Práctica #5 Perceptrón Multicapa

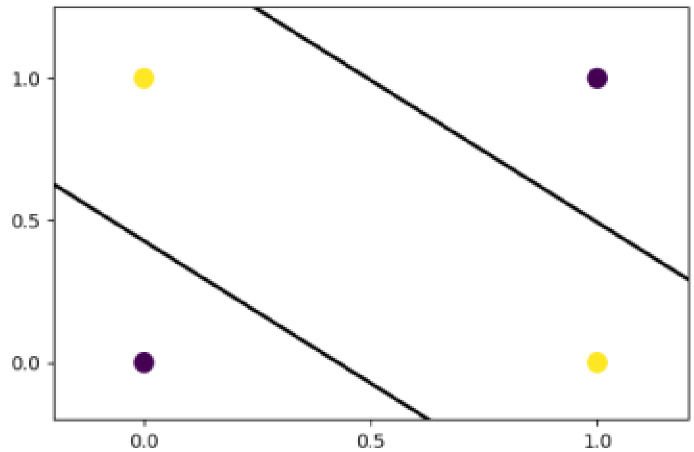
Jorge Gómez Reus

Índice

1. Introducción

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

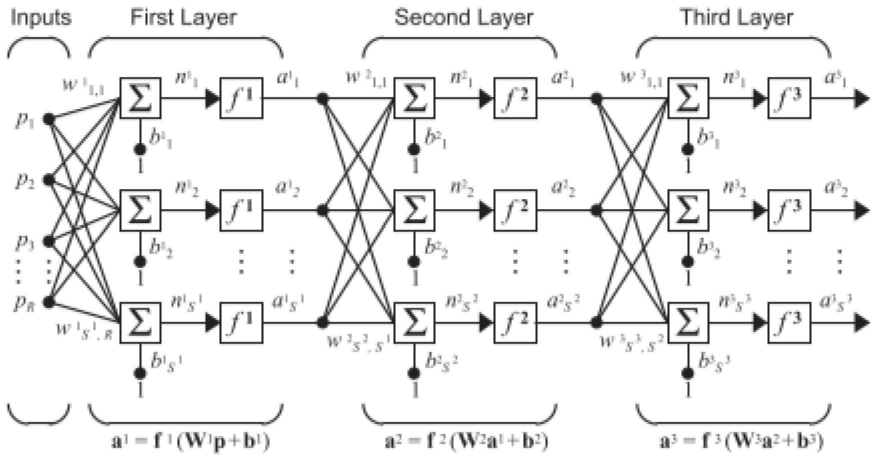
Figura 1: Compuerta XOR



En este caso se necesitan dos fronteras de desición, por ende dos neuron 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 minización 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

Figura 2: Modelo



1.1.1. Foward Propagation

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

donde M es el número de capas.

1.1.2. Foward 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 sensitividades 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

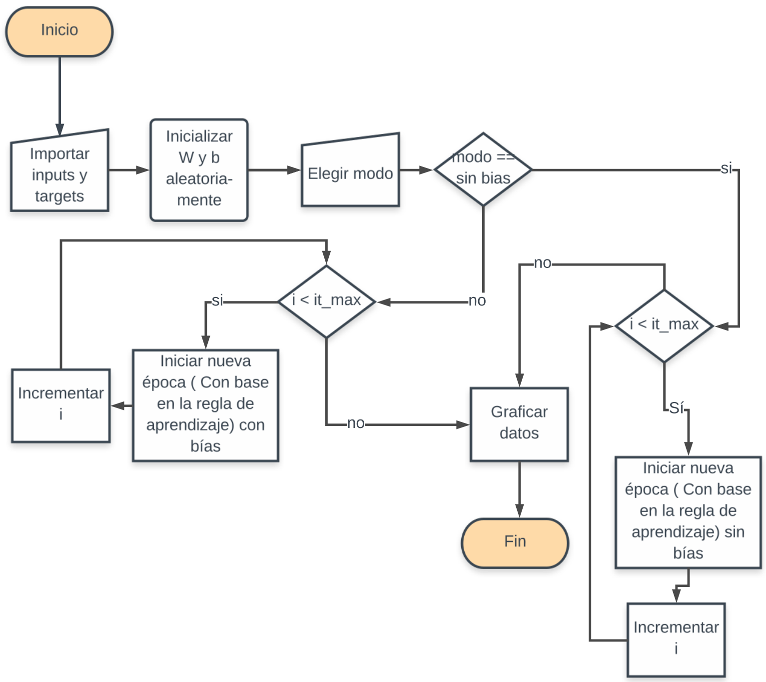


Figura 3: Diagrama de Flujo

3. Resultados

3.1. Polinomio 1

3.2. Inputs

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

3.3. Targets

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

3.3.1. Datos

$$V1 = \begin{bmatrix} 11 & 16 & 10 & 1 \end{bmatrix}$$

$$V2 = \begin{bmatrix} 3 & 2 & 1 \end{bmatrix}$$

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

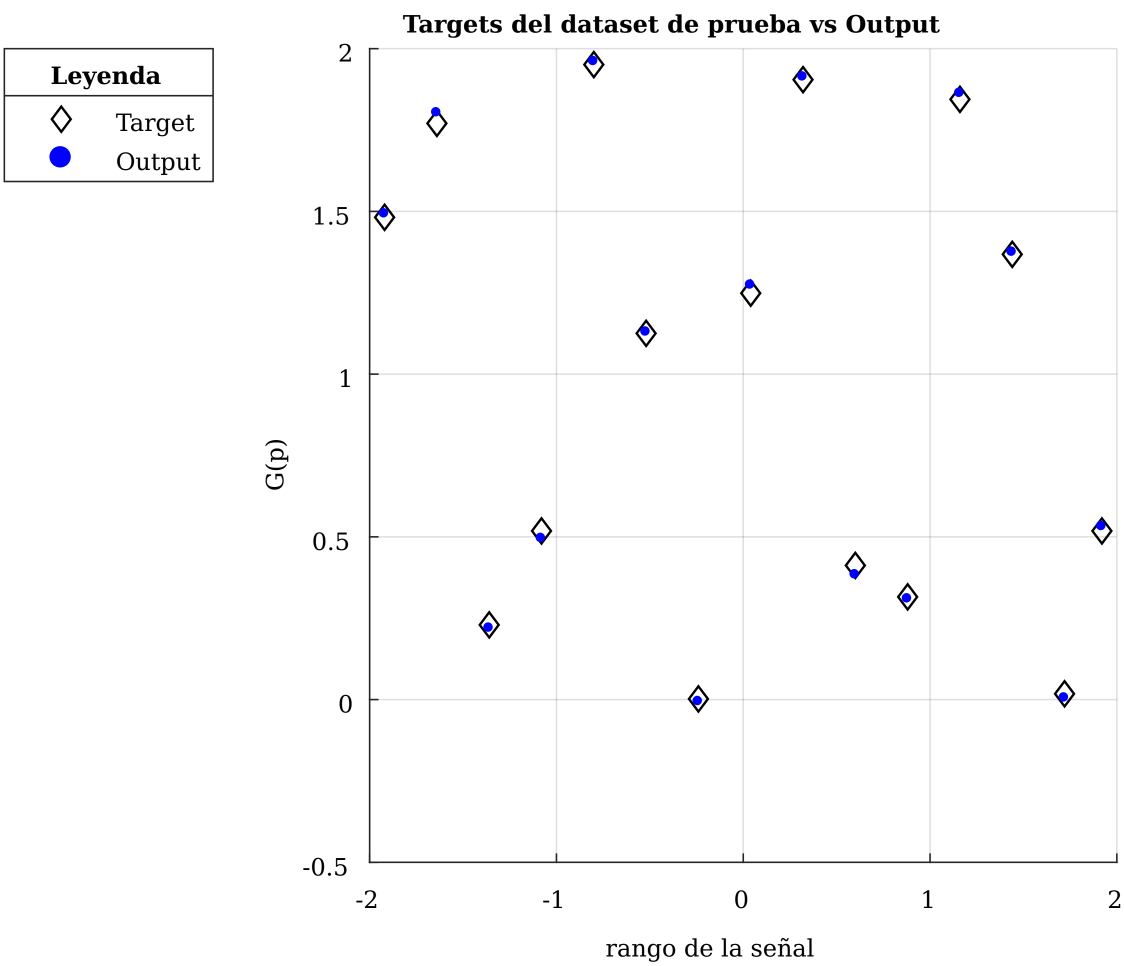


Figura 4: Gráfica 1.1

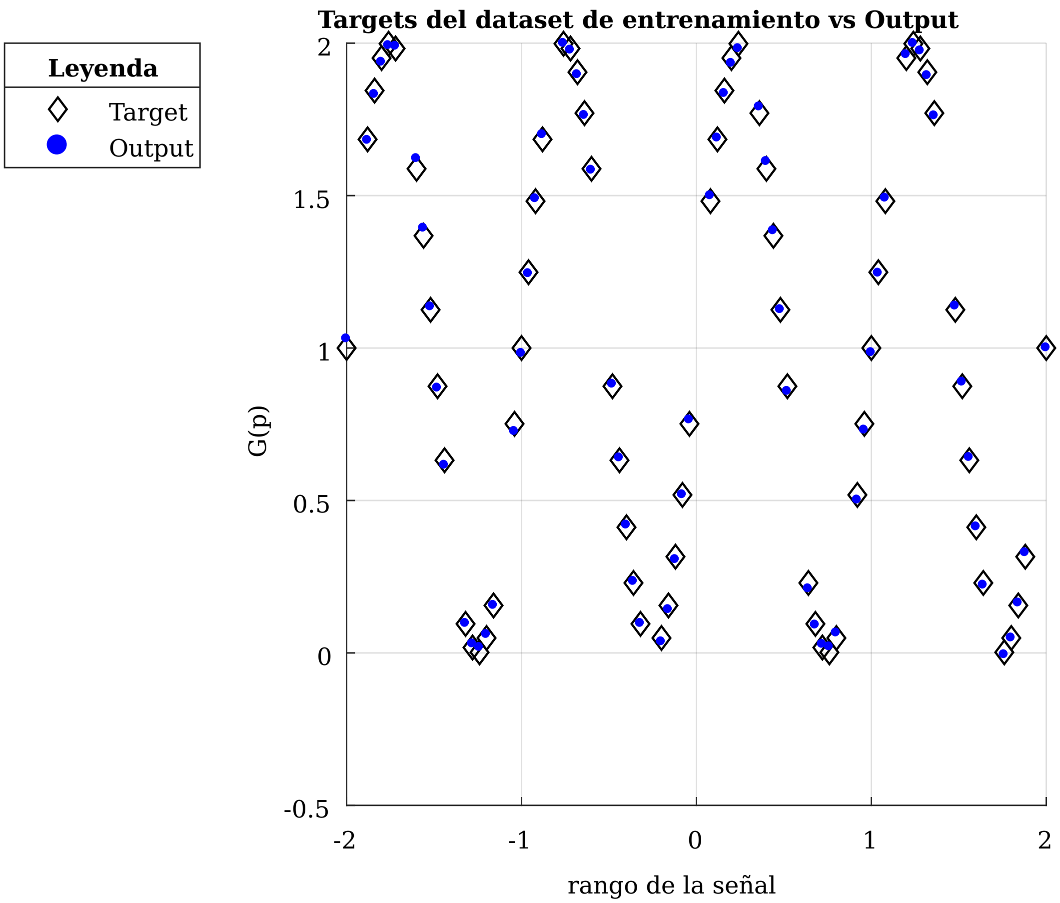


Figura 5: Gráfica 1.2

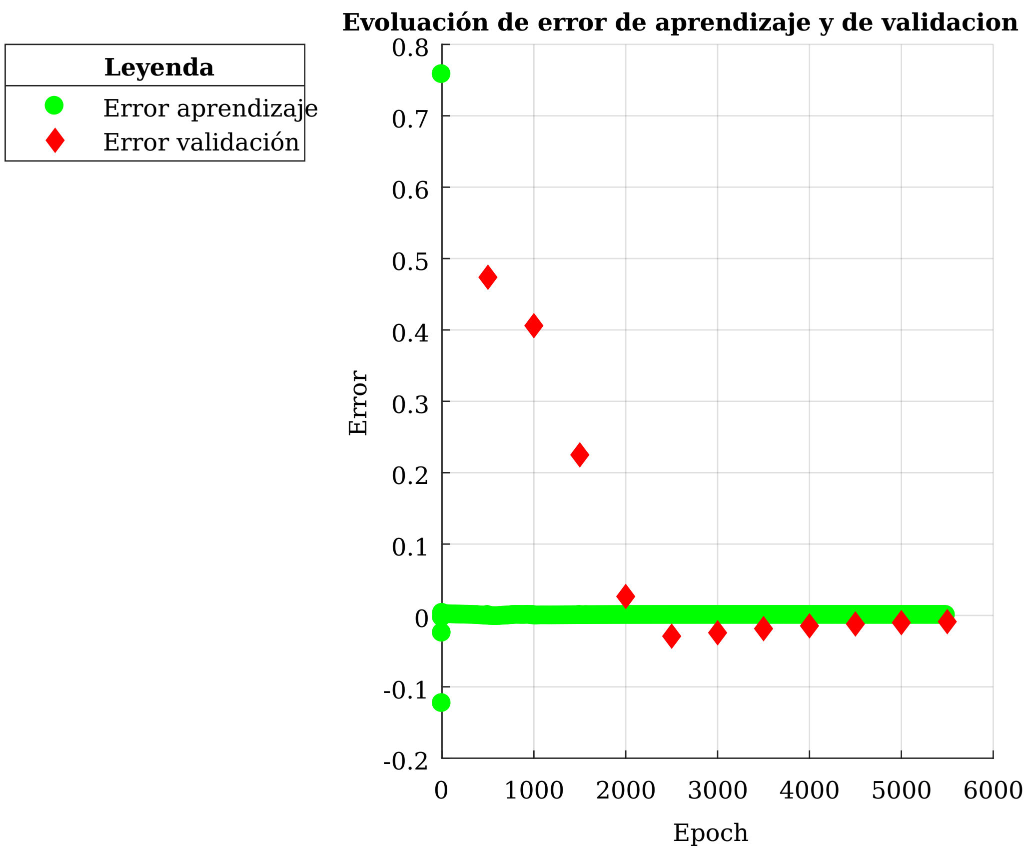


Figura 6: Gráfica 1.3

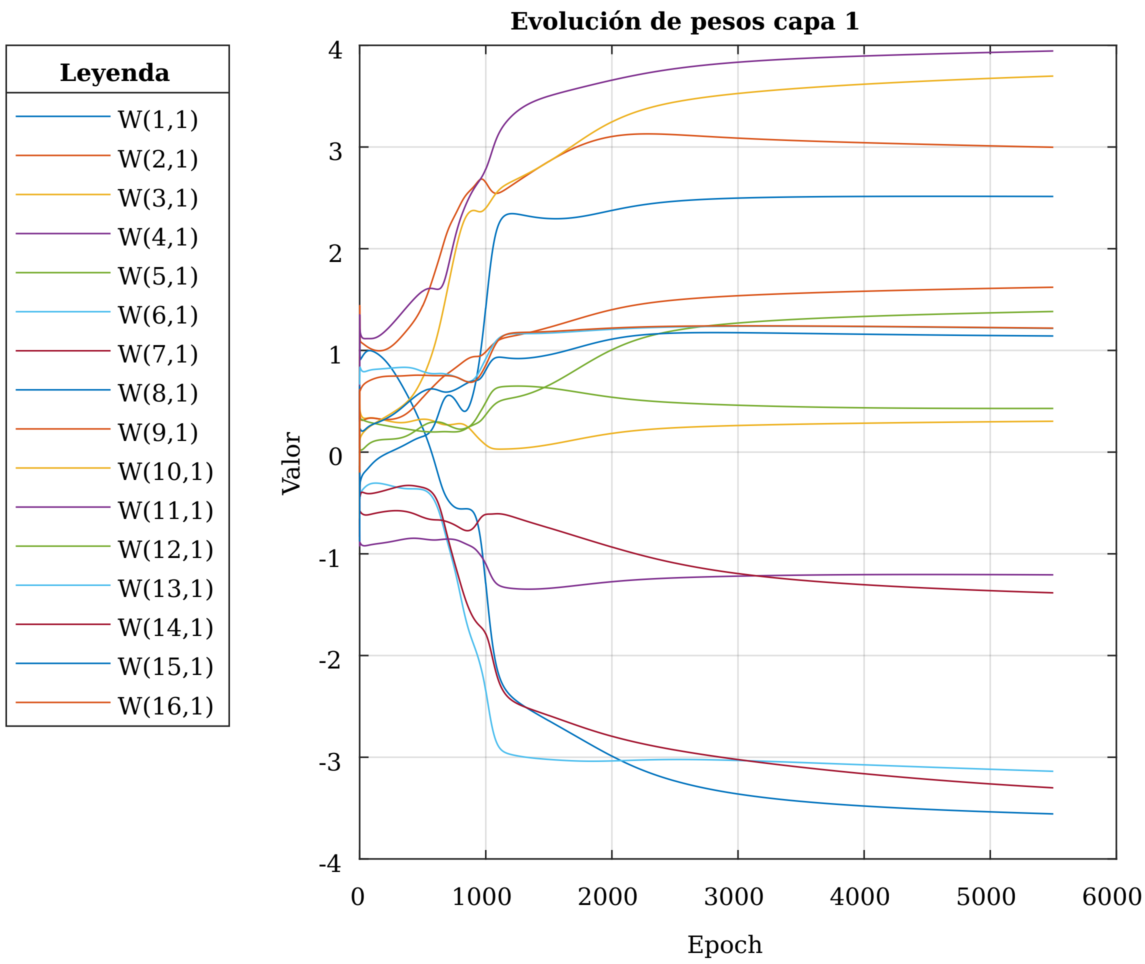


Figura 7: Gráfica 1.4

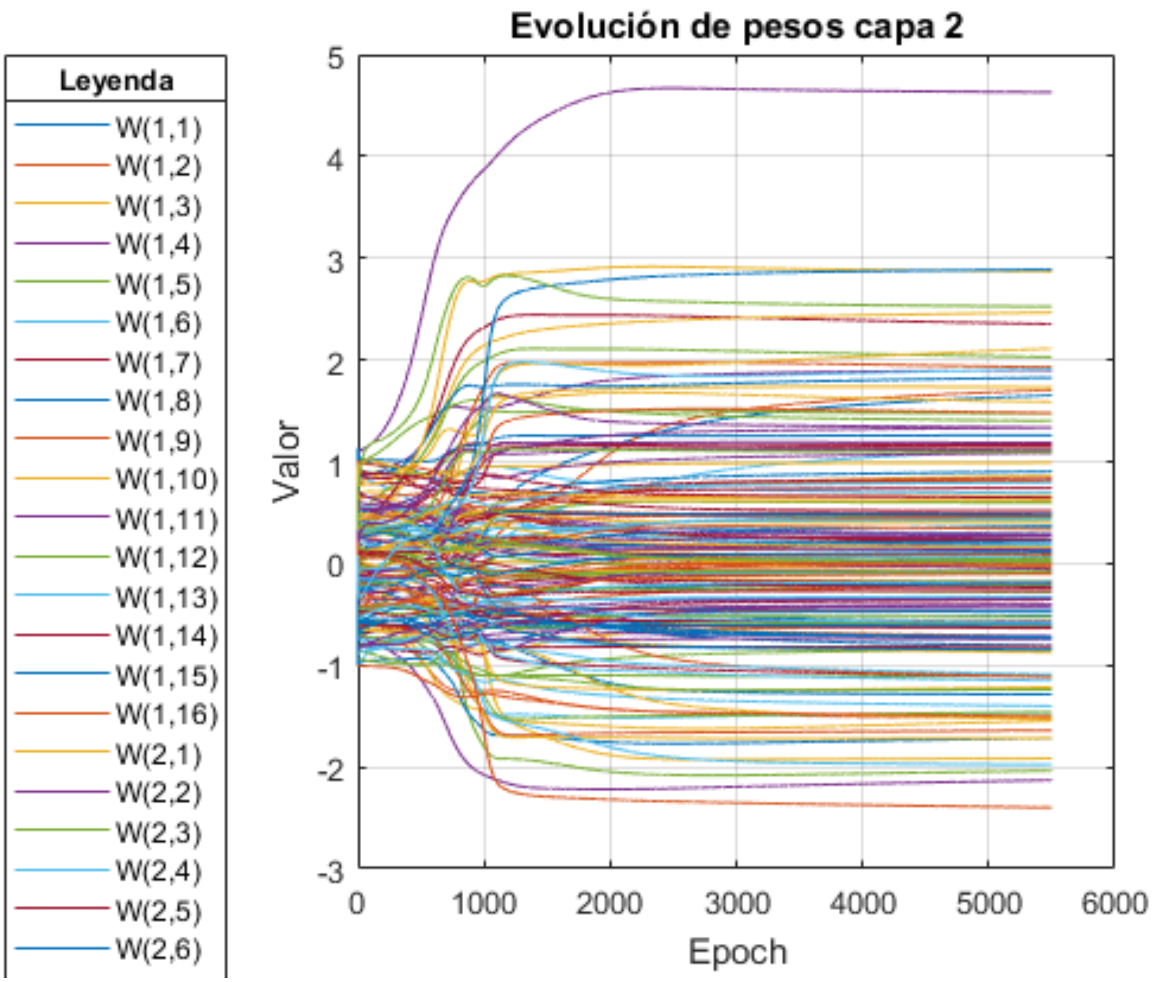


Figura 8: Gráfica 1.5

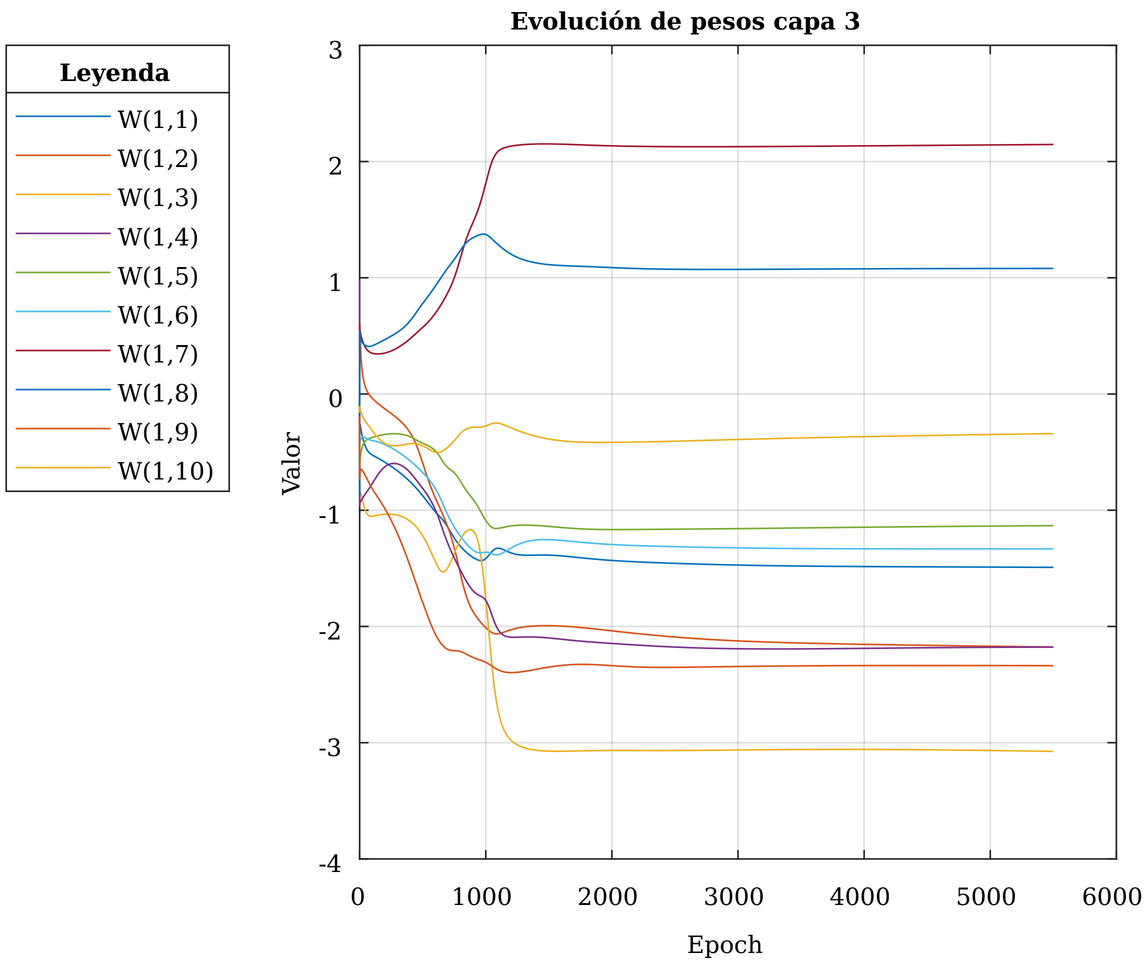


Figura 9: Gráfica 1.6

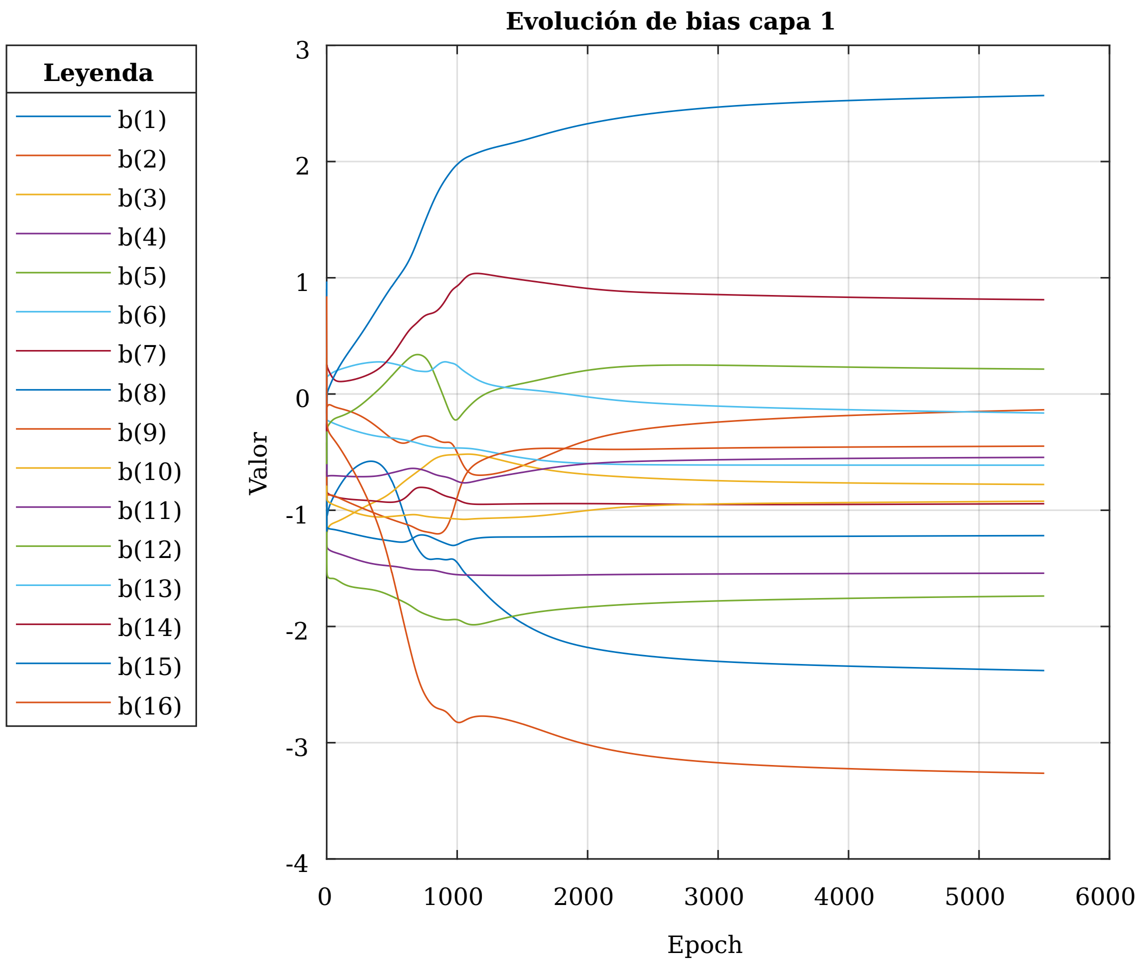


Figura 10: Gráfica 1.7

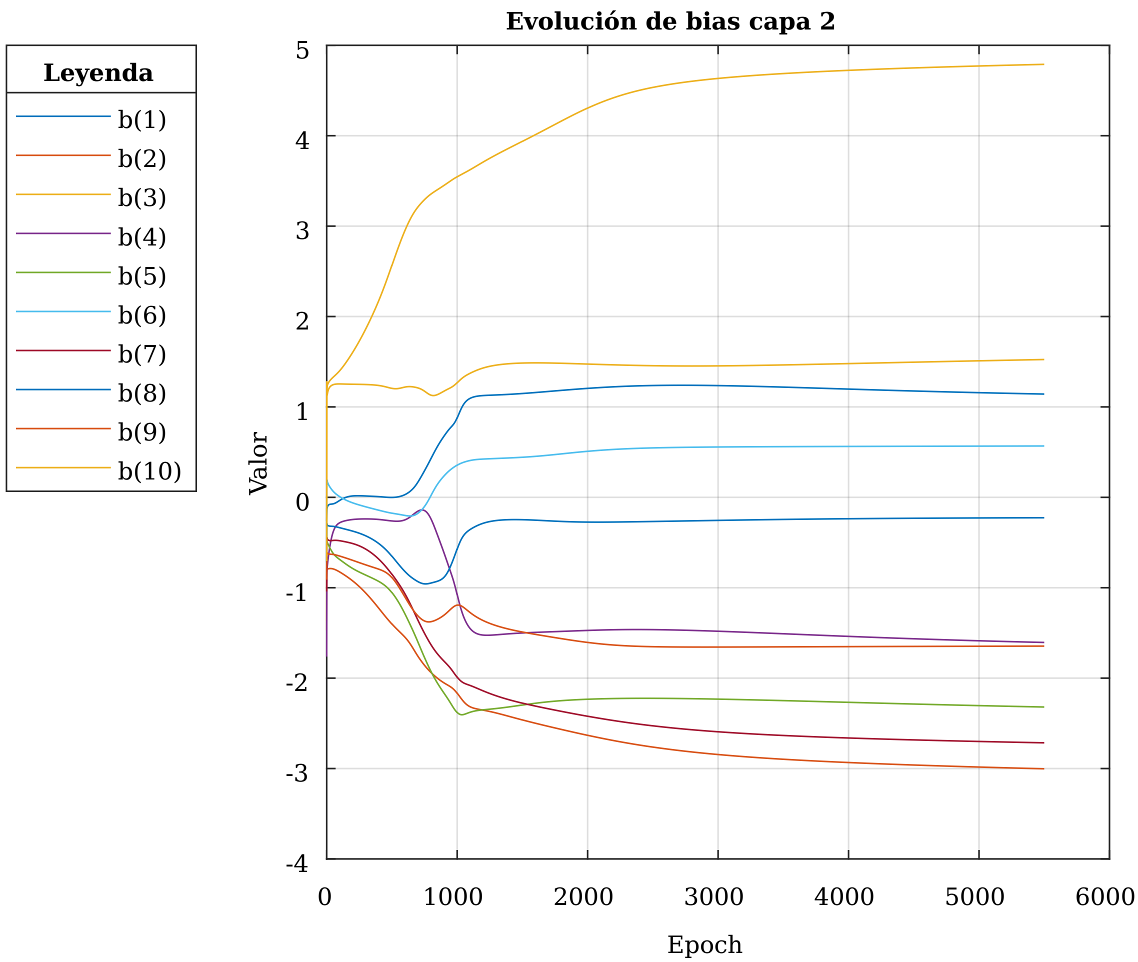


Figura 11: Gráfica 1.8

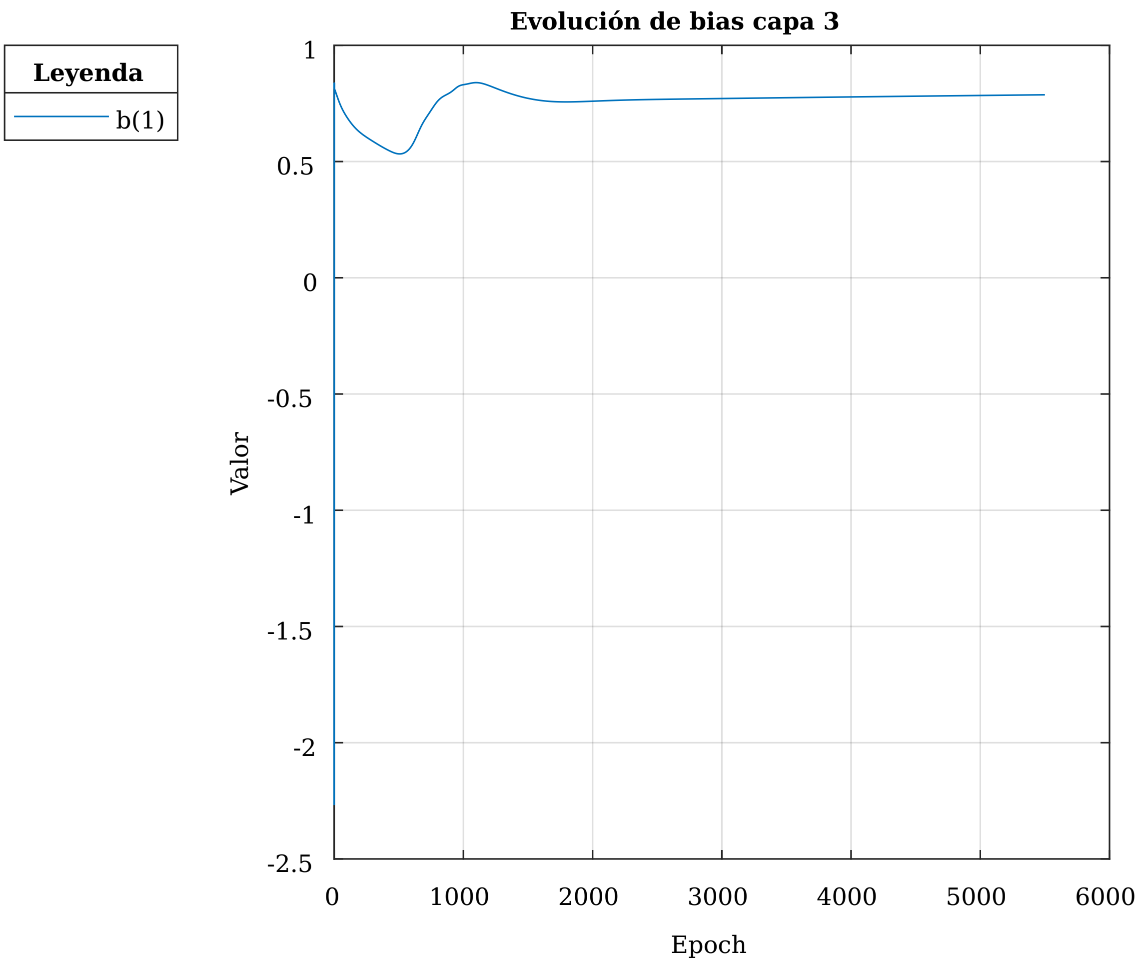


Figura 12: Gráfica 1.9

3.6. Polinomio 2

3.7. Inputs

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

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

3.8. Targets

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

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

3.8.1. Datos

$$V1 = \begin{bmatrix} 11 & 16 & 10 & 1 \end{bmatrix}$$

$$V2 = \begin{bmatrix} 3 & 2 & 1 \end{bmatrix}$$

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

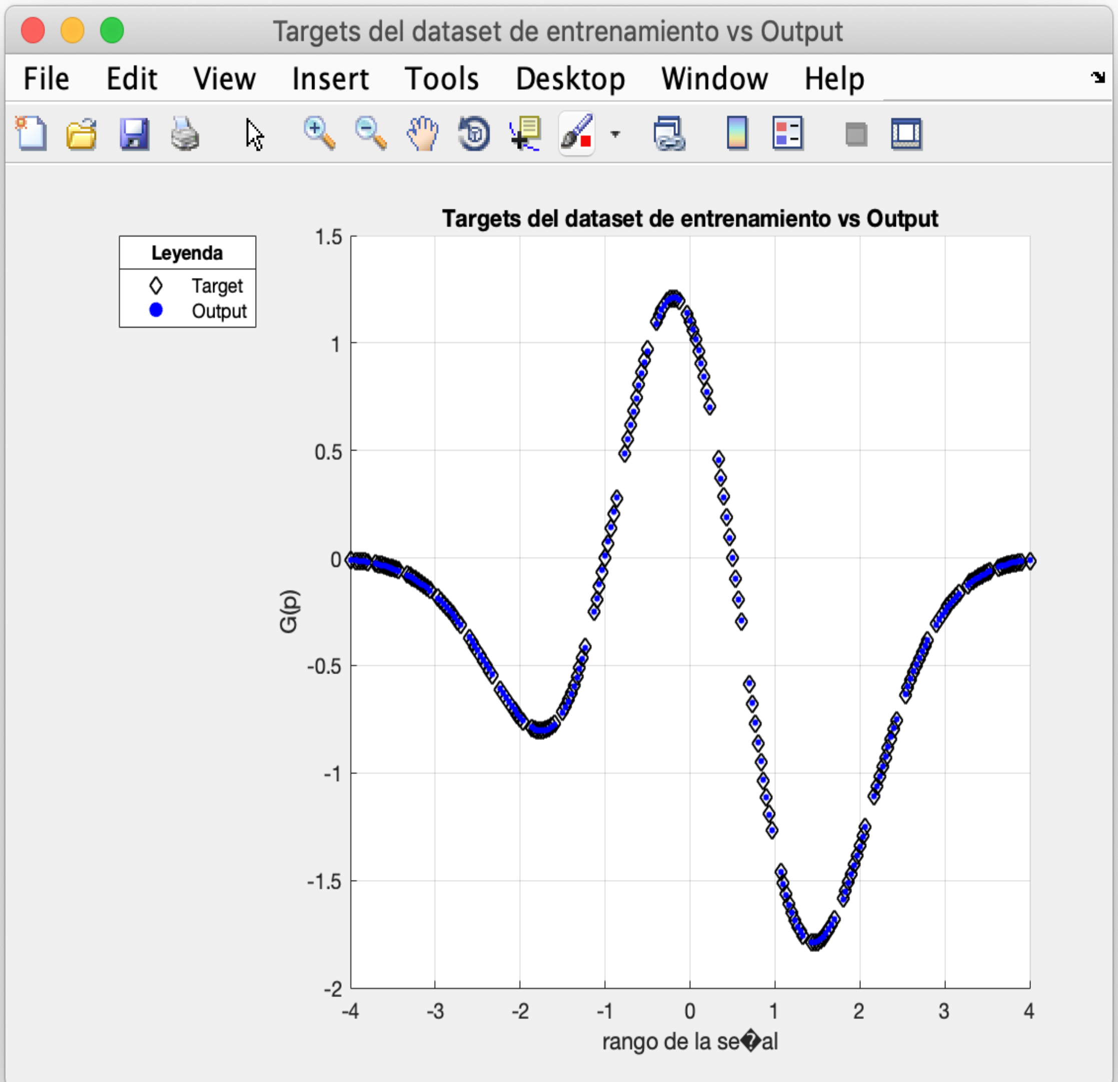


Figura 13: Gráfica 1.1

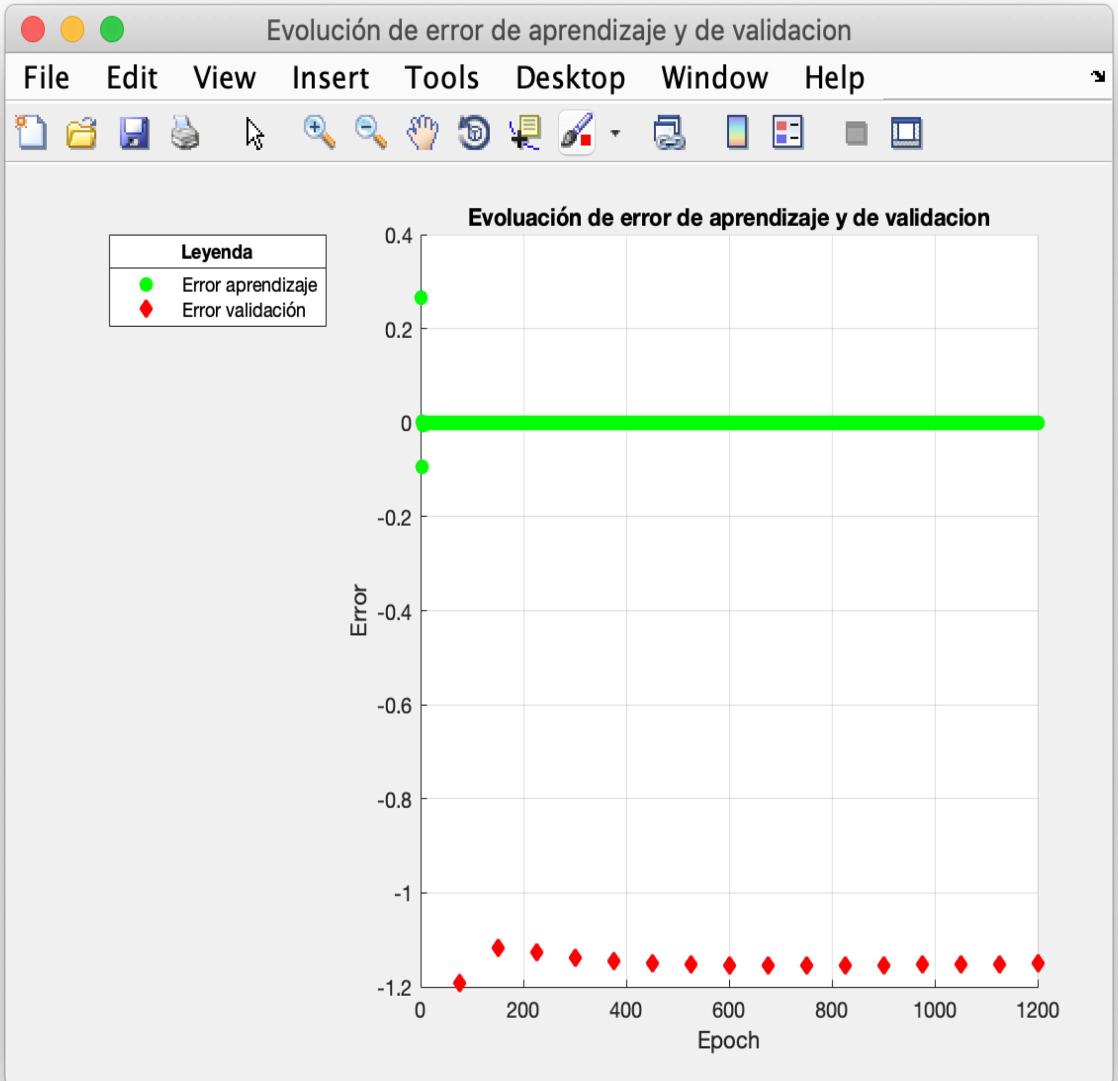


Figura 14: Gráfica 1.2

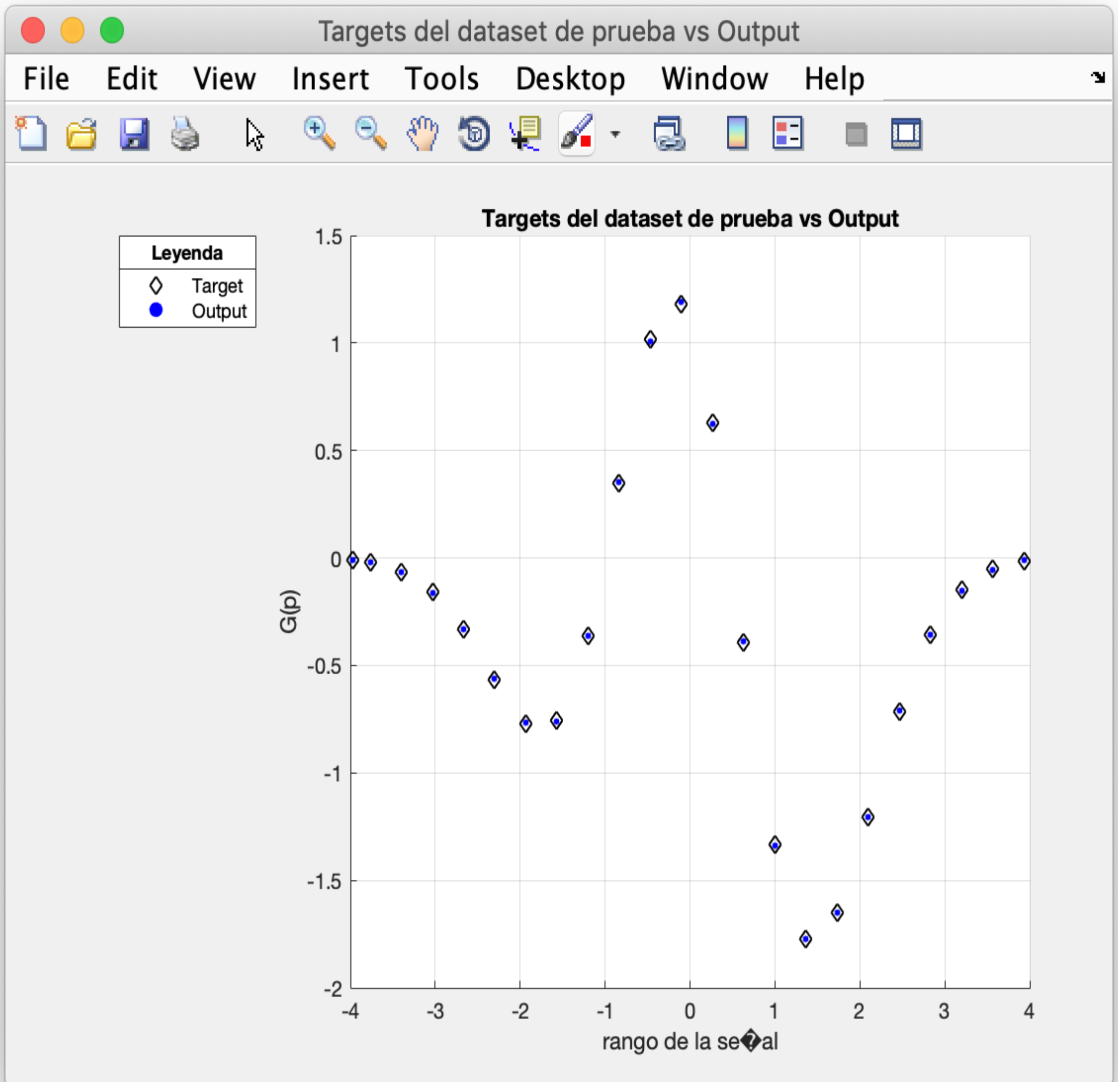


Figura 15: Gráfica 1.3

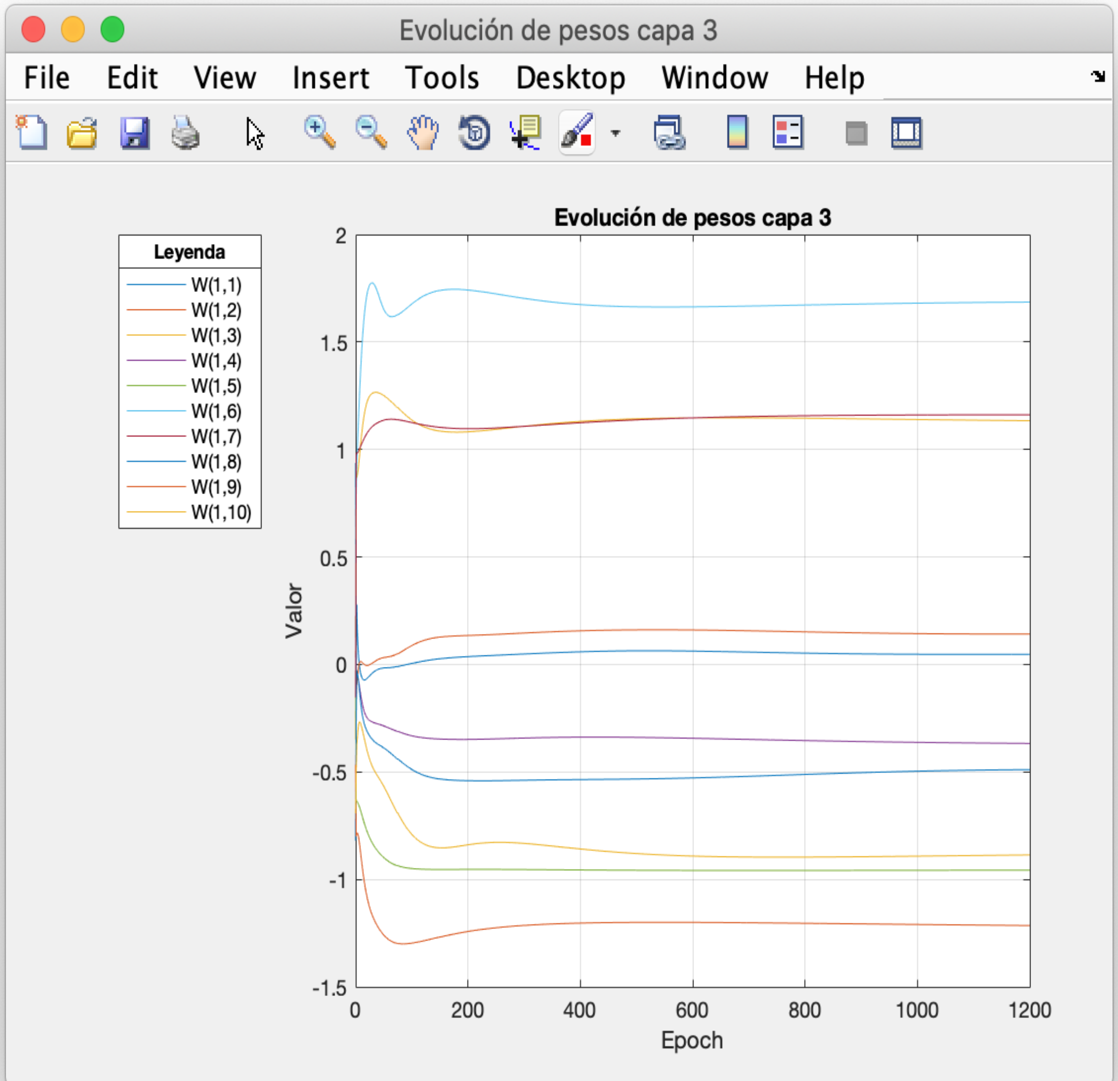


Figura 16: Gráfica 1.4

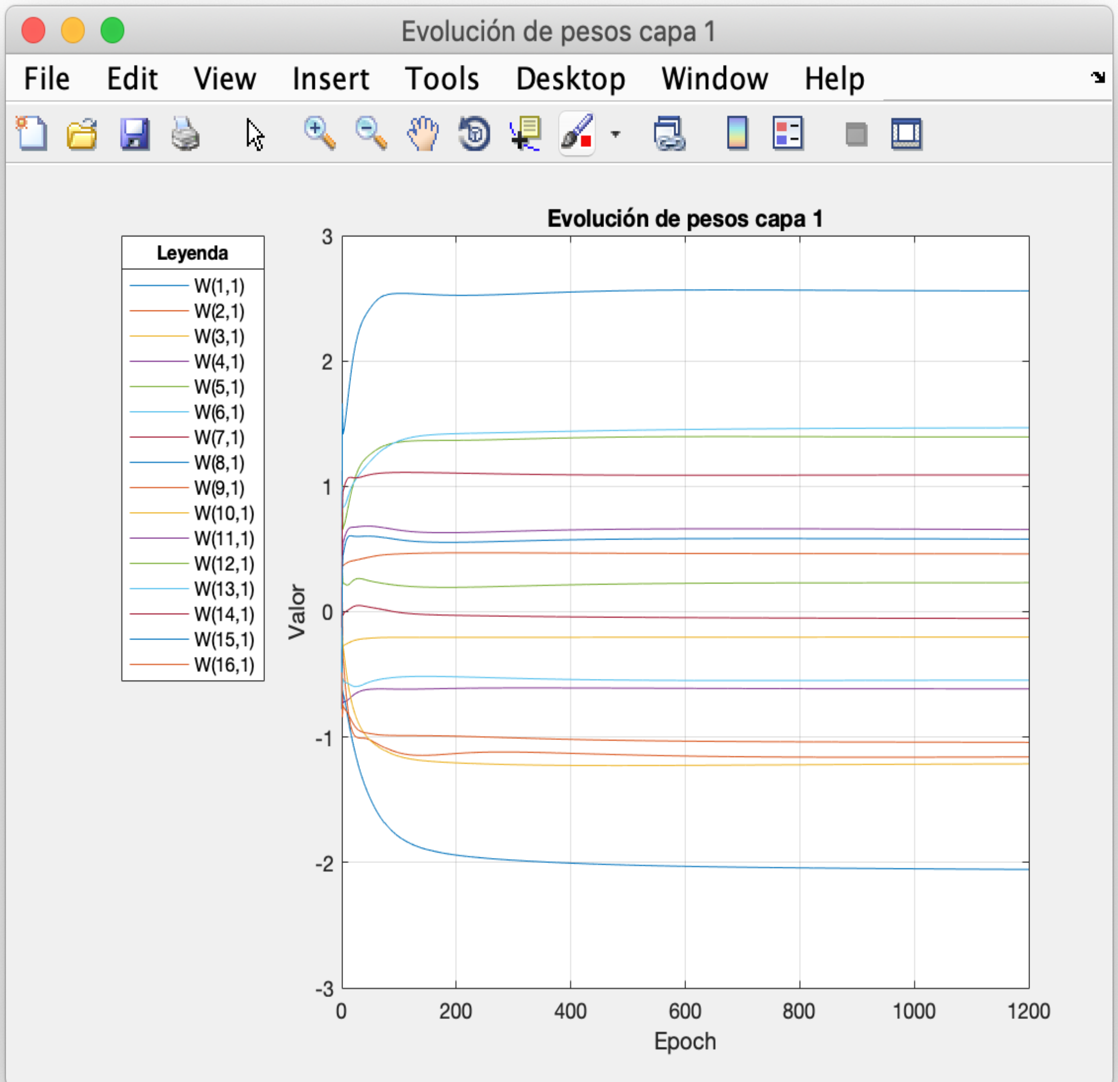


Figura 17: Gráfica 1.5

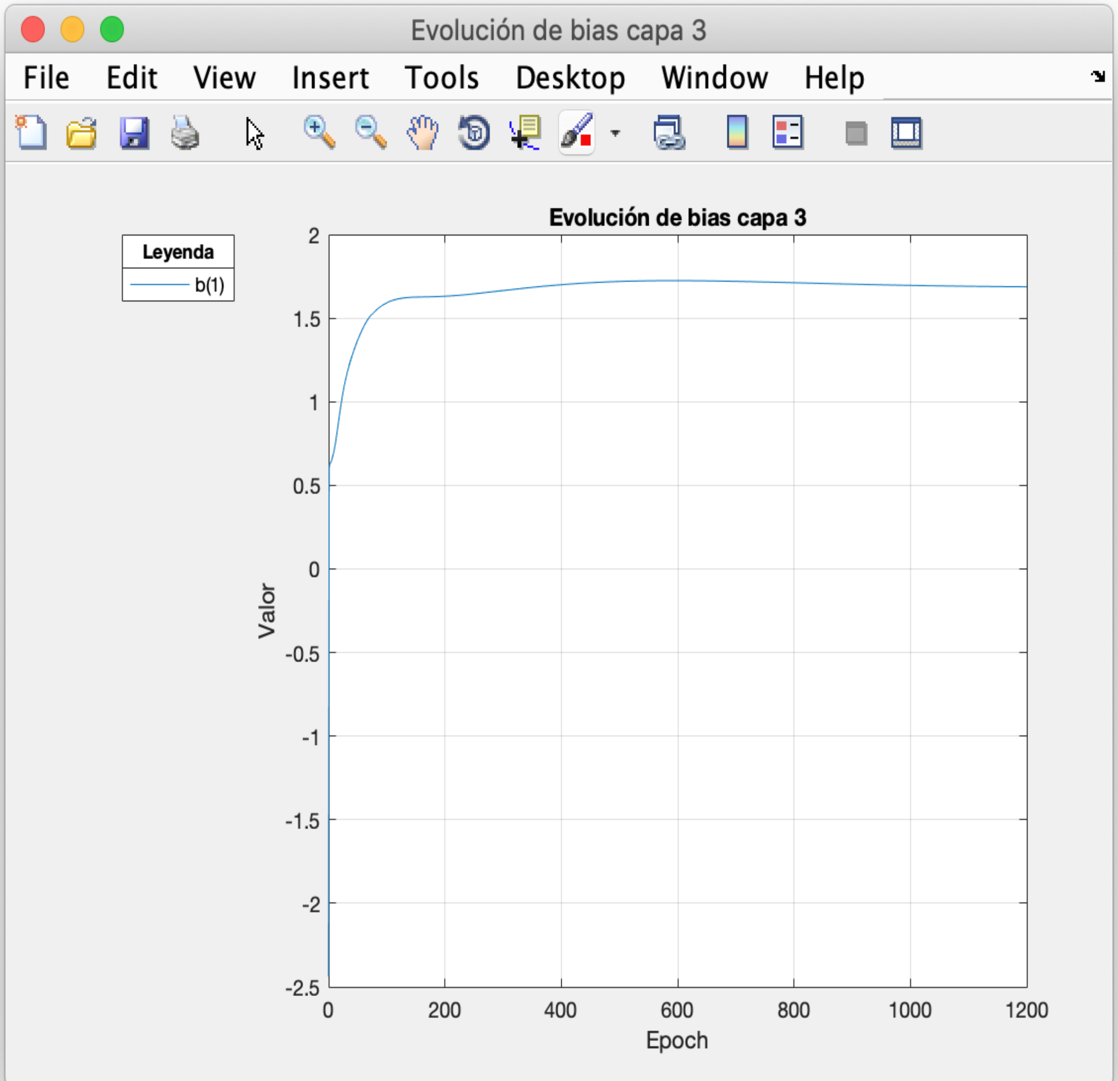


Figura 18: Gráfica 1.6

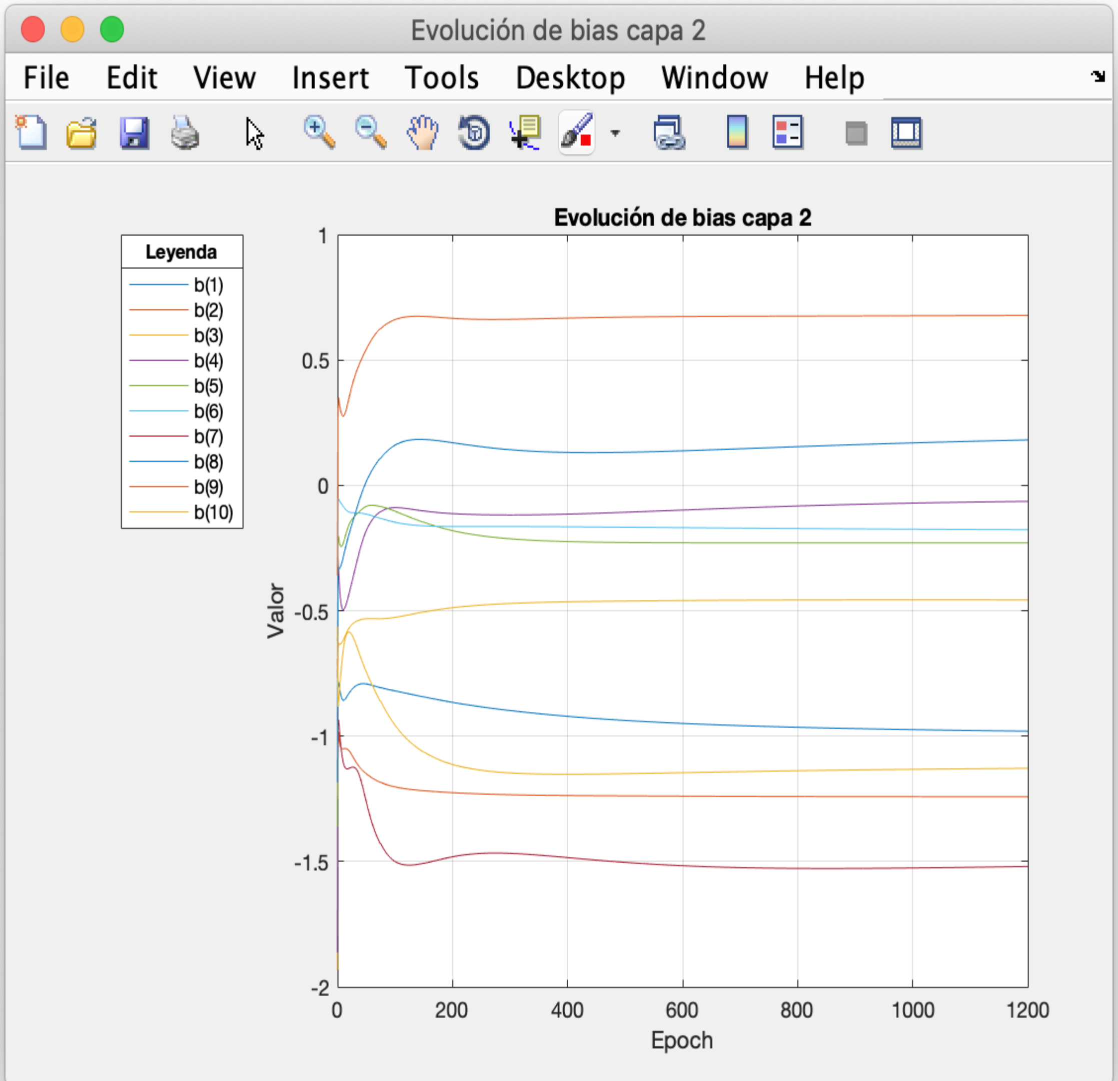


Figura 19: Gráfica 1.7

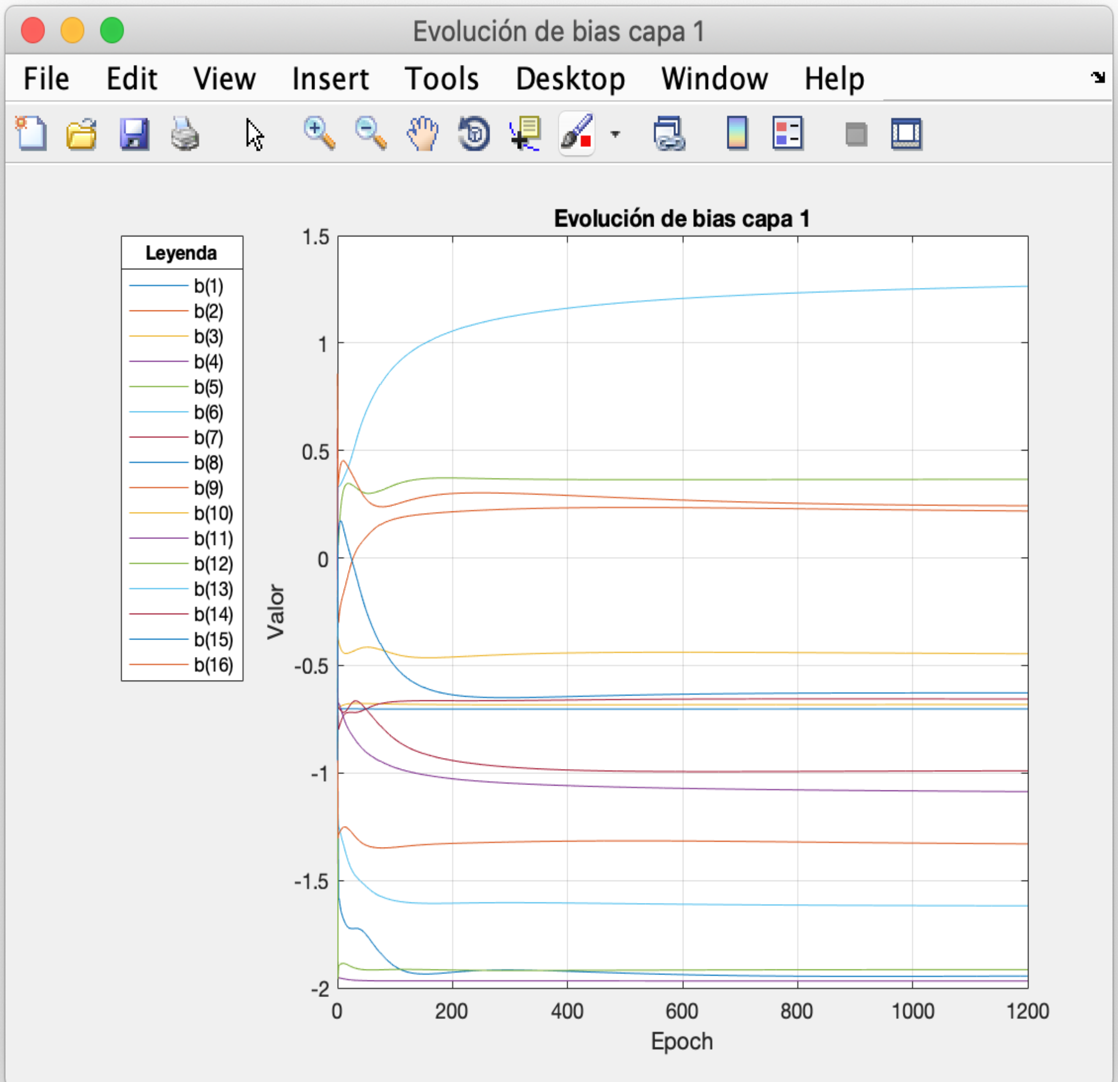


Figura 20: Gráfica 1.8

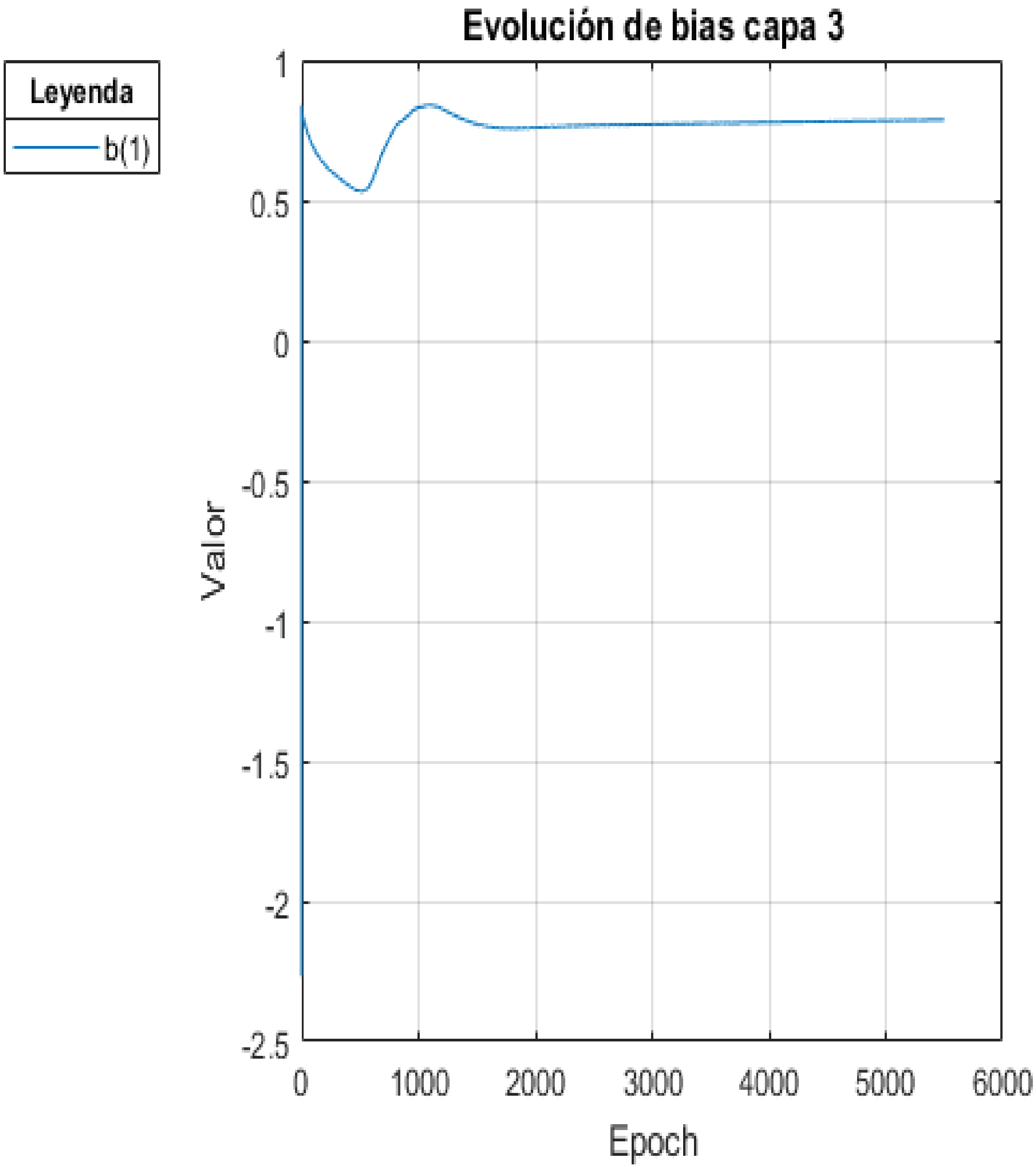


Figura 21: Gráfica 1.9

3.11. Polinomio 3

3.12. Inputs

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

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

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

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

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

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

3.13. Targets

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

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

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

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

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

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

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

3.13.1. Datos

$$V1 = \begin{bmatrix} 11 & 16 & 10 & 1 \end{bmatrix}$$

$$V2 = \begin{bmatrix} 3 & 2 & 1 \end{bmatrix}$$

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

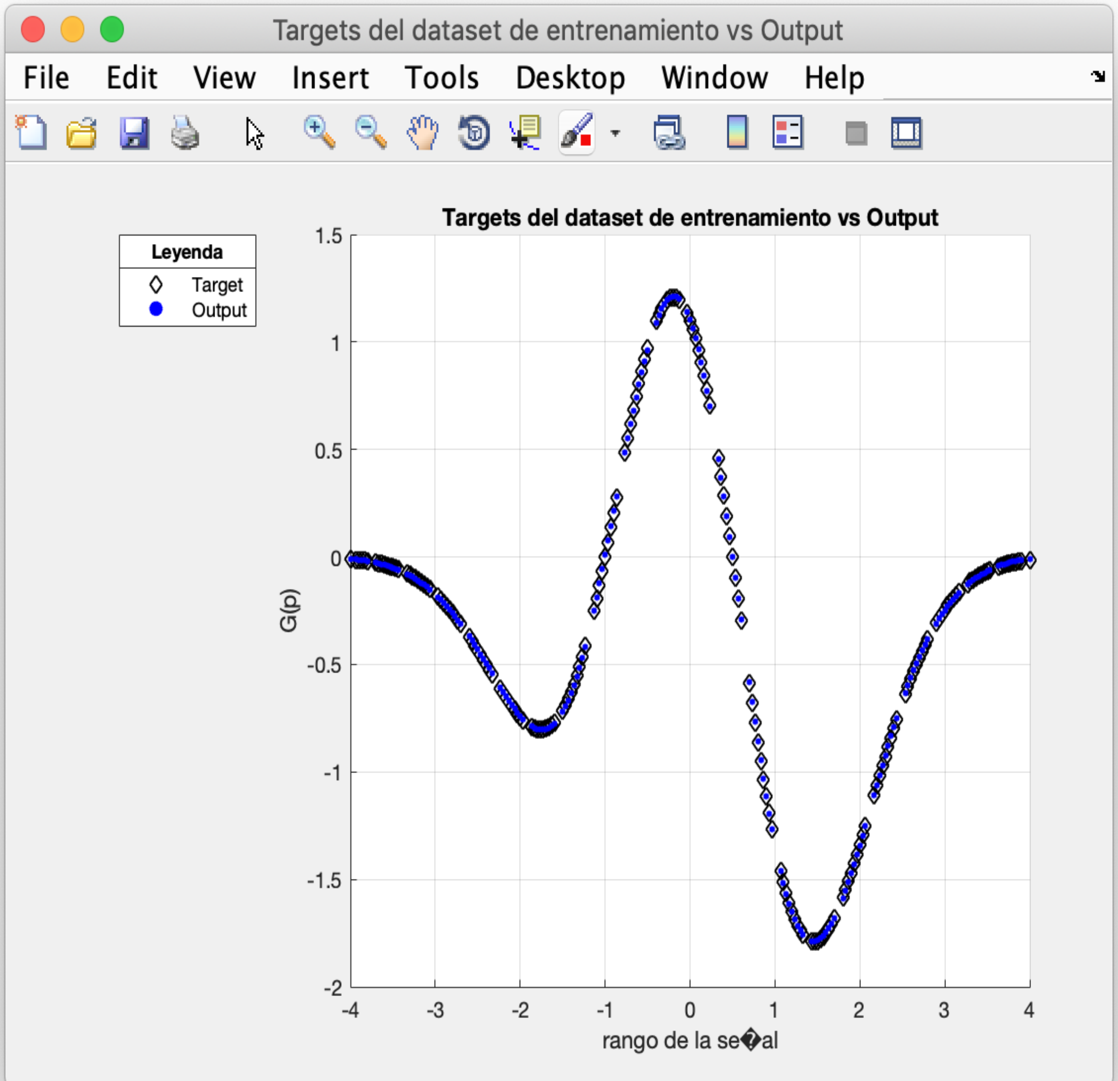


Figura 22: Gráfica 3.1

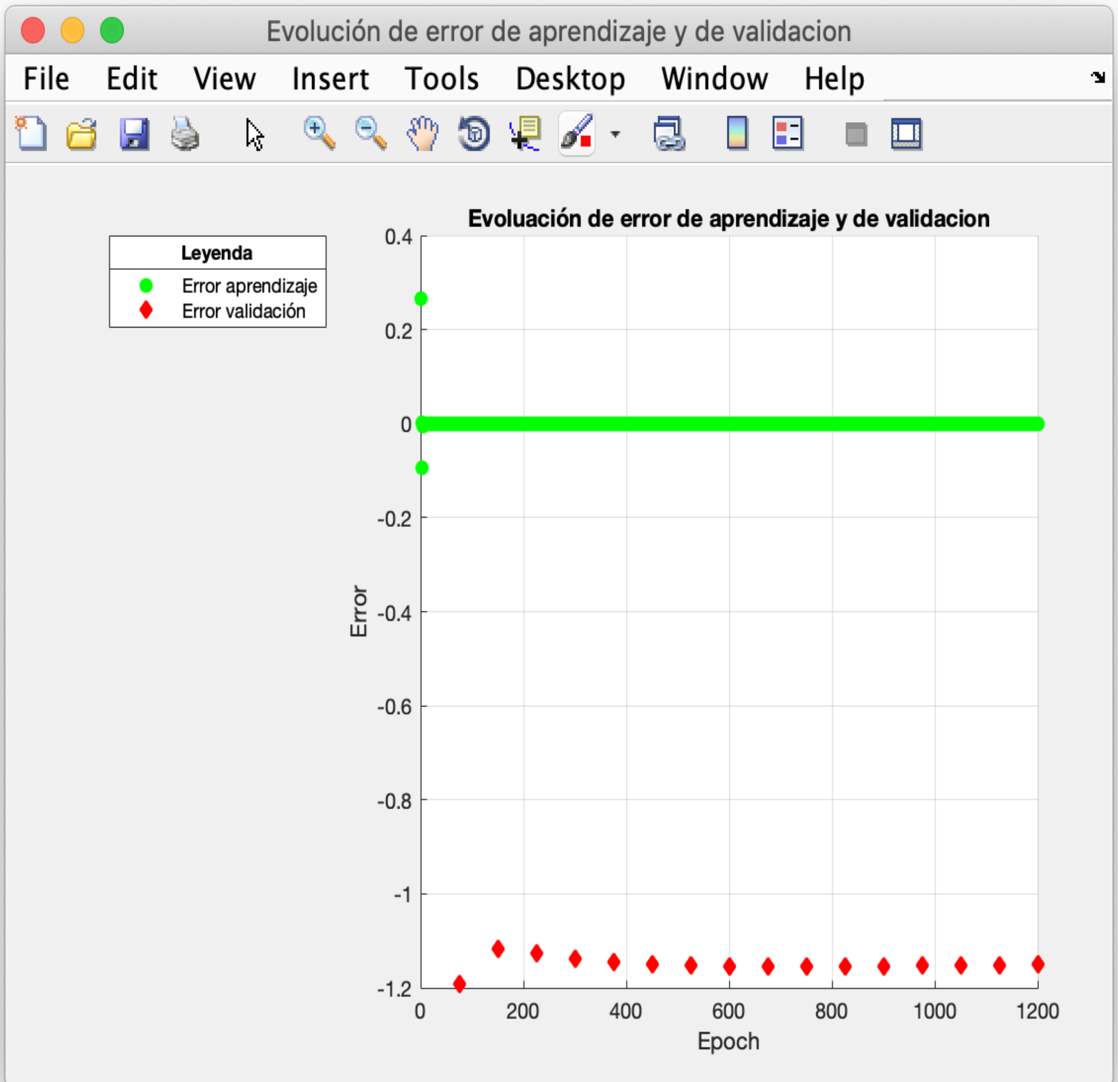


Figura 23: Gráfica 3.2

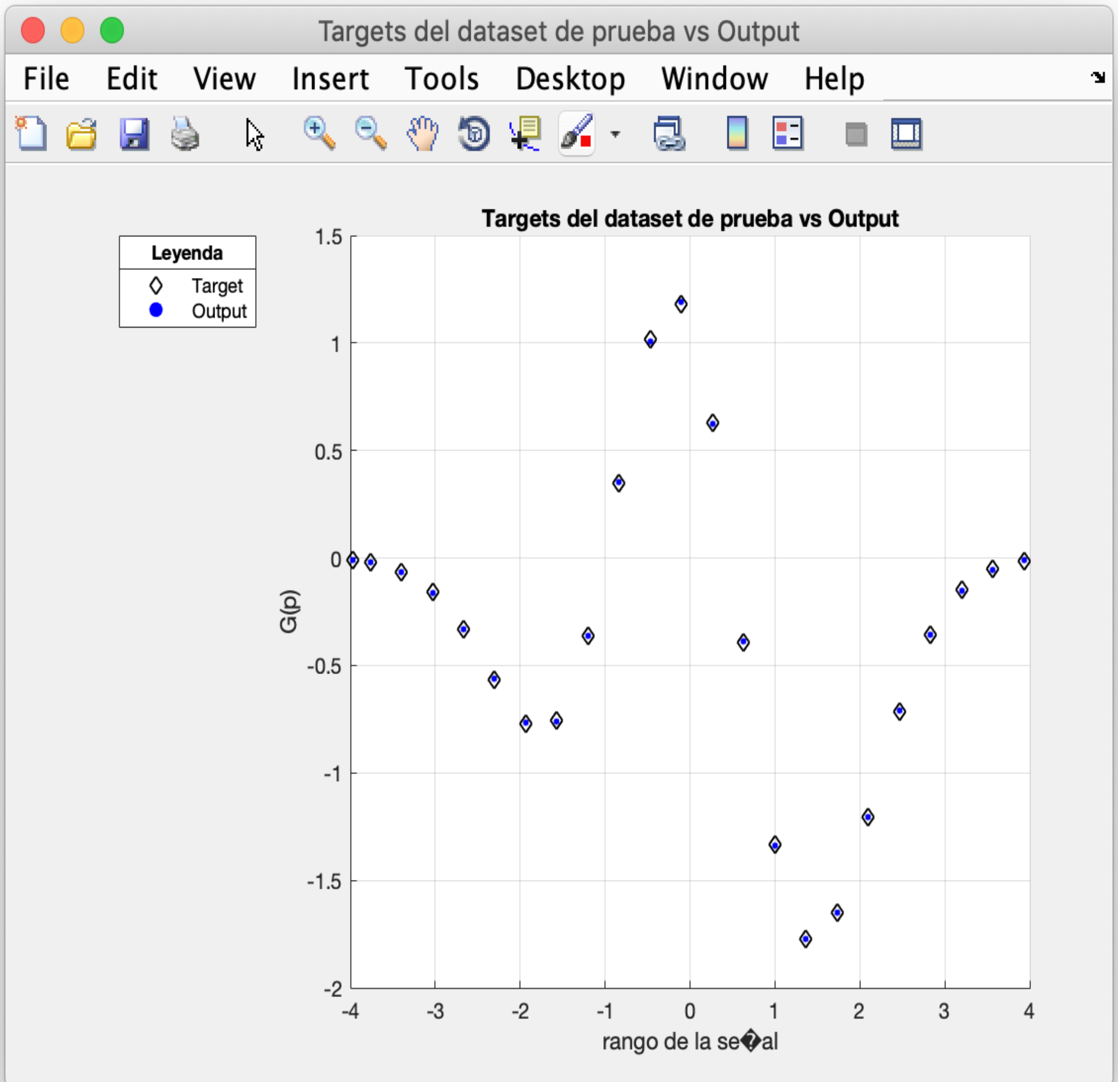


Figura 24: Gráfica 3.3

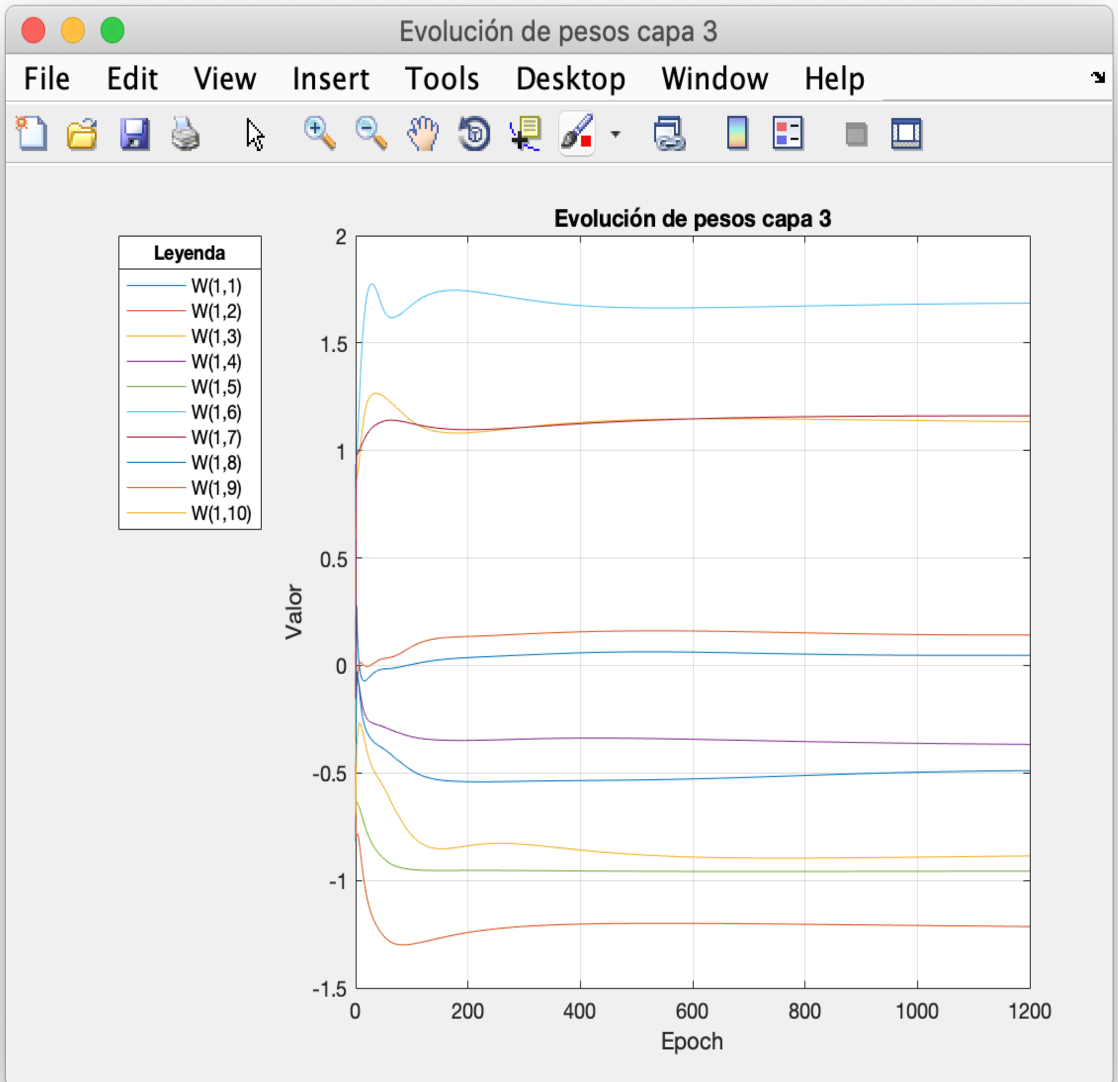


Figura 25: Gráfica 3.4

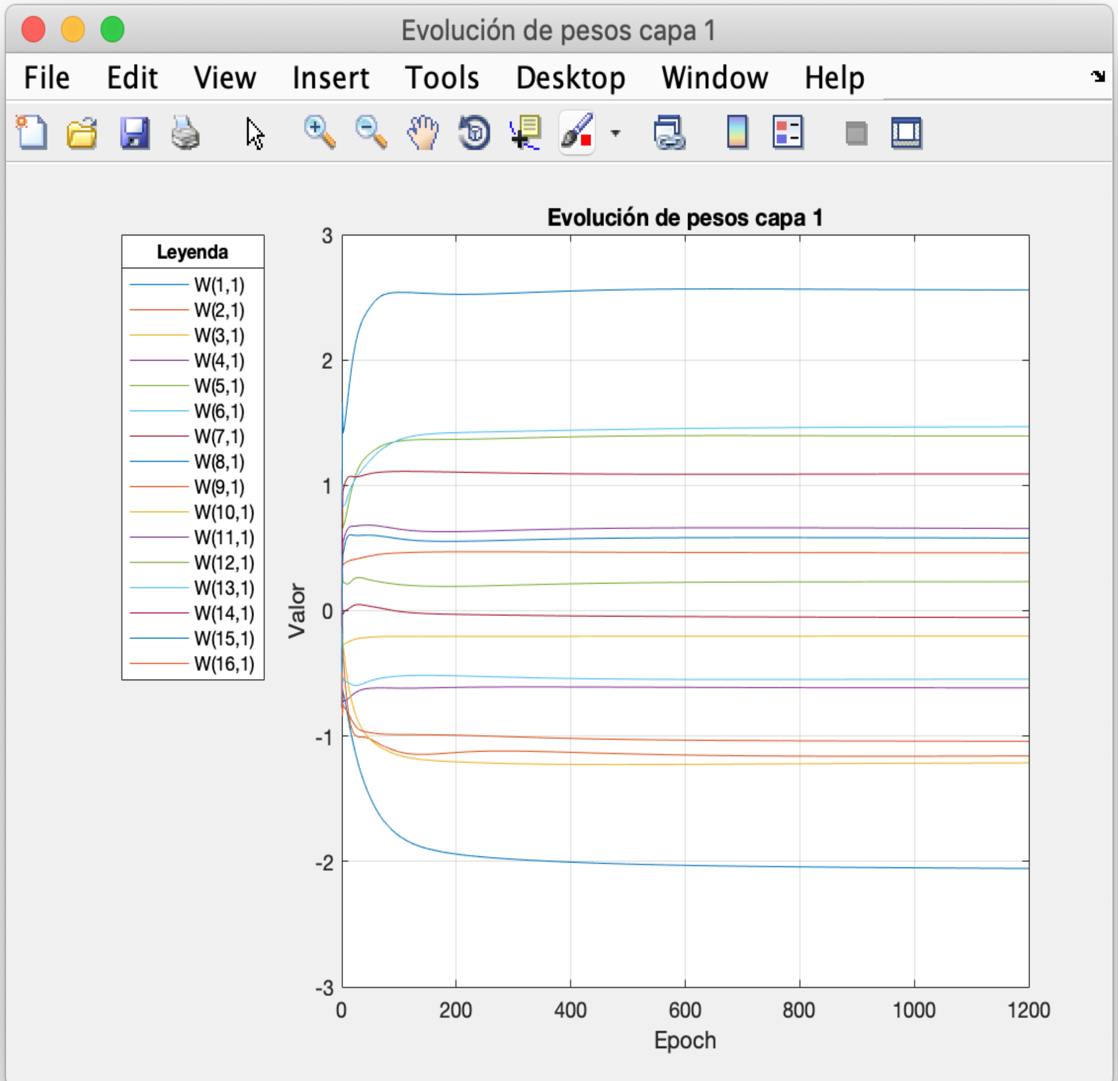


Figura 26: Gráfica 3.5

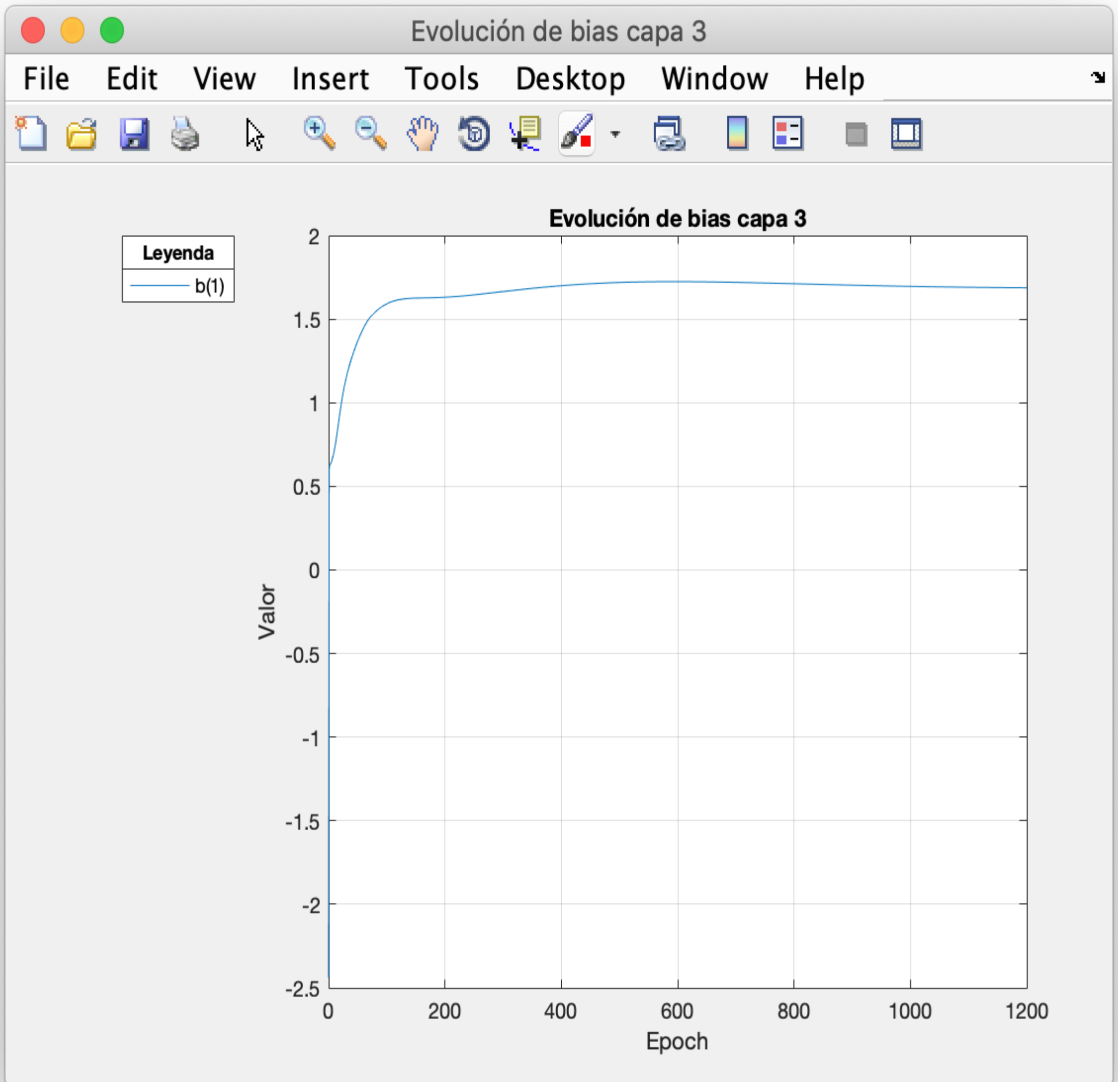


Figura 27: Gráfica 3.6

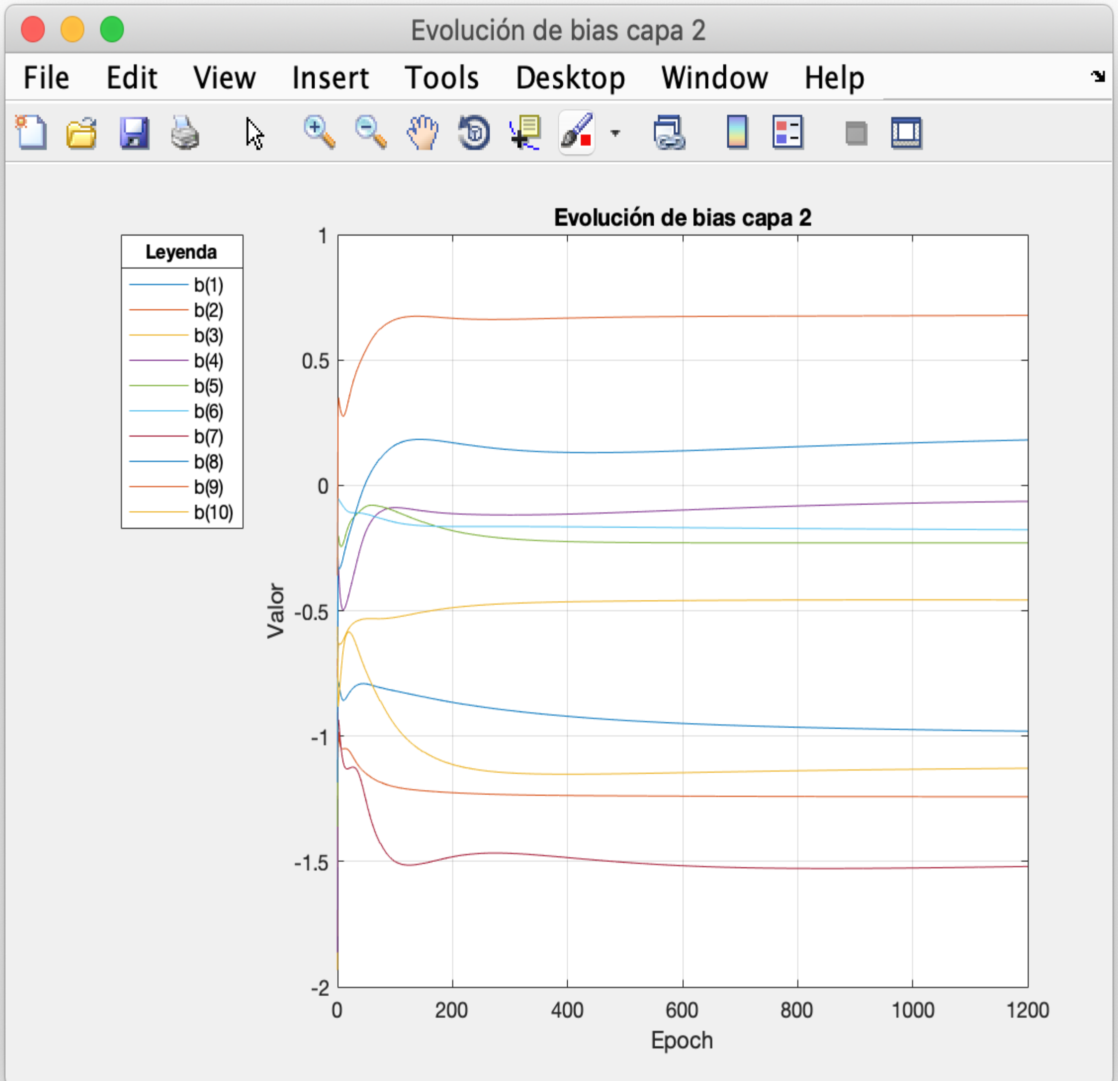


Figura 28: Gráfica 3.7

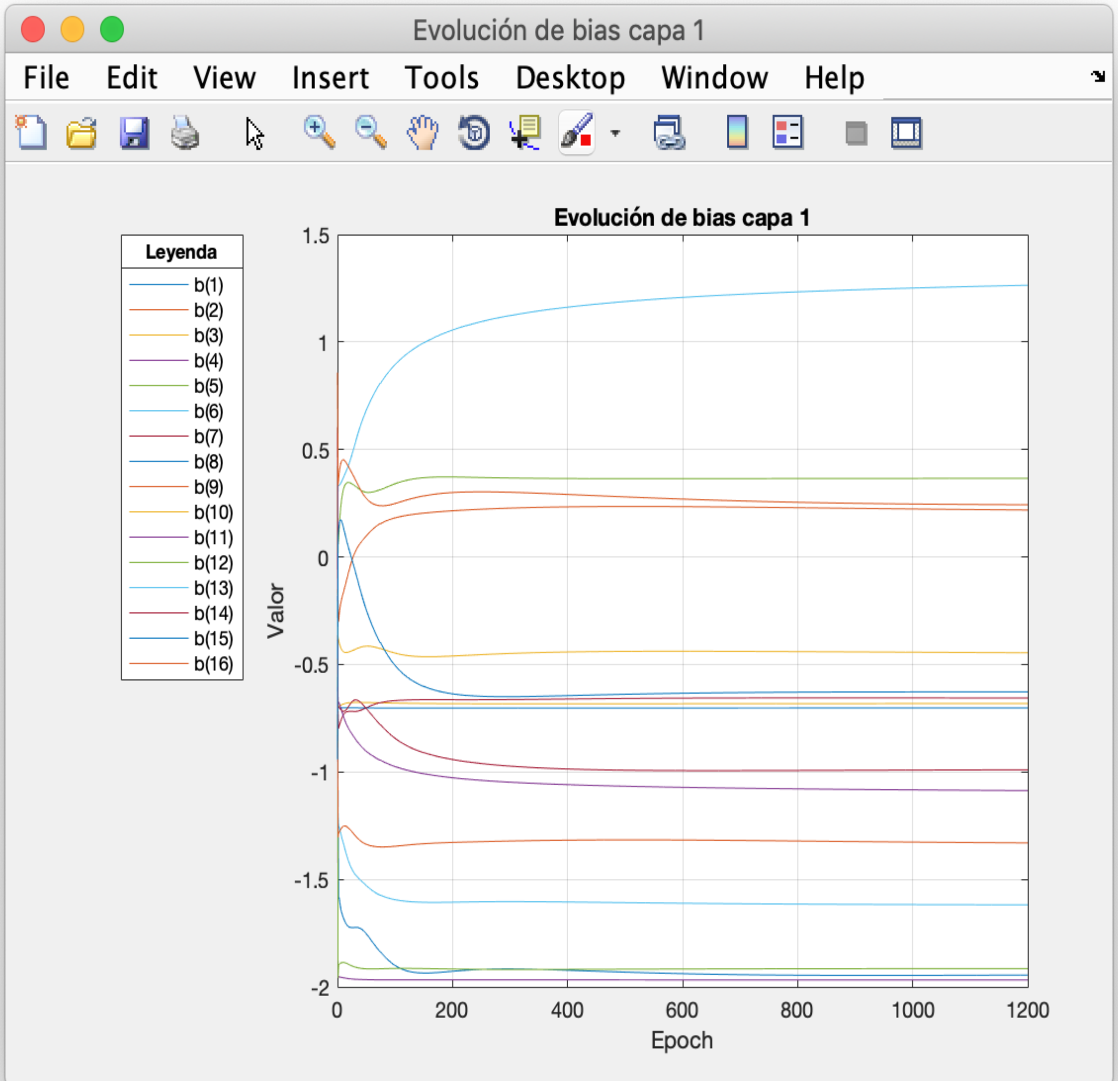


Figura 29: Gráfica 3.8

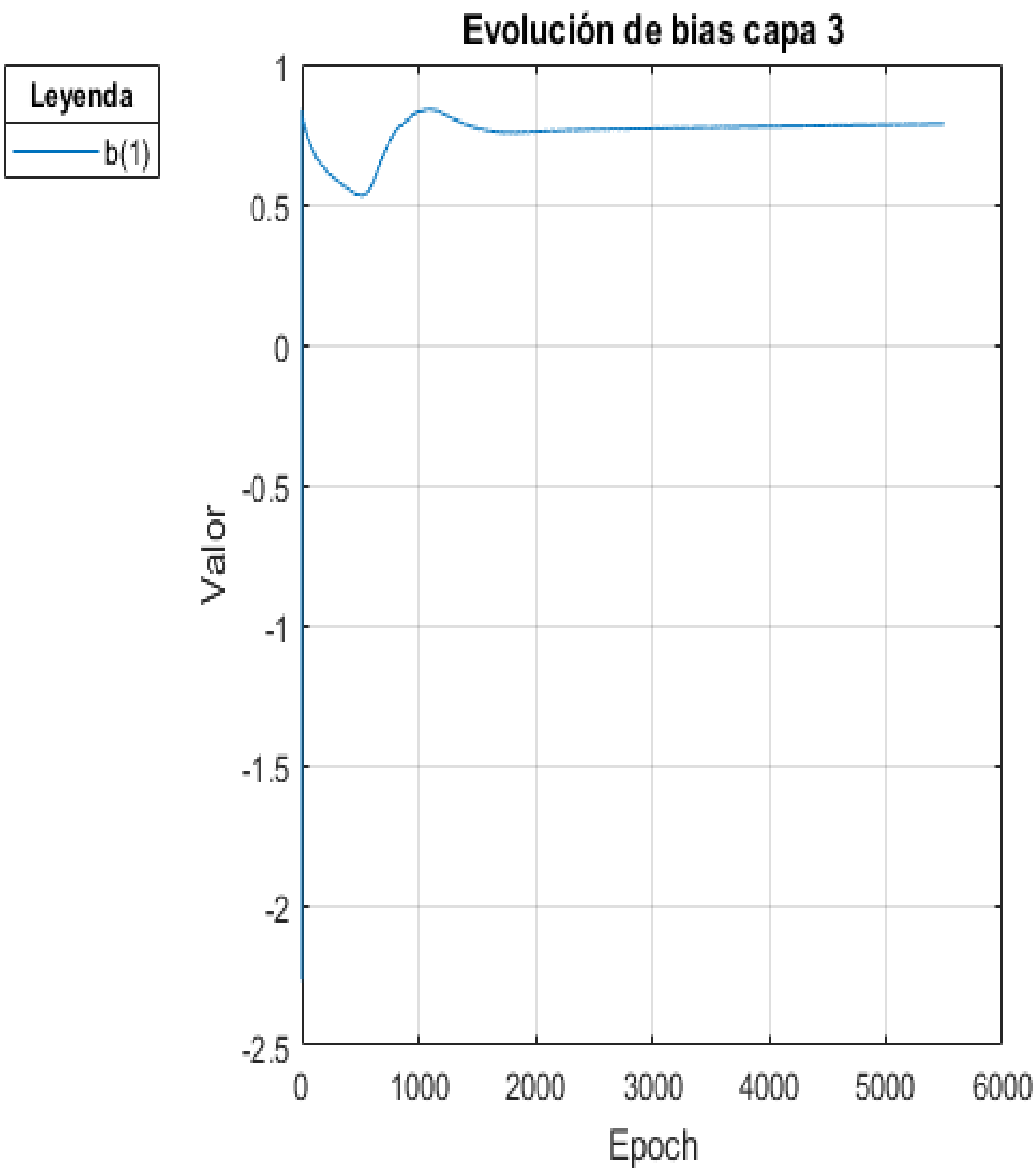


Figura 30: Gráfica 3.9

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 perceptrón es la unidad básico de las redes neuronales que se usan hoy en día, su creador, Frank Rosenblatt, hizp una muy importante aportación para el campo de las redes neuronales artificiales. La práctica estuvo mucho más corta de lo que esperaba, la regla de aprendizaje es "magia".

6. Referencias

Martin T Hagan. Machine Learning, Neural Network Design (2nd Edition), 2014.
`https://medium.com/@thomascountz/calculate-the-decision-boundary-of-a-single-perceptron-visualizing`

7. Apéndice

Listing 1: Código

```
clc
clear

% Read the inputs file
inputs_path = strcat(input('Ingrese el nombre del archivo de inputs sin la extensión: ','s'), '.txt');
%inputs_path = 'inputs.txt';
inputs = importdata(inputs_path);

% Read the targets
targets_file = strcat(input('Ingrese el nombre del archivo de targets sin la extensión: ','s'), '.txt');
%targets_path = 'targets.txt';
targets = importdata(targets_path);

data_size = size(inputs, 1);

% Enter MLP architecture
architecture = str2num(input('Ingrese el vector de la arquitectura: ','s'));
% Calculate layer parameters
%architecture = str2num('1 16 10 1');
num_layers = length(architecture) - 1;
```

```
R = architecture(1);
functions_vector = str2num(input('Ingrese el vector de las funciones de activación: 1) purelin()\n2) logsig()\n3) tansig()\n\n: ','s'));
%functions_vector = str2num('3 2 1');

% Enter the learning factor
alpha = input('Ingresa el valor del factor de aprendizaje(alpha): ');
%alpha = .01;

epochmax = input('Ingresa el número máximo de épocas: ');
% epochmax = 10000;
%validation_iter = 500;
%numval = 7;
%error_epoch_validation = .0000000000000001;
numval = input('Numero maximo de incrementos consecutivos del error de validacion (numval): ');
error_epoch_validation = input('Ingrese el valor minimo del error de epoca (error_epoch_validation): ');
validation_iter = input('Ingrese el múltiplo de épocas para realizar una época de validación  (validation_iter): ');

% Dataset Slicing
config_option = input('Elija una configuración de distribución de datasets: \n1: 80-10-10\n2: 70-15-15\n');
%config_option = 2;
[training_ds, test_ds, validation_ds] = dataset_slices(config_option, inputs, targets);
validation_ds_size = size(validation_ds, 1);
test_ds_size = size(test_ds, 1);
training_ds_size = size(training_ds, 1);

disp('Dataset de entrenamiento:');
disp(training_ds);
disp('Dataset de validacion:');
disp(validation_ds);
disp('Dataset de prueba:');
disp(test_ds);

% Open the files for weights and bias
total_weight_files = 0;
total_bias_files = 0;
for i=1:num_layers
% For neurons
for j=1:architecture(i + 1)
% For weights
for l=1:architecture(i)
total_weight_files = total_weight_files + 1;
end
end
```

```
total_bias_files = total_bias_files + 1;
end

W_files = zeros(total_weight_files, 1);
b_files = zeros(total_bias_files, 1);

current_file = 1;
for i=1:num_layers
path = strcat(pwd, '/historico/capa_', num2str(i), '/pesos/');
if ~exist(path, 'dir')
mkdir(path);
end
% For layers
for j=1:architecture(i + 1)
% For neurons
for k=1:architecture(i)
archivo_pesos = strcat(path, '/pesos', num2str(j), '_', num2str(k),'.txt');
W_files(current_file) = fopen(archivo_pesos,'w');
current_file = current_file +1;
end
end
end

current_file = 1;
for i=1:num_layers
path = strcat(pwd,'/historico/capa_', num2str(i), '/bias/');
if ~exist(path, 'dir')
mkdir(path);
end
for j=1:architecture(i+1)
archivo_bias = strcat(path,'/bias',num2str(j),'.txt');
b_files(current_file) = fopen(archivo_bias,'w');
current_file = current_file +1;
end
end

% Initialize MLP parameters and Print them

num_w_files = 1;
num_b_files = 1;
W = cell(num_layers,1);
b = cell(num_layers,1);
% Output of each layer
a = cell(num_layers + 1, 1);
% Sentitivities
S = cell(num_layers, 1);
% Derivatives of each layer
F_m = cell(num_layers, 1);

```

```
% For each layer
for i=1:num_layers
% Random value
W_r_value = 2 * rand(architecture(i + 1), architecture(i)) - 1;
b_r_value = 2* rand(architecture(i + 1), 1) - i;
W{i} = W_r_value
b{i} = b_r_value
% For each neuron
for j=1:architecture(i + 1)
%For each weight
for k=1:architecture(i)
% Print wights value
fprintf(W_files(num_w_files), '%f\r\n', W_r_value(j, k));
num_w_files = num_w_files + 1;
end
end
% For each neuron
for j=1:architecture(i + 1)
% print bias value
fprintf(b_files(num_b_files), '%f\r\n', b_r_value(j));
num_b_files = num_b_files + 1;
end
end

% Learning algorithm
num_validation_epoch = 0;
early_stopping_increment = 0;
validation_error = 0;
learning_error = 0;
early_s_counter = 0;

% initialize vectors for printing errors
learning_err_values = zeros(epochmax, 1);
evaluation_err_values = zeros(ceil(epochmax / validation_iter), 1);
for epoch=1:epochmax
l_error = 0;
% Reset the values
num_w_files = 1;
num_b_files = 1;
% if isn't a validation epoch
if(mod(epoch ,validation_iter) ~= 0)
for t_data=1:training_ds_size
% initial condition
a{1} = training_ds(t_data, 1);
% Foward propagation
for t_p=1:num_layers
W_aux = cell2mat(W(t_p));
b_aux = cell2mat(b(t_p));
a_aux = cell2mat(a(t_p));
```

```
n_f = W_aux * a_aux + b_aux;
a{t_p + 1} = get_activation_function(n_f, functions_vector(t_p));
end
a_aux = cell2mat(a(num_layers + 1));
t_error = training_ds(t_data, 2) - a_aux;
l_error = l_error + (t_error / data_size);
% Sensitivities calculation
F_m{num_layers} = get_F_matrix(functions_vector(num_layers), architecture(num_layers +
    1), a_aux);
F_m_temp = cell2mat(F_m(num_layers));
S{num_layers} = F_m_temp * (t_error)*(-2);
% Backpropagation
for m = num_layers-1:-1:1
W_aux = cell2mat(W(m+1));
s_aux = cell2mat(S(m+1));
a_aux = cell2mat(a(m+1));
F_m{m} = get_F_matrix(functions_vector(m),architecture(m+1),a_aux);
F_m_temp = cell2mat(F_m(m));
S{m} = F_m_temp * (W_aux')*s_aux;
end
% Learning Rules
for k = num_layers:-1:1
W_aux = cell2mat(W(k));
b_aux = cell2mat(b(k));
s_aux = cell2mat(S(k));
a_aux = cell2mat(a(k));
W{k} = W_aux - (alpha * s_aux * a_aux');
b{k} = b_aux - (alpha * s_aux);
W_aux = cell2mat(W(k));
b_aux = cell2mat(b(k));
end
end
learning_error = l_error;
learning_err_values(epoch) = l_error;
% This epoch is a validation one
else
val_error = 0;
num_validation_epoch = num_validation_epoch + 1;
for t_data = 1:validation_ds_size
% Initial Condition
a{1} = validation_ds(t_data, 1);
% Foward propagation
for k=1:num_layers
W_aux = cell2mat(W(k));
a_aux = cell2mat(a(k));
b_aux = cell2mat(b(k));
n_f = W_aux * a_aux + b_aux;
a{k + 1} = get_activation_function(n_f, functions_vector(k));
end
```

```
a_aux = cell2mat(a(num_layers+1));
val_error = validation_ds(t_data,2)-a_aux;
val_error = val_error+(val_error/validation_ds_size);
end
evaluation_err_values(epoch) = val_error;
if early_stopping_increment == 0
validation_error = val_error;
early_stopping_increment = early_stopping_increment+1;
fprintf('Incremento actual para early stopping = %d\n', early_stopping_increment);
else
if val_error > validation_error
validation_error = val_error;
early_stopping_increment = early_stopping_increment+1;
fprintf('Incremento actual para early stopping = %d\n', early_stopping_increment);
if early_stopping_increment == numval
% Reset the counter
early_s_counter = 1;
fprintf('Early stopping en la época:  %d\n', epoch);
break;
end
else
validation_error = 0;
early_stopping_increment = 0;
fprintf('Incremento actual para early stopping = %d\n', early_stopping_increment);
end
end
end

% Print the values on console
num_w_files = 1;
num_b_files = 1;
for k = num_layers:-1:1
W_aux = cell2mat(W(k));
b_aux = cell2mat(b(k));
for j=1:architecture(k+1)
for l=1:architecture(k)
fprintf(W_files(num_w_files), '%f\r\n', W_aux(j,l));
num_w_files = num_w_files +1;
end
end
for j=1:architecture(k + 1)
fprintf(b_files(num_b_files), '%f\r\n', b_aux(j));
num_b_files = num_b_files + 1;
end
end

% Check stopping calculations
if mod(epoch,validation_iter) ~= 0 && l_error <= error_epoch_validation && l_error >=
    0
```

```
learning_error = l_error;
fprintf('Aprendizaje exitoso en la época %d\n', epoch);
break;
end
end

if epoch == epochmax
disp('Se llego a epochmax');
end

% Print the las final values
if early_s_counter == 1
num_w_files = 1;
num_b_files = 1;
for k = num_layers:-1:1
W_aux = cell2mat(W(k));
b_aux = cell2mat(b(k));
for j = 1:architecture(k + 1)
for l=1:architecture(k)
fprintf(W_files(num_w_files), '%f\r\n', W_aux(j, l));
num_w_files = num_w_files + 1;
end
end
for j=1:architecture(k + 1)
fprintf(b_files(num_b_files), '%f\r\n', b_aux(j));
num_b_files = num_b_files + 1;
end
end
end

% Close all files
for i=1:total_weight_files
fclose(W_files(i));
end
for i=1:total_bias_files
fclose(b_files(i));
end

% Propagate the test dataset
test_error = 0;
output = zeros(test_ds_size,1);
for i=1:test_ds_size
% Initial condition
a{1} = test_ds(i,1);
for k=1:num_layers
W_aux = cell2mat(W(k));
a_aux = cell2mat(a(k));
b_aux = cell2mat(b(k));
n_f = W_aux*a_aux+b_aux;
```

```
a{k+1} = get_activation_function(n_f, functions_vector(k));
end
test_data = cell2mat(a(1));
a_aux = cell2mat(a(num_layers + 1));
test_error = test_error + (1 / test_ds_size) * (test_ds(i,2) - a_aux);
output(i) = a_aux;
end

% Print last errors
fprintf('Error de aprendizaje = %f\n', learning_error);
fprintf('Error de validación = %f\n', validation_error);
fprintf('Error de prueba = %f\n', test_error);


% Output vs test
scatter_output_vs_test(test_ds, output);
% Propagate the training size for ploting

output = zeros(training_ds_size,1);
for i=1:training_ds_size
% Initial Condition
a{1} = training_ds(i, 1);
for k=1:num_layers
W_aux = cell2mat(W(k));
a_aux = cell2mat(a(k));
b_aux = cell2mat(b(k));
a{k+1} = get_activation_function(W_aux*a_aux+b_aux,functions_vector(k));
end
a_aux = cell2mat(a(num_layers + 1));
test_error = test_error + (1 / training_ds_size) * (training_ds(i,2) - a_aux);
output(i) = a_aux;
end

scatter_output_vs_training(training_ds, output);

% Plot the error evolution
error_plot(validation_iter, num_validation_epoch, learning_err_values, epoch,
    evaluation_err_values);
% Plot weight evolution
weight_evolution_plot(architecture, num_layers, epoch);
% Plot bias evolution
bias_evolution_plot(architecture, num_layers, epoch);

% Write final values
for i=1:num_layers
path = strcat(pwd, '/Valores_finales/capa_', num2str(i), '/');
if ~exist(path, 'dir')
mkdir(path);
end
```

```
W_aux = cell2mat(W(i));
res_pesos = strcat(path, '/pesos.txt');
dlmwrite(res_pesos, W_aux, ';');
end

for i=1:num_layers
path = strcat(pwd,'/Valores_finales/capa_', num2str(i), '/');
if ~exist(path, 'dir')
mkdir(path);
end
b_aux = cell2mat(b(i));
res_bias = strcat(path, '/bias.txt');
dlmwrite(res_bias, b_aux, ';');
end
```