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 №1

тема: «Использование одного нейрона. Функции активации.»

Выполнил: ст. группы МИВТ-221

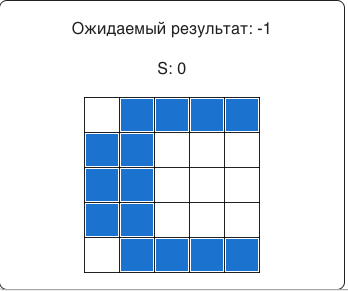
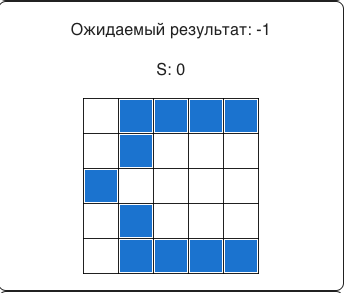
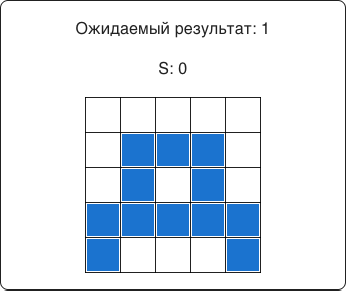
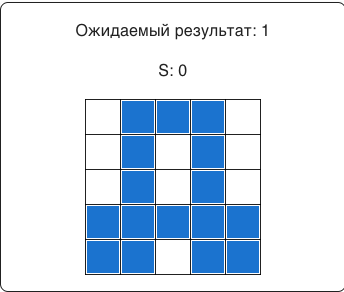
Харитонов Сергей Дмитриевич

Белгород 2022 г.**Лабораторная работа №1**

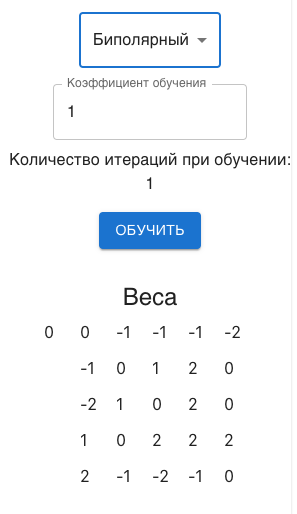
**Использование одного нейрона. Функции активации.**

**Цель работы:** закрепление теоретических знаний, получение практических навыков использования одного нейрона, исследование правила Хебба, исследование функций активации.

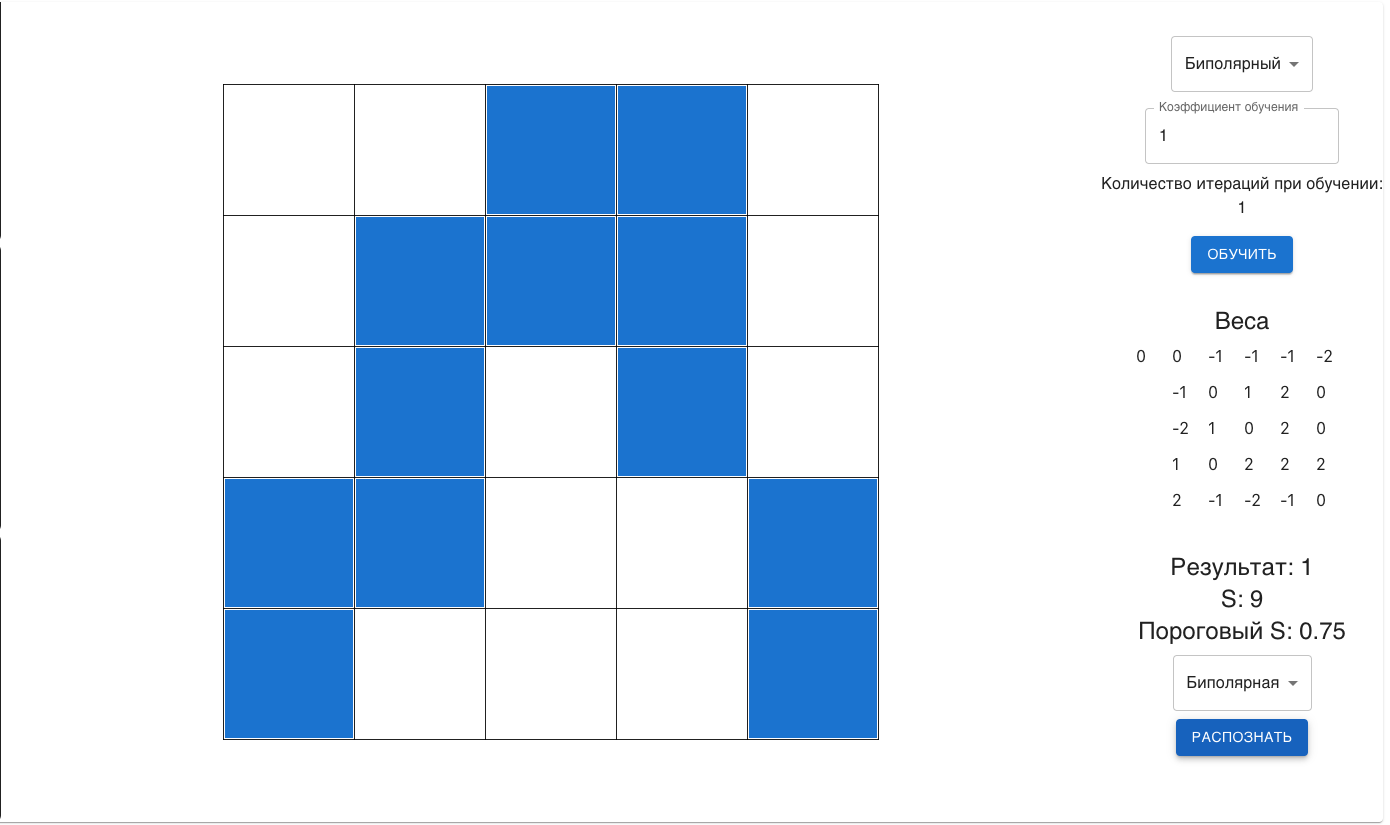
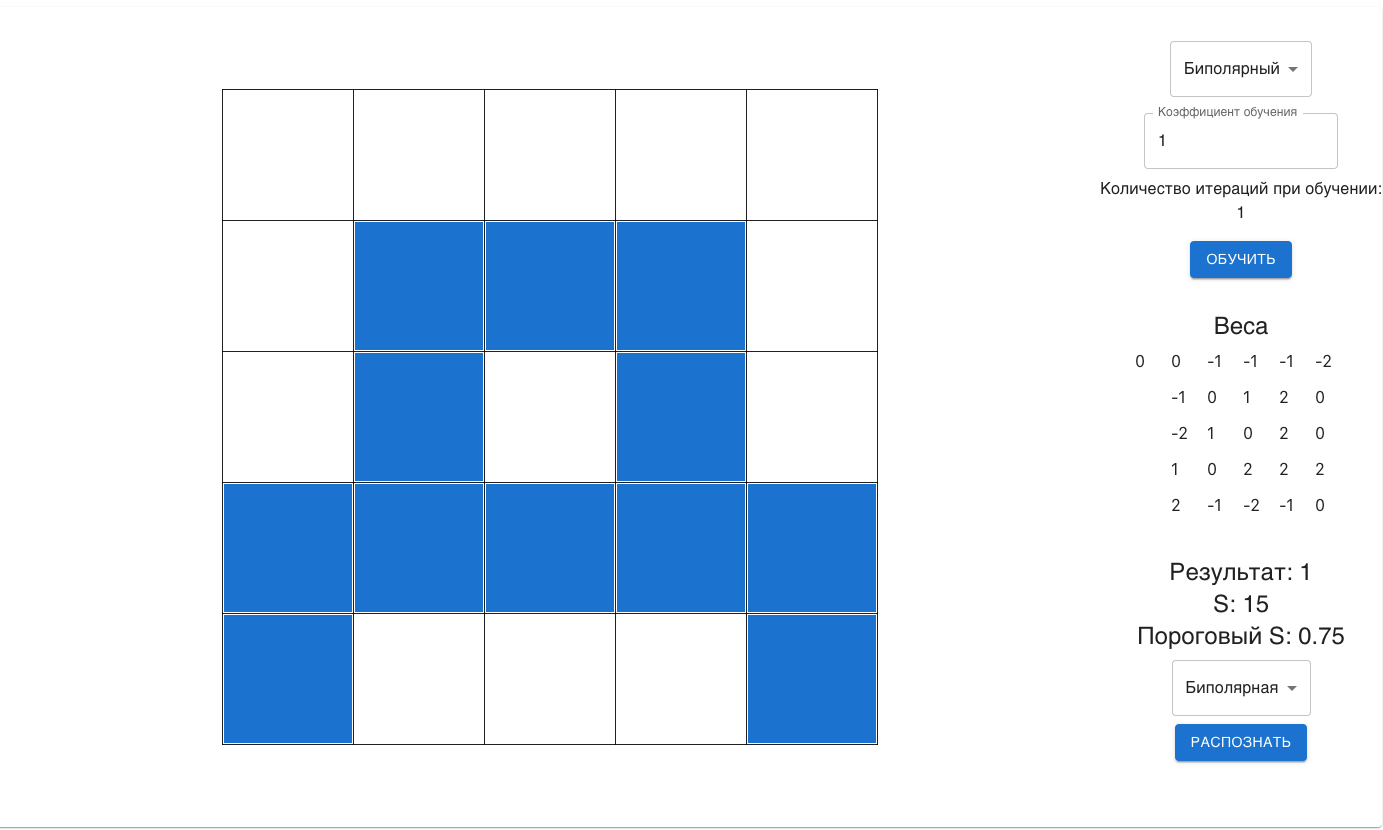
Экранные копии образов для обучения

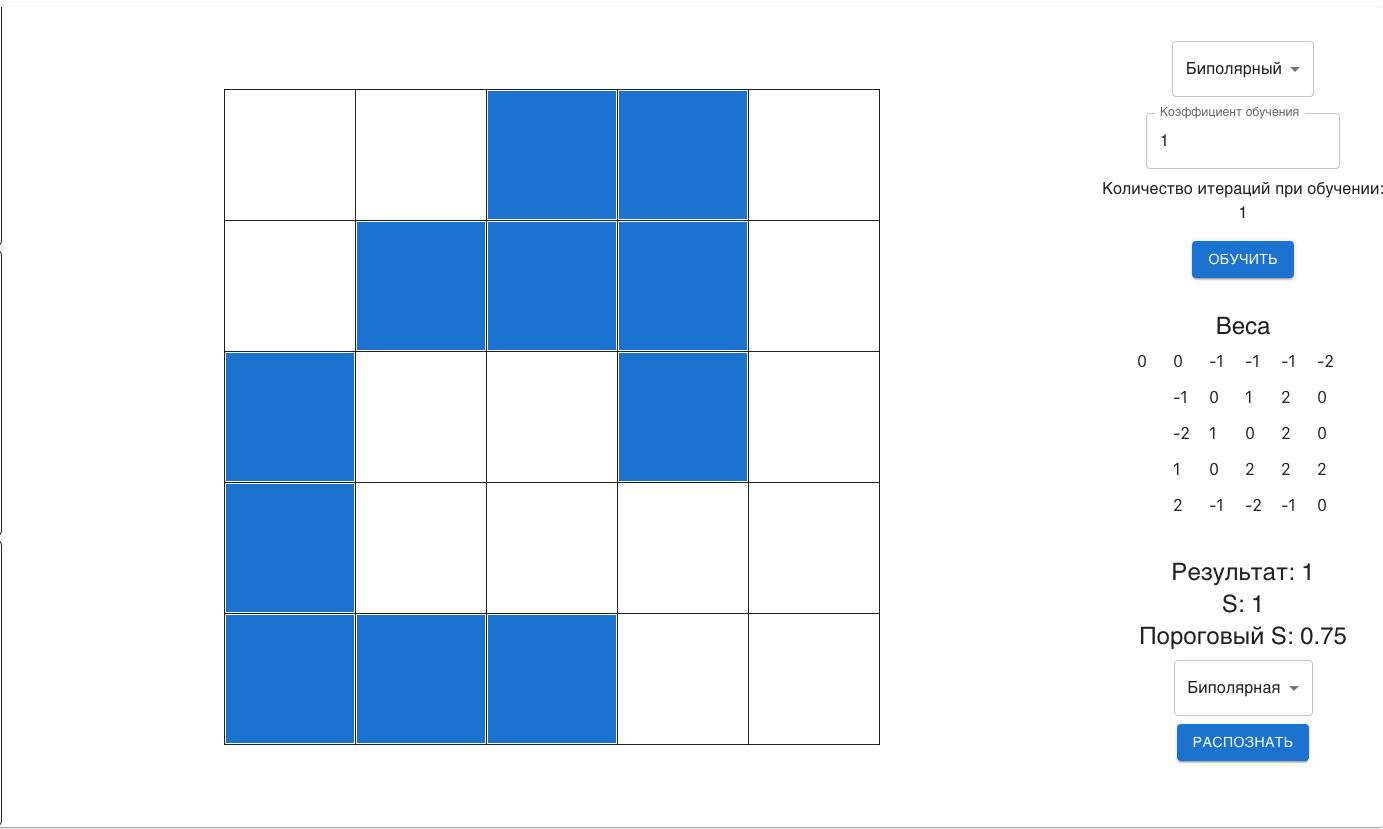
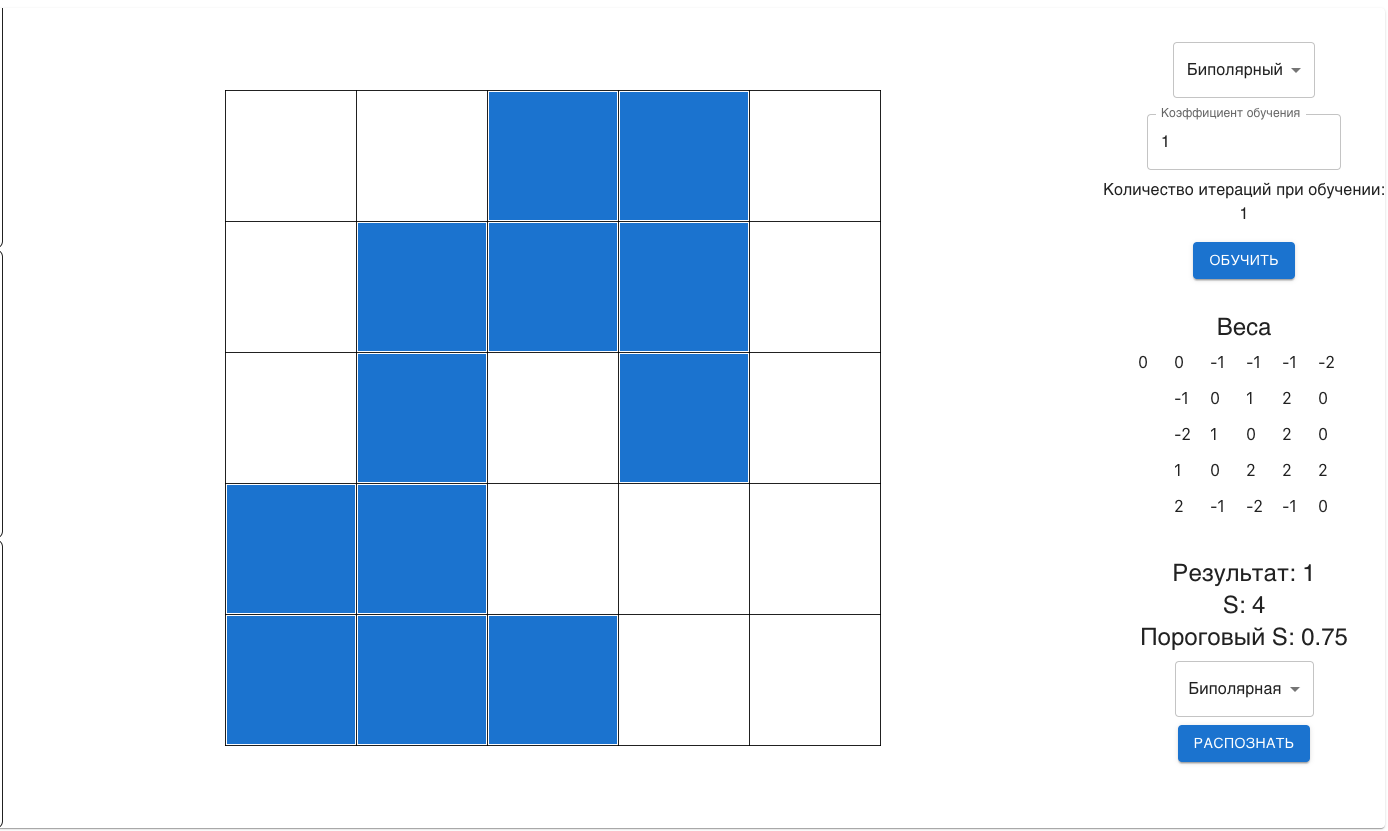


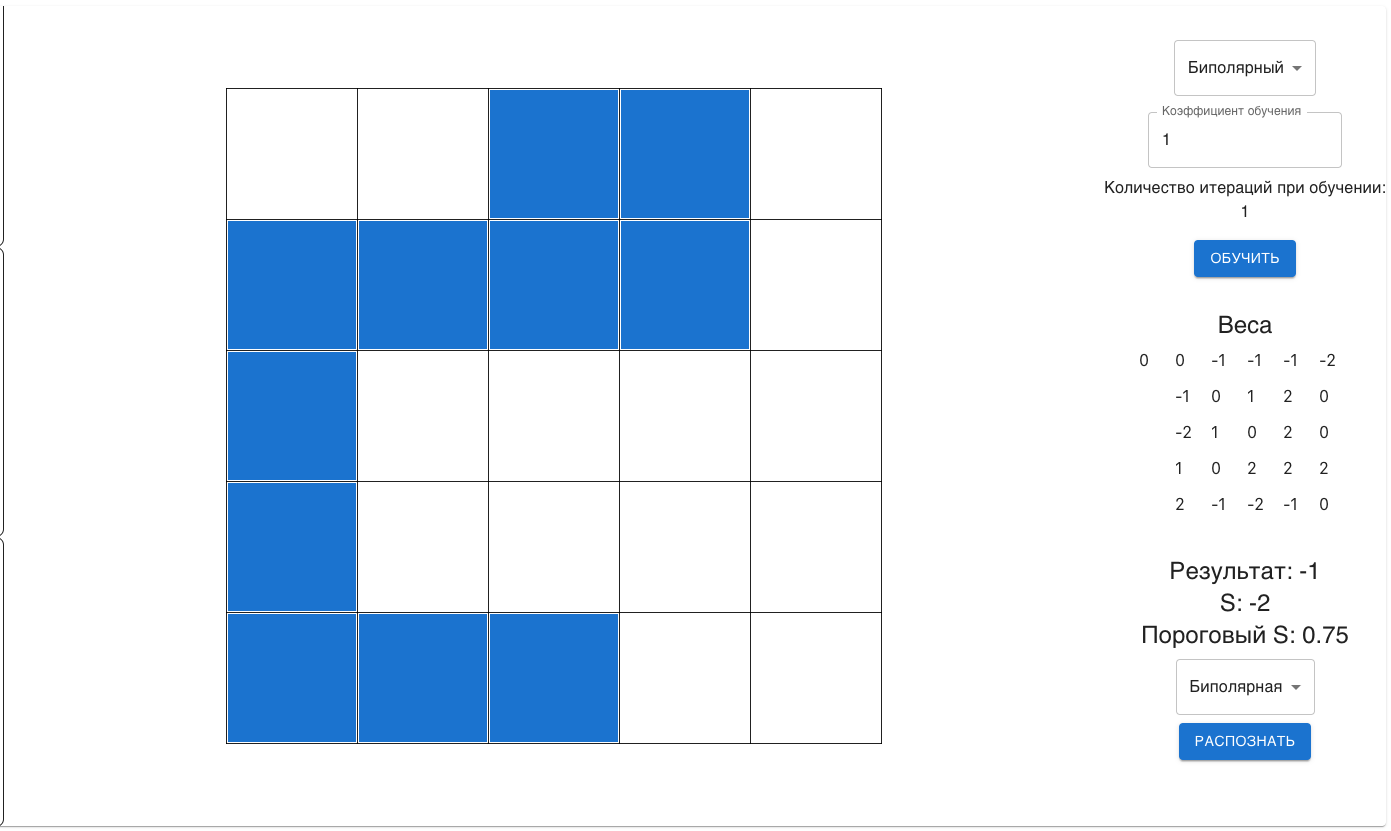
Результаты обучения



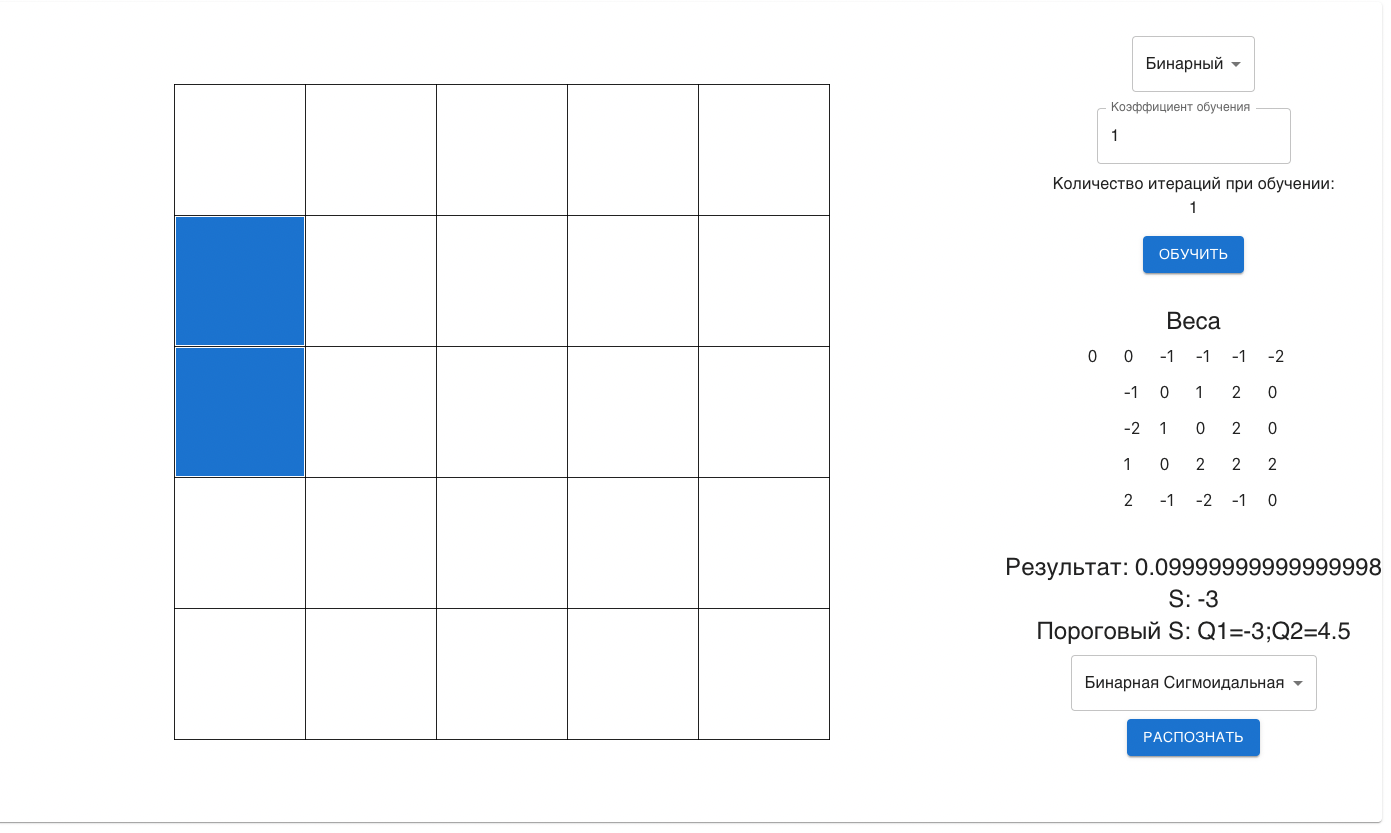
Пошаговое исследование функций активации

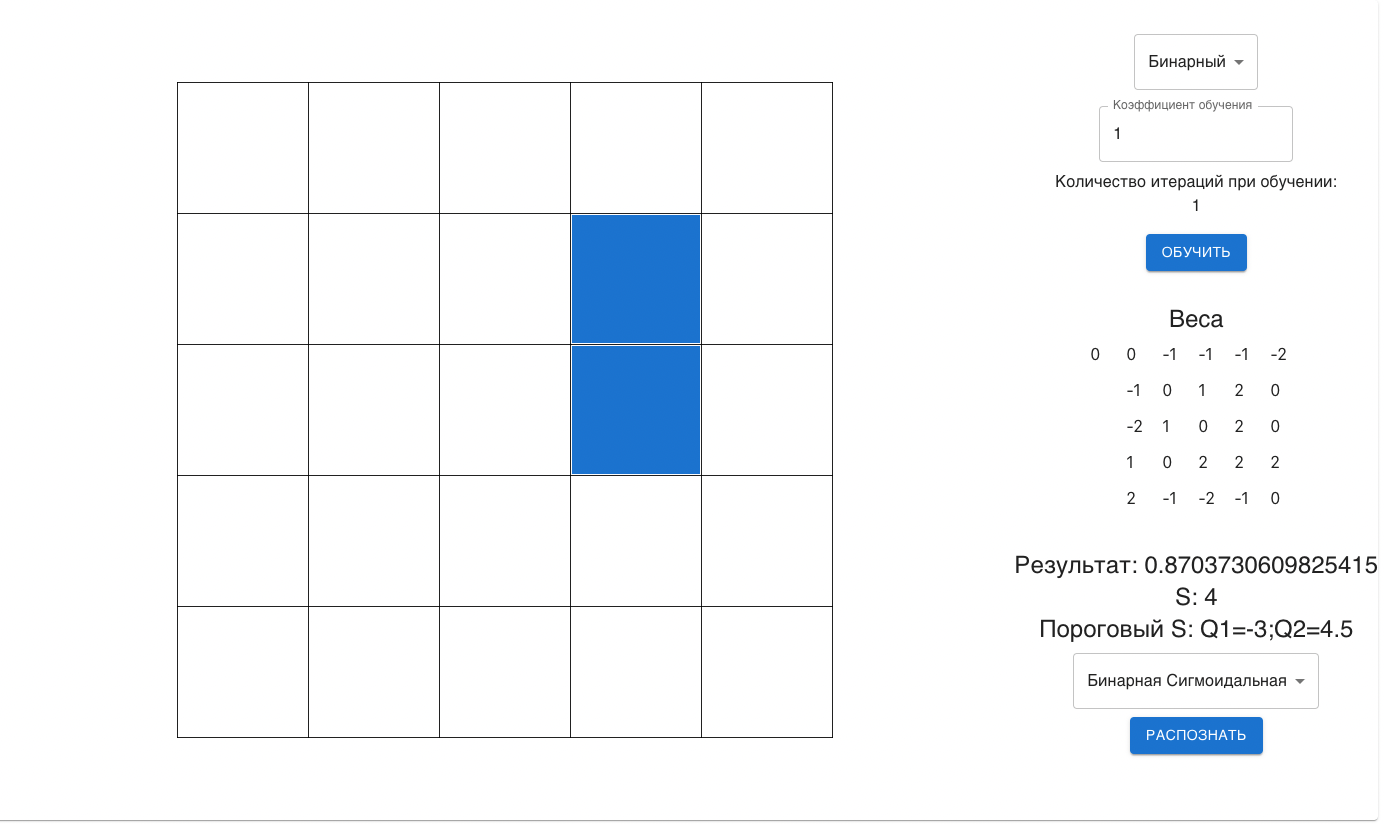




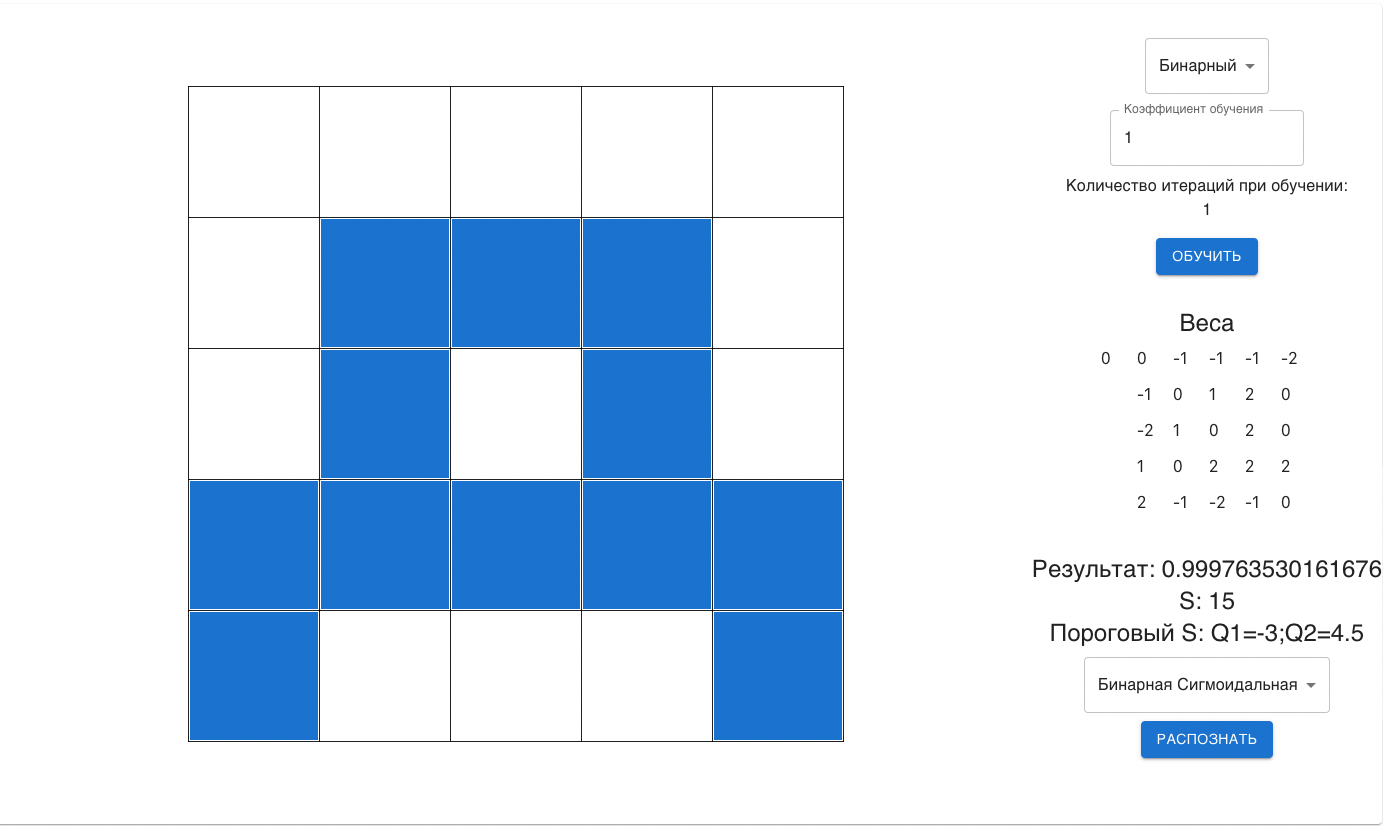


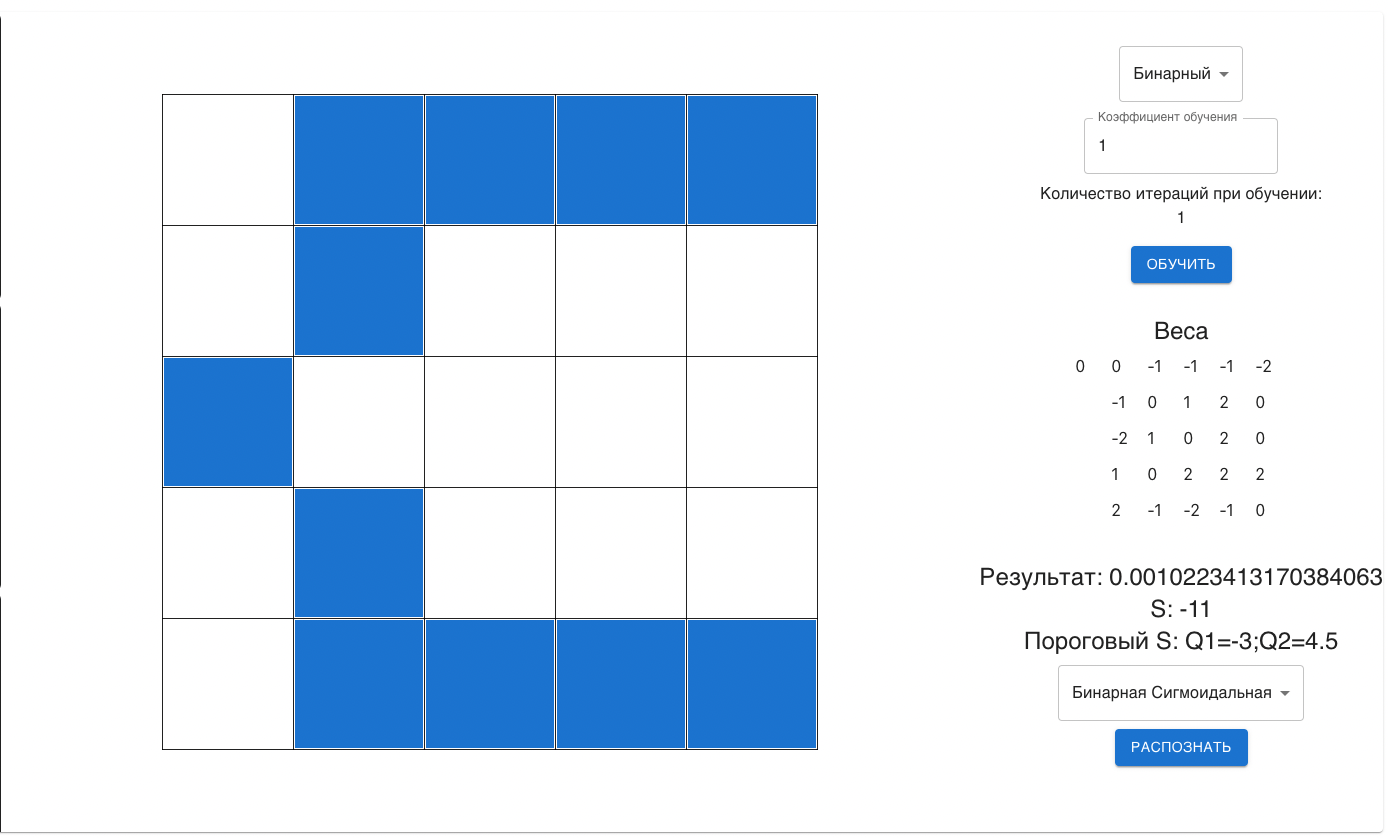
Защита

Корректировака функции активации бинарной сигмоиды заключалась в смещении S, что бы центр отрезка был в нуле и растягивании ее с помощью подбора тау для удовлетворения условиям S=Q1 Y=0.1; S=Q2 Y=0.9  
 Как видим корректировка произошла успешно



При изменении значений в интервале от S [-3,4.5] значение активационной функции медленно меняется, а затем стремится к бинарным





Листинг программы

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

import {getS} from "../teacher";  
import {createEvent, createStore} from "effector";  
  
export function average(nums) {  
 return nums.reduce((a, b) => (a + b)) / nums.length;  
}  
  
export function divider(a, b, part) {  
 return (a + part \* b) / (1. + part)  
}  
  
export const ***recognizeFunctions*** = [  
 {  
 id: 1,  
 recognize: (xs, ws, S1, S0) =>  
 getS(xs, ws) > average([S1, S0])  
 ? 1  
 : -1,  
 name: 'Биполярная',  
 getFrontier: (S1, S0) => average([S1, S0])  
 },  
 {  
 id: 2,  
 recognize: (xs, ws, S1, S0) =>  
 getS(xs, ws) > average([S1, S0])  
 ? 1  
 : 0,  
 name: 'Бинарная',  
 getFrontier: (S1, S0) => average([S1, S0])  
 },  
 {  
 id: 3,  
 recognize: (xs, ws, S1, S0) => {  
 const S = getS(xs, ws)  
 const T =  
 -***Math***.log((1 - 0.9) / 0.9)  
 /  
 average([divider(S1, S0, 1. / 3.), divider(S1, S0, 2. / 3.)])  
 return 1/ (1 + ***Math***.exp(-T\*S))  
 },  
 name: 'Бинарная Сигмоидальная',  
 getFrontier: (S1, S0) => `Q1=${divider(S0, S1, 1. / 3.)};Q2=${divider(S0, S1, 2. / 3.)}`  
 }  
]  
  
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 ***bipolarTeacher*** = ({  
 id: 1,  
 getNextWeight: (old, x, y, k = 1.0) => old + (x || -1) \* y \* k,  
 activation: (xs, ws, S = 0) =>  
 getS(xs, ws) > 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, ws, S = 0) =>  
 getS(xs, ws) > S  
 ? 1  
 : 0,  
 fieldY: 'bin',  
 name: 'Бинарный',  
 learningRate: 1.0  
})  
  
export const ***teachers*** = [***bipolarTeacher***, ***binTeacher***]  
  
export const ***$teacher*** = createStore(***bipolarTeacher***)  
  
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(m => ({t: m.t[teacher.fieldY], x: [1, ...m.x]}))  
 let W = weight.map(() => 0)  
  
 const checkNotEnd = () => teacherMs  
 .some(  
 m => teacher.activation(m.x, W) !== m.t  
 )  
  
 let countOperations = 0;  
 while (checkNotEnd()) {  
 ++countOperations;  
 teacherMs  
 .forEach(  
 m => {  
 W = W.map((oldW, index) => teacher.getNextWeight(oldW, m.x[index], m.t, teacher.learningRate))  
 }  
 )  
 }  
 ***console***.log(teacher)  
 ***setWeights***(W)  
 ***setSs***(  
 teacherMs.map(  
 m => getS(m.x, W)  
 )  
 )  
 ***setCountOperations***(countOperations)  
}  
  
***$teacher*** .on(***changeTeacher***, handleChangeTeacher)  
 .on(***teachEvent***, handleTeach)