Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По дисциплине: «Нелинейные ИНС в задачах распознавания образов»

Вариант 11

Выполнил:

Студент 2-го курса

Группы ПО-7

Лобан К.Ю.

Проверил:

Крощенко А.А.

Брест 2021

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

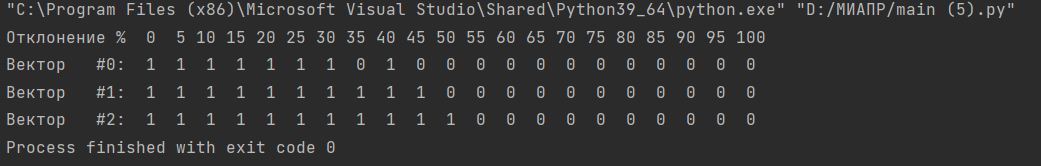
**Задание**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 8 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |

**Код программы:**

import numpy as np  
  
  
class NN:  
 *"""Класс нейронки"""* def \_\_init\_\_(self):  
 *"""Нейронная сеть с количеством нейронов: 8-3-1"""* self.wx = np.random.normal(-0.5, 0.5, (20, 3))  
 self.th = np.random.normal(-0.5, 0.5, 3)  
 self.wh = np.random.normal(-0.5, 0.5, (3, 3))  
 self.ty = np.random.normal(-0.5, 0.5, 3)  
 # Задание весов и порогов нормальным распределением нужных длин  
  
 def go(self, x):  
 *"""Прохождение всей нейронки"""* self.x = x # Входные параметры (1, 20)  
 self.sh = np.dot(self.x, self.wx) - self.th # Вектор сумм ((1, 20) x (20, 3) - (1, 3) = (1, 3))  
 self.h = 1.0 / (1.0 + np.exp(-self.sh)) # Активация (1, 3)  
 self.sy = np.dot(self.h, self.wh) - self.ty # Вектор сумм ((1, 3) x (3, 3) - (1, 3) = (1, 3))  
 self.y = self.sy # Активация и получение выхода нейронки (1, 3)  
 return self.y # (1, 3)  
  
 def back\_propagation(self, error\_y, alpha):  
 *"""Обратное распространение ошибки с изменением весов, порога"""* error\_h = np.dot(error\_y, self.wh.transpose())  
 # Ошибка для скрытого слоя ((1, 3) x (3, 3).T = (1, 3))  
  
 gamma\_y = alpha \* error\_y  
 self.wh -= np.dot(self.h.reshape(-1, 1), gamma\_y.reshape(1, -1))  
 self.ty += gamma\_y  
  
 gamma\_h = alpha \* error\_h \* self.y \* (1.0 - self.y)  
 self.wx -= np.dot(self.x.reshape(-1, 1), gamma\_h.reshape(1, -1))  
 self.th += gamma\_h  
  
 def learn(self, x, e, alpha):  
 *"""Обучение наборами"""* square\_error = 0 # Среднеквадратичная ошибка  
 for i in range(len(e)):  
 y = self.go(x[i]) # Прогон выборки  
 error = y - e[i] # Ошибка нейронной сети  
 square\_error += np.sum(error \*\* 2 / 2) # Суммирование среднеквадратичной ошибки  
 self.back\_propagation(error, alpha) # Обратное распространение  
 return square\_error  
  
  
def main():  
 vectors = np.array(  
 [  
 [1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0],  
 [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],  
 [1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1]  
 ]  
 )  
 e = np.array(  
 [  
 [1, 0, 0],  
 [0, 1, 0],  
 [0, 0, 1]  
 ]  
 )  
  
 nn = NN()  
 alpha = 0.06  
  
 for i in range(30\_000):  
 error = nn.learn(vectors, e, alpha)  
 # print(f"Square error {i: 5}: {error: .8f}") # Обучение с выводом ошибки  
 if error <= 1e-3:  
 break  
  
 print(f"Отклонение % 0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100", end="")  
 for i in range(3):  
 print(f"\nВектор #{i}:", end="")  
 y = nn.go(vectors[i]) # Проверка без отклонений  
 result = y.argmax() == i # Результатом считается тот выходной нейрон, где большее значение  
 print(f"{result: 3}", end="")  
  
 for j in np.random.choice(20, 20, replace=False): # Случайный j до 20 без повторений  
 vectors[i][j] ^= 1 # Смена бита  
 y = nn.go(vectors[i])  
 result = y.argmax() == i  
 print(f"{result: 3}", end="")  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

**Результат:**

****

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