Министерство образования Российской Федерации
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н. Э. БАУМАНА
Факультет: Информатика и системы управления
Кафедра: Информационная безопасность

«Интеллектуальные технологии информационной безопасности»

ЛАБОРАТОРНАЯ РАБОТА № 4

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

Вариант № 6

Преподаватель: Коннова Н.С.
Студент: Кошман А.А.
Группа: ИУ8-61

Москва, 2019

Оглавление

Цель работы

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

Постановка задачи

Получить модель булевой функции (БФ) на основе RBF-НС с двоичными входами $x_1, x_2, x_3, x_4 \in \{0, 1\}$, единичным входом смещения $\varphi_0 = 1$, синаптическими весами $\vartheta_0, \vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4$ двоичным выходом $y \in \{0, 1\}$ с пороговой ФА выходного нейрона, J скрытыми RBF-нейронами с гауссовой ФА $\varphi : R \to (0, 1]$ и координатами центров $c_{j1}, c_{j2}, c_{j3}, c_{j4}$, $(j = \overline{1, J})$ (рис. 1)

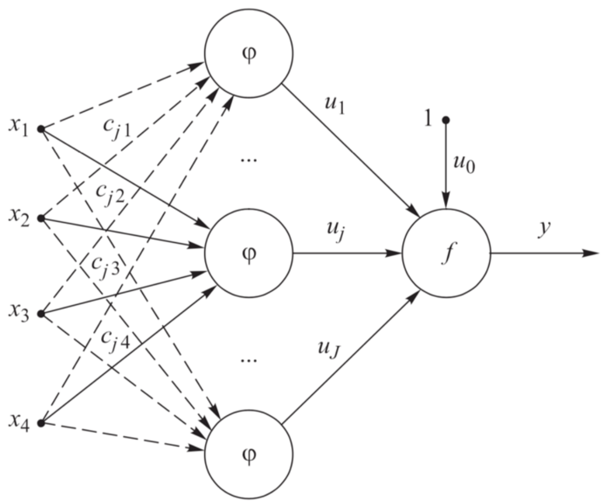

Рисунок 1 – Нейронная сеть RBF

Условие

$$F(x_1, x_2, x_3, x_4) = x_3\, x_4 + \overline{x_1} + \overline{x_2}$$

$$F(x_1, x_2, x_3, x_4) = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1)$$

Задание № 1

Обучение НС с использованием всех комбинаций переменных x_1, x_2, x_3, x_4, используя пороговую ФА :

$$f\ (net) = \begin{cases} 1, net \geq 0, \\ 0, net < 0; \end{cases}$$

Находим количество RBF-нейронов: J = 3

Центры RBF-нейронов располагаем в точках:

$$C^{(1)} = (1, 1, 0, 0)$$
$$C^{(2)} = (1, 1, 0, 1)$$
$$C^{(3)} = (1, 1, 1, 0)$$

Таблица 1. Параметры НС на последовательных эпохах (пороговая ФА)

k	Вектор весов w	Выходной вектор y	Суммарная ошибка E
0	[0, 0, 0, 0, 0]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0]	2
1	[0. -0.259 0.11 0.11]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0]	2
2	[0. -0.329 -0.079 0.221]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0]	2
3	[0. -0.399 0.031 0.031]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]	1
4	[0.3 -0.358 0.141 0.141]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0]	2
5	[0.3 -0.428 -0.048 0.252]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0]	3
6	[0. -0.608 -0.238 0.062]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]	1
7	[0.3 -0.568 -0.127 0.173]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0]	2
8	[0.3 -0.637 -0.017 -0.017]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0]	2
9	[0.3 -0.707 -0.207 0.093]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0]	2
10	[0.3 -0.777 -0.096 -0.096]	[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1]	0

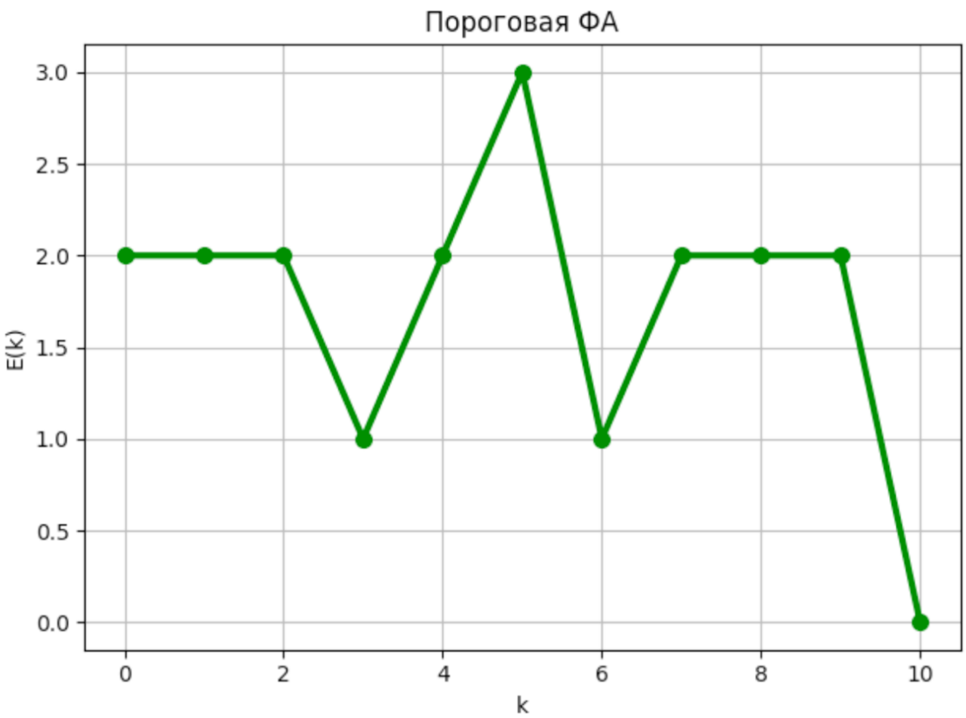

Рисунок 1. График суммарной ошибки НС по эпохам обучения (пороговая ФА)

Задание № 2

Обучение НС с использованием части комбинаций переменных x_1, x_2, x_3, x_4, используя пороговую ФА.

Последовательно увеличивая выборку количества векторов, найдем наименьшее количество необходимых для обучения векторов.

Минимальный набор обучающих векторов :

$$x^{(1)} = (0, 0, 0, 0);\ x^{(2)} = (0, 0, 0, 1);\ x^{(3)} = (1, 1, 0, 0);\ x^{(4)} = (1, 1, 0, 10);$$

$$x^{(5)} = (1, 1, 1, 0);\ x^{(6)} = (1, 1, 1, 1);$$

Вектор синаптических коэффициентов :

w =[0.3   -0.777 -0.096 -0.096]

Для полного обучения потребовалось 10 эпох :

Таблица 3. Параметры НС на последовательных эпохах (пороговая ФА) при наборе из 4 векторов

k	Вектор весов w	Выходной вектор y	Суммарная ошибка E
0	[0 0 0 0]	[1, 1, 1, 0, 0, 0]	2
1	[0. -0.259 0.11 0.11]	[1, 1, 0, 1, 0, 0]	2
2	[0. -0.329 -0.079 0.221]	[1, 1, 0, 0, 1, 0]	2
3	[0. -0.399 0.031 0.031]	[1, 1, 0, 0, 0, 0]	1
4	[0.3 -0.358 0.141 0.141]	[1, 1, 0, 1, 0, 0]	2
5	[0.3 -0.428 -0.048 0.252]	[1, 1, 0, 1, 1, 0]	3
6	[0. -0.608 -0.238 0.062]	[1, 1, 0, 0, 0, 0]	1
7	[0.3 -0.568 -0.127 0.173]	[1, 1, 0, 0, 1, 0]	2
8	[0.3 -0.637 -0.017 -0.017]	[1, 1, 0, 1, 0, 0]	2
9	[0.3 -0.707 -0.207 0.093]	[1, 1, 0, 0, 1, 0]	2
10	[0.3 -0.777 -0.096 -0.096]	[1, 1, 0, 0, 0, 1]	0

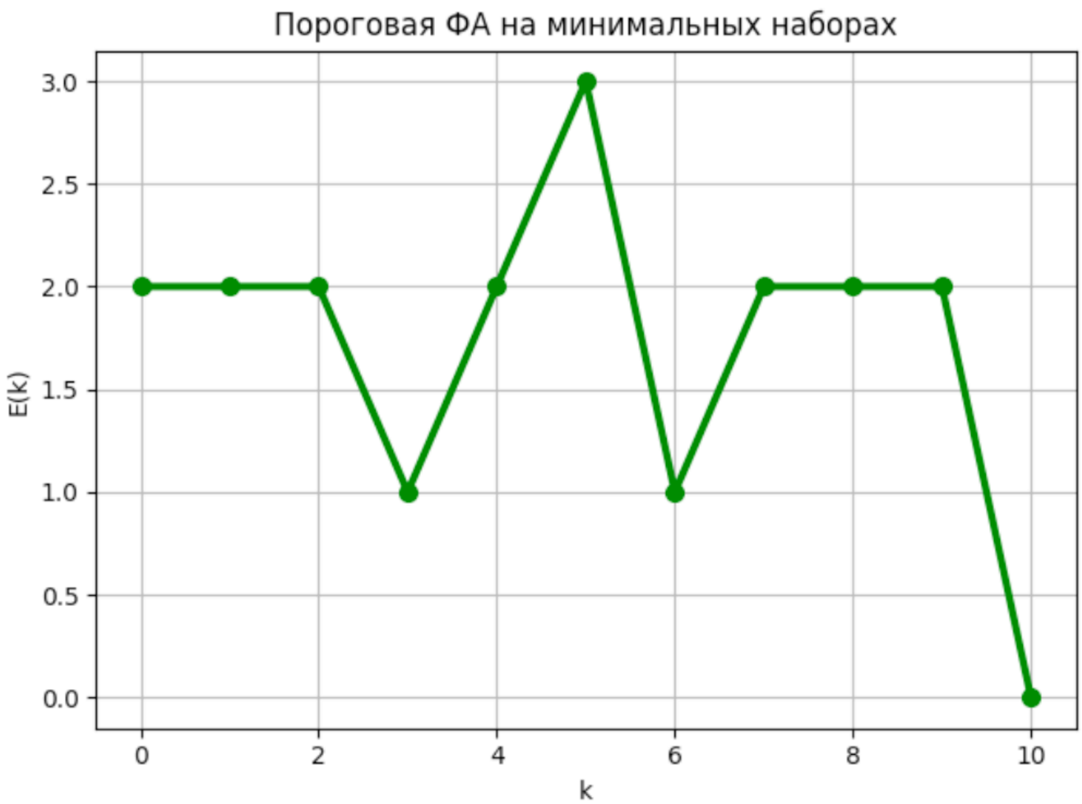

Рисунок 3. График суммарной ошибки НС по эпохам обучения с минимальным количеством наборов (пороговая ФА)

Выводы

В процессе лабораторной работы было исследовано функционирование НС с радиальными базисными функциями и произведено ее обучение по правилу Видроу – Хофа.

Было произведено обучение НС с использованием пороговой ФА и RBF на всех и на минимальных наборах. Количество RBF-нейронов было равно 3, было найдено 6 минимальных наборов, на которых НС полностью обучилась. В обоих случаях НС обучилась за 10 эпох.

Контрольные вопросы

Вопрос № 1. Расскажите о НС RBF и алгоритме ее функционирования

НС RBF - это нейронная сеть прямого распространения сигнала, которая содержит промежуточный (скрытый) слой радиально симметричных нейронов. Такой нейрон преобразовывает расстояние от данного входного вектора до соответствующей ему фиксированной точки пространства X по некоторому нелинейному закону, заданному радиальной функцией.

Вопрос № 2. Назовите типы радиальных базисных функций

Часто используемые радиально-базисные функции включают в себя ($r = \| x - x_i \|$) :

- Функция Гаусса
$$\phi(r) = e^{-(\varepsilon r)^2}$$
- Мультиквадратичная
$$\phi(r) = \sqrt{1 + (\varepsilon r)^2}$$
- Обратная мультиквадратичная
$$\phi(r) = \frac{1}{\sqrt{1 + (\varepsilon r)^2}}$$
- Полигармонический сплайн
$$\phi(r) = r^k, \qquad k = 1, 3, 5, \dots$$
$$\phi(r) = r^k \ln(r), \qquad k = 2, 4, 6, \dots$$
- Тонкий сплайн пластины
$$\phi(r) = r^2 \ln(r)$$

Вопрос № 3. Как происходит нахождение параметров и обучение НС RBF?

Для заданной БФ количество RBF-нейронов необходимо выбирать из соотношения $J = \min\{J_0, J_1\}$, где J_0, J_1 – количество векторов $x = (x_1, x_2, x_3, x_4)$ соответствующих значениям БФ «0» и «1». Центры RBF $C^{(j)} = (c_{j1}, c_{j2}, c_{j3}, c_{j4})$ должны совпадать с концами этих векторов.

Алгоритм функционирования НС с гауссовой RBF имеет вид :

$$\varphi_j(X) = \exp\left(-\sum_{i=1}^{4}(x_i - c_{ji})^2\right), \quad j = \overline{1, J};$$

$$\text{net} = \sum_{j=1}^{J} \upsilon_j \varphi_j(X) + \upsilon_0;$$

$$y(\text{net}) = \begin{cases} 1, & \text{net} \geq 0, \\ 0, & \text{net} < 0, \end{cases}$$

где net – сетевой (комбинированный) вход; y – реальный выход НС.

Каждая эпоха обучения включает в себя цикл последовательного предъявления всех образцов обучающей выборки на вход НС. Каждый элементарный шаг обучения производится коррекция весов по правилу Видроу-Хоффа:

$$\upsilon_j^{(l+1)} = \upsilon_j^{(l)} + \Delta\upsilon_j^{(l)},$$
$$\Delta\upsilon_j^{(l)} = \eta\delta^{(l)}\varphi_j^{(l)}(X),$$

На каждой эпохе суммарная квадратичная ошибка Е(к) равна расстоянию Хэмминга между векторами целевого и реального выходов.

Приложения

Файл ‘script.py’ :

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

n = 0.3

def F(x): # возвращает результат моделируемой булевой функции
    return int(x[2] and x[3] or not(x[0]) or not(x[1]))

def Y(net): # возвращает результат пороговой ФА
    return 1 if net >= 0 else 0

def DeltaW(x, q): # находит величину, на которую изменятся Wi, для пороговой ФА
    return n * q * x

def Net(x, w): # находит значение сетевого входа НС
    return sum([w_i * x_i for w_i, x_i in zip(w[1:], x)]) + w[0]


def Fi(x, c):
    return np.exp((-1) * sum([ (x_i - c_i) ** 2  for x_i, c_i in zip(x, c)]))

def FindC(X):
    RightF = [F(x_i) for x_i in X]
    count_0 = RightF.count(0)
    if (count_0 <= len(RightF)/2) :
        index = [i for i, e in enumerate(RightF) if e == 0]
    else:
        index = [i for i, e in enumerate(RightF) if e == 1]
    return [X[i] for i in index]

def FindFi(X, C):
    fi = [[Fi(X[i], C[j]) for j in range(len(C))] for i in range(len(X))]
    return fi

def MinimazeSet(X): # находит минимальные наборы из общей выборки, на которых возможно обучение 
НС
    RightF = [F(x_i) for x_i in X]
    TryF = [0 for i in range(len(X))]
    for min_num in range(0, len(X) + 1):
        for min_x in list(combinations(X, min_num)):
            C = FindC(min_x)
            if (len(C) == 0): continue
            Q = 0
            fi = FindFi(min_x, C)
            E, w = RBF(min_x, fi)
```

```
            fi = FindFi(X, C)
            for i in range(len(X)):
                TryF[i] = Y(Net(fi[i], w))
                Q += (RightF[i] - TryF[i]) ** 2
            if(Q == 0):
                return E
    return []

def RBF(X, fi): #производит обучение НС и возвращает вектор ошибок E(к)
                            # и вектор синаптических коэффициентов, на которых обучилась НС
    print("X :", X)
    RightF = [F(x_i) for x_i in X]
    w = [0 for i in range(len(fi[0]) + 1)]
    TryF = [0 for i in range(len(X))]
    E = [1]
    k = 0
    while E[k] != 0:
        print("\n\nk: ", k)
        print("w: ", np.round(w, 3))
        E.append(0)
        for i in range(len(X)):
            net = Net(fi[i], w)
            TryF[i] = Y(net)
            q = RightF[i] - TryF[i]
            for j in range(len(fi[i])):
                w[j + 1] += DeltaW(fi[i][j], q)
            w[0] += DeltaW(1, q)
            E[k+1] += q ** 2
        print("TryF: ", TryF)
        print("E: ", E[k+1])
        k += 1
    return E[1:], w

def Graph(E, name): # строит график зависимости вектора ошибок от эпохи
    if(len(E) == 0): return
    plt.plot(E, 'go-', linewidth=3, markersize=7)
    plt.grid(True)
    plt.title(name)
    plt.xlabel('k')
    plt.ylabel('E(k)')
    plt.show()


if __name__=="__main__":

    X = np.unpackbits(np.array([[j] for j in range(2 ** 4)], dtype=np.uint8), axis=1)[:, 4:]

    C = FindC(X)
    fi = FindFi(X, C)

    print("Пороговая ФА")
    E, w = RBF(X, fi)
    Graph(E, "Пороговая ФА")

    print("Пороговая ФА  на минимальных наборах")
    E = MinimazeSet(X)
    Graph(E, "Пороговая ФА на минимальных наборах")
```