**Лабораторная работа № 2**

«Линейная нейронная сеть. Правило обучения Уидроу-Хоффа».

Целью работы является исследование свойств линейной нейронной сети и алгоритмов ее обучения, применение сети в задачах аппроксимации и фильтрации.

Ход работы.

1. Использовать линейную нейронную сеть с задержками для аппроксимации функции. В качестве метода обучения использовать адаптацию.

2. Использовать линейную нейронную сеть с задержками для аппроксимации функции и выполнения многошагового прогноза.

3. Использовать линейную нейронную сеть в качестве адаптивного фильтра для подавления помех. Для настройки весовых коэффициентов использовать метод наименьших квадратов.

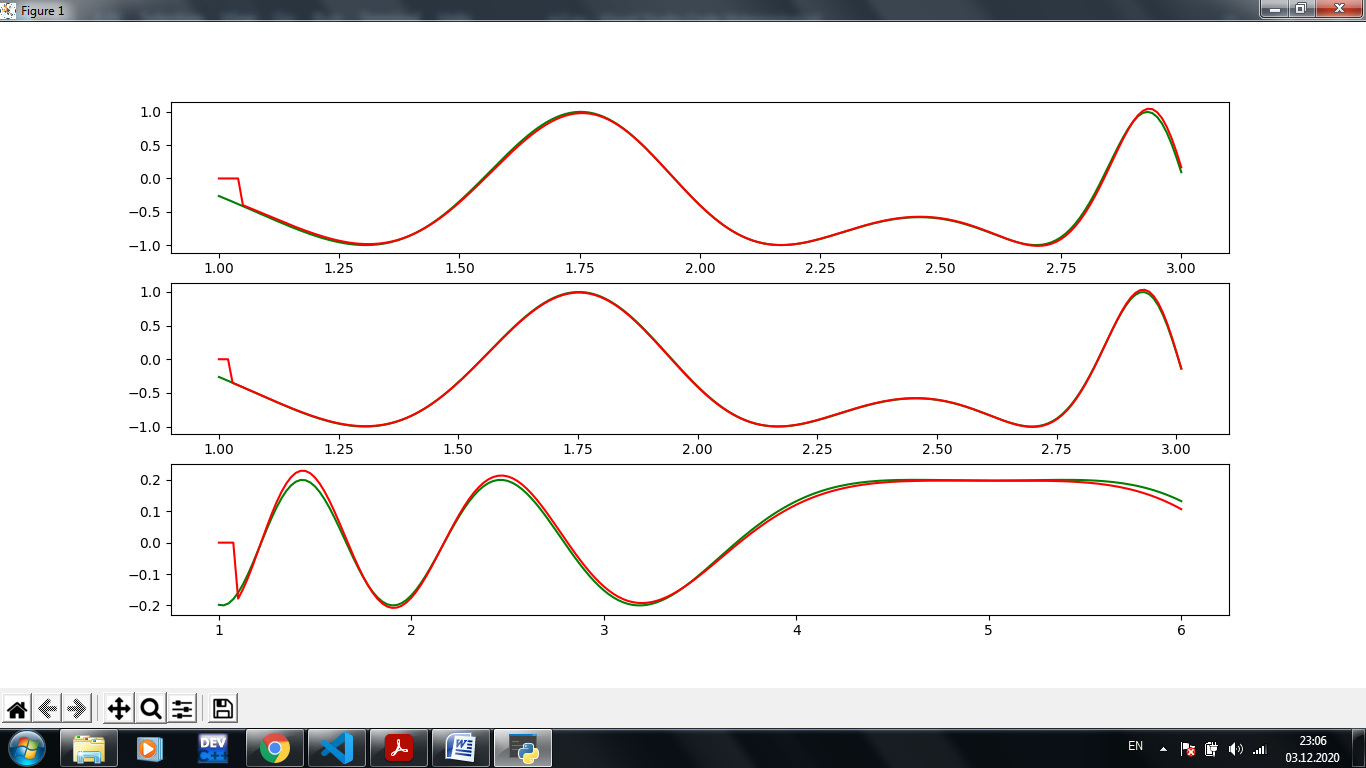
Входные данные и результаты.

**Вариант № 15**

error1 = 0.023463

error2 = 0.009106

error3 = 0.011636



Исходный код.

import sys

import math

import numpy as np

import matplotlib.pyplot as plt

def in1F(t):

    return math.sin((t \*\* 3) \* math.sin(t) - 10)

def in2F(t):

    return math.cos((t \*\* 2) - 10 \* t + 3)

def out2F(t):

    return (1 / 5) \* math.cos((t \*\* 2) - 10 \* t + 6)

out2F = np.vectorize(out2F)

in1F = np.vectorize(in1F)

in2F = np.vectorize(in2F)

error1 = 0

delays1 = 5

learningRate1 = 0.01

# Whi is [0, 1, 1, 1, 1, 1]?

# Because 6 = delays1 + 1(base/bais)

weights1 = np.array([0, 1, 1, 1, 1, 1], float)

time1 = np.arange(1.0, 3.0 + 0.01, 0.01)

ls1 = in1F(time1)

out1 = np.zeros(time1.size)

for \_ in range(50):

    error1 = 0

    for i in range(time1.size - delays1):

        x1 = np.array([1, ls1[i], ls1[i + 1], ls1[i + 2], ls1[i + 3], ls1[i + 4]], float)

        out1[i + delays1] = weights1.dot(x1)

        weights1 -= learningRate1 \* (out1[i + delays1] - ls1[i + delays1]) \* x1

        error1 += (out1[i + delays1] - ls1[i + delays1]) \*\* 2

error1 = math.sqrt(error1 / max(1, time1.size - delays1))

print("error1 = ", error1)

plt.subplot(3, 1, 1)

plt.plot(time1, ls1, color='green')

plt.plot(time1, out1, color='red')

###################################################################

error2 = 0

delays2 = 3

learningRate2 = 0.01

epsilon2 = 10 \*\* (-6)

# Whi is [0, 1, 1, 1]?

# Because 4 = delays2 + 1(base/bais)

weights2 = np.array([0, 1, 1, 1], float)

time2 = np.linspace(1.0, 3.01, num=11 + int((3.0 - 1.0) / 0.01))

ls2 = in1F(time2)

out2 = np.zeros(time2.size)

for \_ in range(600):

    error2 = 0

    for i in range(time2.size - delays2):

        x2 = np.array([1, ls2[i], ls2[i + 1], ls2[i + 2]], float)

        out2[i + 3] = weights2.dot(x2)

        weights2 -= learningRate2 \* (out2[i + delays2] - ls2[i + delays2]) \* x2

        error2 += (out2[i + delays2] - ls2[i + delays2]) \*\* 2

    if error2 < epsilon2:

        break

error2 = math.sqrt(error2 / max(1, time2.size - delays2))

print("error2 = ", error2)

plt.subplot(3, 1, 2)

plt.plot(time2, ls2, color='green')

plt.plot(time2, out2, color='red')

##########################################################################

error3 = 0

delays3 = 4

learningRate3 = 0.001

epsilon3 = 10 \*\* (-6)

weights3 = np.array([0, 1, 1, 1, 1], float)

time3 = np.linspace(1.0, 6.0, num=1 + int((6.0 - 1.0) / 0.025))

ls3 = in2F(time3)

ls3\_out = out2F(time3)

out3 = np.zeros(time3.size)

for \_ in range(1000):

    error3 = 0

    for i in range(time3.size - delays3):

        x3 = np.array([1, ls3[i], ls3[i + 1], ls3[i + 2], ls3[i + 3]], float)

        out3[i + delays3] = weights3.dot(x3)

        weights3 -= learningRate3 \* (out3[i + delays3] - ls3\_out[i + delays3]) \* x3

        error3 += (out3[i + delays3] - ls3\_out[i + delays3]) \*\* 2

error3 = math.sqrt(error3 / max(1, time3.size - delays3))

print("error3 = ", error3)

plt.subplot(3, 1, 3)

plt.plot(time3, ls3\_out, color='green')

plt.plot(time3, out3, color='red')

plt.show()

Выводы:

В данной лабораторной работе была реализована линейная нейронная сеть. Байсы я добавил в вектор весов **weights**, чтобы упростить чтение и проверку кода.   
Правило Уидроу-Хоффа по сути, это вырождение градиентного спуска при условии, что производная активационной функции равна единице.