Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный университет”

Кафедра ИИТ

Лабораторная работа №5

По дисциплине: “МиАПР”

Тема: “ Нелинейные ИНС в задачах распознавания образов”

Выполнил:

Студент 2-го курса

Группы ПО-7

Рекун И.В.

Проверил:

Крощенко А.А.

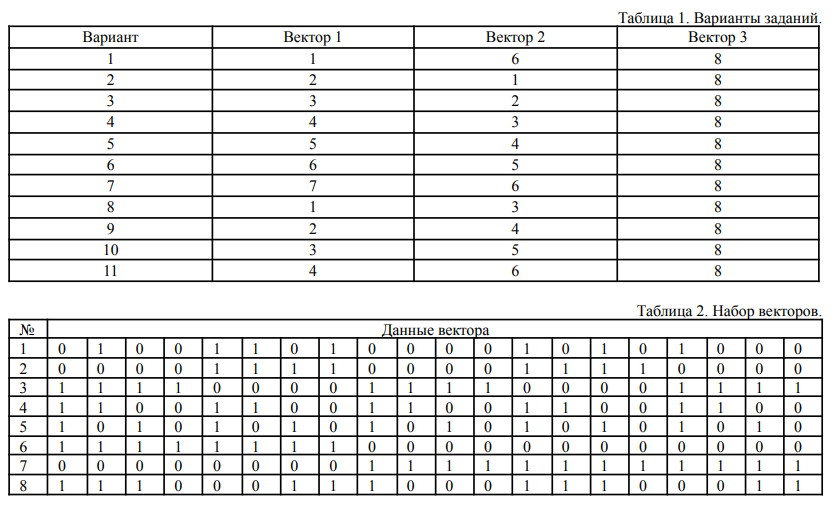
Брест, 2021

**Цель работы:** изучить обучение и функционирование нелинейной ИНС при решении задач распознавания образов.

***Задание.***

Написать на любом ЯВУ программу моделирования нелинейной ИНС для распознавания образов (рекомендуется использовать сигмоидную функцию). Количество НЭ в скрытом слое взять согласно варианту работы №3 (можно варьировать, если сеть не обучается или некорректно функционирует).

Провести исследование полученной модели. При этом на вход сети необходимо подавать искаженные образы, в которых инвертированы некоторые биты. Критерий эффективности процесса распознавания - максимальное кодовое расстояние (количество искаженных битов) между исходным и поданным образом.



**Код программы**:

import random

import math

vector\_3 = [1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1]

vector\_5 = [1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0]

vector\_8 = [1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1]

all\_vectors = [vector\_3, vector\_5, vector\_8]

alpha = 0.2

MIN\_ERROR = 1e-4

input\_neuron\_number = 6

hidden\_neuron\_number = 2

output\_neuron\_number = 1

weight\_i\_j = [[random.uniform(-0.1, 0.1) for \_ in range(hidden\_neuron\_number)] for \_ in range(input\_neuron\_number - 1)]

weight\_i\_k = [[random.uniform(-0.1, 0.1) for \_ in range(hidden\_neuron\_number)] for \_ in range(output\_neuron\_number)]

T\_hidden = [random.uniform(-0.5, 0.5) for \_ in range(hidden\_neuron\_number)]

T\_out = [random.uniform(-0.5, 0.5) for \_ in range(output\_neuron\_number)]

def sigmoid(S):

return 1 / (1 + math.exp(-S))

def calculate\_hidden(y):

s\_hidden = []

for j in range(hidden\_neuron\_number):

value = 0

for i in range(input\_neuron\_number - 1):

value += y[i] \* weight\_i\_j[i][j]

value -= T\_hidden[j]

s\_hidden.append(sigmoid(value))

return s\_hidden

def calculate\_out(y\_hidden):

s\_out = []

for j in range(output\_neuron\_number):

value = 0

for i in range(hidden\_neuron\_number):

value += y\_hidden[i] \* weight\_i\_k[j][i]

value -= T\_out[j]

s\_out.append(sigmoid(value))

return s\_out

def Wjk\_change(y\_hidden, y\_out, error):

global T\_out

for j in range(output\_neuron\_number):

for i in range(hidden\_neuron\_number):

weight\_i\_k[j][i] -= alpha \* error[j] \* y\_out[j] \* (1 - y\_out[j]) \* y\_hidden[i]

T\_out[j] += error[j] \* alpha \* y\_out[j] \* (1 - y\_out[j])

def Wij\_change(y\_hidden, hidden\_error, y):

for j in range(hidden\_neuron\_number):

for i in range(input\_neuron\_number - 1):

weight\_i\_j[i][j] -= alpha \* hidden\_error[j] \* y[i] \* y\_hidden[j] \* (1 - y\_hidden[j])

T\_hidden[j] += alpha \* hidden\_error[j] \* y\_hidden[j] \* (1 - y\_hidden[j])

def main():

hidden\_error = [0 for i in range(hidden\_neuron\_number)]

reference = [0 for i in range(output\_neuron\_number)]

error\_arr = [0 for i in range(output\_neuron\_number)]

iter = 40

epoch = 0

error = 1

while error > MIN\_ERROR:

error = 0

for N in range(output\_neuron\_number):

reference[N] = 1

for i in range(iter):

y = all\_vectors[N]

y\_hidden = calculate\_hidden(y)

y\_out = calculate\_out(y\_hidden)

for i in range(output\_neuron\_number):

error\_arr[i] = y\_out[i] - reference[i]

for j in range(hidden\_neuron\_number):

for k in range(output\_neuron\_number):

hidden\_error[j] += error\_arr[k] \* y\_out[k] \* (1 - y\_out[k]) \* weight\_i\_k[k][j]

Wjk\_change(y\_hidden, y\_out, error\_arr)

Wij\_change(y\_hidden, hidden\_error, y)

error += error\_arr[N] \*\* 2

error /= 2

epoch += 1

print(f'Number epoch : {epoch}')

for i in range(len(all\_vectors)):

input = all\_vectors[i]

print("Vector :", i + 1, end=" : ")

for j in range(len(vector\_3)):

print(input[j], end='')

print("\nResult : ", end='')

hidden\_early = calculate\_hidden(input)

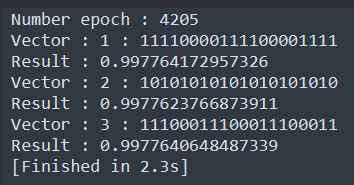
values = calculate\_out(hidden\_early)

print(values[0])

if \_\_name\_\_ == '\_\_main\_\_':

main()

**Результат выполнения программы:**



**Вывод:** Научился реализовывать нелинейную ИНС для распознавания образов.