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 4

Преподаватель:
Коннова Н.С.

Студент:
Куликова А.В.

Группа:
ИУ8-21М

Цель работы

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

Постановка задачи

№ Варианта	Функция x(t)	a	b
4	$0,5\exp(0,5\cos 0,5t)+\sin 0,5t$	-5	3

Ход работы

В ходе исследования был проведен ряд испытаний, на основе которых были получены оптимальные значения НС для предсказания графика функции.

Было проведено исследования зависимости размера ошибки от ширины окна. Как видно из рисунка 1, оптимальная ширина окна – 10.

После нахождения оптимальной ширины окна и нормы обучения, была обучена нейронная сеть по данным параметрам. Результат обучения представлен на рисунке 1. Зависимость ошибки от количества итераций указан на рисунке 2. Начиная с 200 итерации – размер ошибки уменьшается незначительно. Таким образом найдет оптимальный норм критериев.

Результат программы представленной в приложении А:

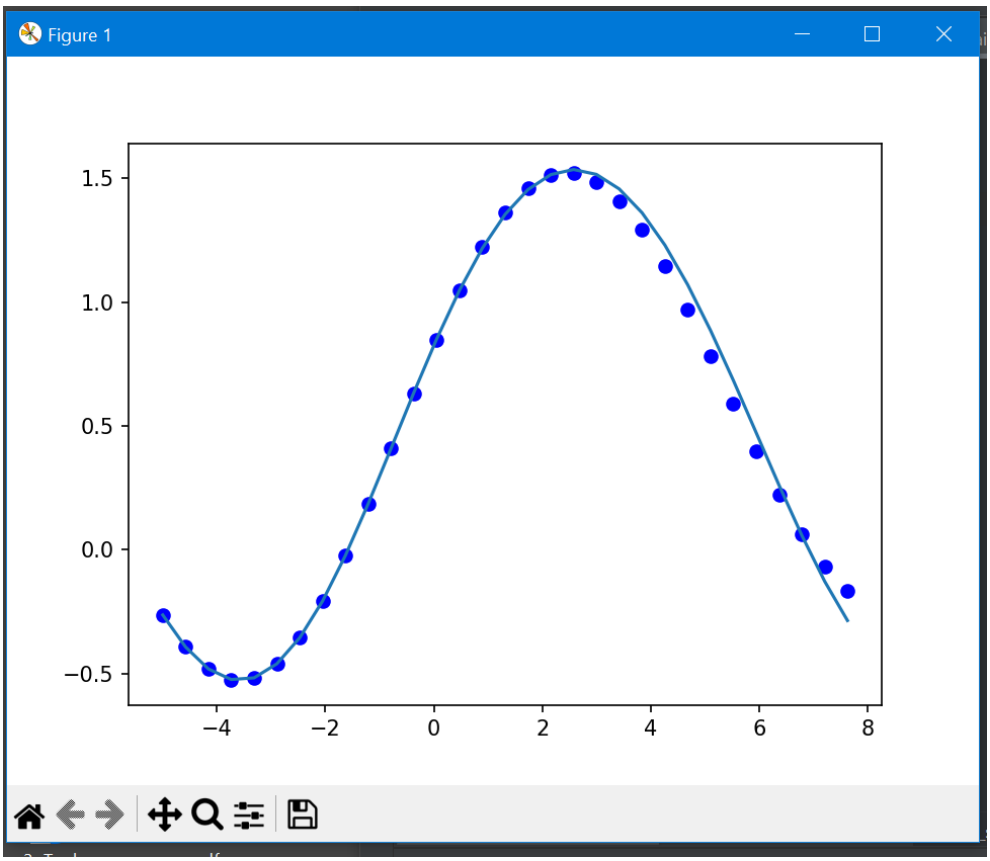

Рисунок 1 - Результат работы НС

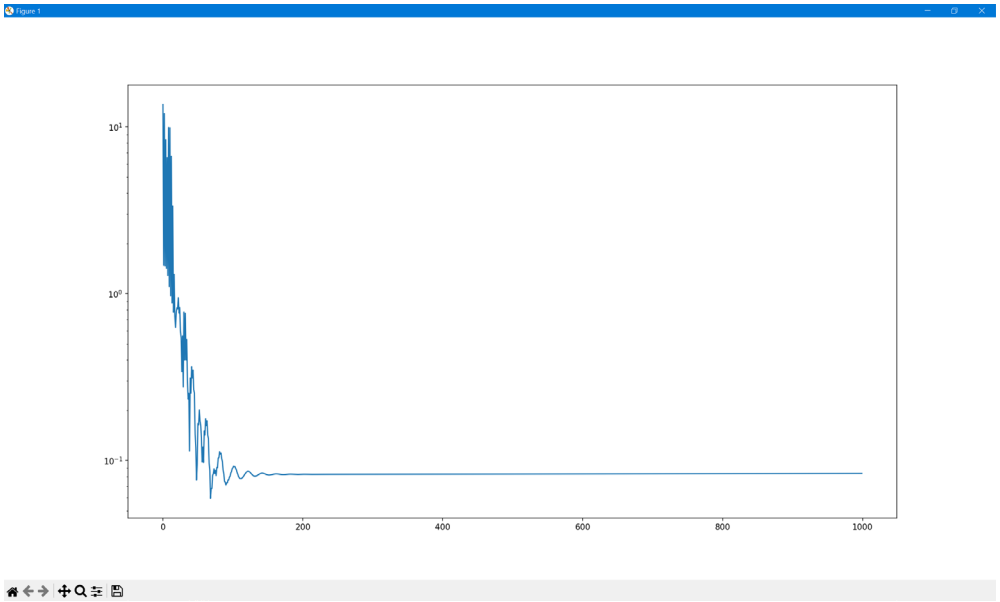

Рисунок 2 - Отношение ошибки к количествам эпох обучения

Создана нейронная сеть со следующими параметрами:
Вектор весов: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
Размер конечной ошибки: 0.084

Выводы

В ходе лабораторной работы было доказано, что ошибка имеет свойство изменяться в зависимости от количества итераций, ширины окна, а также нормы обучения. Помимо этого, выведены наилучшее входные данные для НС.

Контрольные вопросы

1. В чем состоит принцип прогнозирования на основе авторегрессии?

Одной из простейших моделей прогноза является авторегрессионная модель, когда прогнозируемое значение ряда в момент времени $n > m$ выражается через его известные значения в предыдущие моменты времени:

$$\tilde{x}_n = \sum_{k=1}^{p} w_k x_{n-p+k-1}, \qquad (3.1)$$

где p — размер «окна» данных, по которому производится прогноз; w_k — некоторые весовые коэффициенты.

Выражение (3.1) часто используется для центрированных временны́х рядов, среднее значение которых равно 0. Если математическое ожидание временно́го ряда равно $\bar{x}$, то вместо (3.1) можно воспользоваться представлением

$$\tilde{x}_n = \sum_{k=1}^{p} w_k x_{n-p+k-1} + w_0, \qquad (3.2)$$

где $w_0 = \bar{x}$.

Ошибка прогноза (локальная) равна апостериорной разнице спрогнозированного и реального значений временно́го ряда:

$$\delta_n = x_n - \tilde{x}_n.$$

2. Объясните методику обучения НС прогноза.

Простейшая архитектура НС показана на рис. 3.1. Функция активации — линейная, т. е.

$$f(\text{net}) = \text{net},$$

где net — комбинированный вход единственного нейрона,

$$\text{net} = \sum_{k=1}^{p} w_k x_{i+k-1} + w_0.$$

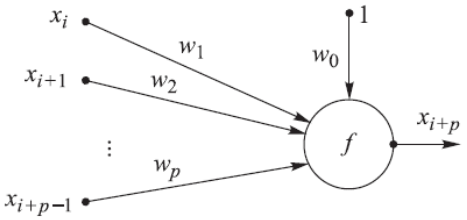

Рис. 3.1. Архитектура нейронной сети

В режиме обучения последовательно, от эпохи к эпохе, на вход сети подаются элементы векторов-столбцов обучающей выборки

$$\mathbf{X} = \begin{pmatrix} x_1 & x_2 & \dots & x_{m-p} \\ x_2 & x_3 & \dots & x_{m-p+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_p & x_{p+1} & \dots & x_{m-1} \end{pmatrix}, \qquad (3.4)$$

а на выходе соответственно получаются прогнозируемые значения:

$$\tilde{x}_{p+1}, \tilde{x}_{p+2}, \dots, \tilde{x}_{m-1}.$$

Эти значения сравниваются с реальными

$$x_{p+1}, x_{p+2}, \dots, x_{m-1}$$

и по (3.3) оценивается ошибка. Коррекция весов на каждой эпохе производится по правилу Видроу — Хоффа:

$$\Delta w_k = \eta \delta x_k, \quad k = \overline{0, p},$$

где η — норма обучения, $\eta \in (0, 1]$.

Если по достижении правого края выборки (3.4) суммарная среднеквадратичная ошибка $\varepsilon = \sqrt{\sum_i [x(t_i) - \tilde{x}(t_i)]^2}$ останется достаточно большой, следует продолжить обучение, снова вернувшись к первому столбцу (3.4) и т. д.

3. Поясните принцип функционирования НС прогноза.

НС прогноза. Нахождение неизвестных весовых коэффициентов w_k осуществляется по известной обучающей выборке значений временно́го ряда $x_i = x(t_i)$ $(i = \overline{1, m})$. Для этого необходимо определить окно длиной $p < m$. Далее, начиная с левого края временно́го ряда, следует прогнозировать его значения в моменты, идущие непосредственно за окном (справа от него). Сравнивая этот прогноз с реальными значениями, можно оценить ошибку (3.3), а на ее основе — скорректировать весовые коэффициенты w_k, например, с помощью методов обучения НС.

Приложение А

Листинг программы:

```
import math
import matplotlib.pyplot as plt

class Network:

    """Инициализация нейронной сети с заданными параметрами."""
    def __init__(self, synapses: list, sample: list, N, p, end, a, b):
        self.synapses = synapses
        self.sample = sample
        self.errors = []
        self.N = N
        self.p = p
        self.end = end
        self.a = a
        self.b = b
        print("Создана нейронная сеть со следующими параметрами: ")
        print("Вектор весов: ", self.synapses)

    """ Рассчитывает выходной сигнал нейронной сети для заданных входных
данных. """
    def iteration(self, inputs: list):
        net = 0
        for i in range(len(self.synapses)):
            net += self.synapses[i] * inputs[i]
        return net

    """ Прогнозирует следующие значения на заданное количество шагов. """
    def predict(self, count_steps: int, previous: list):
        out = []
        for i in range(count_steps):
            previous = previous[1:] + [self.iteration([1] + previous)]  #
сдвигаем окно и добавляем предсказание
            out.append(previous[-1])
        return out

    """ Обучает нейронную сеть на основе предоставленных обучающих данных.
"""
    def learning(self, p, norm_learning):
        error = 0
        for i in range(len(self.sample)-1):
            err = self.sample[i+1][-1] - self.iteration(self.sample[i])
            for j in range(p+1):
                self.synapses[j] += norm_learning * err * self.sample[i][j]
            error += (err**2)
        self.errors.append(self.get_err())
        # self.errors.append(math.sqrt(error))

    def get_err(self):
        predicted = self.predict(self.N-self.p+self.end, [func(self.a +
(self.b-self.a)/(self.N-1) * i) for i in range(self.p)])
        ideal = [func(self.a + (self.b-self.a)/(self.N-1) * i) for i in
range(self.p, self.N+self.end)]
        # ideal = ideal[-len(predicted):]
        # print(ideal)
        # print(predicted)
        return math.sqrt(sum([(ideal[i] - predicted[i])**2 for i in
range(len(predicted))])/self.p)

def func(x):
```

```
    return (float)(0.5 * math.exp(0.5 * math.cos(0.5 * x)) + math.sin(0.5 *
x))

""" обучающая выборка.создаются списки Y(x) со сдвигом на 1 каждый """
def create_sample(m, p, X, out=False):
    sample = []
    # создаем выборку
    for i in range(m-p):
        sample.append([1])  # Добавляем единицу в качестве первого элемента
каждого образца
        for j in range(p):
            sample[i].append(X[j+i])  # Добавляем p элементов из входного
массива X в образец
    # вывод выборки при необходимости в читаемом виде
    if out:
        for i in range(m-p):
            print(sample[i])  # Выводим образцы, если параметр out равен True
    return sample  # Возвращаем сформированную выборку

def main():
    a = -5
    b = 3
    p = 10  # Размер окна
    NORM_LEARNING = 0.5  # Норма обучения
    ITERATIONS = 1000  # Количество итераций обучения

    N = 20  # Количество шагов
    end = 2 * b - a  # Конец прогнозируемой функции
    T = end
    step = (b-a)/(N-1)  # Расстояние между t
    X = [func(a + step * i) for i in range(N)]  # Список прогнозируемых точек

    Net = Network([0] * (p + 1), create_sample(N, p, X), N, p, T, a, b)  #
Создаем НС
    for i in range(ITERATIONS):
        Net.learning(p, NORM_LEARNING)  # Обучаем НС
    print("Размер конечной ошибки: {}".format(round(Net.errors[-1], 3)))  #
Выводим размер конечной ошибки

    x = [(a + step * i) for i in range(N+T)]
    y = Net.predict(N+T-p, [func(a + step * i) for i in range(p)])  #
Предсказываем значения

    # Добавляем значения из окна для красивого вывода
    y = [func(a + step * i) for i in range(p)] + y

    plt.plot(x, y, 'bo')  # Выводим предсказанные точки
    y = [func(a + step * i) for i in range(N+T)]
    plt.plot(x, y)  # Выводим исходную функцию
    plt.show()  # Отображаем график

    plt.semilogy([i for i in range(len(Net.errors))], Net.errors)  # Строим
график логарифмической ошибки
    plt.show()  # Отображаем график


if __name__ == "__main__":
    main()
```