**Министерство образования и науки 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)

**ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ**

**ЛАБОРАТОРНАЯ РАБОТА № 4**

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

**Вариант № 7**

**Преподаватель:** Коннова Н.С.

**Студент:** Кустов И. А.

**Группа:** ИУ8-61

Оглавление

[Цель работы 3](#_Toc5301072)

[Постановка задачи 3](#_Toc5301073)

[Практическая часть 4](#_Toc5301074)

[Вывод 4](#_Toc5301075)

[Приложение 7](#_Toc5301076)

**Цель работы** – исследовать функционирование НС с радиальными базисными функциями (RBF) и обучить ее по правилу Видроу-Хоффа.

**Постановка задачи** – получить модель функции (БФ) на основе RBF-HC с двоичными входами , единичным входом смещения , синаптическими весами двоичным выходом с пороговой ФА выходного нейрона, J скрытыми слоями RBF – нейронами с гауссовой ФА и координатами центров (j = 1…J).

# **Практическая часть**

Получим нейронную модель БФ(таблица 1):

F(*x1, x2, x3, x4) = +*

Таблица 1

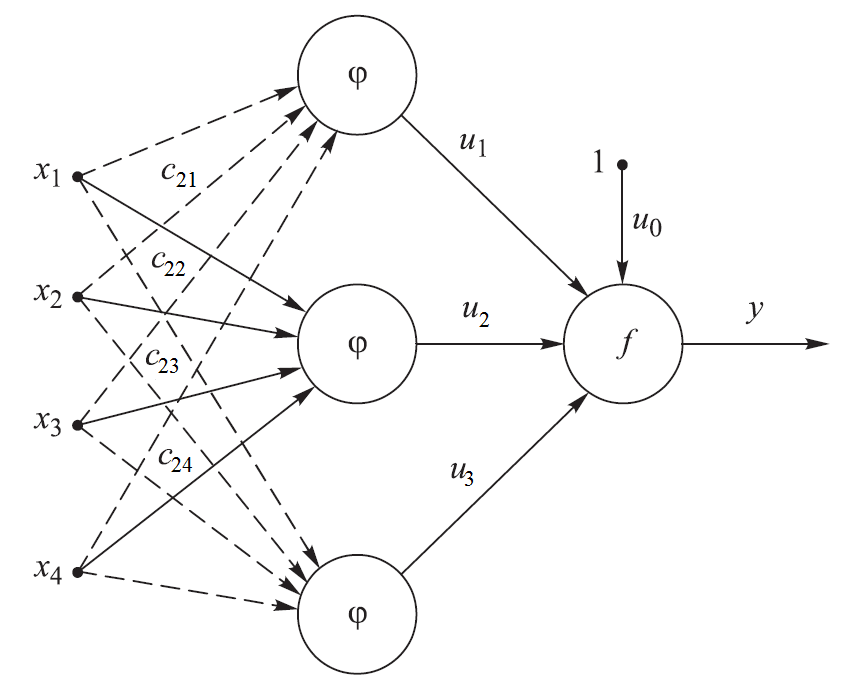
**Таблица истинности БФ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | F |
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 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 | 0 |
| 1 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

Обучение будет проведено с нормой обучения 0.3 и нулевыми начальными весами.

Для данной функции берем 3 RBF-нейрона с следующими центрами:

Схема сети представлена на рисунке 1.



**Рисунок 1.** Схема сети

При обучении на всех векторах НС обучилась за 2 эпохи. В таблице 2 представлена динамика НС. На рисунке 2 приведен график суммарной ошибки НС. Итоговый вектор весов w равен:

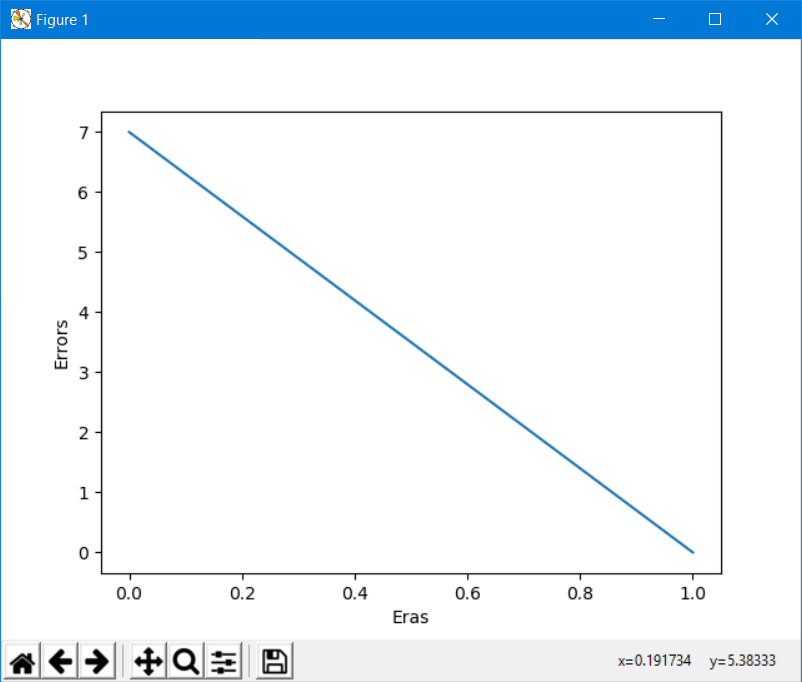
w = (0.3, -0.175, -0.27, -0.289)

Таблица 2

**Параметры НС на последовательных эпохах**

**при минимальном наборе векторов**

|  |  |  |  |
| --- | --- | --- | --- |
| **k** | **Вектор весов w** | **Выходной вектор y** | **Суммарная ошибка E** |
| 0 | [0, 0, 0, 0, 0] | [1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1] | 7 |
| 1 | [0.3, -0.175, -0.27, -0.289 ] | [1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1] | 0 |



**Рисунок 2.** График суммарной ошибки НС при полном наборе обучающих векторов

При обучении на минимальном наборе [[0,0,0,1], [1,1,0,0]] НС обучилась за 4 эпохи. В таблице 3 представлена динамика НС. На рисунке 3 приведен график суммарной ошибки НС. Итоговый вектор весов w равен:

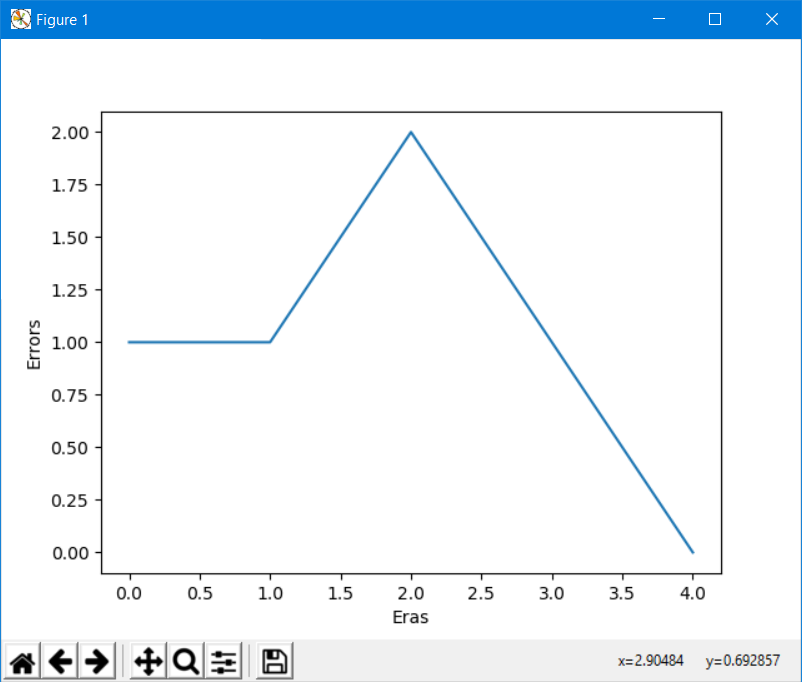
w = (0.3, -0.099, -0.099, -0.555)

Таблица 3

**Параметры НС на последовательных эпохах**

**при минимальном наборе векторов**

|  |  |  |  |
| --- | --- | --- | --- |
| **k** | **Вектор весов w** | **Выходной вектор y** | **Суммарная ошибка E** |
| 0 | [0, 0, 0, 0, 0] | [1, 1] | 1 |
| 1 | [-0.3, -0.11, -0.11, -0.3 ] | [0, 0] | 1 |
| 2 | [ 0, -0.07, -0.07, -0.285] | [0, 1] | 2 |
| 3 | [ 0, -0.14, -0.14, -0.57] | [0, 0] | 1 |
| 4 | [ 0.3, -0.099, -0.099, -0.555] | [1, 0] | 0 |



**Рисунок 3.** График суммарной ошибки НС при минимальном наборе обучающих векторов

# **Вывод**

В процессе выполнения лабораторной работы было проведено 2 обучения при полном и минимальном наборе векторов НС с RBF. Преимуществом данной архитектуры является их быстрое обучение и относительно простое построение сетей.

# **Приложение**

Ссылка на гитхаб с кодом:

<https://github.com/Kustov-Ilya/lab_work_II/blob/master/lab3.py>

Листинг №1

import numpy as np

import itertools

import matplotlib.pyplot as plt

import math

def RBF():

number\_of\_1 = [Function(i) for i in itertools.product([0, 1], repeat = 4)].count(1)

if number\_of\_1 <=8:

return [i for i in itertools.product([0, 1], repeat = 4) if Function(i)]

else:

return [i for i in itertools.product([0, 1], repeat = 4) if not Function(i)]

def Train(selection):

nu = 0.3

c = RBF() #RBF-нейроны

u = np.zeros(len(c) + 1, float)

era = 0

error\_list = []

OutText = ''

while not error\_list or error\_list[era - 1] > 0:

OutText += f'Era: {era}\nw: {np.around(u, 3)} '

era += 1

errors = 0

vector\_y = []

for x in selection:

fi = Compute\_fi(c, x)

t = Function(x)

net = np.dot(fi, u)

y = 1 if net >= 0 else 0

vector\_y.append(str(y))

sigma = t - y

if not sigma == 0:

errors += 1

u += [nu \* sigma \* fi\_i for fi\_i in fi]

OutText += f"Errors: {errors} Vector: {''.join(vector\_y)}\n"

error\_list.append(errors)

#test for verify neuron

if Neuron\_vector(c, u) == Real\_vector():

print(f"Selection: {selection}\n{OutText}Test accessed\n{'\*' \* 70}")

Drow\_plot(era, error\_list)

return era

else:

print(f"\nTest don't accessed\nNeuron can't learn\n{'\*' \* 70}")

return 50

def Drow\_plot(era, list\_errors):

X = [i for i in range(era)]

Y = [list\_errors[i] for i in X]

plt.plot(X, Y)

plt.xlabel('Eras')

plt.ylabel('Errors')

plt.show()

#будевый вектор, который высчитал нейрон

def Neuron\_vector(c, u):

out = ''

for x in [list(i) for i in itertools.product([0, 1], repeat = 4)]:

fi = Compute\_fi(c, x)

net = np.dot(fi, u)

y = 1 if net >= 0 else 0

out += str(y)

return out

#реальный будевый вектор

def Real\_vector():

real = ''

for i in [list(i) for i in itertools.product([0, 1], repeat=4)]:

real+=str(Function(i))

return real

#значения в RBF-нейронах

def Compute\_fi(c, x):

difference= [[np.array(x) - np.array(ci)] for ci in c]

return [1] + [math.exp(-np.sum(np.power(diff, 2))) for diff in difference]

def Function(x):

return int(not (x[0] or x[1]) or x[2] or x[3])

def Selection():

Set = [list(i) for i in itertools.product([0, 1], repeat=4)]

for i in range(1, 16): #поиск минимальной обучающей выборки

for rezult\_selection in itertools.permutations(Set, i):

if Train(list(rezult\_selection)) < 50:

return None

if \_\_name\_\_ == "\_\_main\_\_":

Train([list(i) for i in itertools.product([0, 1], repeat = 4)])

selection = [[0, 0, 0, 1], [1, 1, 0, 0]]

Train(selection)