**Министерство образования Российской Федерации**

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**им. Н.Э. БАУМАНА**

Факультет: Информатика, искусственный интеллект и системы управления

Кафедра: Информационная безопасность (ИУ8)

ИНТЕЛЛЕКТУАЛЬНЫЕ ТЕХНОЛОГИИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Лабораторная работа №1 на тему:

«Исследование однослойных нейронных сетей на примере моделирования булевых выражений»

Вариант 15

**Преподаватель:**

Строганов И.

**Студент**:

Рыбаков Д. В.

**Группа:**

ИУ8-63

Москва 2022 г.

**Цель работы**

Исследовать функционирование простейшей нейронной сети (НС) на базе нейрона с нелинейной функцией активации и ее обучение по правилу Видроу-Хоффа.

**Постановка задачи**

Получить модель булевой функции (БФ) на основе однослойной НС (единичный нейрон) с двоичными входами 𝑥1, 𝑥2, 𝑥3, 𝑥4 ∈ {0, 1}, единичным входом смещения 𝑥0 = 1, синаптическими весами w0, w1, w2, w3, w4, двоичным выходом 𝑦 ∈ {0, 1} и заданной нелинейной функцией активации (ФА) ƒ: 𝑅 → (0, 1).

Для заданной БФ реализовать обучение НС для двух случаев:

1. с использованием всех комбинаций переменных 𝑥1, 𝑥2, 𝑥3, 𝑥4;
2. с использованием части возможных комбинаций переменных

𝑥1, 𝑥2, 𝑥3, 𝑥4; остальные комбинации используются в качестве тестовых.

**Ход работы**

Получим таблицу истинности для моделируемой БФ:



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| x2 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| x3 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| x4 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 𝐹 | **1** | **1** | **1** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** | **1** | **1** | **0** | **0** |

Норма обучения для всех случаев выбирается η = 0.3.

1. Обучение НС с использованием всех комбинаций переменных

𝑥1, 𝑥2, 𝑥3, 𝑥4.

* 1. Используя пороговую ФА:

( ) 1, 𝑛e𝑡 > 0,

f net = {

0, 𝑛e𝑡 ≤ 0

**Таблица 1 Параметры НС на последовательных эпохах (пороговая ФА)**

|  |  |
| --- | --- |
|  | Вектор весов we, выходной вектор y, суммарная ошибка 𝐸rror |

{

weights: [ 0, 0.3, 0, -0.6, -0.3 ],

Error: 4,

y: [

1, 1, 1, 1, 0, 1,

1, 0, 0, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 0

{

weights: [ 0, 0, 0, -1.2, 0 ],

Error: 6,

y: [

1, 0, 0, 1, 1, 1,

0, 0, 1, 1, 1, 0,

1, 0, 1, 0

]

} эпоха 1

{

weights: [ 0.3, 0, 0, -0.8999999999999999, 0 ],

Error: 1,

y: [

1, 1, 0, 0, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 2

{

weights: [ 0.3, 0, 0, -0.8999999999999999, -0.3 ],

Error: 2,

y: [

1, 1, 0, 1, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 3

{

weights: [ 0.3, 0, -0.3, -1.2, 0 ],

Error: 4,

y: [

1, 1, 0, 0, 1, 1,

1, 0, 1, 1, 0, 0,

1, 0, 1, 0

]

} эпоха 4

{

weights: [ 0.6, 0, -0.3, -0.8999999999999999, 0 ],

Error: 1,

y: [

1, 1, 0, 0, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 5

{

weights: [ 0.5999999999999999, 0, -0.3, -1.2, 0 ],

Error: 4,

y: [

1, 1, 0, 1, 1, 0,

1, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 6

{

weights: [ 0.8999999999999999, 0, 0, -1.2, 0 ],

Error: 3,

y: [

1, 1, 0, 1, 1, 0,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 7

{

weights: [ 0.8999999999999999, 0, 0, -1.2, -0.3 ],

Error: 2,

y: [

1, 1, 0, 1, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 8

{

weights: [ 0.8999999999999999, 0, 0, -1.2, -0.6 ],

Error: 2,

y: [

1, 1, 0, 1, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 9

{

weights: [ 0.8999999999999999, 0, -0.3, -1.2, -0.6 ],

Error: 2,

y: [

1, 1, 0, 0, 1, 1,

1, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 10

{

weights: [ 0.8999999999999999, 0, -0.6, -1.5, -0.3 ],

Error: 4,

y: [

1, 1, 0, 0, 1, 1,

1, 0, 1, 1, 0, 0,

1, 0, 1, 0

]

} эпоха 11

{

weights: [ 1.2, 0, -0.3, -1.5, 0 ],

Error: 3,

y: [

1, 1, 0, 0, 1, 1,

0, 0, 1, 1, 1, 0,

1, 0, 0, 0

]

} эпоха 12

{

weights: [ 1.2, 0, -0.3, -1.5, -0.3 ],

Error: 2,

y: [

1, 1, 0, 1, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 13

{

weights: [ 1.2, 0, -0.3, -1.5, -0.6 ],

Error: 2,

y: [

1, 1, 0, 1, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 14

{

weights: [ 1.2, 0, -0.6, -1.5, -0.6 ],

Error: 2,

y: [

1, 1, 0, 0, 1, 1,

1, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 15

{

weights: [ 1.5, 0, -0.3, -1.5, -0.3 ],

Error: 3,

y: [

1, 1, 0, 0, 1, 1,

0, 0, 1, 1, 1, 0,

1, 0, 0, 0

]

} эпоха 16

{

weights: [ 1.2, -0.3, -0.3, -1.8, -0.3 ],

Error: 1,

y: [

1, 1, 1, 0, 1, 1,

0, 0, 1, 1, 1, 0,

1, 1, 0, 0

]

} эпоха 17

{

weights: [ 1.5, -0.3, -0.3, -1.5, -0.3 ],

Error: 1,

y: [

1, 1, 0, 0, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 18

{

weights: [ 1.5, -0.3, -0.3, -1.5, -0.3 ],

Error: 0,

y: [

1, 1, 1, 0, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 19

Получили следующее веса для пороговой ФА - [1.5, -0.3, -0.3, -1.5, -0.3]

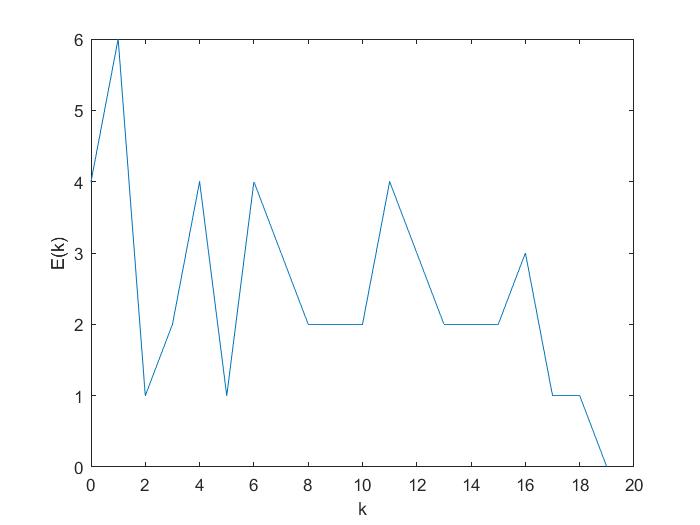


Рисунок 1 График суммарной ошибки НС по эпохам обучения (пороговая ФА)

1.2 Используем гиперболический тангенс (логистическую) как ФА:



**Таблица 2 Параметры НС на последовательных эпохах (логистическая ФА)**

|  |  |
| --- | --- |
|  | Вектор весов weigth, выходной вектор y, суммарная ошибка Error |

{

weights: [

0.00886946803362712,

0,

0.15886946803362711,

-0.287798885749569,

-0.00000663323125132953

],

Error: 4,

y: [

1, 1, 1, 1, 0, 0,

1, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 0

{

weights: [

0.0014509911686257648,

0,

0.012541570030939864,

-0.2952173626145704,

-0.00000663323125132953

],

Error: 2,

y: [

1, 1, 0, 0, 1, 1,

1, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 1

{

weights: [

0.1392160855475386,

0,

0.012541570030939864,

-0.15745226823565756,

-0.00000663323125132953

],

Error: 1,

y: [

1, 1, 0, 0, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 2

{

weights: [

0.14467738168501648,

0.14417137402393176,

0.00957653836262859,

-0.44615468291427474,

-0.1415304327290294

],

Error: 6,

y: [

1, 1, 0, 1, 1, 1,

1, 0, 1, 0, 0, 1,

0, 1, 0, 0

]

} эпоха 3

{

weights: [

0.13647353589267933,

-0.001184642494449717,

0.012185821042159012,

-0.6043579965345544,

0.008469035098912947

],

Error: 4,

y: [

1, 1, 0, 0, 1, 1,

0, 0, 1, 1, 1, 0,

1, 0, 1, 0

weights: [

0.25789521543294425,

-0.001184642494449717,

0.012185821042159012,

-0.48293631699428946,

0.008469035098912947

],

Error: 1,

y: [

1, 1, 0, 0, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 5

{

weights: [

0.25125441437552853,

-0.001184642494449717,

0.012185821042159012,

-0.48957711805170523,

-0.14082454682953924

],

Error: 2,

y: [

1, 1, 0, 1, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 6

{

weights: [

0.2438384836675245,

-0.0008873984719712513,

-0.13702296690285876,

-0.6468010938177409,

0.008983498228492343

],

Error: 4,

y: [

1, 1, 0, 0, 1, 1,

1, 0, 1, 1, 0, 0,

1, 0, 1, 0

]

} эпоха 7

{

weights: [

0.37189472305842647,

-0.0008873984719712513,

-0.13702296690285876,

-0.5187448544268389,

0.008983498228492343

],

Error: 1,

y: [

1, 1, 0, 0, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 8

{

weights: [

0.37228684528605527,

-0.0008873984719712513,

-0.13702296690285876,

-0.5183527321992101,

-0.13743557815884777

],

Error: 2,

y: [

1, 1, 0, 1, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 9

{

weights: [

0.3687035498445672,

-0.0008873984719712513,

-0.13746651878904162,

-0.6714783158608191,

-0.13787772198361942

],

Error: 4,

y: [

1, 1, 0, 1, 1, 0,

1, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 10

{

weights: [

0.5057513900477064,

-0.0008873984719712513,

-0.13746651878904162,

-0.5344304756576799,

-0.13787772198361942

],

Error: 1,

y: [

1, 1, 0, 0, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 11

{

weights: [

0.508258121598114,

-0.0008873984719712513,

-0.13746651878904162,

-0.5319237441072724,

-0.2852476845579541

],

Error: 2,

y: [

1, 1, 0, 1, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 12

{

weights: [

0.5135891518054969,

0.0016785695000368361,

-0.28205151068530426,

-0.6760588443207125,

-0.13578155413713103

],

Error: 4,

y: [

1, 1, 0, 0, 1, 1,

1, 0, 1, 1, 0, 0,

1, 0, 1, 0

]

} эпоха 13

{

weights: [

0.661865189438516,

0.003845418359853553,

-0.1324460836646389,

-0.6773882337083588,

0.013823872883534344

],

Error: 3,

y: [

1, 1, 0, 0, 1, 1,

0, 0, 1, 1, 1, 0,

1, 0, 0, 0

]

} эпоха 14

{

weights: [

0.5244441887274531,

-0.14612754506159725,

-0.1324460836646389,

-0.8148092344194218,

-0.12358802547618294

],

Error: 3,

y: [

1, 1, 0, 1, 1, 1,

0, 0, 1, 1, 1, 0,

1, 1, 0, 0

]

} эпоха 15

{

weights: [

0.6624757286731426,

-0.14612754506159725,

-0.1324460836646389,

-0.6767776944737323,

-0.12358802547618294

],

Error: 1,

y: [

1, 1, 0, 0, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 16

{

weights: [

0.6663160705595369,

-0.14612754506159725,

-0.1324460836646389,

-0.672937352587338,

-0.26971700583935543

],

Error: 2,

y: [

1, 1, 0, 1, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 17

{

weights: [

0.670228773716363,

-0.1422921227832332,

-0.12861066138627483,

-0.8188729376919184,

-0.269784860685335

],

Error: 4,

y: [

1, 1, 0, 1, 1, 1,

0, 0, 1, 1, 0, 0,

1, 0, 1, 0

]

} эпоха 18

{

weights: [

0.8169183090086427,

0.007624037948628176,

-0.12865507797643244,

-0.8220995631315001,

-0.11986869995347366

],

Error: 3,

y: [

1, 1, 0, 0, 1, 1,

1, 0, 1, 1, 0, 0,

1, 0, 0, 0

]

} эпоха 19

{

weights: [

0.6714325234072408,

-0.14235636013550404,

-0.12865507797643244,

-0.9675853487329019,

-0.2653700607336668

],

Error: 3,

y: [

1, 1, 0, 1, 1, 1,

0, 0, 1, 1, 1, 0,

1, 1, 0, 0

]

} эпоха 20

{

weights: [

0.809009234506572,

-0.14235636013550404,

-0.12865507797643244,

-0.8300086376335707,

-0.2653700607336668

],

Error: 1,

y: [

1, 1, 0, 0, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 21

{

weights: [

0.8089704349331129,

-0.14235636013550404,

-0.12865507797643244,

-0.8300474372070298,

-0.4153427335084033

],

Error: 2,

y: [

1, 1, 0, 1, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 22

{

weights: [

0.8122162417101095,

-0.1423748350742043,

-0.2753426549256608,

-0.9767368244868321,

-0.26540753945160445

],

Error: 4,

y: [

1, 1, 0, 0, 1, 1,

1, 0, 1, 1, 0, 0,

1, 0, 1, 0

]

} эпоха 23

{

weights: [

0.9582283426954794,

-0.1423748350742043,

-0.2753426549256608,

-0.8307247235014623,

-0.26540753945160445

],

Error: 1,

y: [

1, 1, 0, 0, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 24

{

weights: [

0.9582283426954794,

-0.1423748350742043,

-0.2753426549256608,

-0.8307247235014623,

-0.26540753945160445

],

Error: 0,

y: [

1, 1, 1, 0, 1, 1,

0, 0, 1, 1, 0, 0,

1, 1, 0, 0

]

} эпоха 25

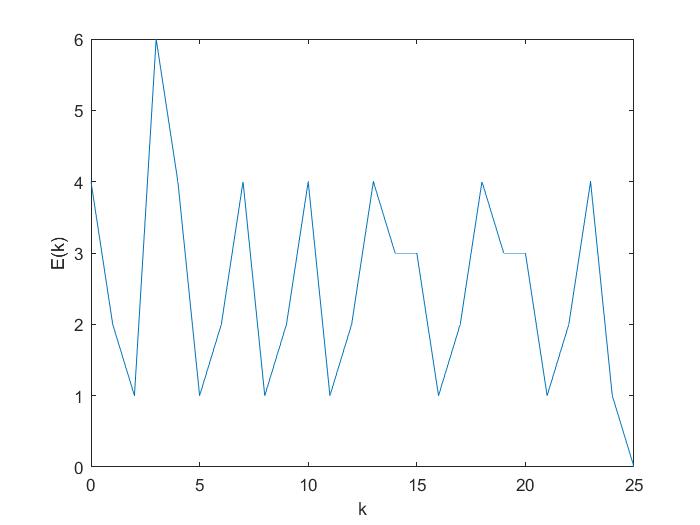


Рисунок 2 График суммарной ошибки НС по эпохам обучения (пороговая ФА)

Получили следующее веса для логистической ФА:

0.9582283426954794,

-0.1423748350742043,

-0.2753426549256608,

-0.8307247235014623,

-0.26540753945160445

**Минимальные вектора**

Пороговая ФА имеет следующие минимальные вектора

**x**(1) = [0,0,1,0]

**x**(2) = [0,1,1,0]

**x**(3) = [1,0,1,1]

**x**(4) = [1,1,0,1]

**w** = [1.2, -0.3, -0.3, -1.2, -0.3]

для этого потребовалось 19 эпох

Логистическая ФА имеет следующие минимальные вектора

**x**(1) = [0,0,1,0]

**x**(2) = [0,1,1,1]

**x**(3) = [1,0,1,0]

**x**(4) = [1,1,0,1]

**w** = [0.3533043547588831, -0.03568376493439629, 0.05563860860398001,-0.33476697294968616,-0.05563860860398001]

для этого потребовалось 11 эпох

**Вывод**: В ходе проделанной работы была изучена работа простейшего нейрона НС на базе нейрона с нелинейной функцией активации и проведено обучение по правилу Видроу-Хоффа. В качестве функций активации брались две различные функции – пороговая и логистическая. Было выявлено, что с использованием логистической функции активации понадобилось меньше эпох, чем для обучения с использованием пороговой функции активации на несколько эпох. В обоих случаях удалось найти наборы, состоящие из минимального кол-ва векторов.

**Исходный код:**

|  |
| --- |
| const norm = 0.3 |
|  | const x0 = 1 |
|  | let N = 16 // кол-во всех возможных значений векторов |
|  | let tTeacher = [1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0] // значение t на наборах переменных |
|  |  |
|  | // значения всех возможных наборов аргументов |
|  | let X1 = [0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1] |
|  | let X2 = [0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1] |
|  | let X3 = [0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1] |
|  | let X4 = [0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1] |
|  |  |
|  | // пороговая ФА |
|  | const porogovayaFunc = net => net >= 0 ? 1 : 0 |
|  |  |
|  | // логистическая ФА |
|  | const logisticFunc = out => out >= 0.5 ? 1 : 0 |
|  |  |
|  | // hyperbolic tangent function |
|  | const hyperbolicTangentOut = x => |
|  | (((Math.exp(x) - Math.exp(-x)) / |
|  | (Math.exp(x) + Math.exp(-x))) + 1) \* 0.5 |
|  |  |
|  | const calculateNet = (x1, x2, x3, x4, w) => x1 \* w[1] + x2 \* w[2] + x3 \* w[3] + x4 \* w[4] + w[0] |
|  |  |
|  | const hammingDistance = (t, y) => { // считаем расстояние Хэмминга |
|  | let E = 0 |
|  | for (let i = 0; i < t.length; i++) { |
|  | if (t[i] !== y[i]) { |
|  | E++ |
|  | } |
|  | } |
|  | return E |
|  | } |
|  |  |
|  | const stepsNeuralNetwork = (X1, X2, X3, X4, w, combination) => { |
|  | let y = 0, outputVector = [], vectorT = [] |
|  | for (let i = 0; i < combination.length; i++) { |
|  | let x1 = X1[combination[i]], x2 = X2[combination[i]], |
|  | x3 = X3[combination[i]], x4 = X4[combination[i]] |
|  | let net = calculateNet(x1, x2, x3, x4, w) |
|  | //let out = hyperbolicTangentOut(net) |
|  | y = porogovayaFunc(net) |
|  | outputVector.push(y) |
|  | let t = tTeacher[combination[i]] |
|  | vectorT.push(t) |
|  | let delta = t - y |
|  | if (delta !== 0) { |
|  |  |
|  | w[0] += norm \* delta \* x0 |
|  | w[1] += norm \* delta \* x1 |
|  | w[2] += norm \* delta \* x2 |
|  | w[3] += norm \* delta \* x3 |
|  | w[4] += norm \* delta \* x4 |
|  | /\* |
|  | w[0] += norm \* delta \* x0 \* (1 / (2 \* Math.pow(((Math.exp(net) + Math.exp(-net)) / 2), 2))) |
|  | w[1] += norm \* delta \* x1 \* (1 / (2 \* Math.pow(((Math.exp(net) + Math.exp(-net)) / 2), 2))) |
|  | w[2] += norm \* delta \* x2 \* (1 / (2 \* Math.pow(((Math.exp(net) + Math.exp(-net)) / 2), 2))) |
|  | w[3] += norm \* delta \* x3 \* (1 / (2 \* Math.pow(((Math.exp(net) + Math.exp(-net)) / 2), 2))) |
|  | w[4] += norm \* delta \* x4 \* (1 / (2 \* Math.pow(((Math.exp(net) + Math.exp(-net)) / 2), 2)))\*/ |
|  | } else { |
|  | w[0] = w[0] |
|  | w[1] = w[1] |
|  | w[2] = w[2] |
|  | w[3] = w[3] |
|  | w[4] = w[4] |
|  | } |
|  | } |
|  |  |
|  | let countsErrorInCurrentEra = hammingDistance(vectorT, outputVector) |
|  |  |
|  | return { |
|  | weights: w, |
|  | Error: countsErrorInCurrentEra, |
|  | y: outputVector |
|  | } |
|  |  |
|  | } |
|  |  |
|  | const searchNetworkMinVector = w => { |
|  | let y = 0, |
|  | outputVector = [], |
|  | vectorT = [] |
|  | for (let i = 0; i < 16; i++) { |
|  | let x1 = X1[i], x2 = X2[i], |
|  | x3 = X3[i], x4 = X4[i] |
|  | let net = calculateNet(x1, x2, x3, x4, w) |
|  | //let out = hyperbolicTangentOut(net) |
|  | y = porogovayaFunc(net) |
|  | outputVector.push(y) |
|  | let t = tTeacher[i] |
|  | vectorT.push(t) |
|  | } |
|  |  |
|  | let countsErrorInCurrentEra = hammingDistance(vectorT, outputVector) |
|  |  |
|  | return { |
|  | weights: w, |
|  | Error: countsErrorInCurrentEra, |
|  | y: outputVector |
|  | } |
|  |  |
|  | } |
|  |  |
|  |  |
|  | const eraNeuralNetwork = (weights, combination) => { |
|  | let era = {} |
|  | let k = 0; |
|  | while (era.Error !== 0) { |
|  | era = stepsNeuralNetwork(X1, X2, X3, X4, weights, combination) |
|  | //console.log(era, 'эпоха ', k) |
|  | weights = era.weights |
|  | k++ |
|  | if (k > 150) break |
|  | } |
|  | return [weights, combination, k] |
|  |  |
|  | } |
|  |  |
|  | /\*const eraNeuralNetwork = weights => { |
|  | let era = {} |
|  | let k = 0; |
|  | while (era.Error !== 0) { |
|  | era = stepsNeuralNetwork(X1, X2, X3, X4, weights, N) |
|  | console.log(era, 'эпоха ', k) |
|  | weights = era.weights |
|  | k++ |
|  | if (k > 150) break |
|  | } |
|  | return weights |
|  |  |
|  | }\*/ |
|  |  |
|  |  |
|  | function\* combinationN(array, n) { |
|  | if (n === 1) { |
|  | for (const a of array) { |
|  | yield [a]; |
|  | } |
|  | return; |
|  | } |
|  |  |
|  | for (let i = 0; i <= array.length - n; i++) { |
|  | for (const c of combinationN(array.slice(i + 1), n - 1)) { |
|  | yield [array[i], ...c]; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | function\* combination\_ordered(array) { |
|  | for (let i = 1; i <= array.length; i++) { |
|  | yield\* combinationN(array, i); |
|  | } |
|  | } |
|  |  |
|  | let index = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] |
|  |  |
|  | let generator = combination\_ordered(index) |
|  |  |
|  | let eraSearchVector = {} |
|  | for (let combination of generator) { |
|  | let weight = [0, 0, 0, 0, 0] |
|  | eraSearchVector = eraNeuralNetwork(weight, combination) |
|  | let minVector = searchNetworkMinVector(eraSearchVector[0]) |
|  | if (minVector.Error !== 0) continue |
|  | else break |
|  | } |
|  |  |
|  | console.log("найденные веса: " + eraSearchVector[0] + "\n", |
|  | "индексы минимальных векторов: " + eraSearchVector[1] + "\n", "количество эпох: " + eraSearchVector[2]) |