МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №2

по дисциплине «Нейронные сети»

тема: «Простейшие однослойные нейрокомпьютерные сети.

Использование правила Хэбба»

Выполнила: ст. группы МИВТ-201

Браткова Ирина Олеговна

Белгород 2020 г

***Цель работы:*** *закрепление теоретических знаний, получение практических навыков использования однослойной нейрокомпьютерной сети, обученной на основе правила Хэбба.*

***Индивидуальные задания:***

* разработать черно-белые изображения пяти первых букв Вашей фамилии (буквы должны быть разными);
* разработать архитектуру нейронной сети;
* реализовать алгоритм обучения нейросети на основе правила Хэбба, при этом использовать бинарную функцию активации;
* исследовать возможности алгоритма Хэбба в однослойной нейросети.

***Выполнение***

Аналогично первой лабораторной работе, при разработке был выбран язык Python. Интерфейс, разработанный для данной лабораторной работы, базируется на интерфейсе первой ЛР, и имеет следующий вид:



При запуске кнопки А1, А2, В1, В2, C1, C2, D1, D2, E1, E2 инициализированы буквами Б, Р, А, Т, К (первые пять букв фамилии).

Нажатие на кнопки «Запомнить <NM>» запоминает текущее состояние поля рисования для буквы N с номером M. Рядом с соответствующей буквой после запоминания устанавливается флажок, указывающий на то, что буква сохранена. Повторное нажатие на кнопки типа «<NM>» (после запоминания) отобразит на поле сохраненную букву.

При желании можно сохранить любые наборы букв, отличные от тех, что предусмотрены в программе.

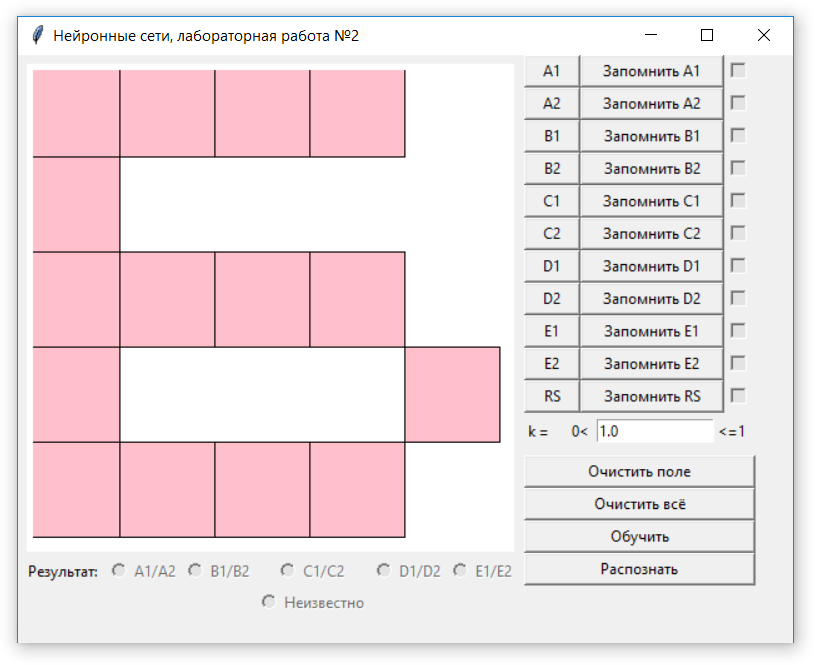
Набор кнопок для буквы RS имеет аналогичное поведение, однако RS не участвует в обучение, это символ для распознавания.

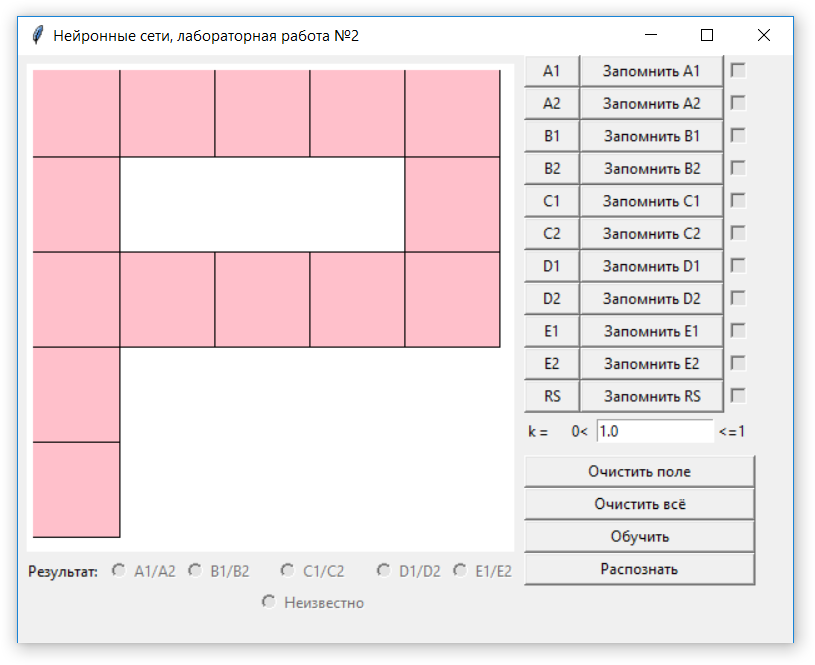
Следующий элемент – коэффициент плавности обучения, по умолчанию установленный в значение 1.0.

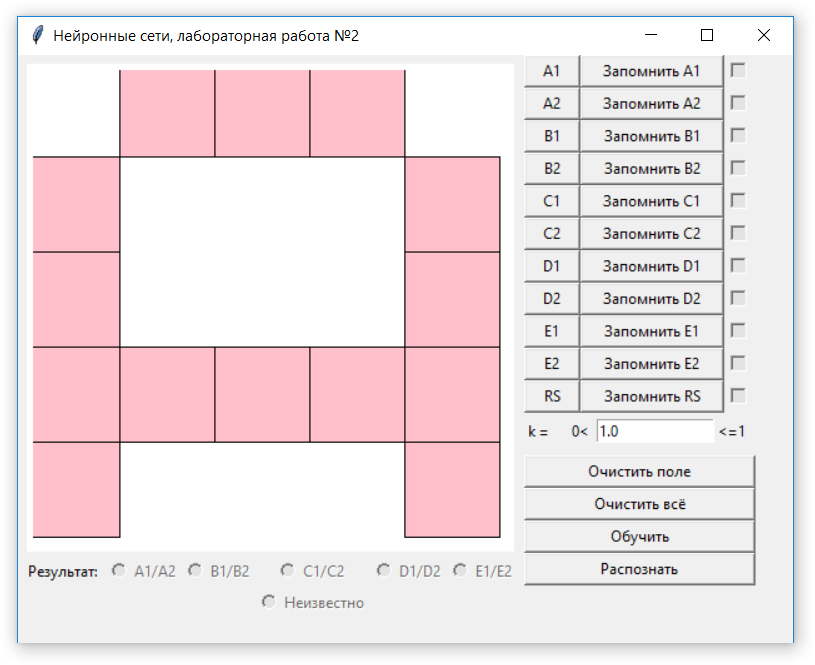
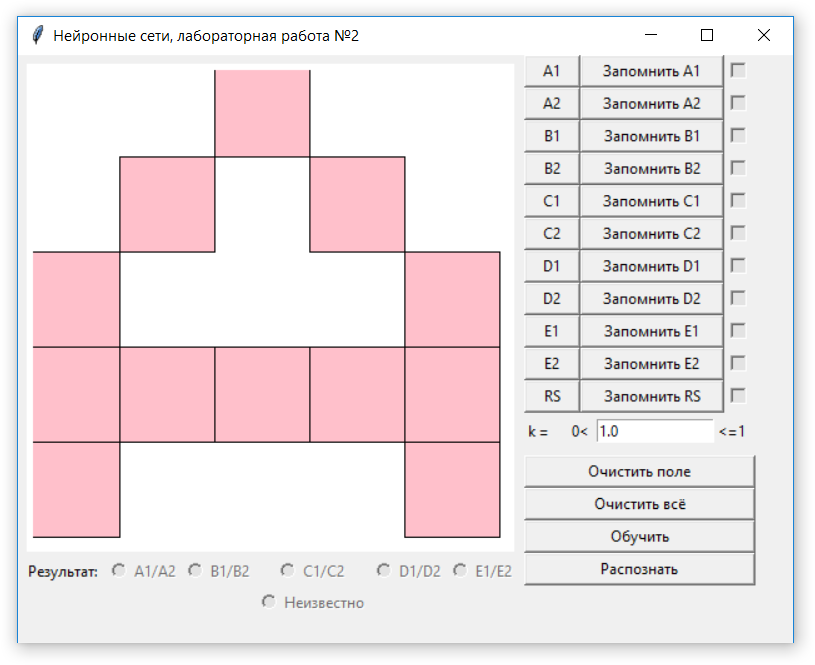
Следующие кнопки отвечают за очистку поля (без очистки информации о буквах), полной очистке поля и буков, хранящихся в памяти (аналог перезапуска программы), а также обучению на основе образов и распознаванию буквы RS.

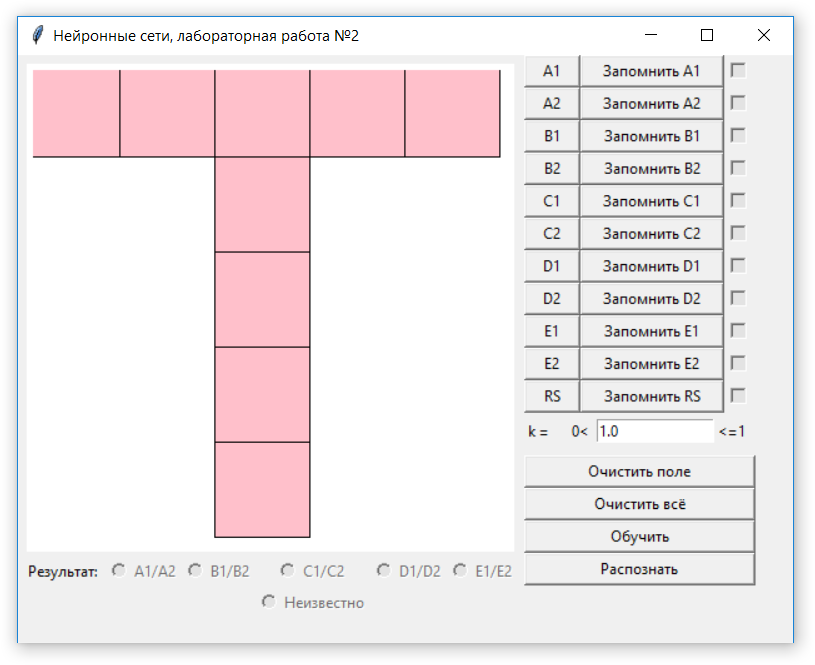
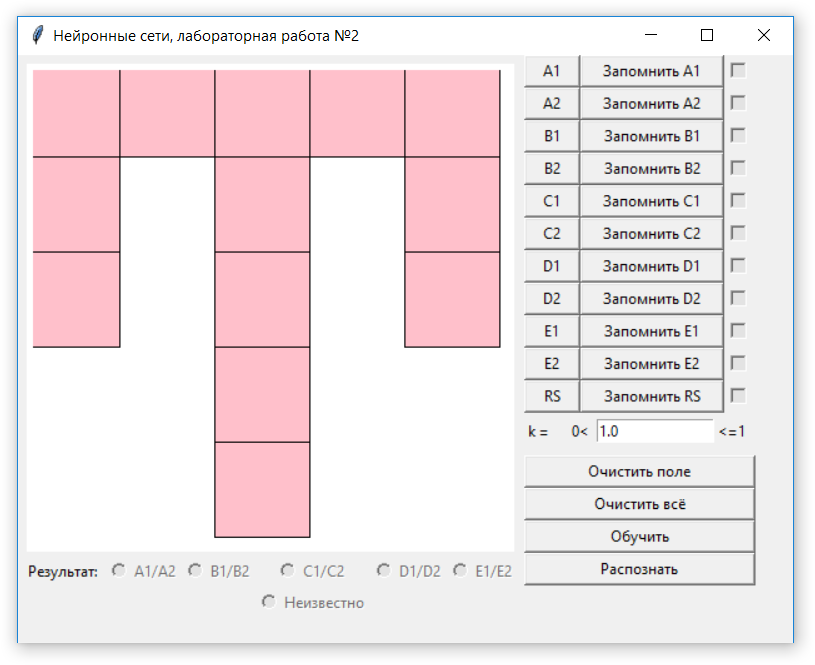
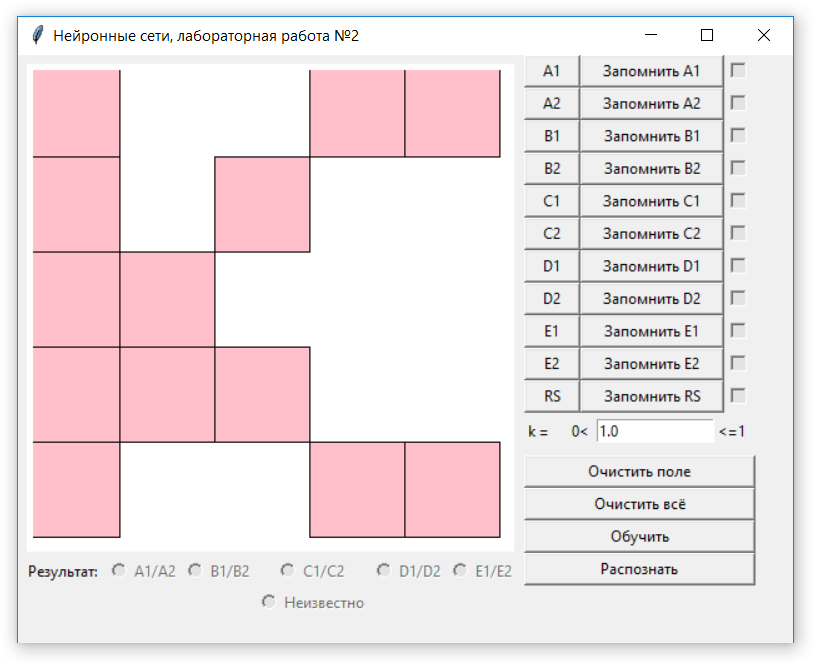
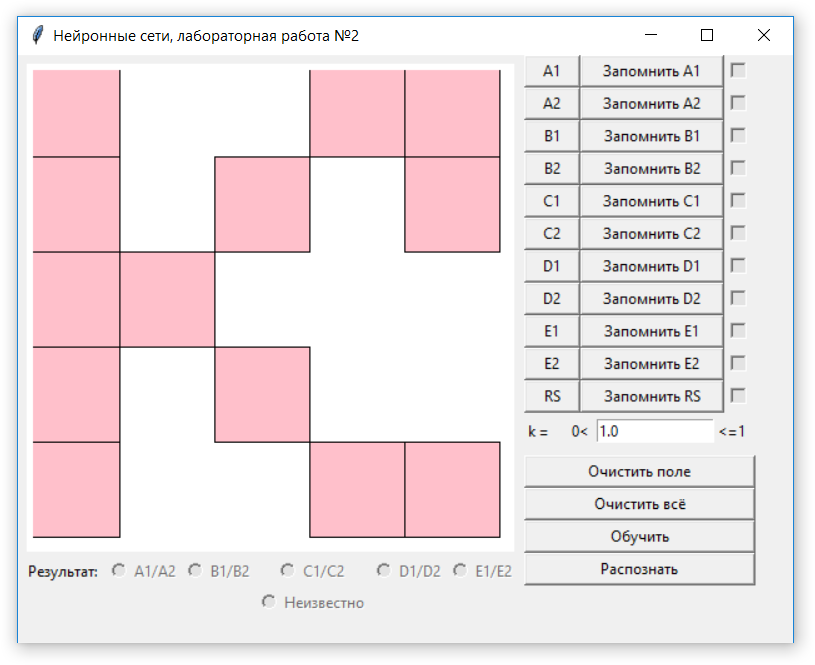
Результат работы программы отображается в блоке «Результат». В зависимости от того, к какому классу принадлежит изображение, устанавливается соответственный флажок.

***Разработанные для выполнения ЛР буквы:***

******

****** ******



***Основные модули программы***

* mainwindow.py – модуль, отвечающий за отображение экранной формы;
* controller.py – модуль, отвечающий за связь экранной формы с модулем вычислений;
* calculating.py – класс, выполняющий вычисления;
* neuron.py – класс «Нейрон»;
* functions.py – класс, содержащий функции активации.

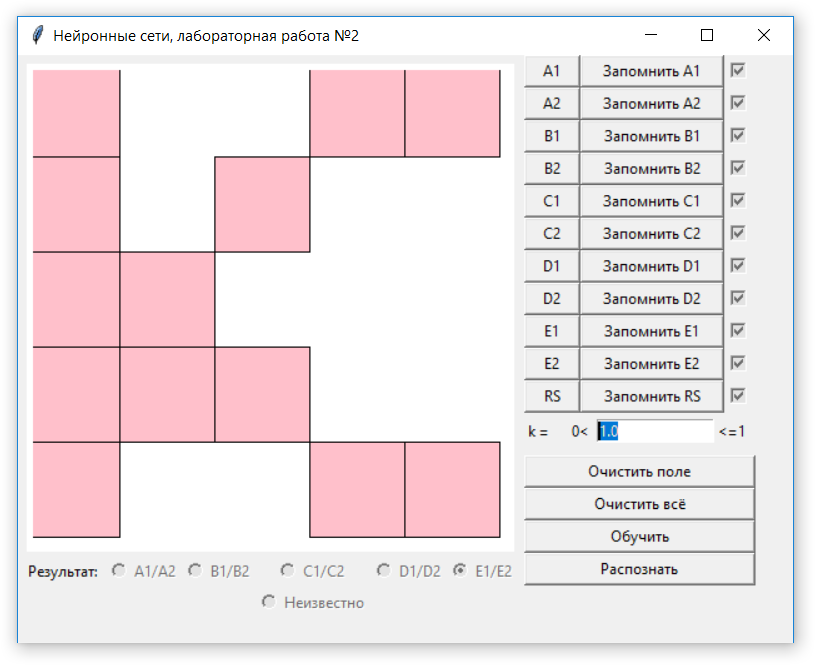
Первый и второй модуль не представляют интереса для данной лабораторной работы, поскольку их реализация полностью зависит от языка программирования и необходима для вывода информации на экран. Модули neuron.py и functions.py взяты из первой лабораторной работы и не требуют доработок при выполнении лабораторной работы №2. Основные изменения сделаны в классе calculating.py.

***Calculating.py***

**class** Calculator:  
 **def** \_\_init\_\_(self):  
 *# БУКВЫ* self.letter\_a1 = **None** self.letter\_a2 = **None** self.letter\_b1 = **None** self.letter\_b2 = **None** self.letter\_c1 = **None** self.letter\_c2 = **None** self.letter\_d1 = **None** self.letter\_d2 = **None** self.letter\_e1 = **None** self.letter\_e2 = **None** self.letter\_rs = **None** self.letters\_list = **None** self.rs = **None** self.symbols = [**'A1'**, **'A2'**, **'B1'**, **'B2'**, **'C1'**, **'C2'**, **'D1'**, **'D2'**, **'E1'**, **'E2'**]  
  
 *# M, K, Neurins, Ws, Ss* self.m = **None** self.k = **None** self.neurons = **None** self.ws = **None** self.ss = **None** self.avgs = **None** self.rs\_s = **None  
  
 def** teaching(self):  
 print(**'\n>> ОБУЧЕНИЕ'**)  
 self.\_\_make\_letters\_list()  
 print(**'\n>> Создание обучающих выборок'**)  
 self.\_\_make\_all\_m()  
 print(**'\n>> Создание нейронов на основе бинарной фунцкии'**)  
 self.\_\_make\_neurons()  
 print(**'\n>> Подсчет весовых коэффициентов для нейронов'**)  
 self.\_\_calculate\_ws()  
 print(**'\n>> Суммарные входные сигналы'**)  
 self.\_\_calculate\_ss()  
 print(**'\n>> Среднее арифметическое'**)  
 self.\_\_calculate\_avgs()  
  
 **def** recognition(self):  
 print(**'\n>> РАСПОЗНАВАНИЕ...'**)  
 print(**'\n>> Буква для распознавания'**)  
 self.\_\_make\_rs\_list\_binary()  
 print(**'\n>> Подсчет схожести'**)  
 self.\_\_calculate\_gemini()  
 print(**'\n>> Суммарный входной сигнал на разных нейронах'**)  
 self.\_\_calculate\_s\_for\_rs()  
 print(**'\n>> Результат'**)  
 **return** self.\_\_image\_class()  
  
 *# Подсчёт сходства и вывод предположения о том, к какому классу будет относиться буква* **def** \_\_calculate\_gemini(self):  
 maximum = 0  
 index = 0  
 **for** i **in** range(len(self.letters\_list)):  
 j = i + 1  
 s = **'Буква №'** + str(j) + **': '** letter = copy\_and\_insert\_one(self.letters\_list[i][0])  
 tmp1 = self.\_\_help\_gemini(self.rs, letter)  
 s += str(tmp1) + **', '** letter = copy\_and\_insert\_one(self.letters\_list[i][1])  
 tmp2 = self.\_\_help\_gemini(self.rs, letter)  
 s += str(tmp2)  
 max\_tmp = tmp1 + tmp2  
 **if** max\_tmp >= maximum:  
 maximum = max\_tmp  
 index = i  
 print(s)  
 print(**'Предположительно rs похожа на букву №'** + str(index+1))  
  
 *# Определяет класс изображения, вовзращая массив result,  
 # где единица находится на том месте, какому классу отноится изображение* **def** \_\_image\_class(self):  
 result = []  
 **for** i **in** range(len(self.neurons)):  
 j = i + 1  
 r = self.neurons[i].function[1](self.rs\_s[i], self.avgs[i])  
 print(str(j) + **': '** + str(r))  
 result.append(r)  
 **return** result  
  
 *# Помощь в подсчёте схожести (t - кол-во одинаковых символов для буквы а и б)* **def** \_\_help\_gemini(self, letter\_a, letter\_b):  
 t = 0  
 **for** i **in** range(len(letter\_a)):  
 **if** letter\_b[i] == letter\_a[i]:  
 t += 1  
 **return** t  
  
 *# Подсчёт среднего значения для функции активации* **def** \_\_calculate\_avgs(self):  
 res = []  
 **for** i **in** range(len(self.ss)):  
 j = i + 1  
 tmp = statistics.mean(self.ss[i])  
 res.append(tmp)  
 print(str(j) + **': '**, end=**''**)  
 print(tmp)  
 self.avgs = res  
  
 *# Подсчёт суммарных входных сигналов на нейронах для каждой буквы* **def** \_\_calculate\_ss(self):  
 res = []  
 **for** i **in** range(len(self.letters\_list)):  
 j = i + 1  
 print(**'\nНейрон '** + str(j))  
 neuron = self.neurons[i]  
 m\_list = self.m[i]  
 tmp = self.\_\_calculate\_s(m\_list, neuron)  
 res.append(tmp)  
 self.ss = res  
  
 **def** \_\_calculate\_s(self, m\_list, neuron):  
 s\_list = []  
 **for** i **in** range(len(m\_list)):  
 s = neuron.calculate\_s(m\_list[i][0])  
 s\_list.append(s)  
 print(self.symbols[i] + **', s = '** + str(s))  
 **return** s\_list  
  
 *# Подсчет суммарного входного сигнала символа RS на каждом из нейронов* **def** \_\_calculate\_s\_for\_rs(self):  
 result = []  
 **for** i **in** range(len(self.letters\_list)):  
 j = i + 1  
 s = self.neurons[i].calculate\_s(self.rs)  
 print(str(j) + **'. '** + str(s))  
 result.append(s)  
 self.rs\_s = result  
  
 *# Вычисляет весовые коэффициенты для всех нейронов* **def** \_\_calculate\_ws(self):  
 res = []  
 **for** i **in** range(len(self.letters\_list)):  
 j = i+1  
 print(**'\nНейрон '** + str(j) + **' и его весовые коэффициенты:'**)  
 neuron = self.neurons[i]  
 m\_list = self.m[i]  
 tmp = self.\_\_calculate\_w(neuron, m\_list)  
 res.append(tmp)  
 self.ws = res  
  
 *# Вычисляет весовые коэффициенты для нейрона* **def** \_\_calculate\_w(self, neuron, m\_list):  
 **for** i **in** range(len(m\_list)):  
 j = i + 1  
 y = copy.copy(m\_list[i][1])  
 x\_list = copy.copy(m\_list[i][0])  
 print(**'w['** + str(j) + **']: '**, end=**''**)  
 neuron.correction\_w\_list(x\_list, y, self.k)  
 **return** neuron.w\_list  
  
 *# Создает список нейронов с бинарной функцией активации* **def** \_\_make\_neurons(self):  
 function = ActivationFunctionConst()  
 count\_input = len(self.letter\_a2) + 1  
 res = []  
 **for** i **in** range(5):  
 j = i+1  
 print(**'Нейрон '** + str(j) + **' готов к бою'**)  
 neuron = Neuron(function.binary\_function, count\_input)  
 res.append(neuron)  
 self.neurons = res  
  
 *# Создает обучающие выборки для всех классов букв* **def** \_\_make\_all\_m(self):  
 res = []  
 **for** i **in** range(5):  
 j = i+1  
 print(**'m-'** + str(j))  
 tmp = self.\_\_make\_m(i)  
 res.append(tmp)  
 self.m = res  
  
 *# Создает обучающую выборку,  
 # устанавливая y = 1 для пар букв на месте index,  
 # иначе y = 0* **def** \_\_make\_m(self, index):  
 result = []  
 lst = self.\_\_upd\_binary\_list()  
 **for** i **in** range(len(lst)):  
 **if** i == index:  
 tmp1 = lst[i][0], 1  
 tmp2 = lst[i][1], 1  
 result.append(tmp1)  
 result.append(tmp2)  
 print(tmp1)  
 print(tmp2)  
 **else**:  
 tmp1 = lst[i][0], 0  
 tmp2 = lst[i][1], 0  
 result.append(tmp1)  
 result.append(tmp2)  
 print(tmp1)  
 print(tmp2)  
 **return** result  
  
 *# Добавляет сдвиг (x[0])* **def** \_\_upd\_binary\_list(self):  
 result = []  
 **for** el **in** self.letters\_list:  
 tmp1 = copy\_and\_insert\_one(el[0])  
 tmp2 = copy\_and\_insert\_one(el[1])  
 result.append((tmp1, tmp2))  
 **return** result  
  
 *# Создает лист буквы предназначенной для распознавания в бинарном представлении* **def** \_\_make\_rs\_list\_binary(self):  
 **if** self.letter\_rs **is not None**:  
 rs = copy\_and\_insert\_one(self.letter\_rs)  
 print(rs)  
 self.rs = rs  
  
 *# Создает сисок из символов для обучения* **def** \_\_make\_letters\_list(self):  
 self.letters\_list = [  
 (self.letter\_a1, self.letter\_a2),  
 (self.letter\_b1, self.letter\_b2),  
 (self.letter\_c1, self.letter\_c2),  
 (self.letter\_d1, self.letter\_d2),  
 (self.letter\_e1, self.letter\_e2)]

***Результат работы программы***

Результат работы программы при символе RS, полностью совпадающем с символом E при k = 1.



>> Вычисления при k = 1.0

>> ОБУЧЕНИЕ

>> Создание обучающих выборок

m-1

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1], 1)

([1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0], 1)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0], 0)

([1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 0)

([1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 0)

([1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 0)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1], 0)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1], 0)

m-2

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1], 0)

([1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0], 1)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0], 1)

([1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 0)

([1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 0)

([1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 0)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1], 0)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1], 0)

m-3

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1], 0)

([1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0], 0)

([1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 1)

([1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 1)

([1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 0)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1], 0)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1], 0)

m-4

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1], 0)

([1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0], 0)

([1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 0)

([1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 0)

([1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 1)

([1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 1)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1], 0)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1], 0)

m-5

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1], 0)

([1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0], 0)

([1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 0)

([1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 0)

([1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 0)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1], 1)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1], 1)

>> Создание нейронов на основе бинарной фунцкии

Нейрон 1 готов к бою

Нейрон 2 готов к бою

Нейрон 3 готов к бою

Нейрон 4 готов к бою

Нейрон 5 готов к бою

>> Подсчет весовых коэффициентов для нейронов

Нейрон 1 и его весовые коэффициенты:

w[1]: [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 0.0, 0.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]

w[2]: [2.0, 2.0, 2.0, 2.0, 2.0, 1.0, 2.0, 0.0, 0.0, 0.0, 0.0, 2.0, 2.0, 2.0, 2.0, 1.0, 2.0, 0.0, 0.0, 0.0, 2.0, 2.0, 2.0, 2.0, 2.0, 1.0]

w[3]: [1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, -1.0, 1.0, 1.0, 1.0, 1.0, 0.0, 1.0, 0.0, 0.0, 0.0, 2.0, 1.0, 2.0, 2.0, 2.0, 1.0]

w[4]: [0.0, 0.0, 0.0, 0.0, 0.0, -1.0, 0.0, 0.0, 0.0, 0.0, -2.0, 0.0, 1.0, 1.0, 1.0, -1.0, 0.0, -1.0, -1.0, -1.0, 1.0, 0.0, 2.0, 2.0, 2.0, 1.0]

w[5]: [-1.0, 0.0, 0.0, -1.0, 0.0, -1.0, 0.0, -1.0, 0.0, -1.0, -2.0, -1.0, 1.0, 1.0, 1.0, -2.0, -1.0, -2.0, -2.0, -2.0, 0.0, -1.0, 2.0, 2.0, 2.0, 0.0]

w[6]: [-2.0, 0.0, -1.0, -2.0, -1.0, -1.0, -1.0, -1.0, 0.0, -1.0, -3.0, -2.0, 1.0, 1.0, 1.0, -3.0, -2.0, -3.0, -3.0, -3.0, -1.0, -2.0, 2.0, 2.0, 2.0, -1.0]

w[7]: [-3.0, -1.0, -2.0, -3.0, -2.0, -2.0, -1.0, -1.0, -1.0, -1.0, -3.0, -2.0, 1.0, 0.0, 1.0, -3.0, -2.0, -3.0, -4.0, -3.0, -1.0, -2.0, 2.0, 1.0, 2.0, -1.0]

w[8]: [-4.0, -2.0, -3.0, -4.0, -3.0, -3.0, -2.0, -1.0, -2.0, -1.0, -4.0, -3.0, 1.0, -1.0, 1.0, -4.0, -2.0, -3.0, -5.0, -3.0, -1.0, -2.0, 2.0, 0.0, 2.0, -1.0]

w[9]: [-5.0, -3.0, -3.0, -4.0, -4.0, -4.0, -3.0, -1.0, -3.0, -1.0, -4.0, -4.0, 0.0, -1.0, 1.0, -4.0, -3.0, -4.0, -6.0, -3.0, -1.0, -3.0, 2.0, 0.0, 1.0, -2.0]

w[10]: [-6.0, -4.0, -3.0, -4.0, -5.0, -5.0, -4.0, -1.0, -4.0, -1.0, -5.0, -5.0, -1.0, -1.0, 1.0, -4.0, -4.0, -4.0, -7.0, -3.0, -1.0, -4.0, 2.0, 0.0, 0.0, -3.0]

Нейрон 2 и его весовые коэффициенты:

w[1]: [-1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, 0.0, 0.0, 0.0, 0.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, 0.0, 0.0, 0.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0]

w[2]: [-2.0, -2.0, -2.0, -2.0, -2.0, -1.0, -2.0, 0.0, 0.0, 0.0, 0.0, -2.0, -2.0, -2.0, -2.0, -1.0, -2.0, 0.0, 0.0, 0.0, -2.0, -2.0, -2.0, -2.0, -2.0, -1.0]

w[3]: [-1.0, -1.0, -1.0, -1.0, -1.0, 0.0, -1.0, 0.0, 0.0, 0.0, 1.0, -1.0, -1.0, -1.0, -1.0, 0.0, -1.0, 0.0, 0.0, 0.0, -2.0, -1.0, -2.0, -2.0, -2.0, -1.0]

w[4]: [0.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 2.0, 0.0, -1.0, -1.0, -1.0, 1.0, 0.0, 1.0, 1.0, 1.0, -1.0, 0.0, -2.0, -2.0, -2.0, -1.0]

w[5]: [-1.0, 0.0, 0.0, -1.0, 0.0, 1.0, 0.0, -1.0, 0.0, -1.0, 2.0, -1.0, -1.0, -1.0, -1.0, 0.0, -1.0, 0.0, 0.0, 0.0, -2.0, -1.0, -2.0, -2.0, -2.0, -2.0]

w[6]: [-2.0, 0.0, -1.0, -2.0, -1.0, 1.0, -1.0, -1.0, 0.0, -1.0, 1.0, -2.0, -1.0, -1.0, -1.0, -1.0, -2.0, -1.0, -1.0, -1.0, -3.0, -2.0, -2.0, -2.0, -2.0, -3.0]

w[7]: [-3.0, -1.0, -2.0, -3.0, -2.0, 0.0, -1.0, -1.0, -1.0, -1.0, 1.0, -2.0, -1.0, -2.0, -1.0, -1.0, -2.0, -1.0, -2.0, -1.0, -3.0, -2.0, -2.0, -3.0, -2.0, -3.0]

w[8]: [-4.0, -2.0, -3.0, -4.0, -3.0, -1.0, -2.0, -1.0, -2.0, -1.0, 0.0, -3.0, -1.0, -3.0, -1.0, -2.0, -2.0, -1.0, -3.0, -1.0, -3.0, -2.0, -2.0, -4.0, -2.0, -3.0]

w[9]: [-5.0, -3.0, -3.0, -4.0, -4.0, -2.0, -3.0, -1.0, -3.0, -1.0, 0.0, -4.0, -2.0, -3.0, -1.0, -2.0, -3.0, -2.0, -4.0, -1.0, -3.0, -3.0, -2.0, -4.0, -3.0, -4.0]

w[10]: [-6.0, -4.0, -3.0, -4.0, -5.0, -3.0, -4.0, -1.0, -4.0, -1.0, -1.0, -5.0, -3.0, -3.0, -1.0, -2.0, -4.0, -2.0, -5.0, -1.0, -3.0, -4.0, -2.0, -4.0, -4.0, -5.0]

Нейрон 3 и его весовые коэффициенты:

w[1]: [-1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, 0.0, 0.0, 0.0, 0.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, 0.0, 0.0, 0.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0]

w[2]: [-2.0, -2.0, -2.0, -2.0, -2.0, -1.0, -2.0, 0.0, 0.0, 0.0, 0.0, -2.0, -2.0, -2.0, -2.0, -1.0, -2.0, 0.0, 0.0, 0.0, -2.0, -2.0, -2.0, -2.0, -2.0, -1.0]

w[3]: [-3.0, -3.0, -3.0, -3.0, -3.0, -2.0, -3.0, 0.0, 0.0, 0.0, -1.0, -3.0, -3.0, -3.0, -3.0, -2.0, -3.0, 0.0, 0.0, 0.0, -2.0, -3.0, -2.0, -2.0, -2.0, -1.0]

w[4]: [-4.0, -4.0, -4.0, -4.0, -4.0, -3.0, -4.0, 0.0, 0.0, 0.0, -2.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -1.0, -1.0, -1.0, -3.0, -4.0, -2.0, -2.0, -2.0, -1.0]

w[5]: [-3.0, -4.0, -4.0, -3.0, -4.0, -3.0, -4.0, 1.0, 0.0, 1.0, -2.0, -3.0, -3.0, -3.0, -3.0, -2.0, -3.0, 0.0, 0.0, 0.0, -2.0, -3.0, -2.0, -2.0, -2.0, 0.0]

w[6]: [-2.0, -4.0, -3.0, -2.0, -3.0, -3.0, -3.0, 1.0, 0.0, 1.0, -1.0, -2.0, -3.0, -3.0, -3.0, -1.0, -2.0, 1.0, 1.0, 1.0, -1.0, -2.0, -2.0, -2.0, -2.0, 1.0]

w[7]: [-3.0, -5.0, -4.0, -3.0, -4.0, -4.0, -3.0, 1.0, -1.0, 1.0, -1.0, -2.0, -3.0, -4.0, -3.0, -1.0, -2.0, 1.0, 0.0, 1.0, -1.0, -2.0, -2.0, -3.0, -2.0, 1.0]

w[8]: [-4.0, -6.0, -5.0, -4.0, -5.0, -5.0, -4.0, 1.0, -2.0, 1.0, -2.0, -3.0, -3.0, -5.0, -3.0, -2.0, -2.0, 1.0, -1.0, 1.0, -1.0, -2.0, -2.0, -4.0, -2.0, 1.0]

w[9]: [-5.0, -7.0, -5.0, -4.0, -6.0, -6.0, -5.0, 1.0, -3.0, 1.0, -2.0, -4.0, -4.0, -5.0, -3.0, -2.0, -3.0, 0.0, -2.0, 1.0, -1.0, -3.0, -2.0, -4.0, -3.0, 0.0]

w[10]: [-6.0, -8.0, -5.0, -4.0, -7.0, -7.0, -6.0, 1.0, -4.0, 1.0, -3.0, -5.0, -5.0, -5.0, -3.0, -2.0, -4.0, 0.0, -3.0, 1.0, -1.0, -4.0, -2.0, -4.0, -4.0, -1.0]

Нейрон 4 и его весовые коэффициенты:

w[1]: [-1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, 0.0, 0.0, 0.0, 0.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, 0.0, 0.0, 0.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0]

w[2]: [-2.0, -2.0, -2.0, -2.0, -2.0, -1.0, -2.0, 0.0, 0.0, 0.0, 0.0, -2.0, -2.0, -2.0, -2.0, -1.0, -2.0, 0.0, 0.0, 0.0, -2.0, -2.0, -2.0, -2.0, -2.0, -1.0]

w[3]: [-3.0, -3.0, -3.0, -3.0, -3.0, -2.0, -3.0, 0.0, 0.0, 0.0, -1.0, -3.0, -3.0, -3.0, -3.0, -2.0, -3.0, 0.0, 0.0, 0.0, -2.0, -3.0, -2.0, -2.0, -2.0, -1.0]

w[4]: [-4.0, -4.0, -4.0, -4.0, -4.0, -3.0, -4.0, 0.0, 0.0, 0.0, -2.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -1.0, -1.0, -1.0, -3.0, -4.0, -2.0, -2.0, -2.0, -1.0]

w[5]: [-5.0, -4.0, -4.0, -5.0, -4.0, -3.0, -4.0, -1.0, 0.0, -1.0, -2.0, -5.0, -3.0, -3.0, -3.0, -4.0, -5.0, -2.0, -2.0, -2.0, -4.0, -5.0, -2.0, -2.0, -2.0, -2.0]

w[6]: [-6.0, -4.0, -5.0, -6.0, -5.0, -3.0, -5.0, -1.0, 0.0, -1.0, -3.0, -6.0, -3.0, -3.0, -3.0, -5.0, -6.0, -3.0, -3.0, -3.0, -5.0, -6.0, -2.0, -2.0, -2.0, -3.0]

w[7]: [-5.0, -3.0, -4.0, -5.0, -4.0, -2.0, -5.0, -1.0, 1.0, -1.0, -3.0, -6.0, -3.0, -2.0, -3.0, -5.0, -6.0, -3.0, -2.0, -3.0, -5.0, -6.0, -2.0, -1.0, -2.0, -3.0]

w[8]: [-4.0, -2.0, -3.0, -4.0, -3.0, -1.0, -4.0, -1.0, 2.0, -1.0, -2.0, -5.0, -3.0, -1.0, -3.0, -4.0, -6.0, -3.0, -1.0, -3.0, -5.0, -6.0, -2.0, 0.0, -2.0, -3.0]

w[9]: [-5.0, -3.0, -3.0, -4.0, -4.0, -2.0, -5.0, -1.0, 1.0, -1.0, -2.0, -6.0, -4.0, -1.0, -3.0, -4.0, -7.0, -4.0, -2.0, -3.0, -5.0, -7.0, -2.0, 0.0, -3.0, -4.0]

w[10]: [-6.0, -4.0, -3.0, -4.0, -5.0, -3.0, -6.0, -1.0, 0.0, -1.0, -3.0, -7.0, -5.0, -1.0, -3.0, -4.0, -8.0, -4.0, -3.0, -3.0, -5.0, -8.0, -2.0, 0.0, -4.0, -5.0]

Нейрон 5 и его весовые коэффициенты:

w[1]: [-1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, 0.0, 0.0, 0.0, 0.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, 0.0, 0.0, 0.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0]

w[2]: [-2.0, -2.0, -2.0, -2.0, -2.0, -1.0, -2.0, 0.0, 0.0, 0.0, 0.0, -2.0, -2.0, -2.0, -2.0, -1.0, -2.0, 0.0, 0.0, 0.0, -2.0, -2.0, -2.0, -2.0, -2.0, -1.0]

w[3]: [-3.0, -3.0, -3.0, -3.0, -3.0, -2.0, -3.0, 0.0, 0.0, 0.0, -1.0, -3.0, -3.0, -3.0, -3.0, -2.0, -3.0, 0.0, 0.0, 0.0, -2.0, -3.0, -2.0, -2.0, -2.0, -1.0]

w[4]: [-4.0, -4.0, -4.0, -4.0, -4.0, -3.0, -4.0, 0.0, 0.0, 0.0, -2.0, -4.0, -3.0, -3.0, -3.0, -3.0, -4.0, -1.0, -1.0, -1.0, -3.0, -4.0, -2.0, -2.0, -2.0, -1.0]

w[5]: [-5.0, -4.0, -4.0, -5.0, -4.0, -3.0, -4.0, -1.0, 0.0, -1.0, -2.0, -5.0, -3.0, -3.0, -3.0, -4.0, -5.0, -2.0, -2.0, -2.0, -4.0, -5.0, -2.0, -2.0, -2.0, -2.0]

w[6]: [-6.0, -4.0, -5.0, -6.0, -5.0, -3.0, -5.0, -1.0, 0.0, -1.0, -3.0, -6.0, -3.0, -3.0, -3.0, -5.0, -6.0, -3.0, -3.0, -3.0, -5.0, -6.0, -2.0, -2.0, -2.0, -3.0]

w[7]: [-7.0, -5.0, -6.0, -7.0, -6.0, -4.0, -5.0, -1.0, -1.0, -1.0, -3.0, -6.0, -3.0, -4.0, -3.0, -5.0, -6.0, -3.0, -4.0, -3.0, -5.0, -6.0, -2.0, -3.0, -2.0, -3.0]

w[8]: [-8.0, -6.0, -7.0, -8.0, -7.0, -5.0, -6.0, -1.0, -2.0, -1.0, -4.0, -7.0, -3.0, -5.0, -3.0, -6.0, -6.0, -3.0, -5.0, -3.0, -5.0, -6.0, -2.0, -4.0, -2.0, -3.0]

w[9]: [-7.0, -5.0, -7.0, -8.0, -6.0, -4.0, -5.0, -1.0, -1.0, -1.0, -4.0, -6.0, -2.0, -5.0, -3.0, -6.0, -5.0, -2.0, -4.0, -3.0, -5.0, -5.0, -2.0, -4.0, -1.0, -2.0]

w[10]: [-6.0, -4.0, -7.0, -8.0, -5.0, -3.0, -4.0, -1.0, 0.0, -1.0, -3.0, -5.0, -1.0, -5.0, -3.0, -6.0, -4.0, -2.0, -3.0, -3.0, -5.0, -4.0, -2.0, -4.0, 0.0, -1.0]

>> Суммарные входные сигналы

Нейрон 1

A1, s = -51.0 A2, s = -39.0

B1, s = -54.0 B2, s = -68.0

C1, s = -47.0 C2, s = -62.0

D1, s = -39.0 D2, s = -57.0

E1, s = -56.0 E2, s = -57.0

Нейрон 2

A1, s = -69.0 A2, s = -59.0

B1, s = -52.0 B2, s = -56.0

C1, s = -43.0 C2, s = -54.0

D1, s = -41.0 D2, s = -53.0

E1, s = -58.0 E2, s = -57.0

Нейрон 3

A1, s = -83.0 A2, s = -73.0

B1, s = -74.0 B2, s = -64.0

C1, s = -27.0 C2, s = -50.0

D1, s = -53.0 D2, s = -69.0

E1, s = -64.0 E2, s = -67.0

Нейрон 4

A1, s = -83.0 A2, s = -71.0

B1, s = -70.0 B2, s = -76.0

C1, s = -59.0 C2, s = -74.0

D1, s = -29.0 D2, s = -49.0

E1, s = -68.0 E2, s = -67.0

Нейрон 5

A1, s = -77.0 A2, s = -67.0

B1, s = -68.0 B2, s = -72.0

C1, s = -49.0 C2, s = -66.0

D1, s = -45.0 D2, s = -63.0

E1, s = -42.0 E2, s = -43.0

>> Среднее арифметическое

1: -53.0

2: -54.2

3: -62.4

4: -64.6

5: -59.2

>> РАСПОЗНАВАНИЕ...

>> Буква для распознавания

[1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1]

>> Подсчет схожести

Буква №1: 15, 14

Буква №2: 15, 16

Буква №3: 13, 15

Буква №4: 14, 14

Буква №5: 26, 24

Предположительно rs похожа на букву №5

>> Суммарный входной сигнал на разных нейронах

1. -56.0

2. -58.0

3. -64.0

4. -68.0

5. -42.0

>> Результат

1: 0

2: 0

3: 0

4: 0

5: 1

Результат работы программы при символе RS, полностью совпадающем с символом E при k = 0.1.

>> Вычисления при k = 0.1

>> ОБУЧЕНИЕ

>> Создание обучающих выборок

m-1

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1], 1)

([1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0], 1)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0], 0)

([1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 0)

([1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 0)

([1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 0)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1], 0)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1], 0)

m-2

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1], 0)

([1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0], 1)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0], 1)

([1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 0)

([1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 0)

([1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 0)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1], 0)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1], 0)

m-3

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1], 0)

([1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0], 0)

([1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 1)

([1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 1)

([1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 0)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1], 0)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1], 0)

m-4

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1], 0)

([1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0], 0)

([1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 0)

([1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 0)

([1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 1)

([1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 1)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1], 0)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1], 0)

m-5

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1], 0)

([1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0], 0)

([1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 0)

([1, 0, 1, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1], 0)

([1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 0)

([1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0], 0)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1], 1)

([1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1], 1)

>> Создание нейронов на основе бинарной фунцкии

Нейрон 1 готов к бою

Нейрон 2 готов к бою

Нейрон 3 готов к бою

Нейрон 4 готов к бою

Нейрон 5 готов к бою

>> Подсчет весовых коэффициентов для нейронов

Нейрон 1 и его весовые коэффициенты:

w[1]: [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.0, 0.0, 0.0, 0.0, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.0, 0.0, 0.0, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]

w[2]: [0.2, 0.2, 0.2, 0.2, 0.2, 0.1, 0.2, 0.0, 0.0, 0.0, 0.0, 0.2, 0.2, 0.2, 0.2, 0.1, 0.2, 0.0, 0.0, 0.0, 0.2, 0.2, 0.2, 0.2, 0.2, 0.1]

w[3]: [0.1, 0.1, 0.1, 0.1, 0.1, 0.0, 0.1, 0.0, 0.0, 0.0, -0.1, 0.1, 0.1, 0.1, 0.1, 0.0, 0.1, 0.0, 0.0, 0.0, 0.2, 0.1, 0.2, 0.2, 0.2, 0.1]

w[4]: [0.0, 0.0, 0.0, 0.0, 0.0, -0.1, 0.0, 0.0, 0.0, 0.0, -0.2, 0.0, 0.1, 0.1, 0.1, -0.1, 0.0, -0.1, -0.1, -0.1, 0.1, 0.0, 0.2, 0.2, 0.2, 0.1]

w[5]: [-0.1, 0.0, 0.0, -0.1, 0.0, -0.1, 0.0, -0.1, 0.0, -0.1, -0.2, -0.1, 0.1, 0.1, 0.1, -0.2, -0.1, -0.2, -0.2, -0.2, 0.0, -0.1, 0.2, 0.2, 0.2, 0.0]

w[6]: [-0.2, 0.0, -0.1, -0.2, -0.1, -0.1, -0.1, -0.1, 0.0, -0.1, -0.30000000000000004, -0.2, 0.1, 0.1, 0.1, -0.30000000000000004, -0.2, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.1, -0.2, 0.2, 0.2, 0.2, -0.1]

w[7]: [-0.30000000000000004, -0.1, -0.2, -0.30000000000000004, -0.2, -0.2, -0.1, -0.1, -0.1, -0.1, -0.30000000000000004, -0.2, 0.1, 0.0, 0.1, -0.30000000000000004, -0.2, -0.30000000000000004, -0.4, -0.30000000000000004, -0.1, -0.2, 0.2, 0.1, 0.2, -0.1]

w[8]: [-0.4, -0.2, -0.30000000000000004, -0.4, -0.30000000000000004, -0.30000000000000004, -0.2, -0.1, -0.2, -0.1, -0.4, -0.30000000000000004, 0.1, -0.1, 0.1, -0.4, -0.2, -0.30000000000000004, -0.5, -0.30000000000000004, -0.1, -0.2, 0.2, 0.0, 0.2, -0.1]

w[9]: [-0.5, -0.30000000000000004, -0.30000000000000004, -0.4, -0.4, -0.4, -0.30000000000000004, -0.1, -0.30000000000000004, -0.1, -0.4, -0.4, 0.0, -0.1, 0.1, -0.4, -0.30000000000000004, -0.4, -0.6, -0.30000000000000004, -0.1, -0.30000000000000004, 0.2, 0.0, 0.1, -0.2]

w[10]: [-0.6, -0.4, -0.30000000000000004, -0.4, -0.5, -0.5, -0.4, -0.1, -0.4, -0.1, -0.5, -0.5, -0.1, -0.1, 0.1, -0.4, -0.4, -0.4, -0.7, -0.30000000000000004, -0.1, -0.4, 0.2, 0.0, 0.0, -0.30000000000000004]

Нейрон 2 и его весовые коэффициенты:

w[1]: [-0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, 0.0, 0.0, 0.0, 0.0, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, 0.0, 0.0, 0.0, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1]

w[2]: [-0.2, -0.2, -0.2, -0.2, -0.2, -0.1, -0.2, 0.0, 0.0, 0.0, 0.0, -0.2, -0.2, -0.2, -0.2, -0.1, -0.2, 0.0, 0.0, 0.0, -0.2, -0.2, -0.2, -0.2, -0.2, -0.1]

w[3]: [-0.1, -0.1, -0.1, -0.1, -0.1, 0.0, -0.1, 0.0, 0.0, 0.0, 0.1, -0.1, -0.1, -0.1, -0.1, 0.0, -0.1, 0.0, 0.0, 0.0, -0.2, -0.1, -0.2, -0.2, -0.2, -0.1]

w[4]: [0.0, 0.0, 0.0, 0.0, 0.0, 0.1, 0.0, 0.0, 0.0, 0.0, 0.2, 0.0, -0.1, -0.1, -0.1, 0.1, 0.0, 0.1, 0.1, 0.1, -0.1, 0.0, -0.2, -0.2, -0.2, -0.1]

w[5]: [-0.1, 0.0, 0.0, -0.1, 0.0, 0.1, 0.0, -0.1, 0.0, -0.1, 0.2, -0.1, -0.1, -0.1, -0.1, 0.0, -0.1, 0.0, 0.0, 0.0, -0.2, -0.1, -0.2, -0.2, -0.2, -0.2]

w[6]: [-0.2, 0.0, -0.1, -0.2, -0.1, 0.1, -0.1, -0.1, 0.0, -0.1, 0.1, -0.2, -0.1, -0.1, -0.1, -0.1, -0.2, -0.1, -0.1, -0.1, -0.30000000000000004, -0.2, -0.2, -0.2, -0.2, -0.30000000000000004]

w[7]: [-0.30000000000000004, -0.1, -0.2, -0.30000000000000004, -0.2, 0.0, -0.1, -0.1, -0.1, -0.1, 0.1, -0.2, -0.1, -0.2, -0.1, -0.1, -0.2, -0.1, -0.2, -0.1, -0.30000000000000004, -0.2, -0.2, -0.30000000000000004, -0.2, -0.30000000000000004]

w[8]: [-0.4, -0.2, -0.30000000000000004, -0.4, -0.30000000000000004, -0.1, -0.2, -0.1, -0.2, -0.1, 0.0, -0.30000000000000004, -0.1, -0.30000000000000004, -0.1, -0.2, -0.2, -0.1, -0.30000000000000004, -0.1, -0.30000000000000004, -0.2, -0.2, -0.4, -0.2, -0.30000000000000004]

w[9]: [-0.5, -0.30000000000000004, -0.30000000000000004, -0.4, -0.4, -0.2, -0.30000000000000004, -0.1, -0.30000000000000004, -0.1, 0.0, -0.4, -0.2, -0.30000000000000004, -0.1, -0.2, -0.30000000000000004, -0.2, -0.4, -0.1, -0.30000000000000004, -0.30000000000000004, -0.2, -0.4, -0.30000000000000004, -0.4]

w[10]: [-0.6, -0.4, -0.30000000000000004, -0.4, -0.5, -0.30000000000000004, -0.4, -0.1, -0.4, -0.1, -0.1, -0.5, -0.30000000000000004, -0.30000000000000004, -0.1, -0.2, -0.4, -0.2, -0.5, -0.1, -0.30000000000000004, -0.4, -0.2, -0.4, -0.4, -0.5]

Нейрон 3 и его весовые коэффициенты:

w[1]: [-0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, 0.0, 0.0, 0.0, 0.0, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, 0.0, 0.0, 0.0, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1]

w[2]: [-0.2, -0.2, -0.2, -0.2, -0.2, -0.1, -0.2, 0.0, 0.0, 0.0, 0.0, -0.2, -0.2, -0.2, -0.2, -0.1, -0.2, 0.0, 0.0, 0.0, -0.2, -0.2, -0.2, -0.2, -0.2, -0.1]

w[3]: [-0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.2, -0.30000000000000004, 0.0, 0.0, 0.0, -0.1, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.2, -0.30000000000000004, 0.0, 0.0, 0.0, -0.2, -0.30000000000000004, -0.2, -0.2, -0.2, -0.1]

w[4]: [-0.4, -0.4, -0.4, -0.4, -0.4, -0.30000000000000004, -0.4, 0.0, 0.0, 0.0, -0.2, -0.4, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.4, -0.1, -0.1, -0.1, -0.30000000000000004, -0.4, -0.2, -0.2, -0.2, -0.1]

w[5]: [-0.30000000000000004, -0.4, -0.4, -0.30000000000000004, -0.4, -0.30000000000000004, -0.4, 0.1, 0.0, 0.1, -0.2, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.20000000000000004, -0.30000000000000004, 0.0, 0.0, 0.0, -0.20000000000000004, -0.30000000000000004, -0.2, -0.2, -0.2, 0.0]

w[6]: [-0.20000000000000004, -0.4, -0.30000000000000004, -0.20000000000000004, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, 0.1, 0.0, 0.1, -0.1, -0.20000000000000004, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.10000000000000003, -0.20000000000000004, 0.1, 0.1, 0.1, -0.10000000000000003, -0.20000000000000004, -0.2, -0.2, -0.2, 0.1]

w[7]: [-0.30000000000000004, -0.5, -0.4, -0.30000000000000004, -0.4, -0.4, -0.30000000000000004, 0.1, -0.1, 0.1, -0.1, -0.20000000000000004, -0.30000000000000004, -0.4, -0.30000000000000004, -0.10000000000000003, -0.20000000000000004, 0.1, 0.0, 0.1, -0.10000000000000003, -0.20000000000000004, -0.2, -0.30000000000000004, -0.2, 0.1]

w[8]: [-0.4, -0.6, -0.5, -0.4, -0.5, -0.5, -0.4, 0.1, -0.2, 0.1, -0.2, -0.30000000000000004, -0.30000000000000004, -0.5, -0.30000000000000004, -0.20000000000000004, -0.20000000000000004, 0.1, -0.1, 0.1, -0.10000000000000003, -0.20000000000000004, -0.2, -0.4, -0.2, 0.1]

w[9]: [-0.5, -0.7, -0.5, -0.4, -0.6, -0.6, -0.5, 0.1, -0.30000000000000004, 0.1, -0.2, -0.4, -0.4, -0.5, -0.30000000000000004, -0.20000000000000004, -0.30000000000000004, 0.0, -0.2, 0.1, -0.10000000000000003, -0.30000000000000004, -0.2, -0.4, -0.30000000000000004, 0.0]

w[10]: [-0.6, -0.7999999999999999, -0.5, -0.4, -0.7, -0.7, -0.6, 0.1, -0.4, 0.1, -0.30000000000000004, -0.5, -0.5, -0.5, -0.30000000000000004, -0.20000000000000004, -0.4, 0.0, -0.30000000000000004, 0.1, -0.10000000000000003, -0.4, -0.2, -0.4, -0.4, -0.1]

Нейрон 4 и его весовые коэффициенты:

w[1]: [-0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, 0.0, 0.0, 0.0, 0.0, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, 0.0, 0.0, 0.0, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1]

w[2]: [-0.2, -0.2, -0.2, -0.2, -0.2, -0.1, -0.2, 0.0, 0.0, 0.0, 0.0, -0.2, -0.2, -0.2, -0.2, -0.1, -0.2, 0.0, 0.0, 0.0, -0.2, -0.2, -0.2, -0.2, -0.2, -0.1]

w[3]: [-0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.2, -0.30000000000000004, 0.0, 0.0, 0.0, -0.1, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.2, -0.30000000000000004, 0.0, 0.0, 0.0, -0.2, -0.30000000000000004, -0.2, -0.2, -0.2, -0.1]

w[4]: [-0.4, -0.4, -0.4, -0.4, -0.4, -0.30000000000000004, -0.4, 0.0, 0.0, 0.0, -0.2, -0.4, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.4, -0.1, -0.1, -0.1, -0.30000000000000004, -0.4, -0.2, -0.2, -0.2, -0.1]

w[5]: [-0.5, -0.4, -0.4, -0.5, -0.4, -0.30000000000000004, -0.4, -0.1, 0.0, -0.1, -0.2, -0.5, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.4, -0.5, -0.2, -0.2, -0.2, -0.4, -0.5, -0.2, -0.2, -0.2, -0.2]

w[6]: [-0.6, -0.4, -0.5, -0.6, -0.5, -0.30000000000000004, -0.5, -0.1, 0.0, -0.1, -0.30000000000000004, -0.6, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.5, -0.6, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.5, -0.6, -0.2, -0.2, -0.2, -0.30000000000000004]

w[7]: [-0.5, -0.30000000000000004, -0.4, -0.5, -0.4, -0.20000000000000004, -0.5, -0.1, 0.1, -0.1, -0.30000000000000004, -0.6, -0.30000000000000004, -0.20000000000000004, -0.30000000000000004, -0.5, -0.6, -0.30000000000000004, -0.20000000000000004, -0.30000000000000004, -0.5, -0.6, -0.2, -0.1, -0.2, -0.30000000000000004]

w[8]: [-0.4, -0.20000000000000004, -0.30000000000000004, -0.4, -0.30000000000000004, -0.10000000000000003, -0.4, -0.1, 0.2, -0.1, -0.20000000000000004, -0.5, -0.30000000000000004, -0.10000000000000003, -0.30000000000000004, -0.4, -0.6, -0.30000000000000004, -0.10000000000000003, -0.30000000000000004, -0.5, -0.6, -0.2, 0.0, -0.2, -0.30000000000000004]

w[9]: [-0.5, -0.30000000000000004, -0.30000000000000004, -0.4, -0.4, -0.20000000000000004, -0.5, -0.1, 0.1, -0.1, -0.20000000000000004, -0.6, -0.4, -0.10000000000000003, -0.30000000000000004, -0.4, -0.7, -0.4, -0.20000000000000004, -0.30000000000000004, -0.5, -0.7, -0.2, 0.0, -0.30000000000000004, -0.4]

w[10]: [-0.6, -0.4, -0.30000000000000004, -0.4, -0.5, -0.30000000000000004, -0.6, -0.1, 0.0, -0.1, -0.30000000000000004, -0.7, -0.5, -0.10000000000000003, -0.30000000000000004, -0.4, -0.7999999999999999, -0.4, -0.30000000000000004, -0.30000000000000004, -0.5, -0.7999999999999999, -0.2, 0.0, -0.4, -0.5]

Нейрон 5 и его весовые коэффициенты:

w[1]: [-0.1, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, 0.0, 0.0, 0.0, 0.0, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1, 0.0, 0.0, 0.0, -0.1, -0.1, -0.1, -0.1, -0.1, -0.1]

w[2]: [-0.2, -0.2, -0.2, -0.2, -0.2, -0.1, -0.2, 0.0, 0.0, 0.0, 0.0, -0.2, -0.2, -0.2, -0.2, -0.1, -0.2, 0.0, 0.0, 0.0, -0.2, -0.2, -0.2, -0.2, -0.2, -0.1]

w[3]: [-0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.2, -0.30000000000000004, 0.0, 0.0, 0.0, -0.1, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.2, -0.30000000000000004, 0.0, 0.0, 0.0, -0.2, -0.30000000000000004, -0.2, -0.2, -0.2, -0.1]

w[4]: [-0.4, -0.4, -0.4, -0.4, -0.4, -0.30000000000000004, -0.4, 0.0, 0.0, 0.0, -0.2, -0.4, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.4, -0.1, -0.1, -0.1, -0.30000000000000004, -0.4, -0.2, -0.2, -0.2, -0.1]

w[5]: [-0.5, -0.4, -0.4, -0.5, -0.4, -0.30000000000000004, -0.4, -0.1, 0.0, -0.1, -0.2, -0.5, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.4, -0.5, -0.2, -0.2, -0.2, -0.4, -0.5, -0.2, -0.2, -0.2, -0.2]

w[6]: [-0.6, -0.4, -0.5, -0.6, -0.5, -0.30000000000000004, -0.5, -0.1, 0.0, -0.1, -0.30000000000000004, -0.6, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.5, -0.6, -0.30000000000000004, -0.30000000000000004, -0.30000000000000004, -0.5, -0.6, -0.2, -0.2, -0.2, -0.30000000000000004]

w[7]: [-0.7, -0.5, -0.6, -0.7, -0.6, -0.4, -0.5, -0.1, -0.1, -0.1, -0.30000000000000004, -0.6, -0.30000000000000004, -0.4, -0.30000000000000004, -0.5, -0.6, -0.30000000000000004, -0.4, -0.30000000000000004, -0.5, -0.6, -0.2, -0.30000000000000004, -0.2, -0.30000000000000004]

w[8]: [-0.7999999999999999, -0.6, -0.7, -0.7999999999999999, -0.7, -0.5, -0.6, -0.1, -0.2, -0.1, -0.4, -0.7, -0.30000000000000004, -0.5, -0.30000000000000004, -0.6, -0.6, -0.30000000000000004, -0.5, -0.30000000000000004, -0.5, -0.6, -0.2, -0.4, -0.2, -0.30000000000000004]

w[9]: [-0.7, -0.5, -0.7, -0.7999999999999999, -0.6, -0.4, -0.5, -0.1, -0.1, -0.1, -0.4, -0.6, -0.20000000000000004, -0.5, -0.30000000000000004, -0.6, -0.5, -0.20000000000000004, -0.4, -0.30000000000000004, -0.5, -0.5, -0.2, -0.4, -0.1, -0.20000000000000004]

w[10]: [-0.6, -0.4, -0.7, -0.7999999999999999, -0.5, -0.30000000000000004, -0.4, -0.1, 0.0, -0.1, -0.30000000000000004, -0.5, -0.10000000000000003, -0.5, -0.30000000000000004, -0.6, -0.4, -0.20000000000000004, -0.30000000000000004, -0.30000000000000004, -0.5, -0.4, -0.2, -0.4, 0.0, -0.10000000000000003]

>> Суммарные входные сигналы

Нейрон 1

A1, s = -5.1000000000000005

A2, s = -3.9000000000000004

B1, s = -5.4

B2, s = -6.800000000000001

C1, s = -4.699999999999999

C2, s = -6.2

D1, s = -3.9000000000000004

D2, s = -5.7

E1, s = -5.6000000000000005

E2, s = -5.7

Нейрон 2

A1, s = -6.900000000000001

A2, s = -5.900000000000001

B1, s = -5.2

B2, s = -5.6000000000000005

C1, s = -4.300000000000001

C2, s = -5.4

D1, s = -4.1000000000000005

D2, s = -5.300000000000001

E1, s = -5.800000000000001

E2, s = -5.700000000000001

Нейрон 3

A1, s = -8.3

A2, s = -7.300000000000001

B1, s = -7.4

B2, s = -6.4

C1, s = -2.7

C2, s = -5.000000000000001

D1, s = -5.300000000000001

D2, s = -6.9

E1, s = -6.4

E2, s = -6.7

Нейрон 4

A1, s = -8.3

A2, s = -7.1

B1, s = -7.0

B2, s = -7.6000000000000005

C1, s = -5.8999999999999995

C2, s = -7.4

D1, s = -2.9000000000000004

D2, s = -4.9

E1, s = -6.8

E2, s = -6.7

Нейрон 5

A1, s = -7.699999999999999

A2, s = -6.700000000000001

B1, s = -6.8

B2, s = -7.2

C1, s = -4.9

C2, s = -6.6

D1, s = -4.5

D2, s = -6.3

E1, s = -4.2

E2, s = -4.3

>> Среднее арифметическое

1: -5.300000000000001

2: -5.420000000000001

3: -6.24

4: -6.46

5: -5.92

>> РАСПОЗНАВАНИЕ...

>> Буква для распознавания

[1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1]

>> Подсчет схожести

Буква №1: 15, 14

Буква №2: 15, 16

Буква №3: 13, 15

Буква №4: 14, 14

Буква №5: 26, 24

Предположительно rs похожа на букву №5

>> Суммарный входной сигнал на разных нейронах

1. -5.6000000000000005

2. -5.800000000000001

3. -6.4

4. -6.8

5. -4.2

>> Результат

1: 0

2: 0

3: 0

4: 0

5: 1

Конечно, результат работы программы, запущенной на одних и тех же данных, за исключением значения коэффициента k, одинаков – изображение принадлежит к пятому классу. Однако, аналогично и первой ЛР, значения стали точнее. Например, суммарные входные сигналы символа RS на каждом из пяти нейронов в двух случаях:

|  |  |
| --- | --- |
| 1. -56.0  2. -58.0  3. -64.0  4. -68.0  5. -42.0 | 1. -5.6  2. -5.8  3. -6.4  4. -6.8  5. -4.2 |

Конечно, р