**Министерство образования и науки Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет**

**имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ «Информатика и системы управления» (ИУ)

КАФЕДРА «Информационная безопасность» (ИУ8)

**ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ**

**ЛАБОРАТОРНАЯ РАБОТА № 2**

**Применение однослойной нейронной сети с линейной функцией для прогнозирования временных рядов**

**Вариант № 7**

**Преподаватель:** Коннова Н.С.

**Студент:** Кустов И. А.

**Группа:** ИУ8-61

Оглавление

[Цель работы 3](#_Toc5301072)

[Постановка задачи 3](#_Toc5301073)

[Практическая часть 4](#_Toc5301074)

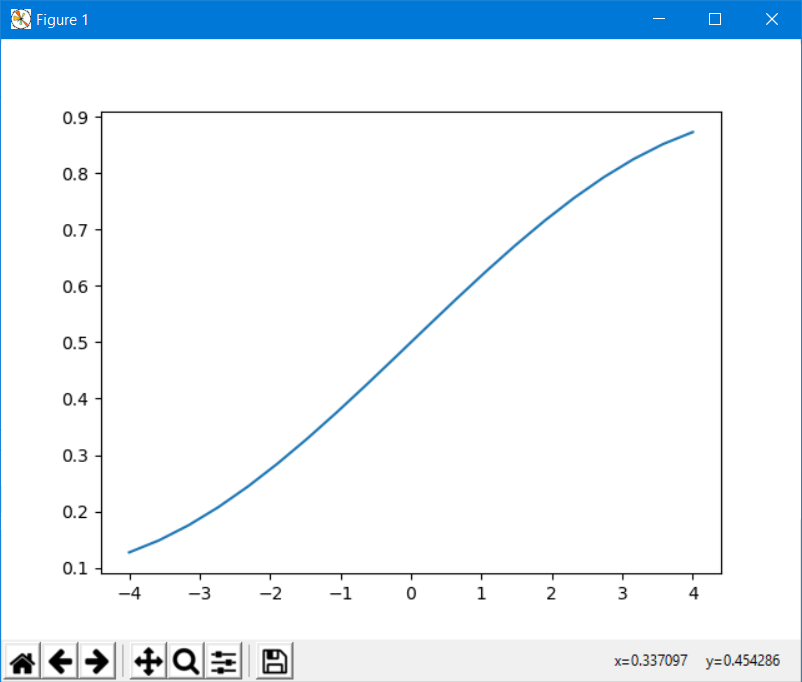
[Вывод 10](#_Toc5301075)

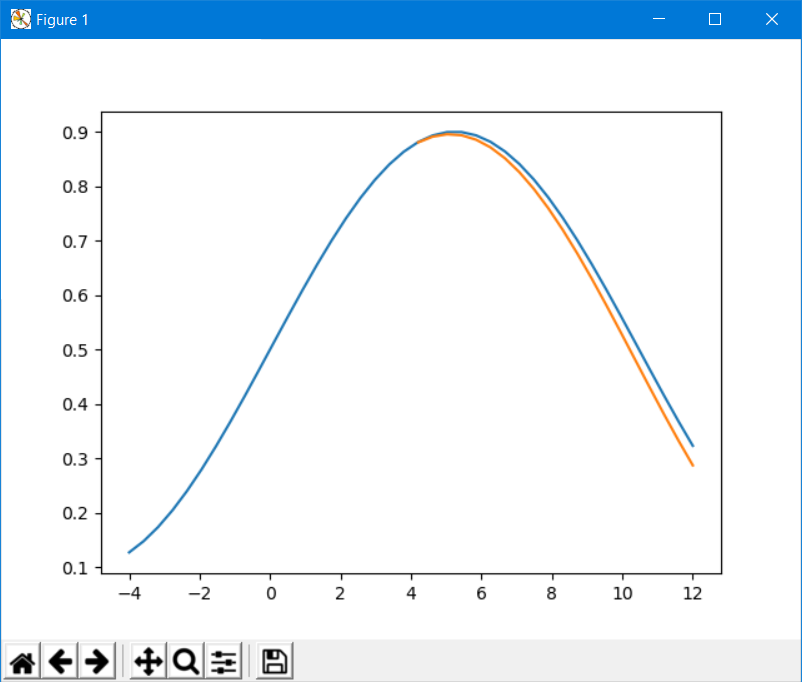
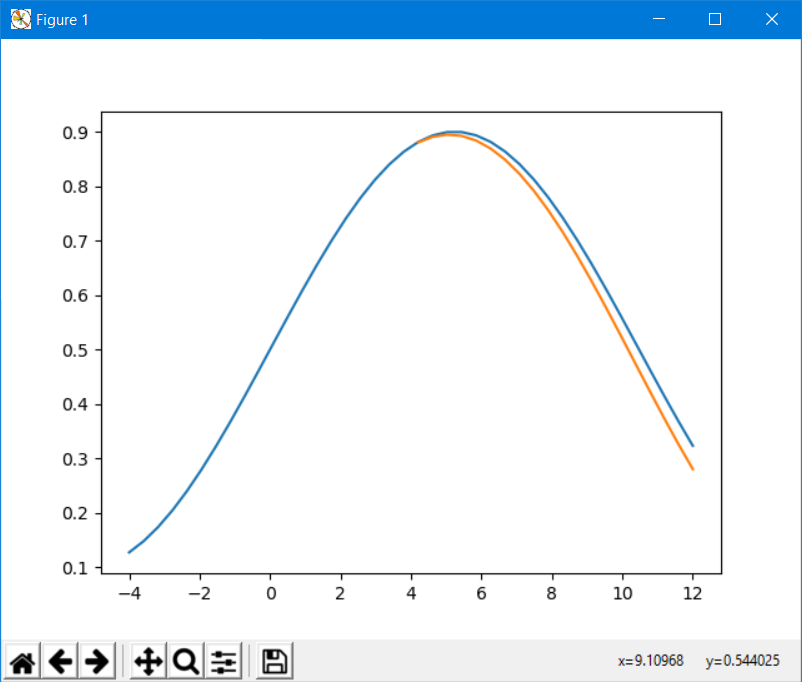
[Приложение 10](#_Toc5301076)

# **Цель работы** – изучить возможности однослойных НС в задачах прогнозирования временных рядов методом скользящего окна (авторегрессия).

**Постановка задачи** – на временном интервале [a, b] задан дискретный набор значений функции x(t). Количество точек – 20, расположение – равномерное. Методом «скользящего окна» спрогнозировать поведение функции x(t) следующего последующего интервала (b, 2b-a]. Для решения использовать однослойную НС с количеством нейронов p и линейной ФА.

# **Практическая часть**

Рассмотрим прогноз функции X(t) = 0.4sin(0.3t)+0.5 по 20 равностоящим исходным значениям x, заданным на интервале [-4, 4]. Выберем длину окна 6, норму обучения 1. Будем считать вес смещения w0 = 0. На рисунке 1 показан график самой функции. На рисунках 2 и 3 показаны график самой функции (голубой) и ее прогноз на интервале (желтый) на интервале [4, 12] при различном количестве эпох обучения. В таблице 1 приведены результаты обучения.

 **Рисунок 1.** Исходная функция

**Рисунок 2.** Исходная функция и **Рисунок 3.** Исходная функция и

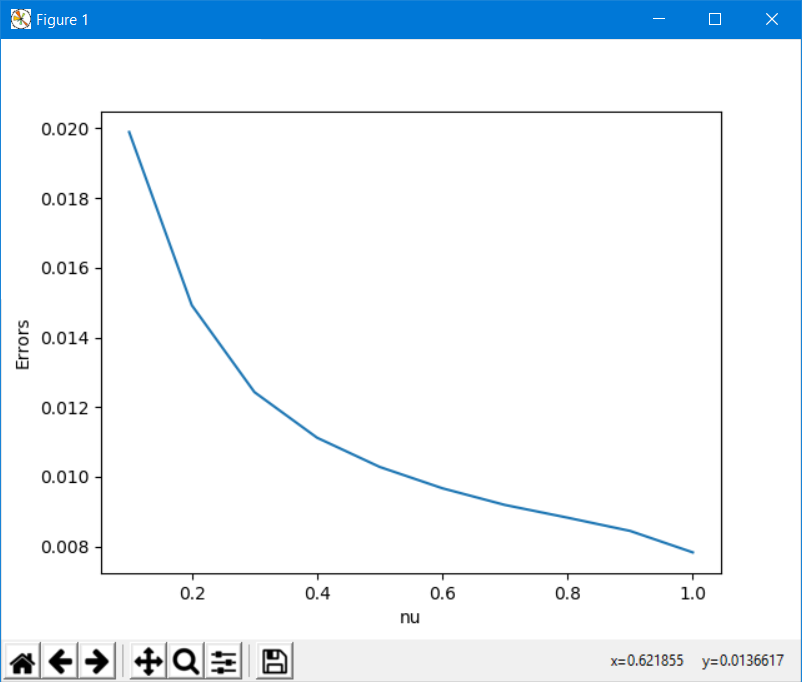
ее прогноз при 2000 эпох (ε≈0,0078) ее прогноз при 4000 эпох (ε≈ 0.0055)

Таблица 1

**Результаты обучения**

|  |  |  |
| --- | --- | --- |
| **k** | **Вектор весов w** | **Квадратичная ошибка E** |
| 1 | [0.112, 0.134, 0.161, 0.193, 0.228, 0.266] | 0.490 |
| 2 | [0.105, 0.128, 0.158, 0.192, 0.232, 0.276] | 0.135 |
| 3 | [0.098, 0.122, 0.154, 0.192, 0.236, 0.286] | 0.133 |
| 4 | [0.091, 0.117, 0.151, 0.192, 0.241, 0.295] | 0.131 |
| 5 | [0.084, 0.111, 0.147, 0.192, 0.245, 0.304] | 0.129 |
| 6 | [0.077, 0.106, 0.144, 0.192, 0.249, 0.314] | 0.127 |
| 7 | [0.071, 0.1, 0.141, 0.192, 0.253, 0.323] | 0.125 |
| 8 | [0.064, 0.095, 0.138, 0.192, 0.257, 0.332] | 0.123 |
| 9 | [0.058, 0.09, 0.134, 0.192, 0.261, 0.34] | 0.121 |
| 10 | [0.052, 0.084, 0.131, 0.192, 0.265, 0.349] | 0.119 |
| … | … | … |
| 1991 | [-0.177, -0.22087, -0.12898, 0.0972, 0.45406, 0.93591] | 0.007818 |
| 1992 | [-0.17694, -0.22087, -0.12902, 0.09715, 0.45404, 0.93596] | 0.007817 |
| 1993 | [-0.17688, -0.22088, -0.12906, 0.09711, 0.45403, 0.93601] | 0.007816 |
| 1994 | [-0.17682, -0.22088, -0.1291, 0.09707, 0.45402, 0.93606] | 0.007815 |
| 1995 | [-0.17676, -0.22089, -0.12914, 0.09702, 0.454, 0.9361] | 0.007815 |
| 1996 | [-0.1767, -0.2209, -0.12918, 0.09698, 0.45399, 0.93615] | 0.007814 |
| 1997 | [-0.17664, -0.2209, -0.12922, 0.09694, 0.45397, 0.9362] | 0.007813 |
| 1998 | [-0.17658, -0.22091, -0.12926, 0.09689, 0.45396, 0.93625] | 0.007812 |
| 1999 | [-0.17652, -0.22092, -0.1293, 0.09685, 0.45395, 0.93629] | 0.007811 |
| 2000 | [-0.17646, -0.22092, -0.12934, 0.09681, 0.45393, 0.93634] | 0.007810 |
| … | … | … |
| 3991 | [-0.06788, -0.23246, -0.20365, 0.01811, 0.42928, 1.0233] | 0.0055589 |
| 3992 | [-0.06783, -0.23247, -0.20368, 0.01807, 0.42927, 1.02334] | 0.0055587 |
| 3993 | [-0.06778, -0.23248, -0.20372, 0.01804, 0.42925, 1.02338] | 0.0055584 |
| 3994 | [-0.06773, -0.23248, -0.20375, 0.018, 0.42924, 1.02342] | 0.0055580 |
| 3995 | [-0.06768, -0.23249, -0.20379, 0.01796, 0.42923, 1.02346] | 0.0055577 |
| 3996 | [-0.06763, -0.23249, -0.20382, 0.01793, 0.42922, 1.0235] | 0.0055574 |
| 3997 | [-0.06758, -0.2325, -0.20386, 0.01789, 0.42921, 1.02354] | 0.0055570 |
| 3998 | [-0.06753, -0.2325, -0.20389, 0.01785, 0.4292, 1.02358] | 0.0055568 |
| 3999 | [-0.06748, -0.23251, -0.20392, 0.01782, 0.42919, 1.02362] | 0.0055564 |
| 4000 | [-0.06743, -0.23251, -0.20396, 0.01778, 0.42917, 1.02366] | 0.0055561 |

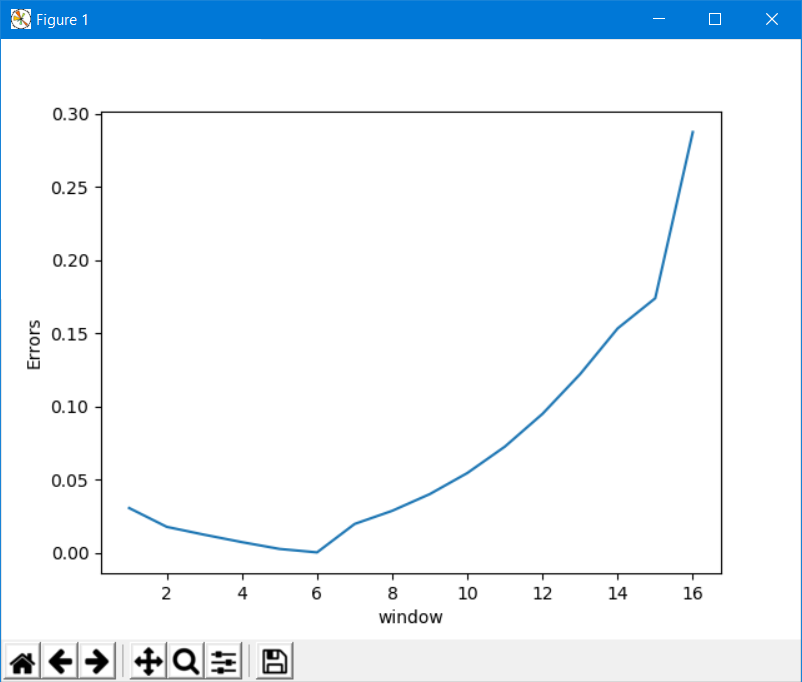
На рисунке 4 приведен график зависимости ошибки от нормы обучения при количестве эпох M = 2000 и окне p = 6.



**Рисунок 4.** График зависимости

ошибки от нормы обучения

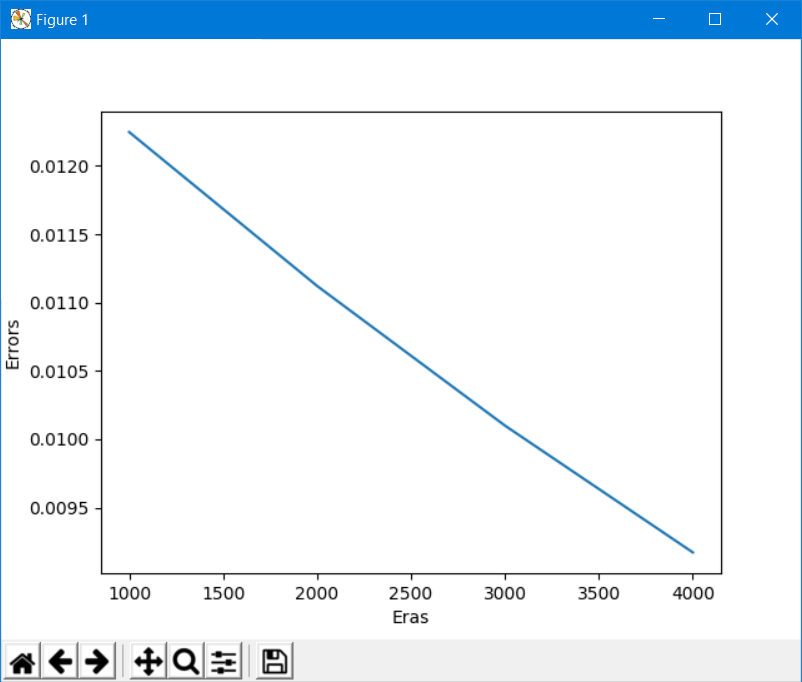
На рисунке 5 приведен график зависимости ошибки от длины окна при количестве эпох M = 500 и норме обучения 0.4.



**Рисунок 5.** График зависимости

ошибки от длины окна

На рисунке 6 приведен график зависимости ошибки от количества при длине окна p = 6 и норме обучения 0.4.



**Рисунок 6.** График зависимости

ошибки от количества эпох обучения

# **Вывод**

В процессе работы было проведено 2 обучения при разном количестве эпох и 3 серии обучения при переменных значениях одного из параметров (количество эпох обучения, норма обучения, длина окна). При повышении количества эпох обучения квадратичная ошибка прогнозирования понижается. При повышении нормы обучения увеличивается скорость обучения. В случае изменения длины окна наименьшая квадратичная ошибка получается при длине окна 4-7.

# **Приложение**

Ссылка на гитхаб с кодом:

<https://github.com/Kustov-Ilya/lab_work_II/blob/master/lab2.py>

Листинг №1

import numpy as np

import matplotlib.pyplot as plt

import math

from multiprocessing import Process, Queue

#p - окно

#eras - колво эпох

#isplot - True, если необходимо построить предсказанную функцию, False, если вернуть ошибку

def Train(p, eras, nu, isplot = True):

array = np.linspace(-4, 4, 20)

matrix = [[Function(array[i + j]) for j in range(p)] for i in range(20 - p)]

w = np.zeros(p)

for \_ in range(eras):

error = 0

for i in range(len(matrix)):

sigma = Function(array[p + i]) - np.dot(w,matrix[i])

error += sigma \*\* 2

w += [nu \* sigma \* k for k in matrix[i]]

error = error \*\* 0.5

print(f'Era:{eras} nu = {np.around(nu, 1)} p = {p}\nVector w:{np.around(w, 3)}')

return Plot(w, p) if isplot else (w)

def Real\_Plot():

x = np.linspace(-4,4, 20)

plt.plot(x, [Function(t) for t in x])

plt.show()

def Plot(w, p):

x = np.linspace(-4, 12, 40)

matrix = np.array([[Function(x[i + j]) for j in range(p)] for i in range(20-p, 40 - p)])

plt.plot(x, [Function(t) for t in x])

plt.plot(x[20:], [np.dot(w, arr) for arr in matrix])

print(f'Error: {np.around(Get\_error(w, p), 4)}')

plt.show()

def Get\_error(w, p):

x = np.linspace(-4, 8, 40)

matrix = np.array([[Function(x[i + j]) for j in range(p)] for i in range(20-p, 40 - p)])

Real\_line = np.array([Function(t) for t in x])

Neuron\_line = [np.dot(w, arr) for arr in matrix]

error = Real\_line[20:]-Neuron\_line

return sum([a\*\*2 for a in error])

def Function(t):

return 0.4 \* math.sin(0.3 \* t) + 0.5

def Test\_nu():

error\_list = []

for i in np.arange(0.1, 1.1, 0.1):

w = Train(6, 2000, i, False)

error\_list.append(Get\_error(w, 6))

plt.plot(np.arange(0.1, 1.1, 0.1), error\_list)

plt.xlabel("nu")

plt.ylabel("Errors")

plt.show()

def Test\_p():

error\_list = []

for i in np.arange(1, 17):

w = Train(i, 500, 0.4, False)

error\_list.append(Get\_error(w, i))

plt.plot(np.arange(1, 17), error\_list)

plt.xlabel("window")

plt.ylabel("Errors")

plt.show()

def Test\_era():

error\_list = []

for i in np.arange(1000, 5000, 1000):

w = Train(6, i, 0.4, False)

error\_list.append(Get\_error(w, 6))

plt.plot(np.arange(1000, 5000, 1000), error\_list)

plt.xlabel("Eras")

plt.ylabel("Errors")

plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

Real\_Plot()

Train(6, 2000, 1)

Train(6, 4000, 1)

Test\_nu()

Test\_p()

Test\_era()