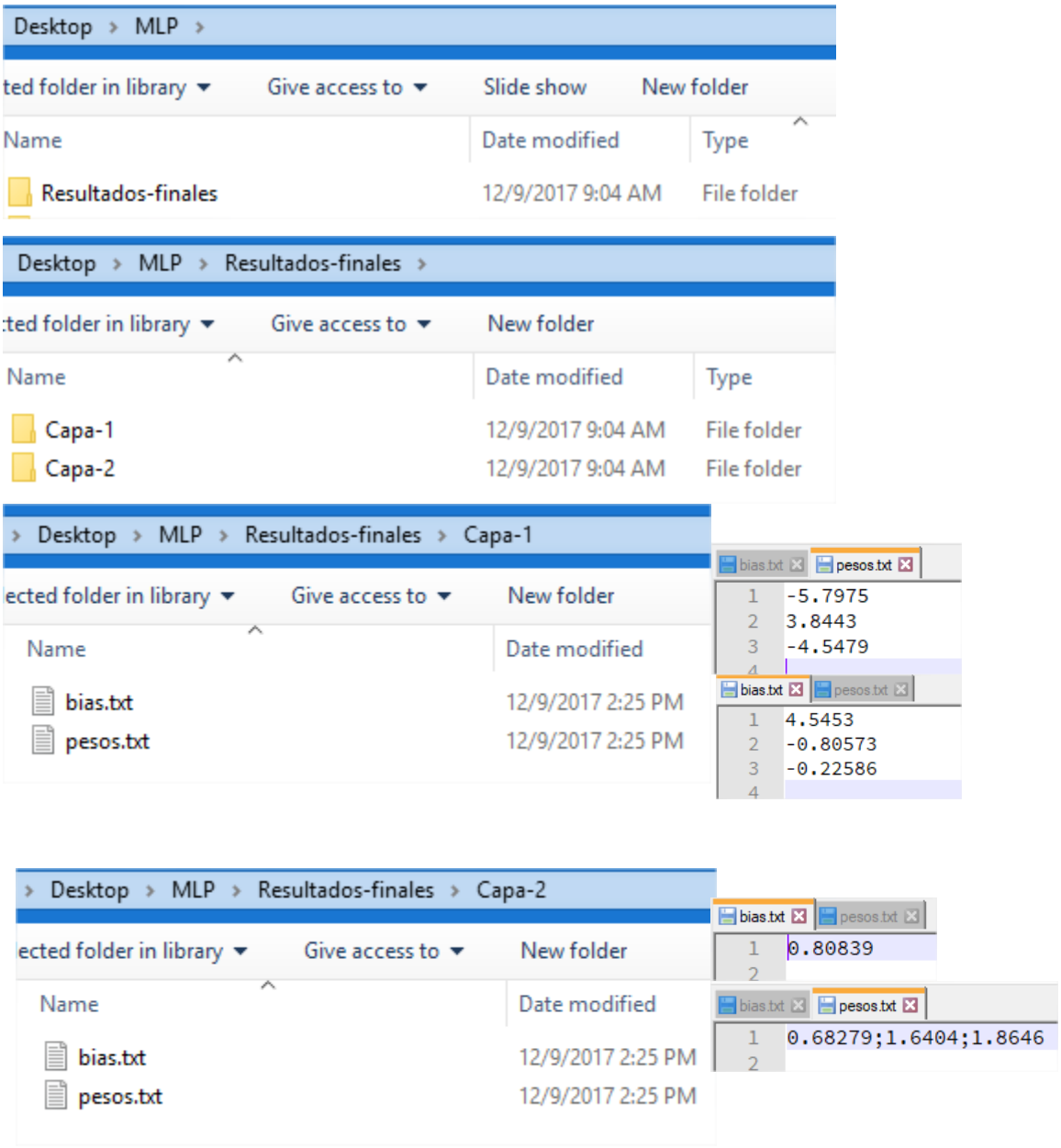

Figura 25. Carpeta contenedora y los respectivos archivos y su contenido.

Experimento 3:

Se utilizará a continuación el siguiente conjunto de entrenamiento que consta de 91 datos, el cuál corresponde a la función: $f(n) = \frac{n}{1+n^2}$ en el intervalo [-3,3]. [6]

n	f(n)
-3.000000	-0.300000
-2.950000	-0.304045
-2.900000	-0.308183
-2.850000	-0.312414
-2.800000	-0.316742
-2.750000	-0.321168
-2.700000	-0.325694
-2.650000	-0.330321
-2.600000	-0.335052
-2.550000	-0.339887
-2.500000	-0.344828
-2.450000	-0.349875
-2.400000	-0.355030
-2.350000	-0.360291
-2.300000	-0.365660
-2.250000	-0.371134
-2.200000	-0.376712
-2.150000	-0.382392
-2.100000	-0.388170
-2.050000	-0.394041
-2.000000	-0.400000
-1.950000	-0.406039
-1.900000	-0.412148
-1.850000	-0.418315
-1.800000	-0.424528
-1.750000	-0.430769
-1.700000	-0.437018
-1.650000	-0.443251
-1.600000	-0.449438
-1.550000	-0.455547
-1.500000	-0.461538
-1.450000	-0.467365
-1.400000	-0.472973

n	f(n)
-1.350000	-0.478299
-1.300000	-0.483271
-1.250000	-0.487805
-1.200000	-0.491803
-1.150000	-0.495156
-1.100000	-0.497738
-1.050000	-0.499405
-1.000000	-0.500000
-0.950000	-0.499343
-0.900000	-0.497238
-0.850000	-0.493469
-0.800000	-0.487805
-0.750000	-0.480000
-0.700000	-0.469799
-0.650000	-0.456942
-0.600000	-0.441176
-0.550000	-0.422265
-0.500000	-0.400000
-0.450000	-0.374220
-0.400000	-0.344828
-0.350000	-0.311804
-0.300000	-0.275229
-0.250000	-0.235294
-0.200000	-0.192308
-0.150000	-0.146699
-0.100000	-0.099010
-0.050000	-0.049875
-0.000000	-0.000000
0.050000	0.049875
0.100000	0.099010
0.150000	0.146699
0.200000	0.192308
0.250000	0.235294
0.300000	0.275229

n	f(n)
0.350000	0.311804
0.400000	0.344828
0.450000	0.374220
0.500000	0.400000
0.550000	0.422265
0.600000	0.441176
0.650000	0.456942
0.700000	0.469799
0.750000	0.480000
0.800000	0.487805
0.850000	0.493469
0.900000	0.497238
0.950000	0.499343
1.000000	0.500000
1.050000	0.499405
1.100000	0.497738
1.150000	0.495156
1.200000	0.491803
1.250000	0.487805
1.300000	0.483271
1.350000	0.478299
1.400000	0.472973
1.450000	0.467365
1.500000	0.461538
1.550000	0.455547
1.600000	0.449438
1.650000	0.443251
1.700000	0.437018
1.750000	0.430769
1.800000	0.424528
1.850000	0.418315
1.900000	0.412148
1.950000	0.406039
2.000000	0.400000

n	f(n)
2.050000	0.394041
2.100000	0.388170
2.150000	0.382392
2.200000	0.376712
2.250000	0.371134
2.300000	0.365660
2.350000	0.360291
2.400000	0.355030
2.450000	0.349875
2.500000	0.344828
2.550000	0.339887
2.600000	0.335052
2.650000	0.330321
2.700000	0.325694
2.750000	0.321168
2.800000	0.316742
2.850000	0.312414
2.900000	0.308183
2.950000	0.304045
3.000000	0.300000

Para este ejemplo se usarán los respectivos valores de acuerdo con el orden establecido en el experimento anterior:

- Archivo de entradas: input
- Archivo de valores deseados: target
- Arquitectura del M.L.P.: 1 2 1
- Funciones de las capas del M.L.P.: 3 1 (Se usará la función tansig(n) en la capa oculta)
- Alfa: 0.1
- Itmax: 20000
- Itval: 200
- Numval: 3
- Eit: 0.0000000000001
- Configuración: 80-10-10 (Para el programa es la opción 1)

Como se muestra en las capturas del programa en la figura 26:

```
Ingresa el nombre del archivo que contiene los valores de entrada "*.txt" (sin extension): input
Ingresa el nombre del archivo que contiene los valores target "*.txt" (sin extension): target

Se trabajara el siguiente rango, de acuerdo al archivo:
[-3.000000, 3.000000], con un incremento de 0.050000
Se trabajara con 121 datos, de acuerdo al archivo.


Ingresa la arquitectura del M.L.P.: 1 2 1
Para las funciones de activacion se tienen las siguientes:
1) purelin()
2) logsig()
3) tansig()

Ingresa las funciones de las capas de la red separadas por un espacio: 3 1
La arquitectura del M.L.P. es:
     1     2     1

     3     1

Ingresa el valor del factor de aprendizaje(alfa): 0.1
Ingresa el numero de iteraciones maximas de la red(itmax): 20000
¿Cada cuanto se hara una iteracion de validacion? (itval): 200
Numero maximo de incrementos consecutivos del error de validacion (numval): 3
Ingrese 1 valor minimo del error en una epoca (eit): 0.0000000000001
```

```
Seleccione una de las siguientes configuraciones a trabajar:
1) 80-10-10
2) 70-15-15
Ingrese su seleccion: 1
Se usaran los siguientes tamanios en los subconjuntos:
Conjunto de entrenamiento: 73 elementos
Conjunto de validacion: 24 elementos
Conjunto de prueba: 24 elementos
```

Figura 26. Se muestran los datos solicitados e ingresados en el programa.

Para ver el la selección de los subconjuntos para el M.L.P., ir al ANEXO 1.3.

A continuación, se muestran las matrices de pesos y los vectores bias iniciales en la figura 27.

```
Valores iniciales de las matrices:
W_1 =
   -0.1225
   -0.2369

b_1 =
    0.5310
    0.5904

W_2 =
   -0.6263   -0.0205

b_2 =
   -0.1088
```

Figura 27. Se muestran las matrices de pesos y los vectores bias iniciales de cada capa.

S le muestra al usuario los valores finales del error de validación *Eval*, el error de aprendizaje *Eap* y el error de prueba *Ep*, además de si se detuvo con early stopping, se llegó a itmax o fue un aprendizaje exitoso, como se muestra en la figura 28.

```
Presiona ENTER para comenzar el aprendizaje...
Count val = 1
Count val = 2
Count val = 3
Early stopping en iteracion 600
Eap = -0.000000
Eval = 0.000026
Ep = 0.000022
```

Figura 28. Se le muestra al usuario los valores finales de Eap, Eval y Ep.

Y por último se le muestra al usuario las siguientes gráficas:

- Comparación de la evaluación del conjunto de prueba (salida de la red) vs. los targets. (figura 29)
- Comparación de la evaluación del conjunto de entrenamiento (salida de la red) vs. los targets. (figura 30)
- Evolución del error de aprendizaje junto con el error de validación. (figura 31)
- Evolución de los pesos (1 gráfica por capa). (figuras 32 y33)
- Evolución del bias (1 gráfica por capa). (figuras 34 y 35)

Para este experimento se tuvieron los siguientes resultados.

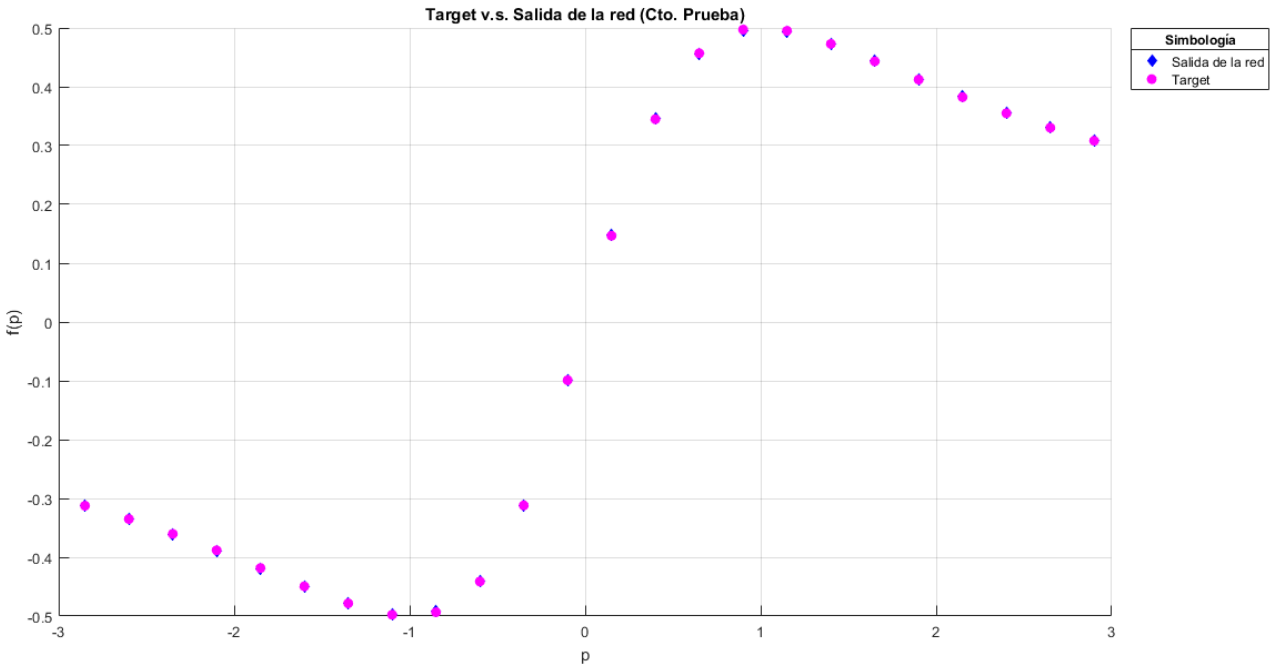

Figura 29. Target vs Salida de la red del conjunto de prueba (Se puede apreciar que el error fue mínimo).

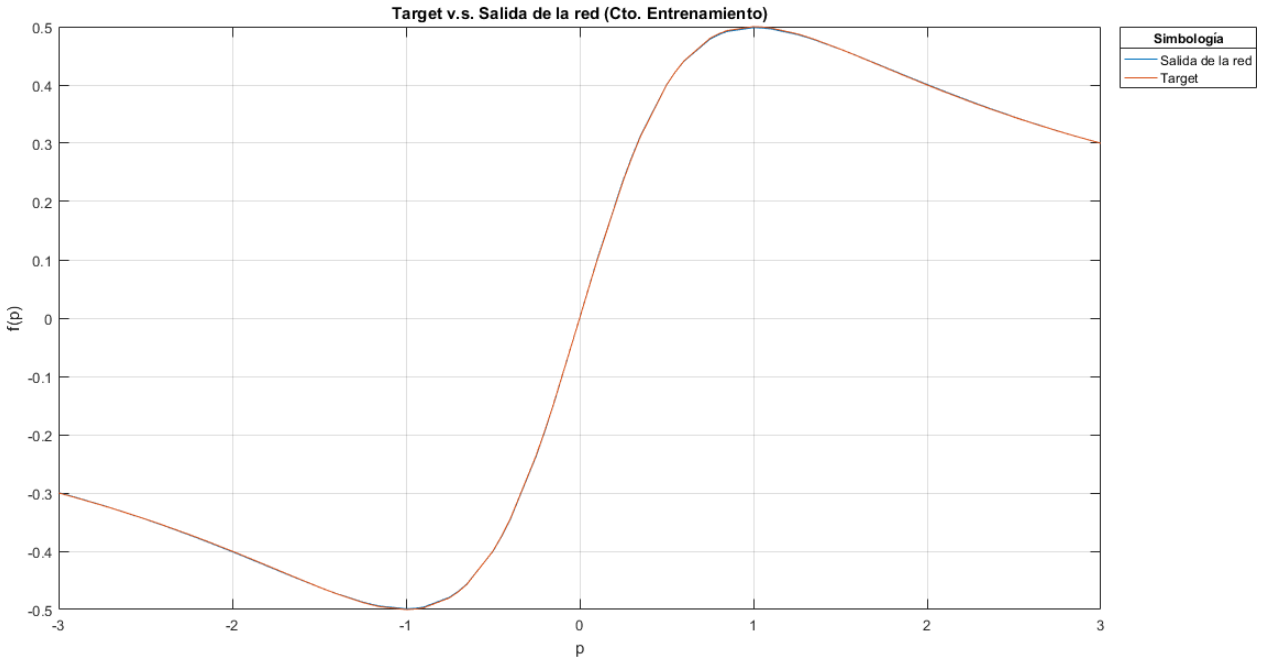

Figura 30. Target vs Salida de la red del conjunto de entrenamiento (Se puede apreciar que el error fue mínimo).

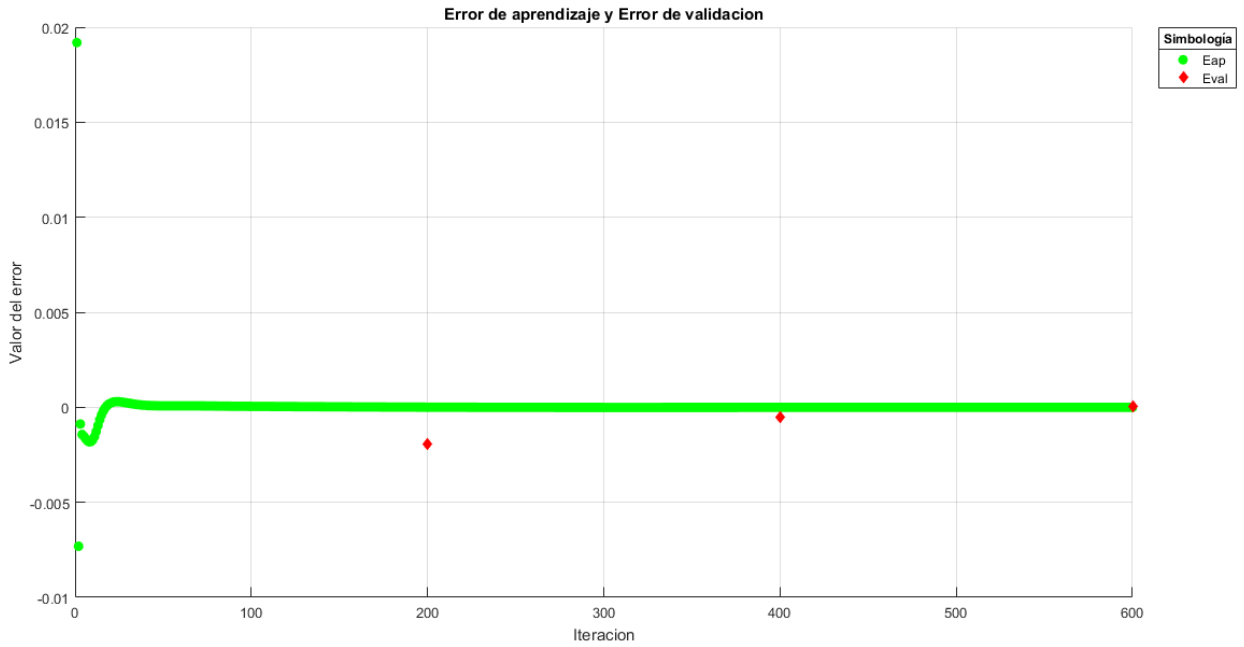

Figura 31. Evolución del error de aprendizaje y el error de validación.

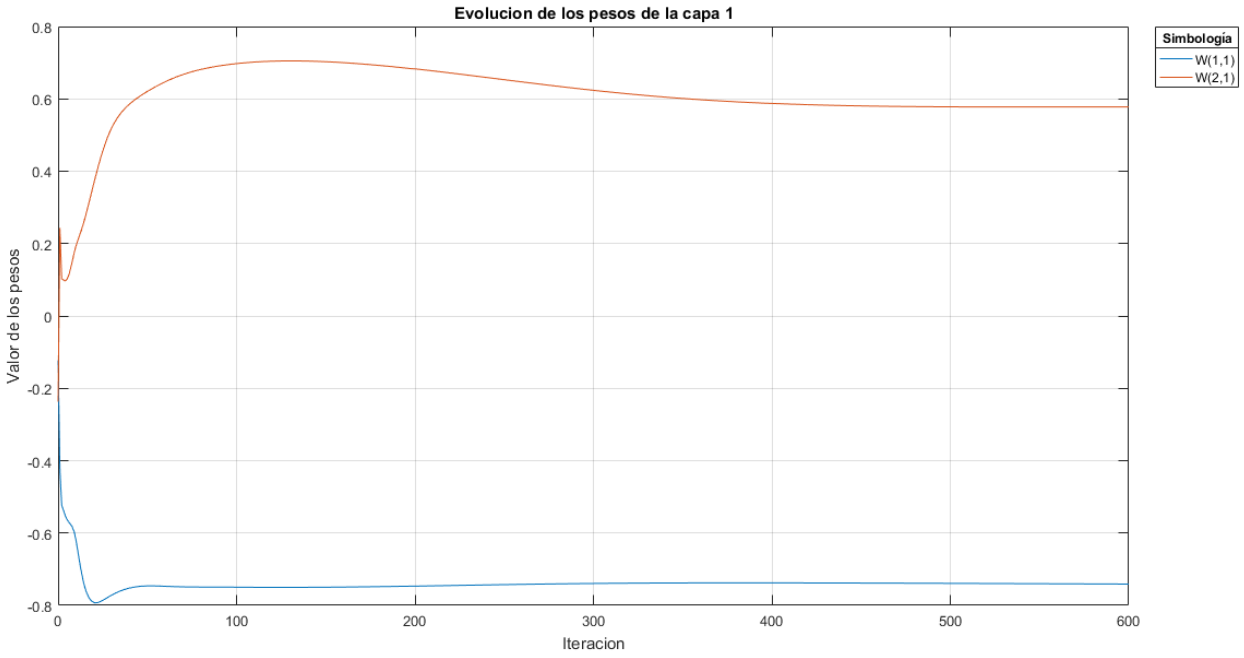

Figura 32. Evolución de los pesos de la capa 1.

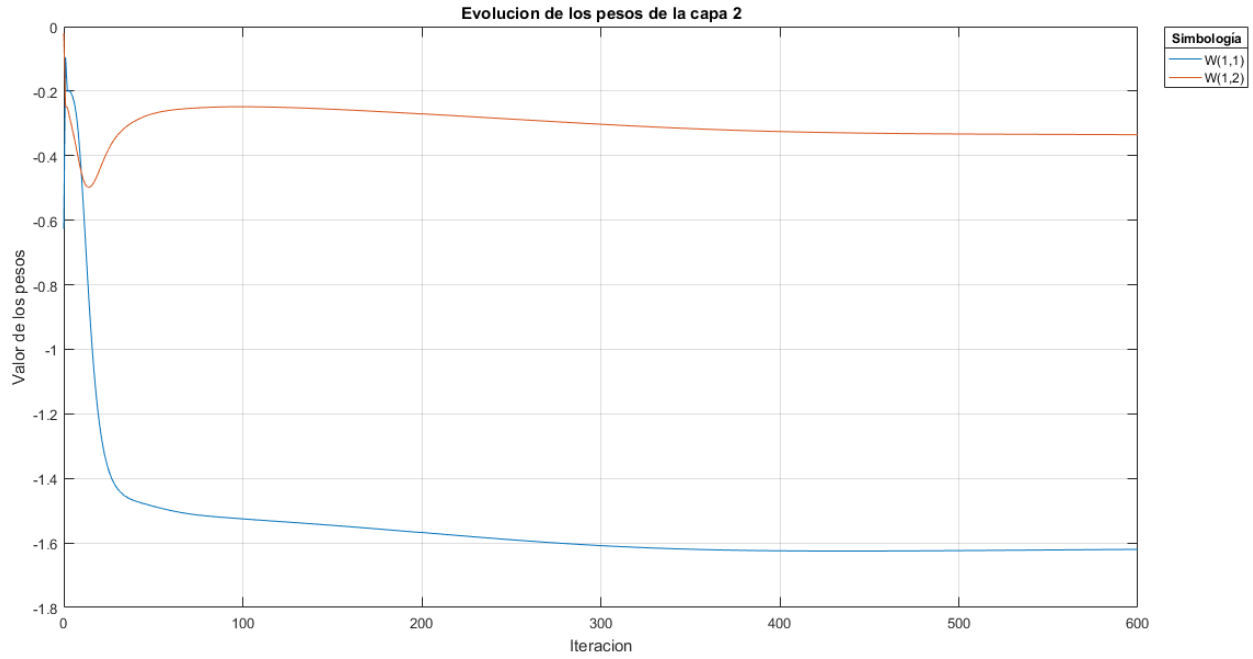

Figura 33. Evolución de los pesos de la capa 2.

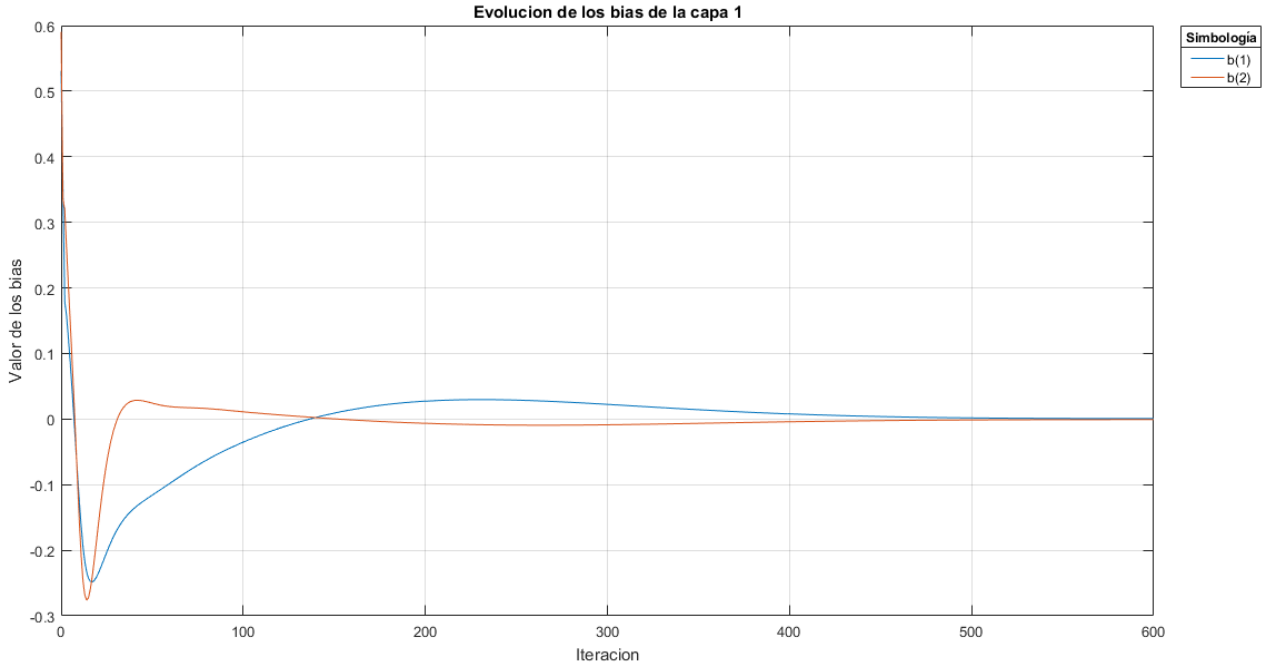

Figura 34. Evolución del bias de la capa 1.

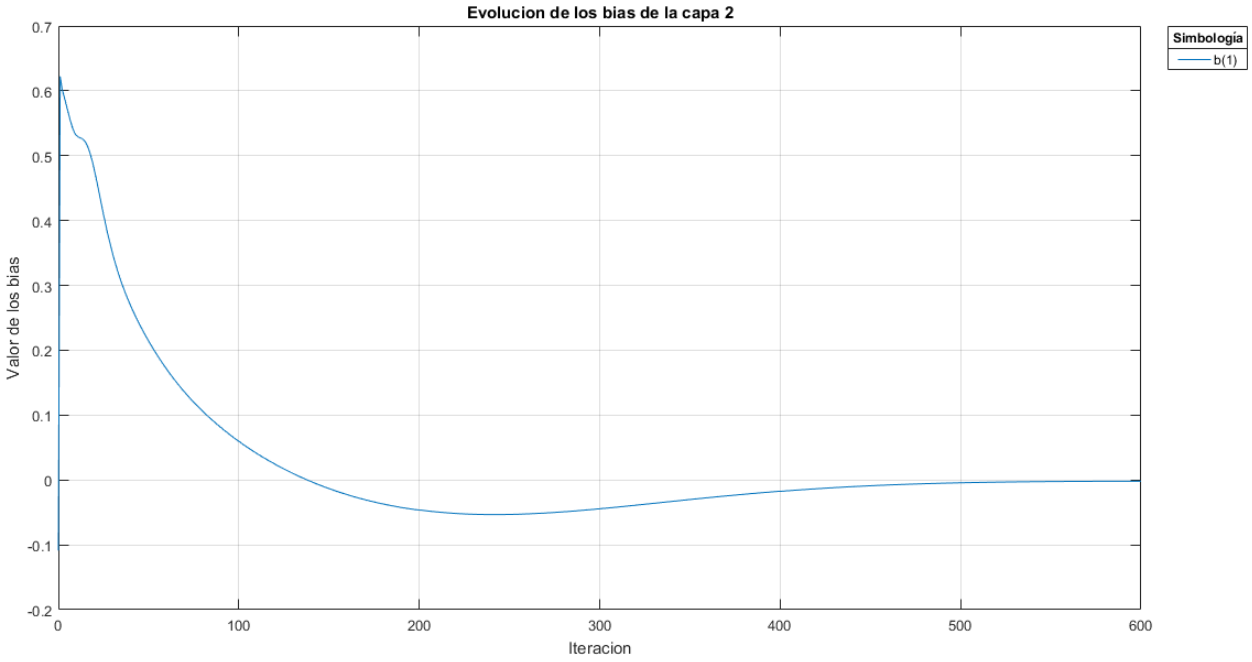

Figura 35. Evolución del bias de la capa 2.

Finalmente verificamos los archivos de los resultados finales como se muestra en la figura 36:

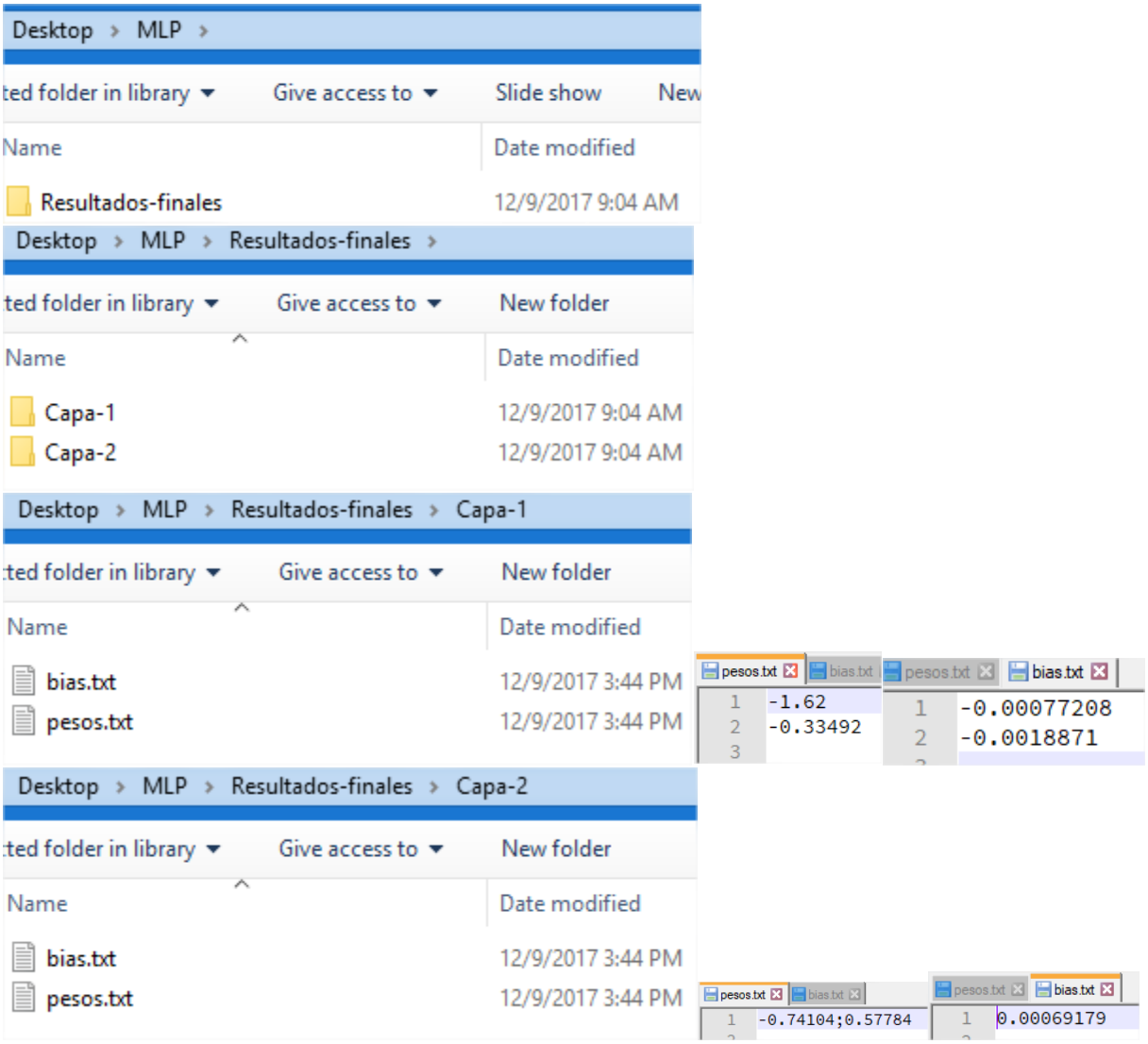

Figura 36. Carpeta contenedora y los respectivos archivos y su contenido.

DISCUSIÓN DE LOS RESULTADOS

Para el primer experimento el cuál fue una tarea previa a la práctica, no presentó ninguna dificultad al volver a realizarse. Este experimento era para demostrar la capacidad de un perceptrón de dos capas con 3 neuronas logsig() en la capa oculta, que no tuvo problemas de aprendizaje, a pesar de que su condición de paro se dio con early-stopping, se tuvieron resultados satisfactorios al comparar la salida de la red con los valores deseados (target).

En cuanto al segundo experimento, me fue difícil encontrar ejemplos de aproximación de señales, por lo que al buscar en internet, logré encontrar un ejemplo en el cuál aproximaban un polinomio de grado 4 con un MLP utilizando la función de activación tansig(). Aunque no decía exactamente cuantas capas y cuantas neuronas utilizó, fue sencillo deducirlos con la experiencia adquirida con los ejercicios previos a la práctica. La condición de paro para este experimento fue llegar a itmax, si bien los resultados no fueron tan acertados como en el primer experimento, se tuvo una buena salida, a la cuál posiblemente se mejoraría con más iteraciones.

Por último, para el tercer experimento se buscó una función no muy compleja con la cuál se ppudiera utilizar nuevamente la función de activación tansig() en la capa oculta, ya que previamente se había intentado aproximar una función sampling, pero los resultados no fueron favorecedores al menos para los valores que se le establecieron a la red. Por lo que se decidió utilizar la función $\frac{1}{1+x^2}$, que era fácil intuir que podía utilizarse una neurona tansig() para aproximarla. Por seguridad, sin llegar al sobre entrenamiento, se utilizaron dos neuronas de este tipo, obteniendo resultados muy buenos.

CONCLUSIONES

Al inicio fue complicado determinar cuales serían los valores adecuados para poder aproximar alguna función, sin embargo, con la práctica, para algunas funciones no muy complejas resultaba sencillo fijarlos valores para la red. El proceso de aprendizaje es muy tardado para funciones de mayor complejidad, por lo que requieren mayor capacidad de procesamiento y mucha paciencia para obtener resultados favorables.

En varias ocasiones tuve complicaciones al no proporcionar una buena elección de los valores de la red, lo que me llevaba a un sobre entrenamiento, o simplemente se hacían todas las iteraciones sin obtener un resultado favorable. Realmente, todo este proceso es mediante prueba y error, lo que me llevó mucho tiempo para que la red me diera resultados buenos.

Para futuro, me gustaría mejorar esta práctica para reducir el tiempo en el proceso de aprendizaje de la red, ya que vi que existen heurísticas para ello. Un ejemplo sería utilizando algoritmos genéticos.

REFERENCIAS

[1] M. T. Hagan, Neural Networks Design, Segunda ed., 2014, p. 357.

[2] M. T. Hagan, Neural Networks Design, Segunda ed., 2014, p. 358.

[3] J. Pérez Valls, «Biblioteca - Universidad de Sevilla,» [En línea]. Available: http://bibing.us.es/proyectos/abreproy/12166/fichero/Volumen+1+-+Memoria+descriptiva+del+proyecto%252F3+-+Perceptron+multicapa.pdf. [Último acceso: 9 Diciembre 2017].

[4] M. T. Hagan, «Multilayer Perceptrons,» de *Neural Networks Design*, p. 358.

[5] «Sobreajuste,» [En línea]. Available: https://es.wikipedia.org/wiki/Sobreajuste. [Último acceso: 9 Diciembre 2017].

[6] J. C. Principe, N. R. Euliano y W. C. Lefebvre, «Computational NeuroEngineering Lab,» 1997. [En línea]. Available: http://www.cnel.ufl.edu/courses/EEL6814/chapter5.pdf. [Último acceso: 12 Diciembre 2017].

[7] H. F. Gutiérrez, «Polilibro Redes Neuronales Artificiales 1.,» [En línea]. Available: http://www.hugo-inc.com/RNA/Unidad%204/4.2.html.

[8] G. B. C., S. D. R., E. D. C. y R. M. C., «Redes Neuronales,» 2001 Noviembre 23. [En línea]. Available: http://www.depi.itchihuahua.edu.mx/apacheco/lengs/ann/pagina_nueva_2.htm.

[9] H. F. Gutiérrez, «Polilibro Redes Neuronales Artificiales 1,» [En línea]. Available: http://www.hugo-inc.com/RNA/Unidad%204/4.1.2.html.

[10] H. F. Gutiérrez, «Polilibro de Redes Neuronales 1,» [En línea]. Available: http://www.hugo-inc.com/RNA/Unidad%202/2.1.html.

[11] H. F. Gutiérrez, «Redes Neuronales Multicapa Con Aprendizaje Supervisado,» [En línea]. Available: http://www.hugo-inc.com/RNA/Unidad%202/2.1.1.html.

ANEXO

SUBCONJUNTOS DE ENTRENAMIENTO, PRUEBA Y VALIDACIÓN

EXPERIMENTO 1

Anexo 1.1. Subconjuntos utilizados en el M.L.P. para el experimento 1.

Conjunto de validación:

-1.7600	0.6319
-1.4800	0.2710
-1.2000	0.0489
-0.9200	0.0079
-0.6400	0.1557
-0.3600	0.4642
-0.0800	0.8747
0.2000	1.3090
0.4800	1.6845
0.7600	1.9298
1.0400	1.9980
1.3200	1.8763
1.6000	1.5878
1.8800	1.1874
-1.7600	0.6319

Conjunto de prueba:

-1.8000	0.6910
-1.5200	0.3155
-1.2400	0.0702
-0.9600	0.0020
-0.6800	0.1237
-0.4000	0.4122
-0.1200	0.8126
0.1600	1.2487
0.4400	1.6374
0.7200	1.9048
1.0000	2.0000
1.2800	1.9048
1.5600	1.6374
1.8400	1.2487
-1.7600	0.6319

Conjunto de entrenamiento:

-2.0000	1.0000
-1.9600	0.9372
-1.9200	0.8747
-1.8800	0.8126
-1.8400	0.7513
-1.7200	0.5742
-1.6800	0.5182
-1.6400	0.4642
-1.6000	0.4122
-1.5600	0.3626
-1.4400	0.2295
-1.4000	0.1910
-1.3600	0.1557
-1.3200	0.1237
-1.2800	0.0952
-1.1600	0.0314
-1.1200	0.0177
-1.0800	0.0079
-1.0400	0.0020
-1.0000	0
-0.8800	0.0177
-0.8400	0.0314
-0.8000	0.0489
-0.7600	0.0702
-0.7200	0.0952
-0.6000	0.1910
-0.5600	0.2295
-0.5200	0.2710
-0.4800	0.3155
-0.4400	0.3626
-0.3200	0.5182
-0.2800	0.5742
-0.2400	0.6319
-0.2000	0.6910
-0.1600	0.7513
-0.0400	0.9372
0.0000	1.0000
0.0400	1.0628
0.0800	1.1253
0.1200	1.1874
0.2400	1.3681
0.2800	1.4258
0.3200	1.4818
0.3600	1.5358
0.4000	1.5878
0.5200	1.7290
0.5600	1.7705
0.6000	1.8090
0.6400	1.8443
0.6800	1.8763
0.8000	1.9511
0.8400	1.9686
0.8800	1.9823
0.9200	1.9921
0.9600	1.9980
1.0800	1.9921
1.1200	1.9823
1.1600	1.9686
1.2000	1.9511
1.2400	1.9298
1.3600	1.8443
1.4000	1.8090
1.4400	1.7705
1.4800	1.7290
1.5200	1.6845
1.6400	1.5358
1.6800	1.4818
1.7200	1.4258
1.7600	1.3681
1.8000	1.3090
1.9200	1.1253

EXPERIMENTO 2

Anexo 1.2. Subconjuntos utilizados en el M.L.P. para el experimento 2.

Conjunto de validacion:

0.0600	-0.2322
0.1300	-0.2450
0.2000	-0.0768
0.2700	0.1856
0.3400	0.4710
0.4100	0.7236
0.4800	0.9034
0.5500	0.9857
0.6200	0.9615
0.6900	0.8375
0.7600	0.6356
0.8300	0.3936
0.9000	0.1647
0.9700	0.0176
0.0600	-0.2322

Conjunto de prueba:

0.0500	-0.2098
0.1200	-0.2565
0.1900	-0.1085
0.2600	0.1452
0.3300	0.4311
0.4000	0.6912
0.4700	0.8832
0.5400	0.9804
0.6100	0.9714
0.6800	0.8607
0.7500	0.6680
0.8200	0.4288
0.8900	0.1942
0.9600	0.0306
0.0600	-0.2322

Conjunto de entrenamiento:

0	0
0.0100	-0.0562
0.0200	-0.1049
0.0300	-0.1465
0.0400	-0.1814
0.0700	-0.2488
0.0800	-0.2600
0.0900	-0.2661
0.1000	-0.2673
0.1100	-0.2640
0.1400	-0.2299
0.1500	-0.2113
0.1600	-0.1897
0.1700	-0.1651
0.1800	-0.1380
0.2100	-0.0432
0.2200	-0.0080
0.2300	0.0286
0.2400	0.0665
0.2500	0.1055
0.2800	0.2264
0.2900	0.2675
0.3000	0.3087
0.3100	0.3498
0.3200	0.3906
0.3500	0.5102
0.3600	0.5485
0.3700	0.5859
0.3800	0.6223
0.3900	0.6574
0.4200	0.7545
0.4300	0.7838
0.4400	0.8113
0.4500	0.8372
0.4600	0.8612
0.4900	0.9215
0.5000	0.9375
0.5100	0.9514
0.5200	0.9633
0.5300	0.9729
0.5600	0.9888
0.5700	0.9896
0.5800	0.9883
0.5900	0.9848
0.6000	0.9792
0.6300	0.9496
0.6400	0.9356
0.6500	0.9197
0.6600	0.9018
0.6700	0.8821
0.7000	0.8127
0.7100	0.7864
0.7200	0.7587
0.7300	0.7296
0.7400	0.6993
0.7700	0.6024
0.7800	0.5685
0.7900	0.5341
0.8000	0.4992
0.8100	0.4641
0.8400	0.3587
0.8500	0.3242
0.8600	0.2903
0.8700	0.2572
0.8800	0.2251
0.9100	0.1369
0.9200	0.1109
0.9300	0.0871
0.9400	0.0656
0.9500	0.0467
0.9800	0.0080
0.9900	0.0021
1.0000	0

EXPERIMENTO 3

Anexo 1.3. Subconjuntos utilizados en el M.L.P. para el experimento 3.

Conjunto de validacion:

-2.8000	-0.3167
-2.5500	-0.3399
-2.3000	-0.3657
-2.0500	-0.3940
-1.8000	-0.4245
-1.5500	-0.4555
-1.3000	-0.4833
-1.0500	-0.4994
-0.8000	-0.4878
-0.5500	-0.4223
-0.3000	-0.2752
-0.0500	-0.0499
0.2000	0.1923
0.4500	0.3742
0.7000	0.4698
0.9500	0.4993
1.2000	0.4918
1.4500	0.4674
1.7000	0.4370
1.9500	0.4060
2.2000	0.3767
2.4500	0.3499
2.7000	0.3257
2.9500	0.3040

Conjunto de prueba:

-2.8500	-0.3124
-2.6000	-0.3351
-2.3500	-0.3603
-2.1000	-0.3882
-1.8500	-0.4183
-1.6000	-0.4494
-1.3500	-0.4783
-1.1000	-0.4977
-0.8500	-0.4935
-0.6000	-0.4412
-0.3500	-0.3118
-0.1000	-0.0990
0.1500	0.1467
0.4000	0.3448
0.6500	0.4569
0.9000	0.4972
1.1500	0.4952
1.4000	0.4730
1.6500	0.4433
1.9000	0.4121
2.1500	0.3824
2.4000	0.3550
2.6500	0.3303
2.9000	0.3082

Conjunto de entrenamiento:

-3.0000	-0.3000
-2.9500	-0.3040
-2.9000	-0.3082
-2.7500	-0.3212
-2.7000	-0.3257
-2.6500	-0.3303
-2.5000	-0.3448
-2.4500	-0.3499
-2.4000	-0.3550
-2.2500	-0.3711
-2.2000	-0.3767
-2.1500	-0.3824
-2.0000	-0.4000
-1.9500	-0.4060
-1.9000	-0.4121
-1.7500	-0.4308
-1.7000	-0.4370
-1.6500	-0.4433
-1.5000	-0.4615
-1.4500	-0.4674
-1.4000	-0.4730
-1.2500	-0.4878
-1.2000	-0.4918
-1.1500	-0.4952
-1.0000	-0.5000
-0.9500	-0.4993
-0.9000	-0.4972
-0.7500	-0.4800
-0.7000	-0.4698
-0.6500	-0.4569
-0.5000	-0.4000
-0.4500	-0.3742
-0.4000	-0.3448
-0.2500	-0.2353
-0.2000	-0.1923
-0.1500	-0.1467
0	0
0.0500	0.0499
0.1000	0.0990
0.2500	0.2353
0.3000	0.2752
0.3500	0.3118
0.5000	0.4000
0.5500	0.4223
0.6000	0.4412
0.7500	0.4800
0.8000	0.4878
0.8500	0.4935
1.0000	0.5000
1.0500	0.4994
1.1000	0.4977
1.2500	0.4878
1.3000	0.4833
1.3500	0.4783
1.5000	0.4615
1.5500	0.4555
1.6000	0.4494
1.7500	0.4308
1.8000	0.4245
1.8500	0.4183
2.0000	0.4000
2.0500	0.3940
2.1000	0.3882
2.2500	0.3711
2.3000	0.3657
2.3500	0.3603
2.5000	0.3448
2.5500	0.3399
2.6000	0.3351
2.7500	0.3212
2.8000	0.3167
2.8500	0.3124
3.0000	0.3000

CÓDIGOS

Para una mejor visualización del código, recomiendo visitar el siguiente repositorio de Github:

https://github.com/IvanovskyOrtega/Redes-Neuronales/blob/master/MLP/

mlp.m

```
clc
clear

% Leemos el archivo de entradas
input1 = input('Ingresa el nombre del archivo que contiene los valores de entrada "*.txt" (sin extension):
','s');
nombreArchivo1 = strcat(input1,'.txt');
p = importdata(nombreArchivo1);

% Leemos el archivo de los valores target
input2 = input('Ingresa el nombre del archivo que contiene los valores target "*.txt" (sin extension):
','s');
nombreArchivo2 = strcat(input2,'.txt');
targets = importdata(nombreArchivo2);
fprintf('\n');

% Se calcula el rango a trabajar
dim_p = size(p);
lim_inf = p(1);
lim_sup = p(dim_p(1));
incremento = (lim_sup-lim_inf)/(dim_p(1)-1);

% Se muestra el rango y el numero de datos a trabajar
fprintf('Se trabajara el siguiente rango, de acuerdo al archivo:\n');
fprintf('[%f, %f], con un incremento de %f\n',lim_inf,lim_sup,incremento);
num_datos = dim_p(1);
fprintf('Se trabajara con %d datos, de acuerdo al archivo.\n\n',num_datos);

% Se solicita la arquitectura del M.L.P.
fprintf('\n');
str_arq = input('Ingresa la arquitectura del M.L.P.: ','s');
arq_mlp = str2num(str_arq);
num_capas = length(arq_mlp)-1;
R = arq_mlp(1);
fprintf('Para las funciones de activacion se tienen las siguientes:\n');
fprintf('1) purelin()\n2) logsig()\n3) tansig()\n\n');
str_fun = input('Ingresa las funciones de las capas de la red separadas por un espacio: ','s');
fun_capa = str2num(str_fun);
disp('La arquitectura del M.L.P. es:');
disp(arq_mlp);
disp(fun_capa);

% Se abren archivos para graficacion en modo escritura (Esto es un poco largo...)
num_archivos_pesos_total = 0;
num_archivos_bias_total = 0;
for i=1:num_capas
    for j=1:arq_mlp(i+1)
        for l=1:arq_mlp(i)
            num_archivos_pesos_total = num_archivos_pesos_total +1;
        end
    end
    num_archivos_bias_total = num_archivos_bias_total +1;
end

archivos_pesos = zeros(num_archivos_pesos_total,1);
archivos_bias = zeros(num_archivos_bias_total,1);
num_archivo = 1;
for i=1:num_capas
```

```
    path = strcat(pwd,'/Valores-de-Graficacion/Capa-',num2str(i),'/Pesos/');
    if ~exist(path, 'dir')
        mkdir(path);
    end
    for j=1:arq_mlp(i+1)
        for k=1:arq_mlp(i)
            archivo_pesos = strcat(path,'/pesos',num2str(j),'_',num2str(k),'.txt');
            archivos_pesos(num_archivo) = fopen(archivo_pesos,'w');
            num_archivo = num_archivo +1;
        end
    end
end

num_archivo = 1;
for i=1:num_capas
    path = strcat(pwd,'/Valores-de-Graficacion/Capa-',num2str(i),'/bias/');
    if ~exist(path, 'dir')
        mkdir(path);
    end
    for j=1:arq_mlp(i+1)
        archivo_bias = strcat(path,'/bias',num2str(j),'.txt');
        archivos_bias(num_archivo) = fopen(archivo_bias,'w');
        num_archivo = num_archivo +1;
    end
end

% Se terminan de abrir los archivos

% Se solicita el valor del factor de aprendizaje
alfa = input('Ingresa el valor del factor de aprendizaje(alfa): ');

% Se solicitan los valores de usuario eit,itmax, itval, numval
itmax = input('Ingresa el numero de iteraciones maximas de la red(itmax): ');
itval = input('¿Cada cuanto se hara una iteracion de validacion? (itval): ');
numval = input('Numero maximo de incrementos consecutivos del error de validacion (numval): ');
eit = input('Ingrese l valor minimo del error en una epoca (eit): ');
fprintf('\n');

% Se solicita la configuracion para dividir en subconjuntos
fprintf('Seleccione una de las siguientes configuraciones a trabajar:\n');
fprintf('1) 80-10-10\n');
fprintf('2) 70-15-15\n');
config = input('Ingrese su seleccion: ');

% Se forman los subconjutnos de acuerdo al tipo de configuracion
switch(config)
    case 1
        num_elem_val = round(num_datos*.2); % Numero de elementos del conjunto de validacion
        num_elem_prueba = num_elem_val; % Numero de elementos del conjunto de prueba
        num_elem_ent = num_datos - 2*num_elem_val; % Numero de elementos del conjunto de entrenamiento
    case 2
        num_elem_val = round(num_datos*.15);
        num_elem_prueba = num_elem_val;
        num_elem_ent = num_datos - 2*num_elem_val;
end
disp('Se usaran los siguientes tamanios en los subconjuntos:');
fprintf('Conjunto de entrenamiento: %d elementos\n',num_elem_ent);
fprintf('Conjunto de validacion: %d elementos\n',num_elem_val);
fprintf('Conjunto de prueba: %d elementos\n',num_elem_prueba);
cto_val = obtenerConjuntoDeValidacion(p,targets,num_datos,num_elem_val);
cto_prueba = obtenerConjuntoDePrueba(p,targets,num_datos,num_elem_prueba);
cto_ent = obtenerConjuntoDeEntrenamiento(p,targets,num_datos,num_elem_ent,cto_val,cto_prueba);
disp('Conjunto de validacion:');
disp(cto_val);
disp('Conjunto de prueba:');
disp(cto_prueba);
disp('Conjunto de entrenamiento:');
```

```
disp(cto_ent);

% Se inicializan la matriz de pesos y el vector bias con valores aleatorios
% entre -1 y 1

num_archivos_pesos = 1;
num_archivos_bias = 1;
W = cell(num_capas,1);
b = cell(num_capas,1);
disp('Valores iniciales de las matrices:');
for i=1:num_capas
    temp_W = -1 + 2*rand(arq_mlp(i+1),arq_mlp(i));
    W{i} = temp_W;
    fprintf('W_%d = \n',i);
    disp(W{i});
    temp_b = -1 + (2)*rand(arq_mlp(i+1),1);
    b{i} = temp_b;
    fprintf('b_%d = \n',i);
    disp(b{i});

    % Se imprimen los valores iniciales en los archivos
    for j=1:arq_mlp(i+1)
        for k=1:arq_mlp(i)
            fprintf(archivos_pesos(num_archivos_pesos),'%f\r\n',temp_W(j,k));
            num_archivos_pesos = num_archivos_pesos +1;
        end
    end
    for j=1:arq_mlp(i+1)
        fprintf(archivos_bias(num_archivos_bias),'%f\r\n',temp_b(j));
        num_archivos_bias = num_archivos_bias + 1;
    end

end

% Se utiliza una cell para guardar las salidas de cada capa
a = cell(num_capas+1,1);

% Se utiliza una cell para guardas las sensitividades de cada capa y las
% matrices de derivadas.
S = cell(num_capas,1);
F_m = cell(num_capas,1);
X = input('Presiona ENTER para comenzar el aprendizaje...');

% Comienza el aprendizaje
early_stopping = 0;
Err_val = 0;
Err_ap = 0;
valores_graficacion_eap = zeros(itmax,1);
valores_graficacion_eval = zeros(ceil(itmax/itval),1);
count_val = 0;
num_it_val = 0; % Numero de iteraciones de validacion realizadas
for it=1:itmax
    num_archivos_pesos = 1;
    num_archivos_bias = 1;
    Eap = 0; % Error de aprendizaje
    % Si no es una iteracion de validacion
    if(mod(it,itval)~=0)
        for dato=1:num_elem_ent

            a{1} = cto_ent(dato,1); % Condicion inicial

            % Se propaga hacia adelante el elemento del cto. de
            % entrenamiento
            for k=1:num_capas
                W_temp = cell2mat(W(k));
                a_temp = cell2mat(a(k));
```

```
            b_temp = cell2mat(b(k));
            a{k+1} = funcionDeActivacion(W_temp*a_temp+b_temp,fun_capa(k));
        end
        a_temp = cell2mat(a(num_capas+1));
        ej = cto_ent(dato,2)-a_temp;
        Eap = Eap+(ej/num_datos);

        % Se calculan las sensitividades y se propagan hacia atras,
        % es decir, inicia el backpropagation.
        F_m{num_capas} = obtenerF(fun_capa(num_capas),arq_mlp(num_capas+1),a_temp);
        F_m_temp = cell2mat(F_m(num_capas));
        S{num_capas} = -2*F_m_temp*(ej);
        for m = num_capas-1:-1:1
            W_temp = cell2mat(W(m+1));
            a_temp = cell2mat(a(m+1));
            S_temp = cell2mat(S(m+1));
            F_m{m} = obtenerF(fun_capa(m),arq_mlp(m+1),a_temp);
            F_m_temp = cell2mat(F_m(m));
            S{m} = F_m_temp*(W_temp')*S_temp;
        end

        % Se aplican las reglas de aprendizaje
        for k=num_capas:-1:1
            W_temp = cell2mat(W(k));
            b_temp = cell2mat(b(k));
            a_temp = cell2mat(a(k));
            S_temp = cell2mat(S(k));
            W{k} = W_temp-(alfa*S_temp*(a_temp'));
            b{k} = b_temp-(alfa*S_temp);
            W_temp = cell2mat(W(k));
            b_temp = cell2mat(b(k));
        end

    end
    Err_ap = Eap;

    % Se guarda el valor de graficación de Eap
    valores_graficacion_eap(it) = Eap;

% Si es una iteracion de validacion
else
    E_val = 0;
    num_it_val = num_it_val + 1;
    for dato=1:num_elem_val
        a{1} = cto_val(dato,1); % Condicion inicial
        % Se propaga hacia adelante el elemento del cto. de
        % validacion.
        for k=1:num_capas
            W_temp = cell2mat(W(k));
            a_temp = cell2mat(a(k));
            b_temp = cell2mat(b(k));
            a{k+1} = funcionDeActivacion(W_temp*a_temp+b_temp,fun_capa(k));
        end
        a_temp = cell2mat(a(num_capas+1));
        e_val = cto_val(dato,2)-a_temp;
        E_val = E_val+(e_val/num_elem_val);
    end

    % Se guarda el valor para graficacion
    valores_graficacion_eval(it) = E_val;

    if count_val == 0
        Err_val = E_val;
        count_val = count_val+1;
        fprintf('Count val = %d\n',count_val);
    else
```

```
            if E_val > Err_val
                Err_val = E_val;
                count_val = count_val+1;
                fprintf('Count val = %d\n',count_val);
                if count_val == numval
                    early_stopping = 1;
                    fprintf('Early stopping en iteracion %d\n',it);
                    break;
                end
            else
                Err_val = 0;
                count_val = 0;
                fprintf('Count val = %d\n',count_val);
            end
        end
    end

    % Se imprimen los valores de pesos y bias modificados a archivo
    num_archivos_pesos = 1;
    num_archivos_bias = 1;
    for k=num_capas:-1:1
        W_temp = cell2mat(W(k));
        b_temp = cell2mat(b(k));
        for j=1:arq_mlp(k+1)
            for l=1:arq_mlp(k)
                fprintf(archivos_pesos(num_archivos_pesos),'%f\r\n',W_temp(j,l));
                num_archivos_pesos = num_archivos_pesos +1;
            end
        end
        for j=1:arq_mlp(k+1)
            fprintf(archivos_bias(num_archivos_bias),'%f\r\n',b_temp(j));
            num_archivos_bias = num_archivos_bias + 1;
        end
    end

    % Se comprueban las condiciones de finalizacion
    if Eap <= eit && Eap >= 0 && mod(it,itval) ~= 0
        Err_ap = Eap;
        fprintf('Aprendizaje exitoso en la iteracion %d\n',it);
        break;
    end
end

if it == itmax
    disp('Se llego a itmax.');
end

% Se imprimen a archivo los ultimos valores de pesos y bias de cada capa
if early_stopping == 1
% Se imprimen los valores de pesos y bias modificados a archivo
    num_archivos_pesos = 1;
    num_archivos_bias = 1;
    for k=num_capas:-1:1
        W_temp = cell2mat(W(k));
        b_temp = cell2mat(b(k));
        for j=1:arq_mlp(k+1)
            for l=1:arq_mlp(k)
                fprintf(archivos_pesos(num_archivos_pesos),'%f\r\n',W_temp(j,l));
                num_archivos_pesos = num_archivos_pesos +1;
            end
        end
        for j=1:arq_mlp(k+1)
            fprintf(archivos_bias(num_archivos_bias),'%f\r\n',b_temp(j));
            num_archivos_bias = num_archivos_bias + 1;
        end
    end
end
```

```
% Se cierran los archivos de valores de graficacion de pesos y bias
for i=1:num_archivos_pesos_total
    fclose(archivos_pesos(i));
end
for i=1:num_archivos_bias_total
    fclose(archivos_bias(i));
end

% Se propaga el conjunto de prueba
Ep = 0; % Error de prueba
salida_red = zeros(num_elem_prueba,1);
for i=1:num_elem_prueba
    a{1} = cto_prueba(i,1); % Condicion inicial
    % Se propaga hacia adelante el elemento del cto. de prueba
    for k=1:num_capas
        W_temp = cell2mat(W(k));
        a_temp = cell2mat(a(k));
        b_temp = cell2mat(b(k));
        a{k+1} = funcionDeActivacion(W_temp*a_temp+b_temp,fun_capa(k));
    end
    dato_entrada = cell2mat(a(1));
    a_temp = cell2mat(a(num_capas+1));
    Ep = Ep+(1/num_elem_prueba)*(cto_prueba(i,2)-a_temp);
    salida_red(i) = a_temp;
end

% Se imprimen los valores finales de Eap, Ep y Eval
fprintf('Eap = %f\n',Err_ap);
fprintf('Eval = %f\n',Err_val);
fprintf('Ep = %f\n',Ep);

% Graficacion del conjunto de prueba, se muestran los targets contra los
% resultados de la red.
figure
rango = cto_prueba(:,1);
s1 = scatter(rango,salida_red,'d');
s1.MarkerFaceColor = [0 0 1];
s1.MarkerEdgeColor = 'b';
grid on
hold on
s2 = scatter(rango,cto_prueba(:,2));
s2.MarkerFaceColor = [1 0 1];
s2.MarkerEdgeColor = 'm';
title('Target v.s. Salida de la red (Cto. Prueba)');
ylabel('f(p)');
xlabel('p');
lgd = legend('Salida de la red','Target','Location','northeastoutside');
title(lgd,'Simbología');
hold off

% Se propaga el conjunto de entrenamiento
salida_red = zeros(num_elem_ent,1);
for i=1:num_elem_ent
    a{1} = cto_ent(i,1); % Condicion inicial
    % Se propaga hacia adelante el elemento del cto. de
    % entrenamiento
    for k=1:num_capas
        W_temp = cell2mat(W(k));
        a_temp = cell2mat(a(k));
        b_temp = cell2mat(b(k));
        a{k+1} = funcionDeActivacion(W_temp*a_temp+b_temp,fun_capa(k));
    end
    dato_entrada = cell2mat(a(1));
    a_temp = cell2mat(a(num_capas+1));
    Ep = Ep+(1/num_elem_ent)*(cto_ent(i,2)-a_temp);
    salida_red(i) = a_temp;
```

```
end

% Graficacion del conjunto de entrenamiento, se muestran los targets contra
% los resultados de la red.
figure
rango = cto_ent(:,1);
plot(rango,salida_red);
grid on
hold on
plot(rango,cto_ent(:,2));
title('Target v.s. Salida de la red (Cto. Entrenamiento)');
ylabel('f(p)');
xlabel('p');
lgd = legend('Salida de la red','Target','Location','northeastoutside');
title(lgd,'Simbología');
hold off

% Se grafica la evolucion de los errores de aprendizaje y validacion por
% epoca.
figure
rango = 1:1:it;
rango2 = itval:itval:num_it_val*itval;
s1 = scatter(rango,valores_graficacion_eap(1:it,1));
s1.MarkerFaceColor = [0 1 0];
s1.MarkerEdgeColor = 'g';
grid on
hold on
s2 = scatter(rango2,valores_graficacion_eval(itval:itval:num_it_val*itval,1),'d');
s2.MarkerFaceColor = [1 0 0];
s2.MarkerEdgeColor = 'r';
title('Error de aprendizaje y Error de validacion');
ylabel('Valor del error');
xlabel('Iteracion');
lgd = legend('Eap','Eval','Location','northeastoutside');
title(lgd,'Simbología');
hold off

%Se grafica la evolucion de los pesos
rango = 0:1:it;
for i=1:num_capas
    figure
    path = strcat(pwd,'/Valores-de-Graficacion/Capa-',num2str(i),'/Pesos/');
    for j=1:arq_mlp(i+1)
        for k=1:arq_mlp(i)
            archivo_pesos = strcat(path,'/pesos',num2str(j),'_',num2str(k),'.txt');
            simb = strcat('W(',num2str(j),',',num2str(k),')');
            evolucion_pesos = importdata(archivo_pesos); % Identificador para la grafica
            plot(rango,evolucion_pesos','DisplayName',simb);
            hold on
            grid on
        end
    end
    titulo = strcat('Evolucion de los pesos de la capa',{' '},num2str(i));
    title(titulo);
    ylabel('Valor de los pesos');
    xlabel('Iteracion');
    lgd = legend('show','Location','northeastoutside');
    title(lgd,'Simbología');
    hold off
end

%Se grafica la evolucion de los bias
rango = 0:1:it;
for i=1:num_capas
    figure
    path = strcat(pwd,'/Valores-de-Graficacion/Capa-',num2str(i),'/bias/');
    for j=1:arq_mlp(i+1)
```

```
        archivo_bias = strcat(path,'/bias',num2str(j),'.txt');
        simb = strcat('b(',num2str(j),')');
        evolucion_bias = importdata(archivo_bias); % Identificador para la grafica
        plot(rango,evolucion_bias','DisplayName',simb);
        hold on
        grid on
    end
    titulo = strcat('Evolucion de los bias de la capa',{' '},num2str(i));
    title(titulo);
    ylabel('Valor de los bias');
    xlabel('Iteracion');
    lgd = legend('show','Location','northeastoutside');
    title(lgd,'Simbología');
    hold off
end

for i=1:num_capas
    path = strcat(pwd,'/Resultados-finales/Capa-',num2str(i),'/');
    if ~exist(path, 'dir')
        mkdir(path);
    end
    W_temp = cell2mat(W(i));
    res_pesos = strcat(path,'/pesos.txt');
    dlmwrite(res_pesos,W_temp,';');
end

for i=1:num_capas
    path = strcat(pwd,'/Resultados-finales/Capa-',num2str(i),'/');
    if ~exist(path, 'dir')
        mkdir(path);
    end
    b_temp = cell2mat(b(i));
    res_bias = strcat(path,'/bias.txt');
    dlmwrite(res_bias,b_temp,';');
end
```

obtenerF.m

```
function F = obtenerF(tipo_funcion,num_neuronas,a)
    switch tipo_funcion
        case 1
            F = diag(ones(1,num_neuronas));
        case 2
            F = diag(logsig('dn',a,a));
        case 3
            F = diag(tansig('dn',a,a));
    end
end
```

obtenerConjuntoDeEntrenamiento.m

```
function M = obtenerConjuntoDeEntrenamiento(entradas,targets,num_datos,num_elem_ent,cto_val,cto_prueba)
M = zeros(num_elem_ent,2);
j = 1;
for i=1:num_datos
    if ismember(entradas(i),cto_val(:,1)) || ismember(entradas(i),cto_prueba(:,1))
    else
        M(j,1) = entradas(i);
        M(j,2) = targets(i);
        j = j+1;
    end
    if j == num_elem_ent+1
        break;
    end
end
end
```

obtenerConjuntoDePrueba.m

```
function M = obtenerConjuntoDePrueba(entradas,targets,num_datos,num_elem)
M = zeros(num_elem,2);
inc = ceil(num_datos/(num_elem+1));
j = 1;
for i=inc:inc:num_datos
    M(j,1) = entradas(i-1);
    M(j,2) = targets(i-1);
    j = j+1;
    if j == num_elem+1
        break;
    end
end
if j ~= num_elem+1
    for i=inc+1:inc:num_datos
        M(j,1) = entradas(i-1);
        M(j,2) = targets(i-1);
        j = j+1;
        if j == num_elem+1
            break;
        end
    end
end
end
```

obtenerConjuntoDeValidacion.m

```
function M = obtenerConjuntoDeValidacion(entradas,targets,num_datos,num_elem)
M = zeros(num_elem,2);
inc = ceil(num_datos/(num_elem+1));
j = 1;
for i=inc:inc:num_datos
    M(j,1) = entradas(i);
    M(j,2) = targets(i);
    j = j+1;
    if j == num_elem+1
        break;
    end
end
if j ~= num_elem+1
    for i=inc+1:inc:num_datos
        M(j,1) = entradas(i-1);
        M(j,2) = targets(i-1);
        j = j+1;
        if j == num_elem+1
            break;
        end
    end
end
end
```