МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
им. Н.Э. БАУМАНА

Факультет: Информатика и системы управления
Кафедра: Информационная безопасность (ИУ8)

ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Лабораторная работа №3 на тему:

«Использование нейронных сетей с радиальными базисными функциями (RFB) на примере моделирования булевых выражений»

Вариант 4

Преподаватель:
Коннова Н.С.

Студент:
Куликова А.В.

Группа:
ИУ8-21М

Цель работы

Исследовать функционирование НС с радиальными базисными функциями (RBF) и обучить её по правилу Видроу-Хоффа

Постановка задачи

№ Варианта	Моделируемая БФ	ФА*
4	$(\overline{x_1} + x_3)x_2 + x_2x_4$	1, 2

Ход работы

Таблица 1 – Таблица истинности

x_1	x_2	x_3	x_4	F
1	0	0	0	**0**
1	0	0	1	**0**
1	0	1	0	**0**
1	0	1	1	**0**
1	1	0	0	**1**
1	1	0	1	**1**
1	1	1	0	**1**
1	1	1	1	**1**
0	0	0	0	**0**
0	0	0	1	**0**
0	0	1	0	**0**
0	0	1	1	**0**
0	1	0	0	**0**
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**1**

Норма обучения $\eta = 0{,}3$

Обучение на полном наборе:

[Полный набор]

	Номер эпохи	Вектор весов w	Выходной вектор y	Суммарная ошибка E
0	0	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	10
1	1	[0.1, 0.255, 0.444, 0.608, 0.842, 1.459, 1.52, 1.606]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	6
2	2	[-2.9, -0.464, -0.094, 0.294, 0.796, 0.517, 0.577, 0.843]	[0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]	3
3	3	[-2.0, 0.187, 0.712, 1.324, 1.861, -0.027, 0.033, 0.23]	[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1]	7
4	4	[-0.8, 0.714, 1.282, 1.955, 2.508, 0.205, 0.429, 0.67]	[0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1]	6
5	5	[-1.1, 0.604, 1.242, 1.915, 2.493, 0.58, 0.804, 1.115]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	9
6	6	[-3.8, -0.17, 0.494, 1.262, 1.911, 0.257, 0.741, 1.301]	[0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1]	2
7	7	[-1.4, 0.572, 1.417, 2.15, 2.842, 0.385, 0.773, 1.289]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	9
8	8	[-4.1, -0.175, 0.739, 1.472, 2.19, 0.088, 0.7, 1.45]	[0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1]	0

Минимальное подмножество для обучения: [0, 0, 0, 0], [0, 1, 0, 1]

Обучение на минимальном наборе:

[Минимальный набор]
([0, 0, 0, 0], [0, 1, 0, 1])

	Номер эпохи	Вектор весов w	Выходной вектор y	Суммарная ошибка E
0	0	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	8
1	1	[0.4, 0.779, 0.659, 0.919, 0.875, 0.875, 0.97, 0.954]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	8
2	2	[-0.2, 0.559, 0.319, 0.838, 0.749, 0.749, 0.94, 0.908]	[0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1]	0

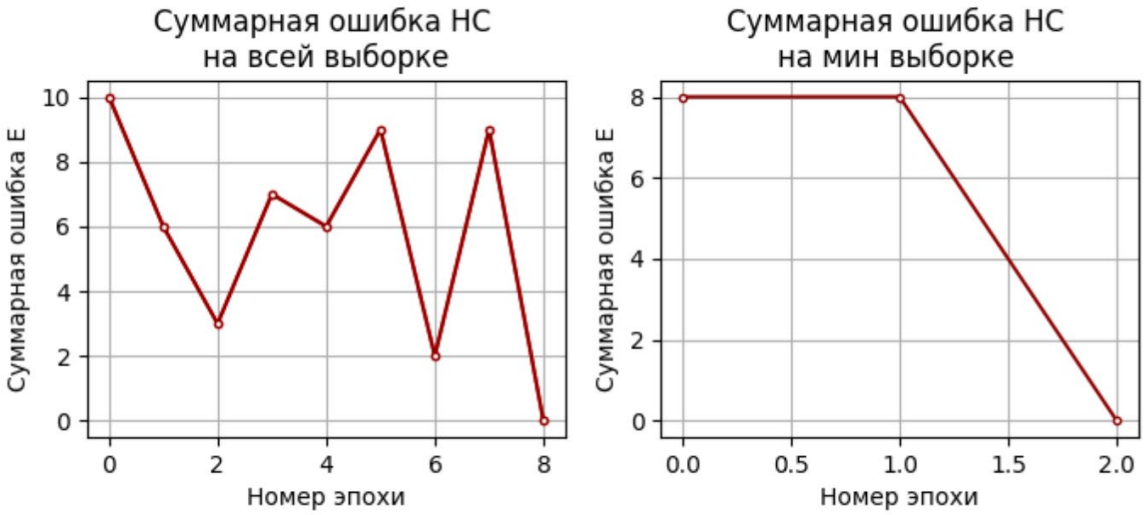

Рисунок 1 - Графики суммарной ошибки НС по эпохам обучения

Конечные значения синаптических весов имеют вид: v = [-0.2, 0.559, 0.319, 0.838, 0.749, 0.749, 0.94, 0.908]

Выводы

В ходе выполнения лабораторной работы было изучено функционирование нейронной сети с радиальными базисными функциями.

В ходе работы был найден минимально возможный набор векторов, на котором можно обучить НС.

Контрольные вопросы

1. Расскажите о НС RBF и алгоритме ее функционирования.

НС RBF — это метод машинного обучения, используемый для классификации и аппроксимации функций. Он объединяет в себе иерархическую кластеризацию данных с радиально-базисными функциями (RBF).

Принцип работы НС RBF:

1. **Иерархическая кластеризация**. Сначала данные разбиваются на кластеры с использованием иерархической кластеризации, например, алгоритмом кластеризации k-means или алгоритмом иерархической кластеризации.

2. **Распределение центров RBF**. Для каждого кластера определяется центр радиально-базисной функции (RBF). Центры могут быть распределены внутри кластера или могут быть выбраны из представителей кластера.

3. **Обучение RBF**. Для каждой радиально-базисной функции вычисляются веса, которые определяют вклад каждой функции в решение задачи классификации или аппроксимации функции.

4. **Классификация или аппроксимация**. После обучения RBF модели можно использовать для классификации новых данных или для аппроксимации неизвестной функции.

Преимущества НС RBF включают в себя способность к эффективной обработке больших объемов данных, возможность автоматического выделения структуры данных и высокую точность классификации.

2. Назовите типы радиальных базисных функций.

В радиальных базисных функциях, используемых в квантовой механике, можно выделить несколько типов.

Наиболее распространенные типы радиальных базисных функций:

1. Гауссовы функции — это функции, которые имеют форму гауссовского распределения и широко используются для аппроксимации волновых функций в квантовой механике.

2. Лагерр-радиальные функции — это функции, которые возникают при решении уравнения Шредингера для атомов с использованием метода разложения по радиальным функциям Лагерра.

3. Экспоненциальные радиальные функции — это функции, которые имеют экспоненциальную зависимость от расстояния и широко используются для описания взаимодействия между частицами.

3. Как происходят нахождение параметров и обучение НС RBF?

Нахождение параметров и обучение НС RBF включает в себя несколько шагов, которые помогают определить оптимальные значения весов и центров радиально-базисных функций.

Общий процесс обучения НС RBF:

1. Инициализация. инициализации параметров модели, таких как центры радиально-базисных функций, ширины функций и веса. Центры могут быть выбраны с помощью кластеризации данных, а ширина функций может быть установлена на основе характеристик данных.

2. Распространение вперед. Подача входных данных на сеть RBF и вычисление активации функций для каждого нейрона в слое RBF. Это позволит получить выходы от радиально-базисных функций.

3. Вычисление выхода. С учетом активаций функций и весов, рассчитать выходное значение сети RBF. Может быть выполнено путем линейной комбинации выходов функций с использованием соответствующих весов.

4. Оценка ошибки. Сравнение выхода модели с ожидаемым значением (целевым) и вычисление ошибки предсказания. Обычно используется функция потерь, как например среднеквадратичная ошибка (MSE).

5. Обновление параметров. Использование метода оптимизации, для обновления параметров модели (весов и центров) с целью минимизации ошибки предсказания. Позволит модели лучше соответствовать данным и улучшить качество прогнозов.

6. Повторение процесса. Повторение пунктов 2 - 5 для всех обучающих примеров пока модель не достигнет определенного уровня точности или стабильности.

7. Оценка производительности. После завершения обучения идет оценка производительность модели на отложенной выборке данных для проверки ее обобщающей способности.

(!!!!) Нужно учитывать, что выбор метода инициализации, функции активации, метода оптимизации и других параметров может существенно влиять на производительность модели.

Приложение А

[x1,x2,x3,x4,f]

[0, 0, 0, 0, 0]

[0, 0, 0, 1, 0]

[0, 0, 1, 0, 0]

[0, 0, 1, 1, 0]

[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, 0]

[1, 0, 1, 0, 0]

[1, 0, 1, 1, 0]

[1, 1, 0, 0, 0]

[1, 1, 0, 1, 1]

[1, 1, 1, 0, 1]

[1, 1, 1, 1, 1]

[Полный набор]

Номер эпохи		Вектор весов w	Выходной вектор y	Суммарная ошибка E
0	0	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	10
1	1	[0.1, 0.255, 0.444, 0.608, 0.842, 1.459, 1.52, 1.606]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	6
2	2	[-2.9, -0.464, -0.094, 0.294, 0.796, 0.517, 0.577, 0.843]	[0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0]	3
3	3	[-2.0, 0.187, 0.712, 1.324, 1.861, -0.027, 0.033, 0.23]	[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1]	7
4	4	[-0.8, 0.714, 1.282, 1.955, 2.508, 0.205, 0.429, 0.67]	[0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1]	6
5	5	[-1.1, 0.604, 1.242, 1.915, 2.493, 0.58, 0.804, 1.115]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	9
6	6	[-3.8, -0.17, 0.494, 1.262, 1.911, 0.257, 0.741, 1.301]	[0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1]	2
7	7	[-1.4, 0.572, 1.417, 2.15, 2.842, 0.385, 0.773, 1.289]	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	9
8	8	[-4.1, -0.175, 0.739, 1.472, 2.19, 0.088, 0.7, 1.45]	[0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1]	0

Проверка набора длины: 2

Найден лучший минимальный набор: ([0, 0, 0, 0], [0, 1, 0, 1])

Проверка набора длины: 3

Проверка набора длины: 4

Проверка набора длины: 5

Проверка набора длины: 6

Проверка набора длины: 7

Проверка набора длины: 8

Проверка набора длины: 9

Проверка набора длины: 10

Проверка набора длины: 11

Проверка набора длины: 12

Проверка набора длины: 13

Проверка набора длины: 14

Проверка набора длины: 15

Проверка набора длины: 16

[Минимальный набор]

([0, 0, 0, 0], [0, 1, 0, 1])	Номер эпохи	Вектор весов w	Выходной вектор y	Суммарная ошибка E
0	0	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	8
1	1	[0.4, 0.779, 0.659, 0.919, 0.875, 0.875, 0.97, 0.954]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]	8
2	2	[-0.2, 0.559, 0.319, 0.838, 0.749, 0.749, 0.94, 0.908]	[0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1]	0

Приложение Б

Листинг программы:

```
import numpy as np
from itertools import combinations
import pandas as pd
import matplotlib.pyplot as plt

NORM_LEARNING = 0.3
my_file = open("Intelligent_information_security_technologies.txt", "w")
def activation_function(net):
    if net >= 0:
        return 1
    return 0

def Calculation_phi(x_values, neuron_arr):
    phi=0
    for i in range(len(neuron_arr)):
        phi+=(x_values[i]-neuron_arr[i])*(x_values[i]-neuron_arr[i])*(-1)
    return np.exp(phi)

def calculation_net(weights,x_values,center_neurons_arrays):
    net = 0
    for i in range(len(center_neurons_arrays)):
        phi = Calculation_phi(x_values, center_neurons_arrays[i])
        net += weights[i+1] * phi
    net += weights[0]
    return net

def find_neuron_centers(matrix,func_vector):
    fal=0
    tru=0
    vic=0
    neuron_centers=[]
    for i in range(len(func_vector)):
        if func_vector[i]==0:
            fal=fal+1
        else:
            tru=tru+1
    if fal<=tru:
        vic=0

    else:
        vic=1
    for i in range(len(func_vector)):
        if func_vector[i]==vic:
            neuron_centers.append(matrix[i])
    return neuron_centers

def reset_model_states(model):
    for layer in model.layers:
        if hasattr(layer, 'reset_states'):
            layer.reset_states()

def learning_process(matrix, func_vector, vector_learning, sample_learning,
n=NORM_LEARNING, eralim=False):

    neuron_centers = find_neuron_centers(matrix, func_vector)
    weights = np.ones(len(neuron_centers) + 1)
    data = {'Номер эпохи': [], 'Вектор весов w': [], 'Выходной вектор y': [],
'Суммарная ошибка E': []}
    generation = 0
```

```
    prev_weights = weights.copy()

    while True:
        generation_s_weights = weights.copy()
        error = 0
        generation_s_y = []

        for i in range(len(sample_learning)):
            net = calculation_net(weights, vector_learning[i],
neuron_centers)
            y = activation_function(net)
            error += sample_learning[i] - y
            phi_array = [1] + [Calculation_phi(vector_learning[i], neuro_i)
for neuro_i in neuron_centers]
            delta = n * error * np.array(phi_array)
            weights += delta
        # error = 0 # 1
        # generation_s_y = list()
        for i in range(len(func_vector)):
            net = calculation_net(weights, matrix[i], neuron_centers)
            y = activation_function(net)
            generation_s_y.append(y)
            if func_vector[i] != generation_s_y[i]:
                error += 1

        data['Номер эпохи'].append(generation)
        data['Вектор весов w'].append(np.round(generation_s_weights, 3))
        data['Выходной вектор y'].append(generation_s_y)
        data['Суммарная ошибка E'].append(error)

        if error == 0 or (eralim and eralim - 1 == 0) or all(x == y for x, y
in zip(generation_s_y, func_vector)):
            break
        # Проверка изменения весов
        if np.allclose(prev_weights, weights, atol=1e-5):
            break

        prev_weights = weights.copy()

        generation += 1
        if eralim:
            eralim -= 1

    return data, error == 0

# Вместо того, чтобы возвращать результат после первой успешной итерации,
# продолжим поиск лучшего набора данных до конца всех комбинаций и вернуть
лучший результат

def find_less_process(matrix, func_vector, n=NORM_LEARNING, lim=20):
    sample = list()
    sample_data = None
    vector_y = []
    flag = False

    for index in range(2, len(matrix)+1):
        all_combinations = list(combinations(matrix, index))
        print('\nПроверка набора длины ' + str(index))
        my_file.write(f"\nПроверка набора длины: {str(index)}\n");
        for subset in all_combinations:
            vector_y = [func_vector[matrix.index(sub)] for sub in subset]
            data, flag = learning_process(matrix, func_vector, subset,
vector_y, n, lim)
            if flag and (sample_data is None or data['Суммарная ошибка E'][-
```

```
1] < sample_data['Суммарная ошибка E'][-1]):
                sample = subset
                sample_data = data
                my_file.write(f"Найден лучший минимальный набор:
{str(sample)}\n");
                print(f"Найден лучший минимальный набор: {str(sample)}")

                # return sample, sample_data # первый лучший вариант

    return sample, sample_data

matrix =[[ 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,],
    [ 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,]]

func_vector = [ 0,0,0,1, 1,1,1,1, 0,0,0,0, 0,1,1,1 ]

from itertools import product

def evaluate_expression(X1, X2, X3, X4):
    return (not X1 or X3) and X2 or X2 and X4

func = []
def truth_table():
    variables = ['X1', 'X2', 'X3', 'X4']
    table = []

    for assignment in product([False, True], repeat=len(variables)):
        values = dict(zip(variables, assignment))
        result = evaluate_expression(**values)
        func.append(int(result))
        row = [int(values[var]) for var in variables] + [int(result)]
        table.append(row)
    return table


result_table = truth_table()
my_file.write("[x1,x2,x3,x4,f]\n");
print("[x1,x2,x3,x4,f]")

for row in result_table:
    my_file.write(str(row)+'\n');
    print(str(row))

func_vector = func

print('[f(x1,x2,x3)]\n', func_vector)
my_file.write("\n[Полный набор]\n");
print('\n[Полный набор] ')
data = learning_process(matrix,
                        func_vector,
```

```
                        matrix,
                        func_vector)[0]
print(pd.DataFrame(data).to_string())
my_file.write(pd.DataFrame(data).to_string());
 # data = {'Номер эпохи': [], 'Вектор весов w': [], 'Выходной вектор y': [],
'Суммарная ошибка E': []}
plt.figure(figsize=(8, 6))
plt.subplot(2, 2, 1)
plt.plot(data['Номер эпохи'], data['Суммарная ошибка E'], marker='.',
color='red')
plt.plot(data['Номер эпохи'], data['Суммарная ошибка E'], marker='.',
color='darkred', markerfacecolor='white')
plt.title('Суммарная ошибка НС \nна всей выборке')
plt.xlabel('Номер эпохи')
plt.ylabel('Суммарная ошибка E')
plt.grid()

# Поиск минимального набора
sample, sample_data = find_less_process(matrix,func_vector)

print('\n[Минимальный набор]\n' + str(sample))
print(pd.DataFrame(sample_data).to_string())
my_file.write('\n[Минимальный набор]\n' + str(sample));
my_file.write(pd.DataFrame(sample_data).to_string());
my_file.close()

plt.subplot(2, 2, 2)
plt.plot(sample_data['Номер эпохи'], sample_data['Суммарная ошибка E'],
marker='.', color='darkred', markerfacecolor='white')
plt.title('Суммарная ошибка НС \nна мин выборке ')
plt.xlabel('Номер эпохи\n')
plt.ylabel('Суммарная ошибка E')
plt.grid()
plt.savefig('Intelligent_information_security_technologies.png')
plt.show()
plt.close()
```