Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По дисциплине: «МиАПР»

Тема: «Нелинейные ИНС в задачах распознавания образов»

Выполнил:

Студент 2 курса

Группы ПО-7(2)

Фурсевич Д.С.

Проверил:

Крощенко А.А.

2021

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

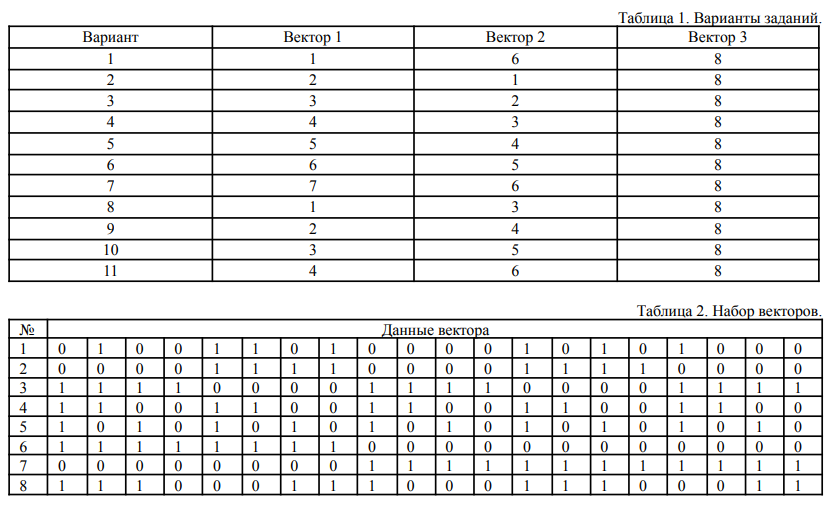
**Ход работы**

Вариант 11

**Задание:**

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

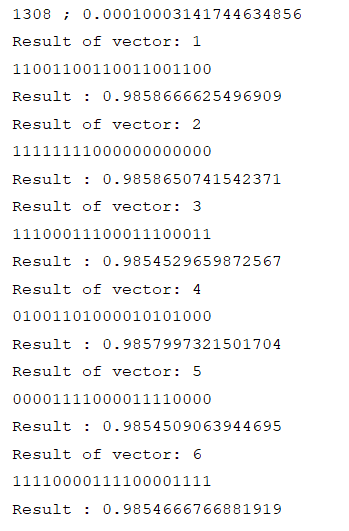
Провести исследование полученной модели. При этом на вход сети необходимо подавать искаженные образы, в которых инвертированы некоторые биты. Критерий эффективности процесса распознавания - максимальное кодовое расстояние (количество искаженных битов) между исходным и поданным образом.



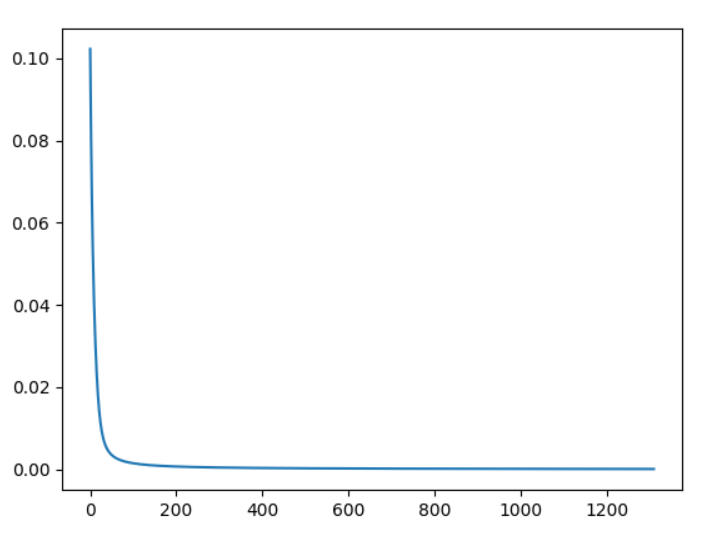
**Код программы:**

import random  
import math  
import matplotlib.pyplot as plt  
  
step = 0.5  
mistake\_min = 1e-4  
num\_in = 8 # количество входов инс  
num\_hid = 3 # количество нэ в скрытом слое  
num\_out = 1  
w\_ij = [[random.uniform(-0.1, 0.1) for i in range(num\_hid)] for j in range(num\_in - 1)]  
w\_jk = [[random.uniform(-0.1, 0.1) for \_ in range(num\_hid)] for j in range(num\_out)]  
thresholdValue\_j = [random.uniform(-0.5, 0.5) for \_ in range(num\_hid)]  
thresholdValue\_k = [random.uniform(-0.5, 0.5) for \_ in range(num\_out)]  
  
vector0 = [1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0]  
vector1 = [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]  
vector2 = [1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1]  
vector3 = [0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0]  
vector4 = [0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0]  
vector5 = [1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1]  
Vector = [vector0, vector1, vector2, vector3, vector4, vector5]  
  
  
def sigmoidActivationFunction(s):  
 return (1 / (1 + math.exp(-1 \* s)))  
  
  
def hiddenLayer\_Sj(y):  
 S\_j = []  
 for i in range(num\_hid):  
 value = 0  
 for j in range(num\_in - 1):  
 value += y[j] \* w\_ij[j][i]  
 value -= thresholdValue\_j[i]  
 S\_j.append(sigmoidActivationFunction(value))  
 return S\_j  
  
  
def outputLayer\_Sk(y\_j):  
 S\_k = []  
 for j in range(num\_out):  
 value = 0  
 for i in range(num\_hid):  
 value += y\_j[i] \* w\_jk[j][i]  
 value -= thresholdValue\_k[j]  
 S\_k.append(sigmoidActivationFunction(value))  
 return S\_k  
  
  
def change\_w\_jk(y\_j, y\_k, mistake):  
 global thresholdValue\_k  
 for j in range(num\_out):  
 for i in range(num\_hid):  
 w\_jk[j][i] -= step \* mistake[j] \* y\_k[j] \* (1 - y\_k[j]) \* y\_j[i]  
 thresholdValue\_k[j] += mistake[j] \* step \* y\_k[j] \* (1 - y\_k[j])  
  
  
def change\_w\_ij(y\_j, mistake\_hid, y):  
 for j in range(num\_hid):  
 for i in range(num\_in - 1):  
 w\_ij[i][j] -= step \* mistake\_hid[j] \* y[i] \* y\_j[j] \* (1 - y\_j[j])  
 thresholdValue\_j[j] += step \* mistake\_hid[j] \* y\_j[j] \* (1 - y\_j[j])  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 x\_points, y\_points = [], []  
 mistakes = [0] \* num\_out  
 etalons = [0] \* num\_out  
 mistake\_hid = [0] \* num\_hid  
 iter = 1  
 n = 0  
 while True:  
 error = 0  
 for N in range(num\_out):  
 y, y\_j, y\_k = [], [], []  
 etalons[N] = 1  
 for i in range(iter):  
 y = Vector[N]  
 y\_j = hiddenLayer\_Sj(y)  
 y\_k = outputLayer\_Sk(y\_j)  
 for i in range(num\_out):  
 mistakes[i] = y\_k[i] - etalons[i]  
 for j in range(num\_hid):  
 for k in range(num\_out):  
 mistake\_hid[j] += mistakes[k] \* y\_k[k] \* (1 - y\_k[k]) \* w\_jk[k][j]  
 change\_w\_jk(y\_j, y\_k, mistakes)  
 change\_w\_ij(y\_j, mistake\_hid, y)  
 error += mistakes[N] \*\* 2  
 error /= 2  
 y\_points.append(error)  
 x\_points.append(n)  
  
 if abs(error) < mistake\_min:  
 print("finaly: ", error)  
 break  
 print(n, ';', error)  
 n += 1  
 plt.plot(x\_points, y\_points)  
 plt.show()  
  
 for i in range(6):  
 input = Vector[i]  
 print("Result of vector: ", i + 1)  
 for j in range(20):  
 print(input[j], end=' ')  
 print("\nResult : ")  
 hid\_pred = hiddenLayer\_Sj(input)  
 Values = outputLayer\_Sk(hid\_pred)  
 print(Values[0])

**Результат выполнения программы:**



**График изменения ошибки в зависимости от итерации:**



**Вывод:** изучила обучение и функционирование нелинейной ИНС при решении задач распознавания образов.