Министерство образования Российской Федерации

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н.Э. БАУМАНА

Факультет: Информатика и системы управления
Кафедра: Информационная безопасность (ИУ8)

ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Лабораторная работа №3 на тему:

«Применение однослойной нейронной сети с линейной функцией активации для прогнозирования временных рядов»

Вариант 8

Преподаватель:
Коннова Н. С.

Студент:
Песоцкий А. А.

Группа:
ИУ8-61

Москва, 2020

Цель работы

Изучить возможности однослойных НС в задачах прогнозирования временных рядов методов скользящего окна (авторегрессия)

Постановка задачи

На временном интервале [a, b] задан дискретный набор значений функции $x(t)$. Количество точек N=20, расположение – равномерное. Методом «скользящего окна» спрогнозировать поведение функции $x(t)$ на N точках последующего интервала (b, 2b – a]. Для решения использовать однослойную НС с количеством нейронов p и линейной функцией активации. Исходное количество нейронов (длина окна) $p = 4$. Обучение производить методом Видроу — Хоффа. Исследовать влияние количества эпох M обучения и коэффициента обучения η на средне-квадратичную погрешность приближения $\varepsilon = \sqrt{\sum_i [x(t_i) - \tilde{x}(t_i)]^2}$.

Исследовать процесс прогнозирования при постепенном изменении (уменьшении/увеличении) размера окна p. Сделать выводы по результатам численного эксперимента.

Ход работы

1. Функция и её прогноз

Рассмотрим прогноз функции $\boldsymbol{X}(\boldsymbol{t}) = \mathbf{sin}^2\,\boldsymbol{t}$ по 20 равностоящим исходным значениям x, заданным на интервале [0, 2]. Выберем начальную длину окна $\boldsymbol{p}$ = **4**, норму обучения $\boldsymbol{\eta} = \mathbf{0.3}$.

Зададим нулевые начальные веса: **W = [0, 0, 0, 0, 0]**. На рисунках 1 и 2 показаны график функции и её прогноз (круглые маркеры) на интервале $t\ \in [2, 4]$ при различном количестве эпох обучения M.

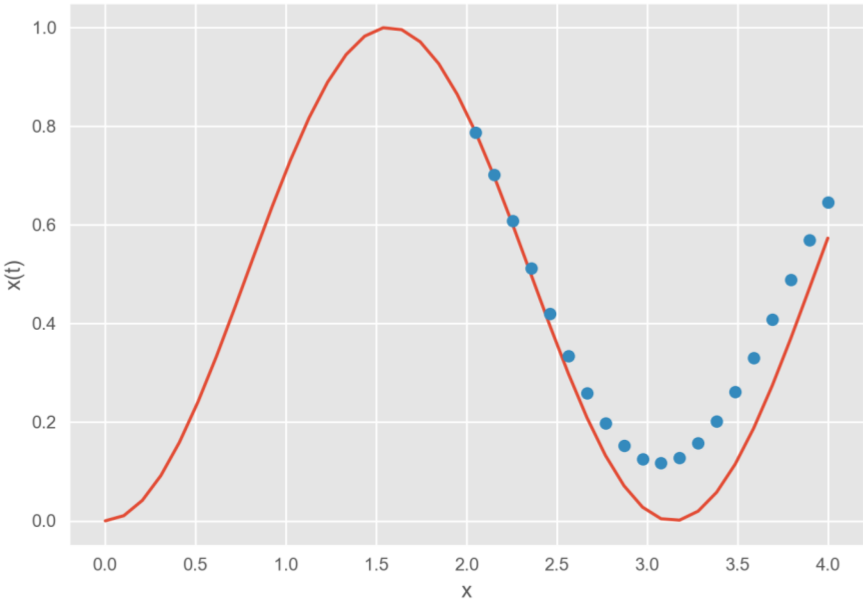

Рисунок 1. Функция и её прогноз при M = 200, $\varepsilon = 0.427$.

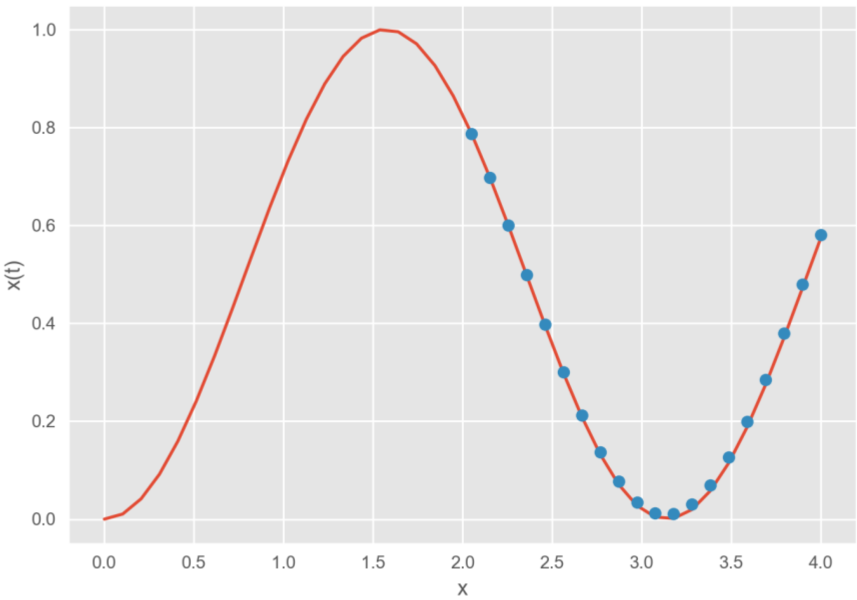

Рисунок 2. Функция и её прогноз при M = 500, $\varepsilon = 0.0021$.

Вектор весовых коэффициентов при M = 500 равен:

W = [0.05285147721580807, -0.5074918755160587, -0.01791562604724876, 0.4735198676502398, 0.9462085955329141]

2. Зависимость погрешности от числа эпох и нормы обучения:

Построим графики зависимости погрешности ε от числа эпох M с различными нормами обучения:

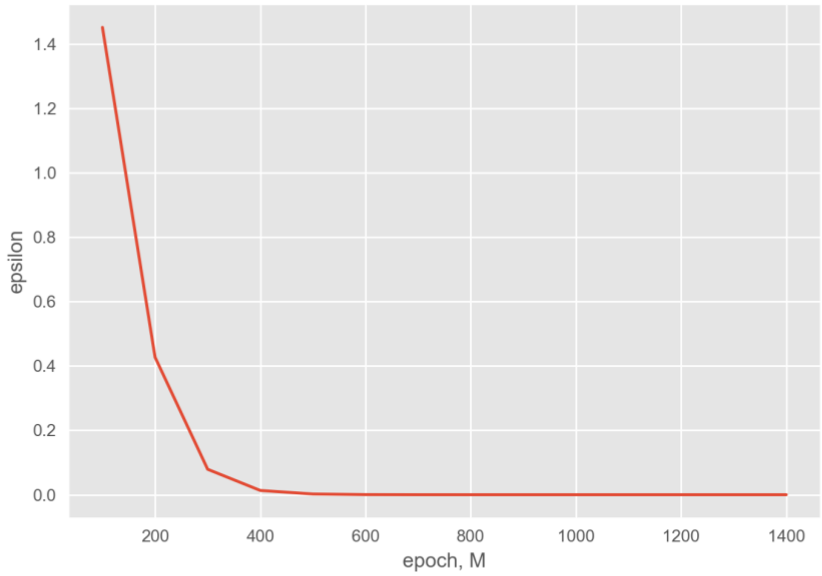

Рисунок 3. Норма обучения $\eta = 0.3$

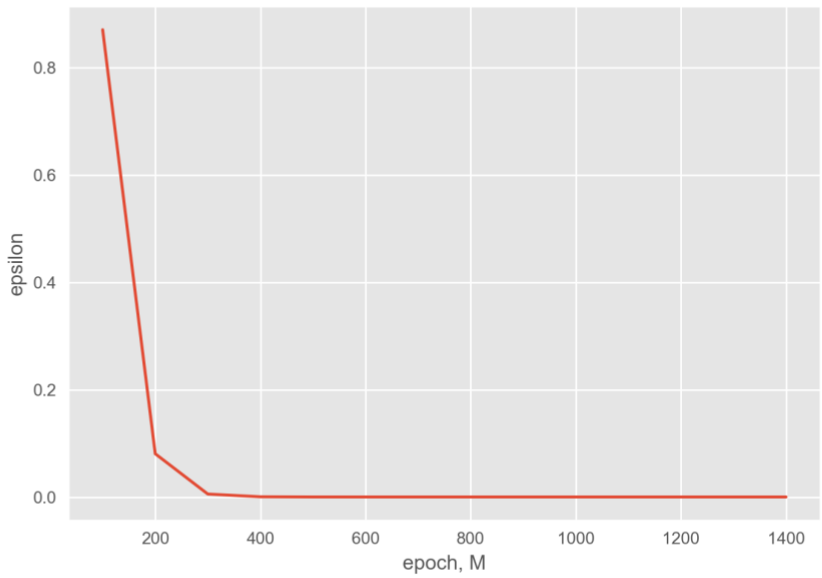

Рисунок 4. Норма обучения η = 0.7.

Построим графики зависимости погрешности ε от нормы обучения η с различным количеством эпох M:

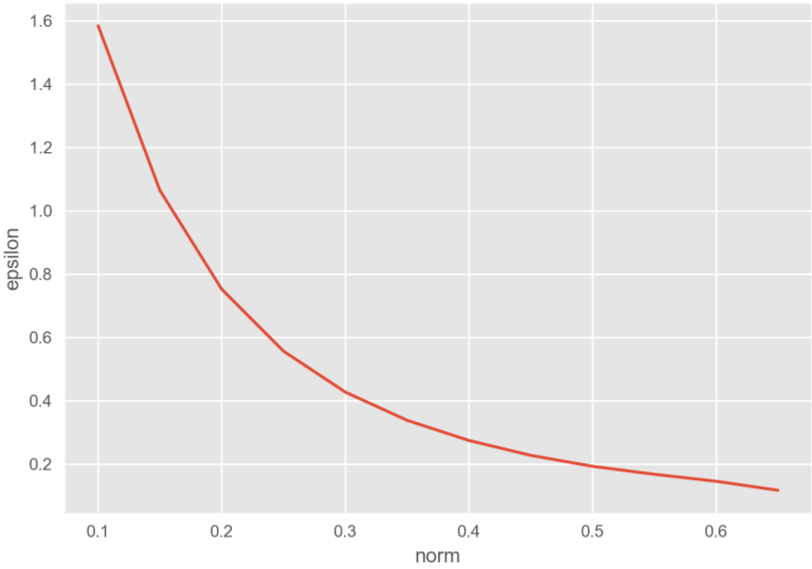

Рисунок 5. M = 200

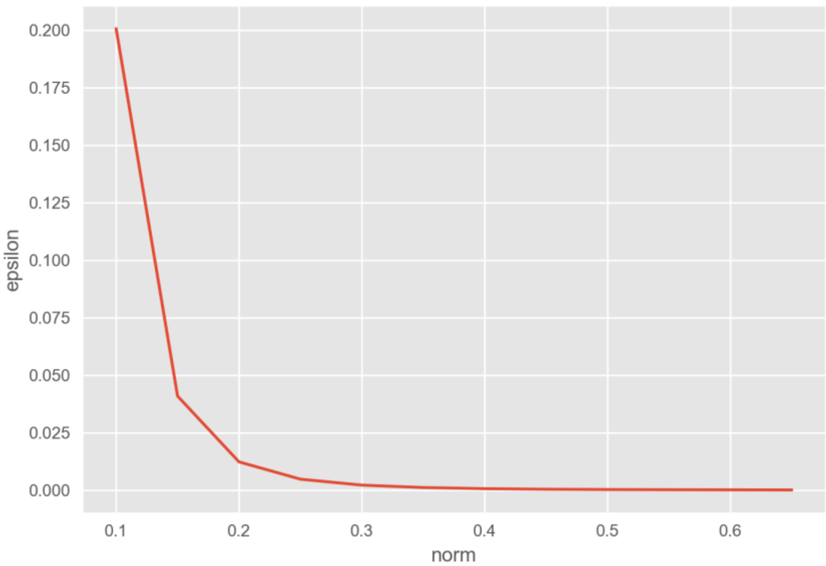

Рисунок 6. M = 500

Попарно сравнив каждый из полученных графиков, можем сделать вывод, что при одинаковом наборе эпох с увеличением нормы обучения увеличивается крутизна графика, что свидетельствует о более быстром достижении меньшей погрешности. Сама же финальная погрешность на последней эпохе становится меньше с ростом нормы обучения.

В случае сравнения графиков с одинаковым набором норм обучения, но разным количеством эпох, наблюдаем очевидную тенденцию: чем больше эпох, тем круче график и меньше финальная погрешность.

3. Прогнозирование при изменении размера окна:

Проведём численный эксперимент, увеличивая размер окна от 4 до 19, на двух случаях:

a) $\boldsymbol{\eta = 0.3}$, M = 500
b) $\boldsymbol{\eta = 0.7}$, M = 500

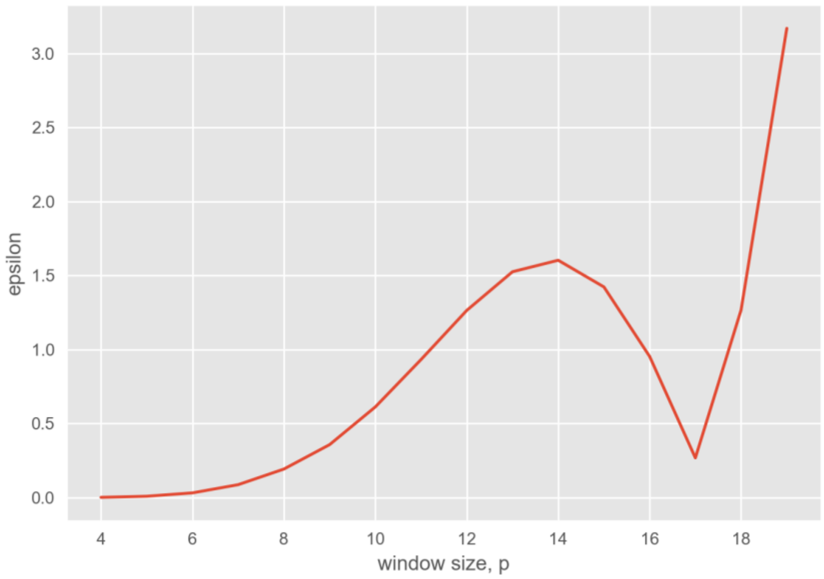

Рисунок 7. Случай a

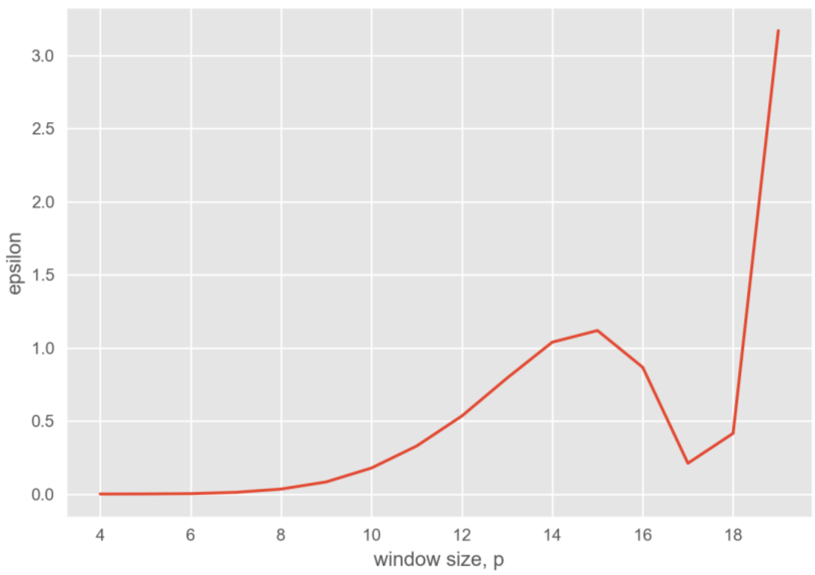

Рисунок 8. Случай b

Проанализировав полученные результаты, можем сделать вывод, что с увеличением размера окна уменьшается точность обучения, следовательно возрастает количество эпох, необходимое для получения приемлемого результата, то есть возрастает погрешность.

Выводы

В ходе выполнения лабораторной работы были изучены возможности однослойных НС в задачах прогнозирования временных рядов методом скользящего окна (авторегрессия).

В ходе выполнения работы пришли к выводу, что число эпох обучения M и норма обучения обратно пропорциональны погрешности ε: чем больше M и норма, тем меньше среднеквадратическая погрешность приближения ε.

Приложение А.

Файл ‘prediction.py’.

```
from computation import *
import numpy as np


'''
Функция обучения
:param points_number: число точек интервала
:param window_size: размер окна
:param norm: норма обучения
:param vector_w: набор весов
:param epoch_max: максимальное число эпох
:param return: набор весов, последняя погрешность
'''
def sliding_window_learn(points_number, window_size, norm, vector_w, epoch_max):
    points = np.linspace(0, 4, 40)
    x = int(window_size)

    values = list()
    for pt in points[0:20]:
        values.append(function(pt))

    epoch = 0

    while epoch < epoch_max:

        s = 0
        f = x

        while f < points_number:

            #подсчёт net
            net = 0
            for (v, w) in zip(values[s:f], vector_w[1:]):
                net += v * w
            net += vector_w[0]

            #Вычисление delta
            delta = values[f] - net

            #Пересчёт весовых коэффициентов
```

```
        for (j, v) in zip(range(1, len(vector_w)), range(s, f)):
            vector_w[j] += get_dw(delta, norm, values[v])
        vector_w[0] += get_dw(delta, norm, 1)

        s += 1
        f += 1

    epoch += 1

  return vector_w


'''
Функция прогнозирования графика
:param points_number: число точек интервала
:param window_size: размер окна
:param vector_w: набор весов
:param return: спрогнозированные значения функции
'''
def predictive_plot(points_number, window_size, vector_w):
  points = pts = np.linspace(0, 4, 40)
  x = int(window_size)

  y = list()
  for pt in points:
    y.append(function(pt))

  values = list()
  new_points = points[(20 - x):20]
  for pt in new_points:
    values.append(function(pt))

  sec_values = list()
  sec_points = points[20:]
  for pt in sec_points:
    sec_values.append(function(pt))

  s = 0
  f = x

  delta_list = list()

  while f < points_number + x:
```

```
        net = 0
        for (v, w) in zip(values[s:], vector_w[1:]):
            net += v * w
        net += vector_w[0]
        values.append(net)

        delta = sec_values[s] - net
        delta_list.append(delta)

        s += 1
        f += 1

    epsilon = get_epsilon(delta_list)

    graph_plot(points, values, y, 20, x, 'x', 'x(t)', 1)

    return values, epsilon


'''
Функция тестирования для отчёта
:param N: число точек интервала
:param ws: размер окна
:param W: набор весов
:param M: максимальное число эпох
'''
def test(N, ws, W, M):
    norm_list = list()
    e_list = list()
    # for epoch in range(100, 1500, 100):
    # for norm in np.arange(0.1, 0.7, 0.05):
    for w_size in range(4, 20):
        W = [0,0,0,0,0]
        new_W = sliding_window_learn(N, w_size, 0.7, W, 500)
        vc, e = predictive_plot(N, w_size, new_W)
        norm_list.append(w_size)
        e_list.append(e)
    style.use('seaborn')
    style.use('ggplot')
    plt.plot(norm_list, e_list)
    plt.xlabel('window size, p')
    plt.ylabel('epsilon')

    plt.grid(True)
```

```
    plt.show()



if __name__ == "__main__":

    N = int(input('Enter N:'))
    M = int(input('Enter M:'))
    p = int(input('Enter sliding window size:'))
    nu = float(input('Enter nu:'))

    W = list()
    for i in range(0, p + 1):
        W.append(0)

    vec = sliding_window_learn(N, p, nu, W, M)
    temp, eps = predictive_plot(N, p, vec)

    W = list()
    for i in range(0, p + 1):
        W.append(0)

    # test(N, p, W, M)
    print(vec)
    print(eps)
```

Файл 'computation.py'.

```
from math import sin, tan
from math import sqrt
import matplotlib.pyplot as plt
import matplotlib.style as style
import matplotlib as mpl


'''
Исходная функция
:param t: параметр t
:param return: значение функции
'''
def function(t):
    return (sin(t)) ** 2
```

```
'''
Функция вычисления эпсилон
:param dlist: набор дельт
:param return: значение эпсилон
'''
def get_epsilon(dlist):
    epsilon = 0
    for delta in dlist:
        epsilon += delta**2
    return sqrt(epsilon)



'''
Функция вычисления коррекции веса
:param delta: дельта
:param return: значение коррекции
'''
def get_dw(delta, norm, x):
    return norm * delta * x



'''
Функция построения графика
:param x_array: значения x
:param y_array1: первый набор значений y
:param y_array2: второй набор значений y (для типа 1)
:param start_x: начальная точка x
:param start_y: начальная точка y
:param label_x: название гор. оси
:param label_y: название верт. оси
:param plot_type: тип графика
'''
def graph_plot(x_array, y_array1, y_array2, start_x, start_y, label_x, label_y, plot_type):
    style.use('seaborn')
    style.use('ggplot')
    plt.grid(True)
    if plot_type == 1:
        plt.plot(x_array, y_array2)
    plt.plot(x_array[start_x:], y_array1[start_y:], 'o')
    plt.xlabel(label_x)
    plt.ylabel(label_y)
    mpl.style.use('bmh')
    plt.show()
```