Лабораторная работа №3

Сеть РБФ (Радиальная Базисная Функция)

Вариант 2: классы N, F, I, P, D

Елисеев Данила, 2025, ИС

26 декабря 2025 г.

Содержание

1. Цель работы

Изучение топологии и алгоритма функционирования сети РБФ (радиальная базисная функция). Реализация программы для распознавания зашумленных образов.

Задачи:

1. Реализовать сеть РБФ на языке C++
2. Обучить сеть на 5 классах образов (N, F, I, P, D) размером 6×6
3. Исследовать способность сети распознавать зашумленные образы
4. Проанализировать процент подобия распознаваемых образов к каждому классу

2. Теоретическая часть

2.1. Общие сведения о сети РБФ

Сеть РБФ (радиальная базисная функция) является аналогом многослойного персептрона. Скорость обучения такой сети гораздо выше, причем допускается полностью аналитический подход к расчету весовых коэффициентов. Однако эти положительные моменты сопровождаются рядом недостатков, главным из которых является ухудшение точности аппроксимации.

2.2. Топология сети

Сеть РБФ состоит из трех слоев:

- **Входной слой** (распределительный) — передает входные данные на РБФ слой
- **РБФ слой** (скрытый) — содержит РБФ ячейки, каждая из которых вычисляет гауссову функцию
- **Выходной слой** (суммирующий) — линейная комбинация выходов РБФ ячеек

Для данной работы используется сеть с $n = 36$ входами (образы размером 6×6), $h = 5$ РБФ ячейками (по одной на класс) и $m = 5$ выходными нейронами (по одному на класс).

2.3. Функция РБФ ячейки

Классический закон, по которому РБФ ячейка функционирует, определяется формулой **гауссового колокола**:

$$g_j = \exp\left(-\frac{\|x - t_j\|^2}{\sigma_j^2}\right)$$

где:

- x — входной вектор
- t_j — вектор, определяющий математическое ожидание (центр кластера) РБФ ячейки
- σ_j — среднеквадратическое отклонение (параметр разброса)

Евклидово расстояние между векторами x и t_j вычисляется как:

$$\|x - t_j\|^2 = \sum_{i=1}^{n} (x_i - t_{ij})^2$$

2.4. Обучение РБФ ячеек

Выбор центров:

- В качестве центра выбирается центр кластера в пространстве признаков
- В простейшем случае, если класс задается одним идеальным образом, этот образ и будет являться вектором t_j

Выбор параметра разброса σ_j:

- Выбирается как половина расстояния до ближайшего центра ячейки, соответствующей другому классу
- $\sigma_j = d_{jk}/2$, где d_{jk} — расстояние до ближайшего центра другого класса

2.5. Выходной слой

Выходной слой РБФ сети состоит из суммирующих ячеек:

$$y_k = \sum_{j=1}^{h} w_{jk} \times g_j$$

где w_{jk} — вес связи от j-й РБФ ячейки к k-му выходному нейрону.

2.6. Обучение выходного слоя

Обучение выходного слоя производится по алгоритму градиентного спуска:

$$w_{jk} := w_{jk} + \alpha \times d_k \times g_j$$

где:

- $d_k = y_k^r - y_k$ — ошибка k-го нейрона
- α — скорость обучения
- y_k^r — желаемый выход

Поскольку функция активации в выходном слое сети РБФ **линейная** и ее производная равна 1, формула упрощается по сравнению с многослойным персептроном.

2.7. Воспроизведение

Сеть функционирует по формулам:
РБФ слой:

$$g_j = \exp\left(-\frac{\|x - t_j\|^2}{\sigma_j^2}\right)$$

Выходной слой:

$$y_k = \sum_{j=1}^{h} w_{jk} \times g_j$$

3. Описание алгоритма

3.1. Алгоритм инициализации РБФ ячеек

1. Для каждого класса $i = 1, \ldots, m$:
   - Установить центр РБФ ячейки: $t_i =$ идеальный образ класса i
2. Для каждой РБФ ячейки $j = 1, \ldots, h$:
   - Найти минимальное расстояние до центра другого класса: $d_{min} = \min_{k \neq j} \|t_j - t_k\|$
   - Установить параметр разброса: $\sigma_j = d_{min}/2$

3.2. Алгоритм обучения выходного слоя

1. Инициализировать веса w_{jk} случайными малыми значениями
2. Повторять до сходимости или достижения максимального числа итераций:
   - Для каждого обучающего образа x^k класса k:
     - Вычислить выходы РБФ слоя: $g_j = \exp\left(-\frac{\|x^k - t_j\|^2}{\sigma_j^2}\right)$
     - Вычислить выходы выходного слоя: $y_l = \sum_{j=1}^{h} w_{jl} \times g_j$
     - Вычислить ошибку: $d_l = y_l^r - y_l$, где $y_l^r = 1$ если $l = k$, иначе 0
     - Скорректировать веса: $w_{jl} := w_{jl} + \alpha \times d_l \times g_j$

3.3. Алгоритм распознавания

1. Подать на вход зашумленный образ x
2. Вычислить выходы РБФ слоя: $g_j = \exp\left(-\frac{\|x - t_j\|^2}{\sigma_j^2}\right)$
3. Вычислить выходы выходного слоя: $y_k = \sum_{j=1}^{h} w_{jk} \times g_j$
4. Нормализовать выходы в проценты: $p_k = \frac{y_k}{\sum_{l=1}^{m} y_l} \times 100\%$
5. Класс с максимальным процентом является результатом распознавания

4. Реализация

4.1. Структура проекта

- `solution.cpp` — основная программа на C++
- `patterns/` — эталонные образы (N.txt, F.txt, I.txt, P.txt, D.txt)
- `tests/` — тестовые зашумленные образы

4.2. Идеальные образы для обучения

Образы классов N, F, I, P, D представлены в виде матриц 6×6:

```
Класс N (6x6):      Класс F (6x6):      Класс I (6x6):
■ □ □ □ □ ■         ■ ■ ■ ■ ■ ■         ■ ■ ■ ■ ■ ■
■ ■ □ □ □ ■         ■ □ □ □ □ □         □ □ ■ ■ □ □
■ □ ■ □ □ ■         ■ □ □ □ □ □         □ □ ■ ■ □ □
■ □ □ ■ □ ■         ■ ■ ■ ■ □ □         □ □ ■ ■ □ □
■ □ □ □ ■ ■         ■ □ □ □ □ □         □ □ ■ ■ □ □
■ □ □ □ □ ■         ■ □ □ □ □ □         ■ ■ ■ ■ ■ ■

Класс P (6x6):      Класс D (6x6):
■ ■ ■ ■ □ □         ■ ■ ■ ■ □ □
■ □ □ □ ■ □         ■ □ □ □ ■ □
■ □ □ □ ■ □         ■ □ □ □ ■ □
■ ■ ■ ■ □ □         ■ □ □ □ ■ □
■ □ □ □ □ □         ■ □ □ □ ■ □
■ □ □ □ □ □         ■ ■ ■ ■ □ □
```

4.3. Ключевые фрагменты кода

4.3.1. Вычисление выхода РБФ ячейки

```
double gaussian(const Pattern& input, const Pattern& center, double sigma)
    {
    double distance_sq = 0.0;
    for (int i = 0; i < INPUT_SIZE; i++) {
        double diff = input[i] - center[i];
        distance_sq += diff * diff;
    }
    return exp(-distance_sq / (sigma * sigma));
}
```

Listing 1: Функция гауссовой активации

4.3.2. Инициализация параметров разброса

```
void initializeSigmas() {
    for (int i = 0; i < NUM_CLASSES; i++) {
        double min_dist = INFINITY;
        for (int j = 0; j < NUM_CLASSES; j++) {
            if (i != j) {
                double dist = euclideanDistance(centers[i], centers[j]);
                if (dist < min_dist) {
                    min_dist = dist;
                }
            }
        }
        // sigma = половинарасстояниядоближайшегоцентра
        sigmas[i] = min_dist / 2.0;
    }
}
```

Listing 2: Вычисление параметров σ для РБФ ячеек

4.3.3. Обучение выходного слоя

```
void trainOutputLayer() {
    for (int epoch = 0; epoch < MAX_EPOCHS; epoch++) {
        double max_error = 0.0;

        for (int c = 0; c < NUM_CLASSES; c++) {
            // ВычислениевыходовРБФслоя
            vector<double> rbf_outputs(NUM_CLASSES);
            for (int j = 0; j < NUM_CLASSES; j++) {
                rbf_outputs[j] = gaussian(patterns[c], centers[j],
                    sigmas[j]);
            }

            // Вычислениевыходасетиикоррекциявесов
            for (int k = 0; k < NUM_CLASSES; k++) {
                double y_k = 0.0;
                for (int j = 0; j < NUM_CLASSES; j++) {
                    y_k += weights[j][k] * rbf_outputs[j];
                }

                double target = (k == c) ? 1.0 : 0.0;
                double error = target - y_k;
                max_error = max(max_error, abs(error));

                // Коррекциявесов
                for (int j = 0; j < NUM_CLASSES; j++) {
                    weights[j][k] += LEARNING_RATE * error *
                        rbf_outputs[j];
                }
            }
        }

        if (max_error < EPSILON) break;
    }
}
```

Listing 3: Функция обучения выходного слоя

5. Результаты экспериментов

5.1. Методика тестирования

Для каждого класса и каждого уровня шума (10%, 20%, 30%, 40%, 50%) было сгенерировано по 3 тестовых образа. Всего: $5 \times 5 \times 3 = 75$ тестов.

5.2. Примеры результатов распознавания

Для каждого тестового образа программа выводит:

- Распознаваемый зашумленный образ (6×6)
- Процент подобия по отношению к каждому из 5 классов
- Количество шагов, затраченных на обучение сети

Пример вывода:

```
Распознаваемый образ (6×6):
  ■ □ □ □ □ ■
  ■ ■ □ □ □ ■
  ■ □ ■ □ □ ■
  ■ □ □ ■ □ ■
  ■ □ □ □ ■ ■
  ■ □ □ □ □ ■

Процент подобия (выход РБФ):
  Класс 1 (N): 85.2%  ◄── Распознан как "N"
  Класс 2 (F): 3.1%
  Класс 3 (I): 5.4%
  Класс 4 (P): 4.8%
  Класс 5 (D): 1.5%

Шагов обучения: 87
```

5.3. Анализ результатов

Сеть РБФ успешно распознает зашумленные образы, показывая процент подобия для каждого класса. Класс с максимальным процентом подобия является результатом распознавания.

6. Сравнение с многослойным персептроном

Таблица 1: Сравнение MLP и RBF сетей

Характеристика	**Многослойный персептрон**	**Сеть РБФ**
Скорость обучения	Медленная	Быстрая
Обучаемые слои	2 (скрытый + выходной)	1 (только выходной)
Точность	Высокая	Ограниченная
Универсальность	Широкий класс задач	Кластеризуемые классы
Параметры	Веса, пороги, α	Центры, σ, веса

7. Выводы

1. **Успешная реализация:** Сеть РБФ реализована и обучена на 5 классах образов (N, F, I, P, D) размером 6×6.
2. **Быстрое обучение:** Обучение выходного слоя завершается за относительно небольшое количество шагов (обычно менее 100 итераций), что значительно быстрее многослойного персептрона.
3. **Распознавание зашумленных образов:** Сеть способна распознавать зашумленные образы, показывая процент подобия для каждого класса.
4. **Процент подобия:** Выход сети показывает степень соответствия входного образа каждому классу, что позволяет оценить уверенность распознавания.
5. **Преимущества:**
   - Высокая скорость обучения (обучается только выходной слой)
   - Простота реализации
   - Хорошая работа для хорошо кластеризуемых классов
6. **Недостатки:**
   - Ограниченная точность аппроксимации по сравнению с многослойным персептроном
   - Требуется правильный выбор центров и параметров разброса
   - Хорошо работает только для ограниченного класса аппроксимируемых функций
7. **Рекомендации:** Сеть РБФ целесообразно использовать в задачах классификации с хорошо кластеризуемыми классами, где важна скорость обучения.