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 №3

тема: «Однослойная нейрокомпьютерная сеть.

Использование Дельта-правила.»

Выполнил: ст. группы МИВТ-221

Харитонов Сергей Дмитриевич

Белгород 2022 г.

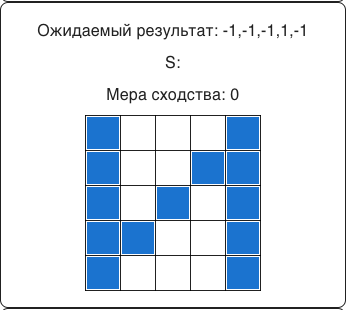
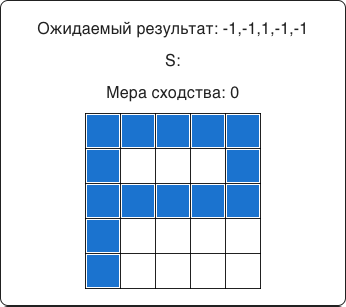
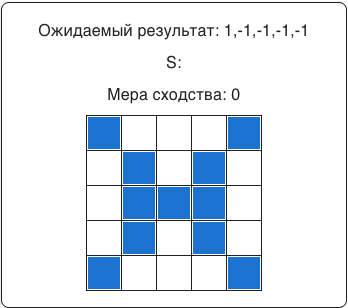
**Лабораторная работа №3**

**Однослойная нейрокомпьютерная сеть.**

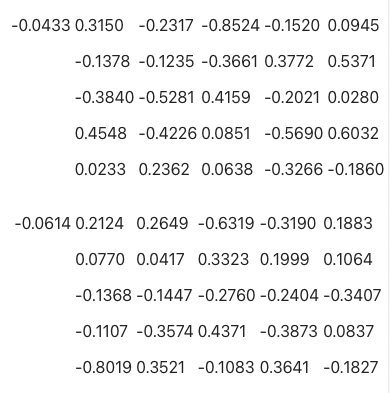
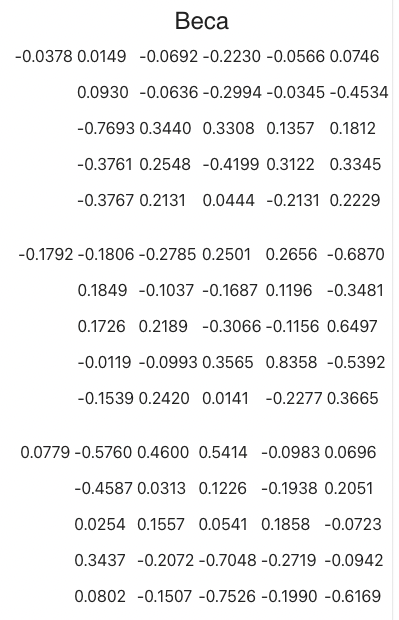
**Использование Дельта-правила**

**Цель работы:** Приобретение и закрепление знаний, получение практических навыков работы с Дельта-правилом.

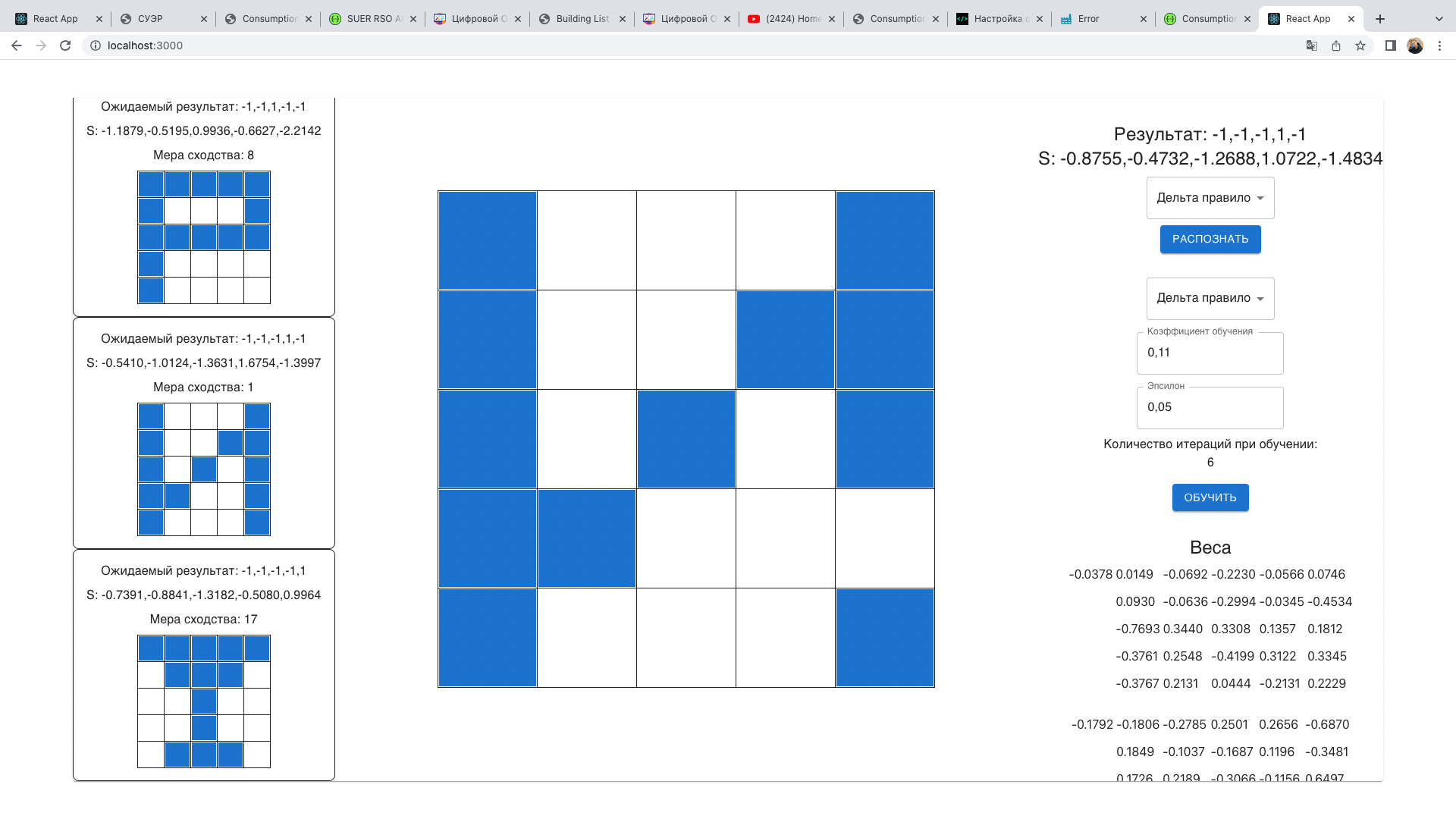
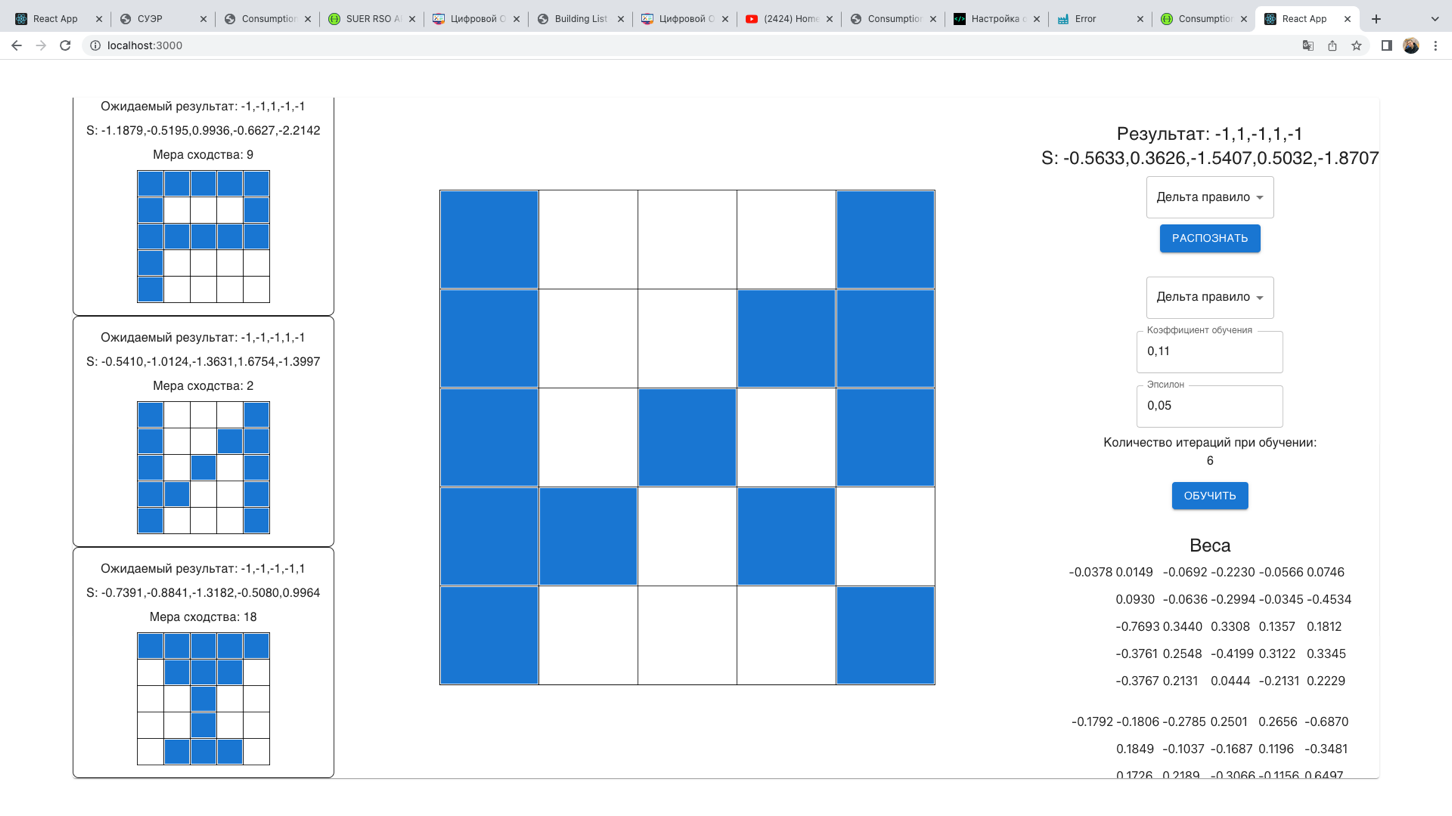
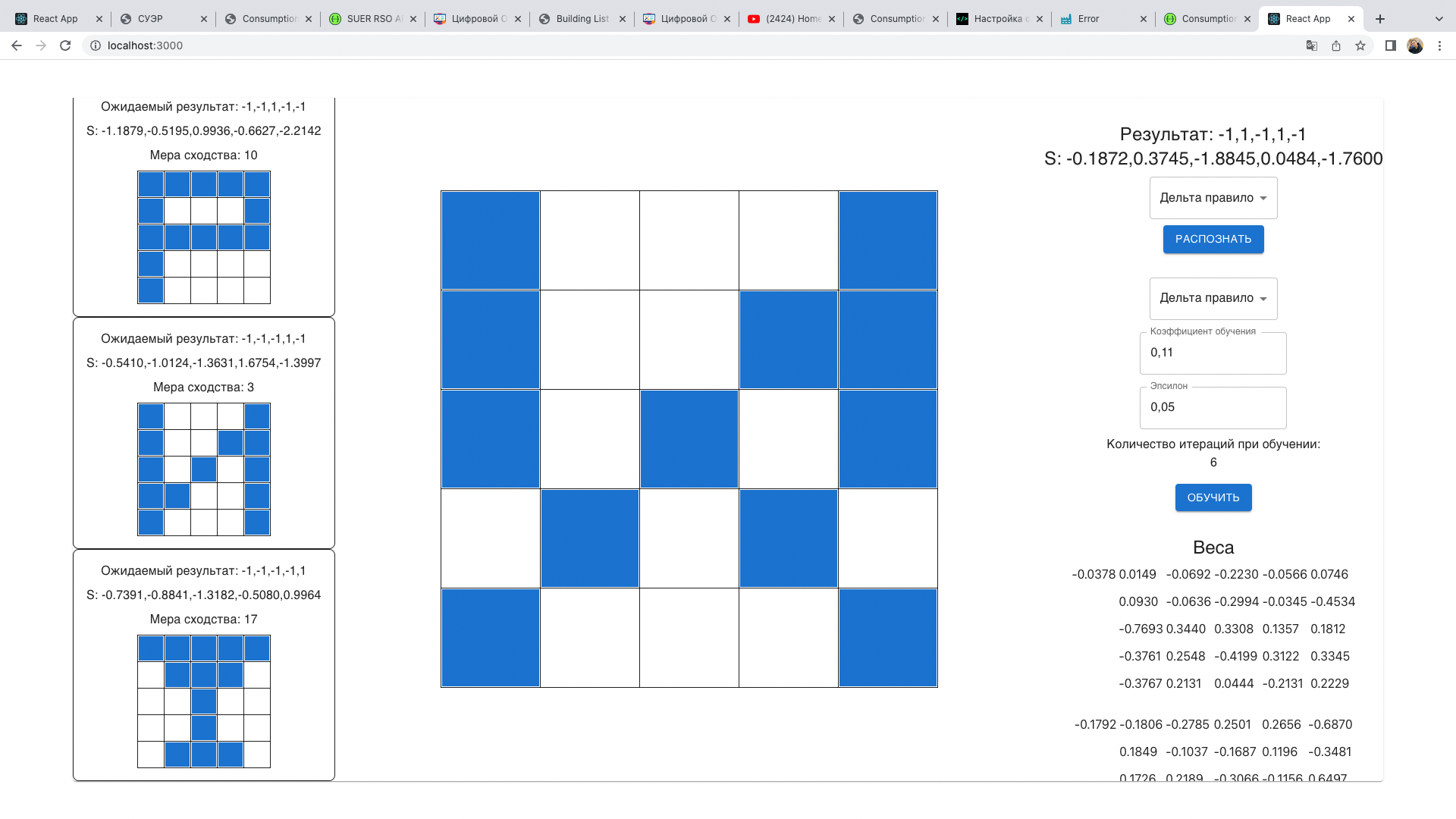
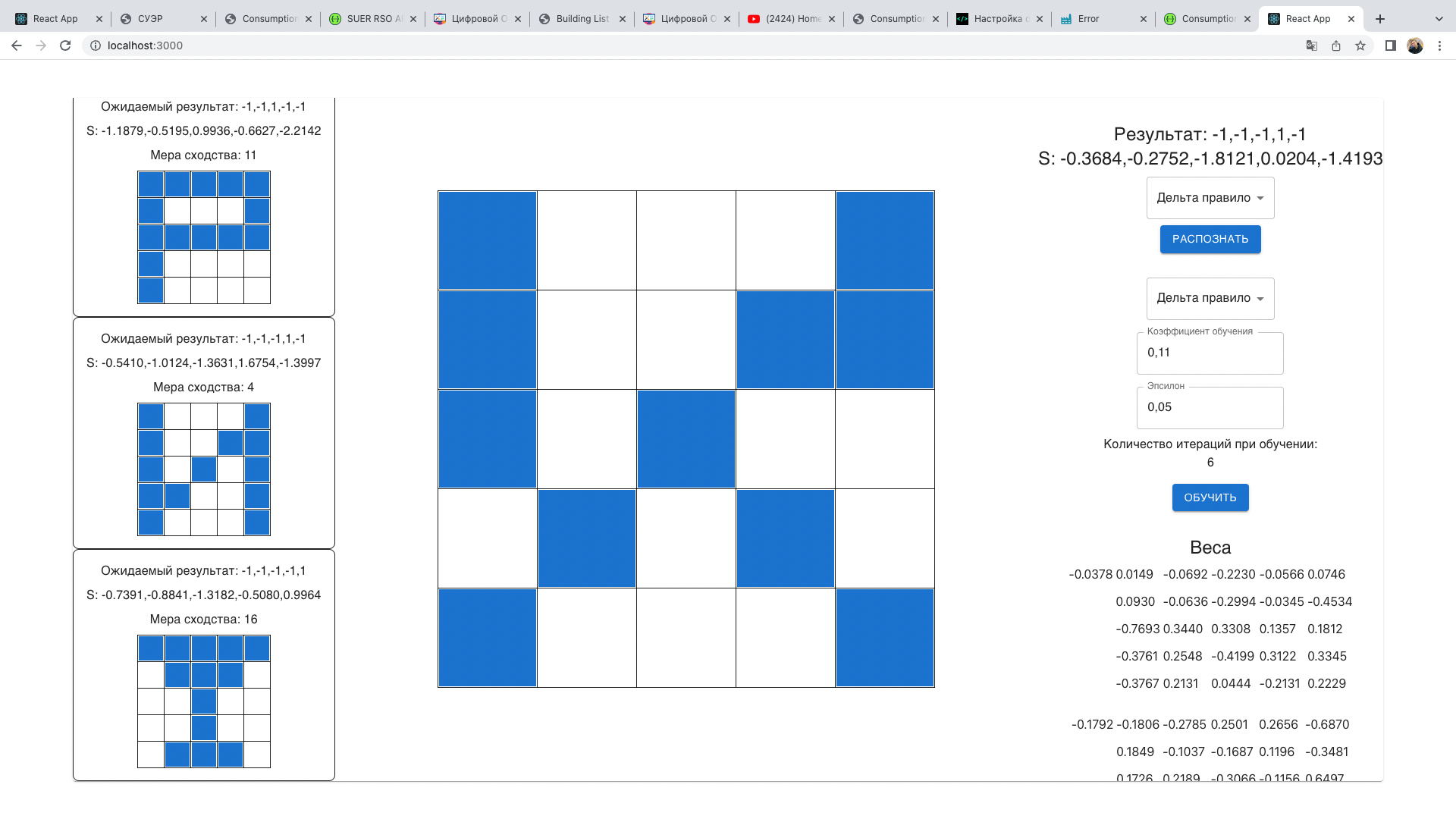
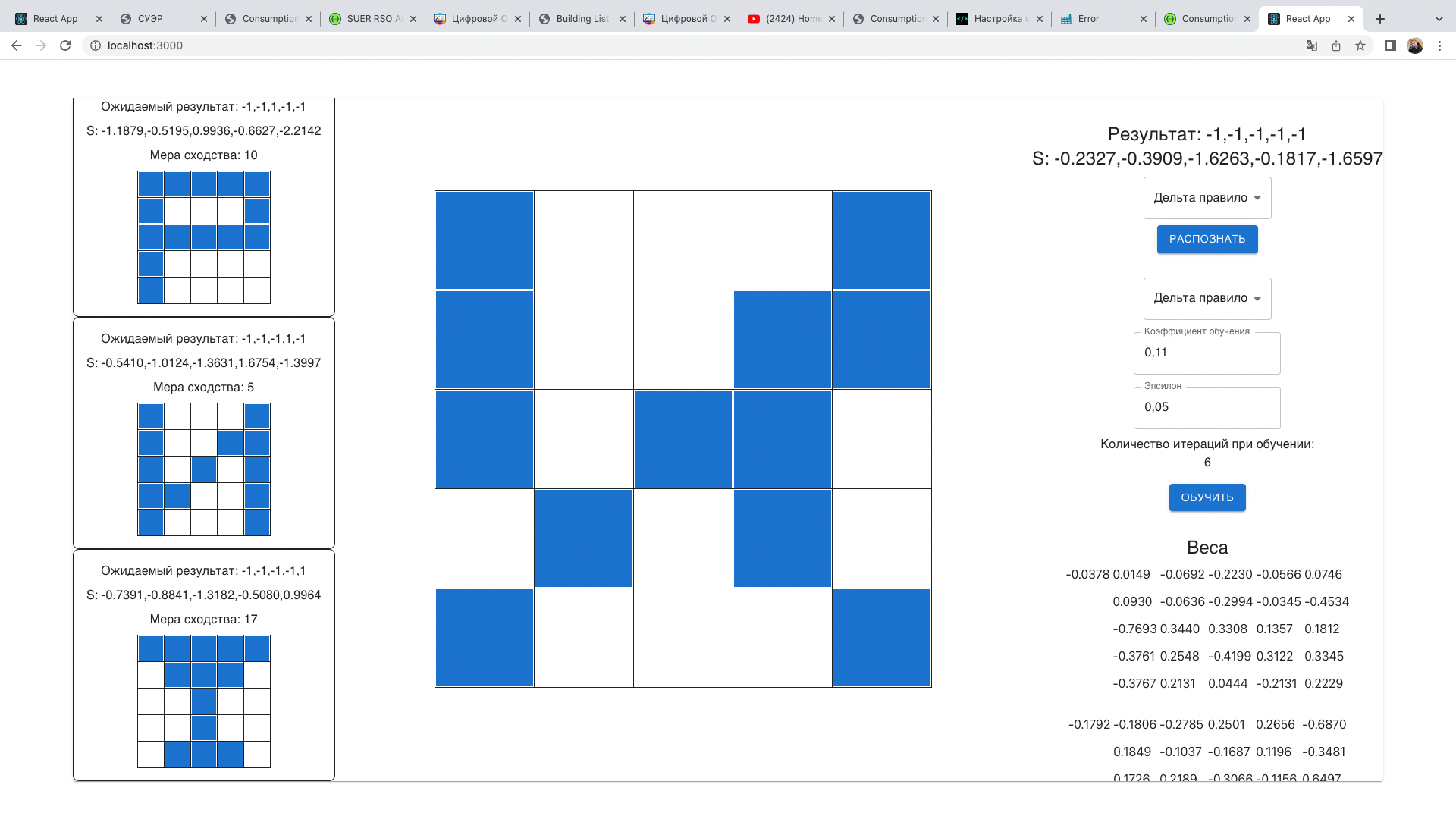
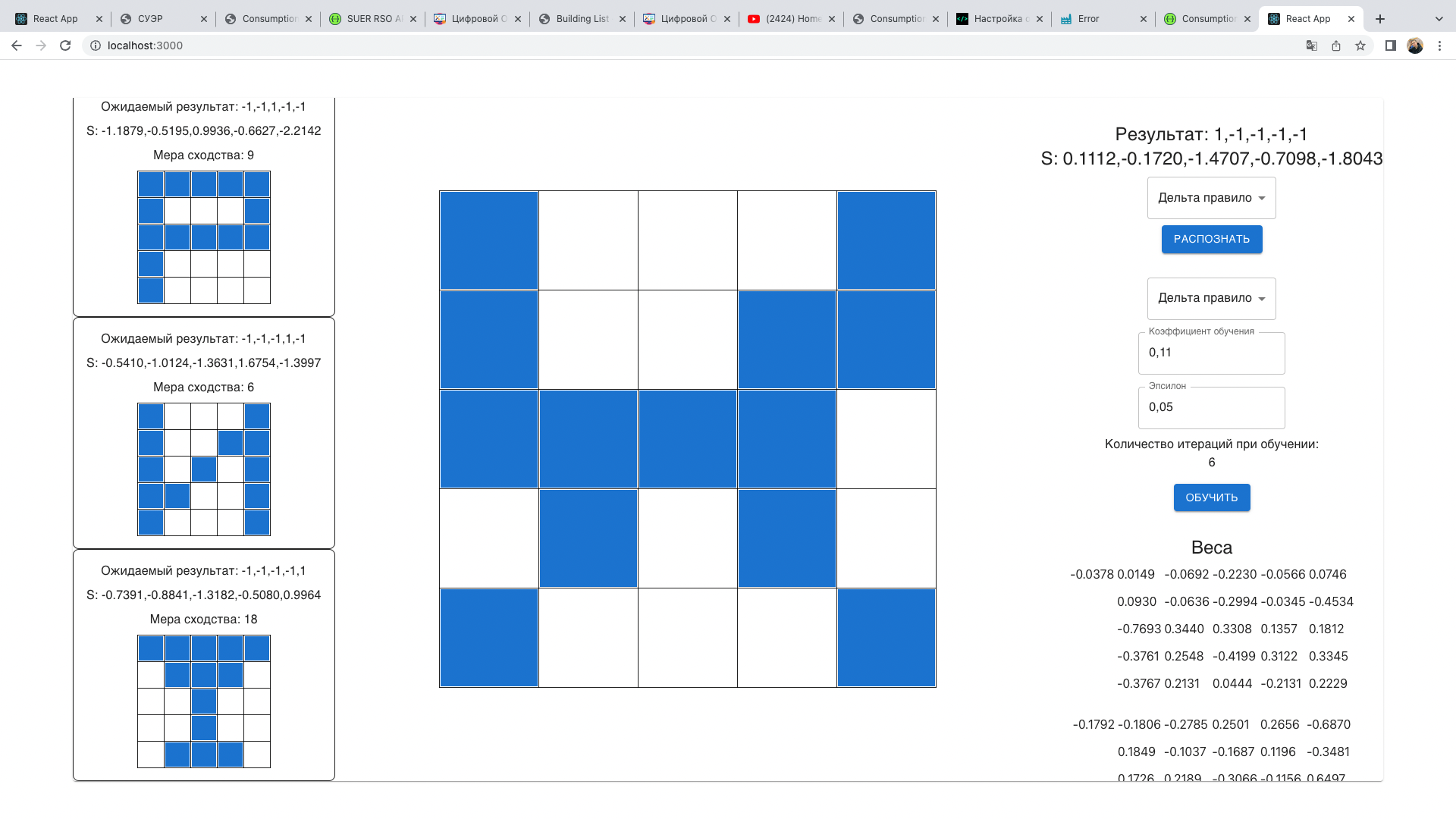
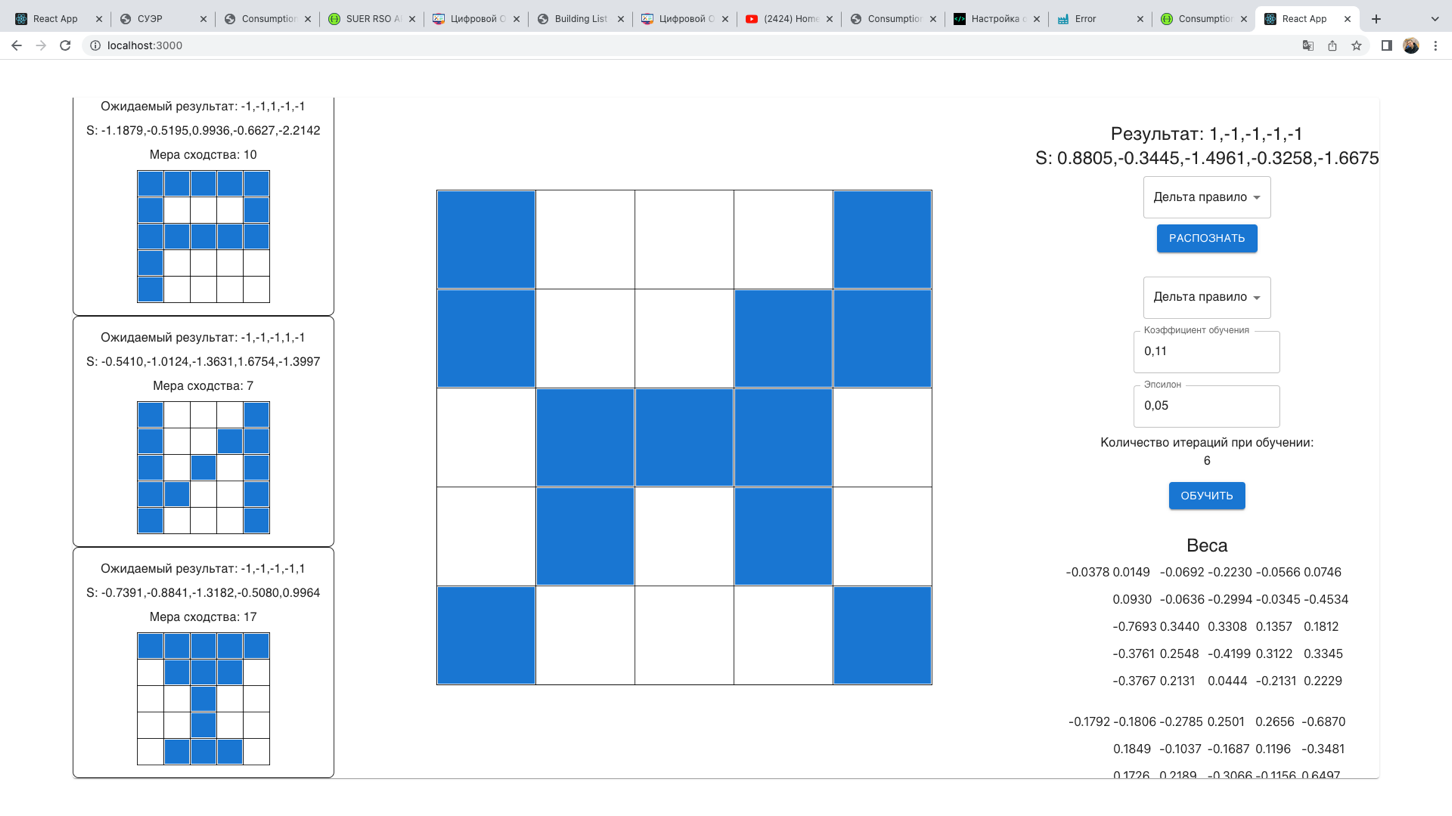
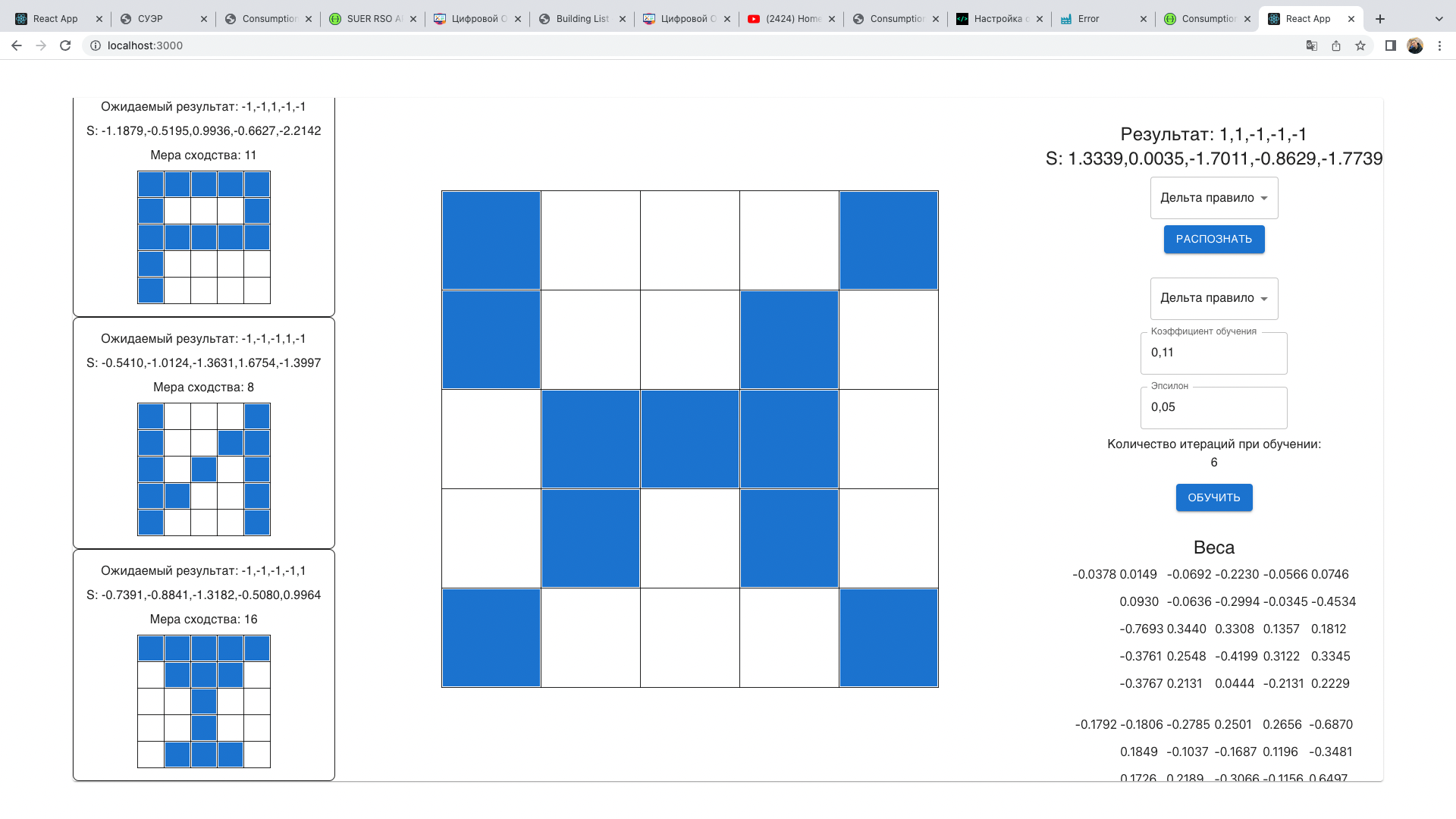
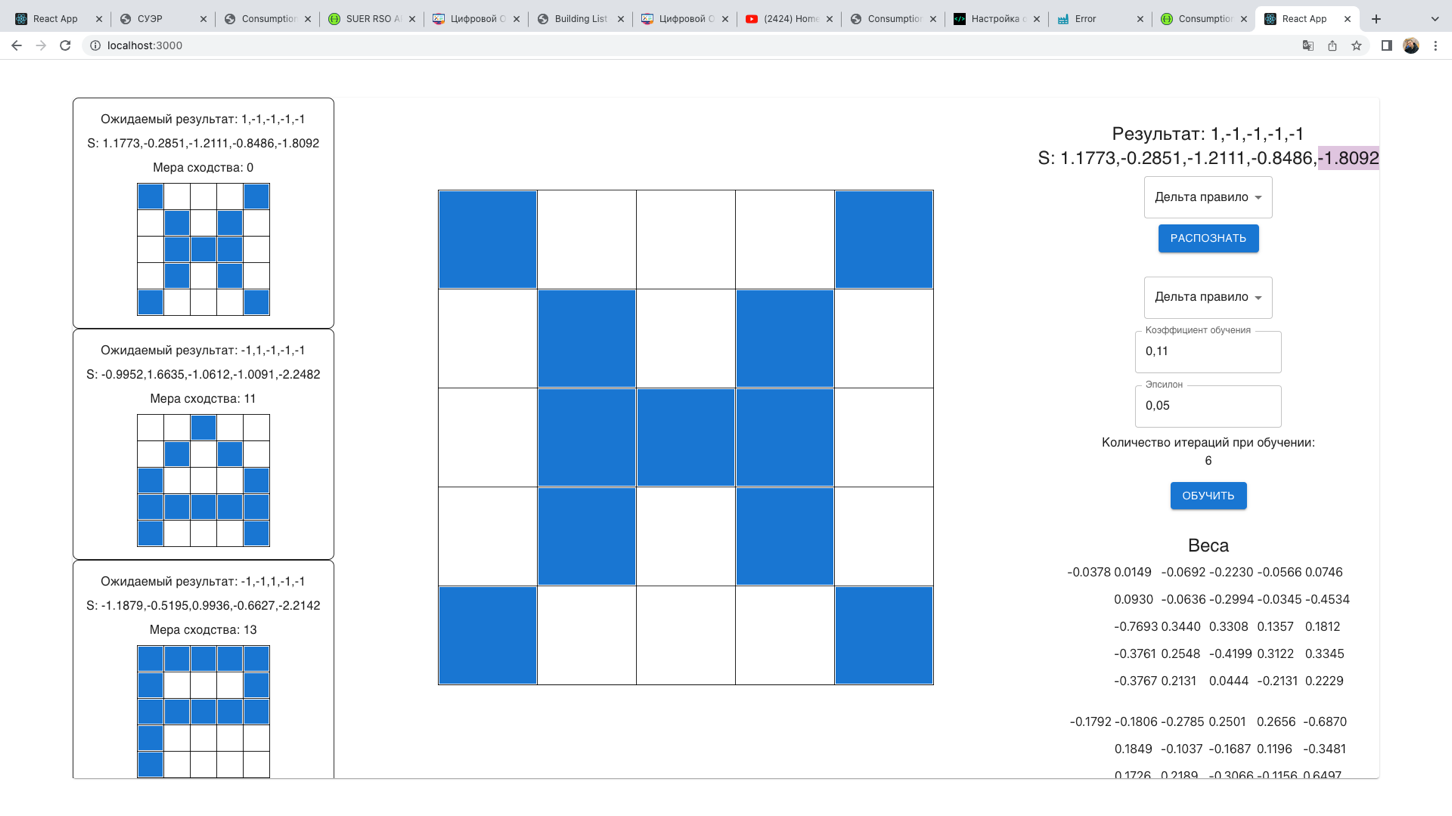
Экранные копии образов для обучения



Результаты обучения



Пошаговое исследование функций активации



Результаты исследования

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Мера сходства 1 | Мера сходства 2 | Мера сходства 3 | Мера сходства 4 | Мера сходства 5 | Result  1 | Result  2 | Result  3 | Result  4 | Result  5 |
| 0 | 11 | 13 | 10 | 14 | 1,1773 | -0,2851 | -1,2111 | -0,8486 | -1,8092 |
| 1 | 12 | 12 | 9 | 15 | 1,2703 | -0,1002 | -1,6698 | -0,9864 | -1,7322 |
| 2 | 13 | 11 | 8 | 16 | 1,3339 | 0,0035 | -1,7011 | -0,8629 | -1,7739 |
| 3 | 14 | 10 | 7 | 17 | 0,8805 | -0,3445 | -1,4961 | -0,3258 | -1,6675 |
| 4 | 13 | 9 | 6 | 18 | 0,1112 | -0,172 | -1,4707 | -0,7098 | -1,8043 |
| 5 | 12 | 10 | 5 | 17 | -0,2327 | -0,3909 | -1,6263 | -0,1817 | -1,6597 |
| 6 | 11 | 11 | 4 | 16 | -0,3684 | -0,2752 | -1,8121 | 0,0204 | -1,4193 |
| 7 | 10 | 10 | 3 | 17 | -0,1872 | 0,3745 | -1,8845 | 0,0484 | -1,76 |
| 8 | 9 | 9 | 2 | 18 | -0,5633 | 0,3626 | -1,5407 | 0,5032 | -1,8707 |
| 9 | 10 | 8 | 1 | 17 | -0,8755 | -0,4732 | -1,2688 | 1,0722 | -1,4834 |
| 10 | 9 | 9 | 0 | 18 | -0,541 | -1,0124 | -1,3631 | 1,6754 | -1,3997 |

Графики полученные в ходе исследования

Как и в случае с правилом Хэбба из второй лабораторной работы, активно реагируют 1ый и 4ый нейрон, 5ый в отличии от правила Хэбба так же спокоен как 2ой и 3ий.

Сравнительный анализ Дельта правила и правила Хэбба

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | epsilon=0,05 | epsilon=0,05 |
| Альфа | Х | E | E' |
| 0,09 | 1 | 13 | 10 |
| 0,1 | 1 | 10 | 7 |
| 0,11 | 1 | 9 | 5 |
| 0,13 | 1 | 7 | 4 |
| 0,15 | 1 | не обучается | не обучается |

Алфа выбиралась из соображения 0,1 <= n\*alfa <= 0,9. Как видим правило Хэбба каждый раз обучалось за одну итерацию, но это происходит исключительно благодаря хорошо выбранным образам для обучения, при их изменении нейронная сеть не способна обучиться по правилу Хэбба, при этом количество итераций Дельта увеличивается не драматически. Что делает Дельта правило гораздо более мощным инструментом, хоть и более затратным в некоторых случаях.

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, -1, -1, -1, -1],  
 S: [],  
 diff: 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: [-1, 1, -1, -1, -1],  
 S: [],  
 diff: 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: [-1, -1, 1, -1, -1],  
 S: [],  
 diff: 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: [-1, -1, -1, 1, -1],  
 S: [],  
 diff: 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: [-1, -1, -1, -1, 1],  
 S: [],  
 diff: 0  
 },  
])  
  
  
export const setSs = *createEvent*()  
export const setDiffs = *createEvent*()

import {*getS*} from "../teacher";  
import {*createEvent*, *createStore*} from "effector";  
  
export function *average*(nums) {  
 return nums.reduce((a, b) => (a + b)) / nums.length;  
}  
  
export const recognizeFunctions = [  
 {  
 id: 1,  
 recognize: (xs, ws) => *getS*(xs, ws),  
 name: 'Дельта правило'  
 },  
  
]  
  
export const $recognize = *createStore*(recognizeFunctions[0])  
  
export const setRecognize = *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 *getDiff* = (A, B) => A.reduce(  
 (res, a, indexB) => {  
 if (a !== B[indexB])  
 return res + 1;  
 return res;  
 },  
 0  
)  
  
export const deltaTeacher = ({  
 id: 1,  
 getNextWeight: (old, a, Uout, Uin, x) => old + a\*(Uout - Uin)\*x,  
 name: 'Дельта правило',  
 learningRate: 0.11,  
 epsilon: 0.05  
})  
  
export const teachers = [deltaTeacher]  
  
export const $teacher = *createStore*(deltaTeacher)  
  
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";  
  
const handleChangeTeacher = (\_, teacher) => teacher  
const handleTeach = (teacher, {M, weight}) => {  
 const teacherMs = M.map(({x, ...m}) => ({...m, x: [1, ...x]}))  
 let W = weight.map(w => [...w])  
  
 const teached = []  
  
 let countOperations = 0;  
 while (teached.length !== 5) {  
 ++countOperations;  
 teacherMs  
 .forEach(  
 m => {  
 W = W.map(  
 (wRow, indexWRow) => {  
 if (teached.includes(indexWRow))  
 return wRow  
  
  
 const newWRow = wRow  
 .map(  
 (oldW, index) =>  
 teacher.getNextWeight(  
 oldW,  
 teacher.learningRate,  
 m.t[indexWRow],  
 *getS*(m.x, wRow),  
 m.x[index]  
 )  
 )  
  
 *// const delta = newWRow.reduce(  
 // (processedDelta, newW, indexW) => Math.abs(newW - wRow[indexW]) + processedDelta,  
 // 0.0  
 // )  
 // if (delta <= teacher.epsilon)  
 // teached.push(indexWRow)* return newWRow  
 }  
 )  
 }  
 )  
  
 W.forEach(  
 (w, indexWRow) => {  
 if (teached.includes(indexWRow))  
 return;  
 const delta = teacherMs.reduce(  
 (delta, m) => delta + Math.pow((m.t[indexWRow] - *getS*(m.x, w)), 2),  
 0.0  
 )  
 if (delta < teacher.epsilon)  
 teached.push(indexWRow)  
 }  
 )  
  
 }  
 setWeights(W)  
 setSs(  
 teacherMs  
 .map(  
 m => W.map(w => *getS*(m.x, w))  
 )  
 )  
 setCountOperations(countOperations)  
}  
  
$teacher  
 .on(changeTeacher, handleChangeTeacher)  
 .on(teachEvent, handleTeach)

import {*createEvent*, *createStore*} from "effector";  
  
const getRandowW = () => [  
 0,  
 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0,  
 0, 0, 0, 0, 0,  
].map(() => (Math.random() - 0.5) / 10.)  
  
export const zeroWeight = [1, 2, 3, 4, 5].map(getRandowW)  
export const $weight = *createStore*(zeroWeight)  
  
export const setWeights = *createEvent*()

import React from "react";  
import {Button, MenuItem, Select, *TextField*, Typography} from "@mui/material";  
import {$teacher, changeTeacher, teach, teachers} from "../../models/teacher";  
import {*useStore*} from "effector-react";  
import {*makeStyles*} from "@mui/styles";  
import {$info} from "../../models/info";  
  
const useStyles = *makeStyles*(() => ({  
 wrapper: {  
 marginBottom: '32px',  
  
 display: 'flex',  
 flexDirection: 'column',  
 justifyContent: 'space-between',  
 alignItems: 'center'  
 }  
}))  
  
export default function *Teacher*() {  
 const classes = useStyles()  
  
 const teacher = *useStore*($teacher)  
 const {countOperations} = *useStore*($info)  
  
 return (  
 <div className={classes.wrapper}>  
 <Select  
 value={teacher.id}  
 onChange={  
 ({target}) =>  
 changeTeacher(  
 teachers.find(({id}) => id === target.value)  
 )  
 }  
 >  
 {  
 teachers.map(({id, name}) => (  
 <MenuItem  
 value={id}  
 key={id}  
 >  
 {name}  
 </MenuItem>  
 ))  
 }  
 </Select>  
 <TextField  
 style={{marginTop: '16px'}}  
 label={'Коэффициент обучения'}  
  
 value={teacher.learningRate}  
 onChange={  
 ({target}) =>  
 changeTeacher(  
 {  
 ...teacher,  
 learningRate: target.value  
 }  
 )  
 }  
  
 type={'number'}  
 />  
 <TextField  
 style={{marginTop: '16px'}}  
 label={'Эпсилон'}  
  
 value={teacher.epsilon}  
 onChange={  
 ({target}) =>  
 changeTeacher(  
 {  
 ...teacher,  
 epsilon: target.value  
 }  
 )  
 }  
  
 type={'number'}  
 />  
 <Typography textAlign={'center'} style={{marginTop: 8}}>  
 Количество итераций при обучении:  
 <br/>  
 {countOperations || 'Пока не обучалось'}  
 </Typography>  
 <Button  
 style={{marginTop: '16px'}}  
 variant={'contained'}  
 onClick={teach}  
 >  
 Обучить  
 </Button>  
 </div>  
 )  
}

import React, {*useEffect*, *useState*} from "react";  
import {Button, MenuItem, Select, Typography} from "@mui/material";  
import {$teacher, *getDiff*, *getS*} from "../../models/teacher";  
import {*useStore*} from "effector-react";  
import {*makeStyles*} from "@mui/styles";  
import {$weight} from "../../models/weight";  
import {$input} from "../../models/input";  
import {$M, setDiffs} from "../../models/presets";  
import {$recognize, 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 = () => {  
 setResult(  
 weights.map(  
 (w) => recognize.recognize([1, ...map], w)  
 )  
 )  
 setS(  
 weights.map(  
 w => *getS*([1, ...map], w)  
 )  
 )  
 setDiffs(  
 M  
 .map(  
 ({x}) => *getDiff*(x, map)  
 )  
 )  
 }  
  
 return (  
 <div className={classes.wrapper}>  
 <Typography variant={'h5'}>  
 {  
 (result)  
 ? `Результат: ${result.map(res => res > 0 ? 1 : -1)}`  
 : 'Сначала посчитай!'  
 }  
 </Typography>  
 <Typography variant={'h5'}>  
 {  
 (result)  
 ? `S: ${S.map(s => s.toFixed(4))}`  
 : '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>  
 )  
}