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 №4

по дисциплине «Нейронные сети»

тема: «Использование перцептрона»

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;

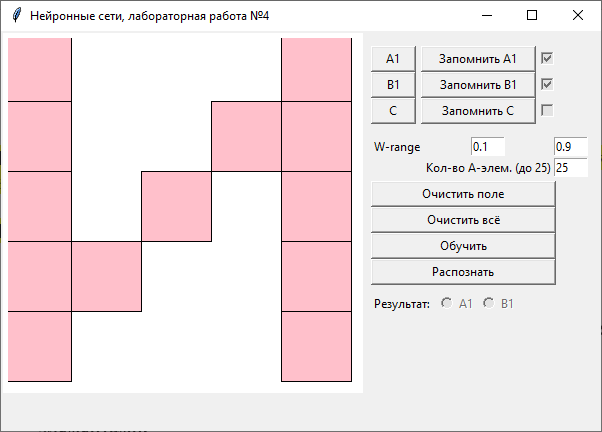
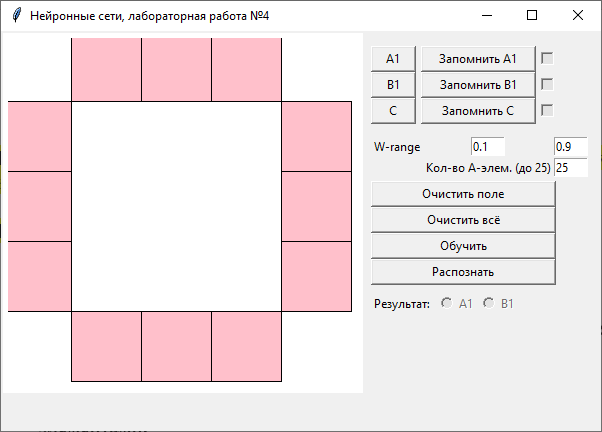
 исследовать возможности реализованного элементарного перцептрона по распознаванию изображений;

 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 Calculating:  
 def \_\_init\_\_(self):  
 self.SUB = str.maketrans(**"0123456789"**, **"₀₁₂₃₄₅₆₇₈₉"**)  
 self.ny = 0.2  
 self.letter\_a1 = None self.letter\_b1 = None self.letter\_c = None  
 self.letters\_array = None *#* self.m\_array = None  
 self.m\_c = None  
 self.input\_count = None  
 self.perceptron = None  
 self.excitatory\_neurons = None  
 self.a\_inputs\_signals = None *# UAвх* self.threshold\_array = None  
 self.threshold\_one = None  
 self.threshold\_all = None  
 self.r\_input\_signals = None *# URвх* self.r\_input\_signals\_info = None  
 self.r\_input\_signals\_teta = None *# URвх* self.flags = None  
 self.c\_s = None  
 self.ex\_as = None  
  
 def teaching(self, w\_range, count\_a):  
 print(**'**\n**>> ОБУЧЕНИЕ...'**)  
 self.input\_count = len(self.letter\_a1[0])  
 self.letters\_array = [self.letter\_a1, self.letter\_b1]  
 self.m\_array = self.make\_m\_binary()  
 print(**'**\n**А-элементов = '** + str(count\_a))  
 print(**'S-элементов = '** + str(self.input\_count))  
 print(**'Диапазон весов = '** + str(w\_range) + **'**\n**'**)  
 self.perceptron = Perceptron(self.input\_count, count\_a, w\_range)self.excitatory\_neurons = self.get\_excitatory\_neurons()  
 self.a\_inputs\_signals = self.get\_a\_elements\_input\_signals()  
 self.threshold\_info()  
 self.check\_threshold()  
 self.get\_r\_elements\_input\_signals()  
 self.gamma\_correct()  
  
 def recognition(self):  
 print(**'**\n**>> РАСПОЗНАВАНИЕ...'**)  
 self.m\_c = copy\_and\_insert\_one(self.letter\_c)  
 print(**'**\n**>> Схожесть буквы С с каждой из буков: '**)  
 self.\_\_calculate\_gemini()  
 print(**'**\n**>> Входной суммарный сигнал С (A-S), реакция на пороги: '**)  
 self.c\_s, self.ex\_as = self.\_\_calculate\_w\_as()  
 print(**'**\n**>> Входной суммарный сигнал С (A-R): '**)  
 self.c\_r, y = self.\_\_calculate\_w\_rs()  
 return y  
  
 def \_\_calculate\_w\_as(self):  
 s = self.perceptron.u\_input\_all\_a(self.m\_c)  
 tetas = self.threshold\_one  
 res1 = [1 if s[i] >= self.threshold\_one[i] else 0 for i in range(len(tetas))]  
 print(s)  
 print(res1)  
 return s, res1  
  
 def \_\_calculate\_w\_rs(self):  
 ex\_as = self.ex\_as  
 res = self.perceptron.u\_input\_r(ex\_as)  
 teta = self.r\_input\_signals\_teta[0]  
 print(res)  
 y = 1 if res >= teta else -1  
 print(**'Изображение похоже на класс: '** + str(y))  
 return res, y  
def \_\_calculate\_gemini(self):  
 a1 = self.\_\_help\_gemini(self.letter\_c, self.letter\_a1[0])  
 print(**'А1 + С: '** + str(a1))  
 b1 = self.\_\_help\_gemini(self.letter\_c, self.letter\_b1[0])  
 print(**'B1 + С: '** + str(b1))  
  
 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  
  
 def gamma\_correct(self):  
 if len(self.flags) == 0:  
 print(**'Неудачно попрошла настройка.**\n**'  
 'Нажмите повторно кнопку "Обучить" или введите перед этим новые параметры.'**)  
 else:  
 if not self.flags[0]:  
 print(**'Настройка перцептрона по гамма-системе: '**)  
 self.gamma\_correction()  
 else:  
 print(**'Перецептрон теперь умный и готов что-нибудь распознать!'**)  
  
 def threshold\_info(self):  
 self.threshold\_all = self.get\_threshold\_value\_for\_all()  
 self.threshold\_one = self.get\_threshold\_value\_for\_neurons()  
  
 def get\_threshold\_value\_for\_all(self):  
 print(**'**\n**Общее пороговое значение: '**)  
 tmp = numpy.ravel(self.a\_inputs\_signals)  
 s = **'min = '** mini = numpy.min(tmp)  
 s += str(mini) + **', max = '** maxi = numpy.max(tmp)  
 s += str(maxi)  
 res = round((mini + maxi) / 2, 1)  
 s += **'**\n**Пороговое значение, θ = '** + str(res)  
 print(s)  
 return res  
  
 def get\_threshold\_value\_for\_neurons(self):  
 print(**'**\n**Пороговые значения для каждого А-элмента: '**)  
 a = self.a\_inputs\_signals[0]  
 b = self.a\_inputs\_signals[1]  
 res = [(a[i] + b[i]) / 2 for i in range(len(a))]  
 [print(**'Пороговое значение для A'** + str(i).translate(self.SUB) + **', θ = '** + str(res[i])) for i in  
 range(len(res))]  
 return res  
  
 def get\_a\_elements\_input\_signals(self):  
 r = [self.perceptron.u\_input\_all\_a(letter[0]) for letter in self.letters\_array]  
 print(**'Сигналы на входах А-элементов: '**)  
 for l in r:  
 print(l)  
 return r  
  
 def get\_r\_elements\_input\_signals(self):  
 self.r\_input\_signals\_teta = []  
 self.r\_input\_signals\_info = []  
 self.r\_input\_signals = []  
 flags = []  
 for element in self.threshold\_array:  
 print(element[1] + **': '**)  
 res, teta, info = self.check\_r(element)  
 flag = True if info[0] == self.m\_array[0][1] and info[1] == self.m\_array[1][1] else False  
 print(**'настройка перцептрона не требуется**\n**'**) if flag else print(**'требуется настройка перцептрона!!**\n**'**)  
 flags.append(flag)  
 self.r\_input\_signals\_teta.append(teta)  
 self.r\_input\_signals\_info.append(info)  
 self.r\_input\_signals.append(res)  
 self.flags = flags  
  
 def check\_r(self, element):  
 res = [self.perceptron.u\_input\_r(el) for el in element[0]]  
 self.r\_input\_signals.append(res)  
 s = **'Вектор UвхR = '** + str(res)  
 teta = numpy.average(res)  
 s += **'. Пороговое значение, Rθ = '** + str(teta) + **'.**\n**'** s += **'Таким образом: '** + str(res[0])  
 s += **' >= '** + str(teta)  
 s += **' > '** + str(res[1]) + **'. '** info = [1 if res[i] >= teta else -1 for i in range(len(res))]  
 s += **'**\n**При подаче исходных сообщений имеем: '** + str(info)  
 s += **'. Изначально имеем: '** + str([self.m\_array[0][1], self.m\_array[1][1]]) + **' => '** print(s, end=**''**)  
 return (res, teta, info)  
  
 def get\_excitatory\_neurons(self):  
 s = **'**\n**Возбуждаемые нейроны, нумерация ведётся с 0:** \n**'** res = [(self.perceptron.get\_excitatory\_neurons(letter[0]), letter[1]) for letter in self.letters\_array]  
 for r in res:  
 s += str(r[1]) + **': '** s += str(r[0]) + **'**\n**'** print(s)  
 return res  
  
 def check\_threshold(self):  
 result = [(self.check\_threshold\_one(), **'Индивидуальные пороги'**)  
 ,(self.check\_threshold\_all(), **'Один порог для всех'**)]  
 if result[0] is None and result[1] is None:  
 print(**'Беда...'**)  
 self.threshold\_array = [el for el in result if el[0] is not None]  
  
 def check\_threshold\_one(self):  
 print(**'**\n**Индивидуальные пороги для А-элементов: '**)  
 a = self.a\_inputs\_signals[0]  
 b = self.a\_inputs\_signals[1]  
 res1 = [1 if a[i] >= self.threshold\_one[i] else 0 for i in range(len(self.threshold\_one))]  
 res2 = [1 if b[i] >= self.threshold\_one[i] else 0 for i in range(len(self.threshold\_one))]  
 print(str(self.letters\_array[0][1]) + **': '** + str(res1))  
 print(str(self.letters\_array[1][1]) + **': '** + str(res2))  
 return [res1, res2] if self.answer\_about\_teta(res1, res2, **'Индивидуальные пороги для всех нейронов'**) else None  
  
 def check\_threshold\_all(self):  
 print(**'**\n**Один порог для всех А-элементов: '**)  
 res1 = [1 if uin >= self.threshold\_all else 0 for uin in self.a\_inputs\_signals[0]]  
 res2 = [1 if uin >= self.threshold\_all else 0 for uin in self.a\_inputs\_signals[1]]  
 print(str(self.letters\_array[0][1]) + **': '** + str(res1))  
 print(str(self.letters\_array[1][1]) + **': '** + str(res2))  
 return [res1, res2] if self.answer\_about\_teta(res1, res2,  
 **'Один порог для всех нейронов'**) else None  
  
 def answer\_about\_teta(self, res1, res2, info=**''**):  
 res = numpy.sum([1 if res1[i] == res2[i] else 0 for i in range(len(res1))])  
 one\_tri = self.input\_count / 3  
 y = True if res == 0 else False  
 if y:  
 print(**'Все нейроны реагируют по-разному'**)  
 print(**'Условие типа** \"**'** + info + **'**\" **удовлетворяет требованиям**\n**'**) \  
 if y else print(**'Условие типа** \"**'** + info + **'**\" **\_НЕ\_ удовлетворяет требованиям**\n**'**)  
 return y  
 y = True if res > one\_tri else False  
 if y:  
 print(**'Не больше одной трети нейронов реагирует по-разному'**)  
 print(**'Условие типа** \"**'** + info + **'**\" **удовлетворяет требованиям**\n**'**) \  
 if y else print(**'Условие типа** \"**'** + info + **'**\" **\_НЕ\_ удовлетворяет требованиям**\n**'**)  
 return y  
  
 def make\_m\_binary(self):  
 result = []  
 for i in range(len(self.letters\_array)):  
 letter = copy\_and\_insert\_one(self.letters\_array[i][0])  
 y = -1 if i % 2 != 0 else 1  
 result.append((letter, y, self.letters\_array[i][1]))  
 print(result[i])  
 return result  
  
 def gamma\_correction(self):  
 r = copy.copy(self.r\_input\_signals[0])  
 ta = copy.copy(self.threshold\_array[0][0])  
 to = copy.copy(self.r\_input\_signals\_teta[0])  
 self.perceptron.gamma\_correction(r, 0.5, ta, to)

***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))

***Пример логов программы***

>> ОБУЧЕНИЕ...

([1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1], 1, 'И')

([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, 'О')

А-элементов = 12

S-элементов = 25

Диапазон весов = (0.0, 0.1)

Возбуждаемые нейроны, нумерация ведётся с 0:

И: [0, 4, 5, 8, 9, 10, 12, 14, 15, 16, 19, 20, 24]

О: [1, 2, 3, 5, 9, 10, 14, 15, 16, 17, 18, 19, 20, 24]

Сигналы на входах А-элементов:

[0.6530480790874087, 0.7188773730097049, 0.40234759621530075, 0.8190797004749789, 0.6002157163392698, 0.521163069488662, 0.8345561097959786, 0.5484420796251207, 0.7976232436222288, 0.6706788566571074, 0.5521578943395121, 0.6715984959823844]

[0.7051963701093606, 0.6717579545040563, 0.4536327172443739, 0.7795692013116317, 0.6380164096849184, 0.573649886795606, 0.880598626104763, 0.574831335132063, 0.8136405632460229, 0.7321591187726453, 0.5374383481537289, 0.7476817652146996]

**Общее пороговое значение:**

min = 0.40234759621530075, max = 0.880598626104763

Пороговое значение, θ = 0.6

**Пороговые значения для каждого А-элмента:**

Пороговое значение для A₀, θ = 0.6791222245983847

Пороговое значение для A₁, θ = 0.6953176637568805

Пороговое значение для A₂, θ = 0.4279901567298373

Пороговое значение для A₃, θ = 0.7993244508933053

Пороговое значение для A₄, θ = 0.6191160630120941

Пороговое значение для A₅, θ = 0.547406478142134

Пороговое значение для A₆, θ = 0.8575773679503709

Пороговое значение для A₇, θ = 0.5616367073785918

Пороговое значение для A₈, θ = 0.8056319034341258

Пороговое значение для A₉, θ = 0.7014189877148763

Пороговое значение для A₁₀, θ = 0.5447981212466205

Пороговое значение для A₁₁, θ = 0.709640130598542

Индивидуальные пороги для А-элементов:

И: [0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0]

О: [1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1]

Все нейроны реагируют по-разному

Условие типа "Индивидуальные пороги для всех нейронов" удовлетворяет требованиям

Один порог для всех А-элементов:

И: [1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1]

О: [1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1]

Не больше одной трети нейронов реагирует по-разному

Условие типа "Один порог для всех нейронов" удовлетворяет требованиям

Индивидуальные пороги:

Вектор UвхR = [0.19581965121423955, 0.501787444092439]. Пороговое значение, Rθ = 0.3488035476533393.

Таким образом: 0.19581965121423955 >= 0.3488035476533393 > 0.501787444092439.

При подаче исходных сообщений имеем: [-1, 1]. Изначально имеем: [1, -1] => требуется настройка перцептрона!!

Один порог для всех:

Вектор UвхR = [0.4841894561327548, 0.4841894561327548]. Пороговое значение, Rθ = 0.4841894561327548.

Таким образом: 0.4841894561327548 >= 0.4841894561327548 > 0.4841894561327548.

При подаче исходных сообщений имеем: [1, 1]. Изначально имеем: [1, -1] => требуется настройка перцептрона!!

Настройка перцептрона по гамма-системе:

UвхR = [0.19581965121423955, 0.501787444092439]

Начальные веса: [0.055919136736593135, 0.07457259045698676, 0.03581990867930843, 0.09802448975893216, 0.06971228418971027, 0.09483519946096469, 0.0680937962259285, 0.05953996003533003, 0.021932644666535286, 0.06683346707433872, 0.023222570998320646, 0.02910104702373]

Сумма UвхR = 0.6976070953066786

t₁: Сумма UвхR = 0.6976070953066786

t₂: Сумма UвхR = 0.6976070953066786

t₃: Сумма UвхR = 0.6976070953066786

>> РАСПОЗНАВАНИЕ...

>> Схожесть буквы С с каждой из буков:

А1 + С: 25

B1 + С: 16

>> Входной суммарный сигнал С (A-S), реакция на пороги:

[0.6189064293015105, 0.47389030613066485, 0.41593222057215623, 0.8595922126370474, 0.561540663169372, 0.5489795882730034, 0.7433151351991965, 0.49493477470332836, 0.7292495270739174, 0.6773073579035905, 0.5717155341686095, 0.7371886716309932]

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

>> Входной суммарный сигнал С (A-R):

0.356808560432071

Изображение похоже на класс: 1

***Анализ работы программы***

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

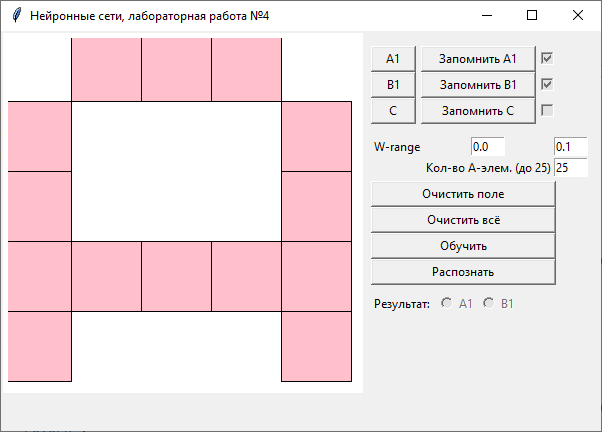
В ходе анализа было выяснено, что наилучшими свойствами к обучению обладает набор параметров №1.

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.*

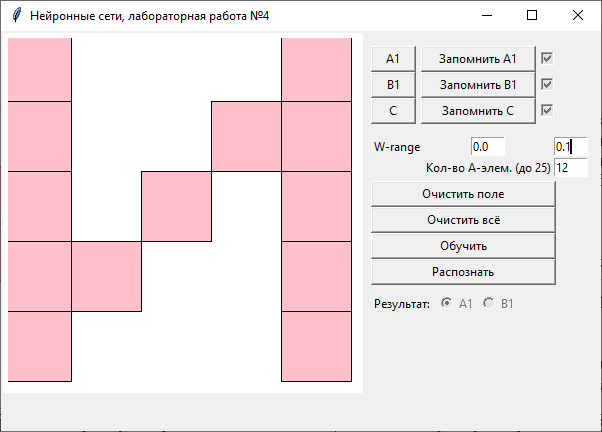
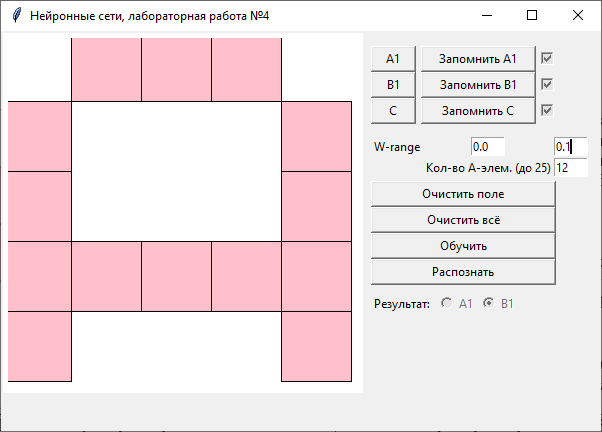
В следующей таблице приведен результат использования при обучении 12 А-элементов. В среднем, для различных диапазонов весовых коэффициентов при таком количестве А-элементов среднее значение итераций для подкрепления *меньше*, чем при использовании 25 элементов.



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

***Случай №1.*** Показ эталонных изображений.

Как видно в блоке «Результат», персептрон верно реагирует на изображения, предложенные ему.

|  |  |
| --- | --- |
| Сигналы на входах А-элементов:  И: [0.5776254633216955, 0.6033679366301496, 0.6837771283226337, 0.6403857462163514, 0.6783881230347212, 0.87346614614013, 0.8646553601312735, 0.6591441962847482, 0.562058440565095, 0.6448512854734719, 0.7511486484066883, 0.4385497812736063]  А: [0.6816708061969659, 0.6222809799247443, 0.7229854457873526, 0.7361012137114256, 0.5876568395059226, 0.9504608997526588, 0.8123764005003641, 0.8691190177341906, 0.7622429172021761, 0.716676371999411, 0.7806757728674085, 0.47020585186167535] | |
| При подаче И  [0.5776254633216955, 0.6033679366301496, 0.6837771283226337, 0.6403857462163514, 0.6783881230347212, 0.87346614614013, 0.8646553601312735, 0.6591441962847482, 0.562058440565095, 0.6448512854734719, 0.7511486484066883, 0.4385497812736063] | При подаче А  [0.6816708061969659, 0.6222809799247443, 0.7229854457873526, 0.7361012137114256, 0.5876568395059226, 0.9504608997526588, 0.8123764005003641, 0.8691190177341906, 0.7622429172021761, 0.716676371999411, 0.7806757728674085, 0.47020585186167535] |

Сигналы совпадают, расчёт происходит верно.

|  |  |
| --- | --- |
| ***Случай №2*** | |
|  | **>> Схожесть буквы С с каждой из буков:**  **А1 + С: 20**  **B1 + С: 15**  >> Входной суммарный сигнал С (A-S), реакция на пороги:  teta = [0.9201364005030404, 0.7597363608322631, 0.6991753991196188, 0.7201138659296412, 0.7963892741126439, 0.7003332733640998, 0.529175535387816, 0.47217670410533297, 0.621121587697562, 0.8409238617506177, 0.5925312095497847, 0.6776580379217686]  сигнал = [1.1553362933704305, 0.8445766016684463, 0.9330442133472475, 0.9122032336044188, 0.9802023824809305, 0.923993658861064, 0.7557811224824591, 0.6944430478888189, 0.8007026289770939, 1.043955545365957, 0.7593860535987375, 0.8733836889600957]  >> Входной суммарный сигнал С (A-R):  0.5412506001999333  teta = 0.27062530009996666  сигнал = 0.5412506001999333  **Изображение похоже на класс: 1** |
| ***Случай №3*** |  |
|  | **>> Схожесть буквы С с каждой из буков:**  **А1 + С: 19**  **B1 + С: 16**  >> Входной суммарный сигнал С (A-S), реакция на пороги:  teta = [0.9201364005030404, 0.7597363608322631, 0.6991753991196188, 0.7201138659296412, 0.7963892741126439, 0.7003332733640998, 0.529175535387816, 0.47217670410533297, 0.621121587697562, 0.8409238617506177, 0.5925312095497847, 0.6776580379217686]  сигнал = [1.2108685664551684, 0.8985822146975078, 0.9778547387675829, 0.9606590012860439, 1.039665580742466, 1.001293255936673, 0.7606899151524897, 0.7546144030161402, 0.8914623334606304, 1.1286724870443148, 0.8458022080167938, 0.9480843275417212]  >> Входной суммарный сигнал С (A-R):  0.5412506001999333  teta = 0.27062530009996666  сигнал = 0.5412506001999333  **Изображение похоже на класс: 1** |
| ***Случай №4*** |  |
|  | **>> Схожесть буквы С с каждой из буков:**  **А1 + С: 18**  **B1 + С: 17**  >> Входной суммарный сигнал С (A-S), реакция на пороги:  teta = [0.5105829602552442, 0.8630499866062118, 0.8814359990484608, 0.7340799412255841, 0.5622863977687558, 0.7954014122647219, 0.6395747682321505, 0.566125066993401, 0.7264745551296775, 0.5088379797250215, 0.6513556235966685, 0.6799723082390876]  сигнал = [0.632579971764918, 1.1134182172583624, 1.1421491772996926, 0.9364620496451341, 0.8165295491442677, 1.01846058202175, 0.895581634751869, 0.8084483477096831, 0.9501144764731637, 0.6649198396053644, 0.8904242904227466, 0.8692615188621688]  >> Входной суммарный сигнал С (A-R):  0.6797311318344055  teta = 0.3398655659172028  сигнал = 0.6797311318344055  **Изображение похоже на класс: 1** |

***Пример последовательного изменения пикселей одной буквы на другую:***

|  |  |
| --- | --- |
|  | А1 + С: 25  B1 + С: 16  >> Входной суммарный сигнал С (A-R):  0.8605697844883364  teta = 0.696925844121713  Изображение похоже на класс: 1 |
|  | >> Схожесть буквы С с каждой из буков:  А1 + С: 24  B1 + С: 17  >> Входной суммарный сигнал С (A-R):  1.3938516882434262  teta = 0.696925844121713  Изображение похоже на класс: 1 |
|  | >> Схожесть буквы С с каждой из буков:  А1 + С: 23  B1 + С: 18  >> Входной суммарный сигнал С (A-R):  1.1280643170054114  teta = 0.696925844121713  сигнал = 1.3938516882434262  Изображение похоже на класс: 1 |
|  | >> Схожесть буквы С с каждой из буков:  А1 + С: 22  B1 + С: 19  >> Входной суммарный сигнал С (A-R):  0.9938516882434262  teta = 0.696925844121713  Изображение похоже на класс: 1 |
|  | >> Схожесть буквы С с каждой из буков:  А1 + С: 21  B1 + С: 20  >> Входной суммарный сигнал С (A-R):  0.8538516882434262  teta = 0.696925844121713  Изображение похоже на класс: 1 |
|  | >> Схожесть буквы С с каждой из буков:  А1 + С: 20  B1 + С: 21  >> Входной суммарный сигнал С (A-R):  0.65919193332940663  teta = 0.696925844121713  сигнал = 0.41919193332940663  Изображение похоже на класс: -1 |

В данной таблице можно проследить, как уменьшаются суммарные входные сигналы для введенной буквы, постепенно приближаясь к значению teta, а после, переступив через него, получается новый результат.

***Вывод:***

Таким образом, можно сказать, что подбор параметров персептрона является важной частью при обучении. В таблицах, приведенных ранее, было проведено примерно 70 опытов по подбору значений, кроме них проводились и другие. Также были поставлены эксперимент по обучению с разными правилами (1/3 нейронов, все нейроны реагируют по-разному), или выбором пороговых значений для А-элементов (индивидуальные пороги или общий). Было выяснено, что при подборе индивидуальных порогов обучение проходит лучше, а выбор общего порога не всегда позволяет ответить на вопрос о том, как же реагируют нейроны (и в каком количестве). Был разработан метод позволяющий проводить подкрепление персептрона по гамма-системе, однако, в ходе опытов были выявлены случаи, при которых подкрепление не нужно. В ходе остальных экспериментов, данные которых не были зафиксированы, также была получена информация о том, что неверный подбор параметров может значительно ухудшить процесс корректировки весов, а в наихудших случаях – работа алгоритма может зациклиться. В таких случаях принималось решение о переобучении персептрона.