Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №3

за 1 семестр

По дисциплине: «МиАПР»

Тема: «Нелинейные ИНС в задачах прогнозирования»

Выполнил:

Студент 2 курса

Группы ПО-4(1)

Иваненко И. Л.

Проверил:

Крощенко А.А.

2020

**Лабораторная работа №3**

Нелинейные ИНС в задачах прогнозирования

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

**Вариант 10**

**Задание:**

**Код программы:**

import numpy as np

import sys

import math

def func(x):

a = 0.2

b = 0.4

c = 0.09

d = 0.4

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

class Network:

def \_\_init\_\_(self, learning\_rate = 0.5):

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.bias\_0\_1 = 1

#self.bias\_1\_2 = 1

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

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

def sigmoid(self, x):

return 1 / (1 + np.exp(-x))

def predict(self, inputs):

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

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

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

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

return outputs\_2

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

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

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

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

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

actual\_predict = outputs\_2[0]

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

gradient\_layer\_2 = actual\_predict \* (1 - actual\_predict)

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.bias\_1\_2 \*= self.learning\_rate \* weights\_delta\_layer\_2

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

gradient\_layer\_1 = outputs\_1 \* (1 - outputs\_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

#self.bias\_0\_1 -= self.learning\_rate \* weights\_delta\_layer\_1

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)

epochs = 1200

learning\_rate = 0.25

network = Network(learning\_rate)

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

Emin = 0.011

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))

sys.stdout.write("\rTraining loss: {}, Epochs: {}".format(str(train\_loss)[:8], str(epoch)))

epoch += 1

if train\_loss <= Emin:

break

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)

))

Результат:

D:\>p

Training loss: 0.010999, Epochs: 1807

РЕЗУЛЬТАТЫ ОБУЧЕНИЯ:

the prediction is: [0.21369575], expected: 0.21566083134884803, mistake: [-0.00196509]

the prediction is: [0.22837371], expected: 0.21895911035459031, mistake: [0.0094146]

the prediction is: [0.22430766], expected: 0.20970576630561158, mistake: [0.0146019]

the prediction is: [0.2016369], expected: 0.18843123836158115, mistake: [0.01320566]

the prediction is: [0.16204101], expected: 0.15635506863238505, mistake: [0.00568594]

the prediction is: [0.11189477], expected: 0.11531599308488084, mistake: [-0.00342122]

the prediction is: [0.06387542], expected: 0.06766653774985257, mistake: [-0.00379112]

the prediction is: [0.03016082], expected: 0.016138162401499512, mistake: [0.01402266]

the prediction is: [0.0126778], expected: -0.03631531778004107, mistake: [0.04899312]

the prediction is: [0.00538919], expected: -0.08668705685864561, mistake: [0.09207625]

the prediction is: [0.00260033], expected: -0.13208954272987805, mistake: [0.13468987]

the prediction is: [0.00150384], expected: -0.16992012101009504, mistake: [0.17142396]

the prediction is: [0.00103968], expected: -0.19801018972723708, mistake: [0.19904987]

the prediction is: [0.00083553], expected: -0.2147495123693462, mistake: [0.21558505]

the prediction is: [0.00075954], expected: -0.2191785231633661, mistake: [0.21993806]

the prediction is: [0.0007699], expected: -0.21104333327177924, mistake: [0.21181324]

the prediction is: [0.00087205], expected: -0.19081028472909783, mistake: [0.19168234]

the prediction is: [0.00112398], expected: -0.15963921782762297, mistake: [0.1607632]

the prediction is: [0.00169718], expected: -0.11931698437419269, mistake: [0.12101416]

the prediction is: [0.00307703], expected: -0.0721550181073362, mistake: [0.07523204]

the prediction is: [0.00663714], expected: -0.020856833953390233, mistake: [0.02749398]

the prediction is: [0.01585683], expected: 0.03163694839911395, mistake: [-0.01578012]

the prediction is: [0.03703002], expected: 0.08231717273037709, mistake: [-0.04528715]

the prediction is: [0.07503322], expected: 0.12827864331182462, mistake: [-0.05324542]

the prediction is: [0.12497017], expected: 0.16688666249216988, mistake: [-0.04191649]

the prediction is: [0.1734292], expected: 0.1959280622380926, mistake: [-0.02249886]

the prediction is: [0.2090926], expected: 0.21373807188989244, mistake: [-0.00464548]

the prediction is: [0.22711139], expected: 0.21929574955851877, mistake: [0.00781564]

the prediction is: [0.22642414], expected: 0.2122825066497665, mistake: [0.01414164]

the prediction is: [0.20705344], expected: 0.1931003706528747, mistake: [0.01395307]

РЕЗУЛЬТАТЫ ПРОГНОЗИРОВАНИЯ

the prediction is: [0.17021868], expected: 0.1628489392965125, mistake: [0.00736974]

the prediction is: [0.12118997], expected: 0.1232623471532765, mistake: [-0.00207238]

the prediction is: [0.07171436], expected: 0.07660985802223053, mistake: [-0.0048955]

the prediction is: [0.03492806], expected: 0.02556578153675297, mistake: [0.00936228]

the prediction is: [0.01486397], expected: -0.026943829095391864, mistake: [0.04180779]

the prediction is: [0.00624425], expected: -0.07790891031330766, mistake: [0.08415316]

the prediction is: [0.0029273], expected: -0.1244079372379214, mistake: [0.12733524]

the prediction is: [0.00163686], expected: -0.1637753973110943, mistake: [0.16541226]

the prediction is: [0.00109781], expected: -0.19375458827156833, mistake: [0.1948524]

the prediction is: [0.00086062], expected: -0.21262698146889367, mistake: [0.2134876]

the prediction is: [0.00076634], expected: -0.21931073488622205, mistake: [0.22007708]

the prediction is: [0.00076198], expected: -0.21342270870649288, mistake: [0.21418469]

the prediction is: [0.00084524], expected: -0.19530042843857692, mistake: [0.19614567]

the prediction is: [0.00106231], expected: -0.16598273658802887, mistake: [0.16704505]

the prediction is: [0.00155549], expected: -0.1271502419970512, mistake: [0.12870573]

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