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 №4

тема: «Использование перцептрона.»

Выполнил: ст. группы МИВТ-221

Харитонов Сергей Дмитриевич

Белгород 2022 г.

**Лабораторная работа №4**

**Использование перцептрона.**

**Цель работы:** закрепление теоретических знаний, получение практических навыков использования перцептрона.

Экранные копии образов для обучения



Условия подбора весовых коэффициентов между S и A элементами.

В ходе экспериментов было принято решение использовать в качестве весов матрицы исходных изображений (1 – закрашенная клетка; 0 - пустая), для первой половины А нейронов первого изображения, для второй половины второго изображения, в качестве порога активации было принято число 6, а количество нейронов А слоя соответствует количеству входных нейронов.

В таком случае нейроны активируются для обучающей выборки, не пересекаясь и в равных количествах (массивы x).

Изображение выглядит как текст

Автоматически созданное описание

Веса между S и A слоем.

Для первой половины А нейронов:

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

Для второй половины А нейронов:

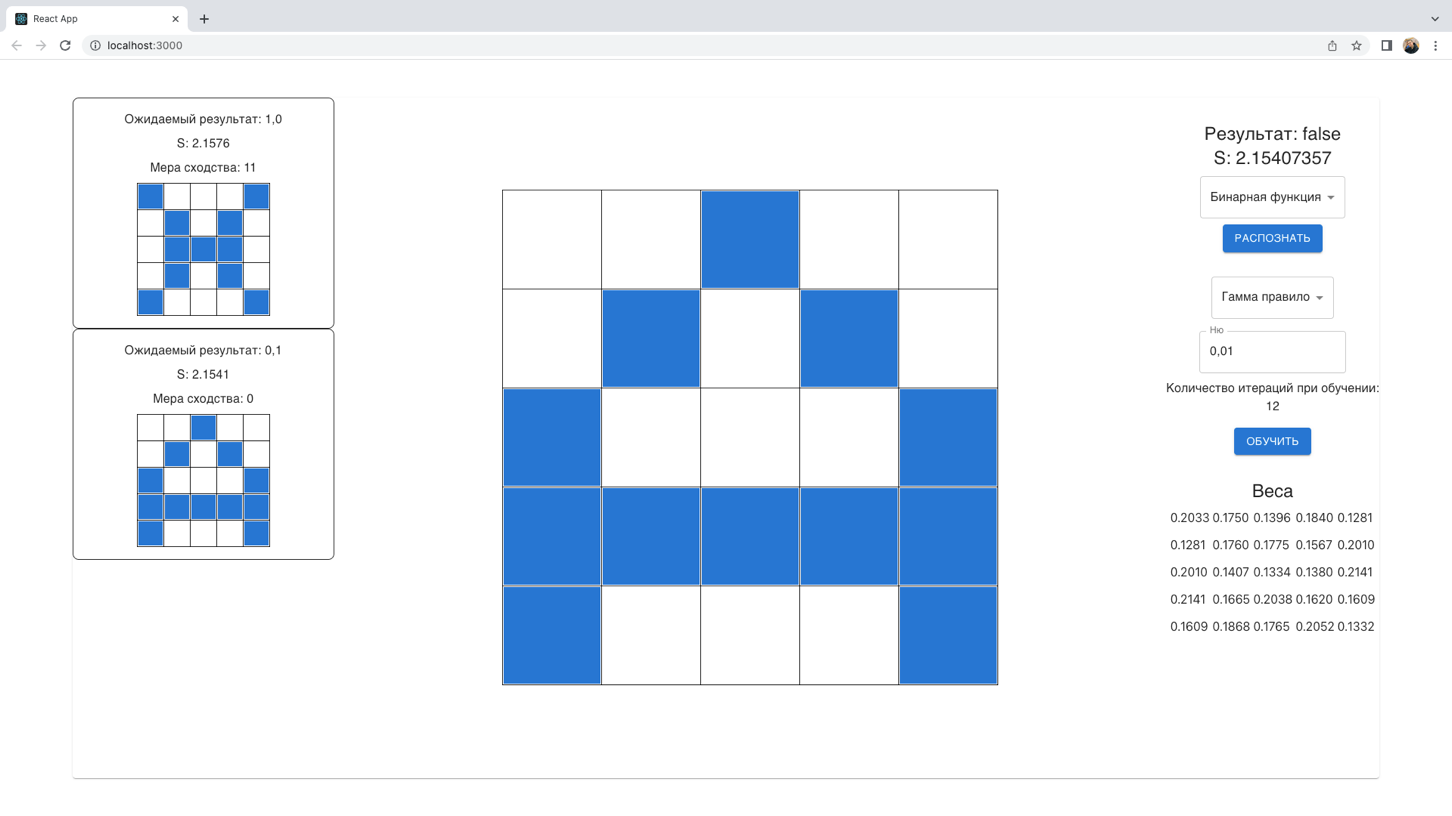
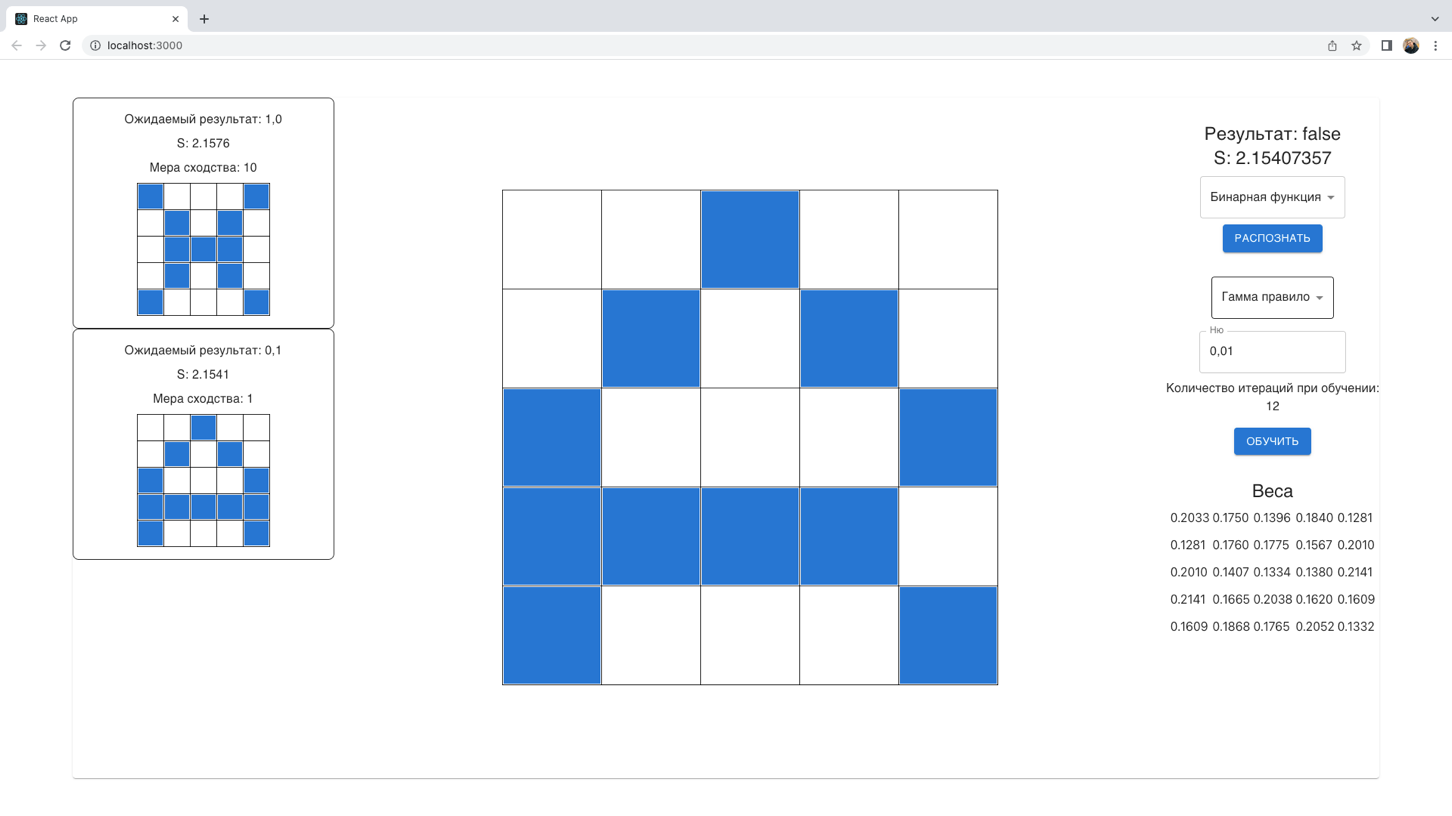
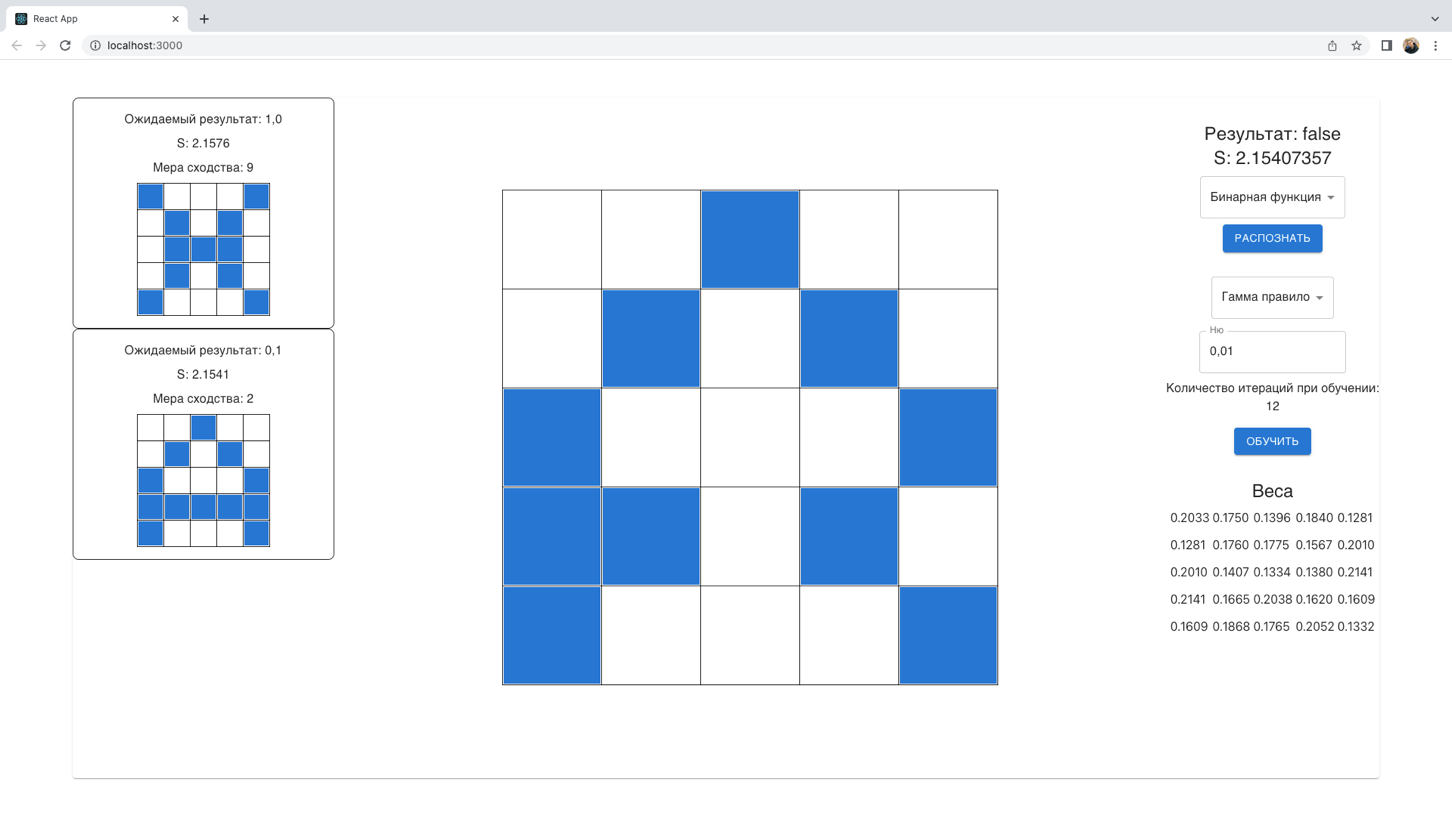
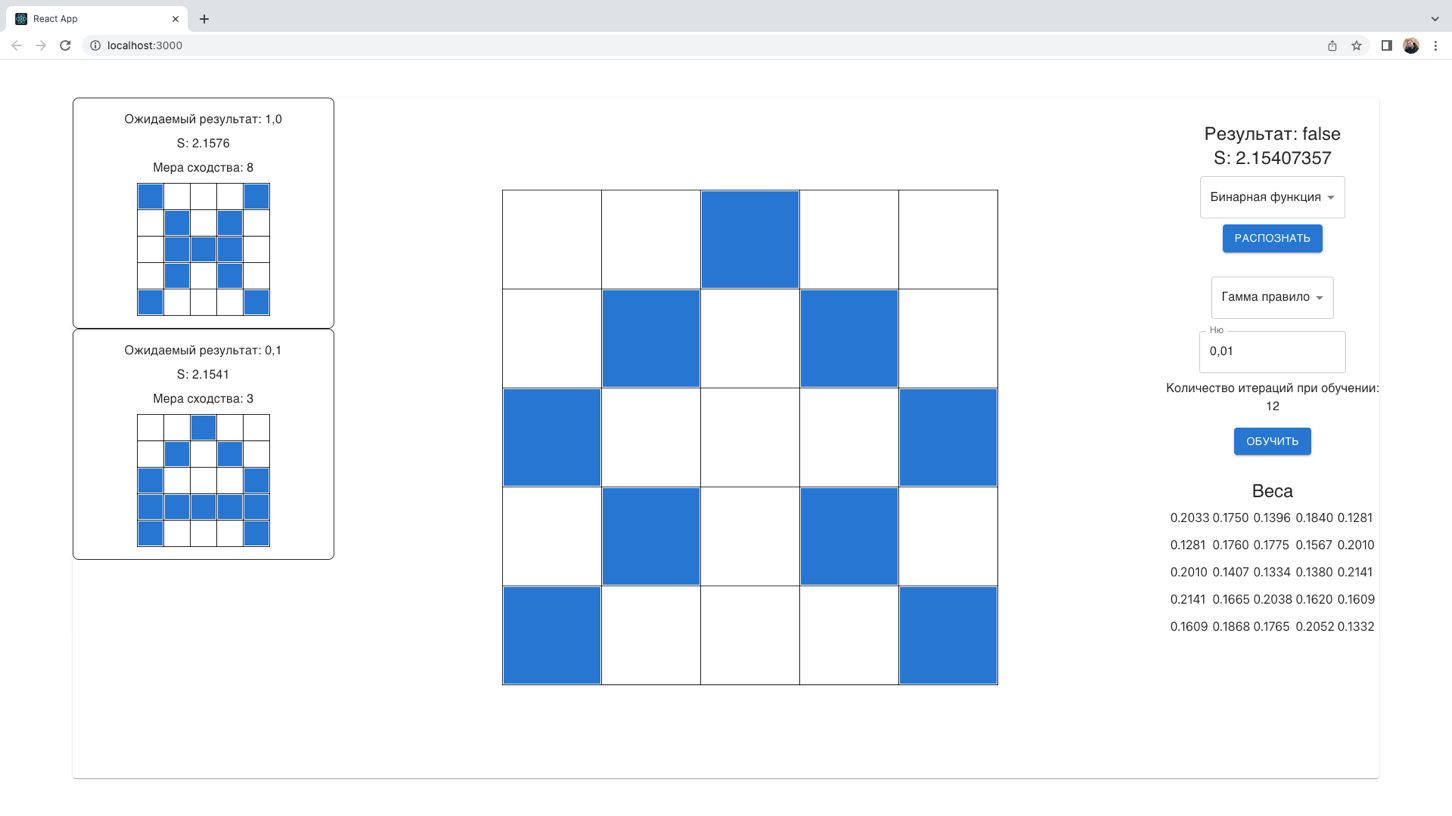
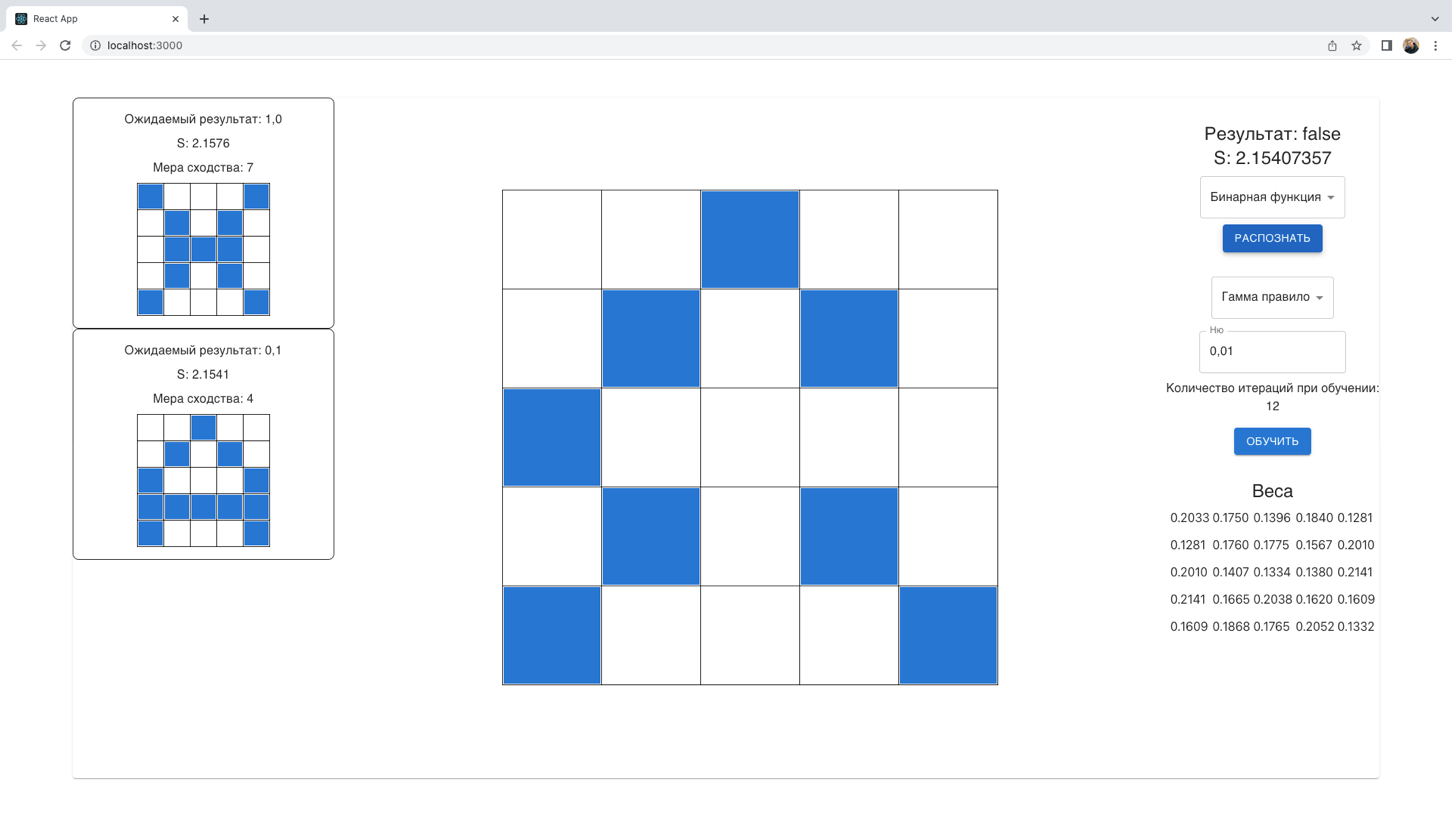
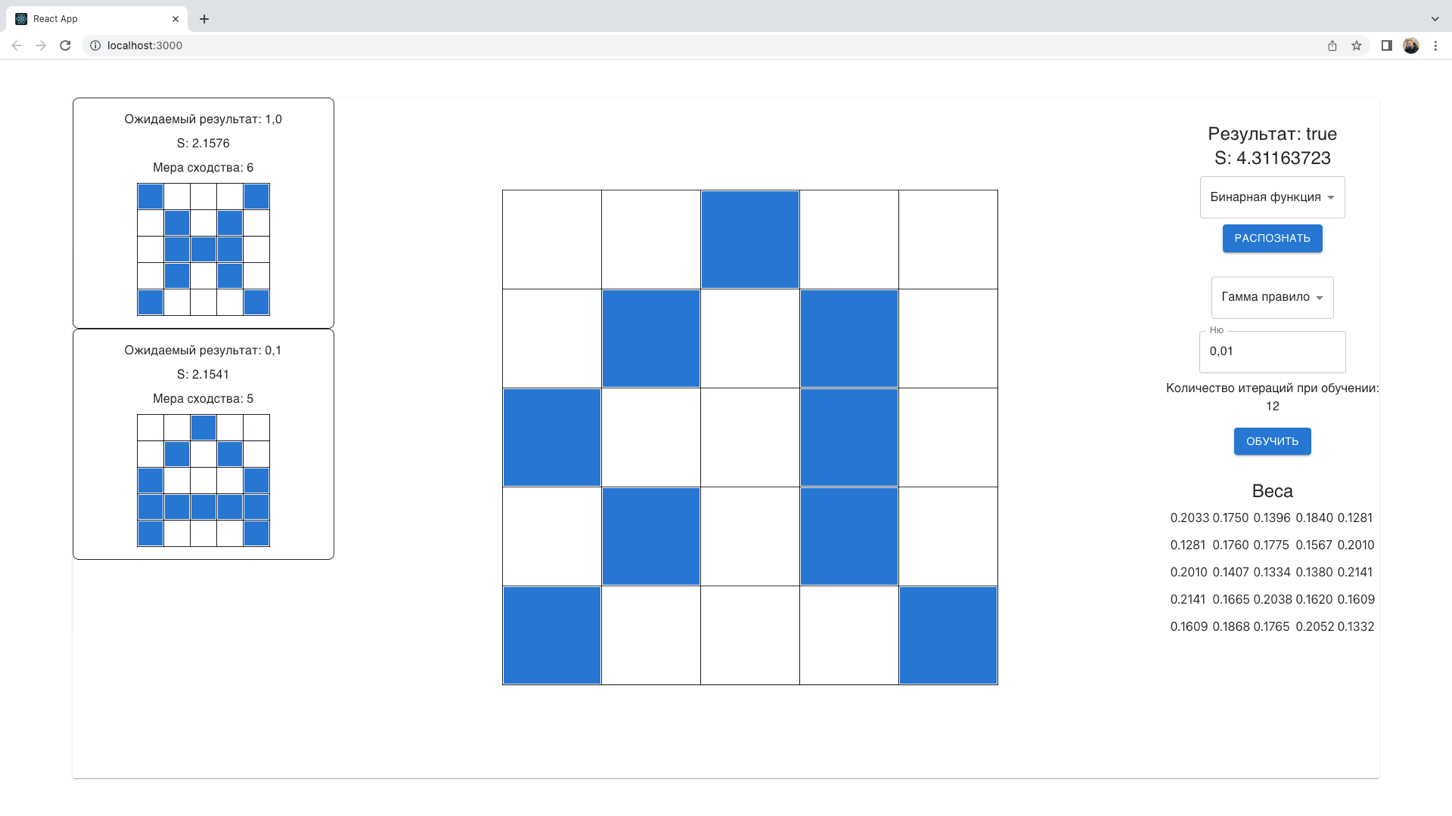
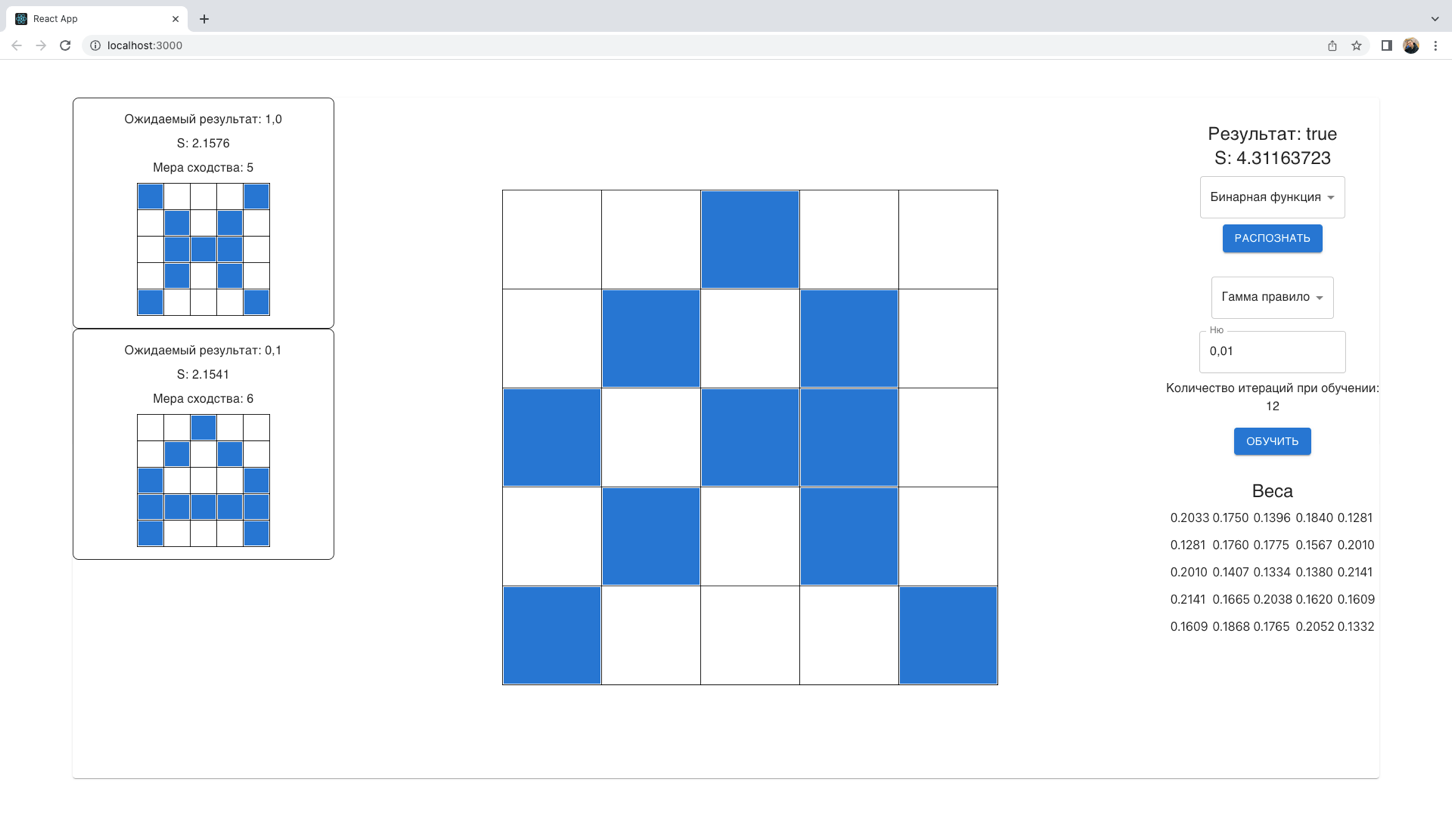
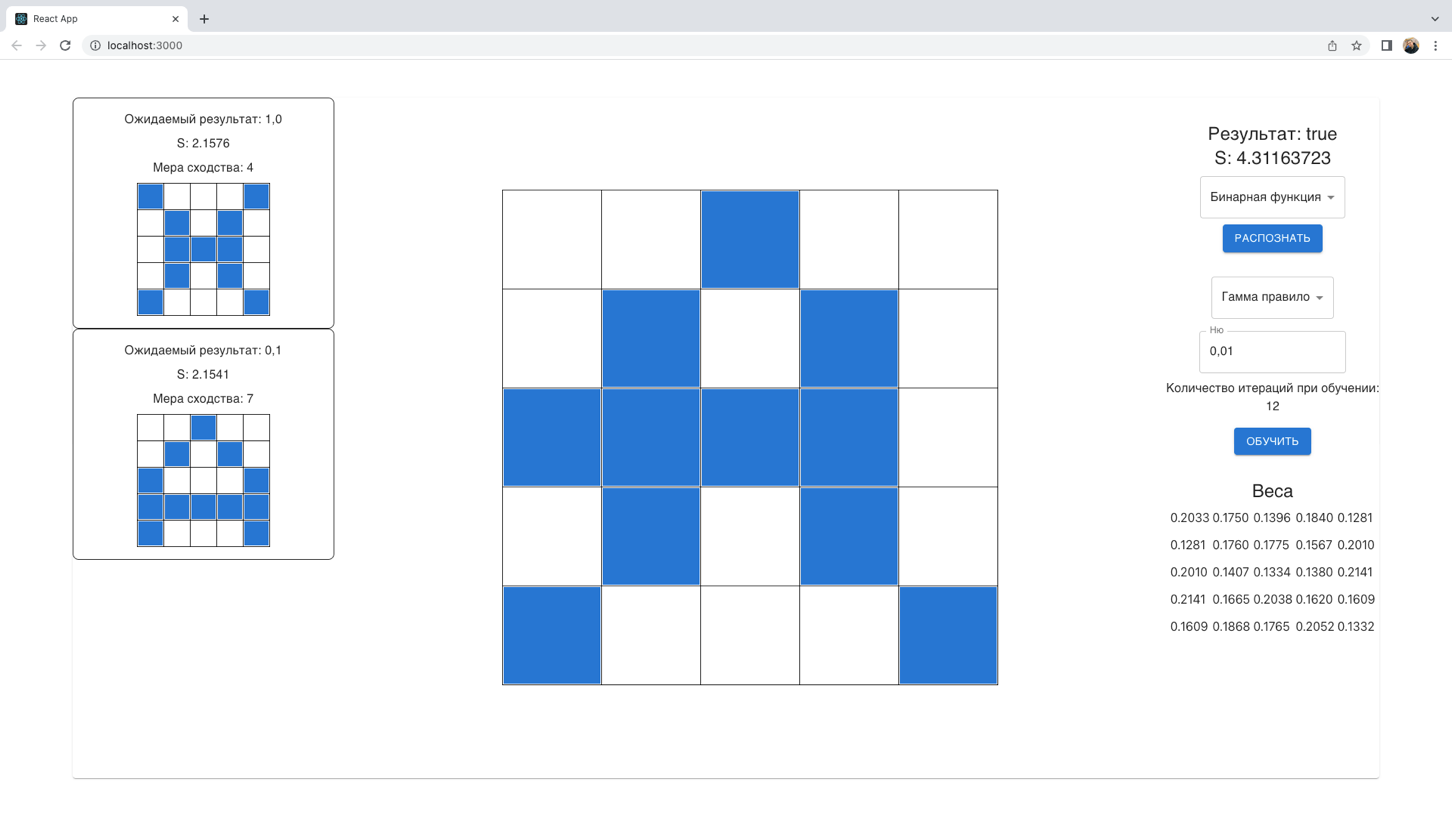
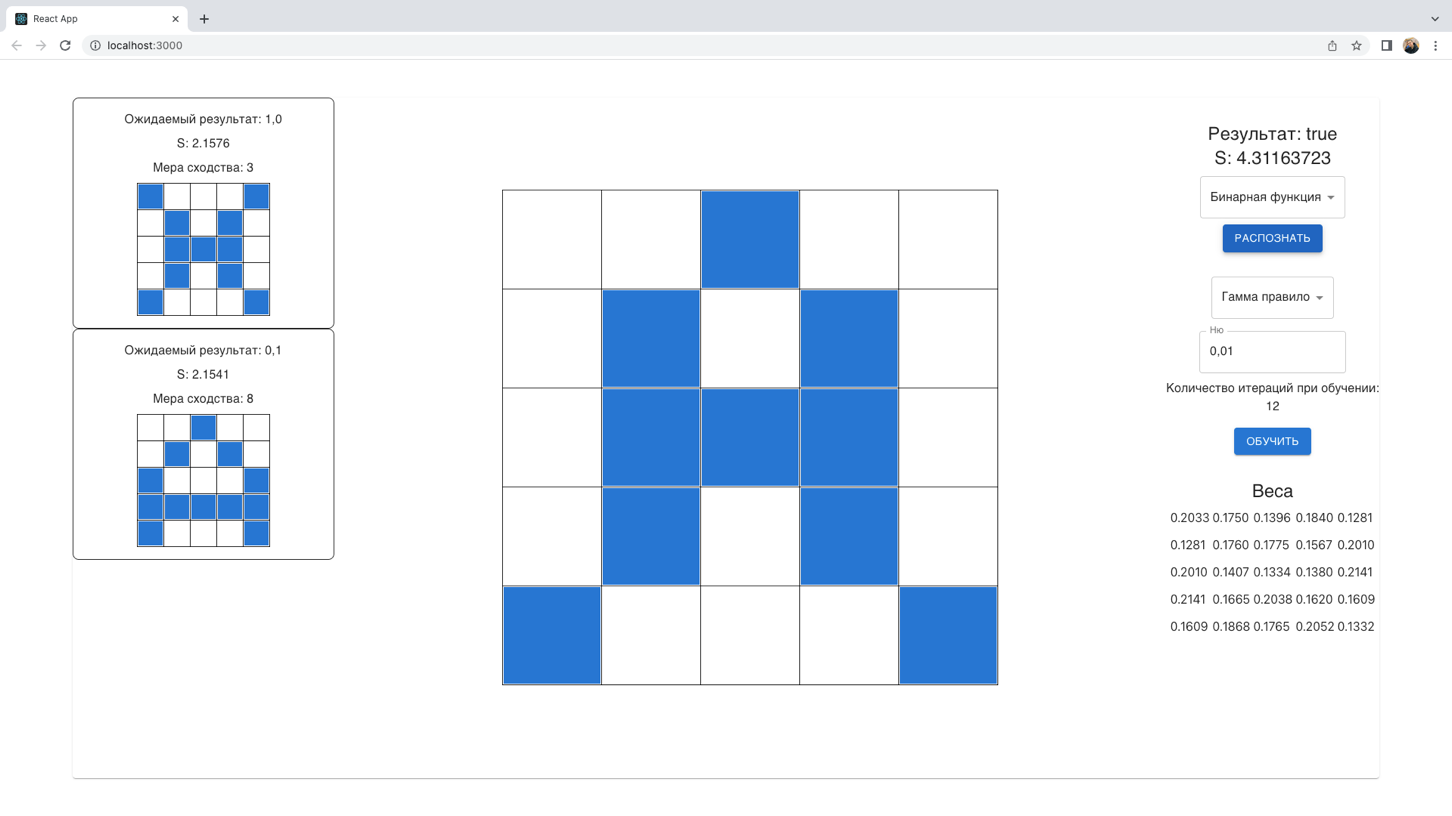
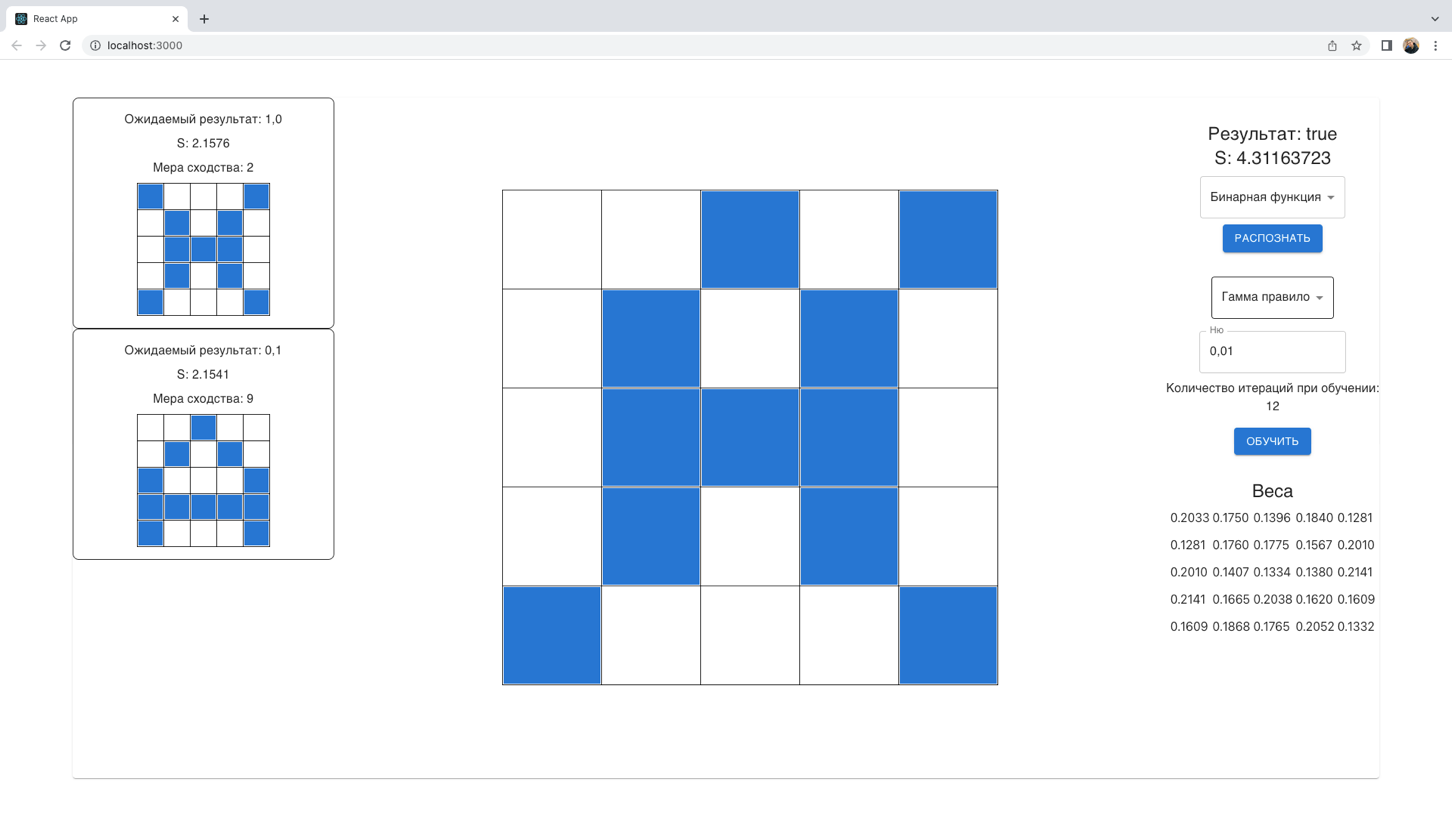
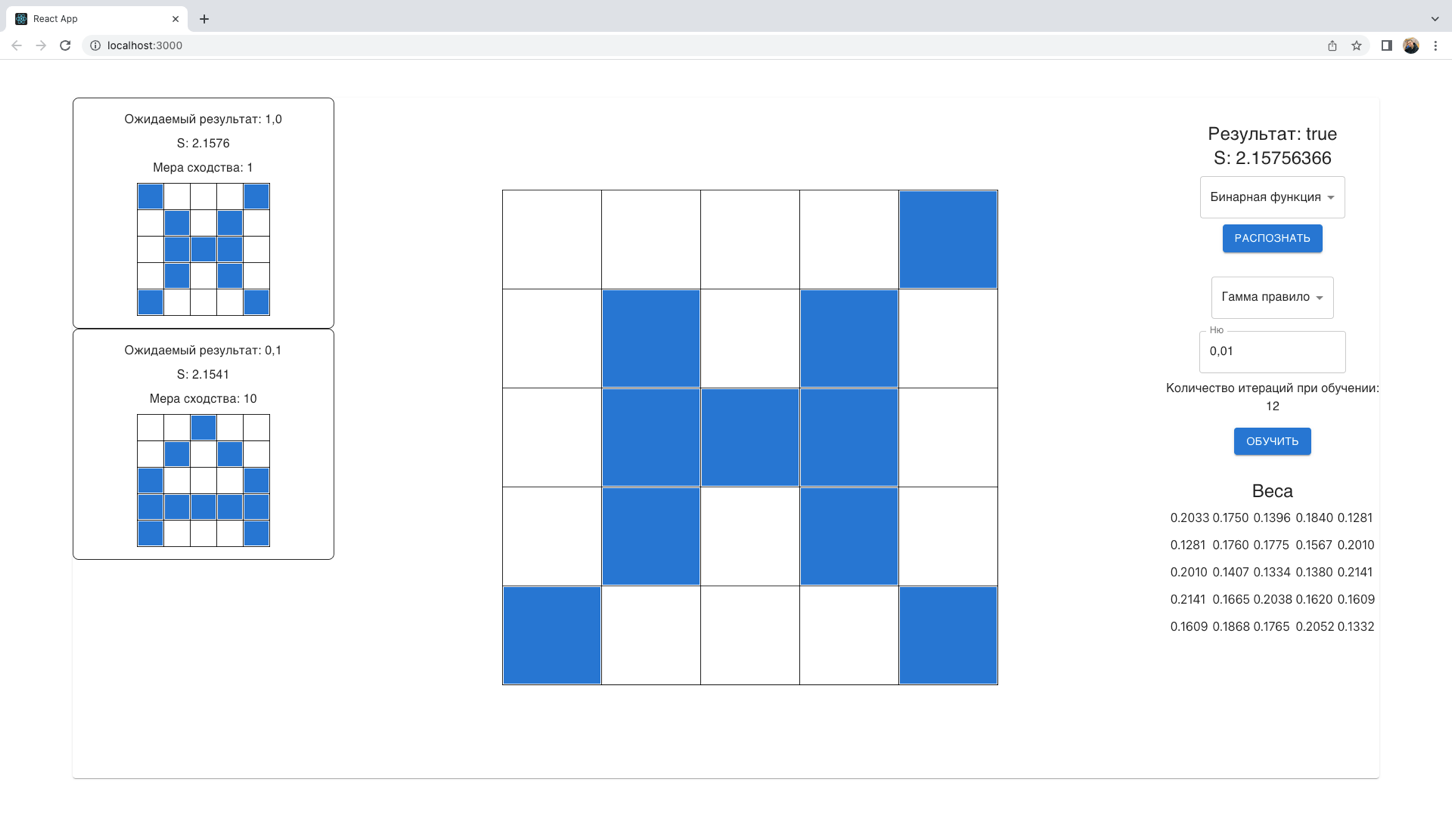
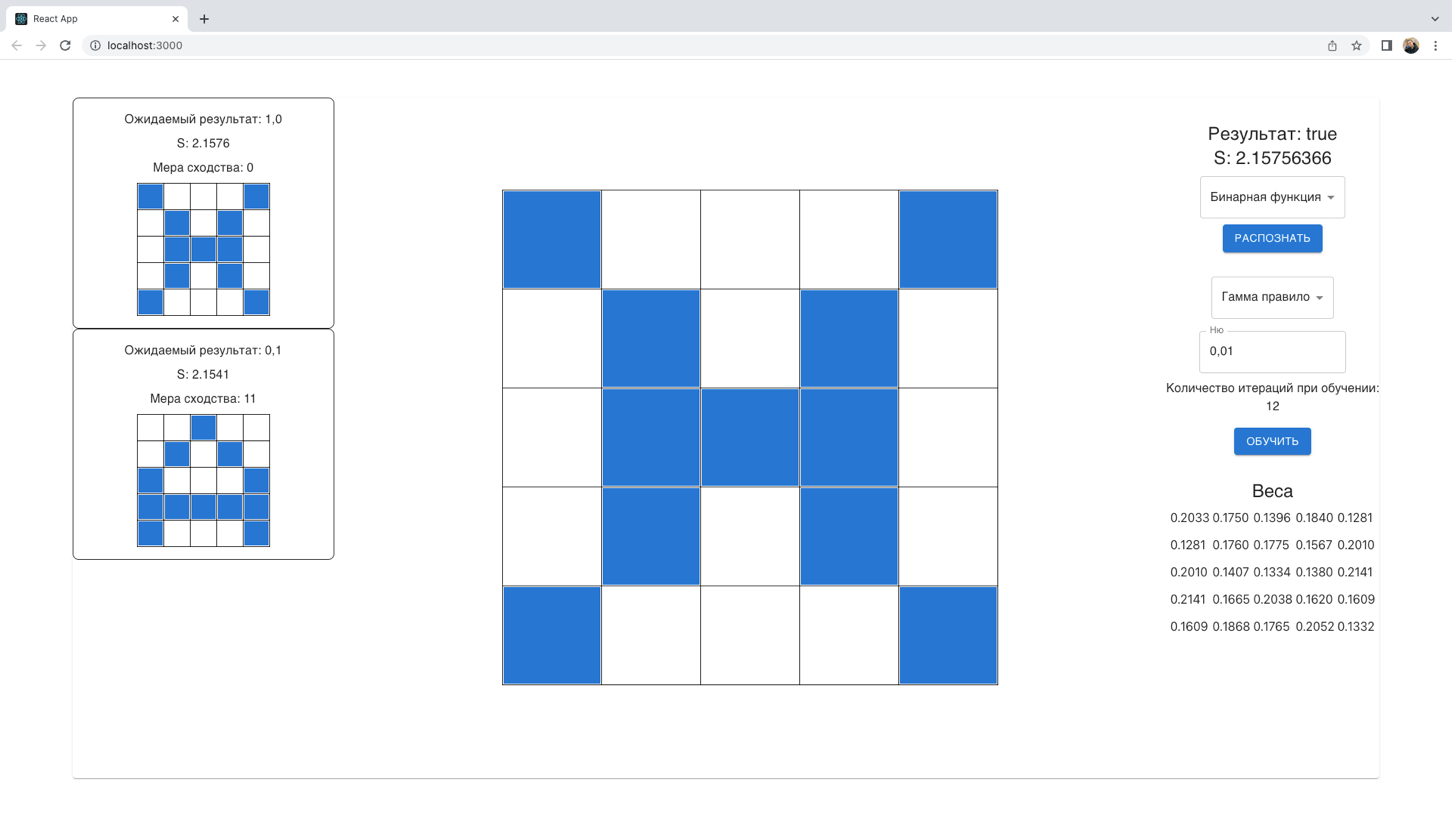
0 0 1 0 0 0 1 0 1 0 1 0 0 0 1 1 1 1 1 1 1 0 0 0 1

Результаты обучения

Изображение выглядит как стол

Автоматически созданное описание

Пошаговое исследование функций активации



Результаты исследования

|  |  |  |  |
| --- | --- | --- | --- |
| Мера сходства 1 | Мера сходства 2 | S | R |
| 0 | 11 | 2,15756 | 1 |
| 1 | 10 | 2,15756 | 1 |
| 2 | 9 | 4,31163 | 1 |
| 3 | 8 | 4,31163 | 1 |
| 4 | 7 | 4,31163 | 1 |
| 5 | 6 | 4,31163 | 1 |
| 6 | 5 | 4,31163 | 1 |
| 7 | 4 | 2,15407357 | 0 |
| 8 | 3 | 2,15407357 | 0 |
| 9 | 2 | 2,15407357 | 0 |
| 10 | 1 | 2,15407357 | 0 |
| 11 | 0 | 2,15407357 | 0 |

Графики полученные в ходе исследования

Как видно из графиков, при примерно равных мерах сходства S вырастает и выходной сигнал равен единице, а по мере приближения ко второму типу буквы он опять падает вплоть до границы, когда выходной сигнал обращается в ноль

Исследование возможности перцептрона по распознаванию нескольких классов изображений (использование нескольких выходных R-элементов)

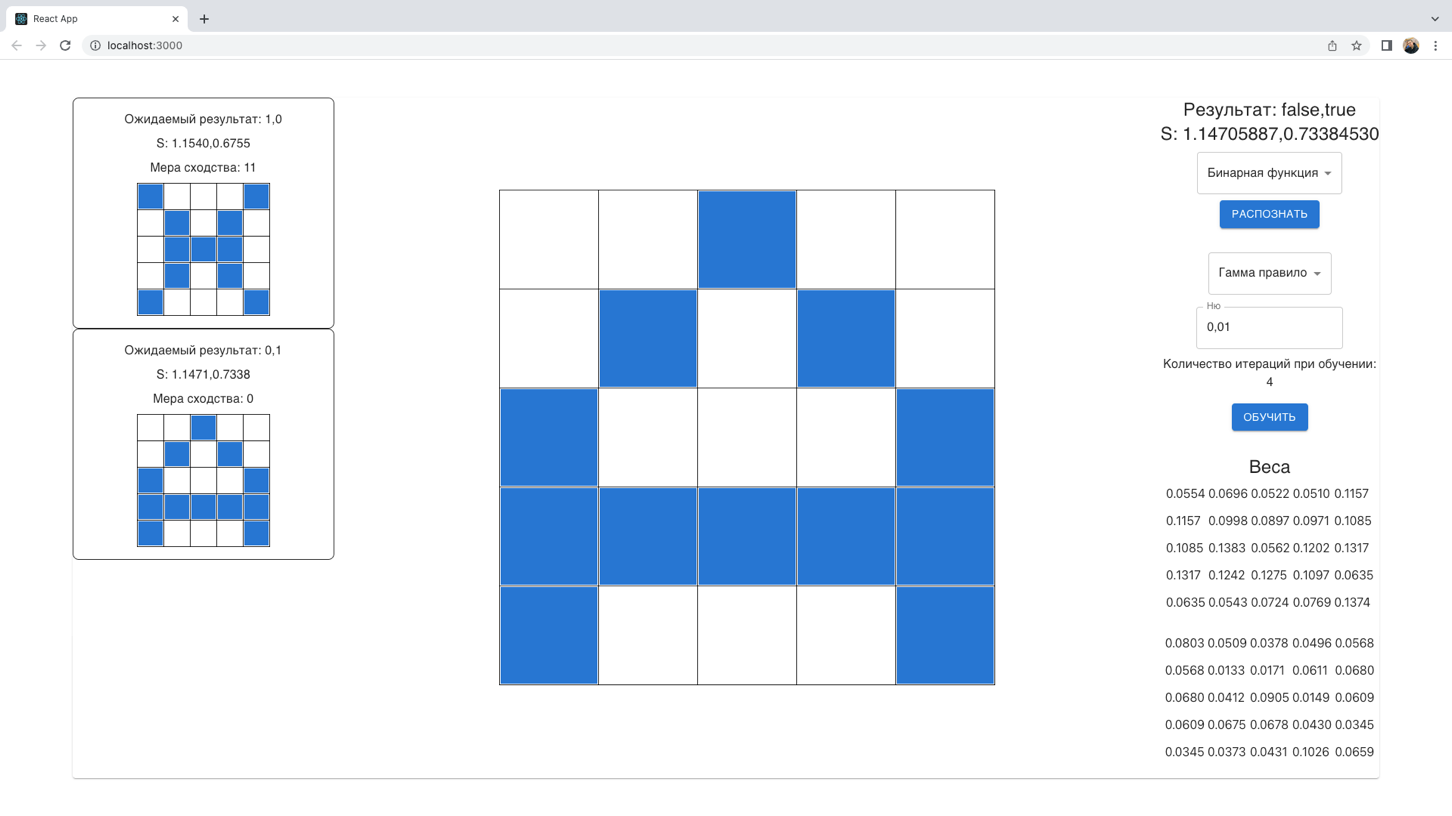
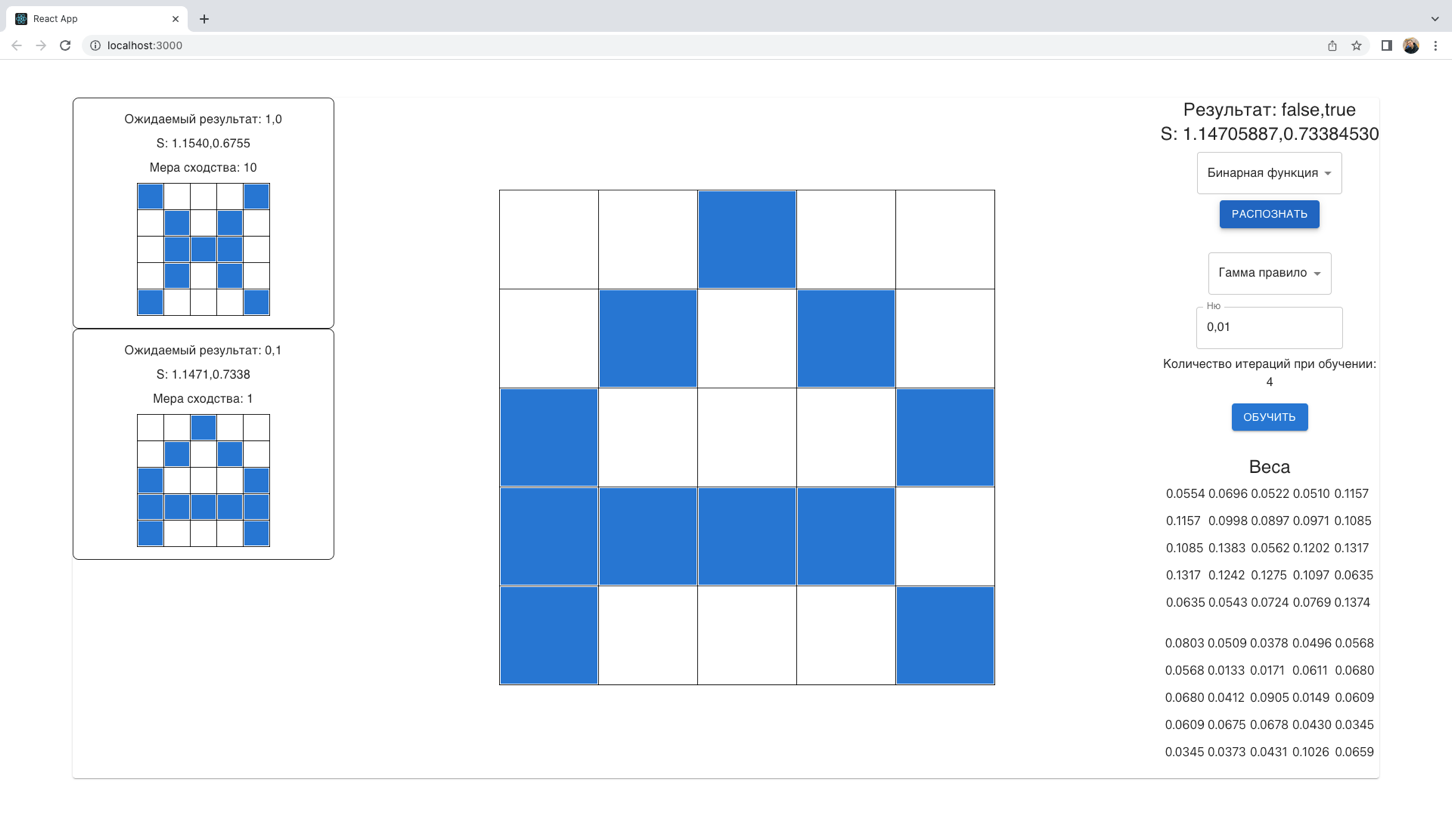
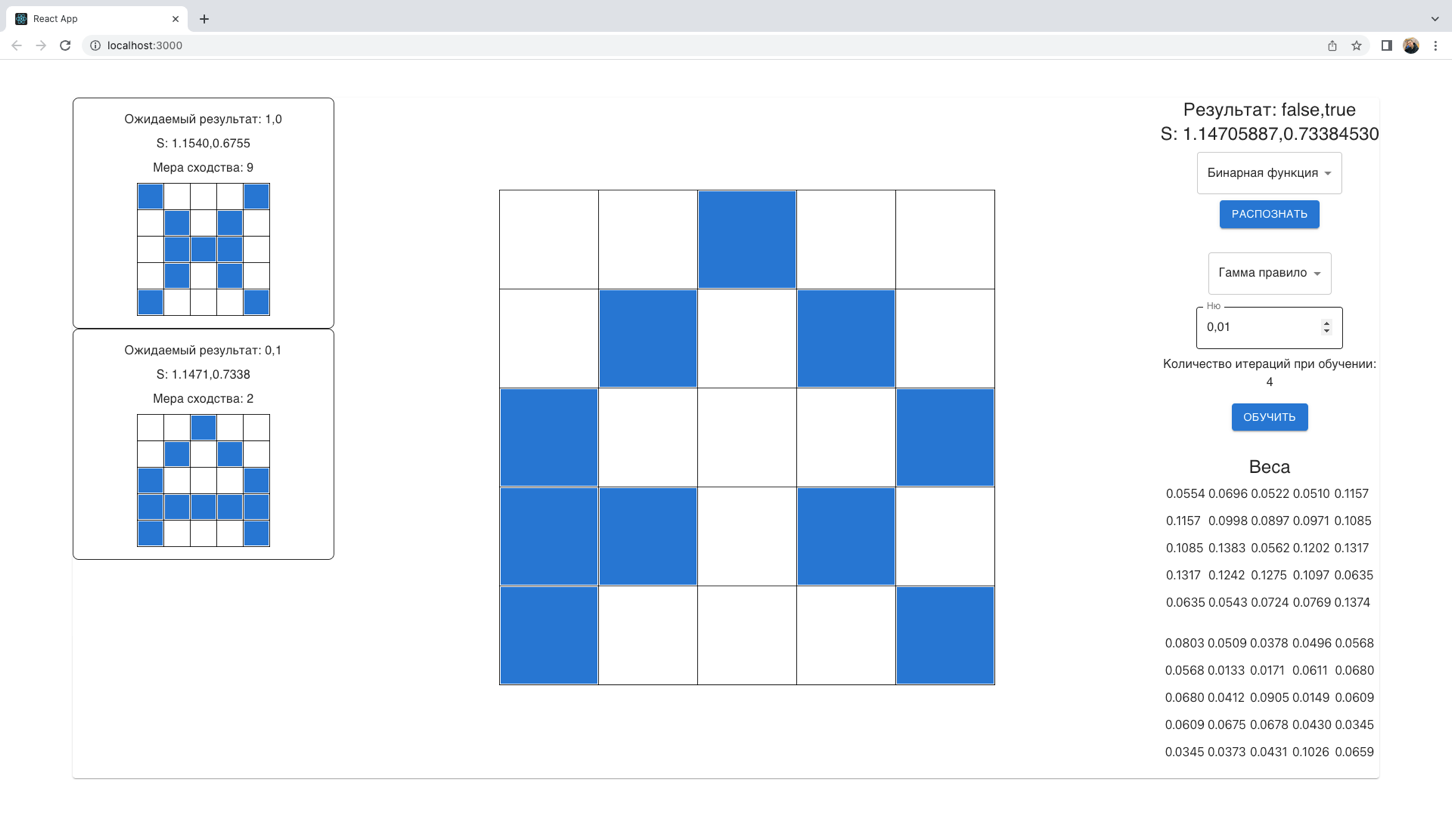
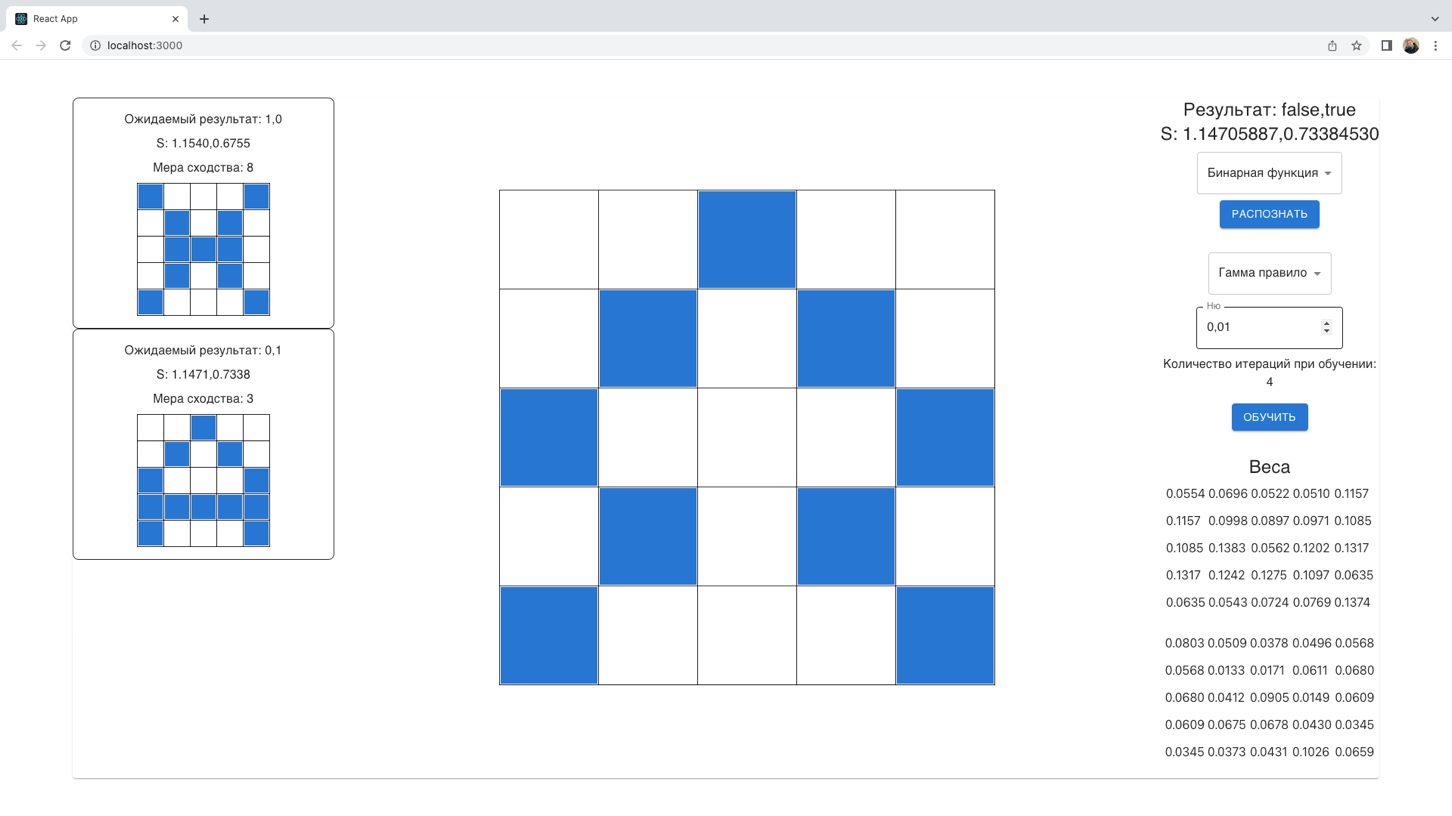
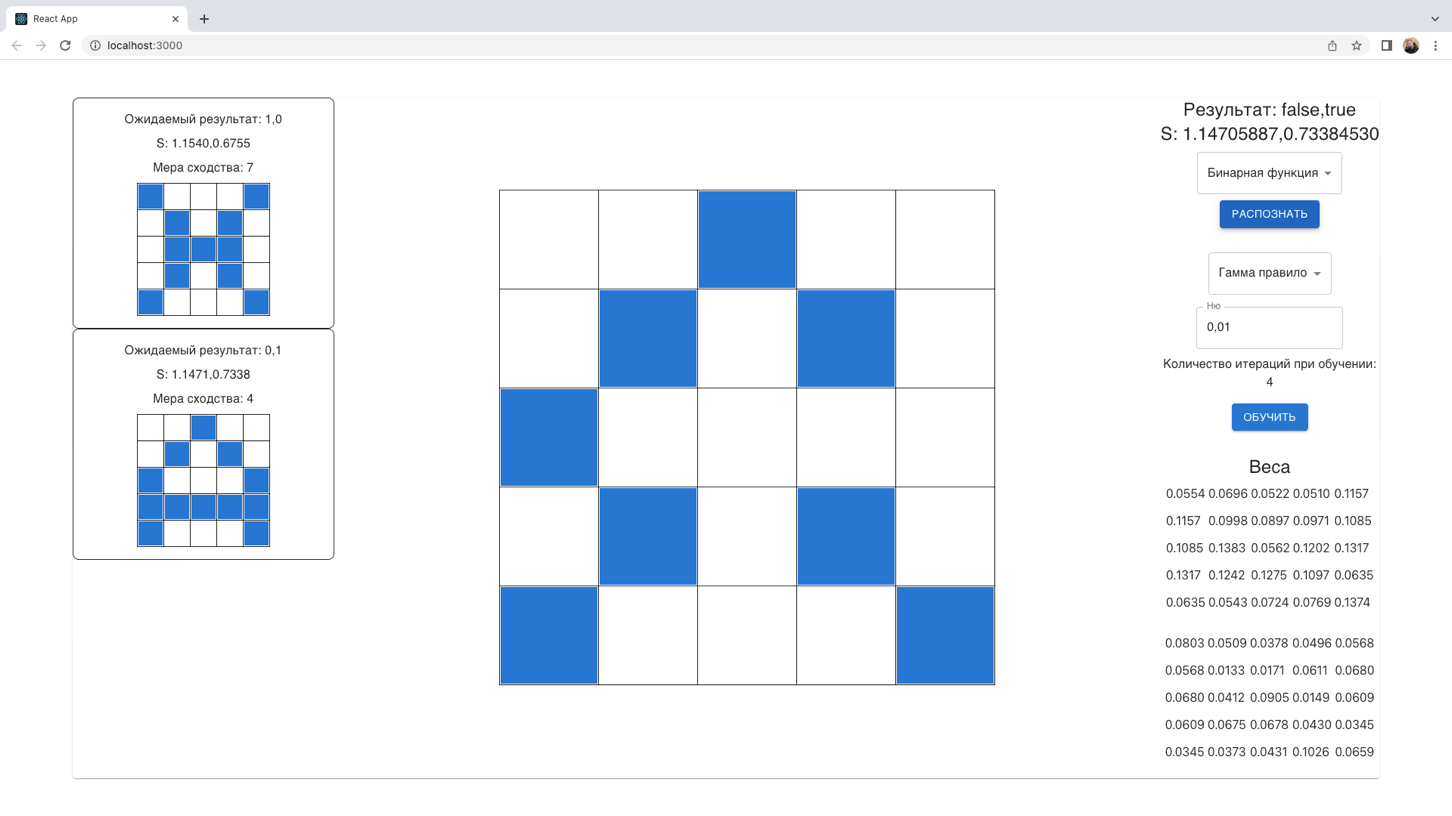
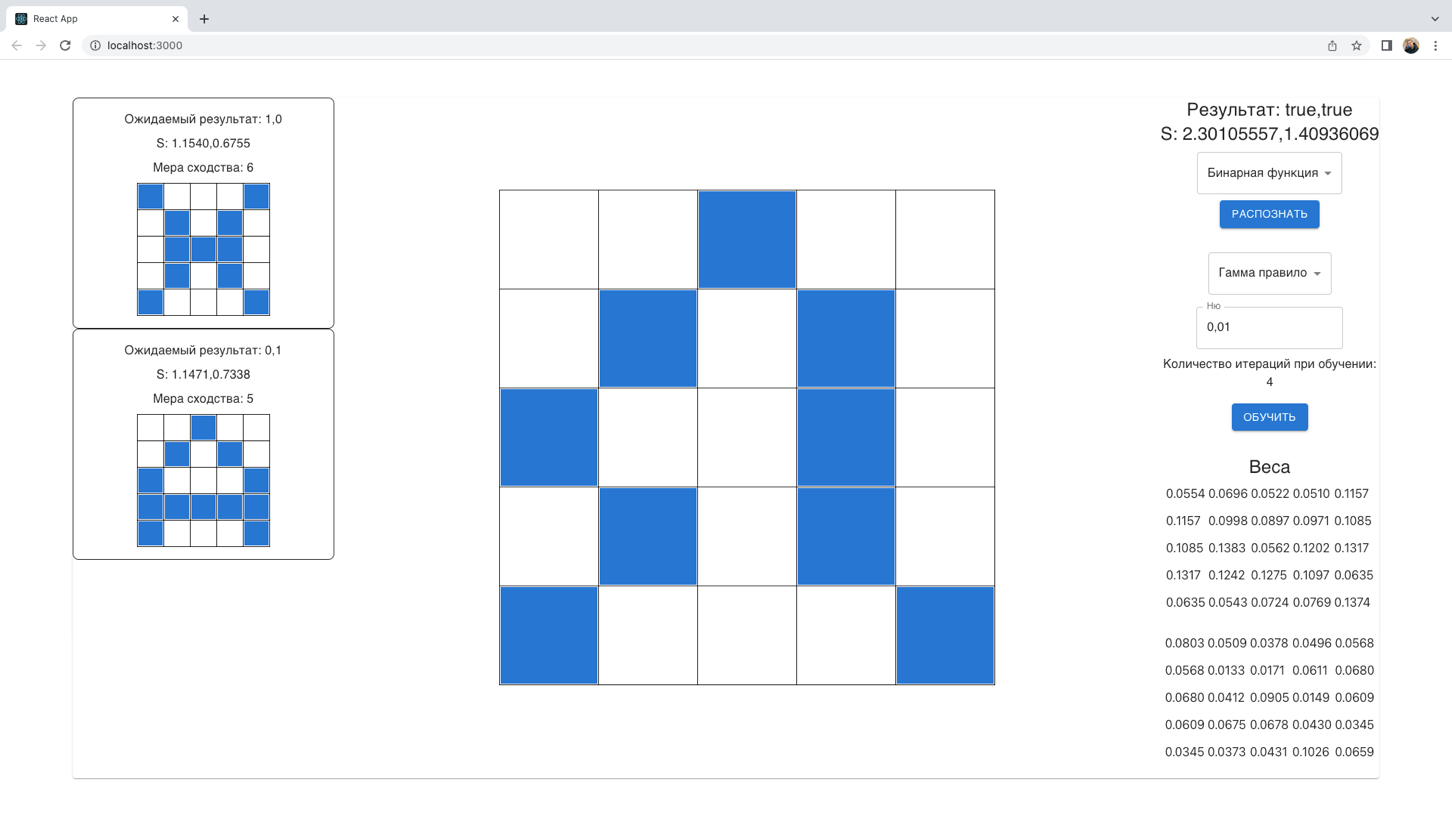
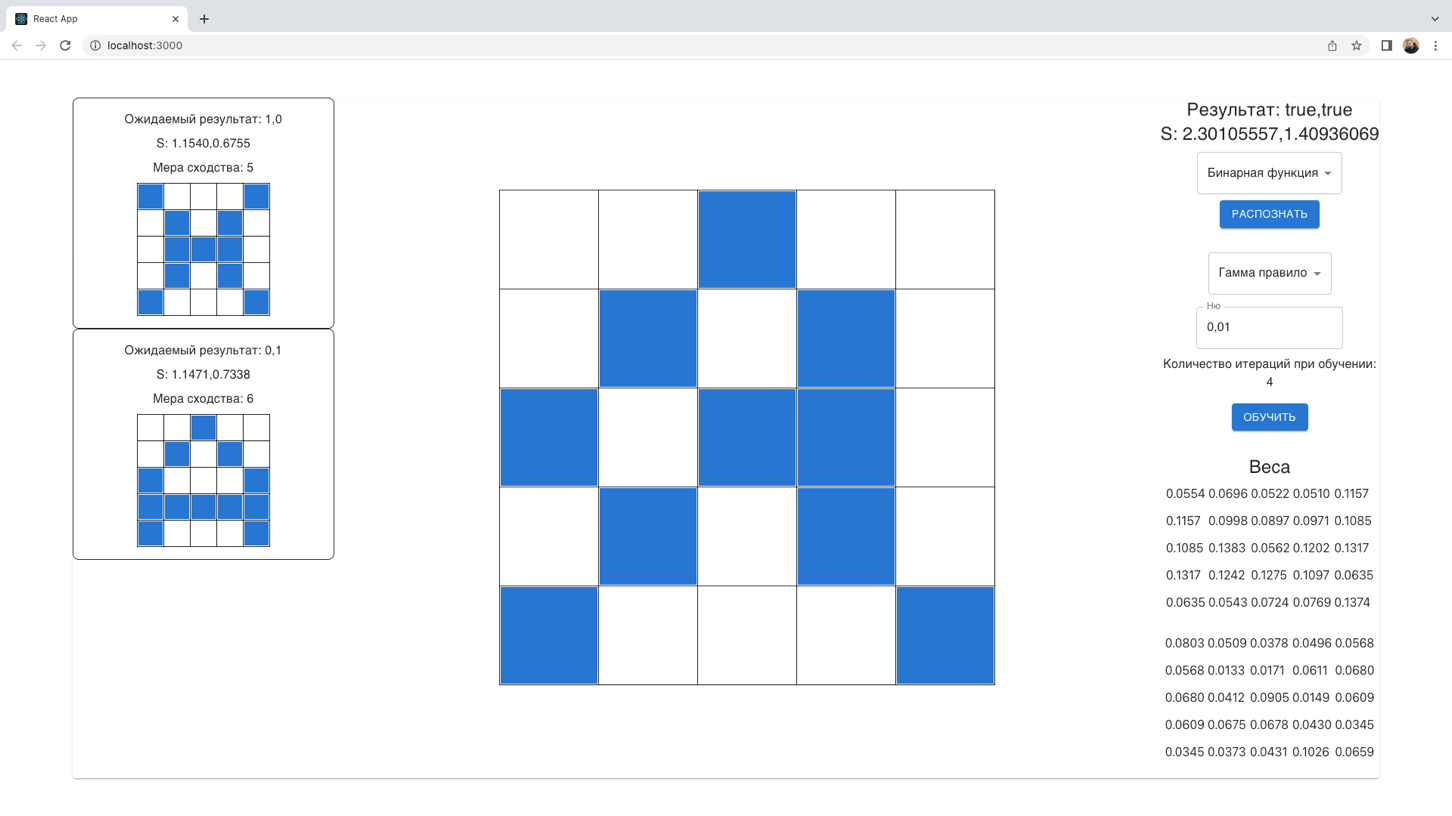
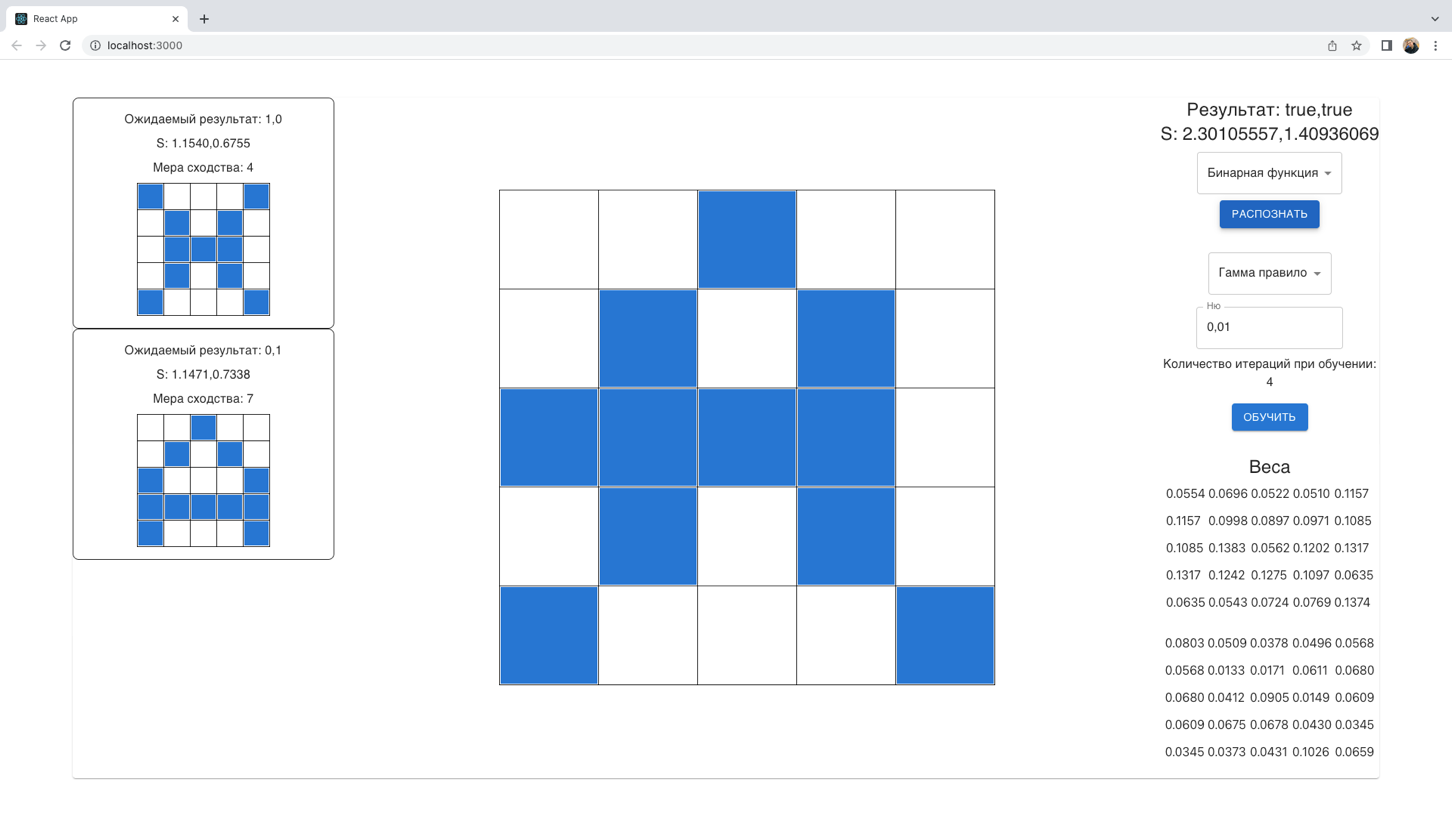
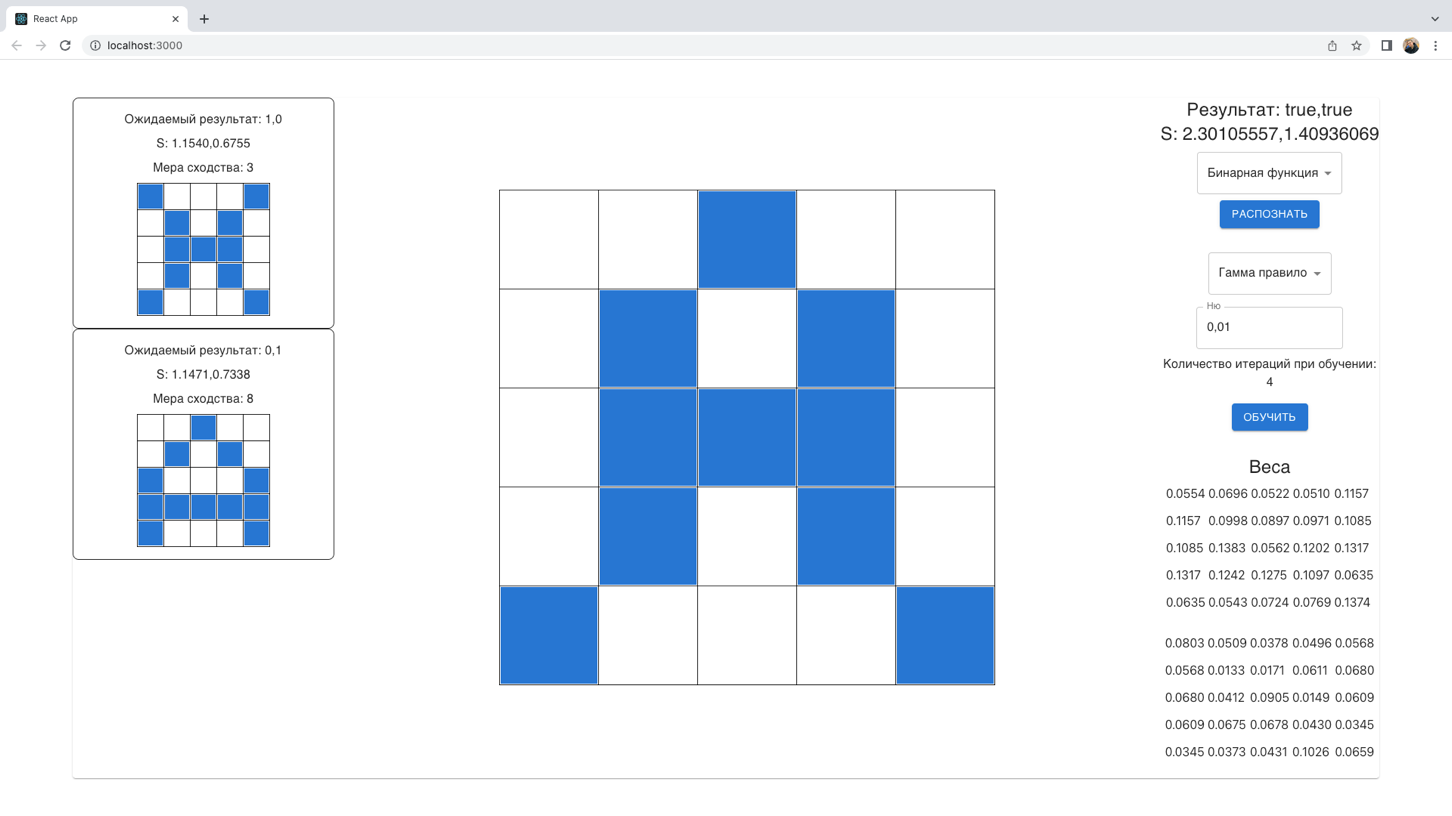
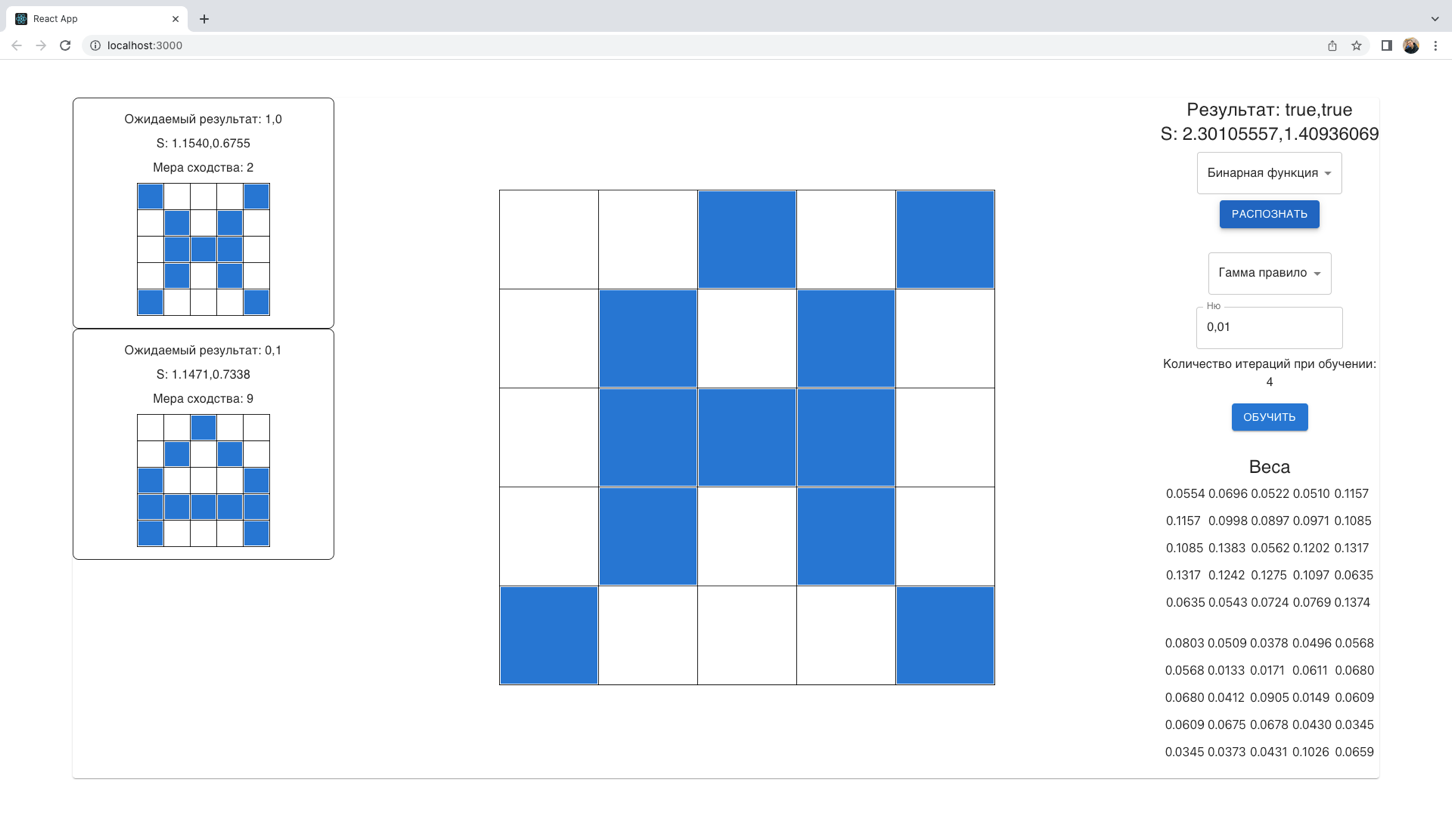
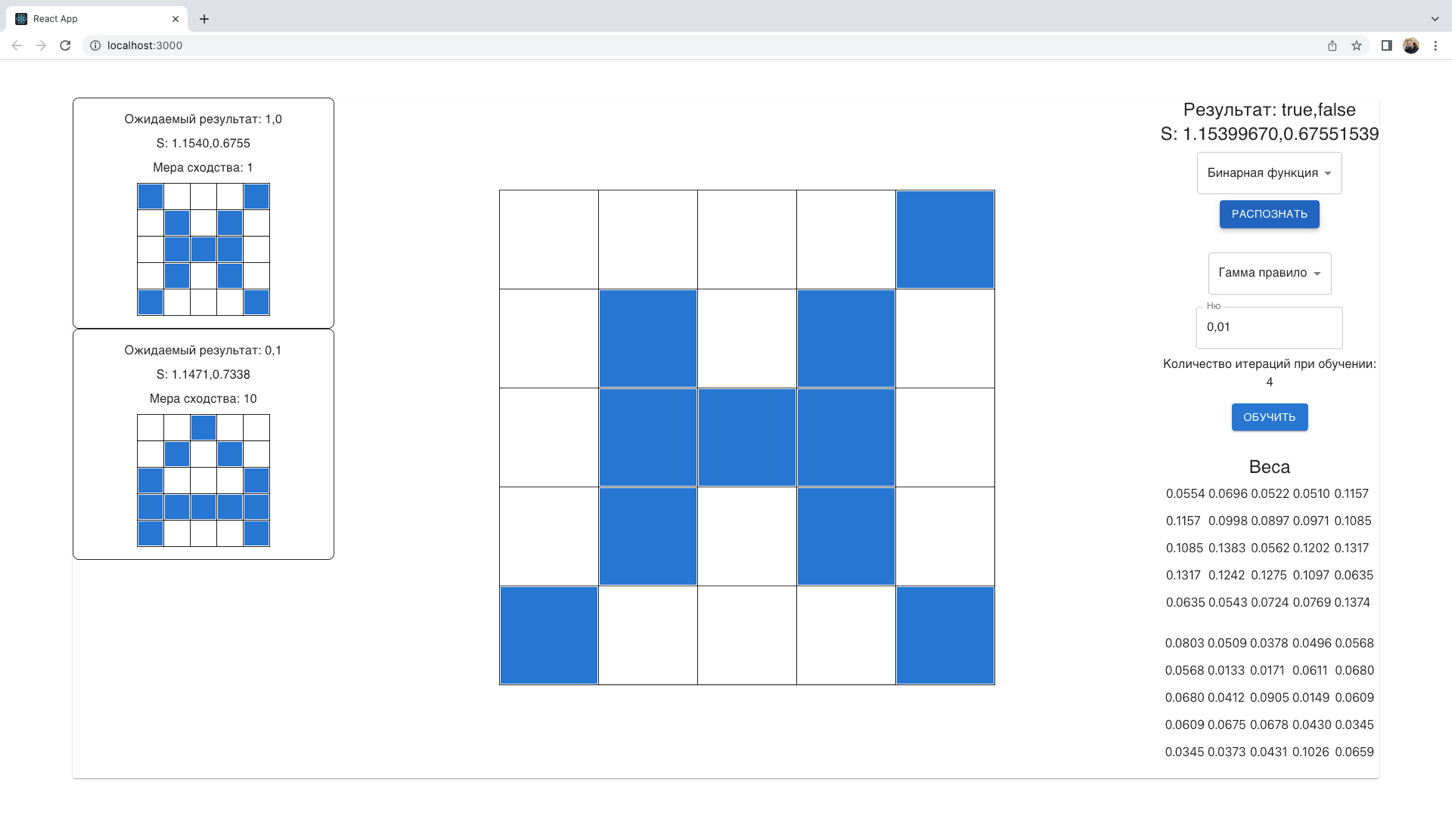
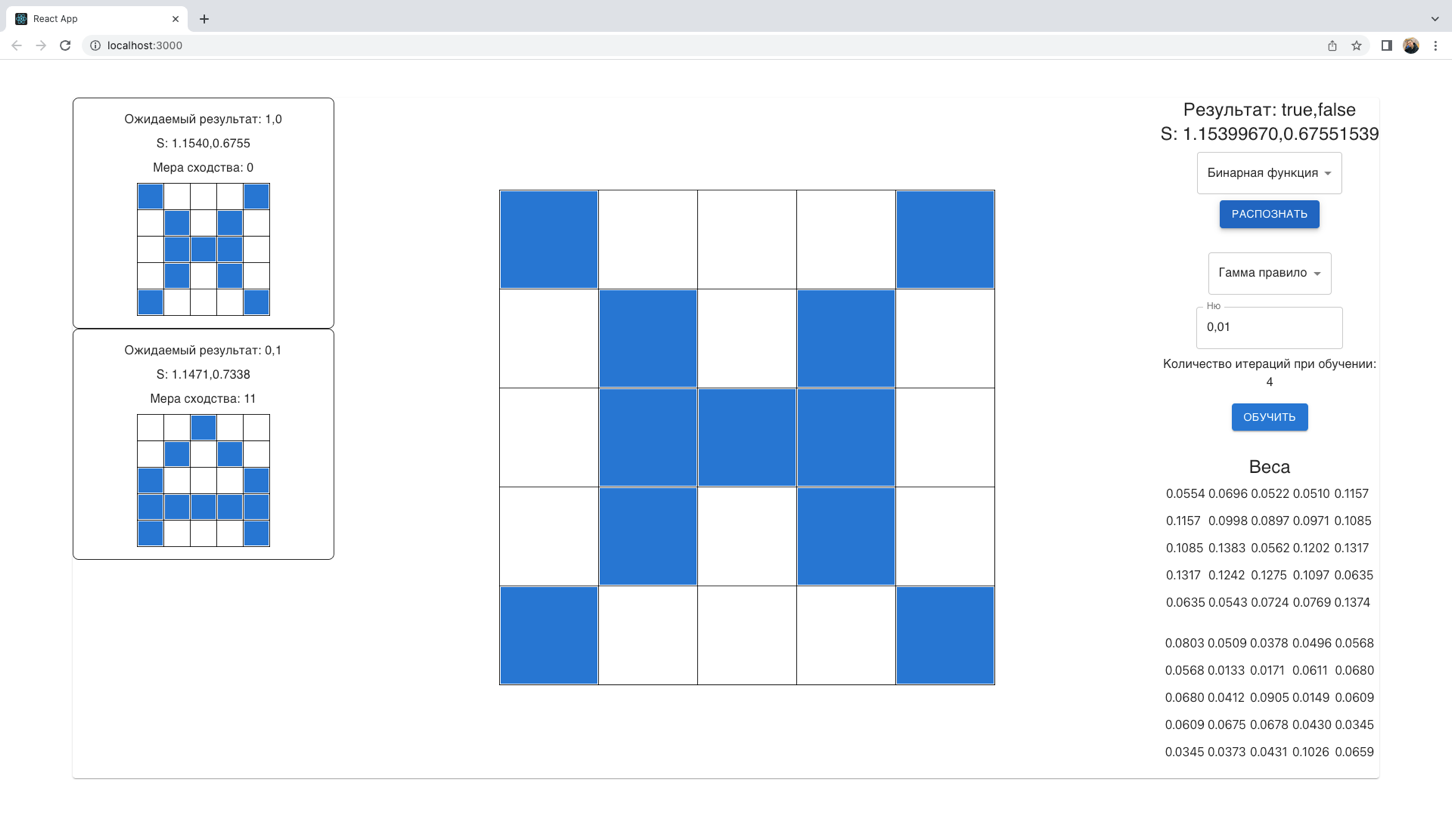
В ходе исследования возможности перцептрона по распознаванию нескольких классов (с несколькими выходными элементами) получилось обучить перцептрон с двумя выходными элементами, увеличение количества классов и выходных нейронов больше двух приводило к невозможности обучить перцептрон.

Результаты обучения для двух выходных нейронов

Изображение выглядит как стол

Автоматически созданное описание

Пошаговое исследование функций активации



Результаты исследования

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Мера сходства 1 | Мера сходства 2 | S1 | S2 | Result1 | Result2 |
| 0 | 11 | 1,5399 | 0,6755 | 1 | 0 |
| 1 | 10 | 1,5399 | 0,6755 | 1 | 0 |
| 2 | 9 | 2,301 | 1,4093 | 1 | 1 |
| 3 | 8 | 2,301 | 1,4093 | 1 | 1 |
| 4 | 7 | 2,301 | 1,4093 | 1 | 1 |
| 5 | 6 | 2,301 | 1,4093 | 1 | 1 |
| 6 | 5 | 2,301 | 1,4093 | 1 | 1 |
| 7 | 4 | 1,14705 | 0,733845 | 0 | 1 |
| 8 | 3 | 1,14705 | 0,733845 | 0 | 1 |
| 9 | 2 | 1,14705 | 0,733845 | 0 | 1 |
| 10 | 1 | 1,14705 | 0,733845 | 0 | 1 |
| 11 | 0 | 1,14705 | 0,733845 | 0 | 1 |

Графики полученные в ходе исследования

Ситуация аналогична с одним выходным R нейроном для обоих выходных нейронов. Первый нейрон активируется в тех же случаях что и с одним выходным нейроном, а второй противоположно в соответствии с ожидаемым результатом.

Листинг программы

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 gammaTeacher = ({  
 id: 1,  
 getNextWeight: (old, nu, a, index, delta) => {  
 let deltaW = 0  
 const Sa = Math.abs(delta)\*nu  
 if (a){  
 if ((old >= 1) || (old <= 0))  
 deltaW = 0  
 else if ((old + nu) > 1)  
 deltaW = 1 - old  
 else if ((old - nu) < 0)  
 deltaW = 0 - old  
 else  
 deltaW = nu - Sa / delta  
 }  
 else  
 deltaW = - Sa/delta  
  
 return old - deltaW  
 },  
 activation: (xs, w, S = 0) =>  
 *getS*(xs, w) > S  
 ? 1  
 : 0,  
 name: 'Гамма правило',  
 nu: 0.01  
})  
  
export const teachers = [gammaTeacher]  
  
export const $teacher = *createStore*(gammaTeacher)  
  
export const changeTeacher = *createEvent*()  
console.log($M)  
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 {*getA*, setWeights} from "../weight";  
import {setSs} from "../presets";  
import {setCountOperations} from "../info";  
import {*average*} from "../recognize";  
  
const handleChangeTeacher = (\_, teacher) => teacher  
const handleTeach = (teacher, {M, weight}) => {  
 let W = weight.map(w => [...w])  
  
 const A = M.map(({x, ...m}, index) => {  
  
 return ({  
 ...m,  
 x: *getA*(x)  
 })  
 }  
 )  
  
 const checkNotEnd = () => W  
 .some(  
 (w, indexW) => {  
 const Ss = A.map(a => *getS*(a.x, w))  
 const {  
 Sone, Sother  
 } = Ss.reduce(  
 (result, s, index) => A[index].t[indexW]  
 ? ({  
 ...result,  
 Sone: result.Sone + s,  
 })  
 : ({  
 ...result,  
 Sother: result.Sother + s,  
 }),  
 {Sone: 0, Sother: 0}  
 )  
 return A.some(  
 (a, indexM) => {  
 const t = teacher.activation(a.x, w, *average*([Sone, Sother])) !== a.t[indexW]  
 return t  
 }  
 )  
 }  
 )  
  
 let countOperations = 0;  
 while (checkNotEnd() && countOperations < 100) {  
 ++countOperations;  
 A  
 .forEach(  
 a => {  
 W = W.map(  
 (wRow, indexWRow) => {  
 const Nak = a.x.filter(x => x === 1);  
 const Nagr = a.x.filter((x, indexW) => (x === 1) && (wRow[indexW] >= 1) && (wRow[indexW] <= 0));  
 const Nabgr = a.x.filter((x, indexW) => (x === 1) && (wRow[indexW] >= 0.999) && (wRow[indexW] <= 0.001))  
 const delta = Nak.length - Nagr.length - Nabgr.length  
  
 const Ss = A.map(a => *getS*(a.x, wRow))  
 const {  
 Sone, Sother  
 } = Ss.reduce(  
 (result, s, index) => A[index].t[indexWRow]  
 ? ({  
 ...result,  
 Sone: result.Sone + s,  
 })  
 : ({  
 ...result,  
 Sother: result.Sother + s,  
 }),  
 {Sone: 0, Sother: 0}  
 )  
  
 const r = teacher.activation(a.x, wRow, *average*([Sone, Sother]))  
  
 if (r !== a.t[indexWRow])  
  
 return wRow  
 .map(  
 (oldW, index) => {  
 const newW = teacher.getNextWeight(  
 oldW,  
 teacher.nu,  
 a.x[index],  
 index,  
 delta  
 )  
 return newW  
 }  
 )  
 return wRow  
  
 }  
 )  
 }  
 )  
 }  
 setWeights(W)  
 setSs(  
 A  
 .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,  
].map(() => (Math.random() / 10.))  
  
export const *getS* = (Xs, Ws) =>  
 Xs  
 .reduce(  
 (result, x, index) => result + x \* Ws[index],  
 0  
 )  
  
const Aw = [  
 {  
 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],  
 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: [0, 1],  
 S: [],  
 diff: 0  
 },  
].map(({x}) => x)  
export const *getA* = (xs) => xs.map((x, indexNewA) => {  
 const indexAw = Math.trunc(indexNewA / 13);  
  
 const S = *getS*(xs, Aw[indexAw])  
 return S > 6 ? 1 : 0  
  
})  
  
export const zeroWeight = [1, 2].map(getRandowW)  
export const $weight = *createStore*(zeroWeight)  
  
export const setWeights = *createEvent*()

import {  
 $weight,  
 setWeights  
} from "./index";  
  
  
const handleSetWeights = (\_, weights) => weights  
  
$weight  
 .on(setWeights, handleSetWeights)

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,  
].map(() => (Math.random() / 10.))  
  
export const *getS* = (Xs, Ws) =>  
 Xs  
 .reduce(  
 (result, x, index) => result + x \* Ws[index],  
 0  
 )  
  
const Aw = [  
 {  
 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],  
 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: [0, 1],  
 S: [],  
 diff: 0  
 },  
].map(({x}) => x)  
export const *getA* = (xs) => xs.map((x, indexNewA) => {  
 const indexAw = Math.trunc(indexNewA / 13);  
  
 const S = *getS*(xs, Aw[indexAw])  
 return S > 6 ? 1 : 0  
  
})  
  
export const zeroWeight = [1, 2].map(getRandowW)  
export const $weight = *createStore*(zeroWeight)  
  
export const setWeights = *createEvent*()

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

import {  
 $M, setDiffs,  
 setSs  
} from "./index";  
  
const handleSetSs = (M, Ss) =>  
 M.map(  
 (m, index) => ({...m, S: Ss[index]})  
 )  
  
$M  
 .on(setSs, handleSetSs)  
  
const handleSetDiffs = (M, diffs) =>  
 M.map(  
 (m, index) => ({...m, diff: diffs[index]})  
 )  
  
$M  
 .on(setSs, handleSetSs)  
 .on(setDiffs, handleSetDiffs)

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, frontier) => *getS*(xs, ws) > frontier,  
 name: 'Бинарная функция'  
 },  
  
]  
  
export const $recognize = *createStore*(recognizeFunctions[0])  
  
export const setRecognize = *createEvent*()

import {  
 $recognize,  
 recognizeFunctions,  
 setRecognize  
} from "./index";  
  
  
$recognize  
 .on(setRecognize, (\_, recognizeId) => recognizeFunctions.find(({id}) => recognizeId === id))