Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №4

за 1 семестр

По дисциплине: «МиАПР»

Тема: «Нелинейные ИНС в задачах прогнозирования»

Выполнил:

Студент 2 курса

Группы ПО-4(1)

Пивник Н.В

Проверил:

Крощенко А.А.

2020

**Лабораторная работа №4**

Нелинейные ИНС в задачах прогнозирования

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

**Вариант 2**

**Задание:**

Спрогнозировать нелинейный временной ряд, применяя параметры лабораторной работы №3. При этом необходимо использовать алгоритм обучения многослойной ИНС с адаптивным шагом.

Y = a\*cos(b\*x)+c\*sin(d\*x)

**Код программы:**

import numpy as np

import sys

import math

def func(x):

a = 0.2

b = 0.6

c = 0.09

d = 0.2

return a \* math.cos(b \* x) + c \* math.sin(d \* x)

class Network:

def \_\_init\_\_(self, learning\_rate = 0.25):

self.weights\_0\_1 = np.random.normal(0.0, 2 \*\* -0.5, (2, 6))

self.weights\_1\_2 = np.random.normal(0.0, 1, (1, 2))

self.tanh\_mapper = np.vectorize(self.tanh)

self.learning\_rate = np.array([learning\_rate])

def tanh(self, x):

return np.tanh(x)

def predict(self, inputs):

inputs\_1 = np.dot(self.weights\_0\_1, inputs)

outputs\_1 = self.tanh\_mapper(inputs\_1)

inputs\_2 = np.dot(self.weights\_1\_2, outputs\_1)

outputs\_2 = self.tanh(inputs\_2)

return outputs\_2

def train(self, inputs, expected\_predict):

inputs\_1 = np.dot(self.weights\_0\_1, inputs)

outputs\_1 = self.tanh\_mapper(inputs\_1)

inputs\_2 = np.dot(self.weights\_1\_2, outputs\_1)

outputs\_2 = self.tanh(inputs\_2)

actual\_predict = outputs\_2[0]

error\_layer\_2 = np.array([actual\_predict - expected\_predict])

gradient\_layer\_2 = -(self.tanh(actual\_predict) \*\* 2) - 1

weights\_delta\_layer\_2 = error\_layer\_2 \* gradient\_layer\_2

self.weights\_1\_2 -= (np.dot(weights\_delta\_layer\_2, outputs\_1.reshape(1, len(outputs\_1)))) \* self.learning\_rate

self.learning\_rate = (np.sum(np.square(error\_layer\_2) \* (np.ones((1, 2)) - np.square(outputs\_2))))/((1 + np.sum(np.square(outputs\_1)))\*(np.sum(np.square(error\_layer\_2)\*np.square(np.ones((1, 2)) - np.square(outputs\_2)))))

error\_layer\_1 = weights\_delta\_layer\_2 \* self.weights\_1\_2

gradient\_layer\_1 = -(self.tanh(outputs\_1) \*\* 2) - 1

weights\_delta\_layer\_1 = error\_layer\_1 \* gradient\_layer\_1

self.weights\_0\_1 -= np.dot(inputs.reshape(len(inputs), 1), weights\_delta\_layer\_1).T \* self.learning\_rate

def MSE(y, Y):

return np.mean((y - Y) \*\* 2)

step = 0.1

counter = 0

train = []

for i in range(-15, 15):

combol = []

inputs = []

for j in range(6):

x = counter \* step

inputs.append(func(x))

counter += 1

combol.append(inputs)

x = counter \* step

combol.append(func(x))

combo = tuple(combol)

train.append(combo)

learning\_rate = 0.05

network = Network(learning\_rate)

losses = {'train':[], 'validation':[]}

Emin = 1e-6

train\_loss = 0

epoch = 0

while True:

inputs = []

correct\_predictions = []

for input\_stat, correct\_predict in train:

network.train(np.array(input\_stat), correct\_predict)

inputs.append(np.array(input\_stat))

correct\_predictions.append(np.array(correct\_predict))

train\_loss = MSE(network.predict(np.array(inputs).T), np.array(correct\_predictions))

epoch += 1

if train\_loss <= Emin:

break

print("epochs: {}, training loss: {}".format(

str(epoch),

str(train\_loss)

))

print("\nРЕЗУЛЬТАТЫ ОБУЧЕНИЯ:")

for input\_stat, correct\_predict in train:

print("the prediction is: {}, expected: {}, mistake: {}".format(

str(network.predict(input\_stat)),

str(correct\_predict),

str(network.predict(input\_stat) - correct\_predict)

))

predict = []

for i in range(30, 45):

combol = []

inputs = []

for j in range(6):

x = counter \* step

inputs.append(func(x))

counter += 1

combol.append(inputs)

x = counter \* step

combol.append(func(x))

combo = tuple(combol)

predict.append(combo)

print("\nРЕЗУЛЬТАТЫ ПРОГНОЗИРОВАНИЯ")

for input\_stat, correct\_predict in predict:

print("the prediction is: {}, expected: {}, mistake: {}".format(

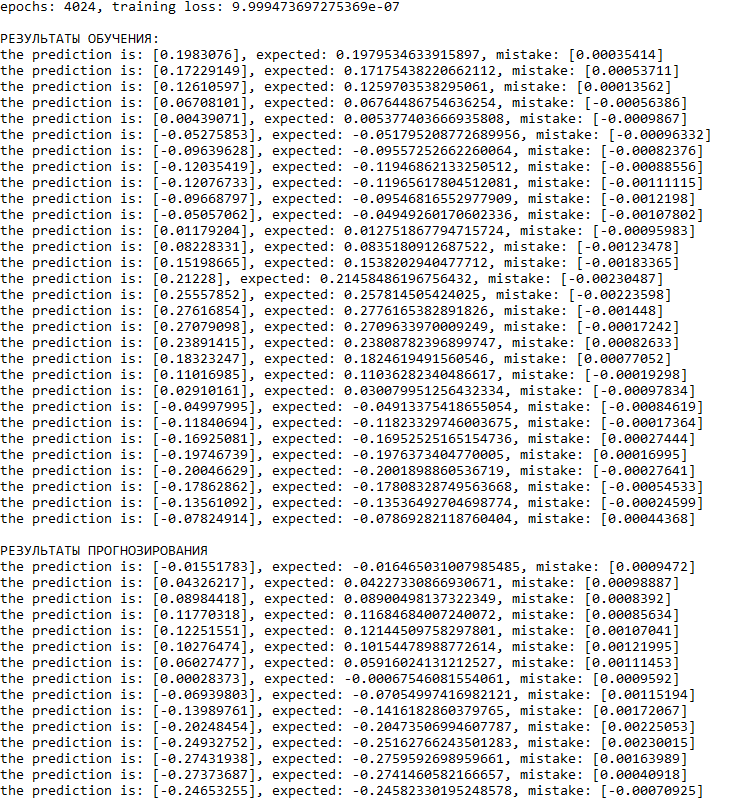
str(network.predict(input\_stat)),

str(correct\_predict),

str(network.predict(input\_stat) - correct\_predict)

))

Результаты:



Вывод: В ходе выполнения работы спроектировал линейную ИНС с использованием адаптивного шага обучения.