Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный университет”

Кафедра ИИТ

Лабораторная работа №4

По дисциплине: “МиАПР”

Тема: “ Нелинейные ИНС в задачах прогнозирования. Адаптивный шаг обучения.”

Выполнил:

Студент 2-го курса

Группы ПО-7

Степанюк А.С,

Проверил:

Крощенко А.А.

Брест, 2021

**Цель работы:** Изучить обучение и функционирование нелинейной 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):

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

alpha = 1 / (1 + np.square(self.hidden).sum())

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

alpha = 4 \* (error\_h \* \*2 \* self.hidden \* (1 - self.hidden)).sum() / (1 + (self.input \* \*2).sum()) / np.square(error\_h \* self.hidden \* (1 - self.hidden)).sum()

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):

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)

return square\_error

def testing(self, input, etalons):

print(" эталонное значение выходное значение среднеквадратичная ошибочка")

for i in range(len(etalons)):

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

delta = output - etalons[i]

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

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

neural\_net = NeuralNet()

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

for i in range(30000):

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

if i % 1000 == 0:

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

if np.isnan(error):

neural\_net = NeuralNet()

test\_input, test\_etalons = predict\_sets(3, 10, 15, 0.1)

neural\_net.testing(test\_input, test\_etalons)

**Результаты обучения:**

**1000: [nan]**

**2000: [nan]**

**3000: [nan]**

**4000: [nan]**

**5000: [nan]**

**6000: [nan]**

**7000: [nan]**

**8000: [nan]**

**9000: [2.12185925e-05]**

**10000: [3.86106056e-05]**

**11000: [7.57483635e-05]**

**12000: [9.83704005e-05]**

**13000: [0.00010914]**

**14000: [0.00011283]**

**15000: [0.00010893]**

**16000: [9.46931157e-05]**

**17000: [7.08428508e-05]**

**18000: [6.02823411e-05]**

**19000: [7.5234796e-05]**

**20000: [9.08051891e-05]**

**21000: [0.00010104]**

**22000: [0.00010745]**

**23000: [0.00011123]**

**24000: [0.00011297]**

**25000: [0.00011279]**

**26000: [0.00011033]**

**27000: [0.00010453]**

**28000: [9.28729235e-05]**

**29000: [6.84717227e-05]**

**эталонное значение выходное значение среднеквадратичная ошибочка**

**0.11079890232504547 0.11255785751485581 1.5469616798803656e-06**

**0.10882287686698608 0.11257030684138636 7.021615706516839e-06**

**0.10674891816507021 0.11258422086925934 1.7025378824758533e-05**

**0.10457889264214158 0.11259969742350773 3.216665467039301e-05**

**0.10231475317469851 0.11261684624674428 5.306656083254673e-05**

**0.09995853733543690 0.11263579001511470 8.035636775219899e-05**

**0.09751236555957221 0.11265666547174424 0.00011467490991490691**

**0.09497843923659116 0.11267962468680315 0.00015666598317139828**

**0.09235903872914948 0.11270483645401486 0.00020697574253056853**

**0.08965652132089966 0.11273248783408674 0.0002662501152588657**

**0.08687331909509466 0.11276278585615351 0.0003351322445859856**

**0.08401193674587720 0.11279595938887654 0.0004142599797563494**

**0.08107494932422404 0.11283226119328971 0.0005042634285745495**

**0.07806499992057400 0.11287197016980821 0.0006057625889655377**

**0.07498479728622486 0.11291539381197291 0.0007193650763995451**

**Вывод:** В ходе лабораторной работы реализовал нелинейную ИНС. С сигмоидной функцией активации на скрытом слое, и линейной функцией активации на выходном слое и адаптивным шагом обучения. Адаптивный шаг значительно сократил количество эпох, необходимых для обучения ИНС.