**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 |  | 1, 2 |

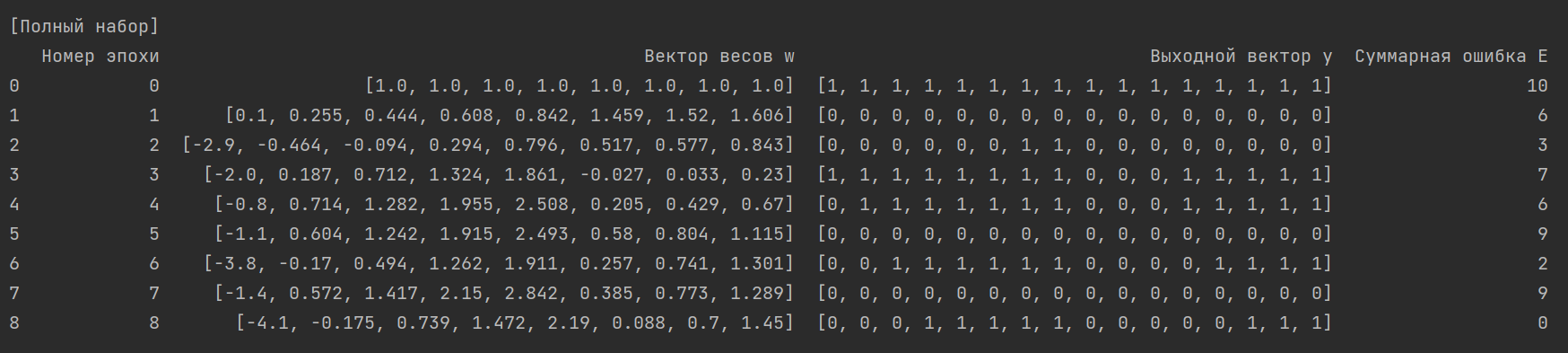
**Ход работы**

Таблица 1 – Таблица истинности

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***x1*** | ***x2*** | ***x3*** | ***x4*** | ***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** |

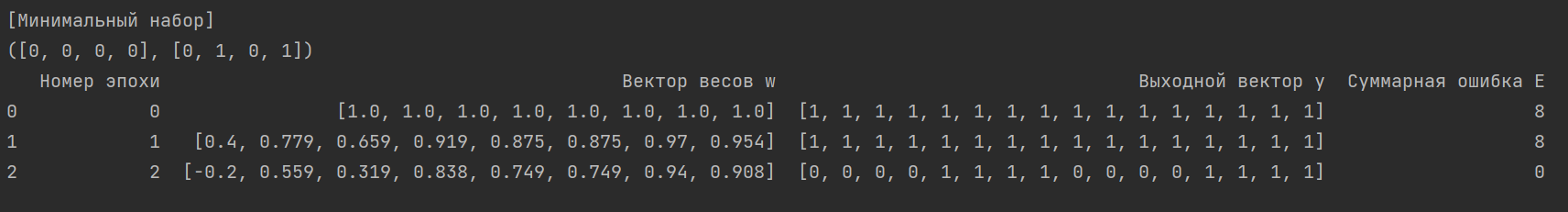
Норма обучения 𝜂 = 0,3

Обучение на полном наборе:



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

Обучение на минимальном наборе:



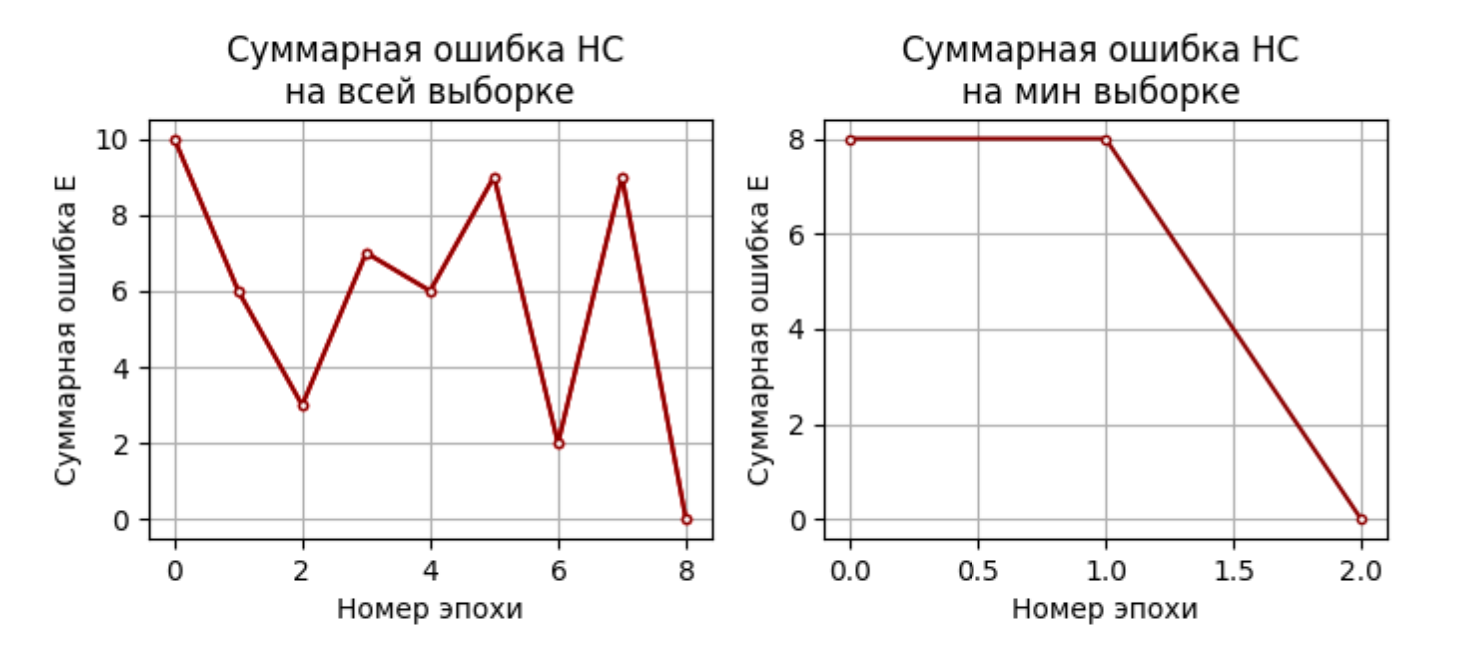


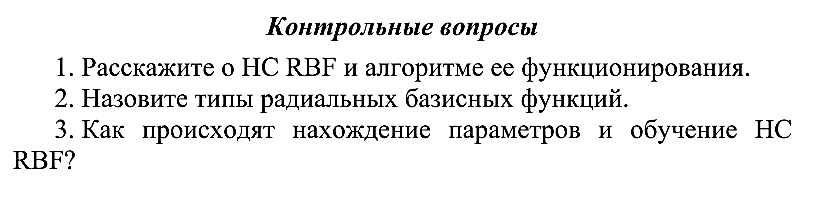
Рисунок 1 - Графики суммарной ошибки НС по эпохам обучения

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.



HC 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 HC RBF:

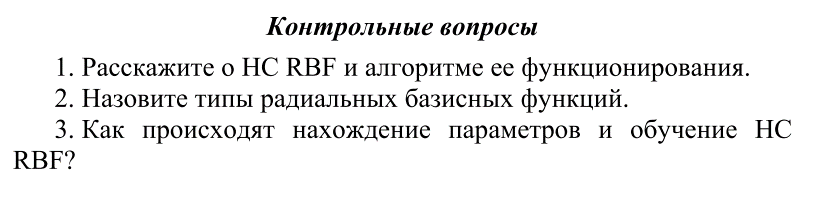
1. **Иерархическая кластеризация**. Сначала данные разбиваются на кластеры с использованием иерархической кластеризации, например, алгоритмом кластеризации k-means или алгоритмом иерархической кластеризации.

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



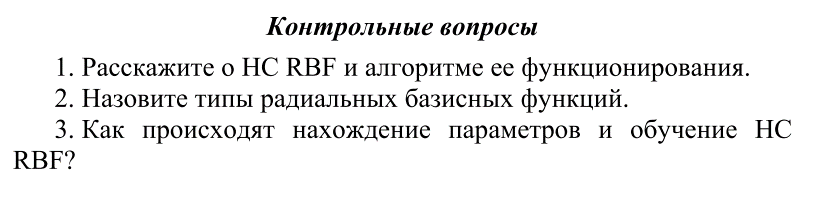
В радиальных базисных функциях, используемых в квантовой механике, можно выделить несколько типов.

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.



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 Е

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

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 Е

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 Сalculation\_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 = Сalculation\_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 Е': []}  
 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] + [Сalculation\_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 Е'].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 Е'][-1] < sample\_data['Суммарная ошибка Е'][-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 Е': []}  
plt.figure(figsize=(8, 6))  
plt.subplot(2, 2, 1)  
plt.plot(data['Номер эпохи'], data['Суммарная ошибка Е'], marker='.', color='red')  
plt.plot(data['Номер эпохи'], data['Суммарная ошибка Е'], marker='.', color='darkred', markerfacecolor='white')  
plt.title('Cуммарная ошибка НС \nна всей выборке')  
plt.xlabel('Номер эпохи')  
plt.ylabel('Суммарная ошибка Е')  
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 Е'], marker='.', color='darkred', markerfacecolor='white')  
plt.title('Cуммарная ошибка НС \nна мин выборке ')  
plt.xlabel('Номер эпохи\n')  
plt.ylabel('Суммарная ошибка Е')  
plt.grid()  
plt.savefig('Intelligent\_information\_security\_technologies.png')  
plt.show()  
plt.close()