|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

Отчёт

по лабораторной работе № 2

по дисциплине «Интеллектуальные технологии информационной безопасности»

**Тема: «Применение однослойной нейронной сети с линейной функцией активации»**

Выполнил: Овсепян А. Н.,

студент группы ИУ8-63

Проверил: Волосова Н.К,

аспирант каф. ИУ8

г. Москва,

2021 г.

1. Цель работы

Изучить возможности однослойный НС в задачах прогнозирования временных рядов методом скользящего окна (авторегрессия).

1. Постановка задачи

На временном интервале [a, b] задан дискретный набор значений функции: *cos(x).* Количество точек N = 20, расположение – равномерное. Методом “скользящего окна” спрогнозировать поведение функции на N точках последующего интервала (b, 2b – a). Для решение использовать однослойную нейронную сеть с количеством нейронов p=4 и линейной функцией активации. Обучение проводить методом Видроу-Хоффа. Исследовать количество эпох обучения и коэффициента обучения на среднеквадратичной погрешность приближения. Исследовать процесс прогнозирование при постепенным изменении (уменьшение /увеличение) размера окна. Сделать выводы по результатам численного эксперимента.

1. Ход работы
2. Сначала были сформированы наборы иксов на интервале [a, b], [a, 2b – a] и вычислены значения функции на заданном интервале.
3. При старте обучения нейронной сети (НС) были сгенерированы наборы весов в количестве **длина окна + 1** и выбрана норма обучения равная nj = 0.8.
4. В процессе самого обучения мы начинали с элемента с индексом длины окна (необходимо для начала обучения).
5. Сравнивая полученное значение с реальным, получается ошибка прогноза.
6. Далее проводится корреляция весов по формуле:



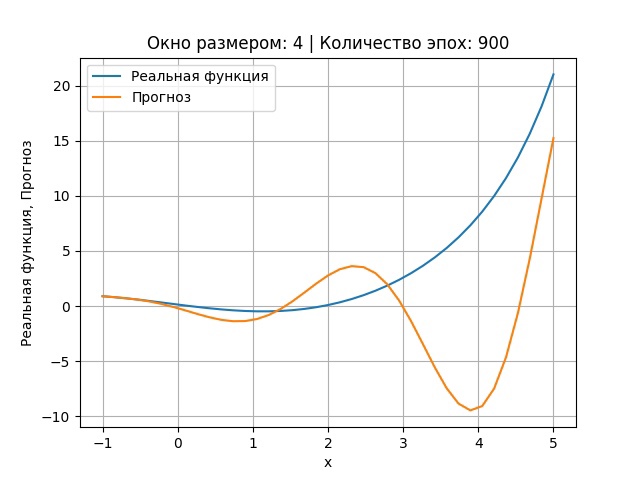
, где:

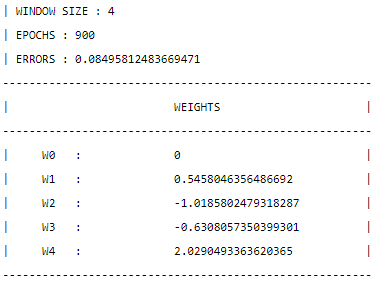


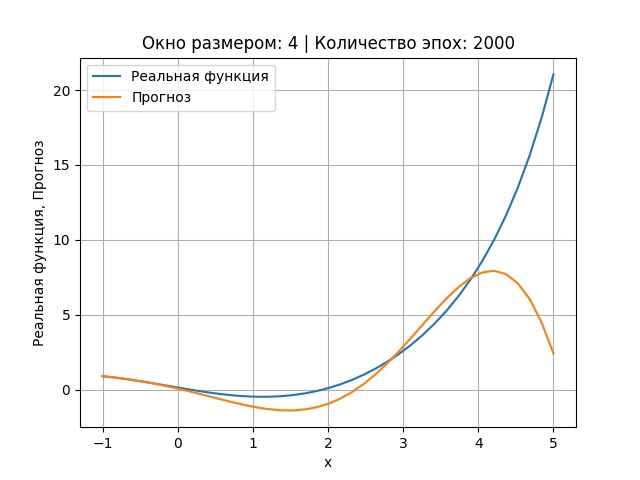


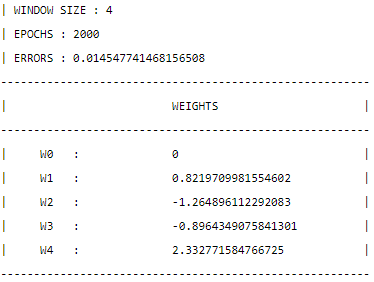
xk – значение функции на k-ом шаге.

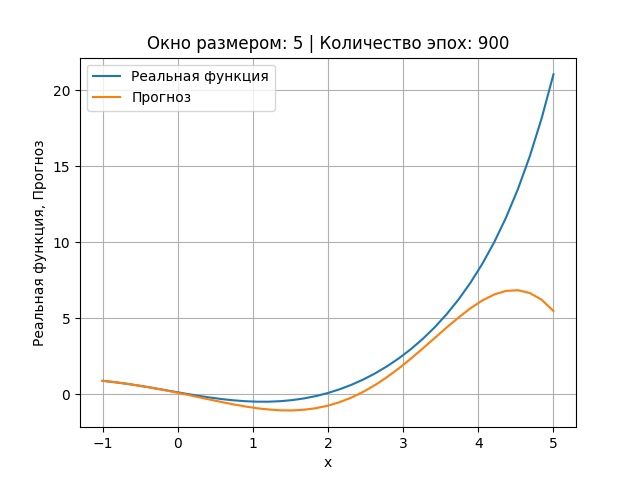
1. Корреляция происходит до достижения минимально допустимой погрешности или прохождения заданного количества эпох.
2. Полученные результаты

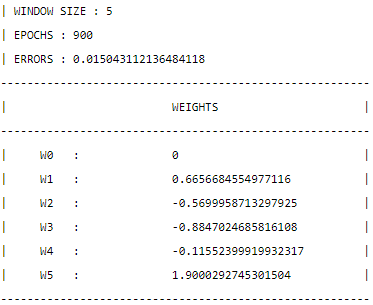


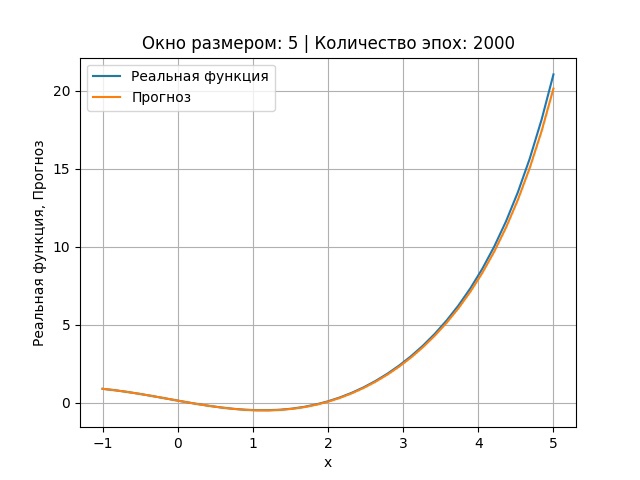


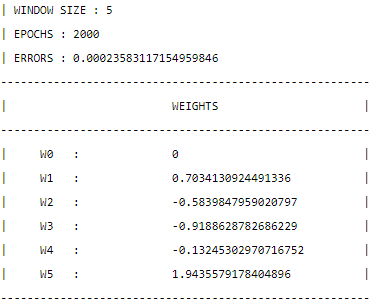


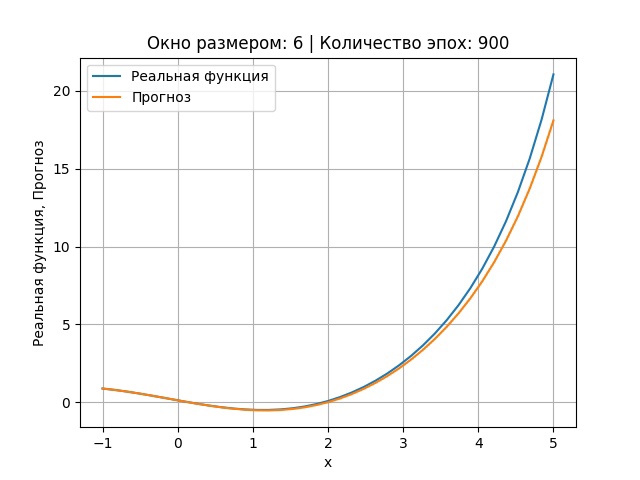


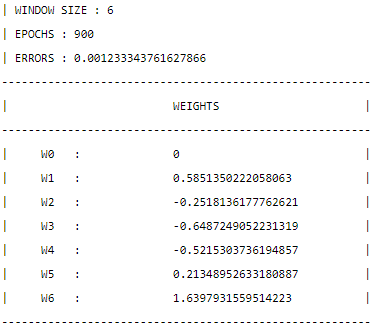


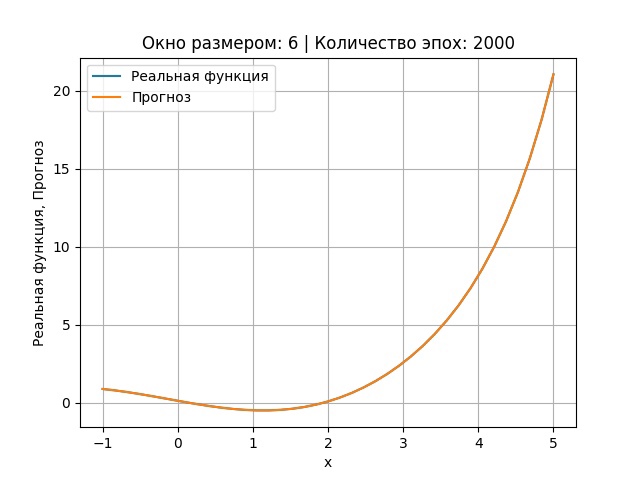


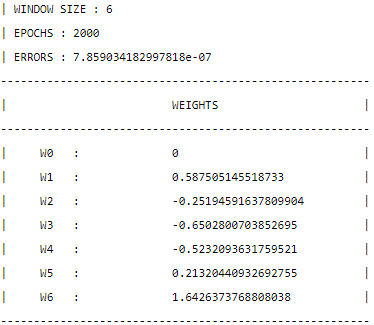












1. Выводы

Исходя из опытов обучения нейронных сетей в этой лабораторной работе можно сделать выводы, что чем больше количество эпох обучения и размер окна – тем точнее прогноз.

**Приложение 1 Исходный код программы.**

1. **import** itertools
2. **from** math **import** exp
3. **import** matplotlib.pyplot as plt

6. **def** fun(x):
7. **return** int((((x[0] **+** x[1] **+** x[2]) **%** 2) **\*** ((x[1] **+** x[2] **+** x[3]) **%** 2)))

10. **def** f4(net):
11. **if** 0.5 **\*** (net **/** (1 **+** abs(net)) **+** 1):
12. **return** 1
13. **else**:
14. **return** 0

17. **def** f\_net(x, v, v0):
18. net **=** v0
19. **for** (i, j) **in** zip(x, v):
20. net **+=** i**\***j
21. **return** net

24. **def** print\_table(first, name):
25. print('----------------')
26. print('', name, ': ')
27. **for** (i, j) **in** zip(first, range(len(first))):
28. print(name[**-**1], ':', i,)
29. print('----------------')

32. **def** y\_net(net):
33. **return** 1 **if** net >**=** 0 **else** 0

36. **def** start(nj, c, x, X, f):
37. l **=** 0
38. v **=** [0] **\*** (len(c) **+** 1)
39. error **=** 1
40. massive\_errors **=** []
41. massive\_epochs **=** []
42. massive\_v **=** []
43. **while** error > 0:
44. error **=** 0
45. massive\_v.append(v.copy())
46. **for** \_ **in** x:
47. qf **=** []
48. **for** k **in** range(len(c)):
49. s **=** 0
50. **for** (i, j) **in** zip(\_, c[k]):
51. s **-=** (i **-** j) **\*\*** 2
52. qf.append(exp(s))
54. y **=** f(f\_net(qf, v[1:], v[0]))
55. b **=** fun(\_) **-** y
56. **for** i **in** range(len(v)):
57. **if** i **==** 0:
58. v[i] **+=** b **\*** nj
59. **else**:
60. v[i] **+=** b **\*** nj **\*** qf[i **-** 1]
62. massive\_epochs.append(l)
64. **for** \_ **in** X:
65. qf **=** []
66. **for** k **in** range(len(c)):
67. s **=** 0
68. **for** (i, j) **in** zip(\_, c[k]):
69. s **-=** (i **-** j) **\*\*** 2
70. qf.append(exp(s))
72. y **=** f(f\_net(qf, v[1:], v[0]))
73. b **=** fun(\_) **-** y
74. error **+=** abs(b)
76. massive\_errors.append(error)
77. **if** error **==** 0:
78. massive\_v.append(v)
79. l **+=** 1
80. **if** l > 100:
81. **return** False
83. answer **=** [massive\_v, massive\_epochs, massive\_errors, c, x]
84. **return** answer

87. **class** Neural:
88. **def** \_\_init\_\_(self, nj, X, c, f):
89. answer **=** []
90. '''for i in range(0, 15, 1):
91. for j in itertools.combinations(X, 16 - i):
92. mass\_0 = []
93. mass\_1 = []
94. for k in j:
95. if fun(k):
96. mass\_1.append(k)
97. else: mass\_0.append(k)
98. c = mass\_0 if len(mass\_1) > len(mass\_0) != 0 else mass\_1
99. get = start(nj, c, j, X, f)
100. if get:
101. answer.append(get)
102. break'''
103. x **=** ([0, 0, 0, 0], [0, 0, 1, 0], [0, 1, 0, 0],
104. [1, 0, 0, 1], [1, 0, 1, 1], [1, 1, 1, 1])
105. get **=** start(nj, c, x, X, f)
106. **if** get:
107. answer.append(get)
108. print\_table(answer[**-**1][4], 'X')
109. print\_table(answer[**-**1][0][**-**1], 'Весовые коэффициенты  V')
110. print\_table(answer[**-**1][3], 'C')
112. print('--------------------------------------------------------------')
113. print('|', '%-7s' **%** 'epochs', '|', ' V', ' ' **\*** 33, '|', '%-7s' **%** 'errors', '|')
114. print('--------------------------------------------------------------')
115. **for** (i, j, k) **in** zip(answer[**-**1][0], answer[**-**1][1], answer[**-**1][2]):
116. print('|', '%-7s' **%** j, end**=**'')
117. **for** t **in** i:
118. print('|', '%-7s' **%** round(t, 4), end**=**' ')
119. print('|', '%-7s' **%** k, '|')
120. print('--------------------------------------------------------------')
122. \_, ax **=** plt.subplots()
123. ax.plot(answer[**-**1][1], answer[**-**1][2], label**=**'суммарная ошибка')
124. ax.legend()
125. plt.title(f'График суммарной ошибки НС по эпохам обучения')  # заголовок
126. plt.xlabel("Эпохи")  # ось абсцисс
127. plt.ylabel("Ошибки")  # ось ординат
128. plt.grid()  # включение отображение сетки
130. plt.show()

133. **if** \_\_name\_\_ **==** '\_\_main\_\_':
134. X **=** [
135. [0, 0, 0, 0], [0, 0, 0, 1], [0, 0, 1, 0], [0, 0, 1, 1],
136. [0, 1, 0, 0], [0, 1, 0, 1], [0, 1, 1, 0], [0, 1, 1, 1],
137. [1, 0, 0, 0], [1, 0, 0, 1], [1, 0, 1, 0], [1, 0, 1, 1],
138. [1, 1, 0, 0], [1, 1, 0, 1], [1, 1, 1, 0], [1, 1, 1, 1]
139. ]
140. f0 **=** []
141. f1 **=** []
142. fs **=** []
143. **for** k **in** X:
144. fs.append(fun(k))
145. **if** fun(k) **==** 1:
146. f1.append(k)
147. **else**:
148. f0.append(k)
150. c **=** []
151. **if** len(f1) > len(f0) !**=** 0:
152. c **=** f0
153. **else**:
154. c **=** f1
156. n1 **=** Neural(0.3, X, c, y\_net)