**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №5

по дисциплине: «Введение в нейронные сети»

на тему: **«**Анализ временных рядов и аппроксимация**»**

Выполнил: студент гр. ИТП-31

Бортновская В.В.

Принял: преподаватель

Гуменников Е.Д.

Гомель 2022

**Цель работы**: разработать программное обеспечение для аппроксимации заданной функции радиально базисной сетью и персептроном с одним скрытым слоем. Ознакомиться с применением искусственных нейронных сетей для решения задач экстраполяции временных рядов. Исследовать качество экстраполяции от представления данных, глубины погружения временного ряда и структуры нейронной сети.

**Задание:**

Для аппроксимации использовать радиально базисную сеть, представленную на рисунке 1.

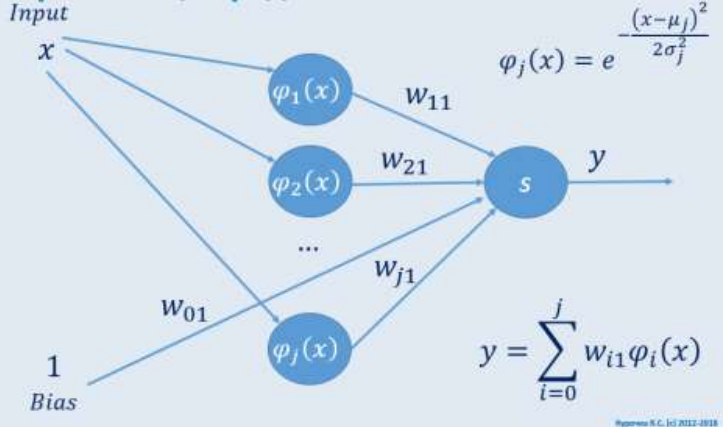


Рисунок 1 – Радиально-базисная сеть

В качестве второй сети использовать персептрон, представленный на рисунке 2.

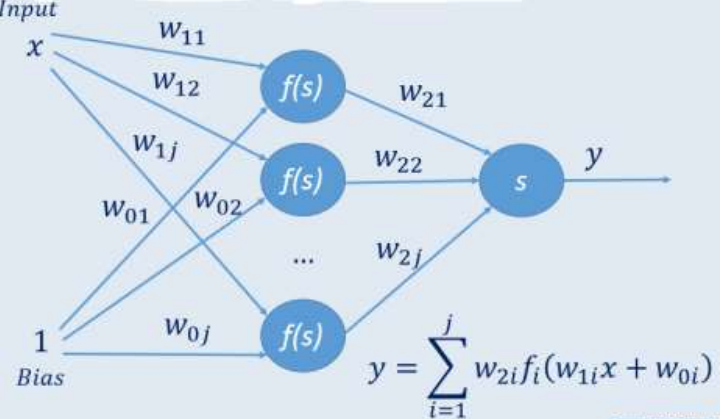


Рисунок 2 – Персептрон

Отобразить результаты аппроксимации графически на одном поле. Дополнить персептрон ещё одним скрытым слоем. Сравнить результаты аппроксимации с помощью радиально-базисной сети, однослойным персептроном и двухслойным персептроном.

Исследовать влияние количества нейронов на скрытом слое на погрешность аппроксимации. Сделать выводы.

Функция:

Количество нейронов на скрытом слое: 5/7.

Используя метод скользящего окна выполнить экстраполяцию данных на январь месяц 2021 года персептроном (рисунок 3).

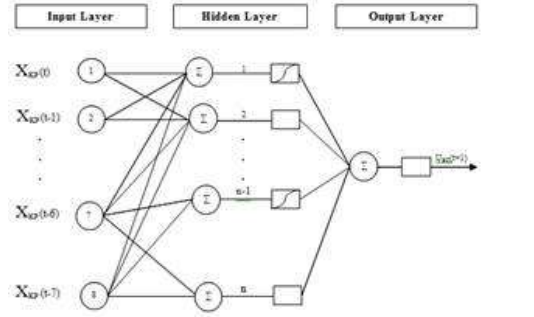


Рисунок 3 – Персептрон со скользящим окном

В качестве второй сети для экстраполяции данных использовать сеть Элмана (рисунок 4).

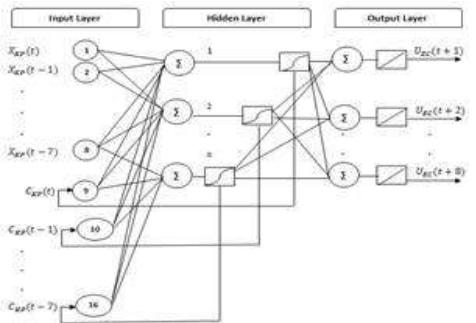


Рисунок 4 – Сеть Элмана

Исследовать качество экстраполяции от представления данных, глубины погружения временного ряда и структуры нейронной сети.

Экстраполируемые данные: температура воздуха в Речице.

**Ход работы**

На рисунке 5 представлено меню программы.

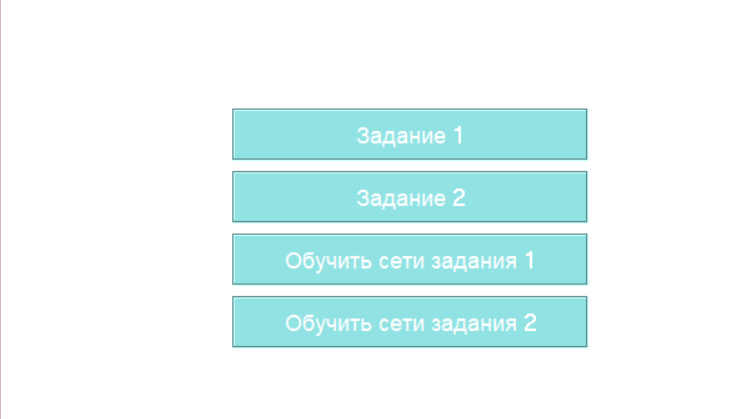


Рисунок 5 – Меню программы

На рисунке 6 представлен результат аппроксимации.

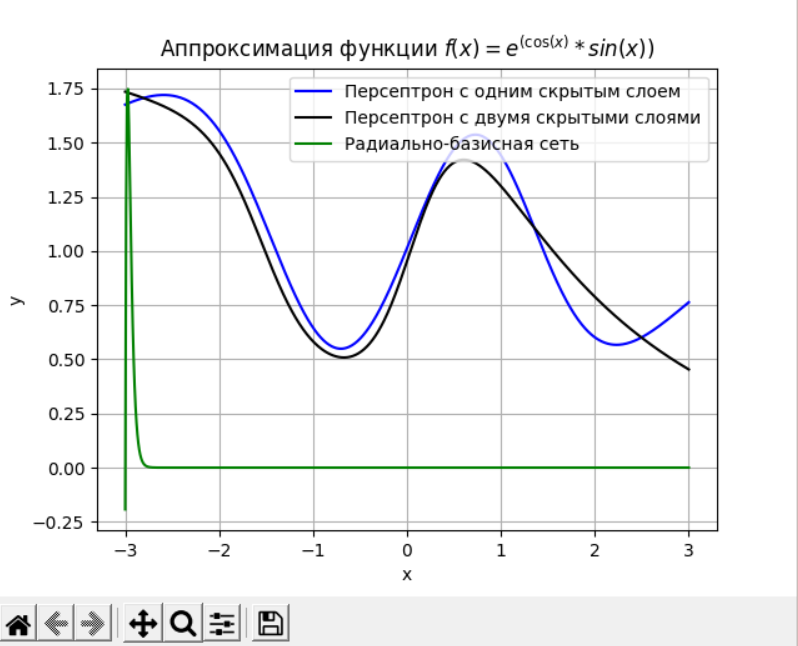


Рисунок 6 – Аппроксимация функции

На рисунке 7 представлен пример файл с экстраполируемыми данными.

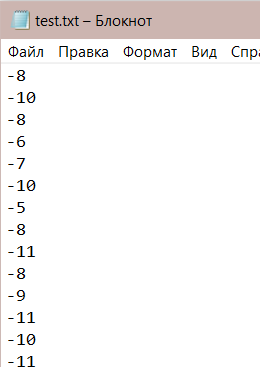


Рисунок 7 – Пример файла с экстраполируемыми данными

На рисунке 8 представлен результат экстраполяции временного ряда.

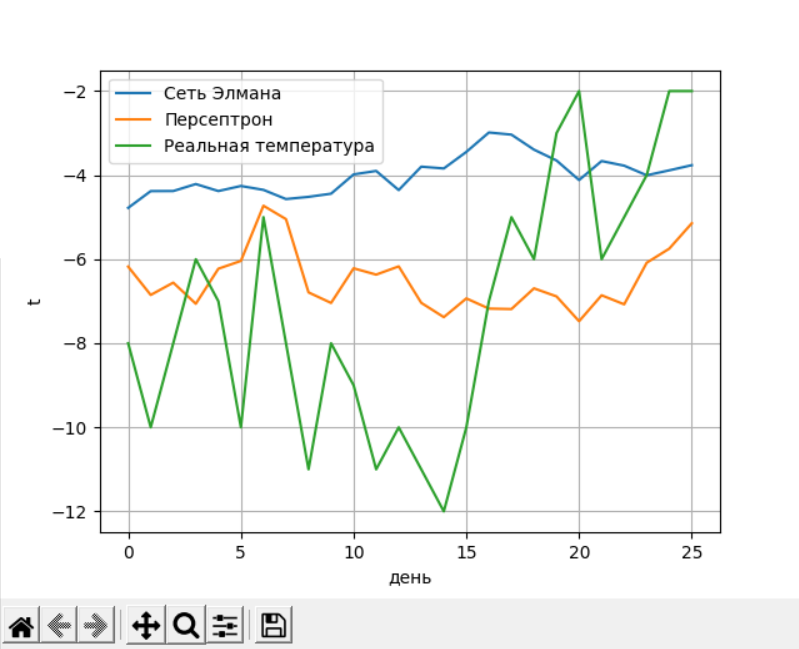


Рисунок 8 – Результат экстраполяции

Радиально-базисная сеть заняла меньше всего строчек кода.

Листинг программы представлен в приложении А.

**Вывод:** в ходе выполнения лабораторной работы изучена аппроксимация и экстраполяция с помощью нейронных сетей, выявлено что количество скрытых слоев не оказывает особого влияния на результат аппроксимации, в аппроксимации данных более хорошие результаты продемонстрированы персептроном, в экстраполяции персептрон и сеть Элмана показали в целом схожие результаты.

**ПРИЛОЖЕНИЕ А**

(обязательное)

**Листинг программы**

# Подключение библиотек

import numpy as np

import matplotlib.pyplot as plt

import NeuralNetwork as nn

def deserialize(filename):

    with open(filename, "r") as file:

        rows = file.read().split()

        data = []

        for i, k in enumerate(rows[:-1]):

            data.append(float(k))

    return data

class RBFNet:

    def \_\_init\_\_(self, hidden\_number, sigma=1.0):

        self.hidden\_number = hidden\_number

        self.sigma = sigma

        self.centers = 0

        self.weights = 0

    def rbf(self, point, center):

        return np.exp(np.linalg.norm((point - center) \*\* 2 / (2 \* self.sigma \*\* 2)))

    def calculate\_interpolation\_matrix(self, x):

        g = np.zeros((len(x), self.hidden\_number))

        for i, point in enumerate(x):

            for j, center in enumerate(self.centers):

                g[i, j] = self.rbf(point, center)

        return g

    def fit(self, x, y):

        self.centers = x[np.random.choice(len(x), self.hidden\_number)]

        g = self.calculate\_interpolation\_matrix(x)

        inv\_g = np.linalg.pinv(g)

        self.weights = inv\_g @ y

    def predict(self, x):

        g = self.calculate\_interpolation\_matrix(x)

        return g @ self.weights

class RNN:

    def \_\_init\_\_(self, input\_number, hidden\_number, output\_number, lr=0.01):

        self.lr = lr

        self.w\_ih = np.random.uniform(-np.sqrt(1 / input\_number), np.sqrt(1 / input\_number),

            size=[input\_number, hidden\_number])

        self.b\_ih = np.random.uniform(size=[1, hidden\_number])

        self.w\_hh = np.random.uniform(-np.sqrt(1 / hidden\_number), np.sqrt(1 / hidden\_number),

            size=[hidden\_number, hidden\_number])

        self.b\_hh = np.random.uniform(size=[1, hidden\_number])

        self.h = np.zeros(shape=[1, hidden\_number])

        self.h\_t\_1 = np.zeros(shape=[1, hidden\_number])

        self.w = np.random.uniform(-np.sqrt(1 / hidden\_number), np.sqrt(1 / hidden\_number),

            size=[hidden\_number, output\_number])

    def forward\_prop(self, x):

        self.x = x

        self.h\_t\_1 = self.h

        self.h = self.x @ self.w\_ih + self.b\_ih + self.h\_t\_1 @ self.w\_hh + self.b\_hh

        self.h = np.tanh(self.h)

        self.out = self.h @ self.w

        return self.out

    def \_\_call\_\_(self, \*args):

        return self.forward\_prop(\*args)

    def backward\_prop(self, y):

        dloss = self.out - y

        self.dw = self.h.T @ dloss

        dh = dloss @ self.w.T

        grad = (1 - np.tanh(self.h) \*\* 2) \* dh

        self.dw\_ih = self.x.T @ grad

        self.db\_ih = 1 \* grad

        self.dw\_hh = self.h\_t\_1.T @ grad

        self.db\_hh = 1 \* grad

    def update\_weights(self):

        self.w -= self.lr \* self.dw

        self.w\_ih -= self.lr \* self.dw\_ih

        self.b\_ih -= self.lr \* self.db\_ih

        self.w\_hh -= self.lr \* self.dw\_hh

        self.b\_hh -= self.lr \* self.db\_hh

def draw\_menu():

    print("Меню")

    print("1. Задание 1. Обучение сетей")

    print("2. Задание 1. График аппроксимации")

    print("3. Задание 1. Сравнение сетей с разным кол-вом нейронов на скрытом слое")

    print("4. Задание 2. Обучение сетей")

    print("5. Задание 2. График курса евро к доллару")

    print("6. Задание 2. График предсказаний на январь")

    print("7. Выйти")

def menu\_item\_1(fx, rbf, perceptron\_1, perceptron\_2, perceptron\_1x2, perceptron\_1x3):

    x = np.random.uniform(-4.5, 4.5, size=[75000, 1])

    y = fx(x)

    rbf.sigma = np.std(y)

    rbf.fit(x, y)

    for i in range(75000):

        xi = x[i].reshape(1, 1)

        yi = y[i].reshape(1, 1)

        perceptron\_1.forward\_prop(xi)

        perceptron\_1.backward\_prop(yi)

        perceptron\_1.update\_weights()

        perceptron\_2.forward\_prop(xi)

        perceptron\_2.backward\_prop(yi)

        perceptron\_2.update\_weights()

        perceptron\_1x2.forward\_prop(xi)

        perceptron\_1x2.backward\_prop(yi)

        perceptron\_1x2.update\_weights()

        perceptron\_1x3.forward\_prop(xi)

        perceptron\_1x3.backward\_prop(yi)

        perceptron\_1x3.update\_weights()

    return True

def menu\_item\_2(fx, rbf, perceptron\_1, perceptron\_2):

    points = np.linspace(-4.5, 4.5, 400)

    p1 = []

    p2 = []

    for i in points:

        p1.append(perceptron\_1(i.reshape(1, 1)).reshape(-1))

        p2.append(perceptron\_2(i.reshape(1, 1)).reshape(-1))

    plt.plot(points, rbf.predict(points), "g", label="Радиально-базисная сеть")

    plt.plot(points, p1, "r", label="Персептрон с 1 скрытым слоем")

    plt.plot(points, p2, "b", label="Персептрон с 2 скрытыми слоями")

    plt.plot(points, fx(points), "y", label="График функции")

    plt.xlabel("x")

    plt.ylabel("y")

    plt.title(r"Аппроксимация функции $f(x) = {2^{x√e^{(\sin(x)\cos(x))}}}$")

    plt.legend()

    plt.grid()

    plt.show()

def menu\_item\_3(fx, perceptron\_1, perceptron\_1x2, perceptron\_1x3):

    points = np.linspace(-4.5, 4.5, 400)

    p1 = []

    p1\_x2 = []

    p1\_x3 = []

    for i in points:

        p1.append(perceptron\_1(i.reshape(1, 1)).reshape(-1))

        p1\_x2.append(perceptron\_1x2(i.reshape(1, 1)).reshape(-1))

        p1\_x3.append(perceptron\_1x3(i.reshape(1, 1)).reshape(-1))

    plt.plot(points, p1, "g", label="Персептрон с 5 нейронами на скрытом слое")

    plt.plot(points, p1\_x2, "r", label="Персептрон с 10 нейронами на скрытом слое")

    plt.plot(points, p1\_x3, "b", label="Персептрон с 20 нейронами на скрытом слое")

    plt.plot(points, fx(points), "y", label="График функции")

    plt.xlabel("x")

    plt.ylabel("y")

    plt.title(r"Аппроксимация функции $f(x) = {2^{x√e^{(\sin(x)\cos(x))}}}$")

    plt.legend()

    plt.grid()

    plt.show()

def menu\_item\_4(seq\_length, train, rnn, perceptron\_3, mean, std):

    for epoch in range(500):

        q = np.random.randint(0, seq\_length)

        for i in range(q, len(train) - seq\_length, seq\_length):

            x = (np.array(train[i            : i+seq\_length]).reshape(1, seq\_length) - mean) / std

            y = (np.array(train[i+seq\_length : i+seq\_length+1]).reshape(1, 1) - mean) / std

            rnn(x)

            rnn.backward\_prop(y)

            rnn.update\_weights()

            perceptron\_3.forward\_prop(x)

            perceptron\_3.backward\_prop(y)

            perceptron\_3.update\_weights()

    return True

def menu\_item\_5(train):

    plt.plot(train)

    plt.xlabel("День (начиная с 1 января 2016 года)")

    plt.title("Курс евро к доллару")

    plt.grid()

    plt.show()

def menu\_item\_6(seq\_length, train, test, rnn, perceptron\_3, mean, std):

    rnn\_y = []

    perceptron\_y = []

    for i in train[-seq\_length:]:

        rnn\_y.append((i - mean) / std)

        perceptron\_y.append((i - mean) / std)

    for i in range(len(test)):

        out\_rnn = rnn(np.array(rnn\_y[i:i+seq\_length]).reshape(1, seq\_length)).reshape(-1)

        out\_perceptron\_3 = perceptron\_3(np.array(perceptron\_y[i:i+seq\_length]).reshape(1, seq\_length)).reshape(-1)

        rnn\_y.append(out\_rnn[0])

        perceptron\_y.append(out\_perceptron\_3[0])

    plt.plot(np.array(rnn\_y[seq\_length:]) \* std + mean, label="Сеть Элмана")

    plt.plot(np.array(perceptron\_y[seq\_length:]) \* std + mean, label="Персептрон")

    plt.plot(test, label="Правильный курс")

    plt.legend()

    plt.grid()

    plt.show()

def create\_nets\_for\_task\_1():

    rbf = RBFNet(7)

    perceptron\_1 = nn.NN(0.0001)

    perceptron\_1.add\_layer(1, 5, "tanh", need\_bias=True)

    perceptron\_1.add\_layer(5, 1)

    perceptron\_2 = nn.NN(0.0001)

    perceptron\_2.add\_layer(1, 5, "tanh", need\_bias=True)

    perceptron\_2.add\_layer(5, 5, "tanh", need\_bias=True)

    perceptron\_2.add\_layer(5, 1)

    perceptron\_1x2 = nn.NN(0.0001)

    perceptron\_1x2.add\_layer(1, 10, "tanh", need\_bias=True)

    perceptron\_1x2.add\_layer(10, 1)

    perceptron\_1x3 = nn.NN(0.0001)

    perceptron\_1x3.add\_layer(1, 20, "tanh", need\_bias=True)

    perceptron\_1x3.add\_layer(20, 1)

    return rbf, perceptron\_1, perceptron\_2, perceptron\_1x2, perceptron\_1x3

def create\_nets\_for\_task\_2(seq\_length):

    rnn = RNN(seq\_length, 50, 1, 0.003)

    perceptron\_3 = nn.NN(0.003)

    perceptron\_3.add\_layer(seq\_length, 50, "tanh")

    perceptron\_3.add\_layer(50, 1)

    return rnn, perceptron\_3

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

    train = deserialize(r"C:/Users/nikit/Desktop/Neuron/Lab5/train.txt")

    train.reverse()

    test = deserialize(r"C:/Users/nikit/Desktop/Neuron/Lab5/test.txt")

    test.reverse()

    mean = np.mean(train)

    std = np.std(train)

    # Задание 1

    training\_complete = False

    fx = lambda x: 2\*\*(x\*np.sqrt(np.exp(np.sin(x)\*np.cos(x))))

    rbf, perceptron\_1, perceptron\_2, perceptron\_1x2, perceptron\_1x3 = create\_nets\_for\_task\_1()

    # Задание 2

    training\_complete\_2 = False

    seq\_length = 80

    rnn, perceptron\_3 = create\_nets\_for\_task\_2(seq\_length)

    flag = False

    k = 0

    while not flag:

        draw\_menu()

        try:

            k = int(input("Введите номер пункта меню: "))

        except ValueError:

            pass

        print()

        if k == 1:

            training\_complete = menu\_item\_1(fx, rbf, perceptron\_1, perceptron\_2, perceptron\_1x2, perceptron\_1x3)

            print("\nОбучение завершено\n")

        elif k == 2:

            if training\_complete:

                menu\_item\_2(fx, rbf, perceptron\_1, perceptron\_2)

            else:

                print("Сначала проведите обучение нейронных сетей!\n")

        elif k == 3:

            if training\_complete:

                menu\_item\_3(fx, perceptron\_1, perceptron\_1x2, perceptron\_1x3)

            else:

                print("Сначала проведите обучение нейронных сетей!\n")

        elif k == 4:

            training\_complete\_2 = menu\_item\_4(seq\_length, train, rnn, perceptron\_3, mean, std)

            print("\nОбучение завершено\n")

        elif k == 5:

            menu\_item\_5(train)

        elif k == 6:

            if training\_complete\_2:

                menu\_item\_6(seq\_length, train, test, rnn, perceptron\_3, mean, std)

            else:

                print("Сначала проведите обучение нейронных сетей!\n")

        elif k == 7:

            flag = True