Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №3**

По дисциплине: «Аппаратное обеспечение интеллектуальных систем»

Тема: «Моделирование ассоциативная памяти при помощи нейронных сетей»

**Выполнил:**

Студент 2 курса

Группы ИИ-21

Кирилович А. А.

**Проверил:**

Михно Е.В.

Брест 2023

Цель работы: изучить обучение и функционирование релаксационных ИНС в качестве ассоциативной памяти при решении задач распознавания образов.

Порядок работы:

1. Изучить теоретические сведения.
2. Написать на любом ЯВУ программу моделирования ИНС для распознавания векторов согласно варианту. ИНС содержит n нейронных элементов в первом слое и m во втором слое. Если n меньше размерности вектора, тогда из вектора использовать только первые n элементов.
3. 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вариант | n | m | № векторов | Модель ИНС |
| 4 | 13 | 13 | 3,8,4,9 | Хопфилда |

import numpy as np

class Hopfield():

def \_\_init\_\_(self, Y, n):

self.n = n

Y = self.prepareY(Y)

self.L = Y.shape[0]

self.I = np.eye(Y.shape[1])

self.W = (2 \* Y - 1).T @ (2 \* Y - 1) - self.L \* self.I

self.func = {

'sync' : self.\_sync,

'async' : self.\_async

}

def prepareY(self, Y):

return Y[:self.n]

def nextY(self, Y, W):

return np.where(Y @ W <= 0, 0, 1)

def \_sync(self, Y):

Y = self.prepareY(self, Y)

prevY = Y

for \_ in range(10):

Y = self.nextY(Y, self.W)

if np.allclose(Y, self.nextY(Y, self.W), atol=0):

return Y

if np.allclose(prevY, self.nextY(Y, self.W), atol=0):

return

prevY = Y

def \_async(self, Y):

Y = self.prepareY(self, Y)

for \_ in range(10):

list\_idx = list(range(Y.shape[1]))

np.random.shuffle(list\_idx)

for idx in list\_idx:

Y[0, idx] = self.nextY(Y, self.W[:, idx])

if np.allclose(Y, self.nextY(Y, self.W), atol=0):

return Y

model = Hopfield(np.array([

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

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

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

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

]), 13)

Y\_noise = np.array([[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1]])

print(model.func['async'](Y\_noise))

При повреждении примерно половины значений практически во всех случаях нейросеть восстанавливает исходный массив.

Вывод: изучил обучение и функционирование релаксационных ИНС в качестве ассоциативной памяти при решении задач распознавания образов.