

Лабораторная работа №1

Нейронная сеть Хопфилда

Вариант 5: буквы Д, Н, Х

Елисеев Данила, 2025, ИС

26 декабря 2025 г.

Содержание

1	Цель работы	2
2	Теоретическая часть	2
2.1	Общие сведения о сети Хопфилда	2
2.2	Топология сети	2
2.3	Обучение сети	2
2.4	Воспроизведение	3
2.5	Режимы работы	3
2.6	Ёмкость сети	3
3	Описание алгоритма	3
3.1	Алгоритм обучения	3
3.2	Алгоритм воспроизведения	3
4	Реализация	4
4.1	Структура проекта	4
4.2	Эталонные образы	4
4.3	Ключевые фрагменты кода	4
4.3.1	Обучение сети по правилу Хебба	4
4.3.2	Синхронное воспроизведение	5
5	Результаты экспериментов	5
5.1	Методика тестирования	5
5.2	Результаты для буквы Д	5
5.3	Результаты для буквы Н	6
5.4	Результаты для буквы Х	6
5.5	Сводная статистика	6
6	Выводы	6

1. Цель работы

Изучение топологии и алгоритма функционирования нейронной сети Хопфилда. Реализация программы для распознавания зашумленных образов русских букв.

Задачи:

1. Реализовать нейронную сеть Хопфилда на языке C++
2. Обучить сеть на 3 образах букв (Д, Н, Х) размером 10×10
3. Исследовать устойчивость распознавания при различных уровнях шума
4. Сравнить синхронный и асинхронный режимы работы сети

2. Теоретическая часть

2.1. Общие сведения о сети Хопфилда

Сеть Хопфилда — это однослойная, симметричная, нелинейная, автоассоциативная нейронная сеть, которая запоминает бинарные или биполярные образы. Сеть характеризуется наличием обратных связей: информация с выхода каждого нейрона поступает на вход всех остальных нейронов.

2.2. Топология сети

Для данной работы используется сеть из $n = 100$ нейронов (для образов размером 10×10). Образы кодируются биполярным вектором: $a_i \in \{-1, 1\}$.

Каждый нейрон связан со всеми остальными нейронами (полносвязная сеть), кроме самого себя ($w_{ii} = 0$).

2.3. Обучение сети

Обучение сети осуществляется по правилу Хебба:

$$w_{ij} = \begin{cases} \sum_{k=1}^m a_i^k \cdot a_j^k, & i \neq j \\ 0, & i = j \end{cases}$$

где:

- w_{ij} — вес связи от i -го нейрона к j -му
- m — количество образов для обучения
- a_i^k — i -й элемент k -го образа

Матрица весовых коэффициентов W является **симметричной** ($w_{ij} = w_{ji}$) с **нулевой главной диагональю** ($w_{ii} = 0$).

2.4. Воспроизведение

Для воспроизведения используется формула:

$$a_i(t+1) = f\left(\sum_{j=1}^n w_{ij} \cdot a_j(t)\right)$$

где f — биполярная пороговая функция активации:

$$f(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases}$$

2.5. Режимы работы

Синхронный режим: все нейроны одновременно обновляют свои состояния.

Асинхронный режим: на каждом шаге обновляется только один случайно выбранный нейрон.

2.6. Ёмкость сети

Максимальное количество образов:

$$m \approx \frac{n}{2 \ln n + \ln \ln n}$$

Для $n = 100$: $m \approx 9$ образов.

3. Описание алгоритма

3.1. Алгоритм обучения

1. Инициализировать матрицу весов W нулями
2. Для каждой пары (i, j) , где $i \neq j$: вычислить $w_{ij} = \sum_{k=1}^m a_i^k \cdot a_j^k$
3. Диагональные элементы: $w_{ii} = 0$

3.2. Алгоритм воспроизведения

1. Подать на вход зашумленный образ $a(0)$
2. Повторять до стабилизации:
 - Вычислить: $sum_i = \sum_{j=1}^n w_{ij} \cdot a_j(t)$
 - Обновить: $a_i(t+1) = \text{sign}(sum_i)$
3. Если $a(t+1) = a(t)$ — сеть стабилизировалась

4. Реализация

4.1. Структура проекта

- solution.cpp — основная программа на C++
- patterns/ — эталонные образы (D.txt, N.txt, X.txt)
- tests/ — тестовые образы (360 файлов)
- tests/results.csv — результаты тестирования

4.2. Эталонные образы

Буквы Д, Н, Х представлены в виде матриц 10×10:

Буква Д:

```
#####
#   #
#   #
#   #
#   #
#   #
##  ##
#####
#       #
#       #
```

Буква Н:

```
#   #
#   #
#   #
#   #
#####
#####
#   #
#   #
#   #
#   #
```

Буква Х:

```
#       #
#       #
#   #
#   #
##
##
#   #
#       #
#       #
```

4.3. Ключевые фрагменты кода

4.3.1. Обучение сети по правилу Хебба

```
1 void train() {
2     // Инициализация весов нулями
3     for (int i = 0; i < N; i++) {
4         for (int j = 0; j < N; j++) {
5             weights[i][j] = 0;
6         }
7     }
8
9     // Обучение по правилу Хебба
10    for (int i = 0; i < N; i++) {
11        for (int j = 0; j < N; j++) {
12            if (i != j) {
13                for (int k = 0; k < NUM_PATTERNS; k++) {
14                    weights[i][j] += patterns[k][i] * patterns[k][j];
15                }
16            }
17        }
18    }
19 }
```

Listing 1: Функция обучения сети Хопфилда

4.3.2. Синхронное воспроизведение

```
1 Pattern recallSync(const Pattern& input, int& iterations) {
2     Pattern current = input;
3     Pattern next;
4     iterations = 0;
5
6     for (int iter = 0; iter < MAX_ITERATIONS; iter++) {
7         iterations++;
8
9         for (int i = 0; i < N; i++) {
10             int sum = 0;
11             for (int j = 0; j < N; j++) {
12                 sum += weights[i][j] * current[j];
13             }
14             next[i] = (sum >= 0) ? 1 : -1;
15         }
16
17         if (next == current) break; // Стабилизация
18         current = next;
19     }
20
21     return current;
22 }
```

Listing 2: Функция синхронного воспроизведения

5. Результаты экспериментов

5.1. Методика тестирования

Для каждой буквы и каждого уровня шума (10%, 20%, ..., 100%) было сгенерировано по 10 тестовых образов. Всего: $3 \times 12 \times 10 = 360$ тестов.

5.2. Результаты для буквы Д

Таблица 1: Результаты распознавания буквы Д

Шум	Sync успех	Sync итер.	Async успех	Async итер.
10%	10/10	2.0	10/10	200
20%	10/10	2.0	10/10	200
30%	10/10	2.0	10/10	200
35%	10/10	2.2	10/10	220
40%	6/10	2.2	7/10	230
45%	5/10	2.8	5/10	250
50%	4/10	2.3	4/10	200
60%	1/10	2.3	2/10	220
70%+	0/10	2.0	0/10	200

5.3. Результаты для буквы Н

Таблица 2: Результаты распознавания буквы Н

Шум	Sync успех	Sync итер.	Async успех	Async итер.
10%	10/10	2.0	10/10	200
20%	10/10	2.0	10/10	200
30%	10/10	2.0	10/10	200
35%	10/10	2.0	10/10	200
40%	7/10	2.3	7/10	200
45%	5/10	2.5	6/10	220
50%	1/10	102	3/10	210
60%+	0/10	2.2	0/10	210

5.4. Результаты для буквы Х

Таблица 3: Результаты распознавания буквы Х

Шум	Sync успех	Sync итер.	Async успех	Async итер.
10%	10/10	2.0	10/10	200
20%	10/10	2.0	10/10	200
30%	10/10	2.0	10/10	200
35%	10/10	2.1	10/10	200
40%	8/10	2.2	7/10	200
45%	5/10	2.8	5/10	220
50%	0/10	302	1/10	200
60%+	0/10	2.3	0/10	210

5.5. Сводная статистика

Таблица 4: Сводная статистика по уровням шума

Шум	Синхронный	Асинхронный
10%	100%	100%
20%	100%	100%
30%	100%	100%
35%	100%	100%
40%	70%	70%
45%	50%	53%
50%	17%	27%
60%	3%	7%
70%+	0%	0%

6. Выводы

1. **Успешная реализация:** Нейронная сеть Хопфилда реализована и обучена на 3 образах букв (Д, Н, Х) размером 10×10.

2. **Устойчивость к шуму:** При уровне шума до 35% — 100% успешное распознавание.
3. **Критический порог:** При 40-50% шума качество падает до 17-70%.
4. **Потеря информации:** При шуме выше 60% распознавание невозможно.
5. **Сравнение режимов:** Синхронный быстрее (2-3 итерации), асинхронный немного лучше при высоком шуме.
6. **Рекомендации:** Использовать сеть при шуме до 30-35%.