1.1

const hebbFunction = (data) => {

  const N = data.length;

  const dentriesLength = data[0].input.length;

  const weights = [];

  let resultsVector = [];

  for (let j = 0; j < N; j++) {

    resultsVector[j] = data[j].output;

  }

  // w0

  weights[0] = resultsVector.reduce((acc, x) => acc + x, 0) / N;

  //w1, w2, ..., wN

  for (let i = 0; i < dentriesLength; i++) {

    let scalarProduct = 0;

    for (let j = 0; j < N; j++) {

      const currentVal = data[j].input[i];

      const expectedResult = resultsVector[j];

      scalarProduct += (1/currentVal) \* (1/expectedResult);

    }

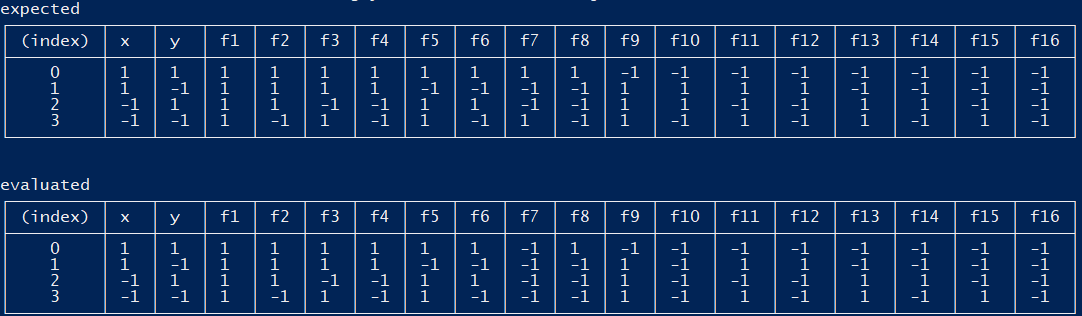
    weights[i + 1] = scalarProduct / N;

  }

  return weights;

}

1.2



Incorrect f7, f10 (XOR, NXOR)

2.1

|  |  |  |  |
| --- | --- | --- | --- |
| Очікувалось (14 ст) | Отримано | Очікувалось (15 ст) | Отримано |
| 1 | 1 | 1 | 1 |
| 1 | 1 | -1 | -1 |
| 1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | -1 | -1 |
| -1 | -1 | 1 | 1 |
| -1 | -1 | -1 | -1 |

Ваги (14 ст) W = (0.25, 0.25, 0.75, 0.25)

Ваги (15 ст) W = (-0.25, -0.25, 0.25, 0.75)

2.2

|  |  |  |  |
| --- | --- | --- | --- |
| Очікувалось f1 | Отримано f1 | Очікувалось f2 | Отримано f2 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | -1 | -1 |
| 1 | -1 | -1 | -1 |
| -1 | -1 | -1 | -1 |
| 1 | 1 | 1 | 1 |
| 1 | 1 | -1 | -1 |
| -1 | -1 | 1 | 1 |
| -1 | -1 | -1 | -1 |

f1 W(0.25, 0.023809…, 0.8571428…, 0.16666…)

f2 W(-0.25, -0.625, 0.5, 0.183333 …)