

# Práctica #5 Perceptrón Multicapa

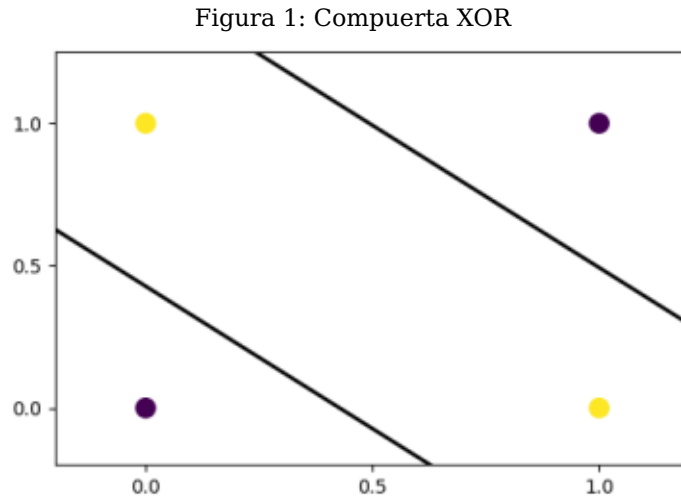
Jorge Gómez Reus

## Índice

<b>1. Introducción</b>	<b>1</b>
1.1. Modelo . . . . .	3
1.1.1. Forward Propagation . . . . .	3
1.1.2. Forward Propagation . . . . .	3
<b>2. Diagrama de Flujo</b>	<b>4</b>
<b>3. Resultados</b>	<b>5</b>
3.1. Polinomio 1 . . . . .	5
3.2. Inputs . . . . .	5
3.3. Targets . . . . .	5
3.3.1. Datos . . . . .	6
3.4. Resultado . . . . .	7
3.5. Imágenes . . . . .	7
3.6. Polinomio 2 . . . . .	16
3.7. Inputs . . . . .	16
3.8. Targets . . . . .	18
3.8.1. Datos . . . . .	20
3.9. Resultado . . . . .	20
3.10. Imágenes . . . . .	20
3.11. Polinomio 3 . . . . .	30
3.12. Inputs . . . . .	30
3.13. Targets . . . . .	35
3.13.1. Datos . . . . .	41
3.14. Resultado . . . . .	41
3.15. Imágenes . . . . .	41
<b>4. Discusión de Resultados</b>	<b>51</b>
<b>5. Conclusiones</b>	<b>51</b>
<b>6. Referencias</b>	<b>51</b>
<b>7. Apéndice</b>	<b>51</b>

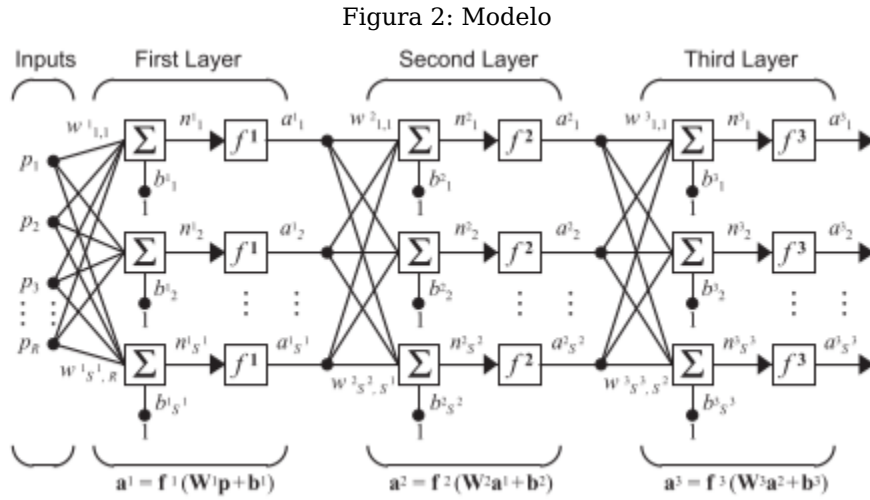
## 1. Introducción

Un perceptrón multicapa es tipo de red neuronal artificial compuesta de varias neuronas y varias capas, puede resolver problemas que un perceptrón simple puede. El clásico ejemplo es el de la compuerta XOR:



En este caso se necesitan dos fronteras de decisión, por ende dos neuronas y se necesita otra capa para “combinar los resultados”. La manera la cual se actualizan los pesos y bías del MLP es usando un algoritmo para propagar los resultados a las neuronas de la capa actual y de las anteriores, es algoritmo es llamado backpropagation, el cual es un algoritmo de minimización basado en el descenso en gradiente el cual encuentra los mínimos locales de una función. En esta práctica se usan MLP's para aproximar señales leyendo datos de archivos de texto.

## 1.1. Modelo



### 1.1.1. Forward Propagation

$$a^0 = p$$

$$a^{m+1} = f^{m+1}(W^{m+1} \cdot a^m + b^{m+1}), m = 0, 1, 2, 3, \dots, M-1$$

$$a = a^M$$

donde  $M$  es el número de capas.

### 1.1.2. Forward Propagation

Para poder usar backpropagation se necesita que las funciones de activación en cada capa sean continuas y derivables. El algoritmo es el siguiente:

1. Calcular las sensibilidades de cada capa desde la última capa hasta la primera:

$$s^M = -2\dot{F}^M(n^M)(t - a)$$

$$s^m = \dot{F}^m(n^m)(W^{m+1})^T s^{m+1}$$

2. Actualizar los pesos y bias:

$$w^m(k+1) = W^m(k) - \alpha s^m(a^m - 1)^T$$

$$b^m(k+1) = b^m(k) - \alpha s^m$$

## 2. Diagrama de Flujo

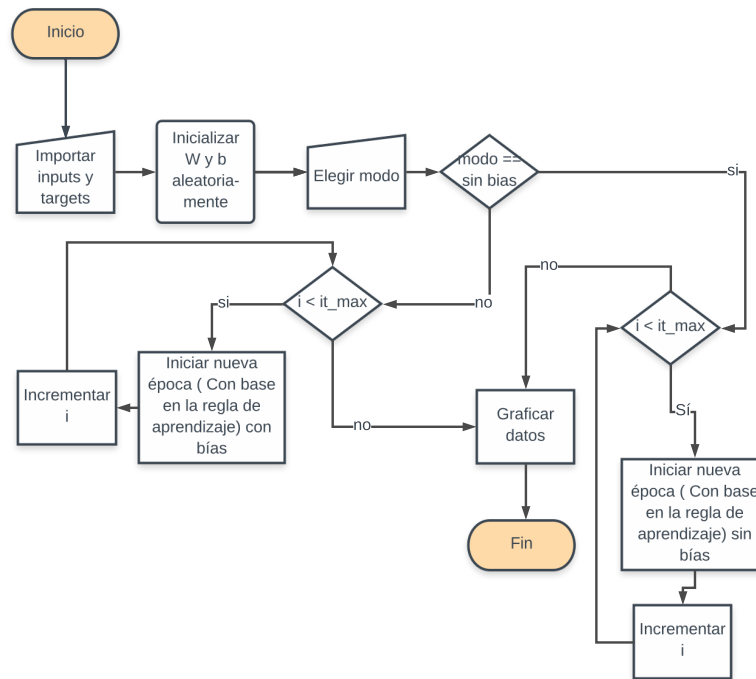


Figura 3: Diagrama de Flujo

### 3. Resultados

#### 3.1. Polinomio 1

#### 3.2. Inputs

1. -2.0000	27. -0.9600	53. 0.0800	79. 1.1200
2. -1.9600	28. -0.9200	54. 0.1200	80. 1.1600
3. -1.9200	29. -0.8800	55. 0.1600	81. 1.2000
4. -1.8800	30. -0.8400	56. 0.2000	82. 1.2400
5. -1.8400	31. -0.8000	57. 0.2400	83. 1.2800
6. -1.8000	32. -0.7600	58. 0.2800	84. 1.3200
7. -1.7600	33. -0.7200	59. 0.3200	85. 1.3600
8. -1.7200	34. -0.6800	60. 0.3600	86. 1.4000
9. -1.6800	35. -0.6400	61. 0.4000	87. 1.4400
10. -1.6400	36. -0.6000	62. 0.4400	88. 1.4800
11. -1.6000	37. -0.5600	63. 0.4800	89. 1.5200
12. -1.5600	38. -0.5200	64. 0.5200	90. 1.5600
13. -1.5200	39. -0.4800	65. 0.5600	91. 1.6000
14. -1.4800	40. -0.4400	66. 0.6000	92. 1.6400
15. -1.4400	41. -0.4000	67. 0.6400	93. 1.6800
16. -1.4000	42. -0.3600	68. 0.6800	94. 1.7200
17. -1.3600	43. -0.3200	69. 0.7200	95. 1.7600
18. -1.3200	44. -0.2800	70. 0.7600	96. 1.8000
19. -1.2800	45. -0.2400	71. 0.8000	97. 1.8400
20. -1.2400	46. -0.2000	72. 0.8400	98. 1.8800
21. -1.2000	47. -0.1600	73. 0.8800	99. 1.9200
22. -1.1600	48. -0.1200	74. 0.9200	100. 1.9600
23. -1.1200	49. -0.0800	75. 0.9600	101. 2.0000
24. -1.0800	50. -0.0400	76. 1.0000	
25. -1.0400	51. 0	77. 1.0400	
26. -1.0000	52. 0.0400	78. 1.0800	

#### 3.3. Targets

1. 1.0000	27. 1.2487	53. 1.4818	79. 1.6845
2. 1.2487	28. 1.4818	54. 1.6845	80. 1.8443
3. 1.4818	29. 1.6845	55. 1.8443	81. 1.9511
4. 1.6845	30. 1.8443	56. 1.9511	82. 1.9980
5. 1.8443	31. 1.9511	57. 1.9980	83. 1.9823
6. 1.9511	32. 1.9980	58. 1.9823	84. 1.9048
7. 1.9980	33. 1.9823	59. 1.9048	85. 1.7705
8. 1.9823	34. 1.9048	60. 1.7705	86. 1.5878
9. 1.9048	35. 1.7705	61. 1.5878	87. 1.3681
10. 1.7705	36. 1.5878	62. 1.3681	88. 1.1253
11. 1.5878	37. 1.3681	63. 1.1253	89. 0.8747
12. 1.3681	38. 1.1253	64. 0.8747	90. 0.6319
13. 1.1253	39. 0.8747	65. 0.6319	91. 0.4122
14. 0.8747	40. 0.6319	66. 0.4122	92. 0.2295
15. 0.6319	41. 0.4122	67. 0.2295	93. 0.0952
16. 0.4122	42. 0.2295	68. 0.0952	94. 0.0177
17. 0.2295	43. 0.0952	69. 0.0177	95. 0.0020
18. 0.0952	44. 0.0177	70. 0.0020	96. 0.0489
19. 0.0177	45. 0.0020	71. 0.0489	97. 0.1557
20. 0.0020	46. 0.0489	72. 0.1557	98. 0.3155
21. 0.0489	47. 0.1557	73. 0.3155	99. 0.5182
22. 0.1557	48. 0.3155	74. 0.5182	100. 0.7513
23. 0.3155	49. 0.5182	75. 0.7513	101. 1.0000
24. 0.5182	50. 0.7513	76. 1.0000	
25. 0.7513	51. 1.0000	77. 1.2487	
26. 1.0000	52. 1.2487	78. 1.4818	

### 3.3.1. Datos

$$V1 = [11 \ 16 \ 10 \ 1]$$

$$V2 = [3 \ 2 \ 1]$$

epochmax = 10000

Múltiplo para las épocas de validación = 500

numval = 7

alpha=.0701

error de validación = .0000000000000001

Configuración: 80-15-15

### 3.4. Resultado

### 3.5. Imágenes

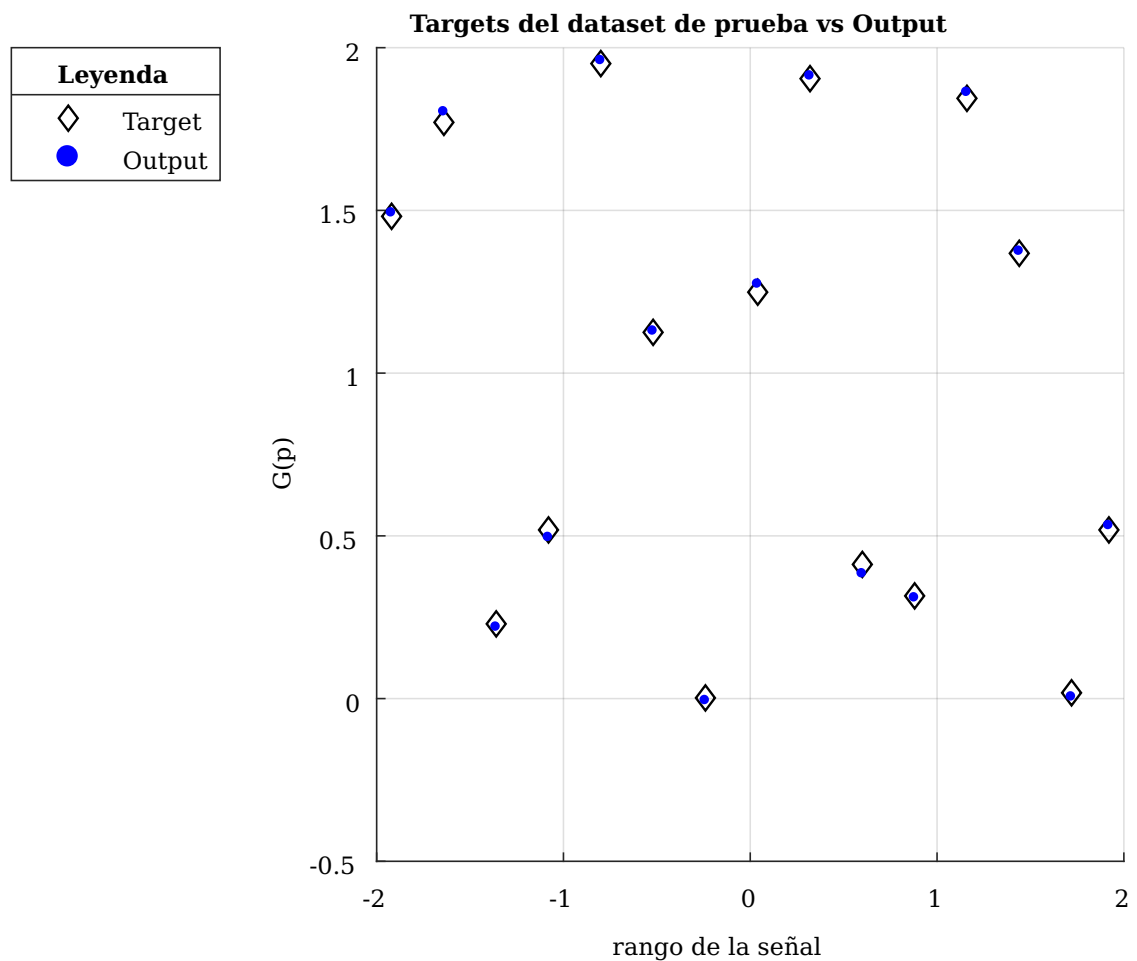


Figura 4: Gráfica 1.1

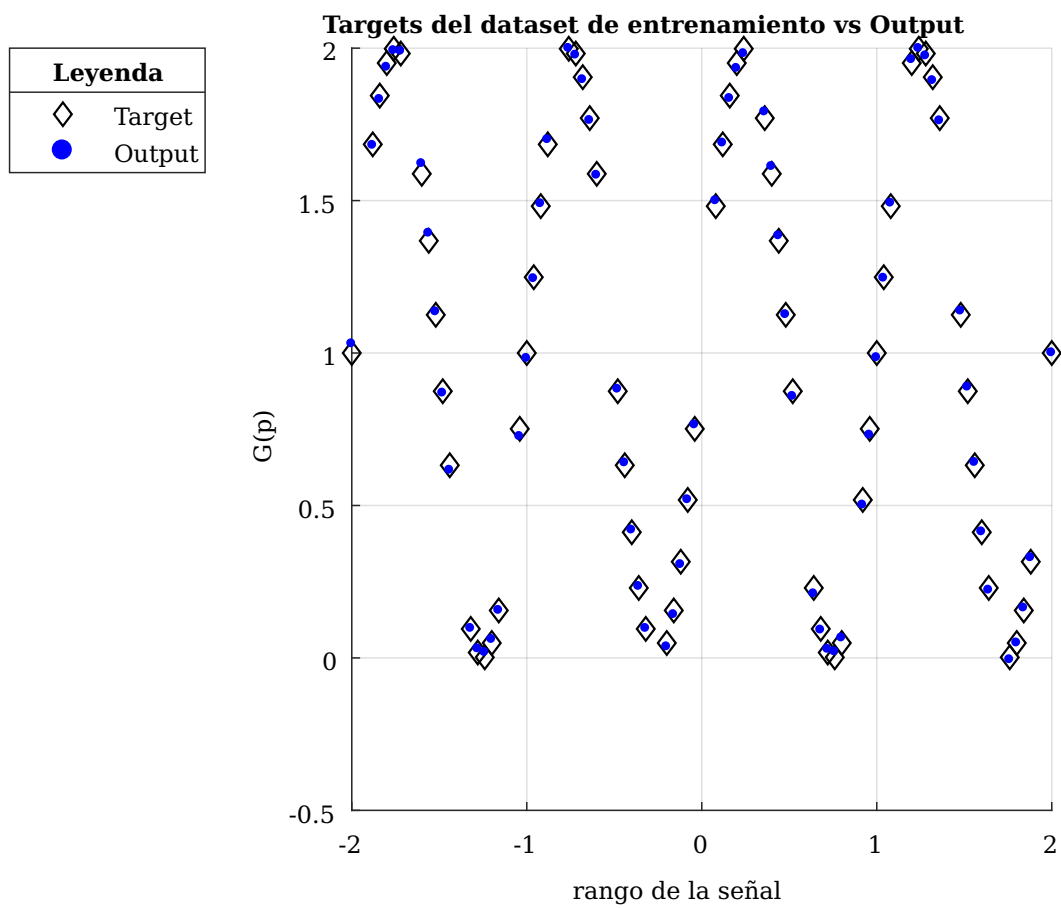


Figura 5: Gráfica 1.2



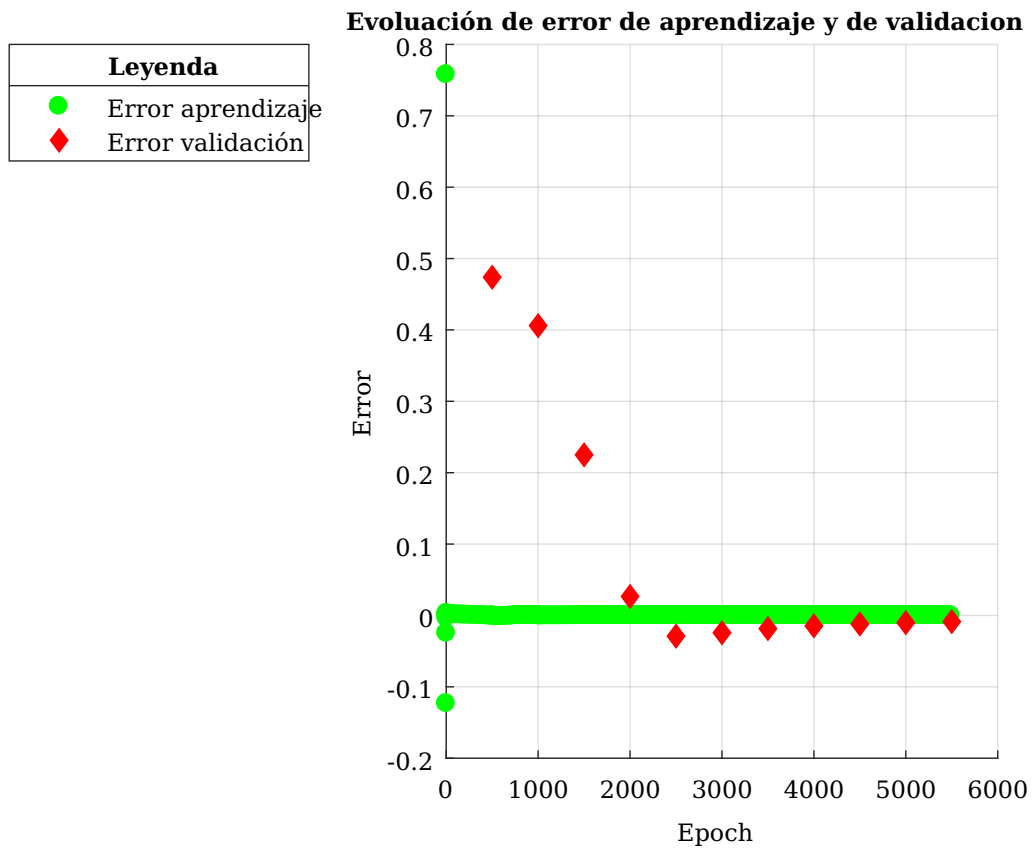


Figura 6: Gráfica 1.3

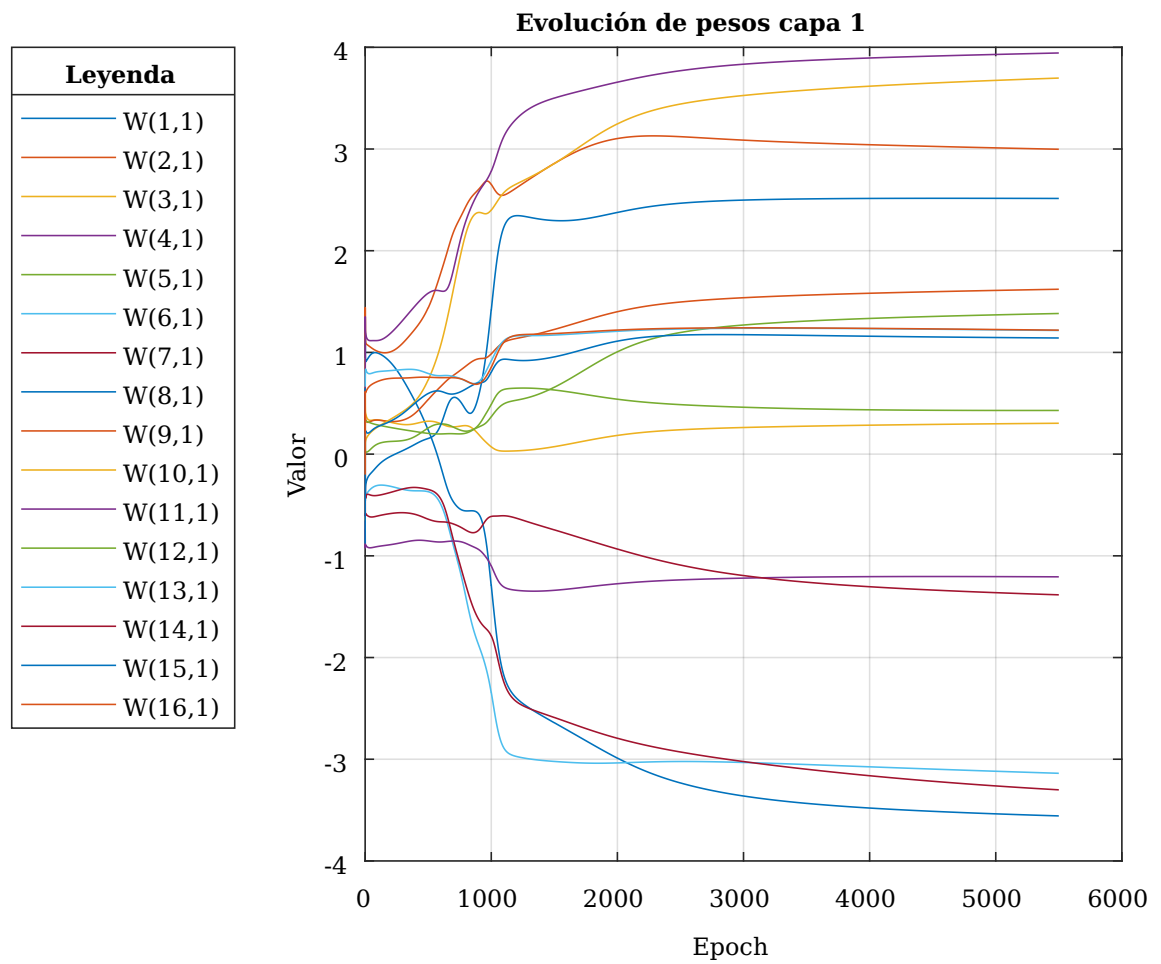


Figura 7: Gráfica 1.4

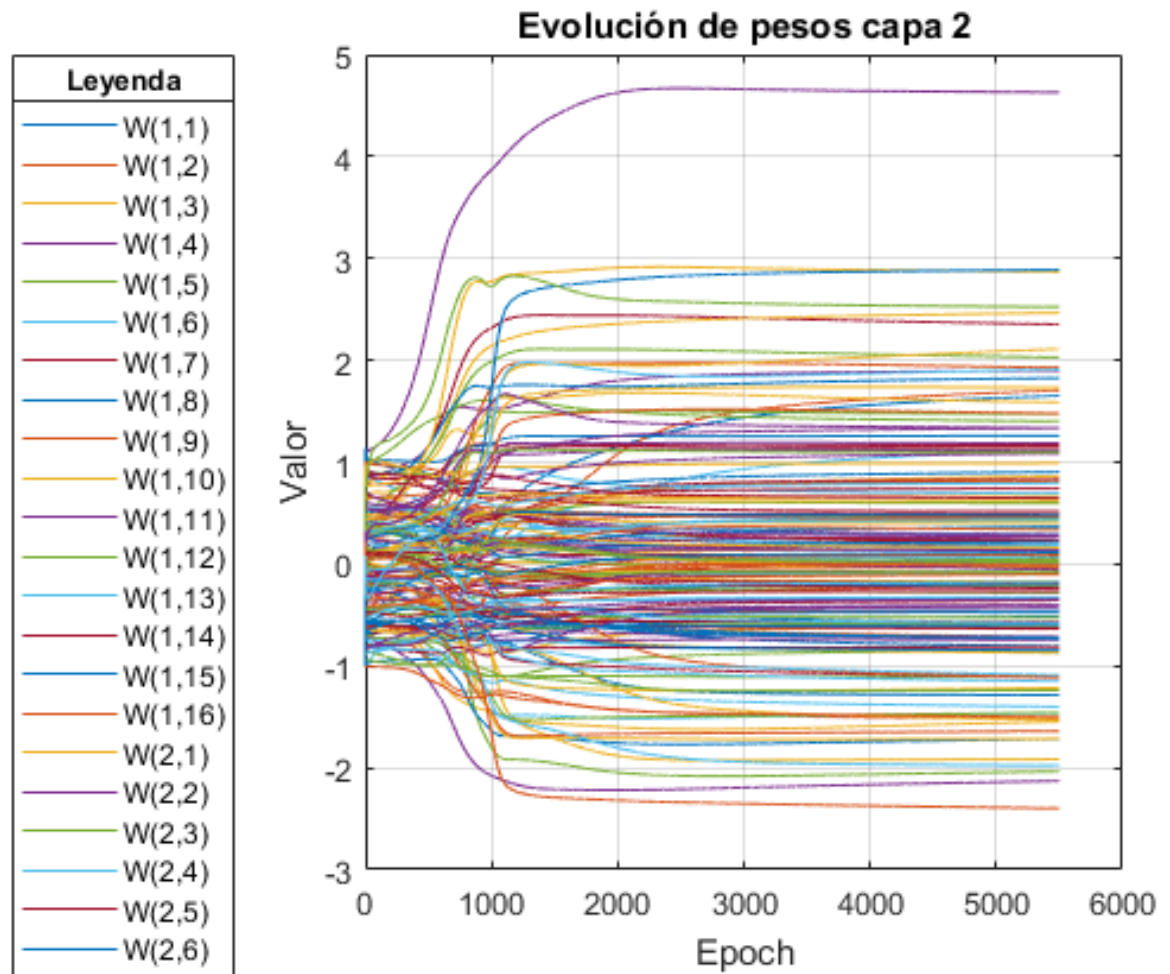


Figura 8: Gráfica 1.5

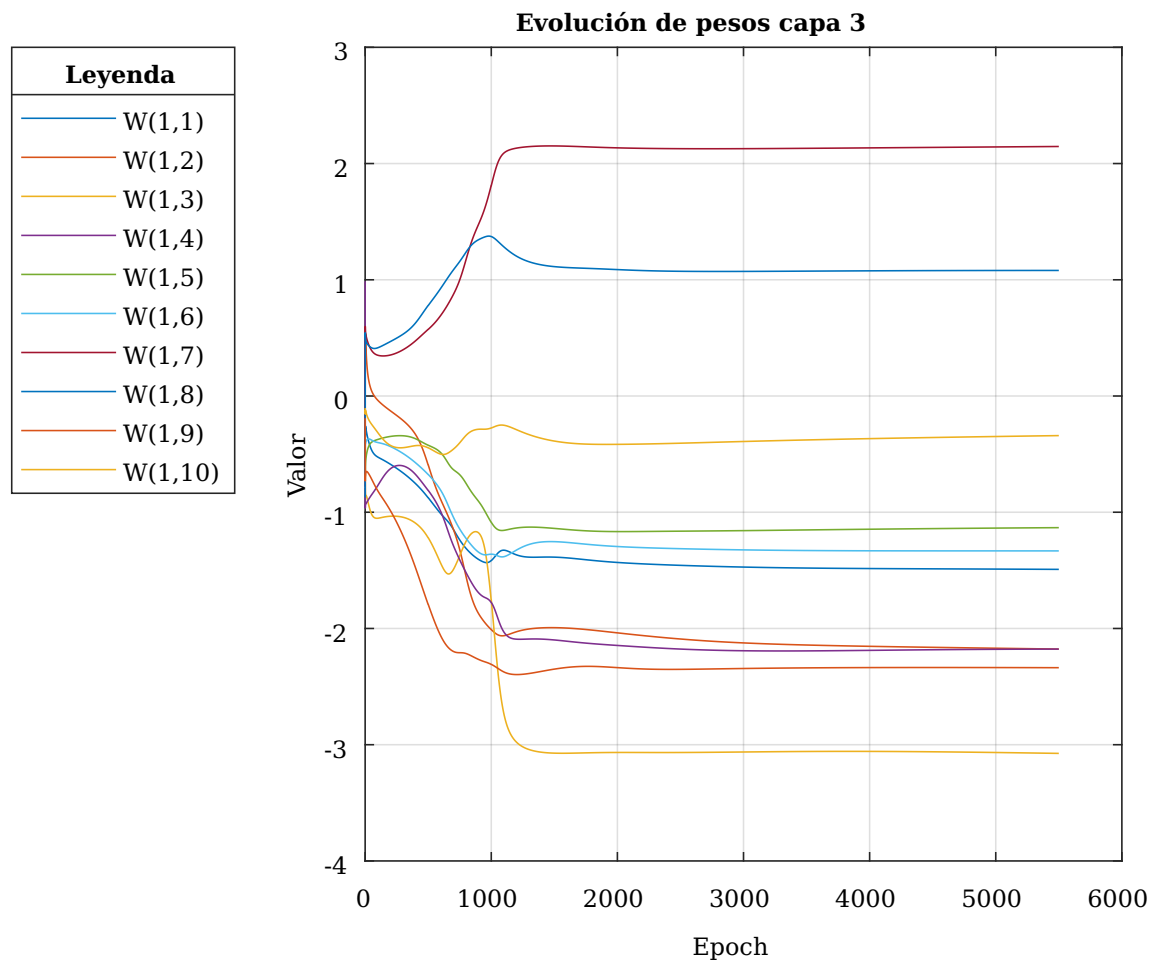


Figura 9: Gráfica 1.6

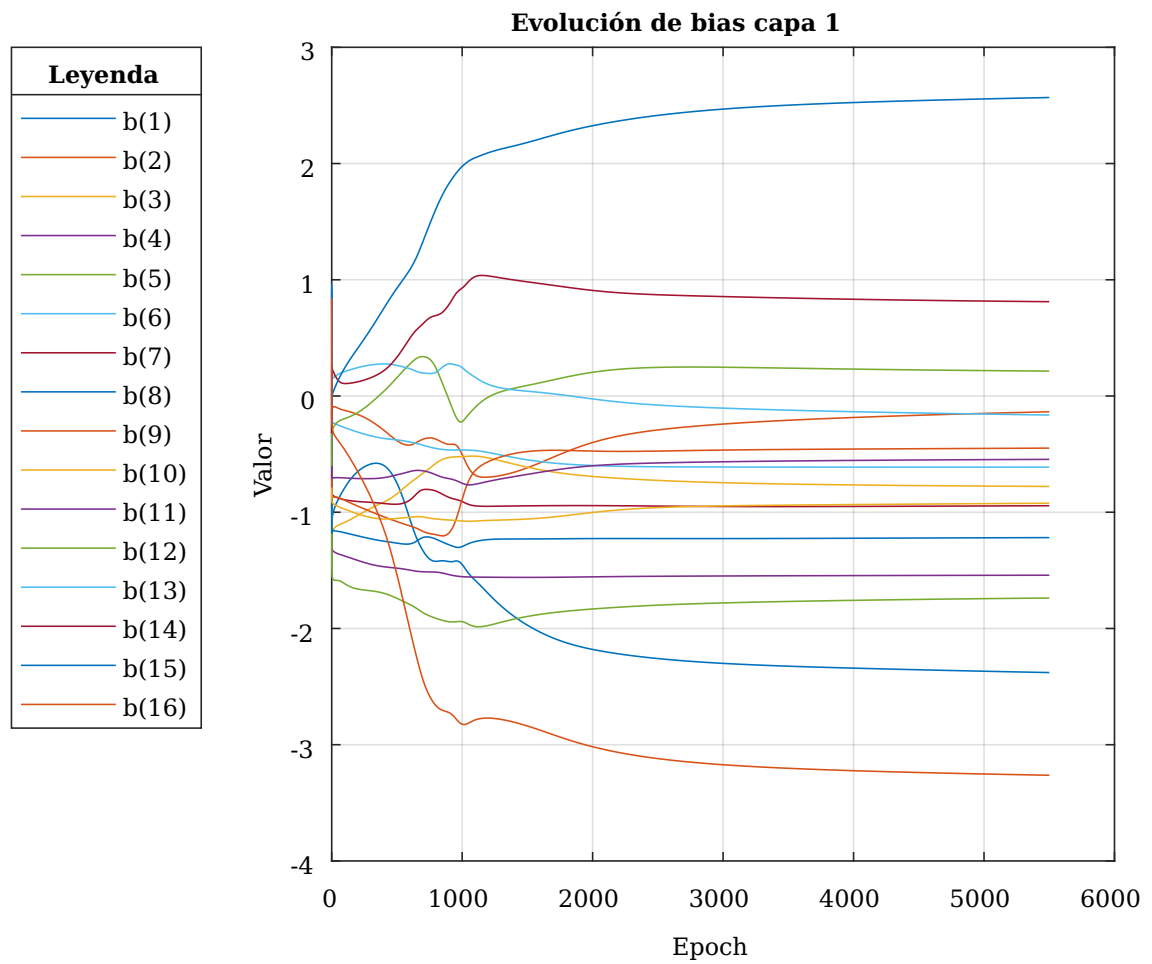


Figura 10: Gráfica 1.7

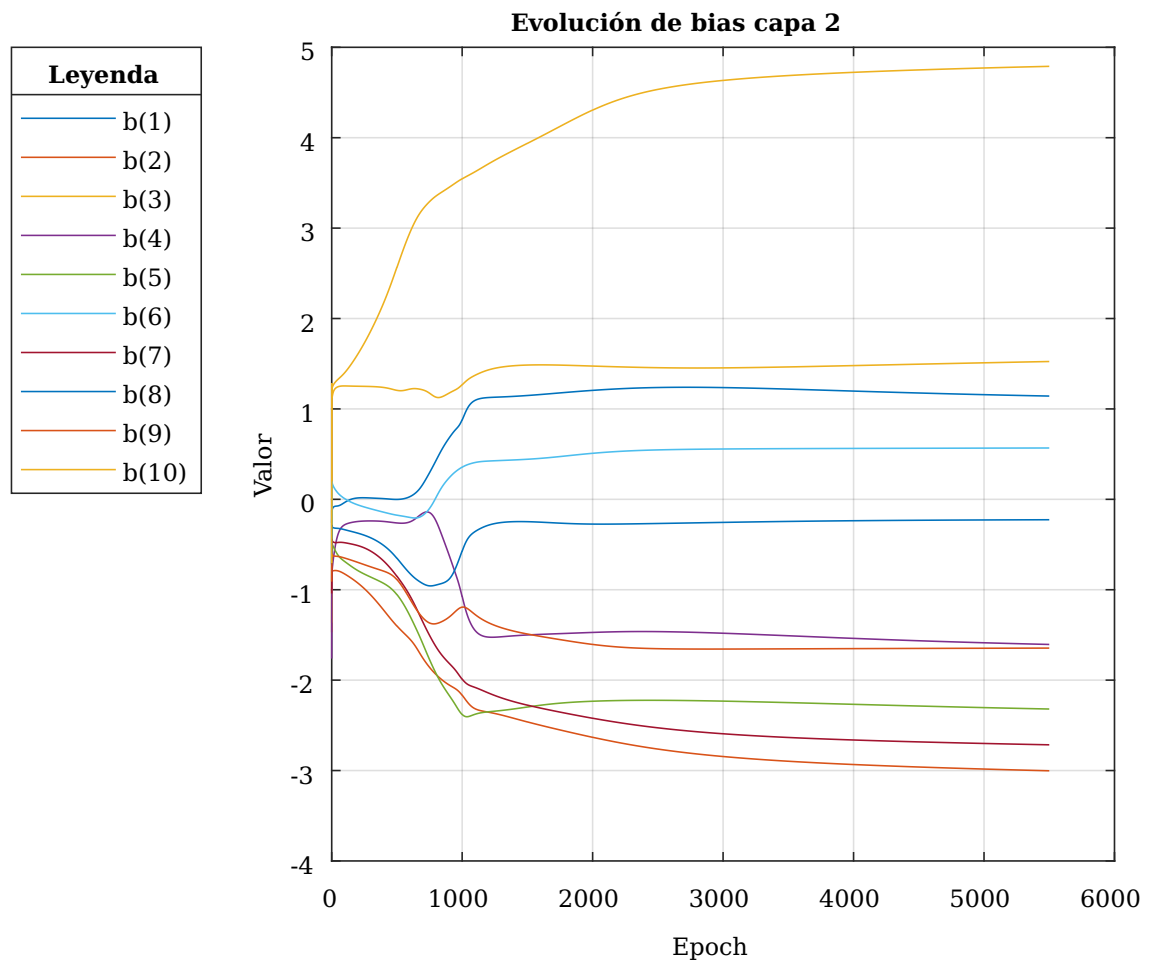


Figura 11: Gráfica 1.8

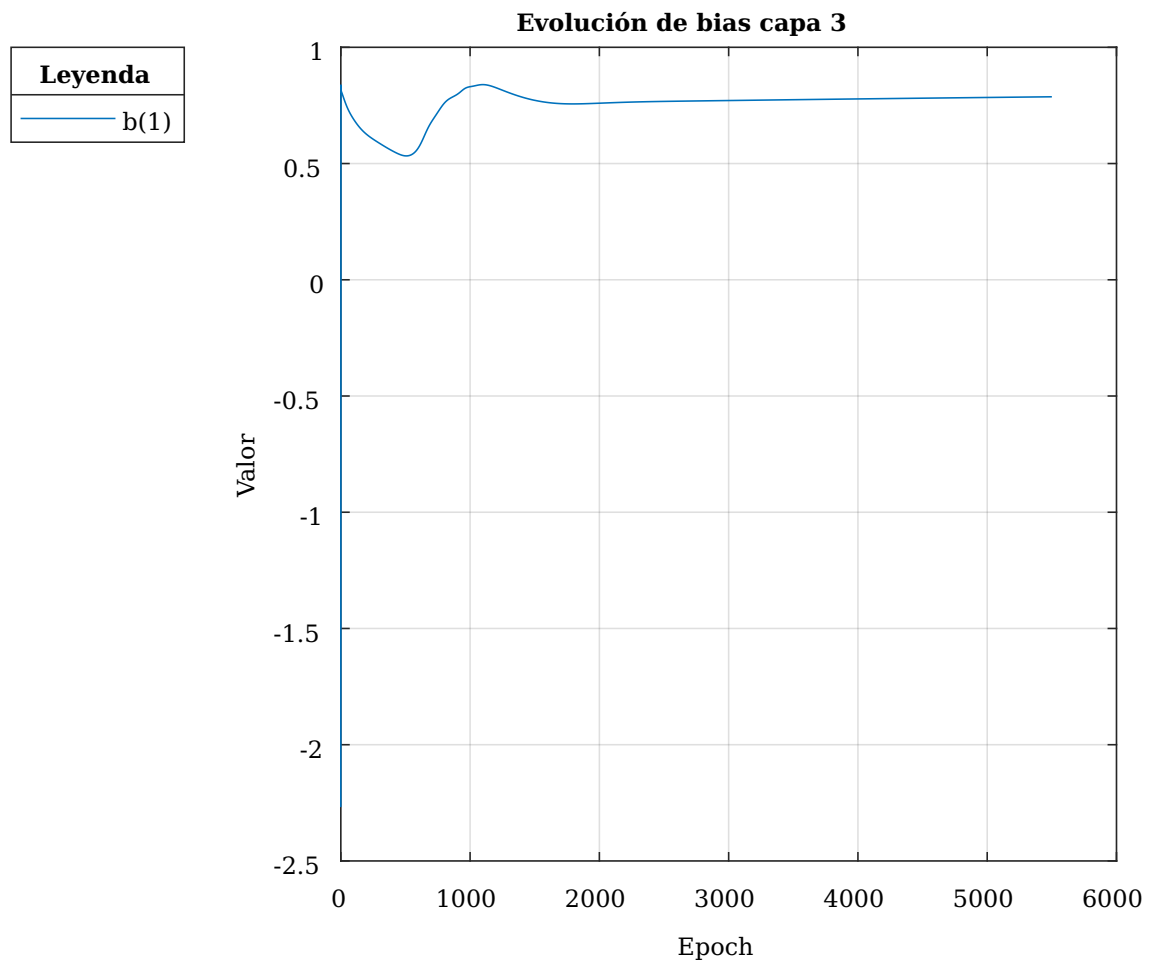


Figura 12: Gráfica 1.9

### 3.6. Polinomio 2

### 3.7. Inputs

1. 4.0000	30. 3.0333	59. 2.0667	88. 1.1000
2. 3.9667	31. 3.0000	60. 2.0333	89. 1.0667
3. 3.9333	32. 2.9667	61. 2.0000	90. 1.0333
4. 3.9000	33. 2.9333	62. 1.9667	91. 1.0000
5. 3.8667	34. 2.9000	63. 1.9333	92. 0.9667
6. 3.8333	35. 2.8667	64. 1.9000	93. 0.9333
7. 3.8000	36. 2.8333	65. 1.8667	94. 0.9000
8. 3.7667	37. 2.8000	66. 1.8333	95. 0.8667
9. 3.7333	38. 2.7667	67. 1.8000	96. 0.8333
10. 3.7000	39. 2.7333	68. 1.7667	97. 0.8000
11. 3.6667	40. 2.7000	69. 1.7333	98. 0.7667
12. 3.6333	41. 2.6667	70. 1.7000	99. 0.7333
13. 3.6000	42. 2.6333	71. 1.6667	100. 0.7000
14. 3.5667	43. 2.6000	72. 1.6333	101. 0.6667
15. 3.5333	44. 2.5667	73. 1.6000	102. 0.6333
16. 3.5000	45. 2.5333	74. 1.5667	103. 0.6000
17. 3.4667	46. 2.5000	75. 1.5333	104. 0.5667
18. 3.4333	47. 2.4667	76. 1.5000	105. 0.5333
19. 3.4000	48. 2.4333	77. 1.4667	106. 0.5000
20. 3.3667	49. 2.4000	78. 1.4333	107. 0.4667
21. 3.3333	50. 2.3667	79. 1.4000	108. 0.4333
22. 3.3000	51. 2.3333	80. 1.3667	109. 0.4000
23. 3.2667	52. 2.3000	81. 1.3333	110. 0.3667
24. 3.2333	53. 2.2667	82. 1.3000	111. 0.3333
25. 3.2000	54. 2.2333	83. 1.2667	112. 0.3000
26. 3.1667	55. 2.2000	84. 1.2333	113. 0.2667
27. 3.1333	56. 2.1667	85. 1.2000	114. 0.2333
28. 3.1000	57. 2.1333	86. 1.1667	115. 0.2000
29. 3.0667	58. 2.1000	87. 1.1333	116. 0.1667



117. 0.1333	149. -0.9333	181. -2.0000	213. -3.0667
118. 0.1000	150. -0.9667	182. -2.0333	214. -3.1000
119. 0.0667	151. -1.0000	183. -2.0667	215. -3.1333
120. 0.0333	152. -1.0333	184. -2.1000	216. -3.1667
121. 0.0000	153. -1.0667	185. -2.1333	217. -3.2000
122. -0.0333	154. -1.1000	186. -2.1667	218. -3.2333
123. -0.0667	155. -1.1333	187. -2.2000	219. -3.2667
124. -0.1000	156. -1.1667	188. -2.2333	220. -3.3000
125. -0.1333	157. -1.2000	189. -2.2667	221. -3.3333
126. -0.1667	158. -1.2333	190. -2.3000	222. -3.3667
127. -0.2000	159. -1.2667	191. -2.3333	223. -3.4000
128. -0.2333	160. -1.3000	192. -2.3667	224. -3.4333
129. -0.2667	161. -1.3333	193. -2.4000	225. -3.4667
130. -0.3000	162. -1.3667	194. -2.4333	226. -3.5000
131. -0.3333	163. -1.4000	195. -2.4667	227. -3.5333
132. -0.3667	164. -1.4333	196. -2.5000	228. -3.5667
133. -0.4000	165. -1.4667	197. -2.5333	229. -3.6000
134. -0.4333	166. -1.5000	198. -2.5667	230. -3.6333
135. -0.4667	167. -1.5333	199. -2.6000	231. -3.6667
136. -0.5000	168. -1.5667	200. -2.6333	232. -3.7000
137. -0.5333	169. -1.6000	201. -2.6667	233. -3.7333
138. -0.5667	170. -1.6333	202. -2.7000	234. -3.7667
139. -0.6000	171. -1.6667	203. -2.7333	235. -3.8000
140. -0.6333	172. -1.7000	204. -2.7667	236. -3.8333
141. -0.6667	173. -1.7333	205. -2.8000	237. -3.8667
142. -0.7000	174. -1.7667	206. -2.8333	238. -3.9000
143. -0.7333	175. -1.8000	207. -2.8667	239. -3.9333
144. -0.7667	176. -1.8333	208. -2.9000	240. -3.9667
145. -0.8000	177. -1.8667	209. -2.9333	241. -4.0000
146. -0.8333	178. -1.9000	210. -2.9667	
147. -0.8667	179. -1.9333	211. -3.0000	
148. -0.9000	180. -1.9667	212. -3.0333	

**3.8. Targets**

1. -0.01292	31. -0.24440	61. -1.33982	91. -1.33437
2. -0.01451	32. -0.26413	62. -1.38403	92. -1.26546
3. -0.01628	33. -0.28507	63. -1.42719	93. -1.19232
4. -0.01825	34. -0.30725	64. -1.46909	94. -1.11519
5. -0.02043	35. -0.33070	65. -1.50947	95. -1.03433
6. -0.02284	36. -0.35544	66. -1.54811	96. -0.95005
7. -0.02550	37. -0.38150	67. -1.58477	97. -0.86267
8. -0.02844	38. -0.40891	68. -1.61920	98. -0.77252
9. -0.03167	39. -0.43767	69. -1.65116	99. -0.67999
10. -0.03523	40. -0.46778	70. -1.68040	100. -0.58546
11. -0.03914	41. -0.49926	71. -1.70668	101. -0.48934
12. -0.04342	42. -0.53210	72. -1.72976	102. -0.39205
13. -0.04812	43. -0.56628	73. -1.74941	103. -0.29402
14. -0.05325	44. -0.60178	74. -1.76541	104. -0.19569
15. -0.05886	45. -0.63858	75. -1.77754	105. -0.09754
16. -0.06497	46. -0.67663	76. -1.78559	106. 0.00000
17. -0.07162	47. -0.71589	77. -1.78938	107. 0.09646
18. -0.07886	48. -0.75630	78. -1.78873	108. 0.19138
19. -0.08671	49. -0.79779	79. -1.78348	109. 0.28432
20. -0.09522	50. -0.84028	80. -1.77349	110. 0.37483
21. -0.10442	51. -0.88368	81. -1.75865	111. 0.46247
22. -0.11437	52. -0.92790	82. -1.73885	112. 0.54683
23. -0.12510	53. -0.97281	83. -1.71402	113. 0.62751
24. -0.13666	54. -1.01829	84. -1.68412	114. 0.70412
25. -0.14909	55. -1.06421	85. -1.64912	115. 0.77632
26. -0.16243	56. -1.11042	86. -1.60902	116. 0.84375
27. -0.17674	57. -1.15676	87. -1.56387	117. 0.90613
28. -0.19204	58. -1.20305	88. -1.51372	118. 0.96317
29. -0.20839	59. -1.24912	89. -1.45866	119. 1.01463
30. -0.22583	60. -1.29478	90. -1.39883	120. 1.06030
			121. 1.10000
			122. 1.13359
			123. 1.16097

124. 1.18207	154. -0.19222	184. -0.69370	214. -0.13619
125. 1.19687	155. -0.25207	185. -0.67457	215. -0.12586
126. 1.20536	156. -0.30943	186. -0.65456	216. -0.11614
127. 1.20760	157. -0.36409	187. -0.63383	217. -0.10702
128. 1.20367	158. -0.41589	188. -0.61251	218. -0.09847
129. 1.19368	159. -0.46467	189. -0.59073	219. -0.09048
130. 1.17779	160. -0.51031	190. -0.56861	220. -0.08302
131. 1.15617	161. -0.55272	191. -0.54628	221. -0.07607
132. 1.12905	162. -0.59180	192. -0.52384	222. -0.06961
133. 1.09666	163. -0.62752	193. -0.50140	223. -0.06360
134. 1.05928	164. -0.65983	194. -0.47905	224. -0.05804
135. 1.01720	165. -0.68874	195. -0.45688	225. -0.05288
136. 0.97075	166. -0.71424	196. -0.43498	226. -0.04812
137. 0.92025	167. -0.73636	197. -0.41341	227. -0.04373
138. 0.86605	168. -0.75517	198. -0.39224	228. -0.03969
139. 0.80854	169. -0.77072	199. -0.37153	229. -0.03597
140. 0.74809	170. -0.78309	200. -0.35132	230. -0.03256
141. 0.68508	171. -0.79239	201. -0.33168	231. -0.02943
142. 0.61990	172. -0.79871	202. -0.31262	232. -0.02656
143. 0.55296	173. -0.80218	203. -0.29419	233. -0.02395
144. 0.48465	174. -0.80293	204. -0.27640	234. -0.02156
145. 0.41536	175. -0.80109	205. -0.25928	235. -0.01938
146. 0.34547	176. -0.79682	206. -0.24285	236. -0.01741
147. 0.27537	177. -0.79027	207. -0.22710	237. -0.01561
148. 0.20543	178. -0.78158	208. -0.21205	238. -0.01398
149. 0.13599	179. -0.77093	209. -0.19770	239. -0.01250
150. 0.06741	180. -0.75846	210. -0.18405	240. -0.01117
151. 0.00000	181. -0.74434	211. -0.17108	241. -0.00996
152. -0.06593	182. -0.72874	212. -0.15879	
153. -0.13009	183. -0.71180	213. -0.14717	

### 3.8.1. Datos

$$V1 = [11 \ 16 \ 10 \ 1]$$

$$V2 = [3 \ 2 \ 1]$$

epochmax = 2200

alpha=.0103

Múltiplo para las épocas de validación = 500

numval = 7

error de validación = .000000000001

Configuración: 80-10-10

### 3.9. Resultado

### 3.10. Imágenes

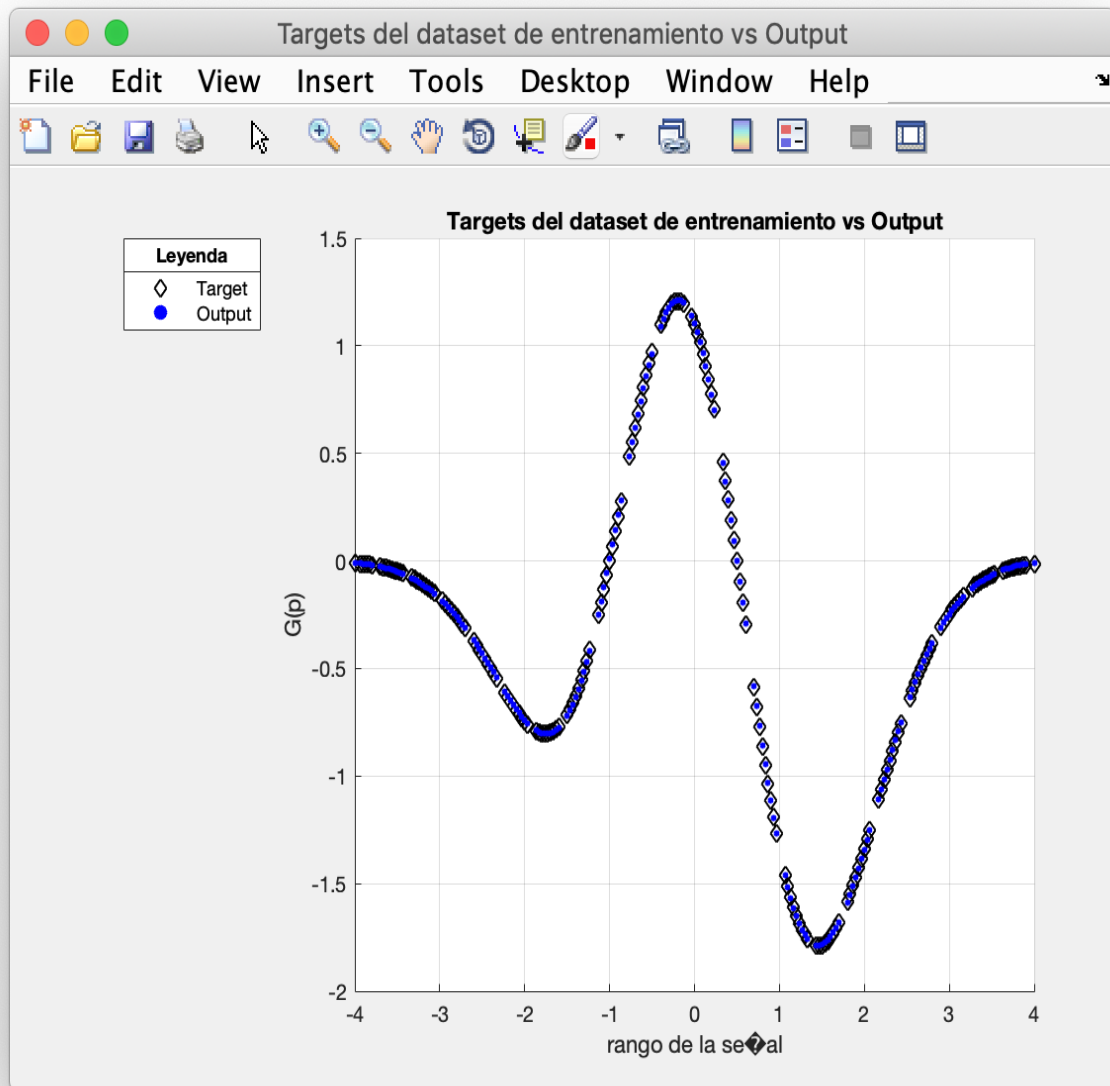


Figura 13: Gráfica 1.1

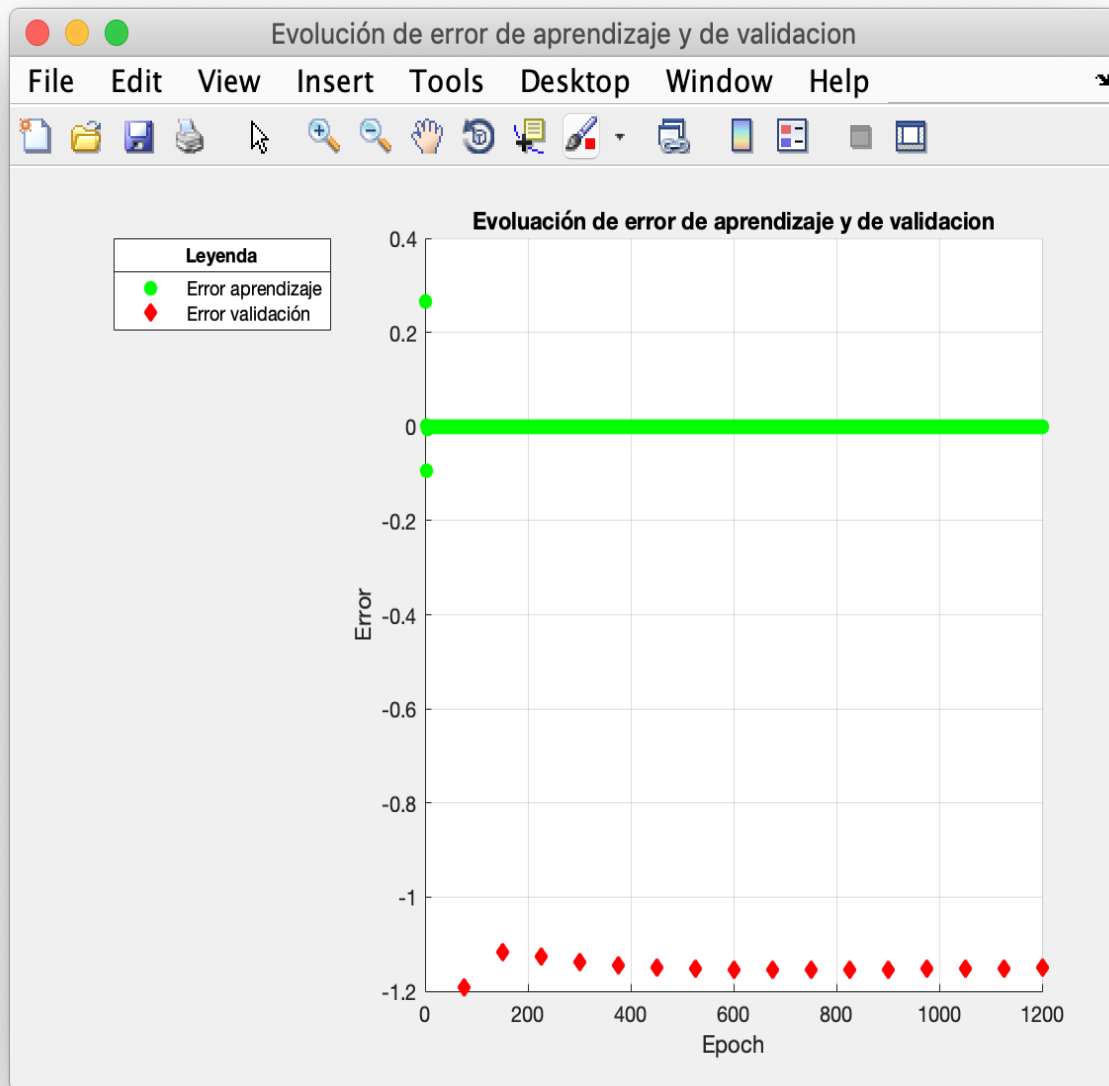


Figura 14: Gráfica 1.2

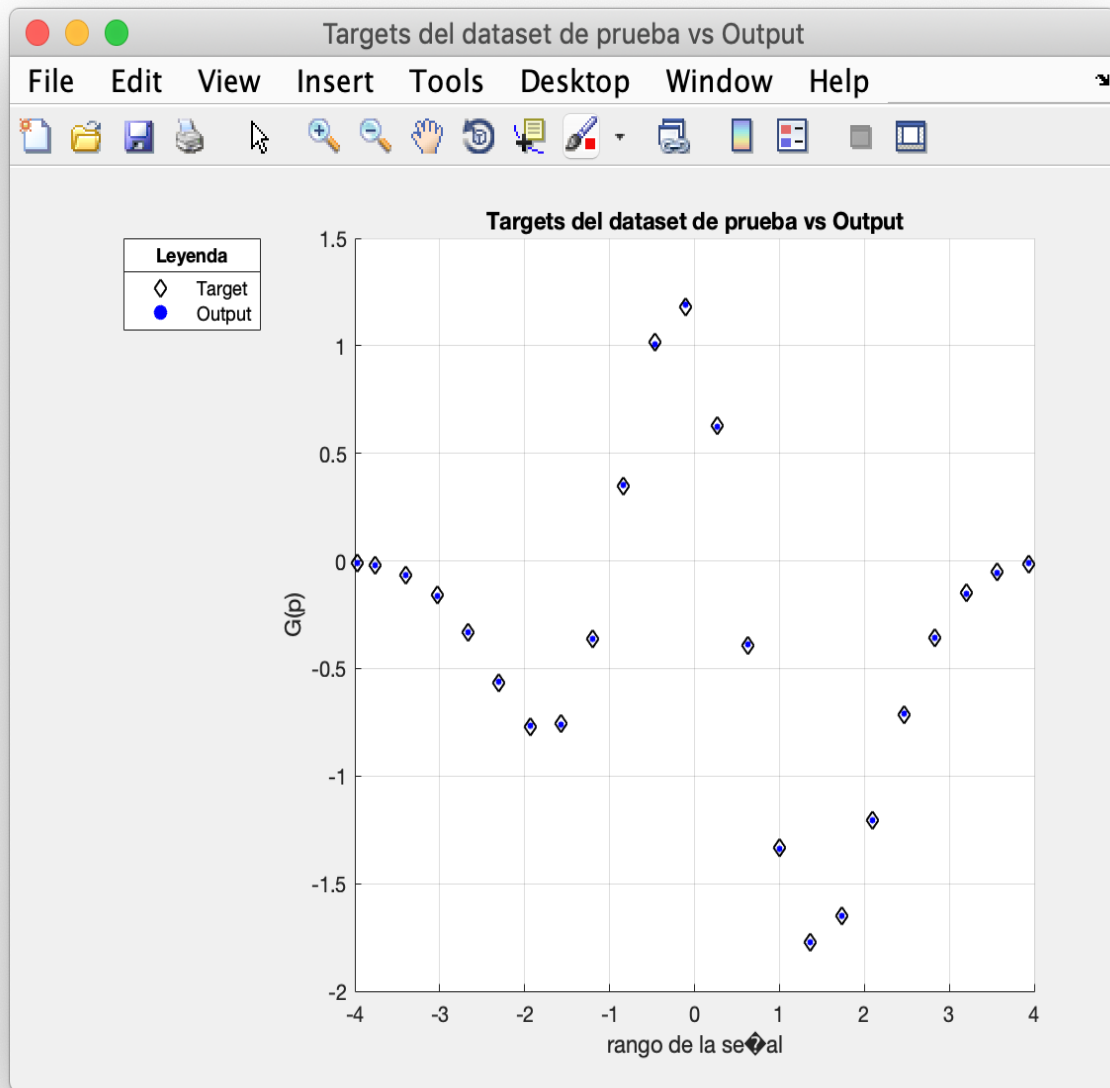


Figura 15: Gráfica 1.3

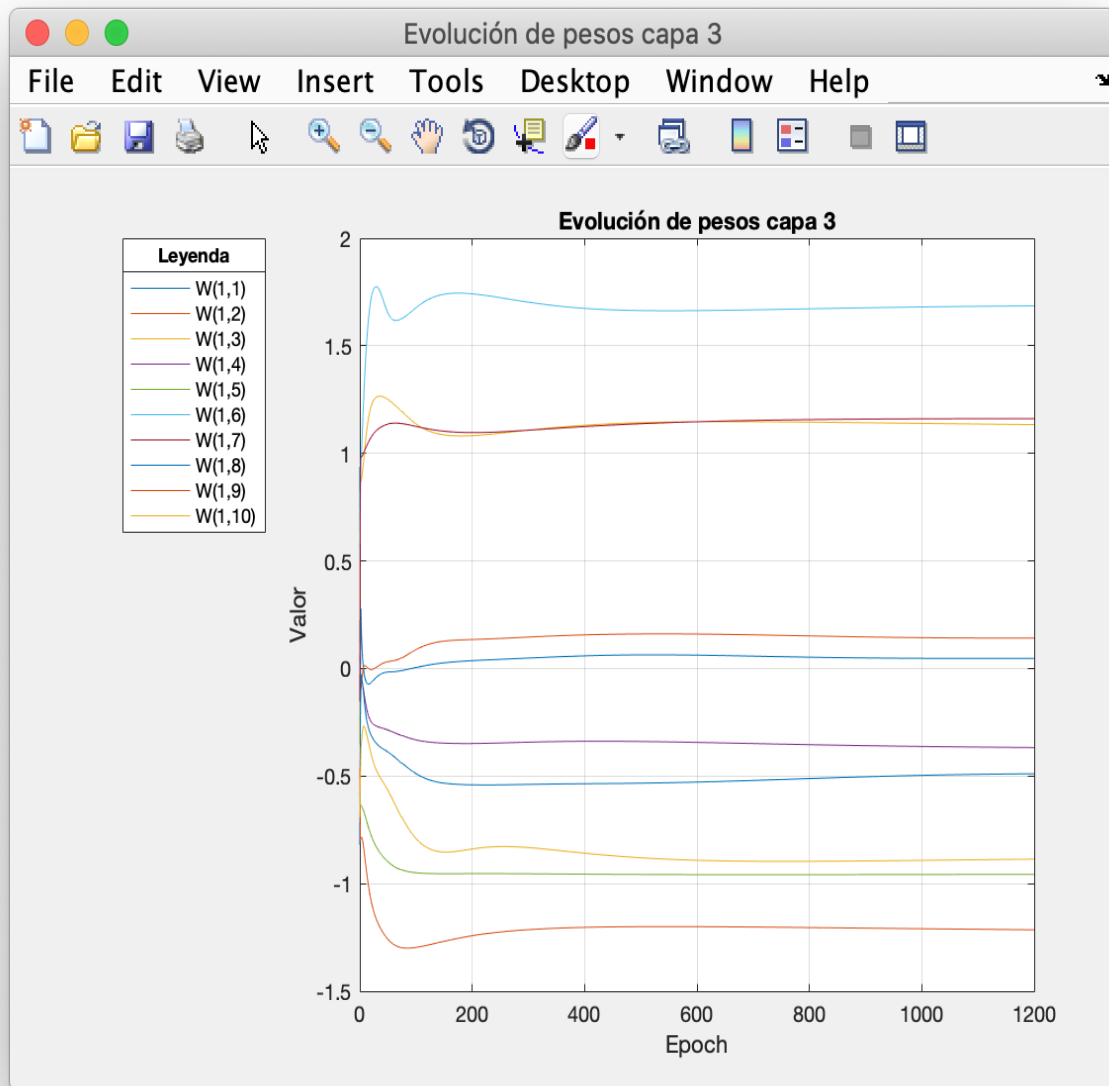


Figura 16: Gráfica 1.4



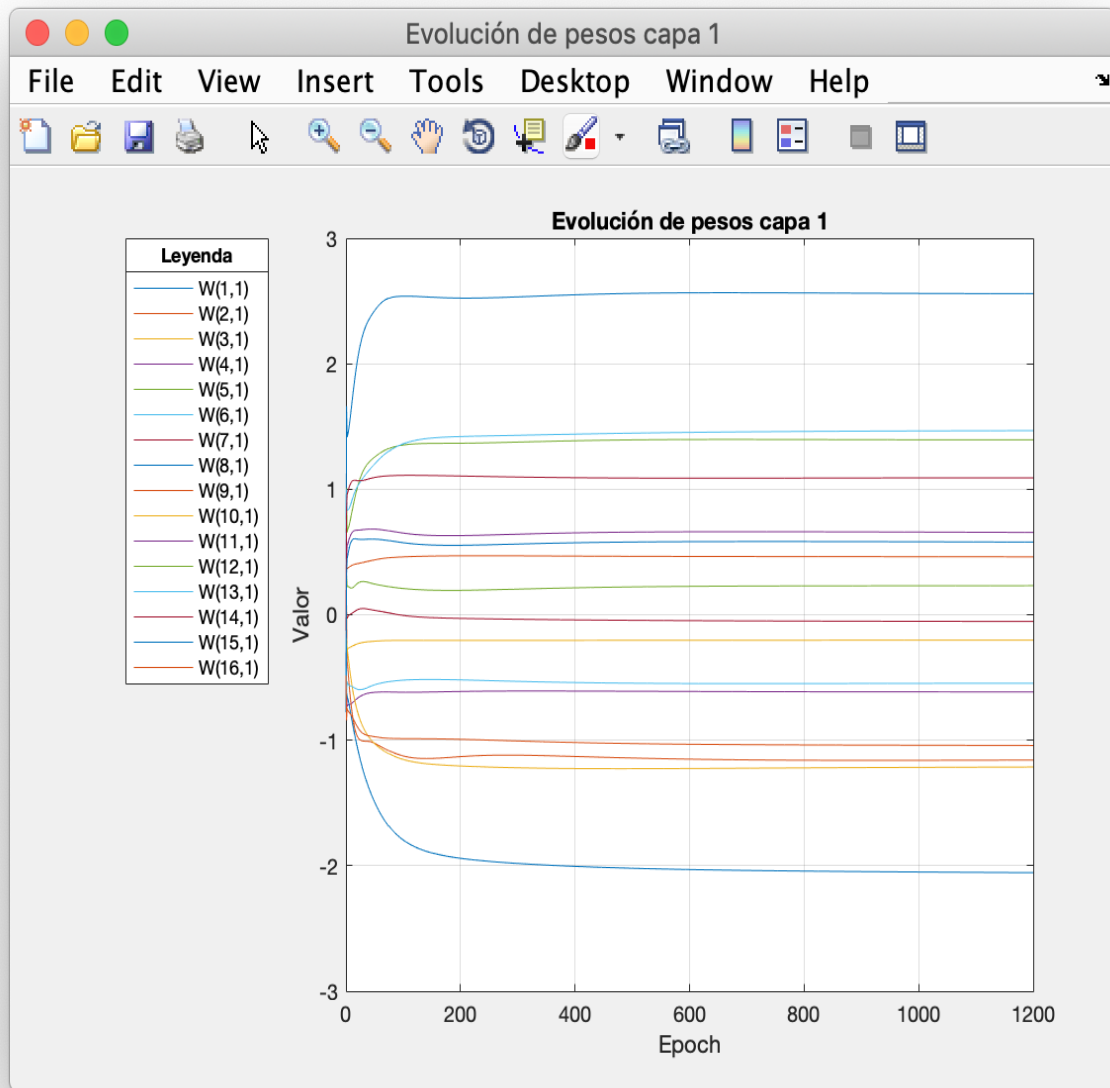


Figura 17: Gráfica 1.5

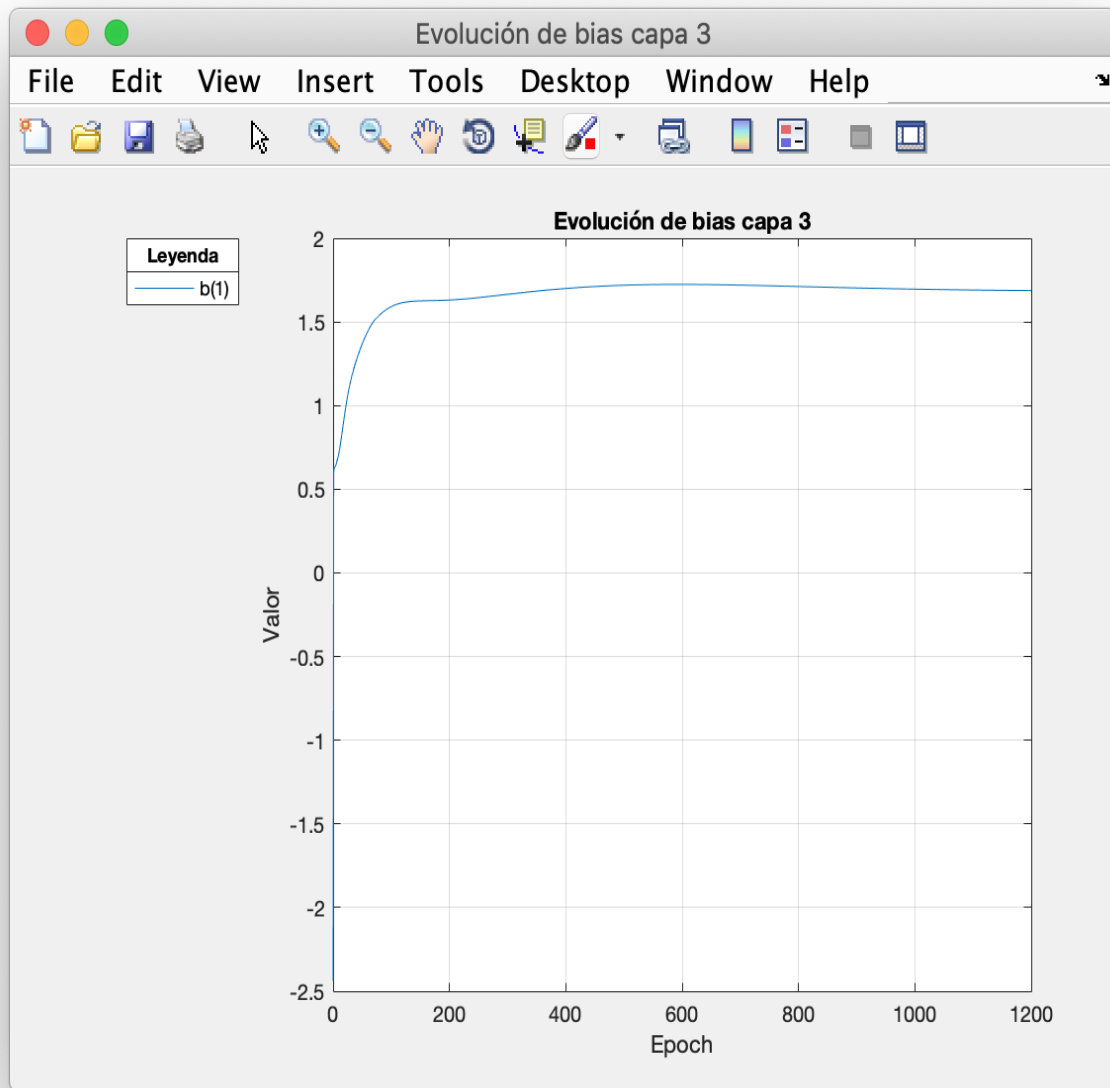


Figura 18: Gráfica 1.6

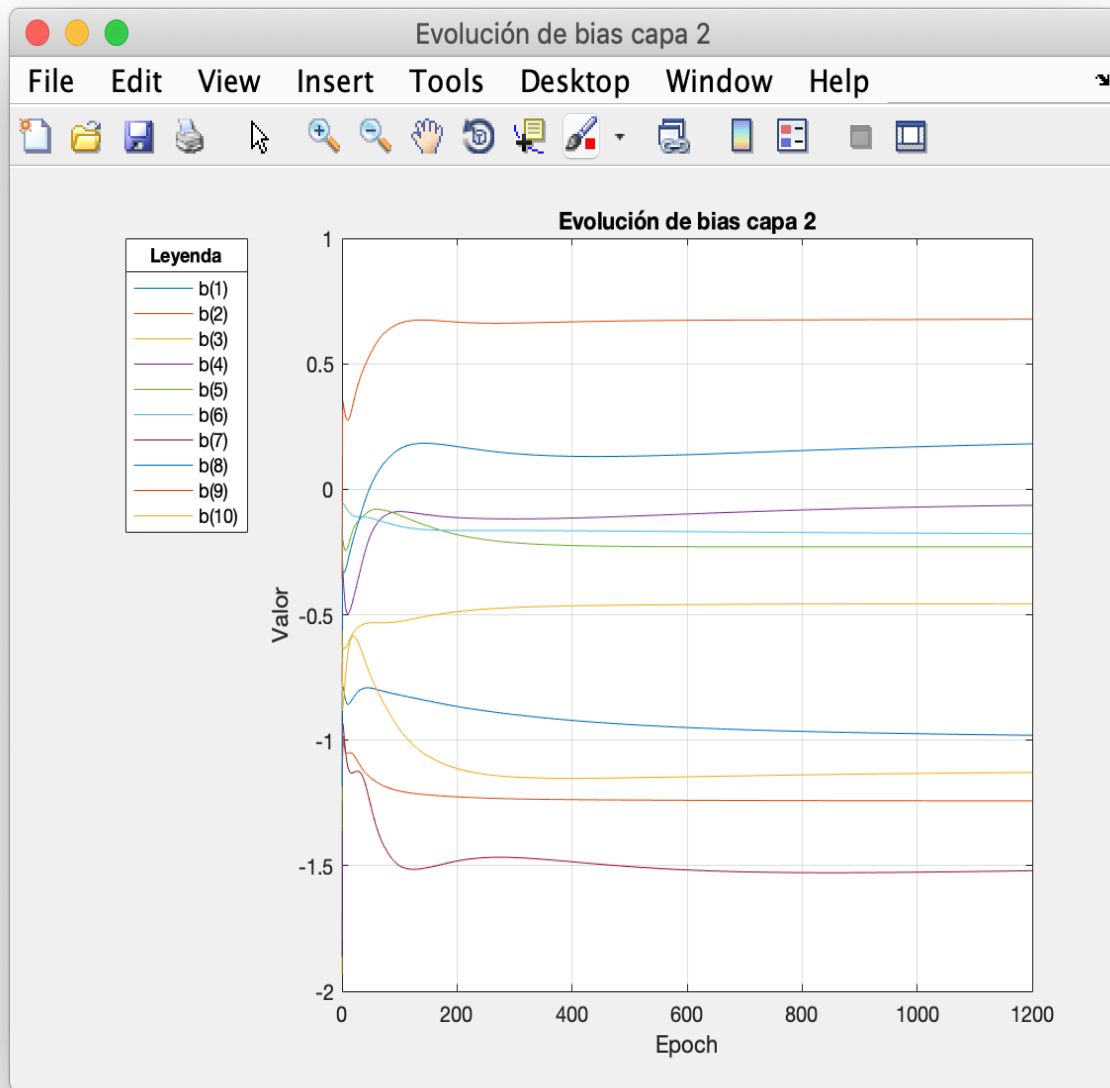


Figura 19: Gráfica 1.7

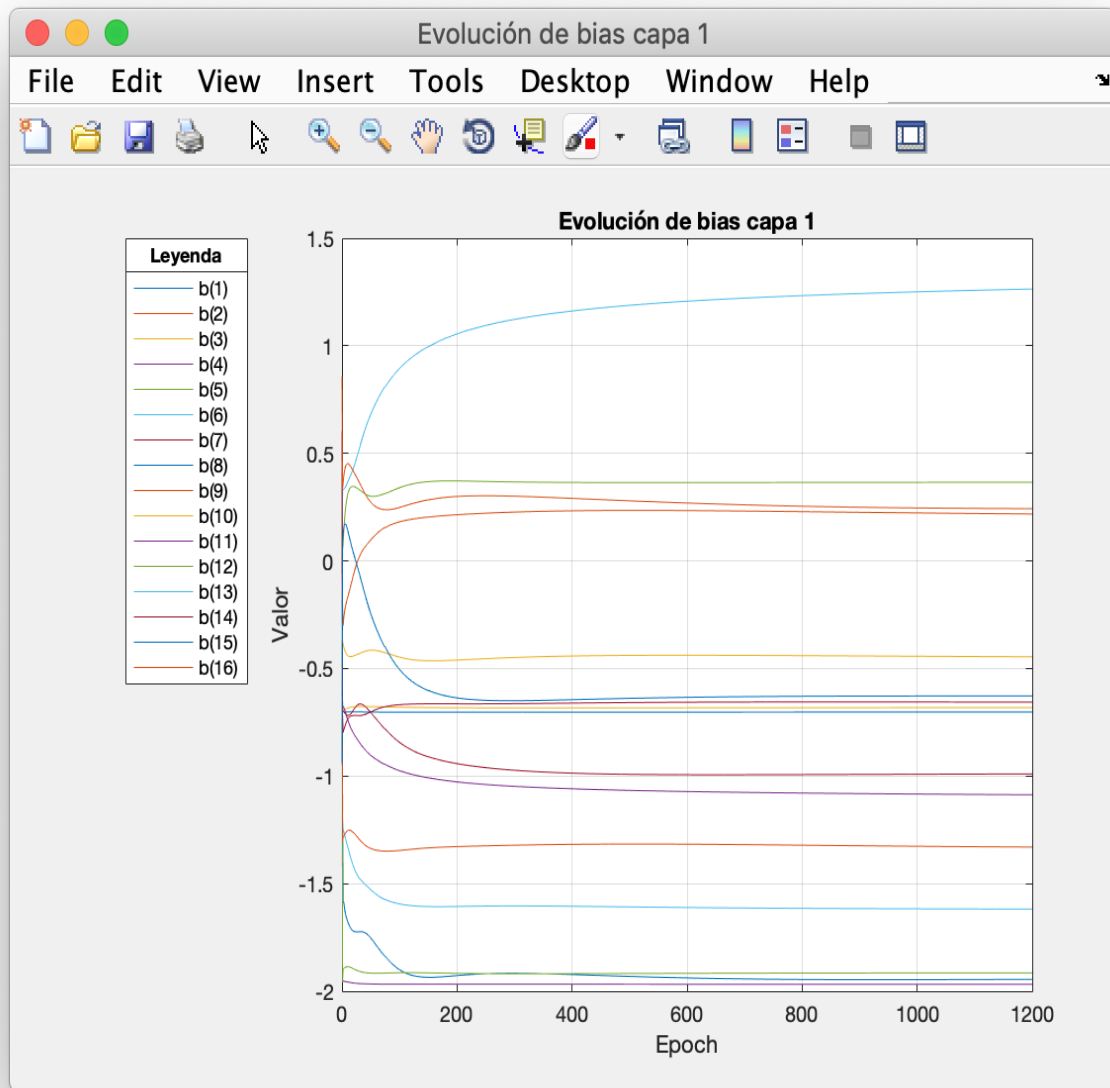


Figura 20: Gráfica 1.8

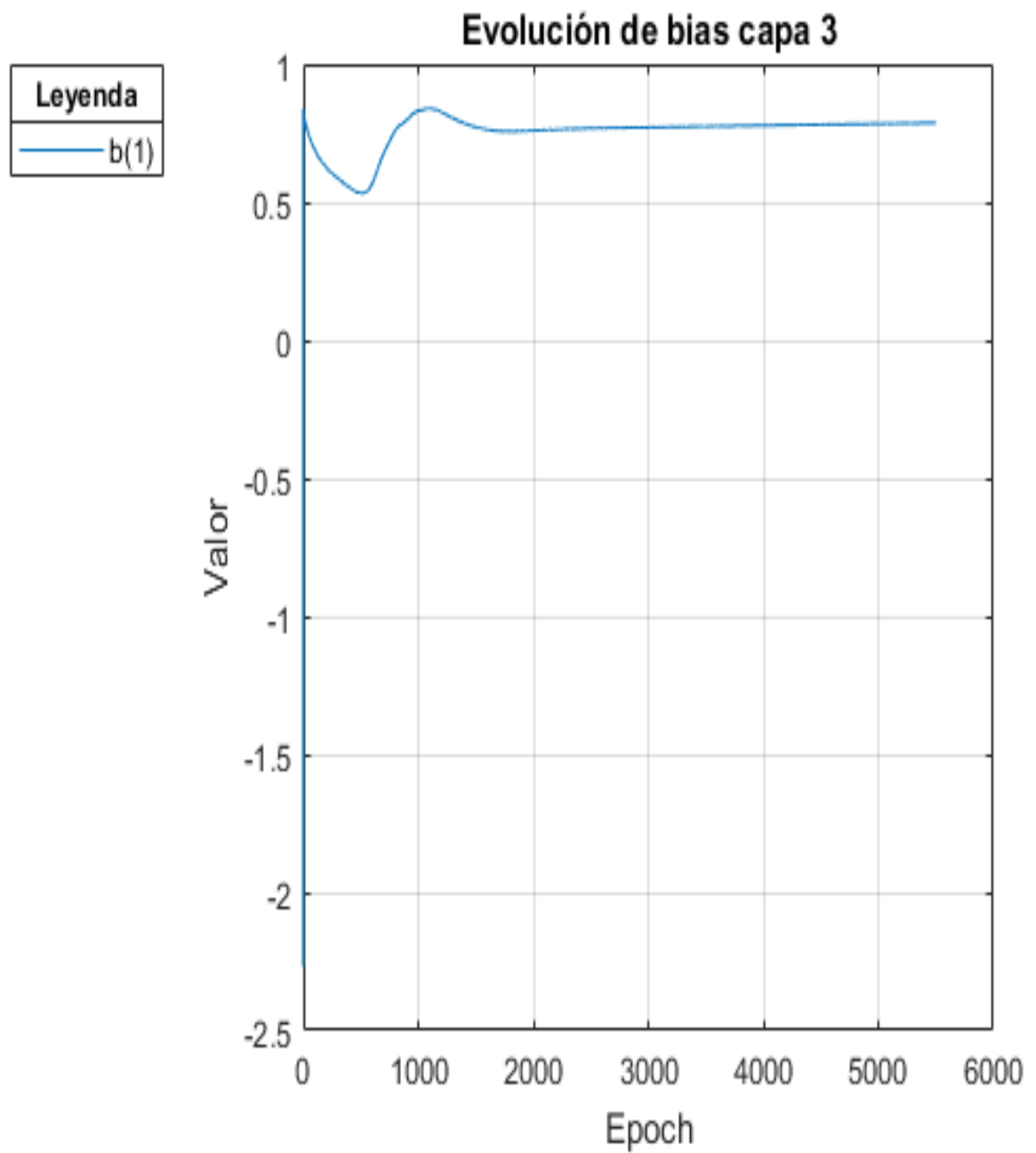


Figura 21: Gráfica 1.9

### 3.11. Polinomio 3

### 3.12. Inputs

1. -3.5	30. -3.21	59. -2.92	88. -2.63
2. -3.49	31. -3.2	60. -2.91	89. -2.62
3. -3.48	32. -3.19	61. -2.9	90. -2.61
4. -3.47	33. -3.18	62. -2.89	91. -2.6
5. -3.46	34. -3.17	63. -2.88	92. -2.59
6. -3.45	35. -3.16	64. -2.87	93. -2.58
7. -3.44	36. -3.15	65. -2.86	94. -2.57
8. -3.43	37. -3.14	66. -2.85	95. -2.56
9. -3.42	38. -3.13	67. -2.84	96. -2.55
10. -3.41	39. -3.12	68. -2.83	97. -2.54
11. -3.4	40. -3.11	69. -2.82	98. -2.53
12. -3.39	41. -3.1	70. -2.81	99. -2.52
13. -3.38	42. -3.09	71. -2.8	100. -2.51
14. -3.37	43. -3.08	72. -2.79	101. -2.5
15. -3.36	44. -3.07	73. -2.78	102. -2.49
16. -3.35	45. -3.06	74. -2.77	103. -2.48
17. -3.34	46. -3.05	75. -2.76	104. -2.47
18. -3.33	47. -3.04	76. -2.75	105. -2.46
19. -3.32	48. -3.03	77. -2.74	106. -2.45
20. -3.31	49. -3.02	78. -2.73	107. -2.44
21. -3.3	50. -3.01	79. -2.72	108. -2.43
22. -3.29	51. -3	80. -2.71	109. -2.42
23. -3.28	52. -2.99	81. -2.7	110. -2.41
24. -3.27	53. -2.98	82. -2.69	111. -2.4
25. -3.26	54. -2.97	83. -2.68	112. -2.39
26. -3.25	55. -2.96	84. -2.67	113. -2.38
27. -3.24	56. -2.95	85. -2.66	114. -2.37
28. -3.23	57. -2.94	86. -2.65	115. -2.36
29. -3.22	58. -2.93	87. -2.64	116. -2.35

117. -2.34	148. -2.03	179. -1.72	210. -1.41
118. -2.33	149. -2.02	180. -1.71	211. -1.4
119. -2.32	150. -2.01	181. -1.7	212. -1.39
120. -2.31	151. -2	182. -1.69	213. -1.38
121. -2.3	152. -1.99	183. -1.68	214. -1.37
122. -2.29	153. -1.98	184. -1.67	215. -1.36
123. -2.28	154. -1.97	185. -1.66	216. -1.35
124. -2.27	155. -1.96	186. -1.65	217. -1.34
125. -2.26	156. -1.95	187. -1.64	218. -1.33
126. -2.25	157. -1.94	188. -1.63	219. -1.32
127. -2.24	158. -1.93	189. -1.62	220. -1.31
128. -2.23	159. -1.92	190. -1.61	221. -1.3
129. -2.22	160. -1.91	191. -1.6	222. -1.29
130. -2.21	161. -1.9	192. -1.59	223. -1.28
131. -2.2	162. -1.89	193. -1.58	224. -1.27
132. -2.19	163. -1.88	194. -1.57	225. -1.26
133. -2.18	164. -1.87	195. -1.56	226. -1.25
134. -2.17	165. -1.86	196. -1.55	227. -1.24
135. -2.16	166. -1.85	197. -1.54	228. -1.23
136. -2.15	167. -1.84	198. -1.53	229. -1.22
137. -2.14	168. -1.83	199. -1.52	230. -1.21
138. -2.13	169. -1.82	200. -1.51	231. -1.2
139. -2.12	170. -1.81	201. -1.5	232. -1.19
140. -2.11	171. -1.8	202. -1.49	233. -1.18
141. -2.1	172. -1.79	203. -1.48	234. -1.17
142. -2.09	173. -1.78	204. -1.47	235. -1.16
143. -2.08	174. -1.77	205. -1.46	236. -1.15
144. -2.07	175. -1.76	206. -1.45	237. -1.14
145. -2.06	176. -1.75	207. -1.44	238. -1.13
146. -2.05	177. -1.74	208. -1.43	239. -1.12
147. -2.04	178. -1.73	209. -1.42	240. -1.11

241. -1.1	272. -0.79	303. -0.48	334. -0.17
242. -1.09	273. -0.78	304. -0.47	335. -0.16
243. -1.08	274. -0.77	305. -0.46	336. -0.15
244. -1.07	275. -0.76	306. -0.45	337. -0.14
245. -1.06	276. -0.75	307. -0.44	338. -0.13
246. -1.05	277. -0.74	308. -0.43	339. -0.12
247. -1.04	278. -0.73	309. -0.42	340. -0.11
248. -1.03	279. -0.72	310. -0.41	341. -0.1
249. -1.02	280. -0.71	311. -0.4	342. -0.09
250. -1.01	281. -0.7	312. -0.39	343. -0.08
251. -1	282. -0.69	313. -0.38	344. -0.07
252. -0.99	283. -0.68	314. -0.37	345. -0.06
253. -0.98	284. -0.67	315. -0.36	346. -0.05
254. -0.97	285. -0.66	316. -0.35	347. -0.04
255. -0.96	286. -0.65	317. -0.34	348. -0.03
256. -0.95	287. -0.64	318. -0.33	349. -0.02
257. -0.94	288. -0.63	319. -0.32	350. -0.01
258. -0.93	289. -0.62	320. -0.31	351. 0
259. -0.92	290. -0.61	321. -0.3	352. 0.01
260. -0.91	291. -0.6	322. -0.29	353. 0.02
261. -0.9	292. -0.59	323. -0.28	354. 0.03
262. -0.89	293. -0.58	324. -0.27	355. 0.04
263. -0.88	294. -0.57	325. -0.26	356. 0.05
264. -0.87	295. -0.56	326. -0.25	357. 0.06
265. -0.86	296. -0.55	327. -0.24	358. 0.07
266. -0.85	297. -0.54	328. -0.23	359. 0.08
267. -0.84	298. -0.53	329. -0.22	360. 0.09
268. -0.83	299. -0.52	330. -0.21	361. 0.1
269. -0.82	300. -0.51	331. -0.2	362. 0.11
270. -0.81	301. -0.5	332. -0.19	363. 0.12
271. -0.8	302. -0.49	333. -0.18	364. 0.13



365. 0.14	396. 0.45	427. 0.76	458. 1.07
366. 0.15	397. 0.46	428. 0.77	459. 1.08
367. 0.16	398. 0.47	429. 0.78	460. 1.09
368. 0.17	399. 0.48	430. 0.79	461. 1.1
369. 0.18	400. 0.49	431. 0.8	462. 1.11
370. 0.19	401. 0.5	432. 0.81	463. 1.12
371. 0.2	402. 0.51	433. 0.82	464. 1.13
372. 0.21	403. 0.52	434. 0.83	465. 1.14
373. 0.22	404. 0.53	435. 0.84	466. 1.15
374. 0.23	405. 0.54	436. 0.85	467. 1.16
375. 0.24	406. 0.55	437. 0.86	468. 1.17
376. 0.25	407. 0.56	438. 0.87	469. 1.18
377. 0.26	408. 0.57	439. 0.88	470. 1.19
378. 0.27	409. 0.58	440. 0.89	471. 1.2
379. 0.28	410. 0.59	441. 0.9	472. 1.21
380. 0.29	411. 0.6	442. 0.91	473. 1.22
381. 0.3	412. 0.61	443. 0.92	474. 1.23
382. 0.31	413. 0.62	444. 0.93	475. 1.24
383. 0.32	414. 0.63	445. 0.94	476. 1.25
384. 0.33	415. 0.64	446. 0.95	477. 1.26
385. 0.34	416. 0.65	447. 0.96	478. 1.27
386. 0.35	417. 0.66	448. 0.97	479. 1.28
387. 0.36	418. 0.67	449. 0.98	480. 1.29
388. 0.37	419. 0.68	450. 0.99	481. 1.3
389. 0.38	420. 0.69	451. 1	482. 1.31
390. 0.39	421. 0.7	452. 1.01	483. 1.32
391. 0.4	422. 0.71	453. 1.02	484. 1.33
392. 0.41	423. 0.72	454. 1.03	485. 1.34
393. 0.42	424. 0.73	455. 1.04	486. 1.35
394. 0.43	425. 0.74	456. 1.05	487. 1.36
395. 0.44	426. 0.75	457. 1.06	488. 1.37

489. 1.38	520. 1.69	551. 2	582. 2.31
490. 1.39	521. 1.7	552. 2.01	583. 2.32
491. 1.4	522. 1.71	553. 2.02	584. 2.33
492. 1.41	523. 1.72	554. 2.03	585. 2.34
493. 1.42	524. 1.73	555. 2.04	586. 2.35
494. 1.43	525. 1.74	556. 2.05	587. 2.36
495. 1.44	526. 1.75	557. 2.06	588. 2.37
496. 1.45	527. 1.76	558. 2.07	589. 2.38
497. 1.46	528. 1.77	559. 2.08	590. 2.39
498. 1.47	529. 1.78	560. 2.09	591. 2.4
499. 1.48	530. 1.79	561. 2.1	592. 2.41
500. 1.49	531. 1.8	562. 2.11	593. 2.42
501. 1.5	532. 1.81	563. 2.12	594. 2.43
502. 1.51	533. 1.82	564. 2.13	595. 2.44
503. 1.52	534. 1.83	565. 2.14	596. 2.45
504. 1.53	535. 1.84	566. 2.15	597. 2.46
505. 1.54	536. 1.85	567. 2.16	598. 2.47
506. 1.55	537. 1.86	568. 2.17	599. 2.48
507. 1.56	538. 1.87	569. 2.18	600. 2.49
508. 1.57	539. 1.88	570. 2.19	601. 2.5
509. 1.58	540. 1.89	571. 2.2	602. 2.51
510. 1.59	541. 1.9	572. 2.21	603. 2.52
511. 1.6	542. 1.91	573. 2.22	604. 2.53
512. 1.61	543. 1.92	574. 2.23	605. 2.54
513. 1.62	544. 1.93	575. 2.24	606. 2.55
514. 1.63	545. 1.94	576. 2.25	607. 2.56
515. 1.64	546. 1.95	577. 2.26	608. 2.57
516. 1.65	547. 1.96	578. 2.27	609. 2.58
517. 1.66	548. 1.97	579. 2.28	610. 2.59
518. 1.67	549. 1.98	580. 2.29	611. 2.6
519. 1.68	550. 1.99	581. 2.3	612. 2.61

613. 2.62	636. 2.85	659. 3.08	682. 3.31
614. 2.63	637. 2.86	660. 3.09	683. 3.32
615. 2.64	638. 2.87	661. 3.1	684. 3.33
616. 2.65	639. 2.88	662. 3.11	685. 3.34
617. 2.66	640. 2.89	663. 3.12	686. 3.35
618. 2.67	641. 2.9	664. 3.13	687. 3.36
619. 2.68	642. 2.91	665. 3.14	688. 3.37
620. 2.69	643. 2.92	666. 3.15	689. 3.38
621. 2.7	644. 2.93	667. 3.16	690. 3.39
622. 2.71	645. 2.94	668. 3.17	691. 3.4
623. 2.72	646. 2.95	669. 3.18	692. 3.41
624. 2.73	647. 2.96	670. 3.19	693. 3.42
625. 2.74	648. 2.97	671. 3.2	694. 3.43
626. 2.75	649. 2.98	672. 3.21	695. 3.44
627. 2.76	650. 2.99	673. 3.22	696. 3.45
628. 2.77	651. 3	674. 3.23	697. 3.46
629. 2.78	652. 3.01	675. 3.24	698. 3.47
630. 2.79	653. 3.02	676. 3.25	699. 3.48
631. 2.8	654. 3.03	677. 3.26	700. 3.49
632. 2.81	655. 3.04	678. 3.27	701. 3.5
633. 2.82	656. 3.05	679. 3.28	
634. 2.83	657. 3.06	680. 3.29	
635. 2.84	658. 3.07	681. 3.3	

### 3.13. Targets

1. 1	8. 0.539262216	15. 0.162833868	22. -0.138929956
2. 0.928632986	9. 0.480557639	16. 0.115369147	23. -0.176473016
3. 0.859165985	10. 0.423544301	17. 0.069400646	24. -0.212701118
4. 0.791568247	11. 0.368191451	18. 0.024900852	25. -0.247637729
5. 0.725809023	12. 0.314472386	19. -0.018156939	26. -0.281307125
6. 0.661858371	13. 0.262357974	20. -0.059797003	27. -0.313734391
7. 0.599686352	14. 0.211821512	21. -0.100046044	28. -0.344942186

29. -0.374954786	60. -0.828056913	91. -0.706825886	122. -0.382822685
30. -0.403794847	61. -0.83063423	92. -0.698015263	123. -0.371782703
31. -0.431485838	62. -0.832646723	93. -0.689016905	124. -0.360771043
32. -0.458049606	63. -0.834108956	94. -0.679839713	125. -0.34979337
33. -0.483508809	64. -0.835037114	95. -0.670491778	126. -0.33885292
34. -0.507885297	65. -0.835444953	96. -0.660983621	127. -0.327954549
35. -0.531201726	66. -0.83534704	97. -0.651323333	128. -0.317103111
36. -0.553479136	67. -0.834757939	98. -0.641519007	129. -0.306303463
37. -0.574738566	68. -0.833691406	99. -0.631579544	130. -0.295558841
38. -0.595001056	69. -0.8321612	100. -0.621513844	131. -0.284873291
39. -0.614288454	70. -0.830181885	101. -0.611329192	132. -0.274251668
40. -0.632620181	71. -0.827765599	102. -0.601033678	133. -0.263697208
41. -0.650016467	72. -0.824926099	103. -0.590634587	134. -0.253213959
42. -0.666498352	73. -0.821677142	104. -0.580140818	135. -0.242805157
43. -0.682084447	74. -0.818030865	105. -0.569559655	136. -0.232474848
44. -0.696794982	75. -0.814001024	106. -0.558897572	137. -0.222226268
45. -0.710649379	76. -0.809598141	107. -0.548161851	138. -0.212062655
46. -0.723665439	77. -0.80483597	108. -0.537360584	139. -0.201987245
47. -0.735864202	78. -0.799725841	109. -0.526500246	140. -0.192004085
48. -0.747261853	79. -0.794279893	110. -0.515587309	141. -0.182114792
49. -0.757877812	80. -0.788510262	111. -0.504628247	142. -0.172322604
50. -0.767729881	81. -0.782428278	112. -0.493630344	143. -0.162630757
51. -0.776836673	82. -0.77604527	113. -0.482599263	144. -0.153042489
52. -0.78521518	83. -0.769371758	114. -0.471541479	145. -0.143559417
53. -0.792882396	84. -0.762419879	115. -0.460463464	146. -0.134183969
54. -0.799855314	85. -0.755199344	116. -0.449370884	147. -0.124919383
55. -0.806151737	86. -0.747721481	117. -0.438268593	148. -0.115768085
56. -0.811788658	87. -0.739996003	118. -0.427163875	149. -0.106731694
57. -0.816780643	88. -0.732033427	119. -0.416061584	150. -0.097812638
58. -0.821145494	89. -0.723845083	120. -0.404966576	151. -0.089012535
59. -0.824899396	90. -0.715439062	121. -0.393885324	152. -0.080333813

153. -0.071778899	184. 0.125612872	215. 0.188093198	246. 0.143391102
154. -0.063348603	185. 0.12968479	216. 0.188086724	247. 0.140715871
155. -0.055045352	186. 0.133612671	217. 0.187966962	248. 0.137984805
156. -0.046870764	187. 0.137396513	218. 0.187737957	249. 0.135199522
157. -0.038826459	188. 0.141037126	219. 0.187399709	250. 0.13236164
158. -0.030913244	189. 0.144535318	220. 0.186953837	251. 0.129472779
159. -0.02313274	190. 0.147890282	221. 0.186402769	252. 0.126535365
160. -0.015486563	191. 0.151104443	222. 0.185747313	253. 0.123551827
161. -0.007975523	192. 0.154177803	223. 0.184989897	254. 0.120523782
162. -0.000601239	193. 0.157110362	224. 0.18413214	255. 0.11745285
163. 0.006634671	194. 0.159903737	225. 0.183174851	256. 0.114340648
164. 0.013732207	195. 0.162557929	226. 0.182120457	257. 0.111189604
165. 0.020690559	196. 0.165075365	227. 0.180970577	258. 0.108002146
166. 0.02750811	197. 0.167454428	228. 0.179726829	259. 0.104779892
167. 0.03418405	198. 0.169698353	229. 0.178391641	260. 0.101523652
168. 0.04071838	199. 0.171807141	230. 0.176965822	261. 0.098236662
169. 0.04710948	200. 0.1737816	231. 0.175451799	262. 0.09492054
170. 0.05335816	201. 0.175622542	232. 0.173851192	263. 0.091577715
171. 0.059461992	202. 0.177332391	233. 0.172165618	264. 0.088208995
172. 0.065422595	203. 0.178910341	234. 0.170397506	265. 0.084815999
173. 0.071237541	204. 0.180359627	235. 0.168547663	266. 0.081402773
174. 0.076908449	205. 0.18167944	236. 0.166619328	267. 0.077968507
175. 0.0824337	206. 0.182872208	237. 0.164613309	268. 0.074517249
176. 0.087813294	207. 0.18393874	238. 0.162531225	269. 0.071049806
177. 0.09304804	208. 0.184880654	239. 0.160376313	270. 0.067567797
178. 0.09813632	209. 0.18569957	240. 0.15815019	271. 0.064073651
179. 0.103078944	210. 0.186395486	241. 0.155853667	272. 0.060569794
180. 0.107876719	211. 0.186970831	242. 0.15348917	273. 0.057056226
181. 0.112528838	212. 0.187427222	243. 0.151058319	274. 0.053536994
182. 0.1170353	213. 0.18776547	244. 0.148564349	275. 0.050012098
183. 0.121396105	214. 0.187987192	245. 0.146008071	276. 0.046483964

277. 0.042954213	308. -0.058149462	339. -0.116162168	370. -0.108392992
278. 0.03942527	309. -0.06087891	340. -0.116992412	371. -0.107008442
279. 0.035897946	310. -0.063559805	341. -0.117751446	372. -0.105557538
280. 0.032374668	311. -0.066192149	342. -0.11843927	373. -0.10403947
281. 0.028857055	312. -0.068774322	343. -0.119055075	374. -0.102455856
282. 0.025347533	313. -0.071305514	344. -0.11959967	375. -0.100807506
283. 0.021846104	314. -0.073784109	345. -0.120071437	376. -0.09909361
284. 0.018355194	315. -0.076209296	346. -0.120471184	377. -0.097316596
285. 0.014876422	316. -0.078580266	347. -0.120798103	378. -0.095476464
286. 0.011412216	317. -0.080895401	348. -0.121053002	379. -0.093574023
287. 0.007963385	318. -0.083154701	349. -0.121235073	380. -0.091610083
288. 0.004532356	319. -0.085356548	350. -0.121344316	381. -0.089585452
289. 0.00111913	320. -0.087500131	351. -0.12138073	382. -0.087500131
290. -0.002273056	321. -0.089585452	352. -0.121344316	383. -0.085356548
291. -0.005642586	322. -0.091610083	353. -0.121235073	384. -0.083154701
292. -0.008989457	323. -0.093574023	354. -0.121053002	385. -0.080895401
293. -0.012311243	324. -0.095476464	355. -0.120798103	386. -0.078580266
294. -0.015606325	325. -0.097316596	356. -0.120471184	387. -0.076209296
295. -0.018873085	326. -0.09909361	357. -0.120071437	388. -0.073784109
296. -0.022110714	327. -0.100807506	358. -0.11959967	389. -0.071305514
297. -0.025317593	328. -0.102455856	359. -0.119055075	390. -0.068774322
298. -0.028491294	329. -0.10403947	360. -0.11843927	391. -0.066192149
299. -0.031631818	330. -0.105557538	361. -0.117751446	392. -0.063559805
300. -0.034736737	331. -0.107008442	362. -0.116992412	393. -0.06087891
301. -0.037805242	332. -0.108392992	363. -0.116162168	394. -0.058149462
302. -0.040835714	333. -0.109709568	364. -0.115261523	395. -0.055373889
303. -0.043826536	334. -0.110958171	365. -0.114290477	396. -0.052552192
304. -0.046777706	335. -0.112137992	366. -0.11324903	397. -0.049685988
305. -0.049685988	336. -0.11324903	367. -0.112137992	398. -0.046777706
306. -0.052552192	337. -0.114290477	368. -0.110958171	399. -0.043826536
307. -0.055373889	338. -0.115261523	369. -0.109709568	400. -0.040835714

401. -0.037805242	432. 0.067567797	463. 0.160376313	494. 0.184880654
402. -0.034736737	433. 0.071049806	464. 0.162531225	495. 0.18393874
403. -0.031631818	434. 0.074517249	465. 0.164613309	496. 0.182872208
404. -0.028491294	435. 0.077968507	466. 0.166619328	497. 0.18167944
405. -0.025317593	436. 0.081402773	467. 0.168547663	498. 0.180359627
406. -0.022110714	437. 0.084815999	468. 0.170397506	499. 0.178910341
407. -0.018873085	438. 0.088208995	469. 0.172165618	500. 0.177332391
408. -0.015606325	439. 0.091577715	470. 0.173851192	501. 0.175622542
409. -0.012311243	440. 0.09492054	471. 0.175451799	502. 0.1737816
410. -0.008989457	441. 0.098236662	472. 0.176965822	503. 0.171807141
411. -0.005642586	442. 0.101523652	473. 0.178391641	504. 0.169698353
412. -0.002273056	443. 0.104779892	474. 0.179726829	505. 0.167454428
413. 0.00111913	444. 0.108002146	475. 0.180970577	506. 0.165075365
414. 0.004532356	445. 0.111189604	476. 0.182120457	507. 0.162557929
415. 0.007963385	446. 0.114340648	477. 0.183174851	508. 0.159903737
416. 0.011412216	447. 0.11745285	478. 0.18413214	509. 0.157110362
417. 0.014876422	448. 0.120523782	479. 0.184989897	510. 0.154177803
418. 0.018355194	449. 0.123551827	480. 0.185747313	511. 0.151104443
419. 0.021846104	450. 0.126535365	481. 0.186402769	512. 0.147890282
420. 0.025347533	451. 0.129472779	482. 0.186953837	513. 0.144535318
421. 0.028857055	452. 0.13236164	483. 0.187399709	514. 0.141037126
422. 0.032374668	453. 0.135199522	484. 0.187737957	515. 0.137396513
423. 0.035897946	454. 0.137984805	485. 0.187966962	516. 0.133612671
424. 0.03942527	455. 0.140715871	486. 0.188086724	517. 0.12968479
425. 0.042954213	456. 0.143391102	487. 0.188093198	518. 0.125612872
426. 0.046483964	457. 0.146008071	488. 0.187987192	519. 0.121396105
427. 0.050012098	458. 0.148564349	489. 0.18776547	520. 0.1170353
428. 0.053536994	459. 0.151058319	490. 0.187427222	521. 0.112528838
429. 0.057056226	460. 0.15348917	491. 0.186970831	522. 0.107876719
430. 0.060569794	461. 0.155853667	492. 0.186395486	523. 0.103078944
431. 0.064073651	462. 0.15815019	493. 0.18569957	524. 0.09813632

525. 0.09304804	556. -0.134183969	587. -0.460463464	618. -0.762419879
526. 0.087813294	557. -0.143559417	588. -0.471541479	619. -0.769371758
527. 0.0824337	558. -0.153042489	589. -0.482599263	620. -0.77604527
528. 0.076908449	559. -0.162630757	590. -0.493630344	621. -0.782428278
529. 0.071237541	560. -0.172322604	591. -0.504628247	622. -0.788510262
530. 0.065422595	561. -0.182114792	592. -0.515587309	623. -0.794279893
531. 0.059461992	562. -0.192004085	593. -0.526500246	624. -0.799725841
532. 0.05335816	563. -0.201987245	594. -0.537360584	625. -0.80483597
533. 0.04710948	564. -0.212062655	595. -0.548161851	626. -0.809598141
534. 0.04071838	565. -0.222226268	596. -0.558897572	627. -0.814001024
535. 0.03418405	566. -0.232474848	597. -0.569559655	628. -0.818030865
536. 0.02750811	567. -0.242805157	598. -0.580140818	629. -0.821677142
537. 0.020690559	568. -0.253213959	599. -0.590634587	630. -0.824926099
538. 0.013732207	569. -0.263697208	600. -0.601033678	631. -0.827765599
539. 0.006634671	570. -0.274251668	601. -0.611329192	632. -0.830181885
540. -0.000601239	571. -0.284873291	602. -0.621513844	633. -0.8321612
541. -0.007975523	572. -0.295558841	603. -0.631579544	634. -0.833691406
542. -0.015486563	573. -0.306303463	604. -0.641519007	635. -0.834757939
543. -0.02313274	574. -0.317103111	605. -0.651323333	636. -0.83534704
544. -0.030913244	575. -0.327954549	606. -0.660983621	637. -0.835444953
545. -0.038826459	576. -0.33885292	607. -0.670491778	638. -0.835037114
546. -0.046870764	577. -0.34979337	608. -0.679839713	639. -0.834108956
547. -0.055045352	578. -0.360771043	609. -0.689016905	640. -0.832646723
548. -0.063348603	579. -0.371782703	610. -0.698015263	641. -0.83063423
549. -0.071778899	580. -0.382822685	611. -0.706825886	642. -0.828056913
550. -0.080333813	581. -0.393885324	612. -0.715439062	643. -0.824899396
551. -0.089012535	582. -0.404966576	613. -0.723845083	644. -0.821145494
552. -0.097812638	583. -0.416061584	614. -0.732033427	645. -0.816780643
553. -0.106731694	584. -0.427163875	615. -0.739996003	646. -0.811788658
554. -0.115768085	585. -0.438268593	616. -0.747721481	647. -0.806151737
555. -0.124919383	586. -0.449370884	617. -0.755199344	648. -0.799855314



649. -0.792882396	663. -0.614288454	677. -0.247637729	691. 0.368191451
650. -0.78521518	664. -0.595001056	678. -0.212701118	692. 0.423544301
651. -0.776836673	665. -0.574738566	679. -0.176473016	693. 0.480557639
652. -0.767729881	666. -0.553479136	680. -0.138929956	694. 0.539262216
653. -0.757877812	667. -0.531201726	681. -0.100046044	695. 0.599686352
654. -0.747261853	668. -0.507885297	682. -0.059797003	696. 0.661858371
655. -0.735864202	669. -0.483508809	683. -0.018156939	697. 0.725809023
656. -0.723665439	670. -0.458049606	684. 0.024900852	698. 0.791568247
657. -0.710649379	671. -0.431485838	685. 0.069400646	699. 0.859165985
658. -0.696794982	672. -0.403794847	686. 0.115369147	700. 0.928632986
659. -0.682084447	673. -0.374954786	687. 0.162833868	701. 1
660. -0.666498352	674. -0.344942186	688. 0.211821512	
661. -0.650016467	675. -0.313734391	689. 0.262357974	
662. -0.632620181	676. -0.281307125	690. 0.314472386	

### 3.13.1. Datos

$$V1 = [11 \ 16 \ 10 \ 1]$$

$$V2 = [3 \ 2 \ 1]$$

epochmax = 10000

alpha=.01

Múltiplo para las épocas de validación = 500

numval = 7

error de validación = .0000000000000001

Configuración: 80-10-10

### 3.14. Resultado

### 3.15. Imágenes

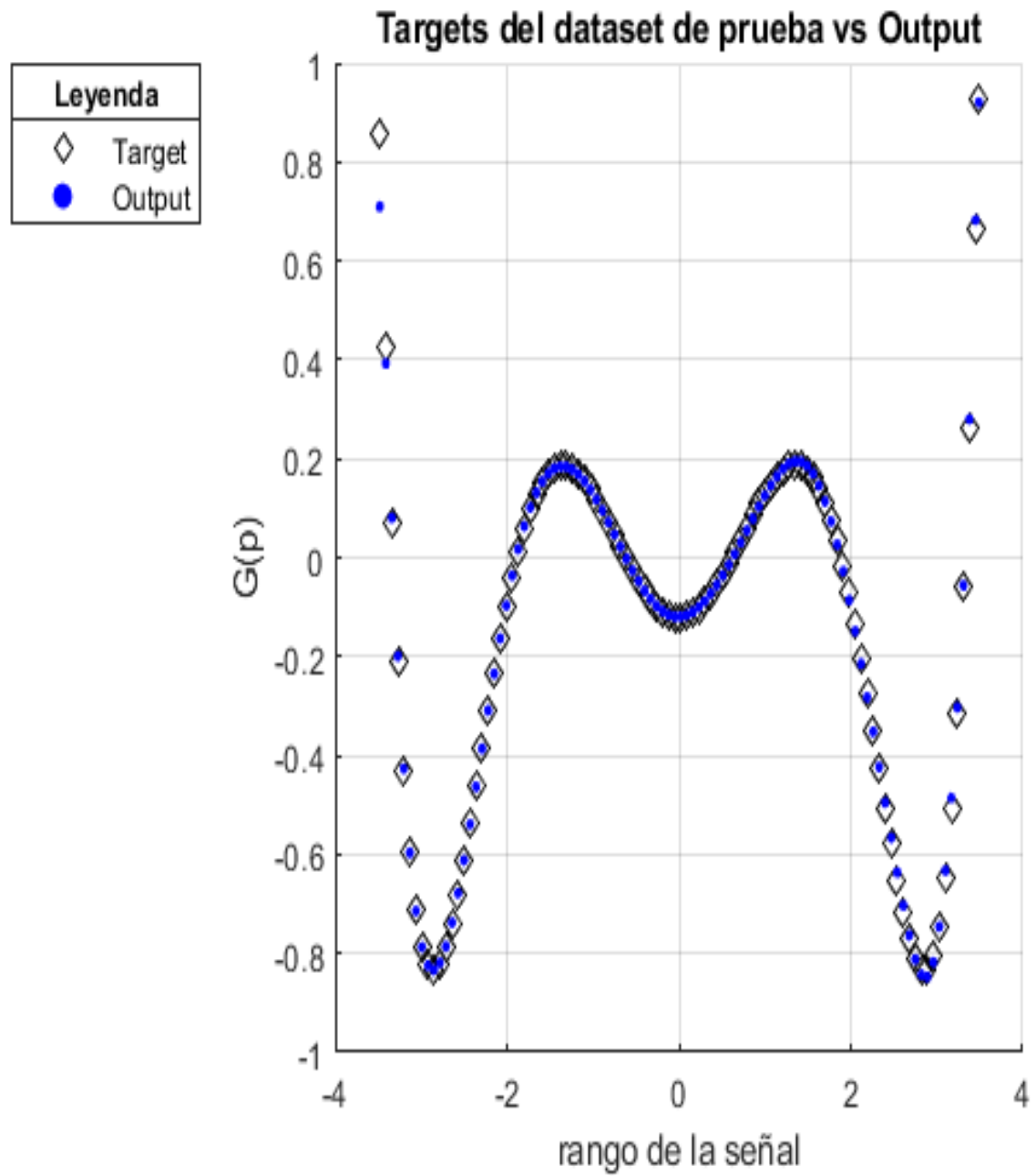


Figura 22: Gráfica 3.1

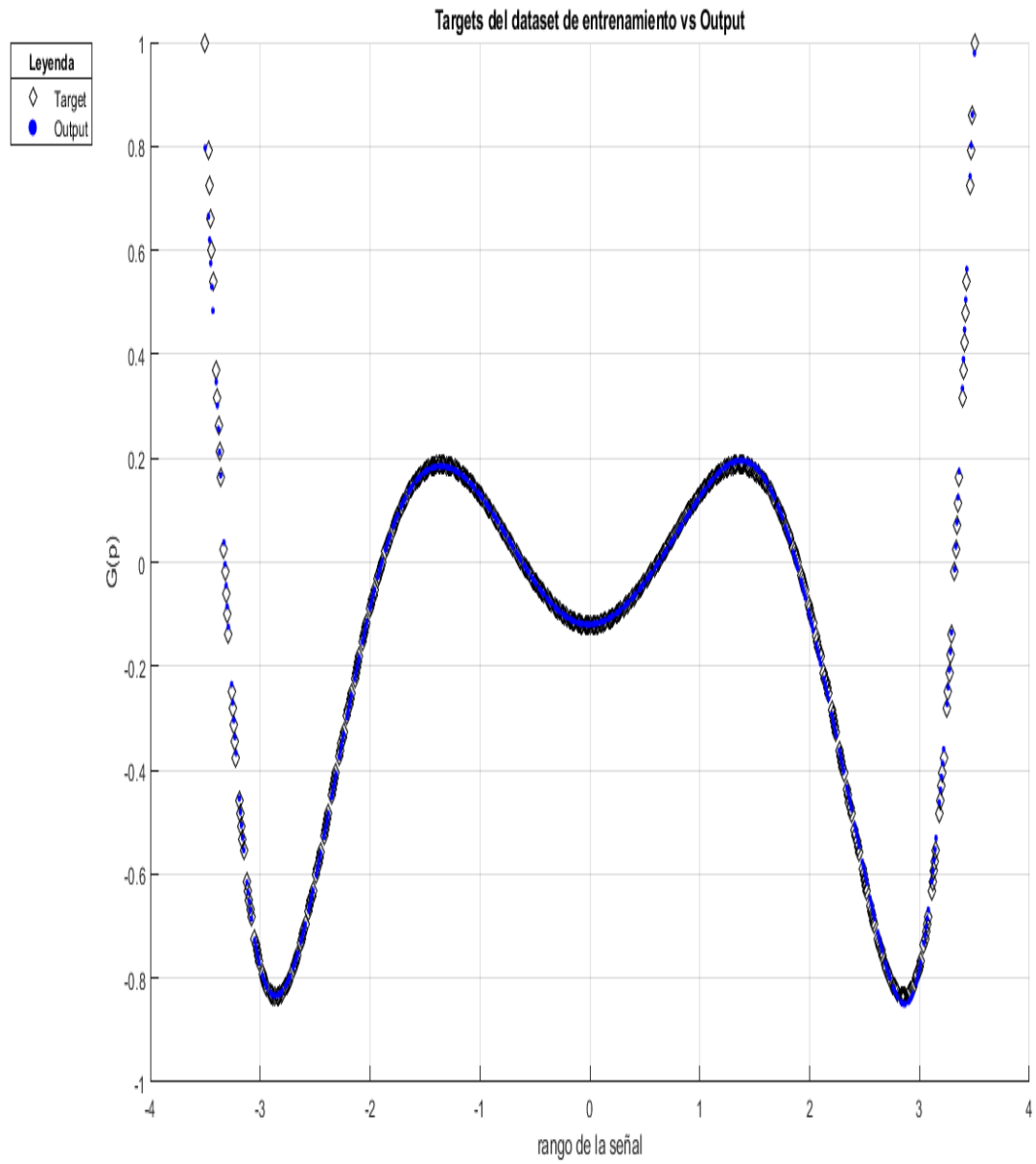


Figura 23: Gráfica 3.2

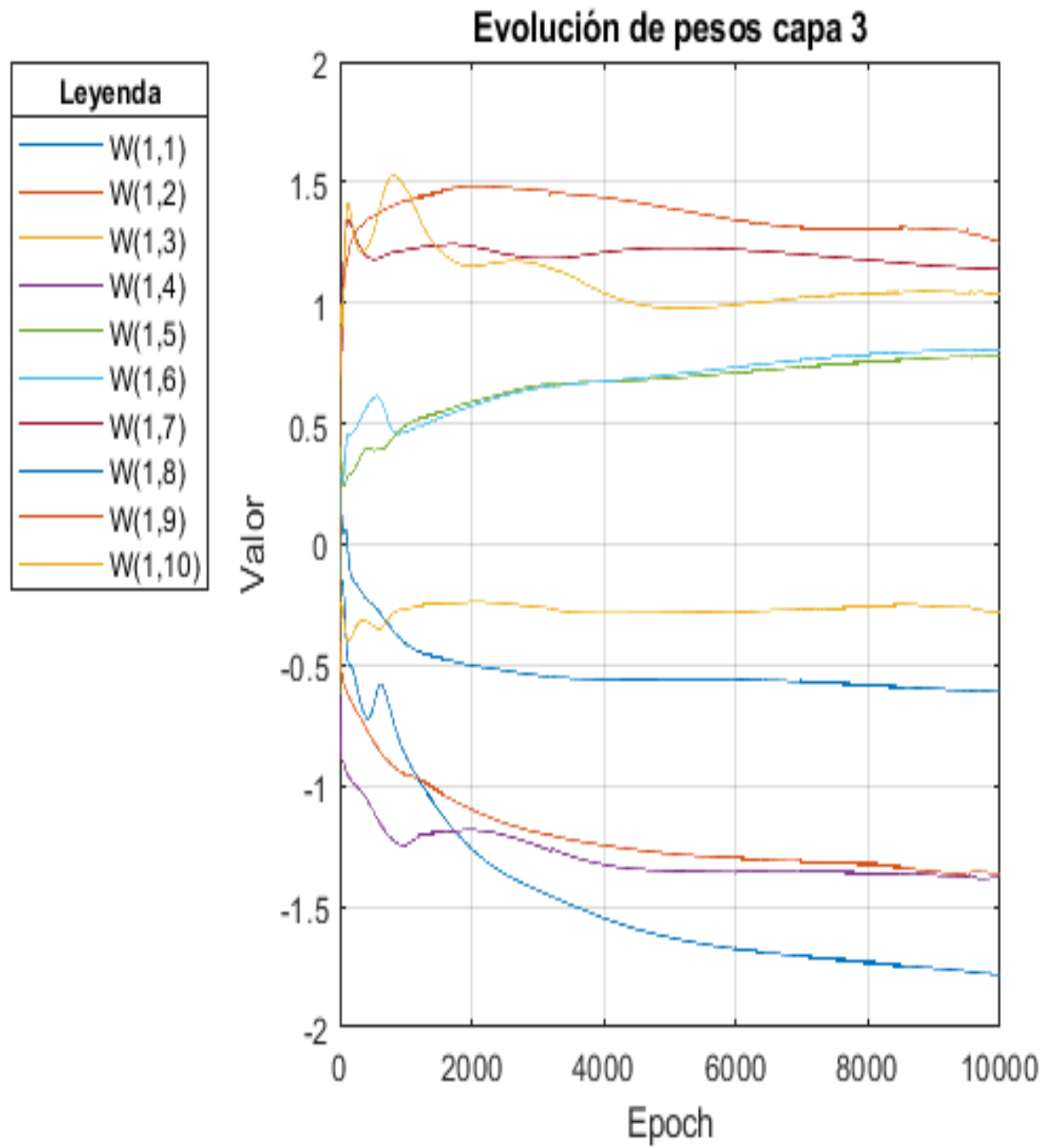


Figura 24: Gráfica 3.3

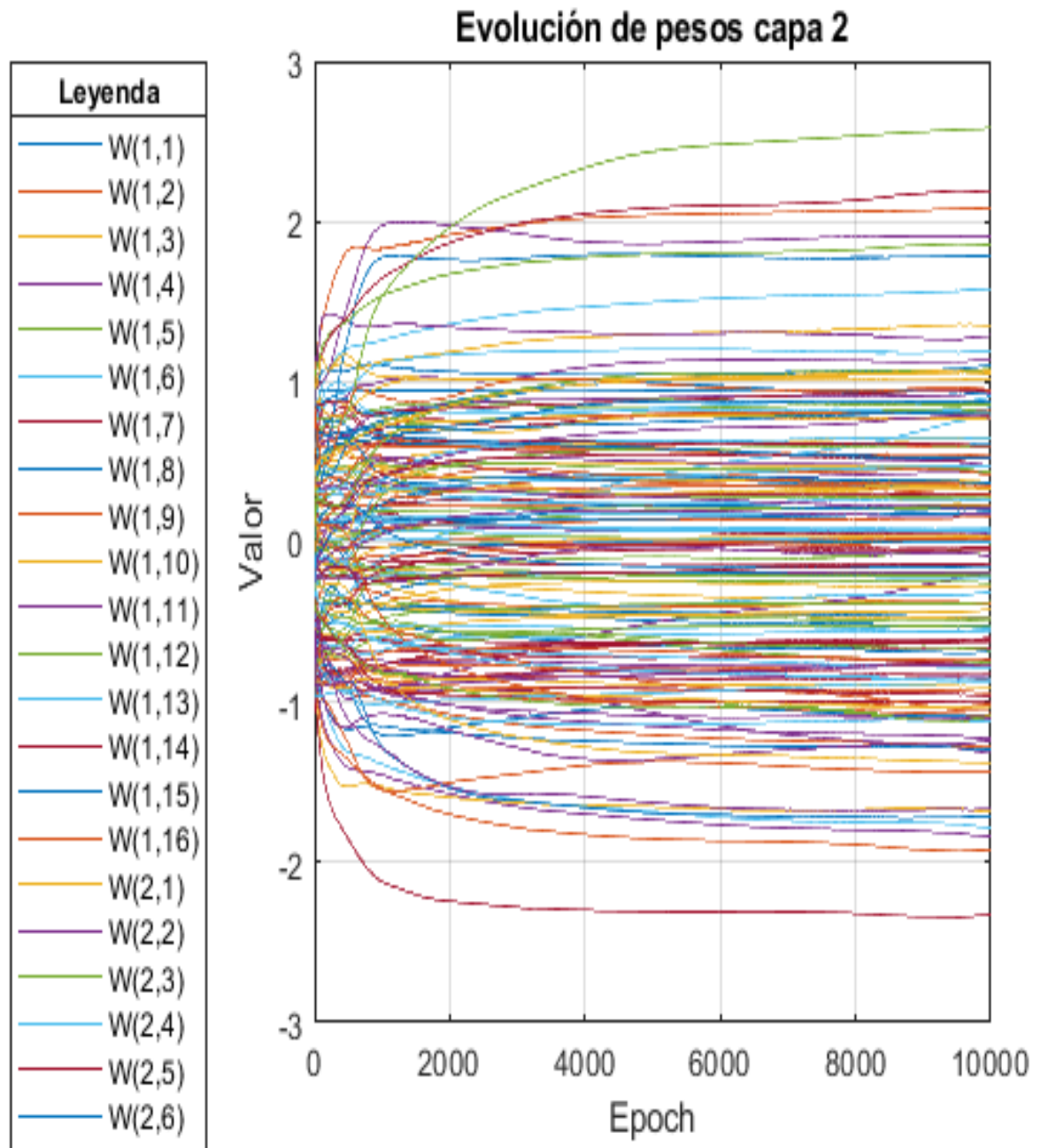


Figura 25: Gráfica 3.4

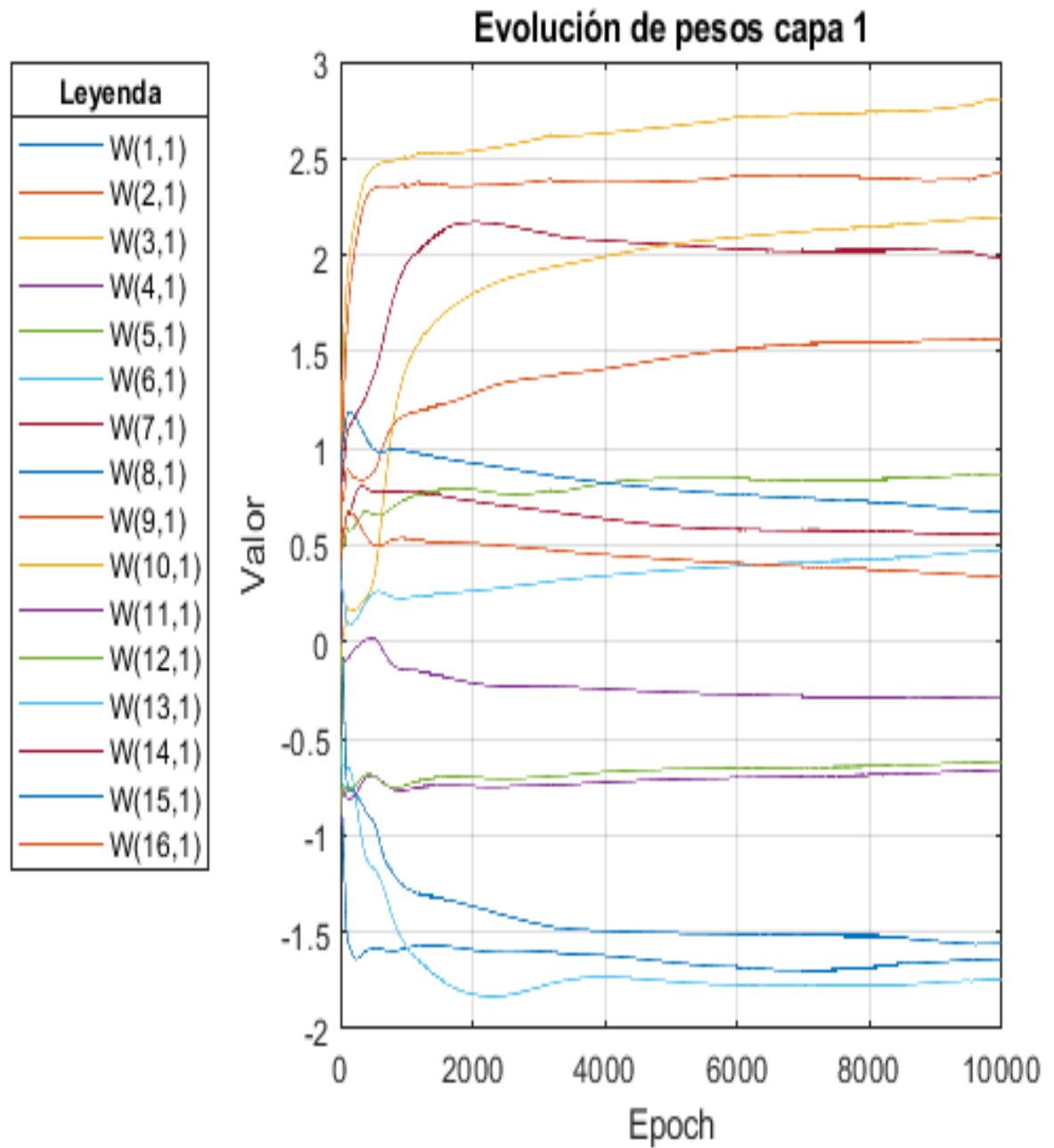


Figura 26: Gráfica 3.5

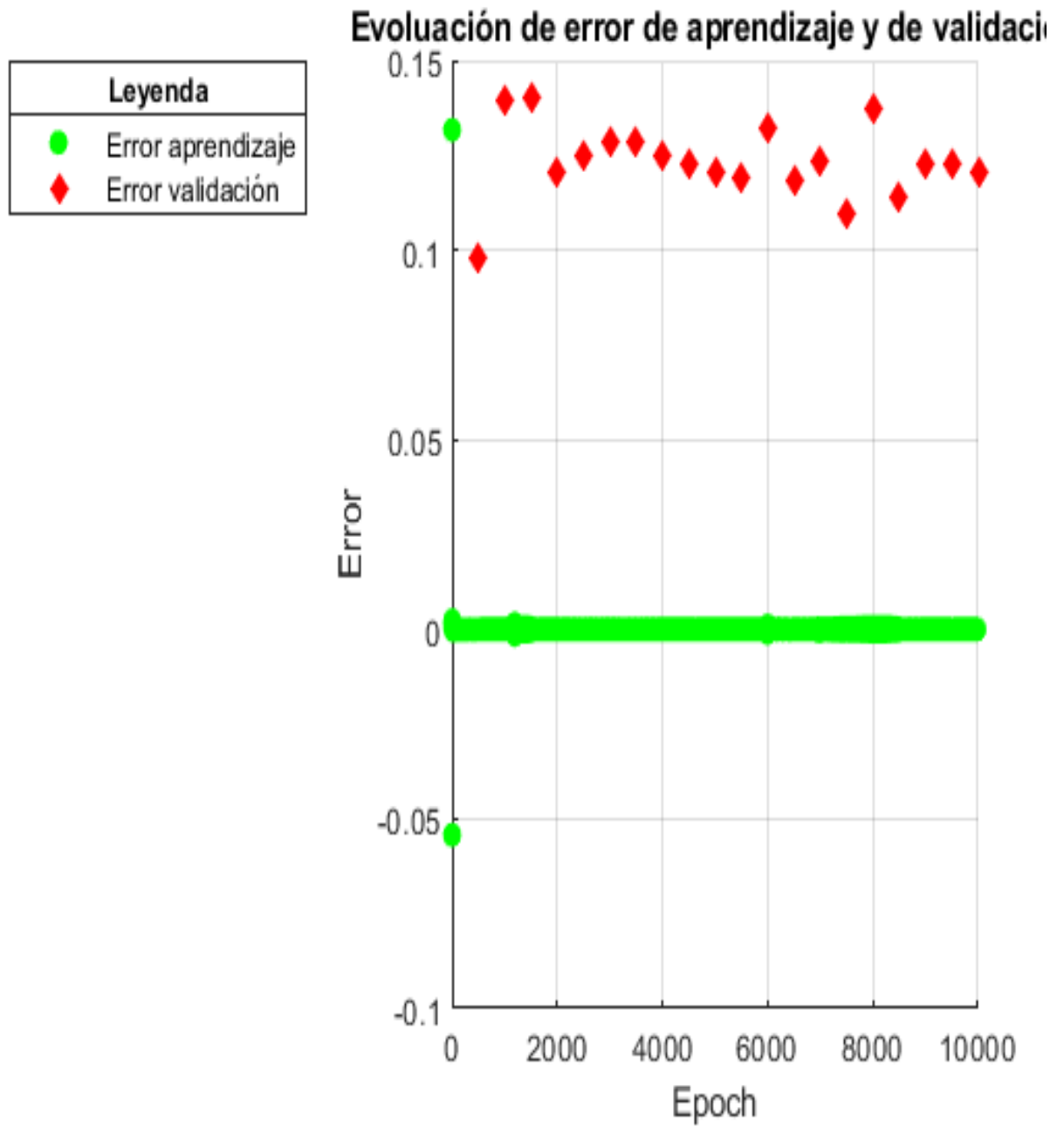


Figura 27: Gráfica 3.6

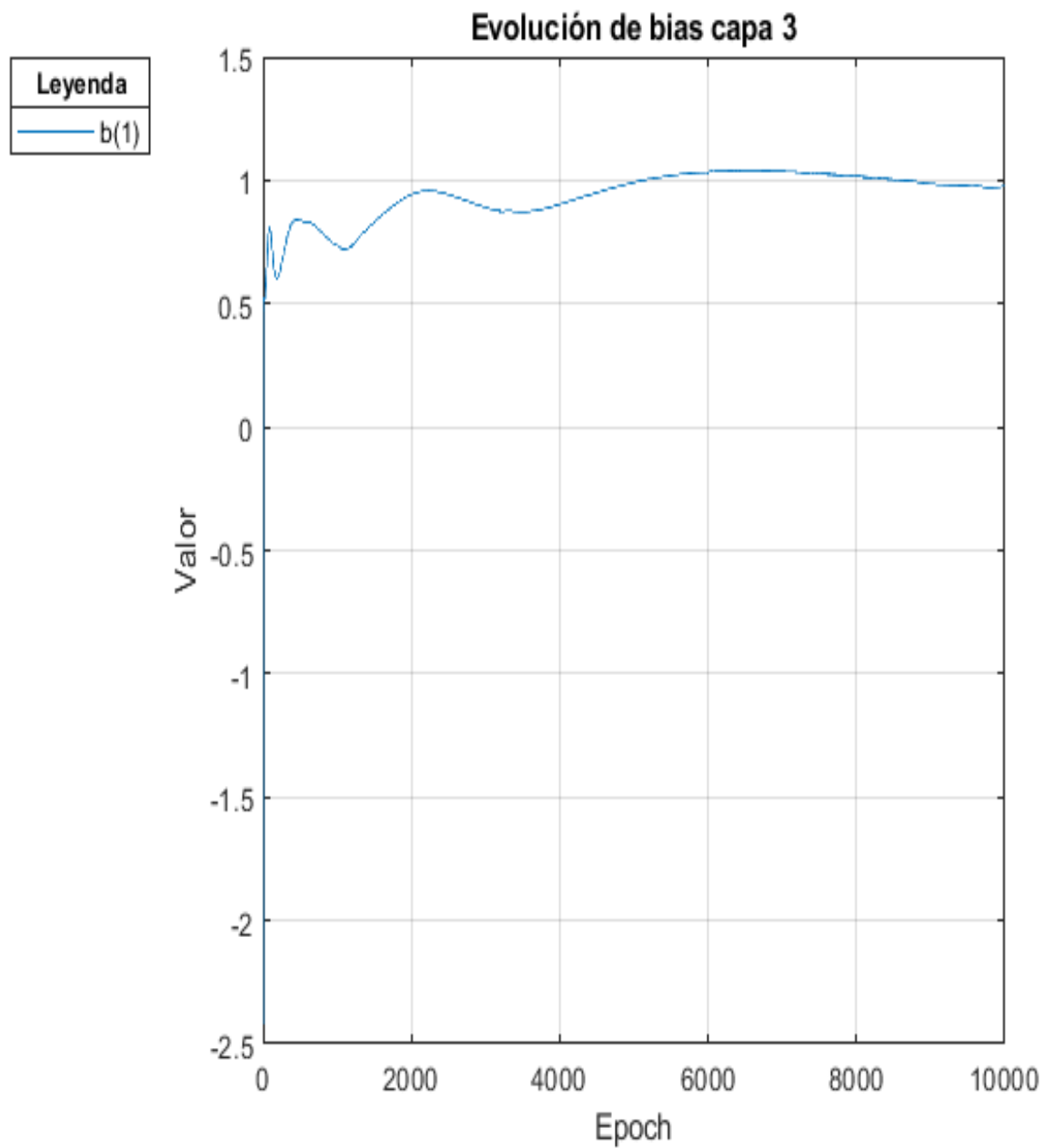


Figura 28: Gráfica 3.7



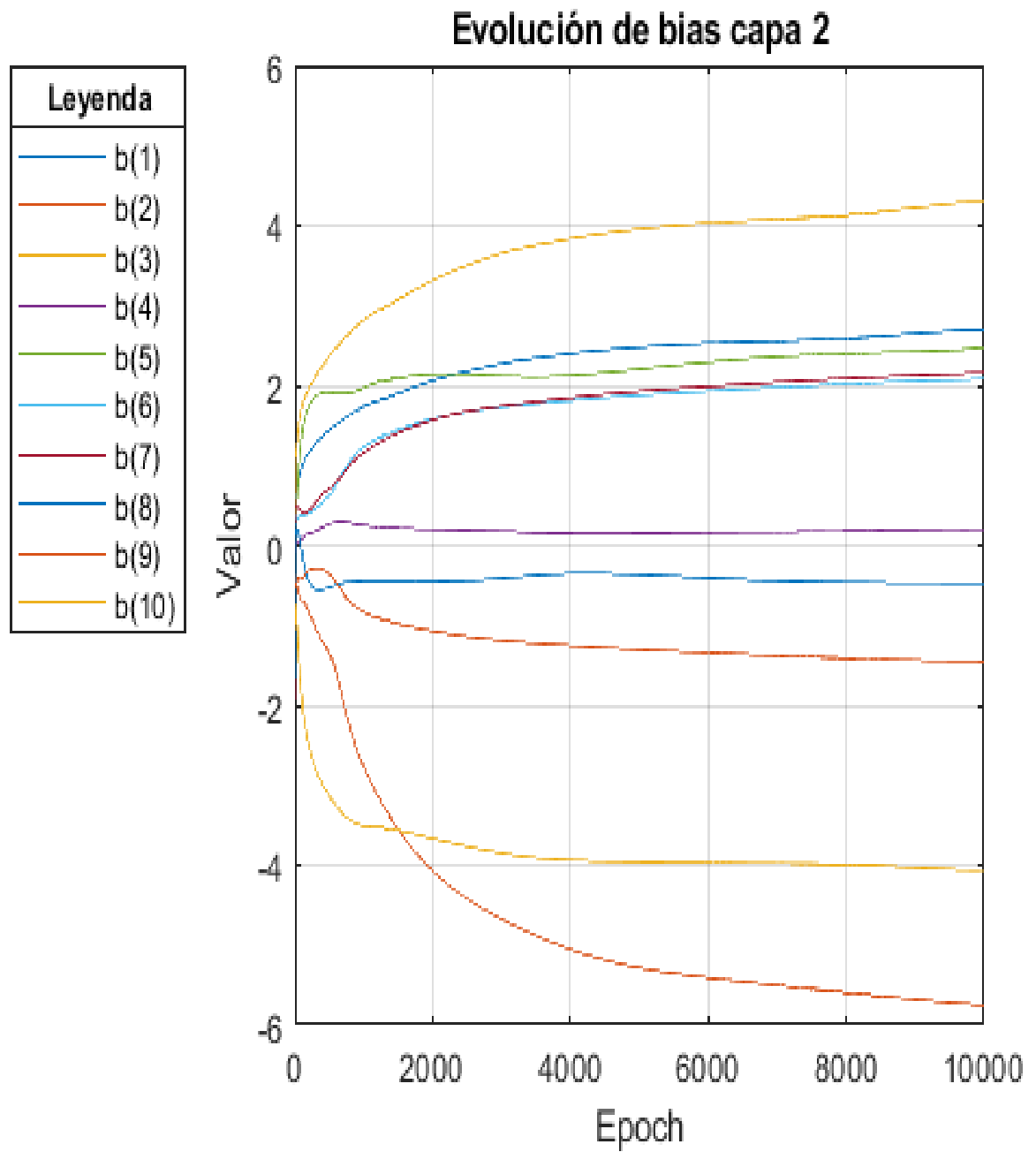


Figura 29: Gráfica 3.8

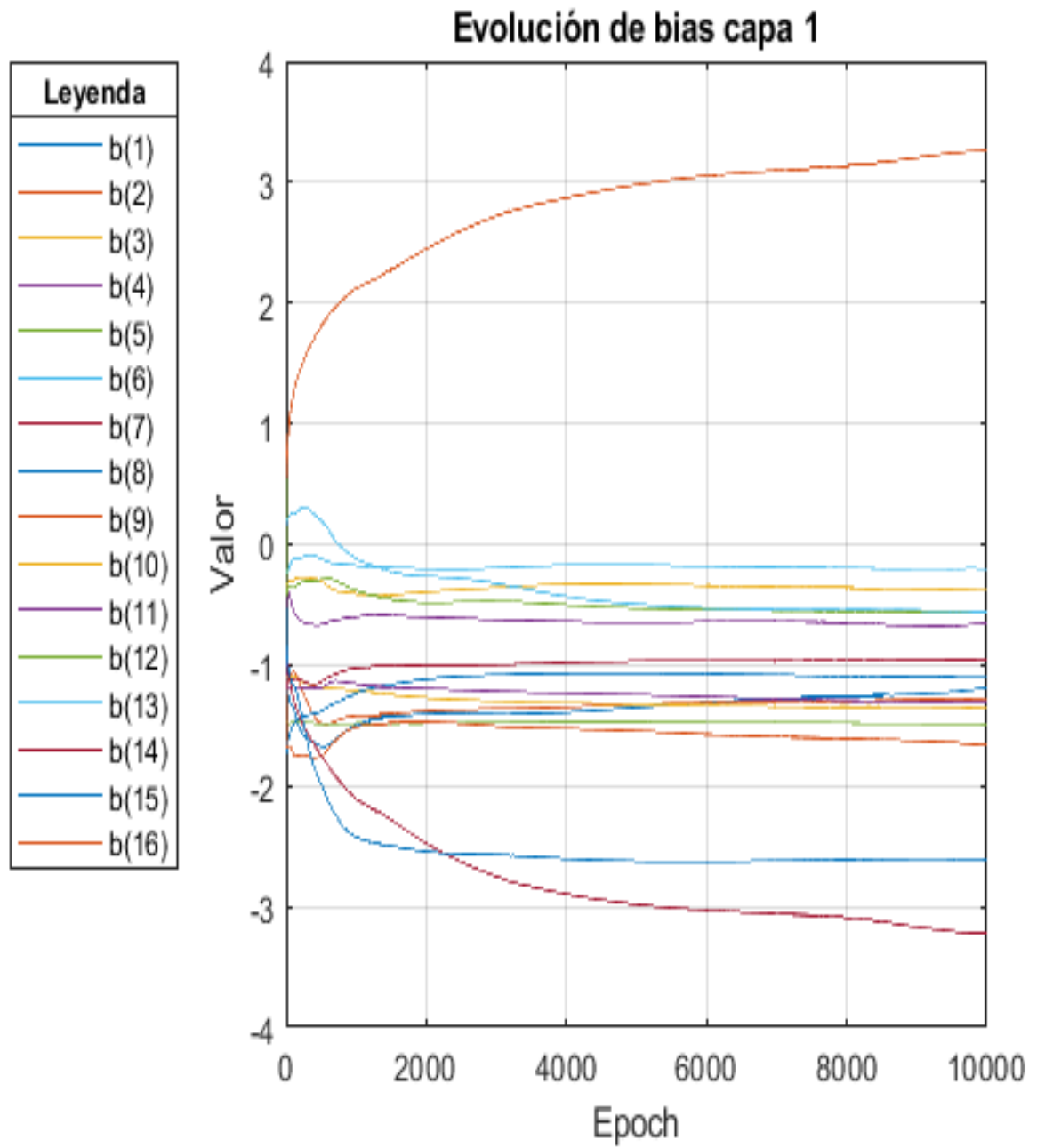


Figura 30: Gráfica 3.9

## 4. Discusión de Resultados

Para cada uno de los resultados se muestra:

1. Los datos con los cuales fue realizado el ejemplo.
2. Los pesos y bias iniciales.
3. La gráfica del “historial” de la evolución de los parámetros del perceptrón
4. La gráfica de los vectores de entrada con su target y la frontera de desición final.

## 5. Conclusiones

El arte de diseñar y entrenar redes neuronales radica en las pruebas con distintas arquitecturas/configuraciones para resolver un problema en específico, realizar la implementación del programa fue bastante complicado, ya que todos en nuestro equipo teníamos diferentes ideas a implementar, pero al final con un sistema de votos se solucionó, aprendimos a usar el tipo de dato cell, para preallocar espacio. El MLP es la base de las redes neuronales artificiales modernas, este sigue siendo usado para resolver muchos problemas, pudimos observar como se comporta con diferentes arquitecturas, funciones de transferencia y configuraciones, una práctica muy interesante y muy difícil de probar.

## 6. Referencias

- [1] Martin T Hagan. Machine Learning, Neural Network Design (2nd Edition), 2014.
- [2] “Perceptrón Multicapa”, notas de clase de Neural Networks, Department of Engineering in Computer Systems, Escuela Superior de Cómputo, 2017.
- [3] Math Works, “MATLAB”. Disponible en: <https://es.mathworks.com/products/matlab>.

## 7. Apéndice

Listing 1: mlp.m

```
1 clc
2 clear
3
4 % Read the inputs file
5 inputs_path = strcat(input('Ingrese el nombre del archivo de inputs sin la extensión:
    ','s'), '.txt');
6 %inputs_path = 'inputs.txt';
7 inputs = importdata(inputs_path);
8
9 % Read the targets
10 targets_path = strcat(input('Ingrese el nombre del archivo de targets sin la extensión
    ':' ','s'), '.txt');
11 %targets_path = 'targets.txt';
12 targets = importdata(targets_path);
13
14 data_size = size(inputs, 1);
```

```
15
16 % Enter MLP architecture
17 architecture = str2num(input('Ingrese el vector de la arquitectura: ','s'));
18 % Calculate layer parameters
19 %architecture = str2num('1 16 10 1');
20 num_layers = length/architecture) - 1;
21 R = architecture(1);
22 functions_vector = str2num(input('Ingrese el vector de las funciones de activación: 1)
    purelin()\n2) logsig()\n3) tansig()\n\n: ','s'));
23 %functions_vector = str2num('3 2 1');
24
25 % Enter the learning factor
26 alpha = input('Ingresa el valor del factor de aprendizaje(alpha): ');
27 %alpha = .01;
28
29 epochmax = input('Ingresa el número máximo de épocas: ');
30 % epochmax = 10000;
31 %validation_iter = 500;
32 %numval = 7;
33 %error_epoch_validation = .00000000000000001;
34 numval = input('Numero maximo de incrementos consecutivos del error de validacion (
    numval): ');
35 error_epoch_validation = input('Ingrese el valor minimo del error de epoca (
    error_epoch_validation): ');
36 validation_iter = input('Ingrese el múltiplo de épocas para realizar una época de
    validación (validation_iter): ');
37
38 % Dataset Slicing
39 config_option = input('Elija una configuración de distribución de datasets: \n1:
    80-10-10\n2: 70-15-15\n');
40 %config_option = 2;
41 [training_ds, test_ds, validation_ds] = dataset_slices(config_option, inputs, targets)
    ;
42 validation_ds_size = size(validation_ds, 1);
43 test_ds_size = size(test_ds, 1);
44 training_ds_size = size(training_ds, 1);
45
46 disp('Dataset de entrenamiento:');
47 disp(training_ds);
48 disp('Dataset de validacion:');
49 disp(validation_ds);
50 disp('Dataset de prueba:');
51 disp(test_ds);
52
53 % Open the files for weights and bias
54 total_weight_files = 0;
55 total_bias_files = 0;
56 for i=1:num_layers
57 % For neurons
```

```
58 for j=1:architecture(i + 1)
59 % For weights
60 for l=1:architecture(i)
61 total_weight_files = total_weight_files + 1;
62 end
63 end
64 total_bias_files = total_bias_files + 1;
65 end
66
67 W_files = zeros(total_weight_files, 1);
68 b_files = zeros(total_bias_files, 1);
69
70 current_file = 1;
71 for i=1:num_layers
72 path = strcat(pwd, '/historico/capa_', num2str(i), '/pesos/');
73 if ~exist(path, 'dir')
74 mkdir(path);
75 end
76 % For layers
77 for j=1:architecture(i + 1)
78 % For neurons
79 for k=1:architecture(i)
80 archivo_pesos = strcat(path, '/pesos', num2str(j), '_', num2str(k), '.txt');
81 W_files(current_file) = fopen(archivo_pesos, 'w');
82 current_file = current_file + 1;
83 end
84 end
85 end
86
87 current_file = 1;
88 for i=1:num_layers
89 path = strcat(pwd, '/historico/capa_', num2str(i), '/bias/');
90 if ~exist(path, 'dir')
91 mkdir(path);
92 end
93 for j=1:architecture(i+1)
94 archivo_bias = strcat(path, '/bias', num2str(j), '.txt');
95 b_files(current_file) = fopen(archivo_bias, 'w');
96 current_file = current_file + 1;
97 end
98 end
99
100 % Initialize MLP parameters and Print them
101
102 num_w_files = 1;
103 num_b_files = 1;
104 W = cell(num_layers, 1);
105 b = cell(num_layers, 1);
106 % Output of each layer
```

```
107 a = cell(num_layers + 1, 1);
108 % Sentitivities
109 S = cell(num_layers, 1);
110 % Derivatives of each layer
111 F_m = cell(num_layers, 1);
112
113 % For each layer
114 for i=1:num_layers
115 % Random value
116 W_r_value = 2 * rand(architecture(i + 1), architecture(i)) - 1;
117 b_r_value = 2* rand(architecture(i + 1), 1) - i;
118 W{i} = W_r_value
119 b{i} = b_r_value
120 % For each neuron
121 for j=1:architecture(i + 1)
122 %For each weight
123 for k=1:architecture(i)
124 % Print wights value
125 fprintf(W_files(num_w_files), '%f\r\n', W_r_value(j, k));
126 num_w_files = num_w_files + 1;
127 end
128 end
129 % For each neuron
130 for j=1:architecture(i + 1)
131 % print bias value
132 fprintf(b_files(num_b_files), '%f\r\n', b_r_value(j));
133 num_b_files = num_b_files + 1;
134 end
135 end
136
137
138
139
140 % Learning algorithm
141 num_validation_epoch = 0;
142 early_stopping_increment = 0;
143 validation_error = 0;
144 learning_error = 0;
145 early_s_counter = 0;
146
147 % initialize vectors for printing errors
148 learning_err_values = zeros(epochmax, 1);
149 evaluation_err_values = zeros(ceil(epochmax / validation_iter), 1);
150 for epoch=1:epochmax
151 l_error = 0;
152 % Reset the values
153 num_w_files = 1;
154 num_b_files = 1;
155 % if isn't a validation epoch
```

```
156 if(mod(epoch ,validation_iter) ~= 0)
157 for t_data=1:training_ds_size
158 % initial condition
159 a{1} = training_ds(t_data, 1);
160 % Foward propagation
161 for t_p=1:num_layers
162 W_aux = cell2mat(W(t_p));
163 b_aux = cell2mat(b(t_p));
164 a_aux = cell2mat(a(t_p));
165 n_f = W_aux * a_aux + b_aux;
166 a{t_p + 1} = get_activation_function(n_f, functions_vector(t_p));
167 end
168 a_aux = cell2mat(a(num_layers + 1));
169 t_error = training_ds(t_data, 2) - a_aux;
170 l_error = l_error + (t_error / data_size);
171 % Sensitivities calculation
172 F_m{num_layers} = get_F_matrix(functions_vector(num_layers), architecture(num_layers +
    1), a_aux);
173 F_m_temp = cell2mat(F_m(num_layers));
174 S{num_layers} = F_m_temp * (t_error)*(-2);
175 % Backpropagation
176 for m = num_layers-1:-1:1
177 W_aux = cell2mat(W(m+1));
178 s_aux = cell2mat(S(m+1));
179 a_aux = cell2mat(a(m+1));
180 F_m{m} = get_F_matrix(functions_vector(m),architecture(m+1),a_aux);
181 F_m_temp = cell2mat(F_m(m));
182 S{m} = F_m_temp * (W_aux')*s_aux;
183 end
184 % Learning Rules
185 for k = num_layers:-1:1
186 W_aux = cell2mat(W(k));
187 b_aux = cell2mat(b(k));
188 s_aux = cell2mat(S(k));
189 a_aux = cell2mat(a(k));
190 W{k} = W_aux - (alpha * s_aux * a_aux');
191 b{k} = b_aux - (alpha * s_aux);
192 W_aux = cell2mat(W(k));
193 b_aux = cell2mat(b(k));
194 end
195 end
196 learning_error = l_error;
197 learning_err_values(epoch) = l_error;
198 % This epoch is a validation one
199 else
200 val_error = 0;
201 num_validation_epoch = num_validation_epoch + 1;
202 for t_data = 1:validation_ds_size
203 % Initial Condition
```

```
204 a{1} = validation_ds(t_data, 1);
205 % Foward propagation
206 for k=1:num_layers
207     W_aux = cell2mat(W(k));
208     a_aux = cell2mat(a(k));
209     b_aux = cell2mat(b(k));
210     n_f = W_aux * a_aux + b_aux;
211     a{k + 1} = get_activation_function(n_f, functions_vector(k));
212 end
213 a_aux = cell2mat(a(num_layers+1));
214 val_error = validation_ds(t_data,2)-a_aux;
215 val_error = val_error+(val_error/validation_ds_size);
216 end
217 evaluation_err_values(epoch) = val_error;
218 if early_stopping_increment == 0
219     validation_error = val_error;
220     early_stopping_increment = early_stopping_increment+1;
221     fprintf('Incremento actual para early stopping = %d\n', early_stopping_increment);
222 else
223     if val_error > validation_error
224         validation_error = val_error;
225         early_stopping_increment = early_stopping_increment+1;
226         fprintf('Incremento actual para early stopping = %d\n', early_stopping_increment);
227         if early_stopping_increment == numval
228             % Reset the counter
229             early_s_counter = 1;
230             fprintf('Early stopping en la época:  %d\n', epoch);
231             break;
232         end
233     else
234         validation_error = 0;
235         early_stopping_increment = 0;
236         fprintf('Incremento actual para early stopping = %d\n', early_stopping_increment);
237     end
238 end
239 end
240
241 % Print the values on console
242 num_w_files = 1;
243 num_b_files = 1;
244 for k = num_layers:-1:1
245     W_aux = cell2mat(W(k));
246     b_aux = cell2mat(b(k));
247     for j=1:architecture(k+1)
248         for l=1:architecture(k)
249             fprintf(W_files(num_w_files), '%f\r\n', W_aux(j,l));
250             num_w_files = num_w_files +1;
251         end
252     end
```



```
253 for j=1:architecture(k + 1)
254 fprintf(b_files(num_b_files), '%f\r\n', b_aux(j));
255 num_b_files = num_b_files + 1;
256 end
257 end
258
259 % Check stopping calculations
260 if mod(epoch,validation_iter) ~= 0 && l_error <= error_epoch_validation && l_error >=
    0
261 learning_error = l_error;
262 fprintf('Aprendizaje exitoso en la época %d\n', epoch);
263 break;
264 end
265 end
266
267 if epoch == epochmax
268 disp('Se llego a epochmax');
269 end
270
271 % Print the las final values
272 if early_s_counter == 1
273 num_w_files = 1;
274 num_b_files = 1;
275 for k = num_layers:-1:1
276 W_aux = cell2mat(W(k));
277 b_aux = cell2mat(b(k));
278 for j = 1:architecture(k + 1)
279 for l=1:architecture(k)
280 fprintf(W_files(num_w_files), '%f\r\n', W_aux(j, l));
281 num_w_files = num_w_files + 1;
282 end
283 end
284 for j=1:architecture(k + 1)
285 fprintf(b_files(num_b_files), '%f\r\n', b_aux(j));
286 num_b_files = num_b_files + 1;
287 end
288 end
289 end
290
291 % Close all files
292 for i=1:total_weight_files
293 fclose(W_files(i));
294 end
295 for i=1:total_bias_files
296 fclose(b_files(i));
297 end
298
299 % Propagate the test dataset
300 test_error = 0;
```

```
301 output = zeros(test_ds_size,1);
302 for i=1:test_ds_size
303     % Initial condition
304     a{1} = test_ds(i,1);
305     for k=1:num_layers
306         W_aux = cell2mat(W(k));
307         a_aux = cell2mat(a(k));
308         b_aux = cell2mat(b(k));
309         n_f = W_aux*a_aux+b_aux;
310         a{k+1} = get_activation_function(n_f, functions_vector(k));
311     end
312     test_data = cell2mat(a(1));
313     a_aux = cell2mat(a(num_layers + 1));
314     test_error = test_error + (1 / test_ds_size) * (test_ds(i,2) - a_aux);
315     output(i) = a_aux;
316 end
317
318 % Print last errors
319 fprintf('Error de aprendizaje = %f\n', learning_error);
320 fprintf('Error de validación = %f\n', validation_error);
321 fprintf('Error de prueba = %f\n', test_error);
322
323
324 % Output vs test
325 scatter_output_vs_test(test_ds, output);
326 % Propagate the training size for plotting
327
328 output = zeros(training_ds_size,1);
329 for i=1:training_ds_size
330     % Initial Condition
331     a{1} = training_ds(i, 1);
332     for k=1:num_layers
333         W_aux = cell2mat(W(k));
334         a_aux = cell2mat(a(k));
335         b_aux = cell2mat(b(k));
336         a{k+1} = get_activation_function(W_aux*a_aux+b_aux,functions_vector(k));
337     end
338     a_aux = cell2mat(a(num_layers + 1));
339     test_error = test_error + (1 / training_ds_size) * (training_ds(i,2) - a_aux);
340     output(i) = a_aux;
341 end
342
343 scatter_output_vs_training(training_ds, output);
344
345 % Plot the error evolution
346 error_plot(validation_iter, num_validation_epoch, learning_err_values, epoch,
            evaluation_err_values);
347 % Plot weight evolution
348 weight_evolution_plot(architecture, num_layers, epoch);
```

```
349 %Plot bias evolution
350 bias_evolution_plot(architecture, num_layers, epoch);
351
352 %Write final values
353 for i=1:num_layers
354 path = strcat(pwd, '/Valores_finales/capa_', num2str(i), '/');
355 if ~exist(path, 'dir')
356 mkdir(path);
357 end
358 W_aux = cell2mat(W(i));
359 res_pesos = strcat(path, '/pesos.txt');
360 dlmwrite(res_pesos, W_aux, ';');
361 end
362
363 for i=1:num_layers
364 path = strcat(pwd, '/Valores_finales/capa_', num2str(i), '/');
365 if ~exist(path, 'dir')
366 mkdir(path);
367 end
368 b_aux = cell2mat(b(i));
369 res_bias = strcat(path, '/bias.txt');
370 dlmwrite(res_bias, b_aux, ';');
371 end
```

Listing 2: bias\_evolution\_plot.m

```
1 function bias_evolution_plot(architecture, num_layers, epoch)
2 num_epochs = 0:1:epoch;
3 for i=1:num_layers
4 figure('Name', sprintf('Evolución de bias capa %d', i), 'NumberTitle', 'off')
5 path = strcat(pwd, '/historico/capa_', num2str(i), '/bias/');
6 for j=1:architecture(i+1)
7 file = strcat(path, '/bias', num2str(j), '.txt');
8 identifier = strcat('b(', num2str(j), ')');
9 evolution = importdata(file);
10 plot(num_epochs, evolution, 'DisplayName', identifier);
11 hold on
12 grid on
13 end
14 title(sprintf('Evolución de bias capa %d', i));
15 ylabel('Valor');
16 xlabel('Epoch');
17 title(legend('show', 'Location', 'northwestoutside'), 'Leyenda');
18 hold off
19 end
20 end
```

Listing 3: dataset\_slices.m

---

```
1 function [training_ds, test_ds, validation_ds] = dataset_slices (opcion, inputs,
   targets)
2 % Get the data size
3 num_data = size (inputs, 1)
4 % COnfiguration options
5 if opcion == 1
6 training_num = ceil (num_data * .8);
7 test_num = floor (num_data * 0.1 );
8 validation_num = floor (num_data * 0.1);
9 else
10 training_num = ceil (num_data * 0.7)
11 test_num = floor (num_data * 0.15)
12 validation_num = floor (num_data * 0.15)
13 end
14 % Initialize the datasets
15 training_ds = zeros (training_num, 2);
16 test_ds = zeros (test_num, 2);
17 validation_ds = zeros (validation_num, 2);
18 a1 = 2:ceil(num_data / validation_num):num_data-1
19 aux = setdiff(1:num_data, a1);
20 j = 2;
21 a2 = [];
22 increment = ceil(size(aux, 2) / test_num);
23 for i=1:test_num
24 if (j < size(aux, 2))
25 a2 = [a2, aux(j)];
26 else
27 a2 = [a2, aux(end - 1)];
28 end
29 j = j + increment;
30 end
31 a2
32 a3 = setdiff(aux, a2)
33
34 % Validation Slice
35 for i = 1:size(a1, 2)
36 validation_ds (i, 1) = inputs(a1(i));
37 validation_ds (i, 2) = targets(a1(i));
38 end
39
40 % test Slice
41 for i = 1:size(a2, 2)
42 test_ds (i, 1) = inputs(a2(i));
43 test_ds (i, 2) = targets(a2(i));
44 end
45
46
47 % training Slice
48 for i = 1:size(a3, 2)
```

```
49 training_ds(i, 1) = inputs(a3(i));
50 training_ds(i, 2) = targets(a3(i));
51 end
52 end
```

Listing 4: get\_activation\_function.m

```
1 function a = get_activation_function(n, config)
2 if (config == 1)
3 a = purelin(n);
4 elseif (config == 2)
5 a = logsig(n);
6 else
7 a = tansig(n);
8 end
9 end
```

Listing 5: get\_F\_matrix.m

```
1 function F = get_F_matrix(option, total_neurons, a)
2 if (option == 1)
3 F = diag(ones(1, total_neurons));
4 elseif (option == 2)
5 F = diag(logsig('dn', a, a));
6 else
7 F = diag(tansig('dn', a, a));
8 end
9 end
```

Listing 6: scatter\_output\_vs\_test.m

```
1 function scatter_output_vs_test(test_ds, output)
2 % Targets vs MLP output PLOT
3 figure('Name','Targets del dataset de prueba vs Output', 'NumberTitle', 'off')
4 grid on
5 hold on
6 title('Targets del dataset de prueba vs Output');
7 ylabel('G(p)');
8 xlabel('rango de la señal');
9 signal_range = test_ds(:, 1);
10 % targets
11 scatter(signal_range, test_ds(:, 2), 'd', 'MarkerEdgeColor', 'black');
12 % outputs
13 scatter(signal_range, output, 5, 'MarkerFaceColor', 'blue', 'MarkerEdgeColor', 'blue'
14 );
15 title(legend('Target', 'Output', 'Location', 'northwestoutside'), 'Leyenda');
16 hold off
17 end
```

Listing 7: scatter\_output\_vs\_training.m

---

```
1 function scatter_output_vs_training(training_ds, output)
2 % Targets vs MLP output PLOT
3 figure('Name','Targets del dataset de entrenamiento vs Output', 'NumberTitle', 'off')
4 grid on
5 hold on
6 title('Targets del dataset de entrenamiento vs Output');
7 ylabel('G(p)');
8 xlabel('rango de la señal');
9 signal_range = training_ds(:, 1);
10 % targets
11 scatter(signal_range, training_ds(:,2), 'd', 'MarkerEdgeColor', 'black');
12 % outputs
13 scatter(signal_range, output, 5, 'MarkerFaceColor', 'blue', 'MarkerEdgeColor', 'blue'
14         );
15 title(legend('Target', 'Output', 'Location', 'northwestoutside'),'Leyenda');
16 hold off
17 end
```

Listing 8: weight\_evolution\_plot.m

```
1 function weight_evolution_plot(architecture, num_layers, epoch)
2 num_epochs = 0:1:epoch;
3 for i=1:num_layers
4 figure('Name',sprintf('Evolución de pesos capa %d', i), 'NumberTitle', 'off')
5 path = strcat(pwd, '/historico/capa_', num2str(i), '/pesos/');
6 for j = 1:architecture(i + 1)
7 for k = 1:architecture(i)
8 W_file = strcat(path, '/pesos',num2str(j), '_', num2str(k), '.txt');
9 identifier = strcat('W(',num2str(j),',',num2str(k),')');
10 evolution = importdata(W_file);
11 plot(num_epochs, evolution, 'DisplayName', identifier);
12 hold on
13 grid on
14 end
15 end
16 title(sprintf('Evolución de pesos capa %d', i));
17 xlabel('Epoch');
18 ylabel('Valor');
19 title(legend('show', 'Location', 'northwestoutside'), 'Leyenda');
20 hold off
21 end
22 end
```