МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-приложений)

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ:

по дисциплине «Информационная безопасность»

Исполнитель

студент (ка) 3 курса группы 6 Розель Станислав Александрович

(Ф.И.О.)

Минск 2025

### Реализация алгоритма обучения Анти-Хэбба

В лабораторной работе нужно было разработать алгоритм Анти-Хэбба c K = 5 N = 8 L = -5. На листинге представлен код персептрона и нейронной сети TPM.

|  |
| --- |
| class Perceptron {  constructor(config, rng) {  this.config = config;  this.weights = [];  this.inputs = [];  for (let i = 0; i < config.inputsCount; i++) {  this.weights[i] = Math.floor(rng() \* (2 \* config.weightLimit + 1)) - config.weightLimit;  if (this.weights[i] === 0) {  this.weights[i] = rng() < 0.5 ? -1 : 1;  }  }  }  sign(x) {  return x <= 0 ? -1 : 1;  }  calculateOutput(inputs) {  this.inputs = [...inputs];  let sum = 0;    for (let i = 0; i < this.config.inputsCount; i++) {  sum += this.weights[i] \* inputs[i];  }    return this.sign(sum);  }  train(networkOutput, otherNetworkOutput, perceptronOutput) {  const myOutput = perceptronOutput !== undefined ? perceptronOutput : this.calculateOutput(this.inputs);  if (networkOutput === otherNetworkOutput) {  if (myOutput === networkOutput) {  for (let i = 0; i < this.config.inputsCount; i++) {  const deltaWeight = myOutput \* this.inputs[i];  this.weights[i] += deltaWeight;  this.weights[i] = Math.max(-this.config.weightLimit,  Math.min(this.config.weightLimit, this.weights[i]));  }  }  }  }  getWeights() {  return [...this.weights];  }  }  class NeuralNetwork {  constructor(config, rng) {  this.config = config;  this.perceptrons = [];  this.output = 0;  this.perceptronOutputs = [];    for (let i = 0; i < config.perceptronsCount; i++) {  this.perceptrons.push(new Perceptron(config.toPerceptronConfig(), rng));  }  }  calculateOutput(inputs) {  this.output = 1;  this.perceptronOutputs = [];    for (let k = 0; k < this.config.perceptronsCount; k++) {  const startIdx = k \* this.config.inputsPerPerceptron;  const endIdx = (k + 1) \* this.config.inputsPerPerceptron;  const perceptronInputs = inputs.slice(startIdx, endIdx);    const perceptronOutput = this.perceptrons[k].calculateOutput(perceptronInputs);  this.perceptronOutputs[k] = perceptronOutput;  this.output \*= perceptronOutput;  }    return this.output;  }  train(otherNetworkOutput) {  for (let k = 0; k < this.config.perceptronsCount; k++) {  if (this.perceptronOutputs[k] === this.output) {  this.perceptrons[k].train(this.output, otherNetworkOutput, this.perceptronOutputs[k]);  }  }  }  getWeightsMatrix() {  return this.perceptrons.map(perceptron => perceptron.getWeights());  }  isSynchronizedWith(otherNetwork) {  const myWeights = this.getWeightsMatrix();  const otherWeights = otherNetwork.getWeightsMatrix();    for (let k = 0; k < this.config.perceptronsCount; k++) {  for (let i = 0; i < this.config.inputsPerPerceptron; i++) {  if (myWeights[k][i] !== otherWeights[k][i]) {  return false;  }  }  }  return true;  }  getWeightDifferences(otherNetwork) {  const myWeights = this.getWeightsMatrix();  const otherWeights = otherNetwork.getWeightsMatrix();  let differences = 0;  for (let k = 0; k < this.config.perceptronsCount; k++) {  for (let i = 0; i < this.config.inputsPerPerceptron; i++) {  if (myWeights[k][i] !== otherWeights[k][i]) {  differences++;  }  }  }  return differences;  }  } |

Листинг 1.1 – Код классов персептрона и нейронной сети TPM

На рисунке 1.1 представлен вывод подсчета минимального, среднего, максимального числа шагов до синхрониации.

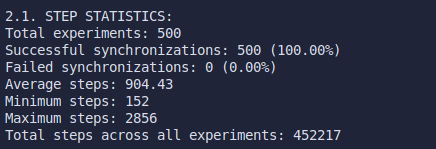


Рисунок 1.1 – Вывод числа шагов до сонхронизации

На рисунке 1.2 представлен вывод составленного распределения: число синхронизаций – число шагов.

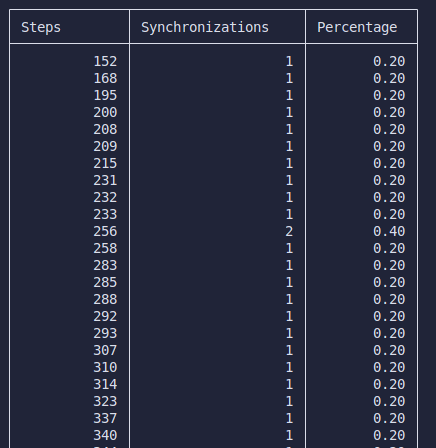


Рисунок 1.2 - Число синхронизаций – число шагов

Так же нужно было подсчитатать среднее время между синхронизациями. Вывод представлен на рисунке 1.3.

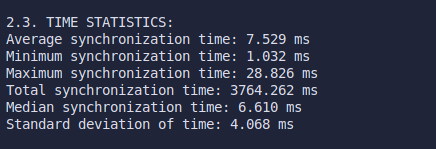


Рисунок 1.3 - Среднее время