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 №3

по дисциплине «Нейронные сети»

тема: «Однослойная нейрокомпьютерная сеть.

Использование Дельта-правила»

Выполнила: ст. группы МИВТ-201

Браткова Ирина Олеговна

Белгород 2020 г

***Цель работы:*** закрепление теоретических знаний, получение практических навыков использования перцептрона.

***Индивидуальные задания:***

 разработать два черно-белых изображения первых букв Вашего имени и отчества (буквы должны быть разными);

 разработать структуру элементарного перцептрона для обучения разработанным классам изображений (выбрать и обосновать количество А-элементов);

 разработать условие подбора весовых коэффициентов между S- и A-элементами;

 реализовать алгоритм обучения элементарного перцептрона на основе Гамма-системы подкрепления;

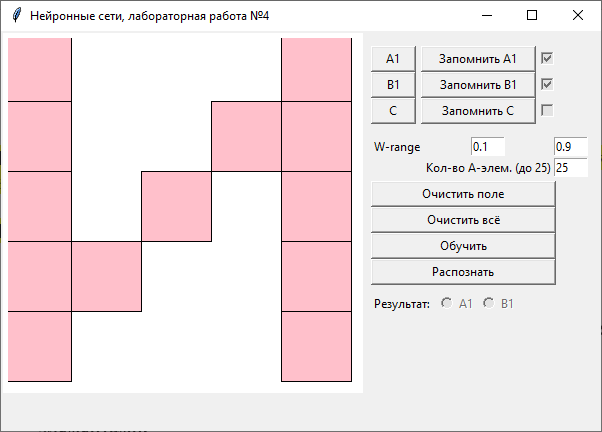
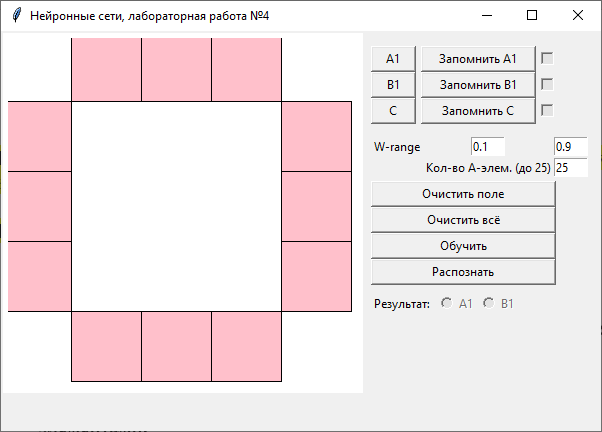
 исследовать возможности реализованного элементарного перцептрона по распознаванию изображений;

 исследовать возможности перцептрона по распознаванию нескольких классов изображений (использование нескольких выходных R-элементов).

***Выполнение***

Интерфейс в данной ЛР схож с интерфейсом программ для предыдущих ЛР.

***Буквы, используемые при выполнении ЛР:***

***Основные модули программы***

* mainwindow.py – модуль, отвечающий за отображение экранной формы;
* controller.py – модуль, отвечающий за связь экранной формы с модулем вычислений;
* calculatingperceptron.py – класс, выполняющий вычисления;
* perceptron.py – класс «Нейрон».

***Calculatingperceptron.py***

class Calculator:  
 def \_\_init\_\_(self):  
 self.symbols = [**'A1'**, **'A2'**, **'B1'**, **'B2'**, **'C1'**, **'C2'**, **'D1'**, **'D2'**, **'E1'**, **'E2'**]  
 self.letters\_list = None  
 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

***Perceptron.py***

class Perceptron:  
 def \_\_init\_\_(self, count\_s, count\_a, w\_range=(0.1, 0.9)):  
 self.SUB = str.maketrans(**"0123456789"**, **"₀₁₂₃₄₅₆₇₈₉"**)  
 self.count\_a = count\_a  
 self.count\_s = count\_s  
 self.s\_a\_matrix = [[random.uniform(w\_range[0], w\_range[1]) for \_ in range(count\_s)] for \_ in range(count\_a)]

self.a\_r\_matrix =[random.uniform(w\_range[0], w\_range[1]) for \_ in range(count\_a)]

def u\_input\_all\_a(self, lst):  
 res = []  
 for i in range(self.count\_a):  
 ts = []  
 for j in range(self.count\_s):  
 t = lst[j] \* self.s\_a\_matrix[i][j]  
 ts.append(round(t, 1))  
 r = numpy.sum(ts)  
 res.append(round(r, 1))  
 return res  
  
 def u\_input\_r(self, lst):  
 r = [lst[j] \* self.a\_r\_matrix[j] for j in range(self.count\_a)]  
 return numpy.sum(r)  
  
 def get\_excitatory\_neurons(self, letter):  
 array = [i for i in range(len(letter)) if letter[i] == 1]  
 return array  
  
 def delta\_w(self, ny, nak, n, flag):  
 return (ny - (nak \* ny / n)) if flag else -(nak\*ny/n)  
  
 def gamma\_correction(self, uvh\_array, nyt, element, teta):  
 current\_w = copy.copy(self.a\_r\_matrix)  
 uvr1 = uvh\_array[0]  
 uvr2 = uvh\_array[1]  
 tmp\_elem = []  
 tmp\_elem.append(element[0])  
 tmp\_elem.append([0 if v == 1 else 1 for v in element[1]])  
 summuvr = uvr1 + uvr2  
 s = **'UвхR = '** + str(uvh\_array) + **'**\n**'** s += **'Начальные веса: '** + str(current\_w) + **'**\n**'** s += **'Сумма UвхR = '** + str(summuvr) + **'**\n**'** print(s)  
 l = 0  
 flag = False  
 while not flag:  
 k = l + 1  
 print(**'t'** + str(k).translate(self.SUB) + **':**\n**'**)  
 activ = self.select\_element(teta, uvh\_array, l)  
 if activ is None:  
 print(**'канец'**)  
 return  
 self.calc\_Ilia(tmp\_elem[activ], self.a\_r\_matrix, nyt)  
 uvh\_array[activ] = self.u\_input\_r(element[activ])  
 s = **'Сумма UвхR = '** + str(numpy.sum(self.a\_r\_matrix)) + **'**\n**'** print(s)  
 l += 1  
 print(**'Перецептрон теперь умный и готов что-нибудь распознать!'**)  
  
 def select\_element(self, teta, uvh, l):  
 if uvh[1] < teta <= uvh[0]:  
 return None  
 if uvh[1] < teta:  
 return 0  
 elif uvh[0] >= teta:  
 return 1  
 else:  
 return 0 if l % 2 == 0 else 1  
  
 @staticmethod  
 def calc\_Ilia(list\_activion, list\_weight, nya):  
 active = []  
 passive = []  
 for i in range(len(list\_activion)):  
 if list\_activion[i] == 1:  
 active.append(i)  
 else:  
 passive.append(i)  
 dw = nya  
 eps = 1e-7  
 count = 0  
 for i in range(len(list\_weight)):  
 w = list\_weight[i]  
 if abs(w - 1) < eps and i in active:  
 active.remove(i)  
 continue  
 if abs(w - 0) < eps and i in passive:  
 passive.remove(i)  
 continue  
 max\_dw = 1 - w if i in active else w  
 if dw > max\_dw:  
 dw = max\_dw  
 if i in active:  
 count = len(passive)  
 else:  
 count = len(active)  
 if count != 0:  
 dw \*= len(list\_activion) / count  
  
 for index in active:  
 list\_weight[index] += dw \* len(passive) / len(list\_activion)  
  
 for index in passive:  
 list\_weight[index] -= dw \* len(active) / (len(list\_activion))

***Анализ работы программы***

Данные по анализу распознавания символов И и О приведены в таблице.

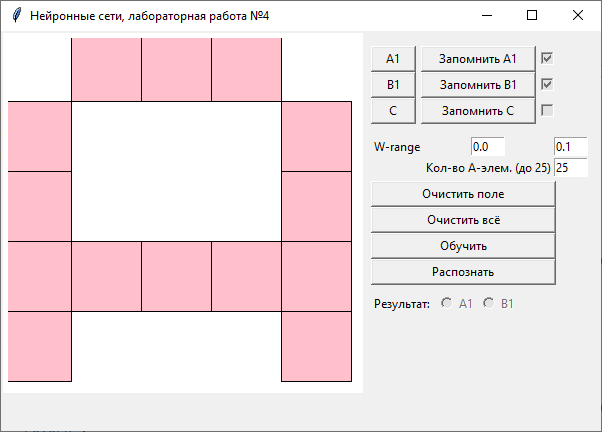
В ходе анализа было выяснено, что наилучшими свойствами к обучению обладает набор параметров №1.

Кроме опытов, приведенных в таблице, были проведены и другие серии опытов, в ходе которых стало ясно, что выбранное условие «все нейроны реагируют по-разному» в большинстве случаев позволяет обойтись без дополнительной настройки перцептрона. Предположительно, это связанно с тем, что выбранные символы довольно сильно отличаются.



Для дальнейшего анализа системы по условию «все нейроны реагируют по-разному или хотя бы одна треть и больше нейронов реагируют по-разному» было принято решение о замене одного из эталонных символов – вместо буквы О теперь используется буква А. Буквы И и О обладают большими различиями за счет чего не всегда требуется подкрепление.

Таблица с результатами обучения и изображение буквы А приведены ниже.



В таблицы приведены данные для 25 А-элементов с различными диапазонами, в которых случайным образом задавались весовые коэффициенты. Для каждого набора данных проводилось по три опыта – столбцы teta[a], teta[r] и итераций настройки соответственно.



В системе используются индивидуальные пороги для каждого А-элемента, однако приведено сразу среднее число для этих порогов для удобства читаемости (teta[a]).

Решение о выборе условия «индивидуальные пороги» было принято после ряда опытов, в которых обучение при индивидуальных порогах проходило лучше, нежели обучение при одном общем пороге. Общий порог не всегда удовлетворял требованию «все нейроны реагируют по-разному или хотя бы одна треть и больше нейронов реагируют по-разному».

*По данной таблице можно увидеть, что чем меньше диапазон выбора весовых коэффициентов, тем быстрее происходит коррекция с помощью гамма-подкрепления.*

В следующей таблице приведен результат использования при обучении 12 А-элементов. В среднем, для различных диапазонов весовых коэффициентов при таком количестве А-элементов среднее значение итераций для подкрепления *меньше*, чем при использовании 25 элементов.



Для распознавания было принято принять количество А-элементов равным 12, а диапазон выборки весов – от 0 до 0.1.