Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Курский государственный университет»

Кафедра программного обеспечения и администрирования

информационных систем

Направление подготовки математическое обеспечение и администрирование

информационных систем

Форма обучения очная

**Отчет**

**по лабораторной работе № 7**

«Временные ряды»

дисциплина «Основы теории нейронных сетей»

Выполнила:

студентка группы 413 Мусонда Салиму

Проверил:

Профессор кафедры ПОиАИС Добрица В. П.

Курск, 2020

Вариант 3

Задача 1. Составить обучающую программу однослойной нейронной сети,

интерпретирующей временной ряд, с 6 входными и одним выходным

нейронами. Параметры обучающей программы: шаг обучения α (0,1; 0,01), предполагаемая точность E = 0,01, временной шаг ∆t = 0,01, начальный момент t0 = 0, длина обучающей серии T = 1000. Значения временного ряда вычисляются по данной временной функции. Функция:

Задача 2. Провести обучение нейросети для того же временного ряда с

адаптивным шагом обучения. Сравнить время обучения сетей в этих задачах

с постоянным шагом обучения и с адаптивным шагом обучения.

Формулы

, где и целевое и выходное значение сети соответственно, где m – количество выходных нейронов, в рассматриваемом случае m = 1

Адаптивный шаг обучения:

В качестве входных значений сначала выбираются элементы временного ряда с 0 по 5, затем с 1 по 6 и т. д. последнее от 988 до 993. Последовательность от 994 до 999 используется для проверки в качестве входных данных, а вычисленное 1000-ое значение сравнивается с 1000-ым значением временного ряда для доказательства обученности сети. Так как все входные значения должны лежать в промежутке от 0 до 1, каждое входное значение делится на одно и то же число. В нашем случае на 102, так как < 102 для любого t от 0 до 100.

Блок-схема алгоритма представлена на рисунках 1 - 4.

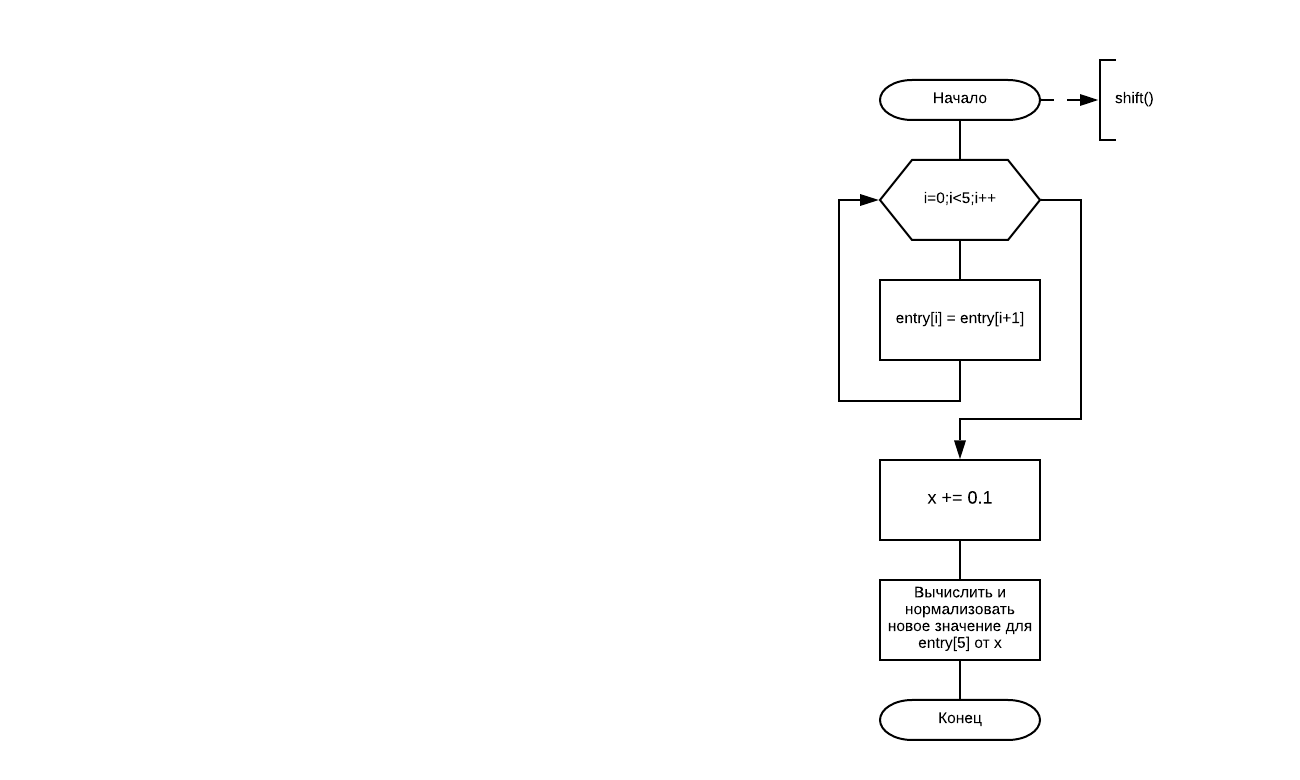


Рисунок 1 – Блок-схема функции shift(), сдвигающей массив входных значений entry на единицу вправо

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Блок-схема функции init(), инициализирующей значения входных данных

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Блок-схема функции compareResult(), проверяющая изменение результатов

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Блок-схема главной функции

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Продолжение

Текст программы

**import** javax.swing.\*;  
**import** javax.swing.event.TableModelEvent;  
**import** javax.swing.event.TableModelListener;  
**import** javax.swing.table.DefaultTableModel;  
**import** javax.swing.table.TableModel;  
**import** java.awt.\*;  
**import** java.awt.event.\*;  
  
**public class** MForm {  
 JFrame **mframe1**;  
 JRadioButton **jrbA**;  
 JButton **jbStudy**;  
 MCanvas **canvas**;  
 JLabel **jlForResult**;  
 JLabel **jlForIdeal**;  
 JLabel **jlForIter**;  
 JLabel **jlResult**;  
 JLabel **jlIdeal**;  
 JLabel **jlIter**;  
  
 **public** MForm() {  
 **mframe1** = **new** JFrame(**"Окно для ввода"**);  
 **mframe1**.setSize(400, 450);  
 **mframe1**.setDefaultCloseOperation(WindowConstants.***EXIT\_ON\_CLOSE***);  
 **canvas** = **new** MCanvas();  
 **mframe1**.add(**canvas**);  
 **jrbA** = **new** JRadioButton(**"Адаптивный шаг обучения"**);  
 **jrbA**.setLocation(10, 10);  
 **jrbA**.setSize(250, 30);  
 **jlForIdeal** = **new** JLabel(**"Эталонное значение: "**);  
 **jlForIdeal**.setSize(250,25);  
 **jlForIdeal**.setLocation(10, 120);  
 **jlIdeal** = **new** JLabel();  
 **jlIdeal**.setLocation(150, 120);  
 **jlIdeal**.setSize(100, 25);  
 **jlForResult** = **new** JLabel(**"Вычисленное значение: "**);  
 **jlForResult**.setLocation(10, 155);  
 **jlForResult**.setSize(250, 25);  
 **jlResult** = **new** JLabel();  
 **jlResult**.setSize(100, 25);  
 **jlResult**.setLocation(160, 155);  
 **jlForIter** = **new** JLabel(**"Количество итераций: "**);  
 **jlForIter**.setLocation(10, 190);  
 **jlForIter**.setSize(250, 25);  
 **jlIter** = **new** JLabel();  
 **jlIter**.setSize(100, 25);  
 **jlIter**.setLocation(150, 190);  
 **jbStudy** = **new** JButton(**"Начать обучение"**);  
 **jbStudy**.setLocation(10, 70);  
 **jbStudy**.setSize(250, 30);  
 **jbStudy**.addActionListener(**new** ActionListener() {  
 @Override  
 **public void** actionPerformed(ActionEvent e) {  
 Network network = **new** Network(**jbStudy**.isSelected());  
 **if**(network.study()) {  
 network.check(**jlIdeal**, **jlResult**, **jlIter**);  
 }  
 **else** {  
 **jlResult**.setText(**"Обучение невозможно"**);  
 }  
 }  
 });  
  
 **canvas**.add(**jlIter**);  
 **canvas**.add(**jlForIter**);  
 **canvas**.add(**jlForIdeal**);  
 **canvas**.add(**jlForResult**);  
 **canvas**.add(**jlResult**);  
 **canvas**.add(**jlIdeal**);  
 **canvas**.add(**jrbA**);  
 **canvas**.add(**jbStudy**);  
 **mframe1**.setVisible(**true**);  
 }  
  
 **public static void** main(String[] args) {  
 MForm mForm = **new** MForm();  
  
 }  
}

**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.util.ArrayList;  
  
**public class** MCanvas **extends** JComponent {  
  
 MCanvas() {  
  
 }  
  
 **public void** paintComponent(Graphics g) {  
 **super**.paintComponents(g);  
 }  
}

**import** javax.swing.\*;  
  
  
  
**public class** Network {  
  
 **private boolean adapt**;  
  
 **private double a**;  
  
 **private double e**;  
  
 **private double** []**w**;  
  
 **private double** []**e0**;  
  
 **private double** []**e1**;  
  
 **private double t**;  
  
 **private double x**;  
  
 **private int iter**;  
  
 **private double** [] **entry**;  
  
 **public** Network(**boolean** adapt) {  
 **this**.**adapt** = adapt;  
 **if**(!adapt){  
 **a** = 0.01;  
 }  
 **x** = 0;  
 **e** = 0.01;  
 **entry** = **new double**[6];  
 **w** = **new double**[6];  
 **double** k = 0;  
 **for**(**int** i = 0; i < 6; i ++) {  
 **w**[i] = Math.*random*();  
 }  
 **t** = Math.*random*();  
 }  
  
 **public double**[] getW() {  
 **return w**;  
 }  
  
 **public void** setW(**double**[] w) {  
 **this**.**w** = w;  
 }  
  
 **public double** getT() {  
 **return t**;  
 }  
  
 **public void** setT(**double** t) {  
 **this**.**t** = t;  
 }  
  
 **public boolean** isAdapt() {  
 **return adapt**;  
 }  
  
 **public void** setAdapt(**boolean** adapt) {  
 **this**.**adapt** = adapt;  
 }  
  
 **public double** getA() {  
 **return a**;  
 }  
  
 **public void** setA(**double** a) {  
 **this**.**a** = a;  
 }  
  
 **public double** getE() {  
 **return e**;  
 }  
  
 **public void** setE(**double** e) {  
 **this**.**e** = e;  
 }  
  
 **private double** formula(**double** k) {  
 **return** ((k + 2 \* Math.*sin*(k/3))/102);  
 }  
  
 **private void** shift() {  
 **for**(**int** i = 0; i < 5; i ++) {  
 **entry**[i] = **entry**[i+1];  
 }  
 **x** += 0.1;  
 **entry**[5] = formula(**x**);  
 }  
  
 **private double** calculateY() {  
 **double** yCalc = 0;  
 **for**(**int** i = 0; i < 6; i ++) {  
 yCalc += **w**[i] \* **entry**[i];  
 }  
 yCalc -= **t**;  
 **return** yCalc;  
 }  
 **private void** calculateCoef(**double** yCalc, **double** y) {  
 **for**(**int** i = 0; i < 6; i ++) {  
 **w**[i] = **w**[i] + **a** \* (y - yCalc) \* **entry**[i];  
 }  
 **t** = **t** - **a** \* (y - yCalc);  
 }  
  
 **private void** calculateA() {  
 **double** sum = 0;  
 **for**(**int** i = 0; i < 6; i ++) {  
 sum += **entry**[i] \* **entry**[i];  
 }  
 **a** = 1 / (sum + 1);  
 }  
  
 **private void** init() {  
 **x** = 0;  
 **for**(**int** i = 0; i < 6; i ++) {  
 **entry**[i] = formula(**x**);  
 **x** += 0.1;  
 }  
 }  
  
 **private boolean** compareResult() {  
 **for**(**int** i = 0; i < 6; i ++) {  
 **if**(**e1**[i] != Math.*abs*(**w**[i] - **e0**[i])) {  
 **for**(**int** j = 0; j < 6; j ++) {  
 **e1**[j] = Math.*abs*(**w**[j] - **e0**[j]);  
 }  
 **return true**;  
 }  
 }  
 **return false**;  
 }  
  
 **public boolean** study() {  
 **double** eRef;  
 **double** yCalc, y;  
 **do** {  
 init();  
 eRef = 0;  
 **iter** ++;  
 **e0** = **w**;  
 **for**(**int** i = 0; i < 993; i ++) {  
 **if**(**adapt**) {  
 calculateA();  
 }  
 yCalc = calculateY();  
 shift();  
 eRef += 0.5 \*(**entry**[5] - yCalc) \* (**entry**[5] - yCalc);  
 calculateCoef(yCalc, **entry**[5]);  
 }  
 **if**(!compareResult()) {  
 **return false**;  
 }  
 }**while**(**e** < eRef);  
 **return true**;  
 }  
 **public void** check(JLabel jlIdeal, JLabel jlResult, JLabel jlIter) {  
 shift();  
 jlIdeal.setText(String.*valueOf*(calculateY()));  
 jlResult.setText(String.*valueOf*(**entry**[5]));  
 jlIter.setText(String .*valueOf*(**iter**));  
  
  
 }  
}

Результаты работы программы представлены на рисунке 5 - 6.

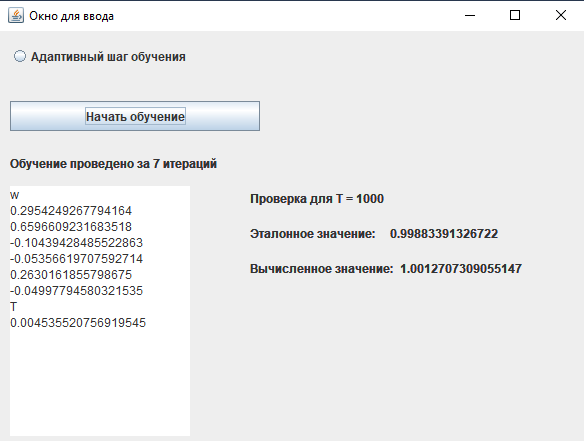


Рисунок 5 – Результат работы программы без адаптивного шага обучения

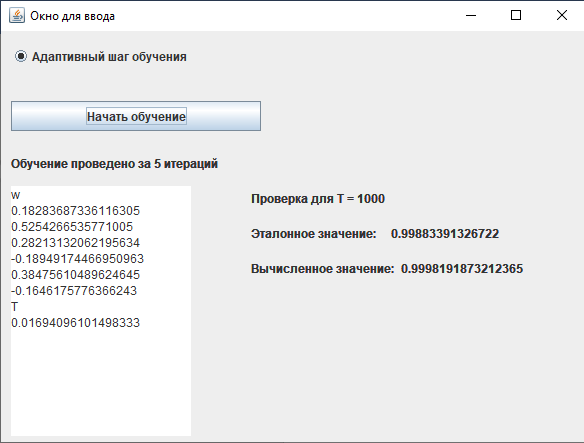


Рисунок 6 – Результат работы программы с адаптивным шагом обучения

Вывод: Адаптивный шаг обучения позволяет повысить скорость и точность обучения.