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 № 5**

«Алгоритм Видроу-Хоффа обучения нейронных сетей»

дисциплина «Основы теории нейронных сетей»

Выполнила:

студентка группы 413 Мусонда Салиму

Проверил:

Профессор кафедры ПОиАИС Добрица В. П.

Курск, 2020

Вариант 3

Задача № 1. Провести обучение однослойной нейронной сети для указанной в задаче №1 лабораторной работы №3 функции по алгоритму Видроу-Хоффа при α = 0.3, E = 0.3.

Функция: Логическая функция дизъюнкция в биполярном случае.

|  |  |  |
| --- | --- | --- |
| x1 | x2 | y = x1 V x2 |
| -1 | -1 | -1 |
| -1 | 1 | 1 |
| 1 | -1 | 1 |
| 1 | 1 | 1 |

Формулы:

, где и целевое и выходное значение сети соответственно

Блок-схема алгоритма представлена на рисунках 1 - 2.

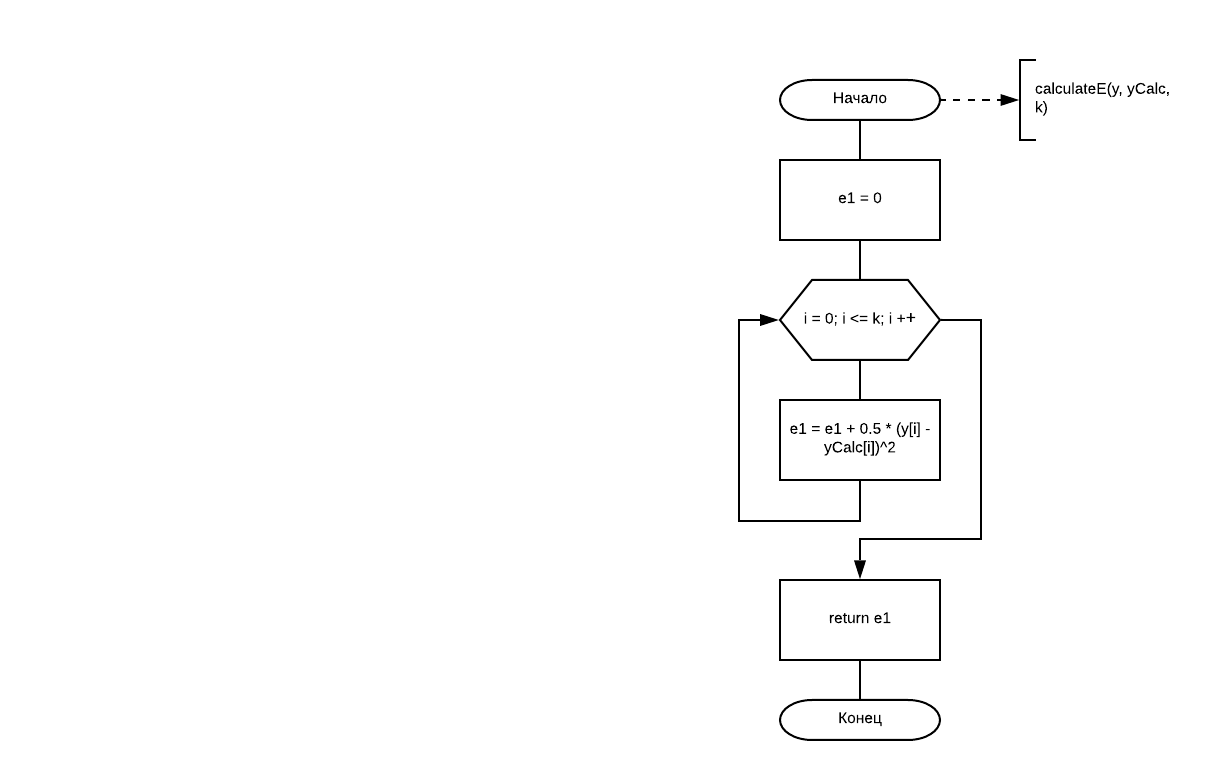


Рисунок 1 – Блок-схема функции CalculateE(y, yCalc, k), вычисляющей E

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 – Блок-схема функции compareResalt(w, e1, e0), проверяет изменение коэффициентов

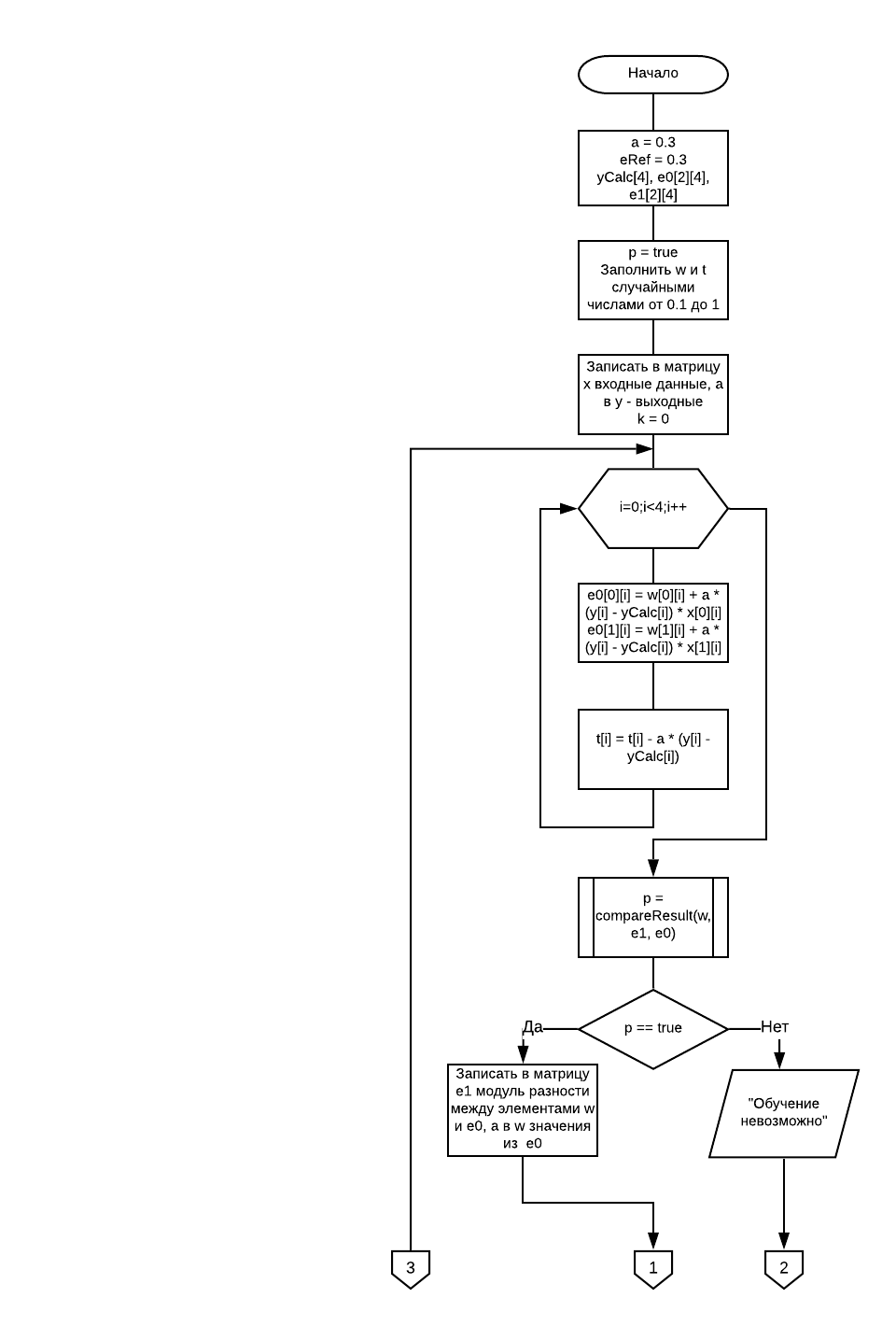


Рисунок 3 – Главная функция

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 – Продолжение

Текст программы

**public class** Main {  
 **public static double** calculateE(**int**[] y, **double**[] yCalc, **int** k) {  
 **double** e1 = 0;  
 **for** (**int** i = 0; i <= k; i++) {  
 e1 = e1 + 0.5 \* Math.*pow*(y[i] - yCalc[i], 2);  
 }  
 **return** e1;  
 }  
 **public static boolean** compareResult(**double** [][]w, **double** [][]e1, **double** [][]e0) {  
 **for**(**int** i = 0; i < 4; i ++) {  
 **if**(e1[0][i] != Math.*abs*(w[0][i] - e0[0][i])) {  
 **return true**;  
 }  
 **if**(e1[1][i] != Math.*abs*(w[1][i] - e0[1][i])) {  
 **return true**;  
 }  
 }  
 **return false**;  
 }  
 **public static void** main(String[] args) {  
 **double** a = 0.3;  
 **double** eReference = 0.3;  
 **int** []y = **new int**[]{-1, 1, 1, 1};  
 **double** []yCalc = **new double**[4];  
 **int** [][]x = **new int**[][] {{-1, -1, 1, 1}, {-1, 1, -1, 1}};  
 **double** e;  
 **double** [][]w = **new double**[2][4];  
 **double** [][]e0 = **new double**[2][4];  
 **double** [][]e1 = **new double**[2][4];  
 **double** [] t = **new double**[4];  
 **int** k = 0;  
 **for**( **int** i = 0; i < 4; i ++) {  
 w[0][i] = Math.*random*() + 0.1;  
 w[1][i] = Math.*random*() + 0.1;  
 t[i] = Math.*random*() + 0.1;  
 }  
 **boolean** p = **true**;  
 **int** g = 1;  
 **do** {  
 **for** (**int** i = 0; i < 4; i++) {  
 e0[0][i] = w[0][i] + a \* (y[i] - yCalc[i]) \* x[0][i];  
 e0[1][i] = w[1][i] + a \* (y[i] - yCalc[i]) \* x[1][i];  
 t[i] = t[i] - a \* (y[i] - yCalc[i]);  
 }  
 **if**(*compareResult*(w, e1, e0)) {  
 **for**(**int** i = 0; i < 4; i ++) {  
 e1[0][i] = Math.*abs*(w[0][i] - e0[0][i]);  
 e1[1][i] = Math.*abs*(w[1][i] - e0[1][i]);  
 w[0][i] = e0[0][i];  
 w[1][i] = e0[1][i];  
 }  
 }  
 **else** {  
 p = **false**;  
 System.***out***.println(**"Обучение невозможно"**);  
 **break**;  
 }  
 **for** (**int** i = 0; i < 4; i++) {  
 yCalc[i] = e0[0][i] \* x[0][i] + e0[1][i] \* x[1][i] - t[i];  
 }  
 k++;  
 e = *calculateE*(y, yCalc, 3);  
 }**while**(e > eReference);  
 **if**(p) {  
 System.***out***.println(**"w1 w2 t"**);  
 **for**(**int** i = 0; i < 4; i ++) {  
 System.***out***.println(w[0][i] + **" "** + w[1][i] + **" "** + t[i]);  
 }  
 System.***out***.println(**"Проверка"**);  
 System.***out***.println(**"y y(выч)"**);  
 **for**(**int** i = 0; i < 4; i ++) {  
 System.***out***.println(y[i] + **" "** + yCalc[i]);  
 }  
 System.***out***.println(**"Количество итераций - "** + k);  
 }  
 }  
 }

Результаты работы программы представлены на рисунке 3.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 4 – Результат работы программы

Количество итераций: 2.

Задача № 2. Провести обучение однослойной нейронной сети для указанных данных в виде таблицы в задаче №2 лабораторной работы №3 с помощью алгоритма Видроу-Хоффа с адаптивным шагом обучения при Е = 0.3.

Функция: Логическая функция «штрих Шеффера».

|  |  |  |
| --- | --- | --- |
| x1 | x2 | y = x1 |x2 |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Формулы

, где и целевое и выходное значение сети соответственно

Блок-схема алгоритма представлена на рисунках 4 - 5.

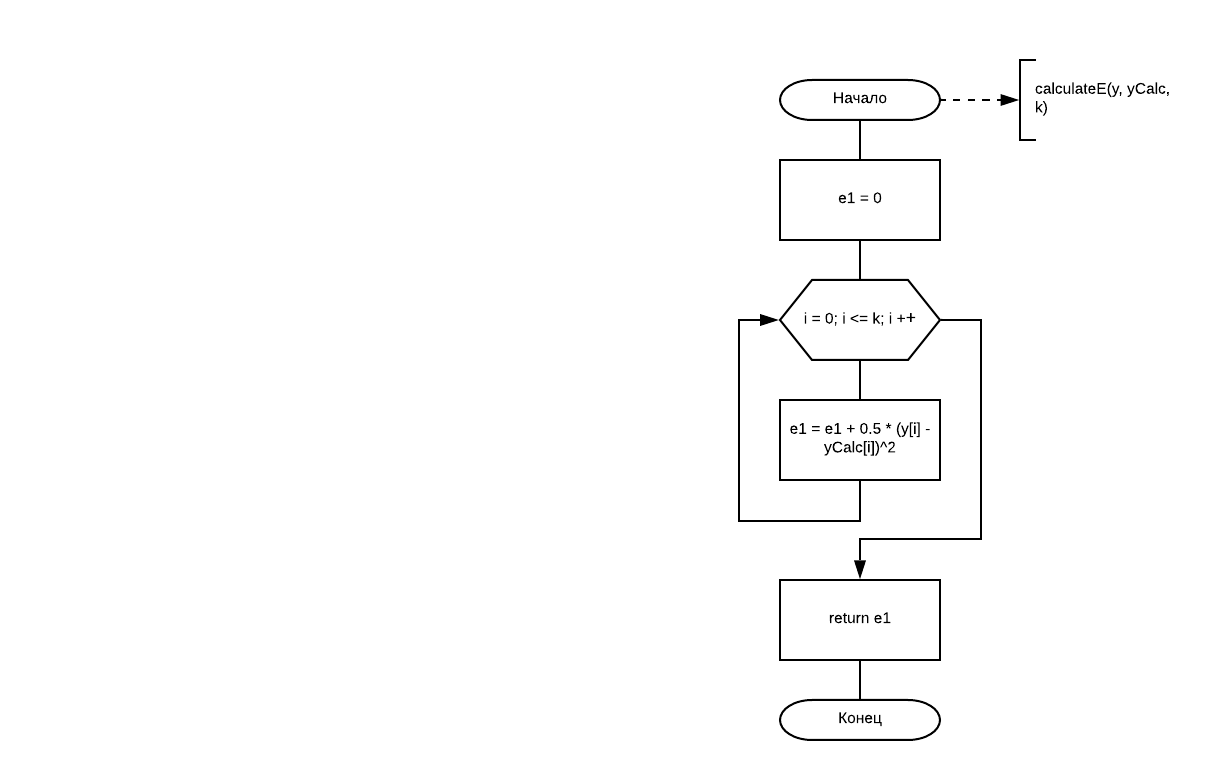


Рисунок 5 – Блок-схема функции CalculateE(y, yCalc, k), вычисляющей E

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 – Блок-схема функции compareResalt(w, e1, e0), проверяет изменение коэффициентов

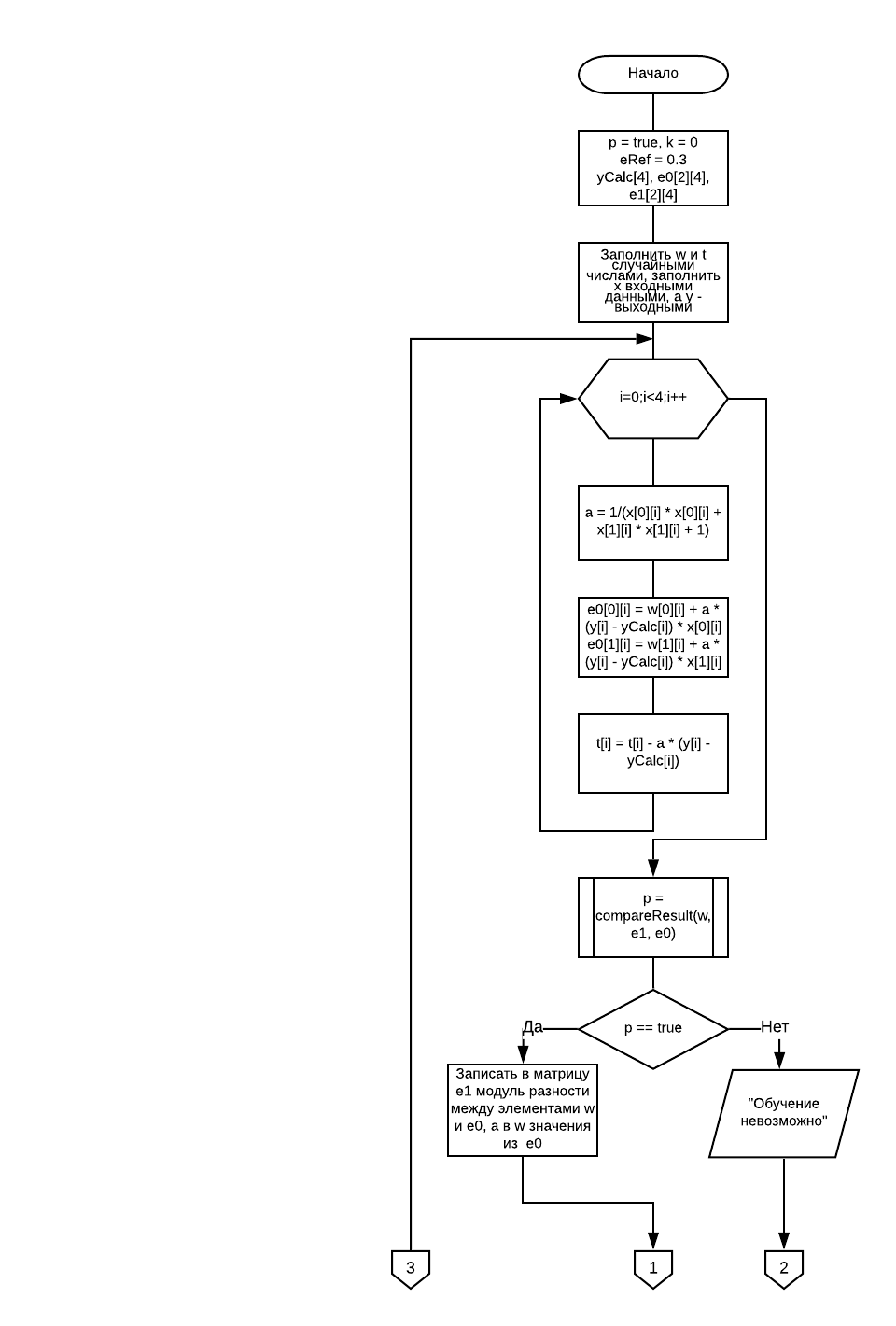


Рисунок 7 – Главная функция

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7 – Продолжение

Текст программы

**public class** Main {  
 **public static double** calculateE(**double**[] y, **double**[] yCalc, **int** k) {  
 **double** e1 = 0;  
 **for**(**int** i = 0; i <= k; i ++) {  
 e1 = e1 + 0.5 \* Math.*pow*(y[i] - yCalc[i], 2);  
 }  
 **return** e1;  
 }  
 **public static boolean** compareResult(**double** [][]w, **double** [][]e1, **double** [][]e0) {  
 **for**(**int** i = 0; i < 4; i ++) {  
 **if**(e1[0][i] != Math.*abs*(w[0][i] - e0[0][i])) {  
 **return true**;  
 }  
 **if**(e1[1][i] != Math.*abs*(w[1][i] - e0[1][i])) {  
 **return true**;  
 }  
 }  
 **return false**;  
 }  
  
 **public static void** main(String[] args) {  
 **double** eReference = 0.3;  
 **double** []y = **new double**[]{1, 1, 1, 0};  
 **double** []yCalc = **new double**[4];  
 **double** [][]x = **new double**[][] {{0, 0, 1, 1}, {0, 1, 0, 1}};  
 **double** a;  
 **double** e;  
 **double** [][]w = **new double**[2][4];  
 **double** [][]e0 = **new double**[2][4];  
 **double** [][]e1 = **new double**[2][4];  
 **double** [] t = **new double**[4];  
 **int** k = 0;  
 **for**( **int** i = 0; i < 4; i ++) {  
 w[0][i] = Math.*random*() + 0.1;  
 w[1][i] = Math.*random*() + 0.1;  
 t[i] = Math.*random*() + 0.1;  
 }  
 **boolean** p = **true**;  
  
 **do** {  
 **for** (**int** i = 0; i < 4; i++) {  
 a = 1 / (Math.*pow*(x[0][i], 2) + Math.*pow*(x[1][i], 2) + 1);  
 e0[0][i] = w[0][i] + a \* (y[i] - yCalc[i]) \* x[0][i];  
 e0[1][i] = w[1][i] + a \* (y[i] - yCalc[i]) \* x[1][i];  
 t[i] = t[i] - a \* (y[i] - yCalc[i]);  
 }  
 **if**(*compareResult*(w, e1, e0)) {  
 **for**(**int** i = 0; i < 4; i ++) {  
 e1[0][i] = Math.*abs*(w[0][i] - e0[0][i]);  
 e1[1][i] = Math.*abs*(w[1][i] - e0[1][i]);  
 w[0][i] = e0[0][i];  
 w[1][i] = e0[1][i];  
 }  
 }  
 **else** {  
 p = **false**;  
 System.***out***.println(**"Обучение невозможно"**);  
 **break**;  
 }  
 **for** (**int** i = 0; i < 4; i++) {  
 yCalc[i] = e0[0][i] \* x[0][i] + e0[1][i] \* x[1][i] - t[i];  
 }  
 k++;  
 e = *calculateE*(y, yCalc, 3);  
 }**while**(e > eReference);  
 **if**(p) {  
 System.***out***.println(**"w1 w2 t"**);  
 **for**(**int** i = 0; i < 4; i ++) {  
 System.***out***.println(w[0][i] + **" "** + w[1][i] + **" "** + t[i]);  
 }  
 System.***out***.println(**"Проверка"**);  
 System.***out***.println(**"y y(выч)"**);  
 **for**(**int** i = 0; i < 4; i ++) {  
 System.***out***.println(y[i] + **" "** + yCalc[i]);  
 }  
 System.***out***.println(**"Количество итераций - "** + k);  
 }  
  
 }  
}

Результаты работы программы представлены на рисунке 6.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок 8 – Результат работы программы

Вывод: Из-за небольшого количества итераций разница в скорости обучения с адаптивным шагом и без него не выявлена.