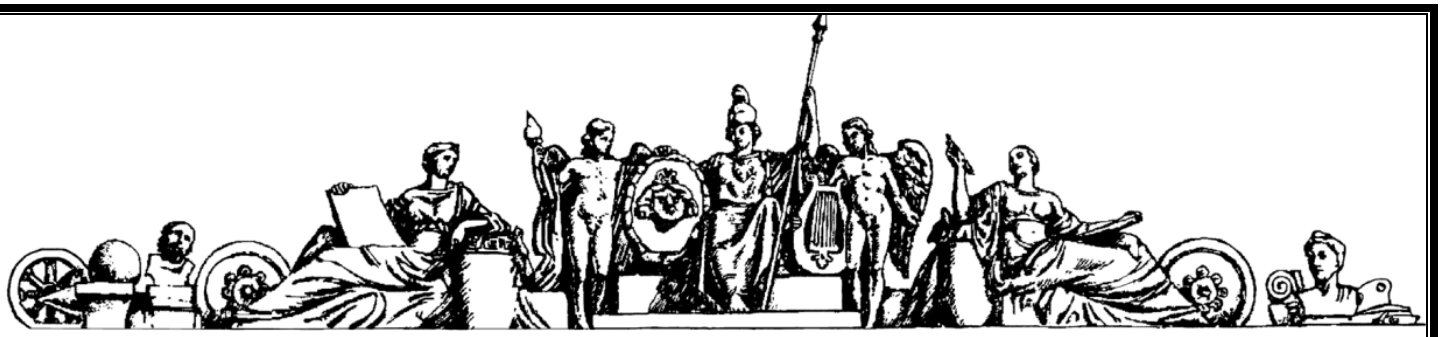

Министерство образования Российской Федерации
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н. Э. БАУМАНА
Факультет: Информатика и системы управления
Кафедра: Информационная безопасность

«Интеллектуальные технологии информационной безопасности»

ЛАБОРАТОРНАЯ РАБОТА № 2

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

Вариант № 6

Преподаватель: Коннова Н.С.
Студент: Кошман А.А.
Группа: ИУ8-61

Москва, 2019

Оглавление

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 *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 функции активации. Исходное количество нейронов (длина окна) *p = 4*. Обучение проводить методом Видроу - Хоффа. Исследовать влияние количества эпох M обучения и коэффициента обучения n на средне-квадратичную погрешность приближения $\varepsilon = \sqrt{\sum_i [x(t_i) - \check{x}(t_i]^2}$.

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

Условие

$$X(t) = \quad 0,4\sin 0,3t + 0,5$$

График данной функции на интервале [0, 2] представлен на рис. 1.

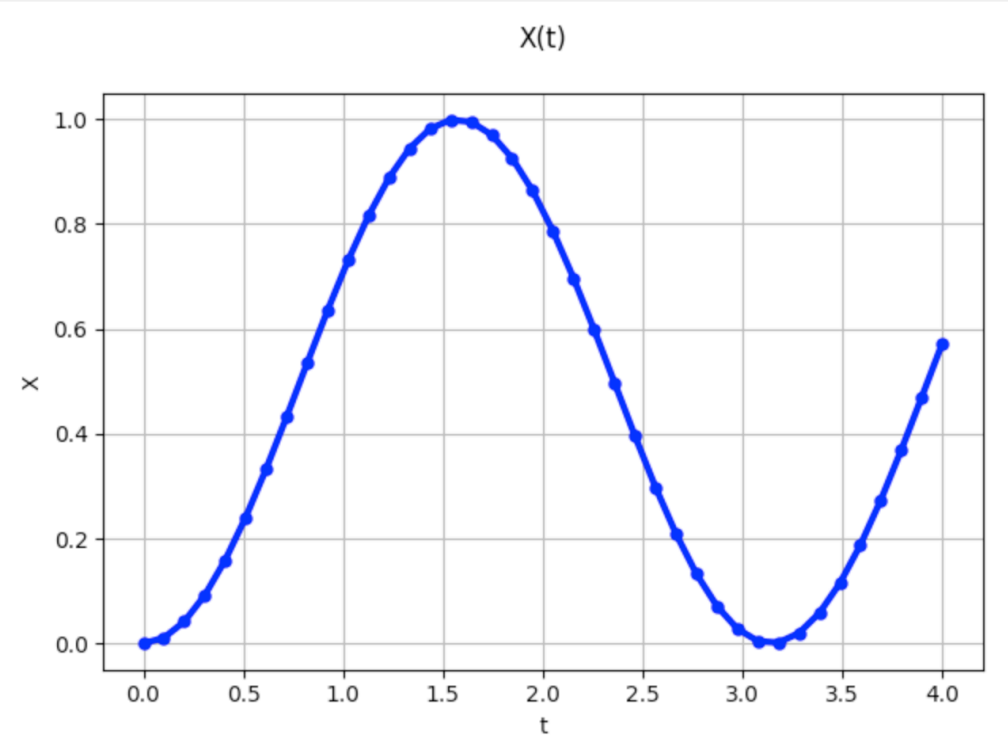

Рисунок 1. График функции *X(t)*

Результаты эксперимента

Задание № 1. Исследование процесса прогнозирования при постепенном изменении размера окна p.

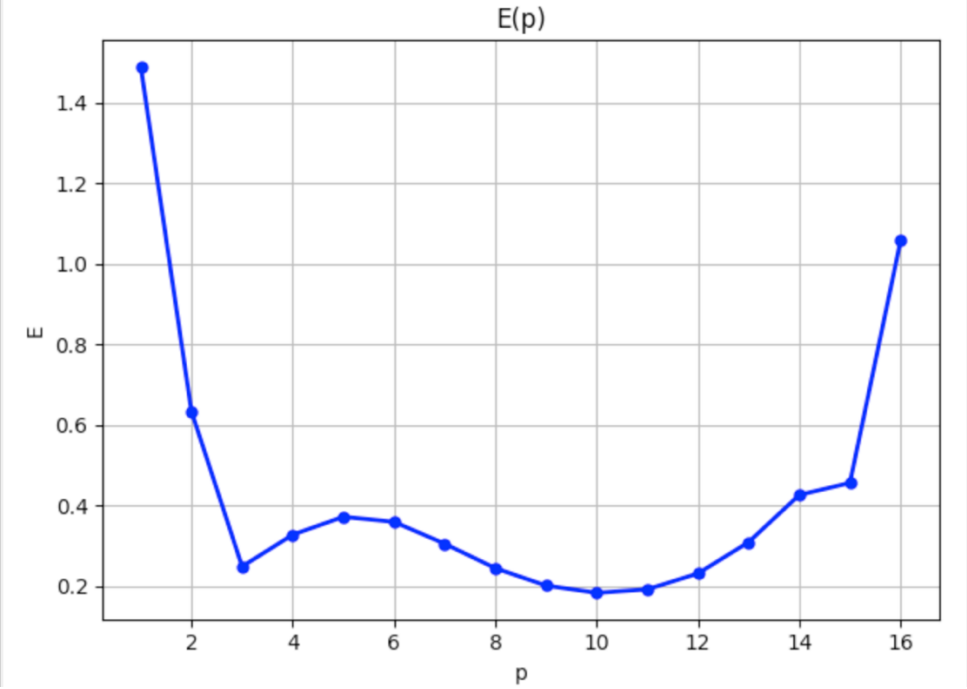

Рисунок 2. Зависимость средне-квадратичной погрешности E от ширины окна p

Задание № 2. Исследование влияния коэффициента обучения η на средне-квадратичную погрешность приближения.

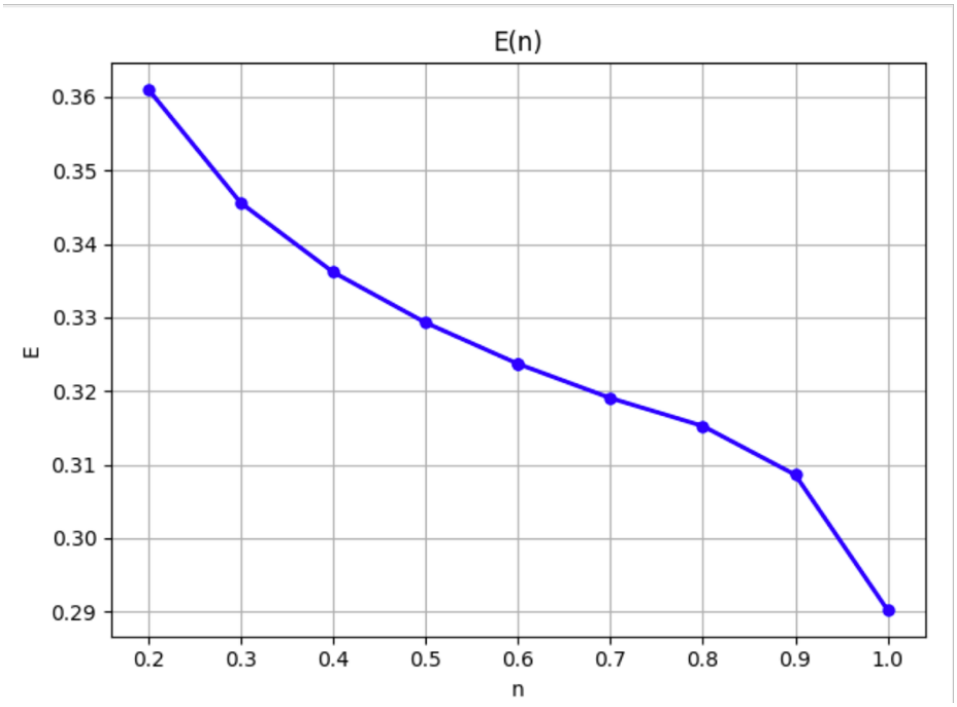

Рисунок 3. Зависимость средне-квадратичной погрешности E от нормы обучения η

Задание № 3. Исследование влияния количества эпох M обучения на средне-квадратичную погрешность приближения.

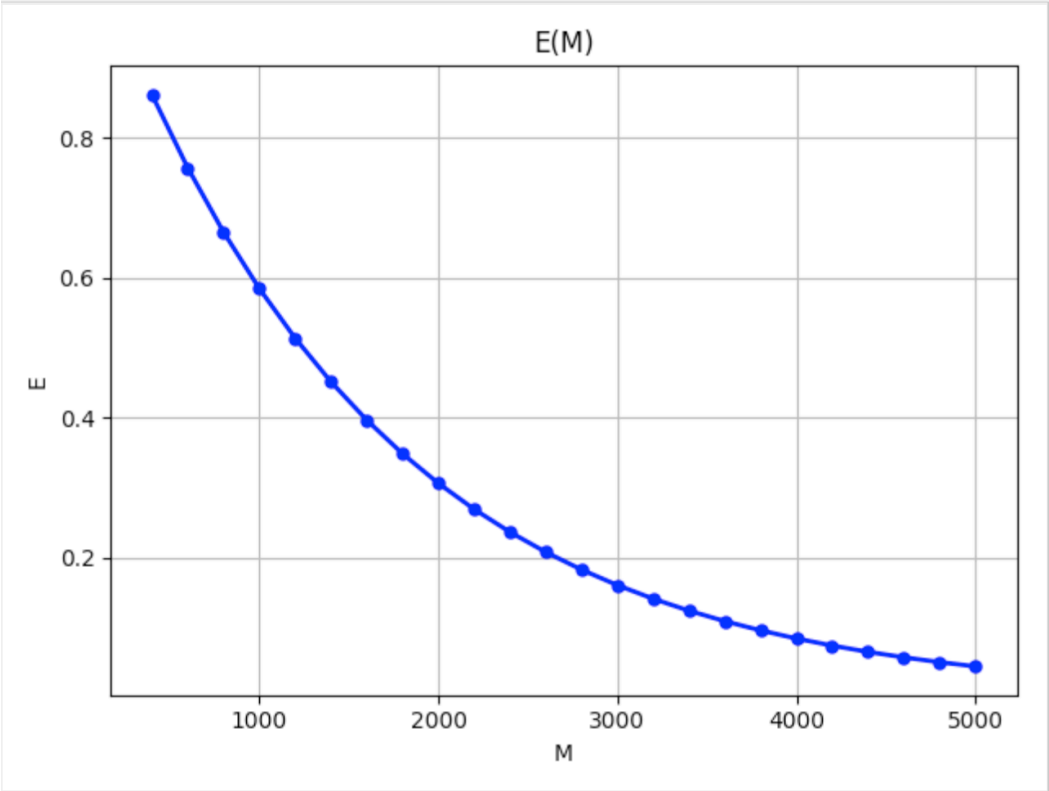

Рисунок 4. Зависимость средне-квадратичной погрешности *E* от количества эпох *M*

На рис. 5 представлены графики исходной и обученной функции в зависимости от временного ряда. Синим цветом обозначен график исходной функции, красным – полученной. Интервал, на котором обучалась НС, обозначен голубым пунктиром. Зеленым пунктиром показана ширина окна.

НС в данном примере обучалась 30000 эпох, с шагом обучения $n = 0.3$, при ширине окна $p = 4$.

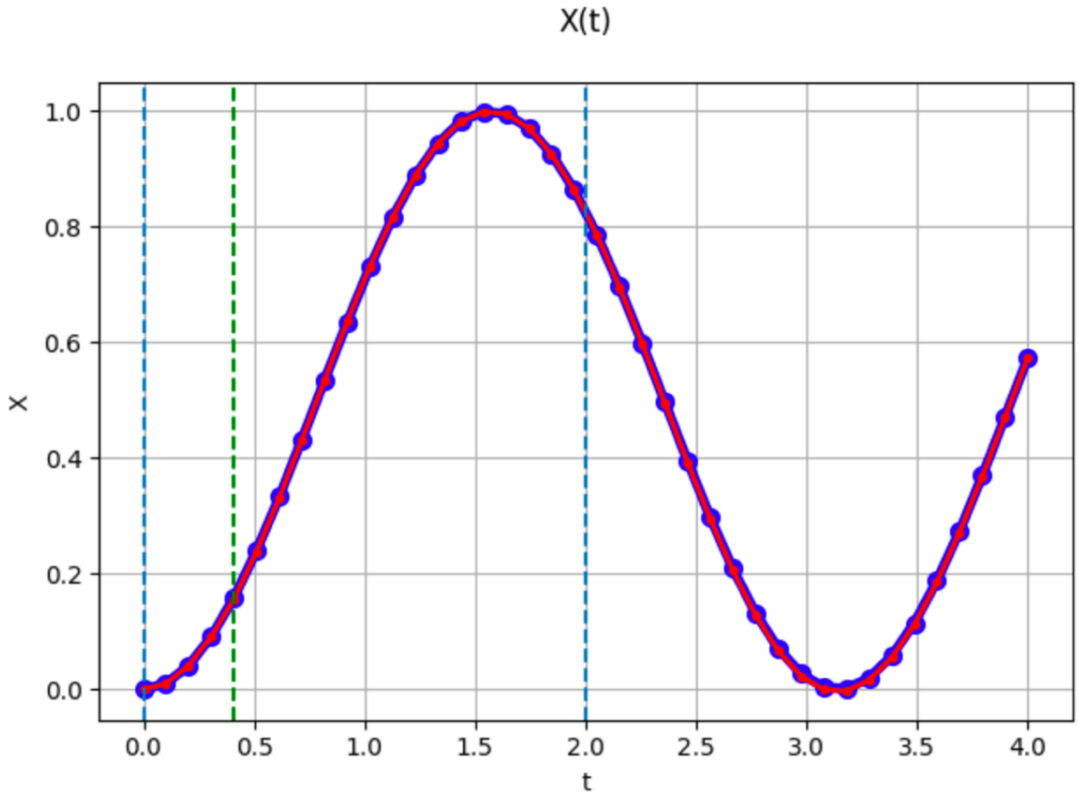

Рисунок 5. Графики исходной и обученной функции в зависимости от временного ряда.

Как видно из графика, при заданных параметрах НС полностью обучилась.

Выводы

В процессе лабораторной работы было изучены возможности однослойных НС в задачах прогнозирования временных рядов методом скользящего окна.

Было установлено, что при увеличении ширины окна средне-квадратичная погрешность сначала уменьшается, при $p = 10$ она достигает своего минимального значения, а затем снова увеличивается.

При увеличении нормы обучения η значения средне-квадратичной погрешности уменьшались.

Сравнивая результаты прогноза при различном количестве эпох, следует отметить, что его качество неудовлетворительно примерно до $M \approx 3000$, а затем быстро улучшается, и при $M > 4000$ средне-квадратичная погрешность $E < 0,1$. При $M = 30000$ прогнозные значения полностью совпадают с точками в пределах графического изображения.

Приложения

Файл ‘script.py’ :

```
import numpy as np
import matplotlib.pyplot as plt

def X(T): # возвращает результат моделируемой функции от значения времени
    return [np.sin(t) ** 2 for t in T]

N = 20 # количество точек

a = 0 # интервал, на котором обучается НС
b = 2

T = list(np.linspace(a, 2 * b - a, 2 * N)) # временной ряд

RightX = X(T) # значения исходной функции

TryX = [0 for i in range(N)]

def DeltaW(x, q, n): # находит величину, на которую изменятся Wi
    return n * q * x

def Net(x, w): # находит значение сетевого входа НС
    return sum([w_i * x_i for w_i, x_i in zip(w, x)])

def MeanSquareError(RightX, TryX, p): # нахождение средне-квадратичной погрешности e
    summa = 0
    for rx_i, tx_i in zip(RightX[p:], TryX[p:]):
        summa += (rx_i - tx_i) ** 2
    return summa ** 0.5

def Learning(p, n, m): # обучение НС методом скользящего окна

    for k in range(0, p): TryX[k] = RightX[k]
    w = [0 for i in range(p)]
    e = 0
    era = 0

    while(era < m):

        for l in range(p, N): # 16 шагов эпохи
            TryX[l] = Net(RightX[l - p:l-1], w)
            q = RightX[l] - TryX[l]
            for k in range(0, p):
                w[k] += DeltaW(RightX[l - p + k], q, n)

        e = MeanSquareError(RightX, TryX, p)

        print("\nera = ",  np.round(era, 3))
        print("TryX : ", np.around(TryX, 3))
        print("w : ", np.around(w, 3))
        print("e = ", np.round(e, 3))
        era += 1

    return list(TryX), w

def Graph(TryX, p, arg = "", name = ""): # строит 2 графика : исходной и полученной функции в
зависмости от временного ряда
    fig, ax = plt.subplots()
    ax.plot(T, RightX, 'bo-', linewidth=4, markersize=7)
    ax.plot(T, TryX, 'ro-', linewidth=2, markersize=3)
    plt.title("X(t)\n" + name + str(arg))
    plt.xlabel('t')
    plt.ylabel('X')
    plt.axvline(x=a,  linestyle='--')
    plt.axvline(x=b, linestyle='--')
    plt.axvline(x=T[p], linestyle='--', color = 'g')
    plt.grid(True)
```

```
    plt.show()

def Graph_E(e, arg, name):  # строит график зависимости ср.-кв. погрешности от arg : n, p, m
    plt.plot(arg, e, 'bo-', linewidth=2, markersize=5)
    plt.title("E(" + name + ")")
    plt.xlabel(name)
    plt.ylabel('E')
    plt.grid(True)
    plt.show()

def Forecast(E, n, p, m): # прогнозирование

    TryX, w = Learning(p, n, m)

    TryX.extend(np.zeros(N))

    for l in range(N, 2 * N):
        TryX[l] = Net(RightX[l - p: l - 1], w)

    E.append(MeanSquareError(RightX, TryX, p))

    return TryX

if __name__=="__main__":

    med_n = 0.3   # норма обучения
    range_n = np.around(np.linspace(0.2, 1, 10), 1)

    med_p = 4 # размер "окна" данных
    range_p = range(1, 17)

    med_m = 1600 # количество эпох
    range_m = range(400, 2001, 100)

    E = []

    for p in range_p:

        print("\n\np = ", p)

        Forecast(E, med_n, p, med_m)

    Graph_E(E, range_p, "p")

    E.clear()

    for n in range_n:

        print("\n\nn = ", n)

        Forecast(E, n, med_p, med_m)

    Graph_E(E, range_n, "n")

    E.clear()

    for m in range_m:

        print("\n\nM = ", m)

        Forecast(E, med_n, med_p, m)

    Graph_E(E, range_m, "M")

    Graph(Forecast(E, med_n, med_p, 30000), med_p)
```