Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

за 1 семестр

По дисциплине: «МиАПР»

Тема: «Линейная искусственная нейронная сеть. Адаптивынй шаг обучения »

Выполнил:

Студент 2 курса

Группы ПО-4(1)

Иваненко И. Л.

Проверил:

Крощенко А.А.

2020

Лабораторная работа №2

Линейная искусственная нейронная сеть. Адаптивный шаг обучения

Цель работы: изучить обучение и функционирование линейной ИНС при решении

задач прогнозирования с применением адаптивного шага.

Вариант 10

Задание:

Модифицировать программу из лабораторной работы №1, используя правило адаптивного шага обучения. Произвести исследование получившейся модели ИНС на задачах прогнозирования, согласно варианту лабораторной работы № 1

Задание из лабораторной работы № 1:

Написать на любом ЯВУ программу моделирования прогнозирующей линейной

ИНС. Для тестирования использовать функцию

y = a\*sin(bx) + d

a = 2, b = 9 d = 0.4, кол-во входов ИНС = 3.

Обучение и прогнозирование производить на 30 и 15 значениях соответственно табулируя функцию с шагом 0.1. Скорость обучения выбирается студентом самостоятельно, для чего моделирование проводится несколько раз для разных а. Результаты оцениваются по двум критериям - скорости обучения и минимальной достигнутой ошибке. Необходимо заметить, что эти критерии в общем случае являются взаимоисключающими, и оптимальные значения для каждого критерия достигаются при разных a.

Код программы:

import math

import random

def function(x, a, b, d):

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

a = 2

b = 9

d = 0.4

amount\_of\_inputs = 3 # Количество входов нейронной сети

amout\_of\_training\_values = 30 # Количество элементов, на которых происходит обучение нейронной сети

amount\_of\_predicated\_values = 15 # Количество элементов, на которых происходит тестирование нейронной сети

min\_error = 0.001 # Минимальная среднеквадратичная ошибка

step = 0.1 # Шаг

training\_speed = 0.1 # Скорость обучения

T = random.uniform(0.5, 1) # Порог нейронной сести

synaptic\_weights = [] # Синаптические веса

#Случайно задаем синаптические веса нейронной сети

for i in range(amount\_of\_inputs):

synaptic\_weights.append(random.uniform(0, 1))

training\_outputs = [] # Эталонные выходные значения

for i in range(amout\_of\_training\_values + amount\_of\_predicated\_values):

x = i \* step

training\_outputs.append(function(x, a, b, d))

error = 10

print(T)

print(synaptic\_weights)

epochas = 0

while(error > min\_error):

error = 0 #Суммарная среднеквадратичная ошибка

for i in range(amout\_of\_training\_values):

X = 0

for j in range(amount\_of\_inputs):

X += (training\_outputs[i + j])\*\*2

training\_speed = 1/(1 + X)

output = 0

# Вычисляем выходное значение нейронной сети

for j in range(amount\_of\_inputs):

output += (synaptic\_weights[j] \* training\_outputs[j + i])

output -= T

#Корректируем порог нейронной сети, веса и ошибку

for j in range(amount\_of\_inputs):

synaptic\_weights[j] -= training\_speed \* (output - training\_outputs[i + amount\_of\_inputs]) \* training\_outputs[i + j]

T += training\_speed \* (output - training\_outputs[i + amount\_of\_inputs])

error += 0.5 \* ((output - training\_outputs[i + amount\_of\_inputs]) \*\* 2)

epochas+=1

print(str(epochas) + " -- ЭПОХИ")

outputs = []

print("Результаты обучения:")

print(" %2s %2s %2s %2s " % (

"N",

"Эталонное значение",

"Полученное значение",

"Ошибка"

))

for i in range(amout\_of\_training\_values):

outputs.append(0)

for j in range(amount\_of\_inputs):

outputs[i] += synaptic\_weights[j] \* training\_outputs[j + i]

outputs[i] -= T

print(" %2d %9lf %18lf %19lf " % (

i,

training\_outputs[i + amount\_of\_inputs],

outputs[i],

outputs[i] - training\_outputs[i + amount\_of\_inputs]

))

print("Результаты прогнозирования:")

print(" %2s %2s %2s %2s " % (

"N",

"Эталонное значение",

"Полученное значение",

"Ошибка"

))

for i in range(amount\_of\_predicated\_values):

outputs.append(0)

for j in range(amount\_of\_inputs):

outputs[i + amout\_of\_training\_values] += synaptic\_weights[j] \* training\_outputs[amout\_of\_training\_values - amount\_of\_inputs + j + i]

outputs[i + amout\_of\_training\_values] -= T

print(" %2d %9lf %18lf %19lf " % (

i + amout\_of\_training\_values,

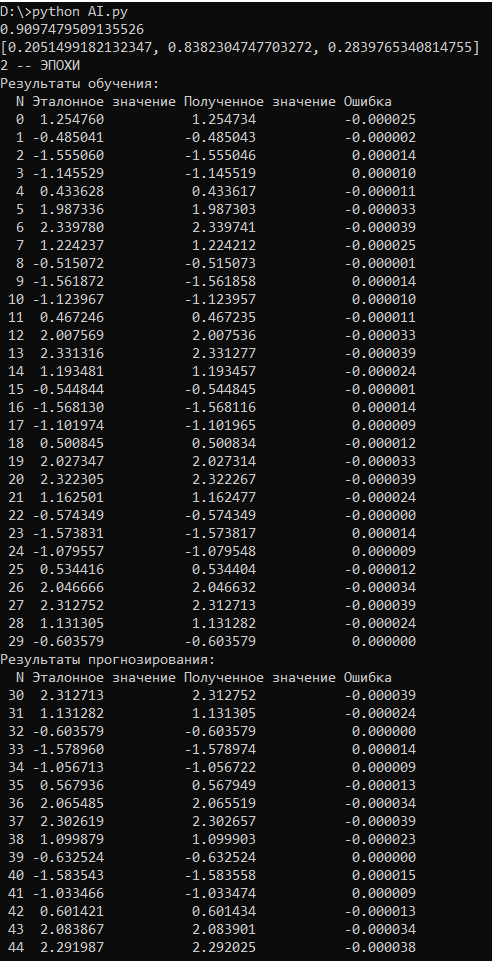
outputs[i + amout\_of\_training\_values],

training\_outputs[i + amout\_of\_training\_values],

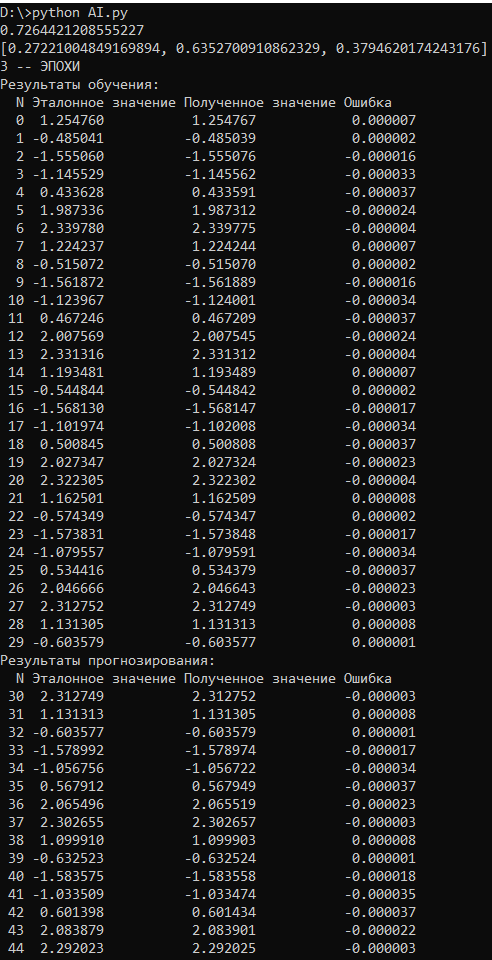
outputs[i + amout\_of\_training\_values] - training\_outputs[i + amout\_of\_training\_values]

))

Результат выполнения с адаптивным шагом обучения:



Результат выполнения без адаптивного шага обучения:



С адаптивным шагом обучения – 2 эпохи

Без адаптивного шага обучения – 3 эпохи

Вывод: адаптивный шаг обучения нейронной сети позволяет сократить количество эпох обучения искусственной нейронной сети.