Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный университет”

Кафедра ИИТ

Лабораторная работа №3

По дисциплине: “МиАПР”

Тема: “ Нелинейные ИНС в задачах прогнозирования.”

Выполнил:

Студент 2-го курса

Группы ПО-7

Степанюк А.С.

Проверил:

Крощенко А.А.

Брест, 2021

**Цель работы:** Изучить обучение и функционирование линейной ИНС с применением адаптивного шага.

***Задание.***

Написать на любом ЯВУ программу моделирования прогнозирующей нелинейной ИНС. Для тестирования использовать функцию

** .

**Main.py**

import numpy as np

def f(x):

return 0.1 \* np.cos(0.3 \* x) + 0.08 \* np.sin(0.3 \* x)

def predict\_sets(begin, lenght, count, step):

base\_array = np.arange(count + lenght)

input = np.zeros((count, lenght))

for i in range(count):

input[i] = base\_array[i: lenght + i]

etalons = base\_array[lenght: lenght + count]

input = input \* step + begin

etalons = etalons \* step + begin

return f(input), f(etalons)

class NeuralNet :

def \_\_init\_\_(self):

self.weights\_x = np.random.normal(-1, 1, (10, 4))

self.threshold\_h = np.random.normal(-1, 1, 4)

self.weights\_h = np.random.normal(-1, 1, (4, 1))

self.threshold\_y = np.random.normal(-1, 1, 1)

def go(self, x):

self.input = x

self.sum\_h = np.dot(self.input, self.weights\_x) - self.threshold\_h

self.hidden = 1.0 / (1.0 + np.exp(-self.sum\_h))

self.sum\_y = np.dot(self.hidden, self.weights\_h) - self.threshold\_y

self.output = self.sum\_y

return self.output

def changing(self, error\_y, alpha):

error\_h = np.dot(error\_y, self.weights\_h.transpose())

gamma\_y = alpha \* error\_y

self.weights\_h -= np.dot(self.hidden.reshape(-1, 1), gamma\_y.reshape(1, -1))

self.threshold\_y += gamma\_y

gamma\_h = alpha \* error\_h \* self.output \* (1.0 - self.output)

self.weights\_x -= np.dot(self.input.reshape(-1, 1), gamma\_h.reshape(1, -1))

self.threshold\_h += gamma\_h

def learning(self, x, e, alpha):

square\_error = 0

for i in range(len(e)):

output = self.go(x[i])

error = output - e[i]

square\_error += error \* \*2 / 2

self.changing(error, alpha)

return square\_error

def testing(self, x, e):

print(" эталонное значение выходное значение среднеквадратичная ошибка")

for i in range(len(e)):

output = self.go(x[i])[0]

delta = output - e[i]

square\_error = delta \* \*2 / 2

print(f"{e[i]: 19.17f}{output: 24.17f}{square\_error: 26}")

neural\_net = NeuralNet()

learn\_input, learn\_output = predict\_sets(0, 10, 30, 0.1)

speed = 0.01

for i in range(30000):

error = neural\_net.learning(learn\_input, learn\_output, speed)

if i % 1000 == 0:

print(f"{i}: {error}")

test\_x, test\_e = predict\_sets(3, 10, 15, 0.1)

neural\_net.testing(test\_x, test\_e)

**Результаты обучения:**

**0: [67.50993124]**

**1000: [0.00054686]**

**2000: [0.00054294]**

**3000: [0.00053915]**

**4000: [0.00053549]**

**5000: [0.00053194]**

**6000: [0.00052851]**

**7000: [0.00052519]**

**8000: [0.00052197]**

**9000: [0.00051884]**

**10000: [0.00051582]**

**11000: [0.00051289]**

**12000: [0.00051004]**

**13000: [0.00050728]**

**14000: [0.0005046]**

**15000: [0.000502]**

**16000: [0.00049947]**

**17000: [0.00049702]**

**18000: [0.00049463]**

**19000: [0.00049231]**

**20000: [0.00049005]**

**21000: [0.00048786]**

**22000: [0.00048573]**

**23000: [0.00048365]**

**24000: [0.00048163]**

**25000: [0.00047966]**

**26000: [0.00047774]**

**27000: [0.00047587]**

**28000: [0.00047405]**

**29000: [0.00047227]**

**эталонное значение выходное значение среднеквадратичная ошибка**

**0.11079890232504547 0.11660310877617264 1.6844406263653127e-05**

**0.10882287686698608 0.11461356492826047 1.6766034111492878e-05**

**0.10674891816507021 0.11247341385955623 1.638492547809449e-05**

**0.10457889264214158 0.11018236626654132 1.569945832967177e-05**

**0.10231475317469851 0.10774007365195337 1.471705114046044e-05**

**0.09995853733543690 0.10514612115756217 1.3455512955787893e-05**

**0.09751236555957221 0.10240002070737519 1.1944586421922499e-05**

**0.09497843923659116 0.09950120460557443 1.0227703291437198e-05**

**0.09235903872914948 0.09644901973588016 8.363972317708845e-06**

**0.08965652132089966 0.09324272250984433 6.430419483794106e-06**

**0.08687331909509466 0.08988147471067853 4.524500103784398e-06**

**0.08401193674587720 0.08636434037650176 2.766901420687795e-06**

**0.08107494932422404 0.08269028386228872 1.304652834932304e-06**

**0.07806499992057400 0.07885816921325683 3.1455876342749254e-07**

**0.07498479728622486 0.07486676097392209 6.966285511018396e-09**

**Вывод:** В ходе лабораторной работы реализовал нелинейную ИНС. С сигмоидной функцией активации на скрытом слое, и линейной функцией активации на выходном слое. Данная нейронная сеть значительно уступает по скорости обучения той, которая использовалась в предыдущей лабораторной работе.