|  |  |
| --- | --- |
| 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 № 3

по дисциплине «Интеллектуальные технологии информационной безопасности»

**Тема: «Исследование нейронных сетей с радиальными базисными функциями (RBF) на примере моделирования булевых выражений»**

Выполнил: Овсепян А. Н.,

студент группы ИУ8-63

Проверил: Волосова Н.К,

аспирант каф. ИУ8

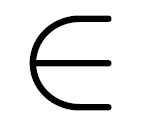
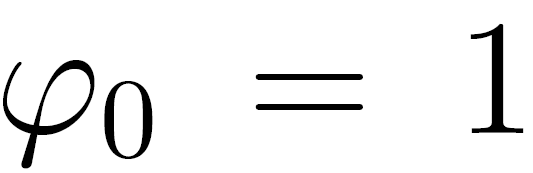
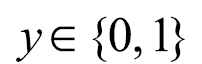
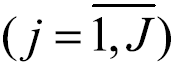
г. Москва,

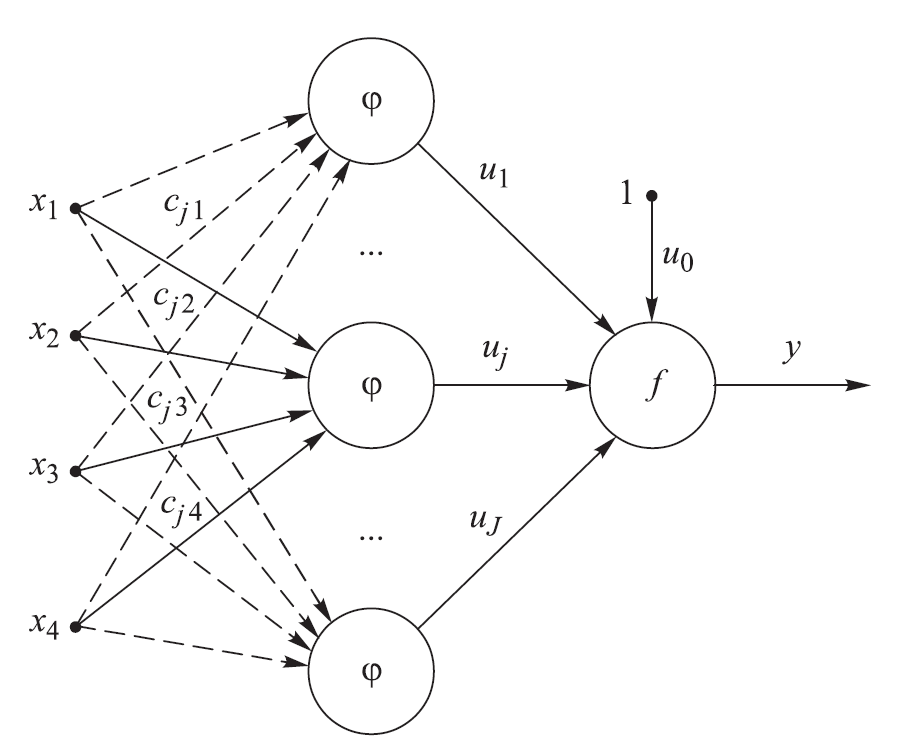
2021 г.

1. Цель работы

Исследовать функционирование НС с радиальными базисными функциями (RBF) и обучить ее по правилу Видроу — Хоффа.

1. Постановка задачи

Получить модель булевой функции (БФ) на основе RBF-НС с двоичными входами x1, x2, x3, x4  {0,1}, единичным входом смещения ; синаптическими весами , двоичным выходом с пороговой ФА выходного нейрона, *J* скрытыми RBF-нейронами с гауссовой ФА  и координатами центров , .



**Рисунок 1. Нейронная сеть RBF.**

Для заданной БФ количество RBF-нейронов необходимо выбирать из соотношения , где *J1, J0* – количество векторов , соответствующих значениям БФ «0» и «1». Центры RBF должны совпадать с концами этих векторов.

Требуется найти минимальный набор векторов **х**, используемых для обучения.

1. Ход работы

Таблица истинности для Fun(x1,x2,x3,x4) = .

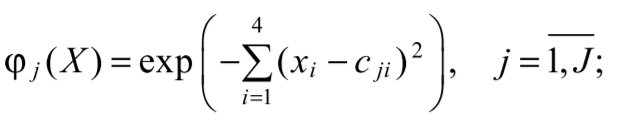
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X1 | X2 | X3 | X4 | Fun(x1,x2,x3,x4) |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Алгоритм работы программы:

1. Получаем наборы, на которых будем пытаться обучить нейронную сеть (НС) и подаем данные для старта обучения. По завершении работы программы будет получен наименьший набор векторов, необходимый для обучения НС.
2. Полученные наборы разделим на два – в одном вектора дающие “0”, в другом – “1”. Размер наименьшего набора является количеством RBF нейронов, а центры RBF нейронов равны векторам этого набора.
3. Количество синаптических весов = количество RBF нейронов + 1 (на старте все равны 0).

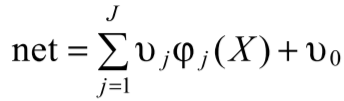
Норма обучения = 0.3

1. Пока суммарная ошибка error > 0 происходит обучение.
2. Наборы из (0) сравниваем с центрами RBF нейронов по формуле:

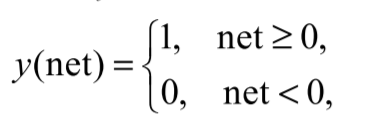


Получившиеся значения записываем в вектор **qf.**

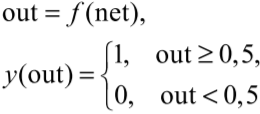
1. Передаем в функцию **f\_net** значения весов ***v* и** вектора **qf.**



1. Считаем значения пороговой ФА:



Либо для логистической ФА:

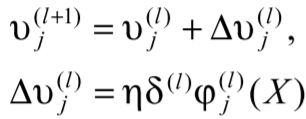


***Обучение НС.***

1. Ошибку считаем, сравнивая реальное значение с получившемся в (3.3),

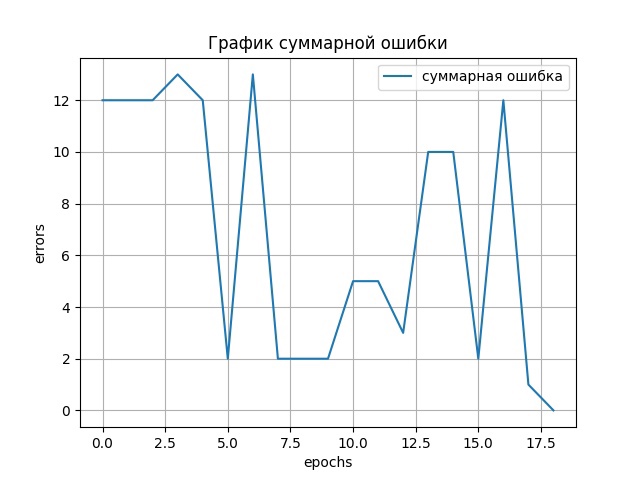


если она есть будет происходить корреляция весов, по формулам:

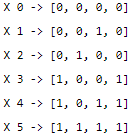


1. Суммарная ошибка считается на всех  наборах и будет равна количеству отличающихся значений.
2. Полученные результаты

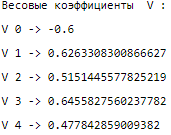
График суммарной квадратичной ошибки:



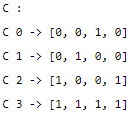
Наборы, на которых происходит обучение:



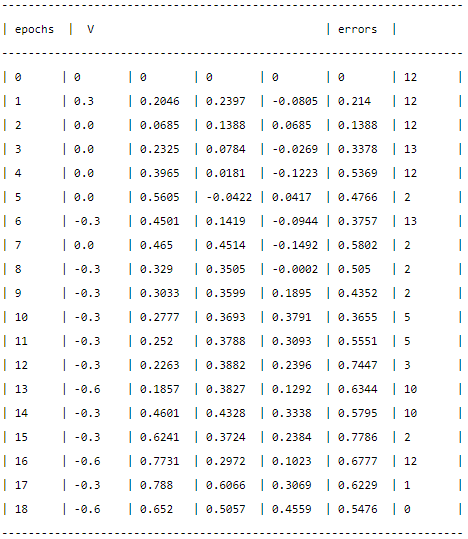
Синаптические веса:



Центры RBF нейронов



Численные выводы:



1. Выводы

В ходе данной лабораторной работы был изучен метод обучения НС с радиальными базисными функциями (RBF), была обучена НС, прогнозирующая поведение булевой функции.

**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('\n|||||||||||||||||||||||||||\n')
26. print(name, ': ')
27. **for** (i, j) **in** zip(first, range(len(first))):
28. print(name[**-**1], j, '->', i,)

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

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

86. **class** Neural:
87. **def** \_\_init\_\_(self, nj, X, c, f):
88. answer **=** []
89. x **=** ([0, 0, 0, 0], [0, 0, 1, 0], [0, 1, 0, 0],
90. [1, 0, 0, 1], [1, 0, 1, 1], [1, 1, 1, 1])
91. get **=** start(nj, c, x, X, f)
92. **if** get:
93. answer.append(get)
94. print\_table(answer[**-**1][4], 'X')
95. print\_table(answer[**-**1][0][**-**1], 'Весовые коэффициенты  V')
96. print\_table(answer[**-**1][3], 'C')
98. print('----------------------------------------------------------------------')
99. print('|', '%-7s' **%** 'epochs', '|', ' V', ' ' **\*** 33, '|', '%-7s' **%** 'errors', '|')
100. print('----------------------------------------------------------------------')
101. **for** (i, j, k) **in** zip(answer[**-**1][0], answer[**-**1][1], answer[**-**1][2]):
102. print('|', '%-7s' **%** j, end**=**'')
103. **for** t **in** i:
104. print('|', '%-7s' **%** round(t, 4), end**=**' ')
105. print('|', '%-7s' **%** k, '|')
106. print('----------------------------------------------------------------------')
108. \_, ax **=** plt.subplots()
109. ax.plot(answer[**-**1][1], answer[**-**1][2], label**=**'суммарная ошибка')
110. ax.legend()
111. plt.title(f'График суммарной ошибки НС по эпохам обучения')  # заголовок
112. plt.xlabel("Эпохи")  # ось абсцисс
113. plt.ylabel("Ошибки")  # ось ординат
114. plt.grid()  # включение отображение сетки
116. plt.show()

119. **if** \_\_name\_\_ **==** '\_\_main\_\_':
120. X **=** [
121. [0, 0, 0, 0], [0, 0, 0, 1], [0, 0, 1, 0], [0, 0, 1, 1], [0, 1, 0, 0], [0, 1, 0, 1], [0, 1, 1, 0], [0, 1, 1, 1],
122. [1, 0, 0, 0], [1, 0, 0, 1], [1, 0, 1, 0], [1, 0, 1, 1], [1, 1, 0, 0], [1, 1, 0, 1], [1, 1, 1, 0], [1, 1, 1, 1]
123. ]
124. f0 **=** []
125. f1 **=** []
126. fs **=** []
127. **for** k **in** X:
128. fs.append(fun(k))
129. **if** fun(k) **==** 1:
130. f1.append(k)
131. **else**:
132. f0.append(k)
134. c **=** []
135. **if** len(f1) > len(f0) !**=** 0:
136. c **=** f0
137. **else**:
138. c **=** f1
140. n1 **=** Neural(0.3, X, c, y\_net)