МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ   
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №2

тема: «Простейшие однослойные нейрокомпьютерные сети.

Использование правила Хэбба.»

Выполнил: ст. группы МИВТ-221

Харитонов Сергей Дмитриевич

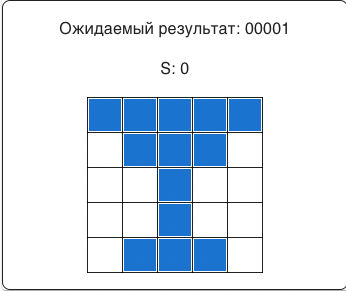
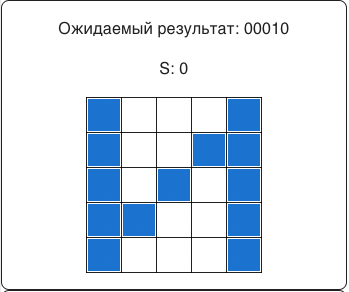
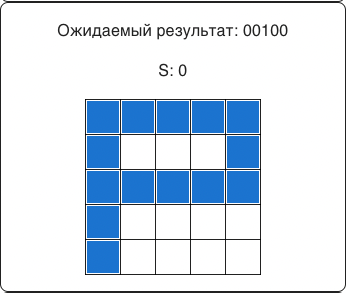
Белгород 2022 г.

**Лабораторная работа №2**

**Простейшие однослойные нейрокомпьютерные сети.   
Использование правила Хэбба.**

**Цель работы:** закрепление теоретических знаний, получение практических навыков использования однослойной нейрокомпьютерной сети, обученной на основе правила Хэбба.

Экранные копии образов для обучения



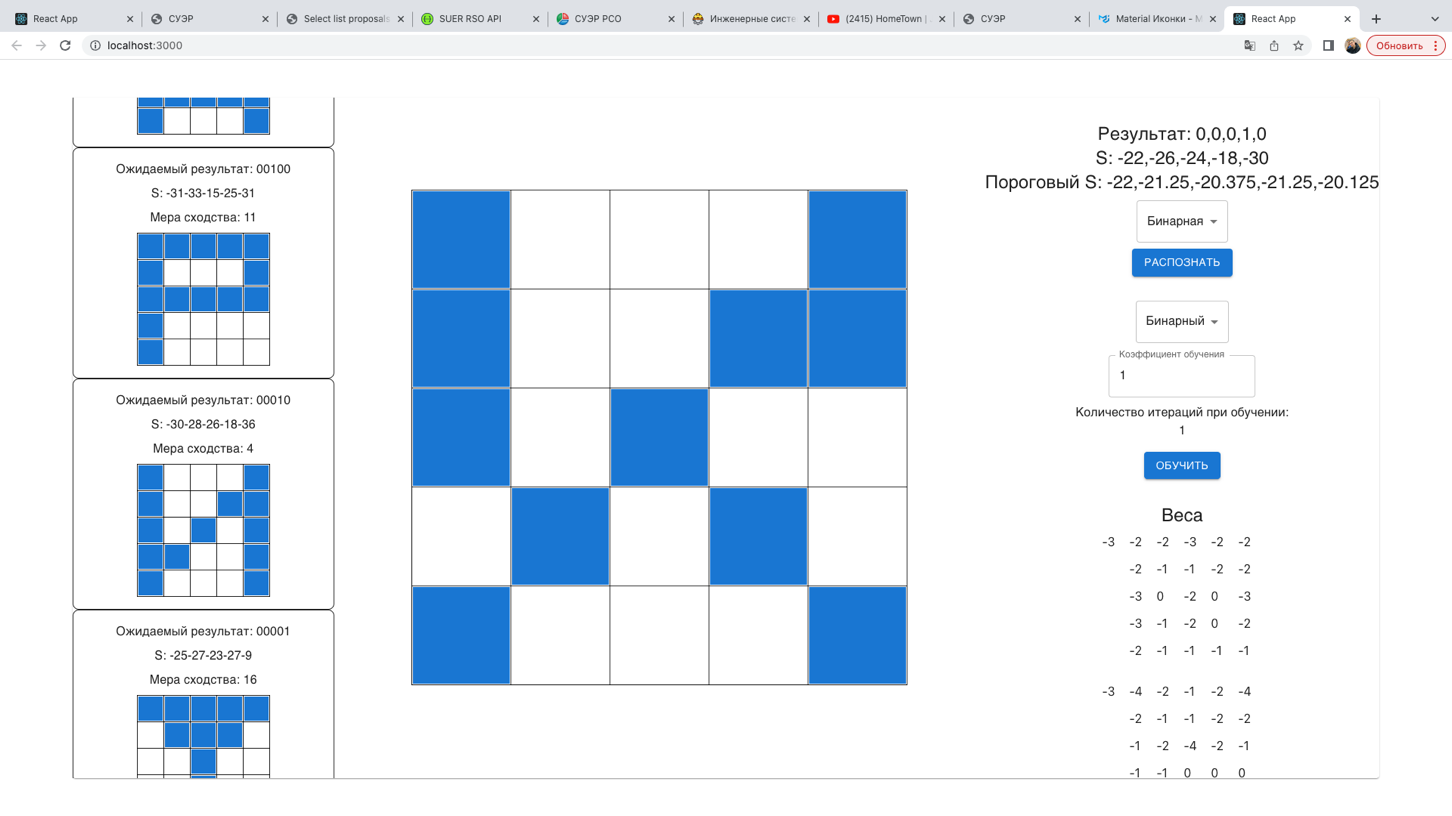
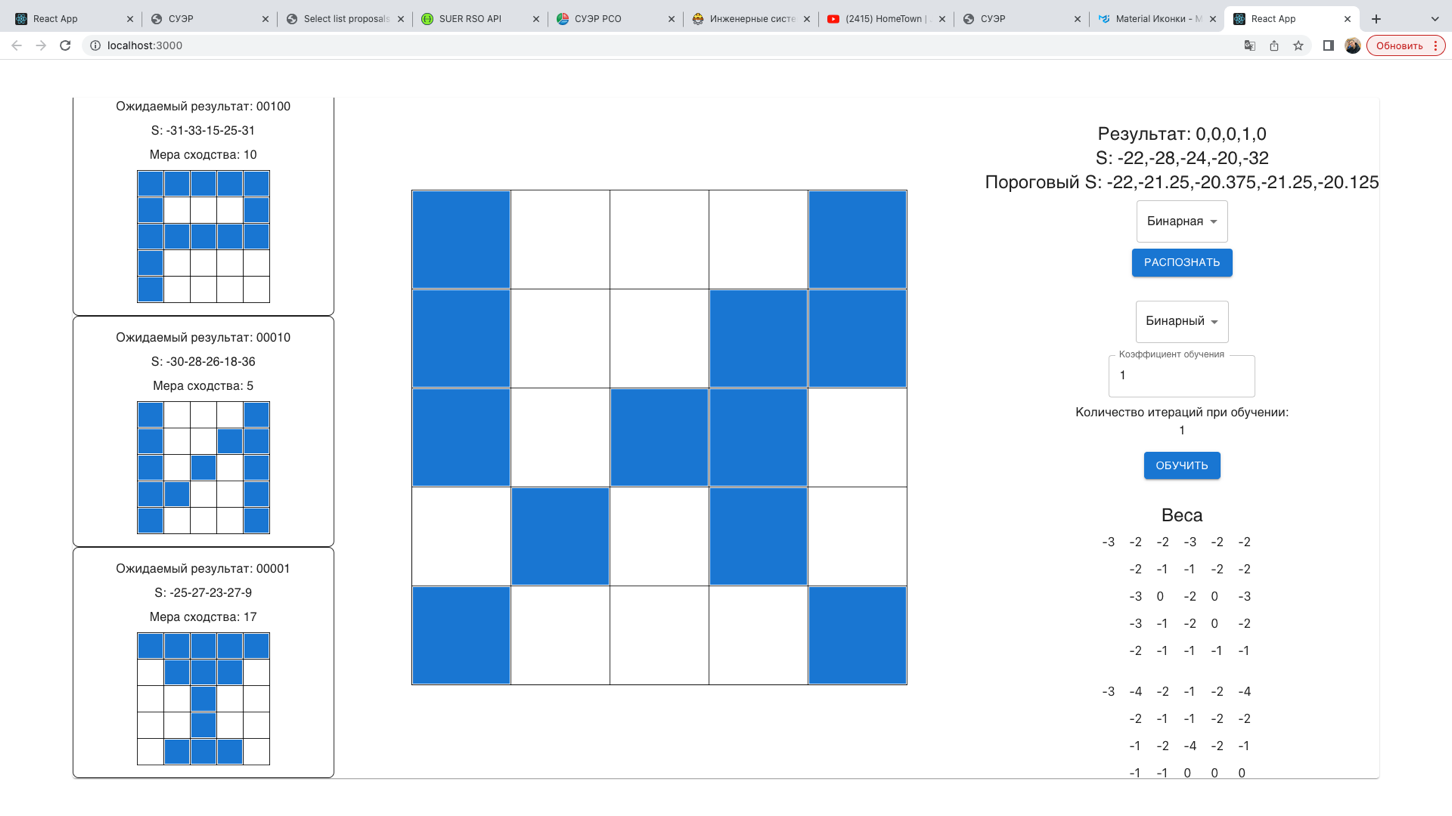
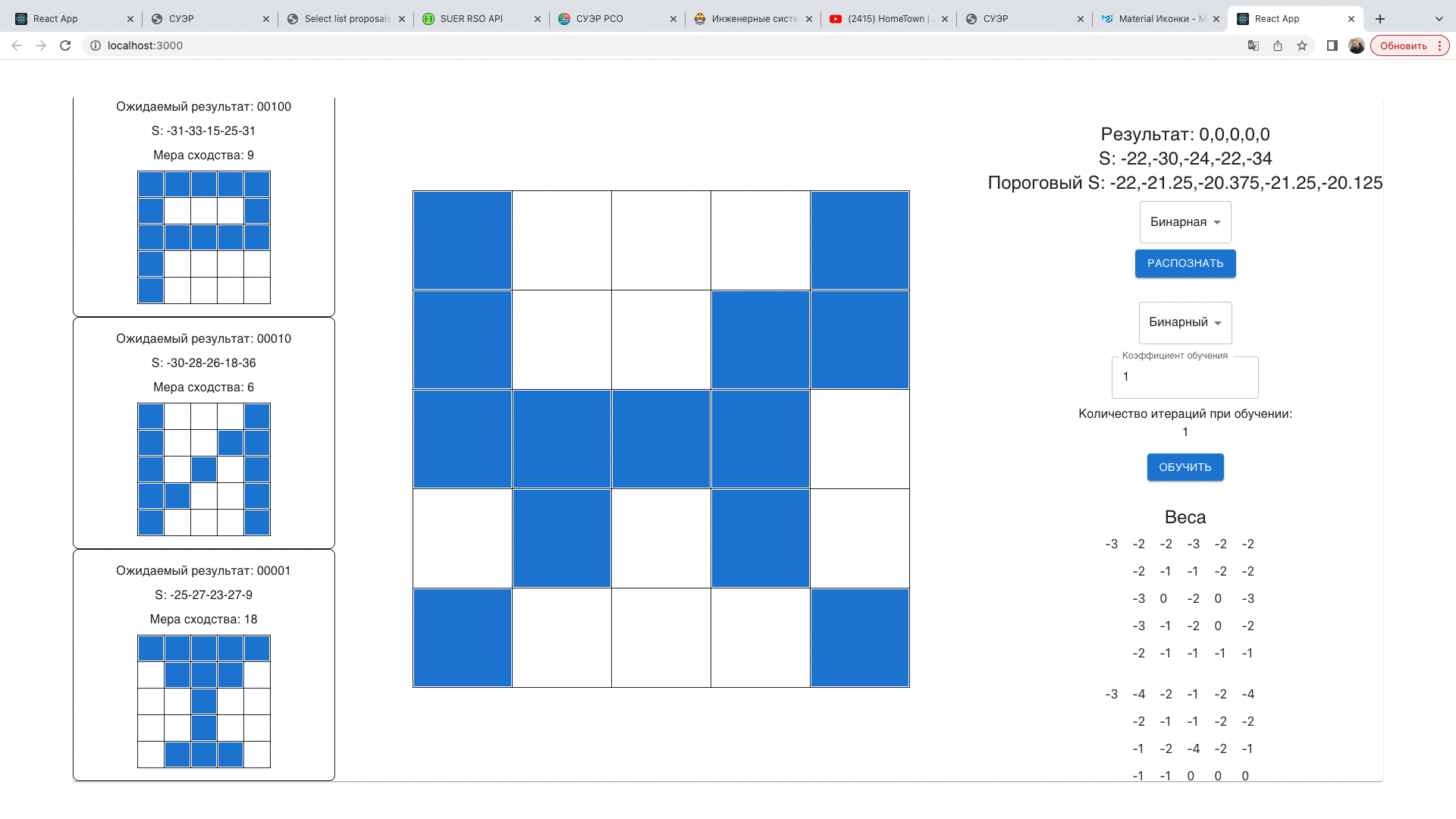
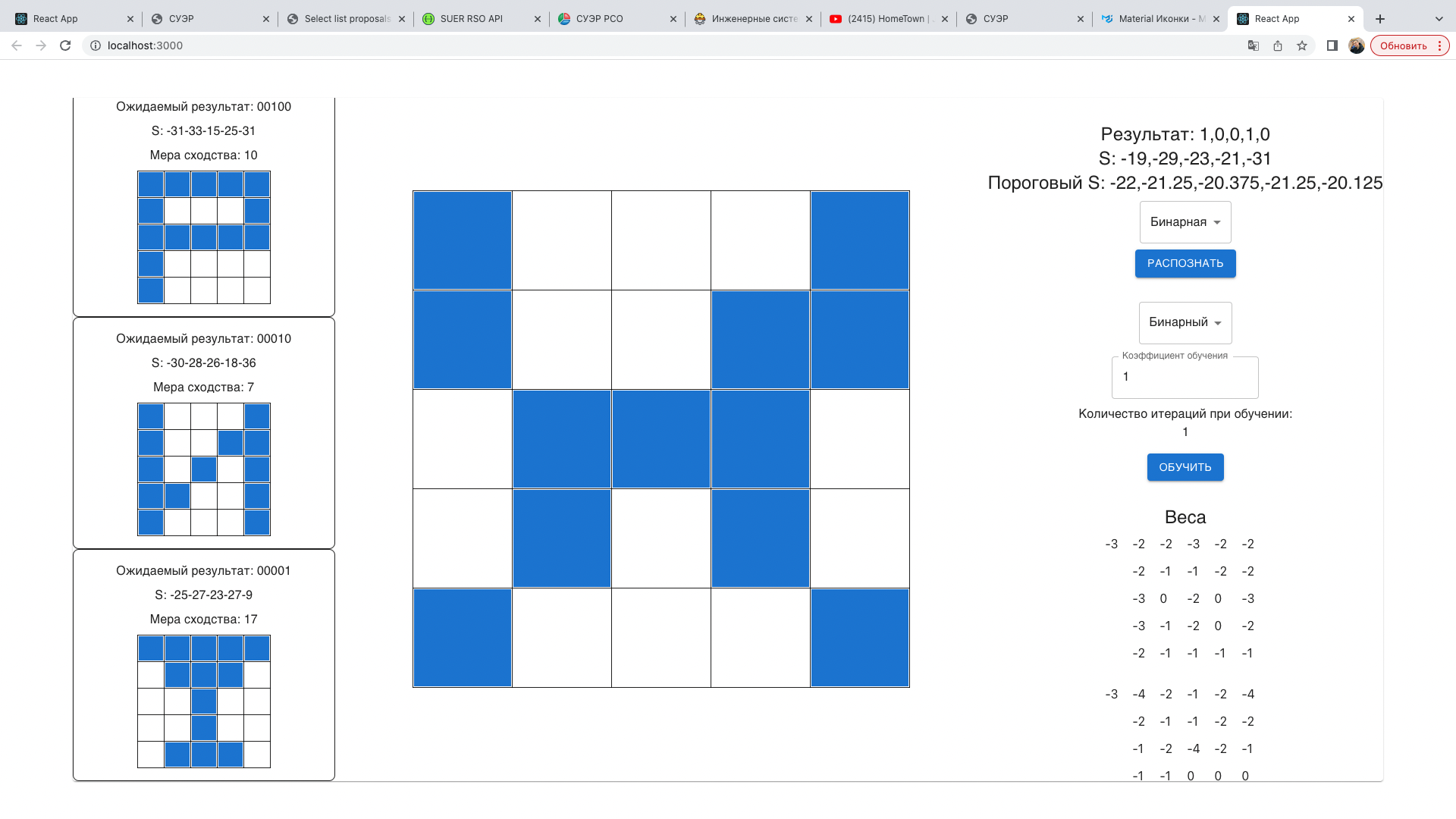
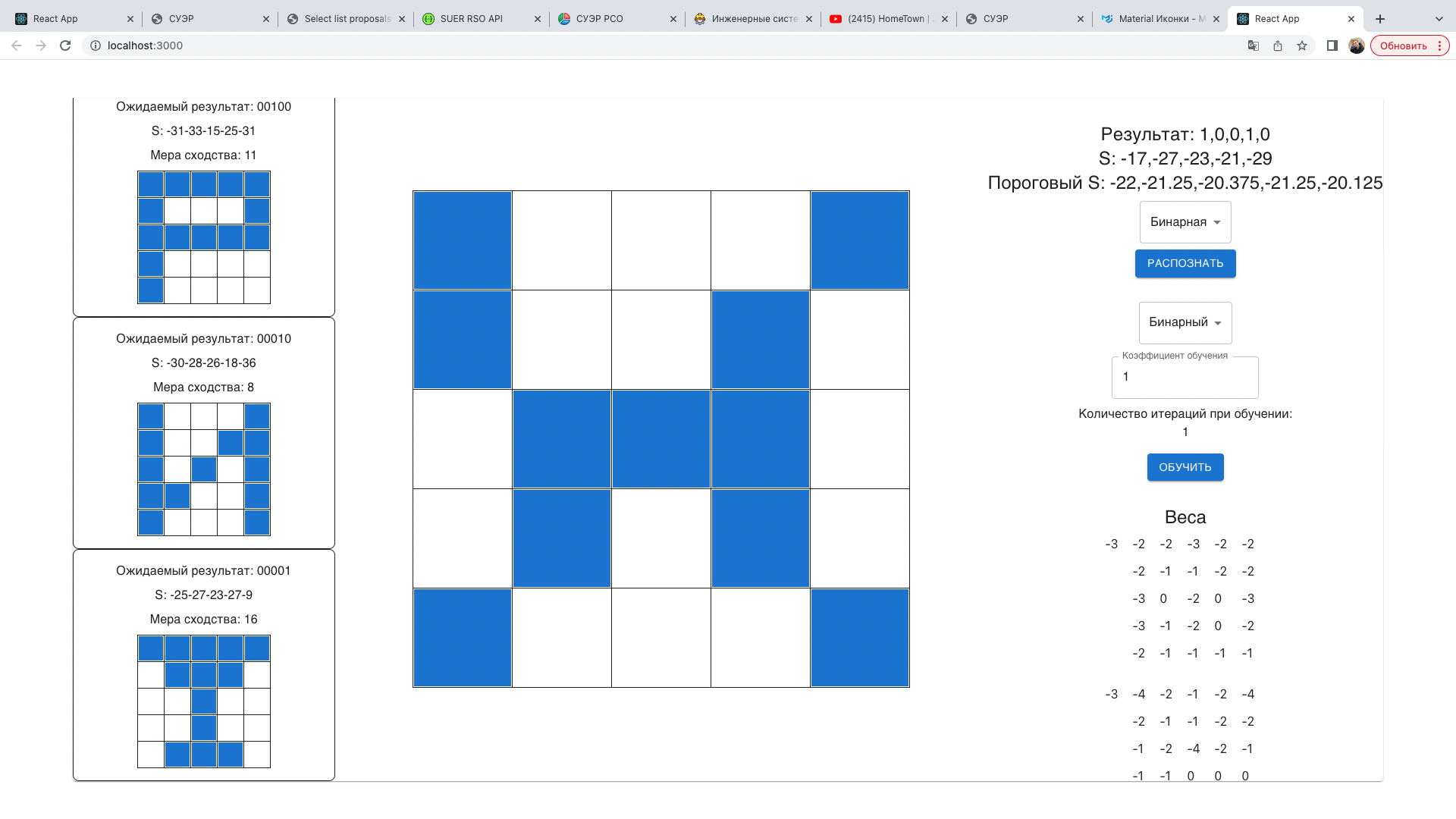
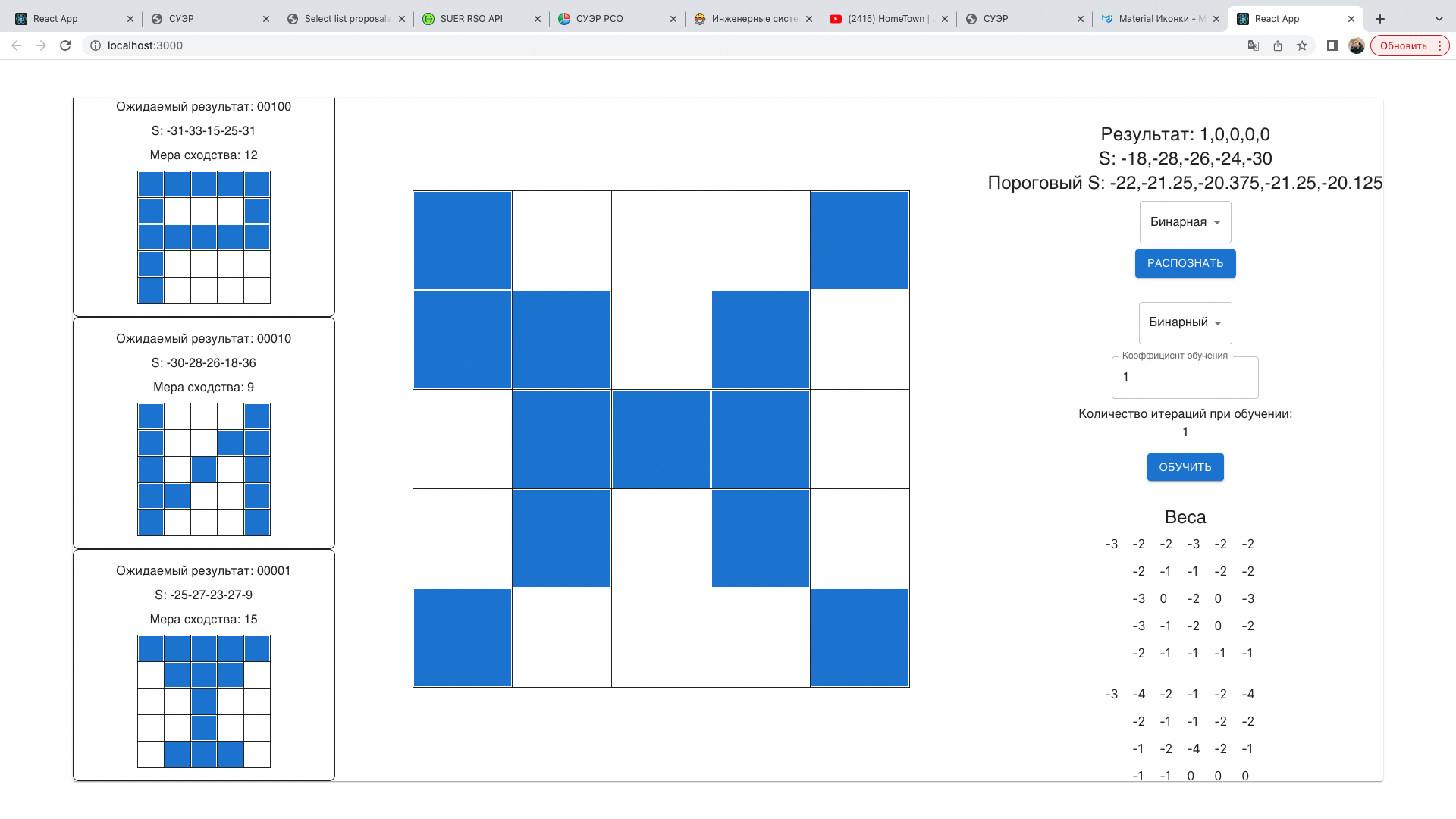
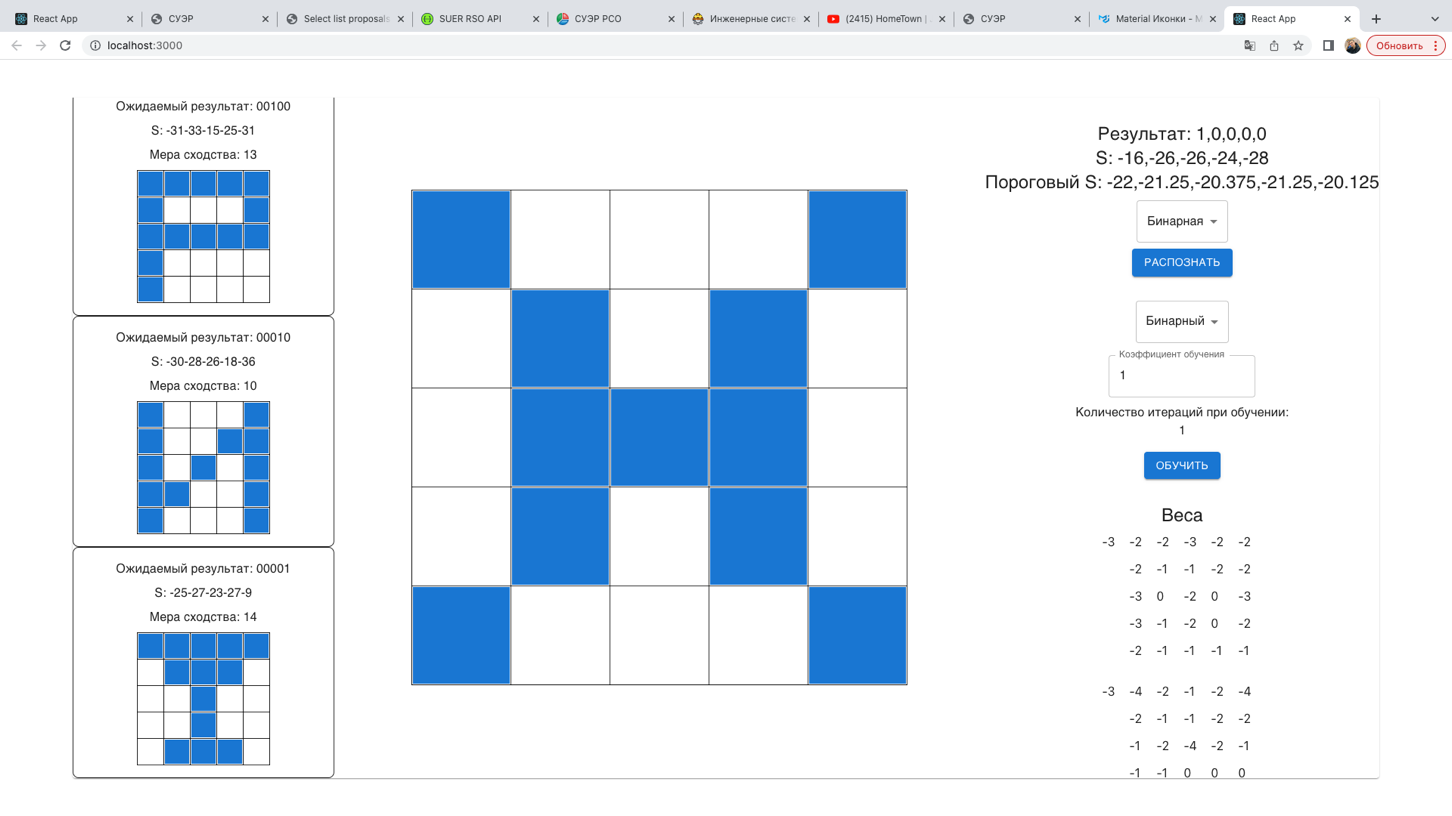
Результаты обучения

Изображение выглядит как текст

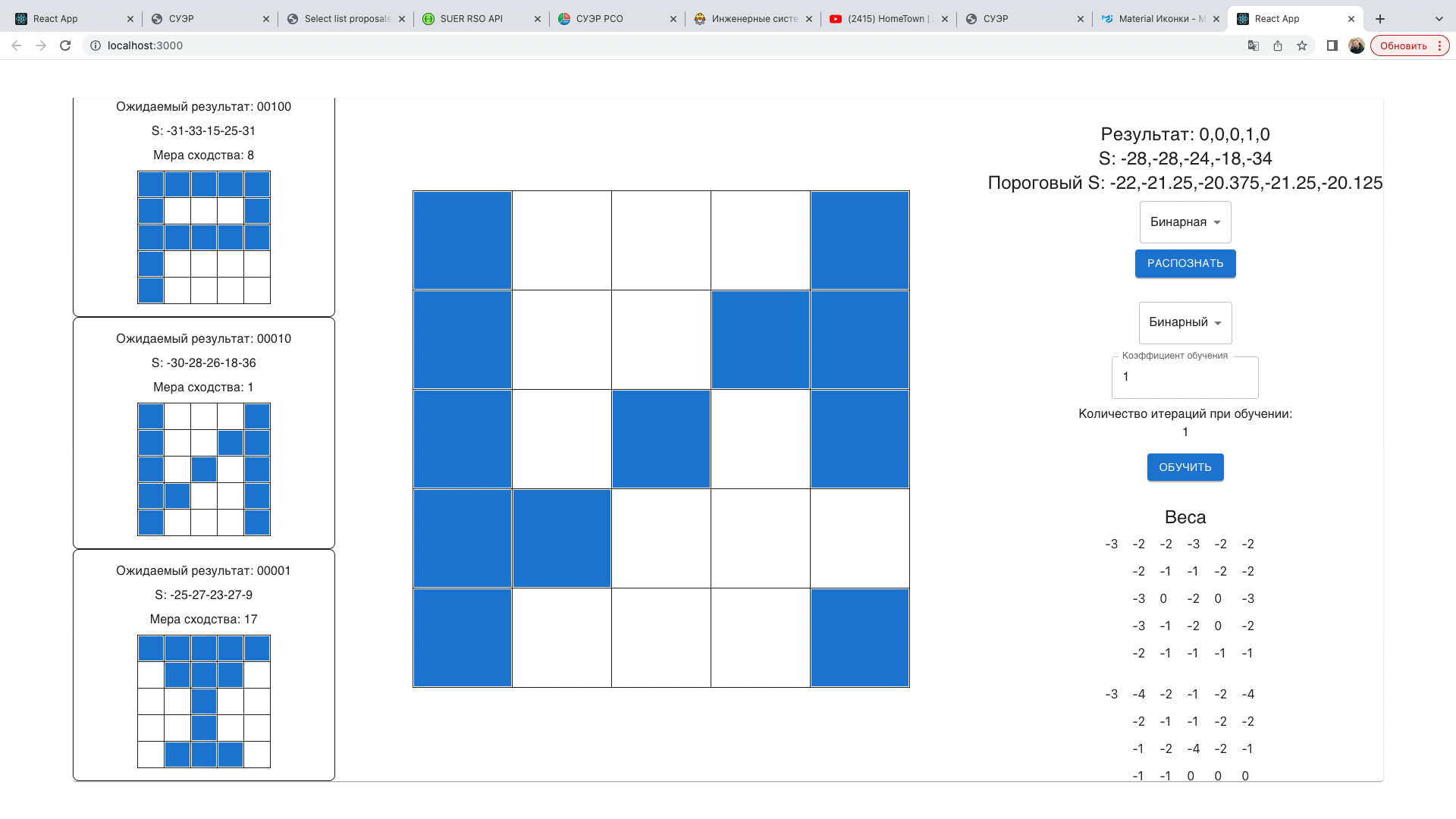
Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

Пошаговое исследование функций активации

Изображение выглядит как стол

Автоматически созданное описаниеИзображение выглядит как стол

Автоматически созданное описаниеИзображение выглядит как стол

Автоматически созданное описание

Результаты исследования

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Мера сходства 1 | | Мера сходства 2 | | Мера сходства 3 | | Мера сходства 4 | | Мера сходства 5 | | S1 | S2 | S3 | S4 | S5 |
| 0 | | 11 | | 13 | | 10 | | 14 | | -16 | -26 | -26 | -24 | -28 |
| 1 | | 12 | | 12 | | 9 | | 15 | | -18 | -28 | -26 | -24 | -30 |
| 2 | | 13 | | 11 | | 8 | | 16 | | -17 | -27 | -23 | -21 | -29 |
| 3 | | 14 | | 10 | | 7 | | 17 | | -19 | -29 | -23 | -21 | -31 |
| 4 | | 13 | | 9 | | 6 | | 18 | | -22 | -30 | -24 | -22 | -34 |
| 5 | | 12 | | 10 | | 5 | | 17 | | -22 | -28 | -24 | -20 | -32 |
| 6 | | 11 | | 11 | | 4 | | 16 | | -22 | -26 | -24 | -18 | -30 |
| 7 | | 10 | | 10 | | 3 | | 17 | | -25 | -27 | -25 | -19 | -33 |
| 8 | | 9 | | 9 | | 2 | | 18 | | -28 | -28 | -26 | -20 | -36 |
| 9 | | 10 | | 8 | | 1 | | 17 | | -28 | -28 | -24 | -18 | -34 |
| 10 | | 9 | | 9 | | 0 | | 18 | | -30 | -28 | -26 | -18 | -36 |
| Result1 | Result2 | | Result3 | | Result4 | | Result5 | |
| 1 | 0 | | 0 | | 0 | | 0 | |
| 1 | 0 | | 0 | | 0 | | 0 | |
| 1 | 0 | | 0 | | 1 | | 0 | |
| 1 | 0 | | 0 | | 1 | | 0 | |
| 0 | 0 | | 0 | | 0 | | 0 | |
| 0 | 0 | | 0 | | 1 | | 0 | |
| 0 | 0 | | 0 | | 1 | | 0 | |
| 0 | 0 | | 0 | | 1 | | 0 | |
| 0 | 0 | | 0 | | 1 | | 0 | |
| 0 | 0 | | 0 | | 1 | | 0 | |
| 0 | 0 | | 0 | | 1 | | 0 | |

Графики полученные в ходе исследования

Исходя из графиков видно, что на плавное изменение картинки от 1го образа к 4му, активно реагируют 1 и 4ый нейрон, обученные для распознавания 1го и 4го образа, так же проявляет активность 5ый нейрон, все больше уходя в минус по мере перехода от 1го образа к 4му

Листинг программы

import {*createEvent*, *createStore*} from "effector";  
  
  
export const $M = *createStore*([  
 {  
 x: [  
 1, 0, 0, 0, 1,  
 0, 1, 0, 1, 0,  
 0, 1, 1, 1, 0,  
 0, 1, 0, 1, 0,  
 1, 0, 0, 0, 1,  
 ],  
 t: [1, 0, 0, 0, 0],  
 S: 0  
 },  
 {  
 x: [  
 0, 0, 1, 0, 0,  
 0, 1, 0, 1, 0,  
 1, 0, 0, 0, 1,  
 1, 1, 1, 1, 1,  
 1, 0, 0, 0, 1,  
 ],  
 t: [0, 1, 0, 0, 0],  
 S: 0  
 },  
 {  
 x: [  
 1, 1, 1, 1, 1,  
 1, 0, 0, 0, 1,  
 1, 1, 1, 1, 1,  
 1, 0, 0, 0, 0,  
 1, 0, 0, 0, 0,  
 ],  
 t: [0, 0, 1, 0, 0],  
 S: 0  
 },  
 {  
 x: [  
 1, 0, 0, 0, 1,  
 1, 0, 0, 1, 1,  
 1, 0, 1, 0, 1,  
 1, 1, 0, 0, 1,  
 1, 0, 0, 0, 1,  
 ],  
 t: [0, 0, 0, 1, 0],  
 S: 0  
 },  
 {  
 x: [  
 1, 1, 1, 1, 1,  
 0, 1, 1, 1, 0,  
 0, 0, 1, 0, 0,  
 0, 0, 1, 0, 0,  
 0, 1, 1, 1, 0,  
 ],  
 t: [0, 0, 0, 0, 1],  
 S: 0  
 },  
])  
  
  
export const setSs = *createEvent*()

import {*attach*, *combine*, *createEffect*, *createEvent*, *createStore*} from "effector";  
import {$M} from '../presets'  
import {$weight} from "../weight";  
  
export const *getS* = (Xs, Ws) =>  
 Xs  
 .reduce(  
 (result, x, index) => result + x \* Ws[index],  
 0  
 )  
  
export const bipolarTeacher = ({  
 id: 1,  
 getNextWeight: (old, x, y, k = 1.0) => old + (x || -1) \* y \* k,  
 activation: (xs, w, S = 0) =>  
 *getS*(xs, w) > S  
 ? 1  
 : -1,  
 fieldY: 'bipolar',  
 name: 'Биполярный',  
 learningRate: 1.0  
})  
  
export const binTeacher = ({  
 id: 2,  
 getNextWeight: (old, x, y, k = 1.0) => {  
 let delta = 0;  
 if ((x === 1) && (y === 1))  
 delta = 1;  
 if ((x !== 0) && (y === 0))  
 delta = -1;  
 return old + delta \* k;  
 },  
 activation: (xs, w, S = 0) =>  
 *getS*(xs, w) > S  
 ? 1  
 : 0,  
 fieldY: 'bin',  
 name: 'Бинарный',  
 learningRate: 1.0  
})  
  
export const teachers = [bipolarTeacher, binTeacher]  
  
export const $teacher = *createStore*(binTeacher)  
  
export const changeTeacher = *createEvent*()  
  
export const teachEvent = *createEvent*()  
export const teachFx = *createEffect*(teachEvent)  
export const teach = *attach*({  
 effect: teachFx,  
 source: *combine*(  
 $M, $weight,  
 (M, weight) => ({M, weight})  
 ),  
 mapParams: (\_, data) => (data)  
})

import {  
 $teacher,  
 changeTeacher, *getS*,  
 teachEvent,  
} from './index'  
import {setWeights} from "../weight";  
import {setSs} from "../presets";  
import {setCountOperations} from "../info";  
import {*average*} from "../recognize";  
  
const handleChangeTeacher = (\_, teacher) => teacher  
  
const equalArray = (a, b) => {  
 if (a.length !== b.length)  
 return false  
  
 let equal = true;  
 a.forEach((itemA, indexB) => {  
 if (itemA !== b[indexB])  
 equal = false  
 })  
  
 return equal  
}  
  
const handleTeach = (teacher, {M, weight}) => {  
 const teacherMs = M.map(({x, ...m}) => ({...m, x: [1, ...x]}))  
 let W = weight.map(weightRow => weightRow.map(() => 0))  
  
 const checkNotEnd = () => W  
 .some(  
 (w, indexW) => {  
 const Ss = teacherMs.map(m => *getS*(m.x, w))  
 const {  
 Sone, Sother  
 } = Ss.reduce(  
 (result, s, index) => teacherMs[index].t[indexW]  
 ? ({  
 ...result,  
 Sone: result.Sone + s,  
 })  
 : ({  
 ...result,  
 Sother: result.Sother + s,  
 }),  
 {Sone: 0, Sother: 0}  
 )  
 return teacherMs.some(  
 (m, indexM) => {  
 const a = teacher.activation(m.x, w, *average*([Sone, Sother / 4]))  
 const a1 = m.t[indexW]  
 const t = teacher.activation(m.x, w, *average*([Sone, Sother / 4])) !== m.t[indexW]  
 return t  
 }  
 )  
 }  
 )  
  
 let countOperations = 0;  
 while (checkNotEnd()) {  
 ++countOperations;  
 teacherMs  
 .forEach(  
 m => {  
 W = W.map(  
 (wRow, indexWRow) =>  
 wRow  
 .map(  
 (oldW, index) =>  
 teacher.getNextWeight(  
 oldW,  
 m.x[index],  
 m.t[indexWRow],  
 teacher.learningRate  
 )  
 )  
 )  
 }  
 )  
 }  
 console.log(teacher)  
 setWeights(W)  
 setSs(  
 teacherMs  
 .map(  
 m => W.map(w => *getS*(m.x, w))  
 )  
 )  
 setCountOperations(countOperations)  
}  
  
$teacher  
 .on(changeTeacher, handleChangeTeacher)  
 .on(teachEvent, handleTeach)

import React, {*useEffect*, *useState*} from "react";  
import {Button, MenuItem, Select, Typography} from "@mui/material";  
import {$teacher, *getS*} from "../../models/teacher";  
import {*useStore*} from "effector-react";  
import {*makeStyles*} from "@mui/styles";  
import {$weight} from "../../models/weight";  
import {$input} from "../../models/input";  
import {$M} from "../../models/presets";  
import {$recognize, *average*, recognizeFunctions, setRecognize} from "../../models/recognize";  
  
const useStyles = *makeStyles*(() => ({  
 wrapper: {  
 margin: '32px 0',  
  
 display: 'flex',  
 flexDirection: 'column',  
 justifyContent: 'space-between',  
 alignItems: 'center'  
 }  
}))  
  
export default function *Result*() {  
 const classes = useStyles()  
  
 const map = *useStore*($input)  
 const teacher = *useStore*($teacher)  
 const weights = *useStore*($weight)  
 const M = *useStore*($M)  
 const recognize = *useStore*($recognize);  
  
 const [result, setResult] = *useState*(null)  
 const [S, setS] = *useState*(null)  
 const [frontierS, setFrontierS] = *useState*(null)  
  
 *useEffect*(() => {  
 setResult(null)  
 setS(null)  
 setFrontierS(null)  
 }, [teacher])  
  
 const handleRecognize = () => {  
 const Sfrontiers = weights.map((w, indexW) => {  
 const Ss = M.map(m => *getS*([1, ...m.x], w))  
 const {  
 Sone, Sother  
 } = Ss.reduce(  
 (result, s, index) => M[index].t[indexW]  
 ? ({  
 ...result,  
 Sone: result.Sone + s,  
 })  
 : ({  
 ...result,  
 Sother: result.Sother + s,  
 }),  
 {Sone: 0, Sother: 0}  
 )  
 return recognize.getFrontier(Sone, Sother / 4.)  
 })  
  
 setFrontierS(Sfrontiers)  
 setResult(  
 weights.map(  
 (w, index) => recognize.recognize([1, ...map], w, Sfrontiers[index], Sfrontiers[index])  
 )  
 )  
 setS(  
 weights.map(  
 w => *getS*([1, ...map], w)  
 )  
 )  
 }  
  
 return (  
 <div className={classes.wrapper}>  
 <Typography variant={'h5'}>  
 {  
 (result)  
 ? `Результат: ${result}`  
 : 'Сначала посчитай!'  
 }  
 </Typography>  
 <Typography variant={'h5'}>  
 {  
 (result)  
 ? `S: ${S}`  
 : 'Сначала посчитай!'  
 }  
 </Typography>  
 <Typography variant={'h5'}>  
 {  
 (result)  
 ? `Пороговый S: ${frontierS}`  
 : 'Сначала посчитай!'  
 }  
 </Typography>  
 <Select  
 style={{marginTop: '8px'}}  
 value={recognize.id}  
 onChange={  
 ({target}) =>  
 setRecognize(  
 target.value  
 )  
 }  
 >  
 {  
 recognizeFunctions.map(({id, name}) => (  
 <MenuItem  
 value={id}  
 key={id}  
 >  
 {name}  
 </MenuItem>  
 ))  
 }  
 </Select>  
 <Button  
 style={{marginTop: '8px'}}  
 variant={'contained'}  
 onClick={handleRecognize}  
 >  
 Распознать  
 </Button>  
 </div>  
 )  
}