

Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №1
За шестой семестр
По дисциплине: «МРЗИС»
Тема: «Бинарная классификация»

Выполнил:
Студент 3 курса
Группы ИИ-26
Рулько М.А.
Проверил:
Андренко К.В.

Брест 2026

Цель работы:

Изучить принципы бинарной классификации и реализовать однослойную нейронную сеть (персептрон) для решения задачи классификации с использованием пороговой функции активации, а также исследовать процесс обучения модели с применением среднеквадратичной ошибки (MSE).

Задачи лабораторной работы:

1. Реализовать алгоритм обучения однослойной нейронной сети с использованием MSE в качестве функции ошибки.
2. Провести обучение сети с разными значениями шага обучения и построить график зависимости MSE от номера эпохи.
3. Выполнить визуализацию результатов классификации:
 - исходные точки обучающей выборки,
 - разделяющую линию (границу между двумя классами).
4. Реализовать режим функционирования сети:
 - пользователь задаёт произвольный входной вектор,
 - сеть вычисляет выходной класс,
 - соответствующая точка отображается на графике,
 - для корректной визуализации рекомендуется выбирать значения из диапазона ВСТАВИТЬ СВОЙ ДИАПАЗОН, например
5. Написать вывод по выполненной работе.

Допускается применение математических и графических библиотек ML-библиотеки и ML-фреймворки использовать нельзя (например: scikit-learn, TensorFlow, PyTorch - запрещены)

Вариант 2

x_1	x_2	e
1	4	0
-1	4	0
1	-4	0
-1	-4	1

КОД ПРОГРАММЫ

```
const dataset = [  
  { x1: 1, x2: 4, label: 0 },  
  { x1: -1, x2: 4, label: 0 },  
  { x1: 1, x2: -4, label: 0 },  
  { x1: -1, x2: -4, label: 1 }  
];  
  
class Perceptron{  
  constructor(learning_rate = 0.1){  
    this.w1 = Math.random() - 0.5;  
    this.w2 = Math.random() - 0.5;  
    this.bias = Math.random() - 0.5;  
    this.lr = 0.1;  
  }  
  
  activate(sum){  
    return sum >= 0 ? 1 : 0;  
  }  
  
  train(dataset = [{}]){  
    let errorSum = 0;
```

```

dataset.forEach(vector => {
    let sum = 0;
    sum += vector.x1 * this.w1 + vector.x2 * this.w2 + this.bias;

    let prediction = this.activate(sum);
    let error = vector.label - prediction;

    if(error!=0){
        this.w1 = this.w1 + this.lr*error*vector.x1;
        this.w2 = this.w2 + this.lr*error*vector.x2;
        this.bias += 0.1 * error;
    }

    errorSum += Math.pow(error, 2);
});

return errorSum / dataset.length;
}
}

class Visualizer {
    constructor(mseCanvasId, decisionCanvasId) {
        this.mseCanvas = document.getElementById(mseCanvasId);
        this.decisionCanvas = document.getElementById(decisionCanvasId);
        this.mseCtx = this.mseCanvas.getContext('2d');
        this.decisionCtx = this.decisionCanvas.getContext('2d');
    }

    drawMSE(history) {
        const ctx = this.mseCtx;
        const w = this.mseCanvas.width;

```

```

const h = this.mseCanvas.height;

ctx.clearRect(0, 0, w, h);

ctx.beginPath();

ctx.strokeStyle = '#2ecc71';

ctx.lineWidth = 1;


history.forEach((mse, i) => {
    const x = (i / (history.length - 1)) * w;
    const y = h - (mse * h); // Инверсия по Y
    if (i === 0) ctx.moveTo(x, y);
    else ctx.lineTo(x, y);
});
ctx.stroke();
}

drawDecisionBoundary(brain, dataset, userPoint = null) {
    const ctx = this.decisionCtx;
    const w = this.decisionCanvas.width;
    const h = this.decisionCanvas.height;
    const scale = 40;
    const offset = { x: w / 2, y: h / 2 };

    ctx.clearRect(0, 0, w, h);

    ctx.strokeStyle = '#eee';
    ctx.beginPath();
    ctx.moveTo(0, offset.y); ctx.lineTo(w, offset.y);
    ctx.moveTo(offset.x, 0); ctx.lineTo(offset.x, h);
    ctx.stroke();

```

```

ctx.strokeStyle = 'red';
ctx.beginPath();
for (let x1 = -10; x1 <= 10; x1 += 0.1) {
    let x2 = (-brain.w1 * x1 - brain.bias) / brain.w2;
    let canvasX = offset.x + x1 * scale;
    let canvasY = offset.y - x2 * scale;
    if (x1 === -10) ctx.moveTo(canvasX, canvasY);
    else ctx.lineTo(canvasX, canvasY);
}
ctx.stroke();

dataset.forEach(p => {
    ctx.fillStyle = p.label === 1 ? 'blue' : 'orange';
    ctx.beginPath();
    ctx.arc(offset.x + p.x1 * scale, offset.y - p.x2 * scale, 5, 0, Math.PI * 2);
    ctx.fill();
});

if (userPoint) {
    ctx.fillStyle = 'black';
    ctx.strokeStyle = 'black';
    const ux = offset.x + userPoint.x1 * scale;
    const uy = offset.y - userPoint.x2 * scale;
    ctx.strokeRect(ux - 6, uy - 6, 12, 12);
    ctx.fillText(`User: class ${userPoint.label}`, ux + 10, uy);
}
}
}

const viz = new Visualizer('mseChart', 'decisionChart');
```

```

let brain = new Perceptron(0.1);
let mseHistory = [];

for (let epoch = 0; epoch < 50; epoch++) {
  let mse = brain.train(dataset);
  mseHistory.push(mse);
  if (mse === 0) break;
}

viz.drawMSE(mseHistory);
viz.drawDecisionBoundary(brain, dataset);

function predictUserPoint(x1, x2) {
  const sum = x1 * brain.w1 + x2 * brain.w2 + brain.bias;
  const label = brain.activate(sum);
  viz.drawDecisionBoundary(brain, dataset, {x1, x2, label});
  return label;
}

```

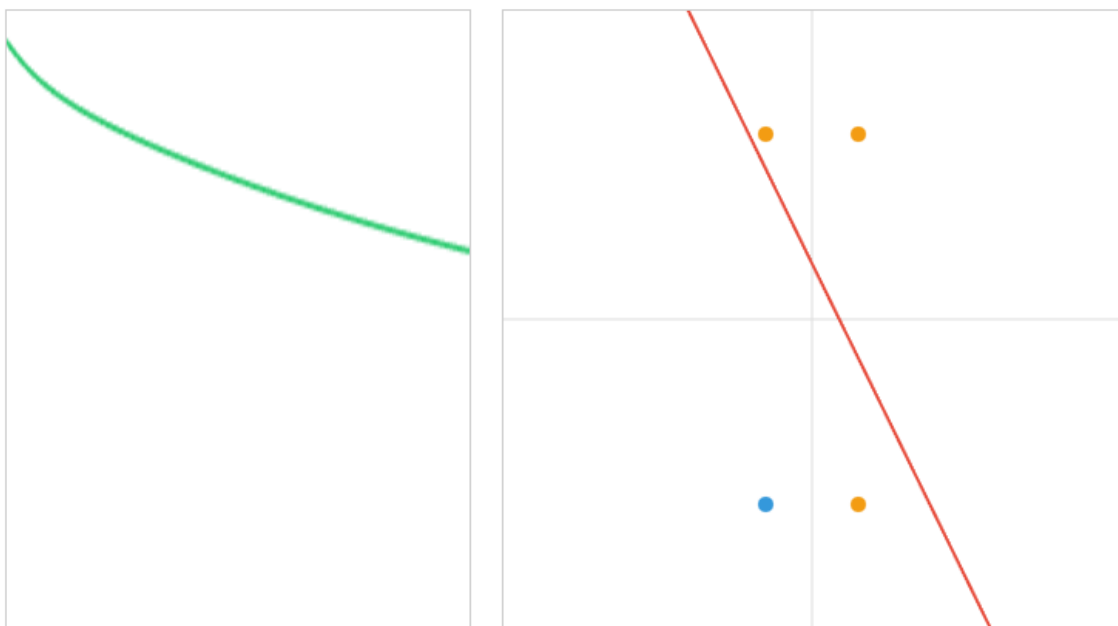


График визуализации (наши точки соответственно при lr 0.1)

Вывод :

В данной лабораторной работе построили однослойную нейронную сеть (персептрон).

