МИНОБРНАУКИ РОССИИ
ФГБОУ ВО «СГУ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

ПРАКТИЧЕСКИЕ ЗАДАНИЯ ПО КУРСУ «НЕЙРОННЫЕ СЕТИ»

ОТЧЕТ О ПРАКТИКЕ

студента 5 курса 531 группы
направления 10.05.01 — Компьютерная безопасность
факультета КНиИТ
Токарева Никиты Сергеевича

Проверил
доцент _____ И. И. Слеповичев

Саратов 2023

1 Создание ориентированного графа

1.1 Описание задачи №1

На вход: текстовый файл с описанием графа в виде списка дуг:

$$(a_1, b_1, n_1), (a_2, b_2, n_2), \ldots (a_k, b_k, n_k),$$

где a_i – начальная вершина дуги i, b_i – конечная вершина дуги i ($a_i \neq b_i$), n_i – порядковый номер дуги в списке всех заходящих в вершину b_i дуг. Т.е. допустим в ориентированном графе будут заданы дуги, например (a_1, b_1) и (a_2, b_1), тогда в описании данного графа будут заданы дуги (a_1, b_1, n_1) и (a_2, b_1, n_2), где номера этих дуг упорядоченны и различны.

На выходе:

а) Ориентированный граф с именованными вершинами и линейно упорядоченными дугами (в соответствии с порядком из текстового файла). Структура графа должна записываться в файл формата XML или JSON.

б) Сообщение об ошибке в формате файла, если ошибка присутствует.

1.2 Примеры исполнения программы

Пример №1:

Рассмотрим ориентированный граф, который имеет следующую запись:

$$(v1, v2, 1), (v3, v4, 1), (v2, v5, 1), (v5, v6, 1), (v5, v7, 1),$$
$$(v4, v8, 1), (v4, v9, 1), (v9, v6, 2)$$

Этот граф можно представить в следующем виде, как показано на рисунке 1.

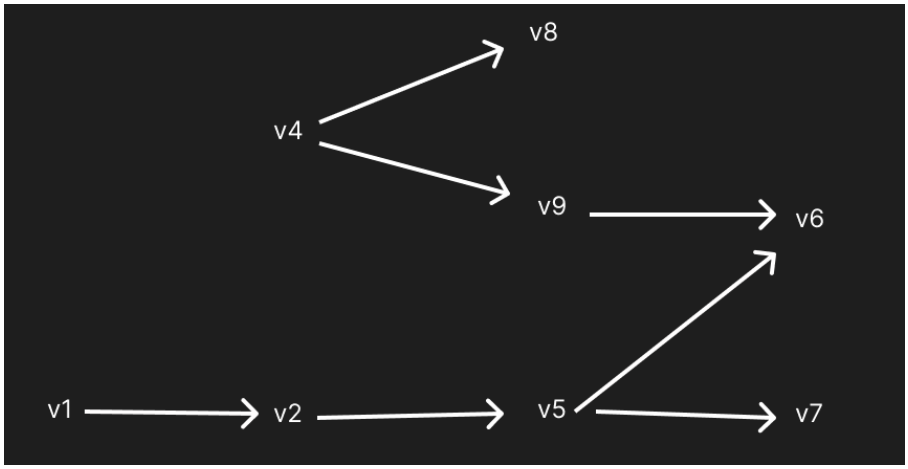

Рисунок 1 – Изображение ориентированного графа

Тогда после выполнении программы результатом будет являться следующая структура, которая будет записана в файл формата JSON.

```
{
    "graph":
    {
        "vertex":
        [
            "v1", "v2", "v3", "v4", "v5",
            "v6", "v7", "v8", "v9"
        ],
        "arc":
        [
            {"from": "v1", "to": "v2", "order": 1},
            {"from": "v3", "to": "v4", "order": 1},
            {"from": "v2", "to": "v5", "order": 1},
            {"from": "v5", "to": "v6", "order": 1},
            {"from": "v5", "to": "v7", "order": 1},
            {"from": "v4", "to": "v8", "order": 1},
            {"from": "v4", "to": "v9", "order": 1},
            {"from": "v9", "to": "v6", "order": 2}
        ]
    }
}
```

Пример 2:

Рассмотрим ориентированный граф, который имеет следующую запись:

$$(a, b, 1), (c, d, 1), (b, e, 1), (d, e, 1), (e, f, 1)$$

На рисунке 2 показано изображение данного графа.

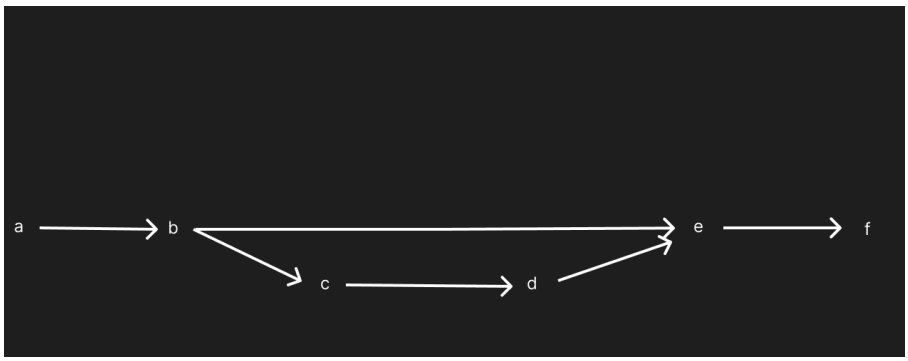

Рисунок 2 – Изображение ориентированного графа

Исходя из рисунка, можно заметить, что такой вариант графа вполне допустим, однако в описании дуг была допущена ошибка: дуги $(b, e, 1)$ и $(d, e, 1)$ имеют одинаковый порядковый номер. В таком случае программа выводит сообщение об ошибке, что в графе некорректно заданы номера.

2 Создание функции по графу

2.1 Описание задачи №2

На входе: ориентированный граф с именованными вершинами как описано в задании 1.

На выходе: линейное представление функции, реализуемой графом в префиксной скобочной записи:

$$A_1(B_1(C_1(\dots), \dots, C_m(\dots)), \dots, B_n(\dots))$$

Способ проверки резульата:

а) выгрузка в текстовый файл результата преобразования графа в имя функции.

б) сообщение о наличии циклов в графе, если они присутствуют.

2.2 Примеры исполнения программы

Пример 1:

Рассмотрим ориентированный граф, который имеет следующую запись:

$$(v1, v2, 1), (v3, v2, 2), (v2, v4, 1), (v4, v5, 1)$$

На рисунке 3 показано изображение данного графа.

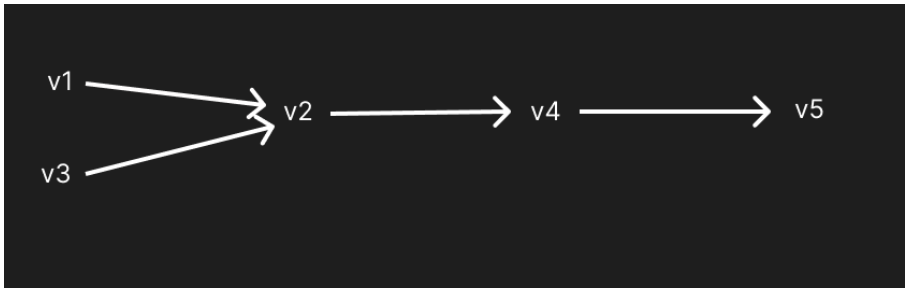

Рисунок 3 – Изображение ориентированного графа

В результате работы программы в файл будет сделана следующая запись:

$$v5(v4(v2(v1(), v3())))$$

Пример 2:

Рассмотрим ориентированный граф, который имеет следующую запись:

$$(v1, v2, 1), (v2, v3, 1), (v3, v1, 1)$$

На рисунке 4 показано изображение данного графа.

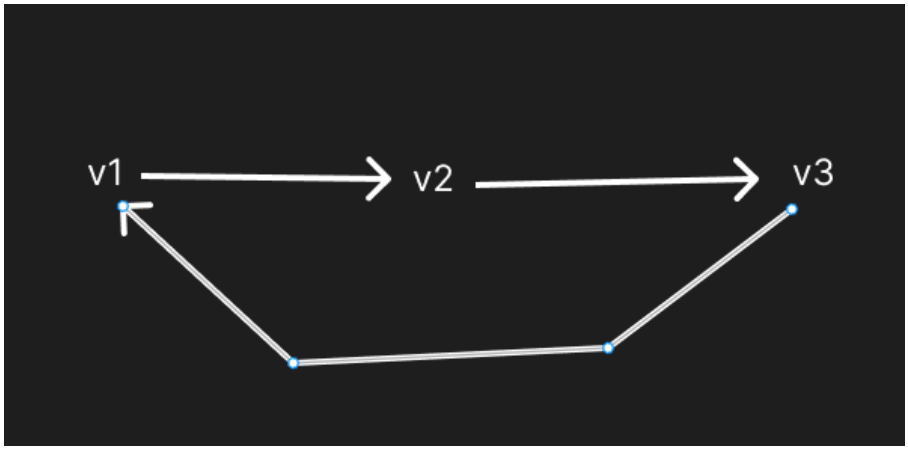

Рисунок 4 – Изображение ориентированного графа

Исходя из рисунка, видно, что в данном графе присутствует цикл. Поэтому результатом программы будет вывод об ошибке в консоль.

3 Вычисление значение функции на графе

3.1 Описание задачи №3

На входе:

а) Текстовый файл с описанием графа в виде списка дуг (смотри задание 1).

б) Текстовый файл соответствий арифметических операций именам вершин:

$$a_1 : 1\text{-я операция}$$
$$a_2 : 2\text{-я операция}$$
$$\dots$$
$$a_n : n\text{-я операция},$$

где a_i – имя i-й вершины, i-я операция – символ операции, соответствующий вершине a_i.

Допустимы следующие символы операций:

$+$ – сумма значений,

$*$ – произведение значений,

exp – экспонирование входного значения,

число – любая числовая константа.

На выходе: значение функции, построенной по графу а) и файлу б).

Способ проверки результата: результат вычисления, выведенный в файл.

3.2 Примеры исполнения программы

Пример 1:

Рассмотрим ориентированный граф, который имеет следующую запись:

$$(v1, v2, 1), (v1, v2, 2), (v2, v6, 1), (v3, v5, 1), (v4, v5, 2), (v6, v7, 1), (v5, v7, 2)$$

Также имеются соответствия арифметических операций именам вершинам, записанных в файле формата JSON:

```
{
    "v1" : 1,
    "v2" : "+",
    "v3" : 5,
    "v4" : 12,
```

```
        "v5" : "*",
        "v6" : "exp",
        "v7" : "+"
    }
```

На рисунке 5 показано изображение данного графа.

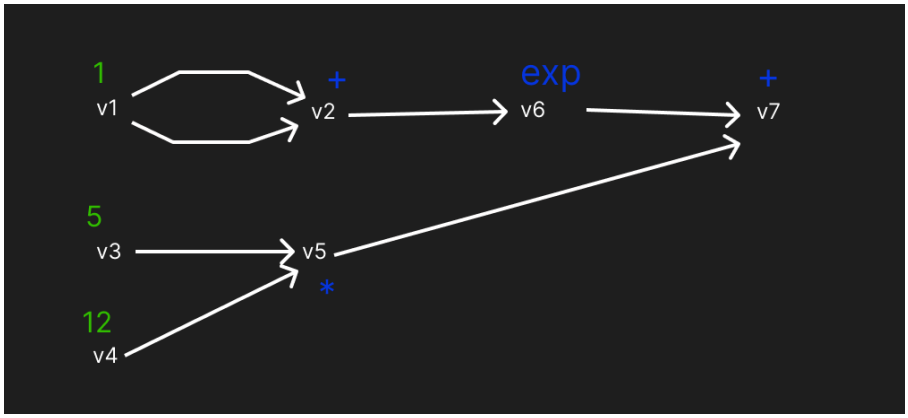

Рисунок 5 – Изображение ориентированного графа

После всех проверок входных данных на корректность, программа представляет ориентированный граф в следующем виде:

$$v7(v6(v2(v1(), v1())), v5(v3(), v4()))$$

Подставив в соответствие именам вершин арифметические операции получается следующая запись:

$$+(exp(+(1, 1)), *(5, 12))$$

Таким образом программе необходимо вычислить выражение, представленное в префиксной записи. В итоге программа записывает в текстовый файл результат данного выражения: 67.38905609893065.

Пример 2:

Теперь рассмотрим ориентированный граф, который имеет достаточную простую запись:

$$(v1, v3, 1), (v2, v3, 2)$$

Также имеются следующие соответствия:

```
{
    "v1" : 1,
    "v2" : "+",
    "v3" : 5
}
```

На рисунке 6 показано изображение данного графа.

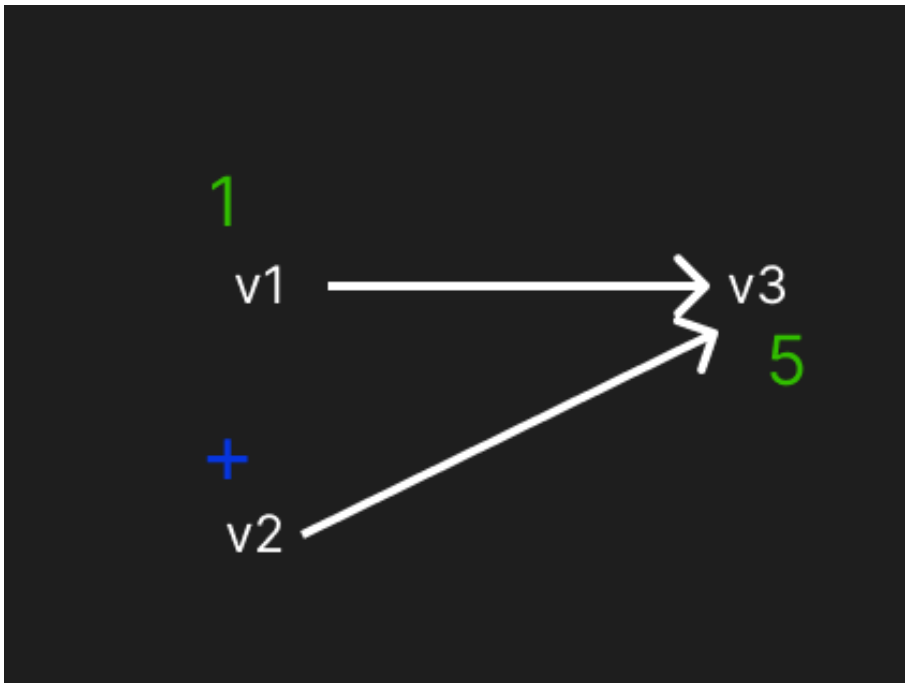

Рисунок 6 – Изображение ориентированного графа

Исходя из рисунка и текущих соответствий, можно заметить, что в данном случае невозможно будет вычислить результат. Поэтому программа выдает в качестве ответа в консоль сообщение об ошибке.

4 Построение многослойной нейронной сети

4.1 Описание задачи №4

На входе:

а) Файл с набором матриц весов межнейронных связей:

$$M_1 : [a^1_{11}, a^1_{12}, \ldots, a^1_{1n_1}], \ldots, [a^1_{m_1 1}, a^1_{m_1 2}, \ldots, a^1_{m_1 n_1}]$$
$$M_2 : [a^2_{11}, a^2_{12}, \ldots, a^2_{1n_2}], \ldots, [a^2_{m_2 1}, a^2_{m_2 2}, \ldots, a^2_{m_2 n_2}]$$
$$\ldots$$
$$M_p : [a^p_{11}, a^p_{12}, \ldots, a^p_{1n_p}], \ldots, [a^p_{m_p 1}, a^p_{m_p 2}, \ldots, a^p_{m_p n_p}]$$

б) Файл с входным вектором в формате:

$$x_1, x_2, \ldots, x_k.$$

На выходе:

а) Сериализованная многослойная нейронная сеть (в формате XML или JSON) с полносвязной межслойной структурой. Файл с выходным вектором – результатом вычислений НС в формате:

$$y_1, y_2, \ldots, y_k.$$

б) Сообщение об ошибке, если в формате входного вектора или файла описания НС допущена ошибка.

4.2 Примеры исполнения программы

Пример 1:

Пусть заданы следующие матрицы весов $M_1 = [[1], [2], [3]]$ размерности (3 x 1) и $M_2 = [[4, 5, 6]]$ размерности (1 x 3), записанные в файл формата JSON:

```
{
    "W" :
    [
        [
            [1], [2], [3]
        ],
        [
            [4, 5, 6]
        ]
```

```
    ]
}
```

Также в качестве входного вектора выступают значения $x = [[0.1], [0.2], [0.3], [0.4], [0.5]]$, которые записаны в файл формата JSON.

```
{
    "x" : [[0.1], [0.2], [0.3], [0.4], [0.5]]
}
```

В данной задаче, чтобы получить выходной вектор необходимо выполнять умножение каждого элемента входного вектора на матрицу весов M_i, после каждого такого умножения также нужно поэлементно применить функцию активации. В программе функцией активации по умолчанию является сигмоидальная функция:

$$f(x) = \frac{1}{1+e^{-x}}.$$

Так как в данном случае $|x| = 5$, то относительно каждого элемента вектора $x = [x_1, x_2, x_3, x_4, x_5]$ выполняются следующие действия:

1. Умножение матрицы весов M_1 на вектор x_i $(1 \leq i \leq 5)$: $out_1 = M_1 \cdot x_i$. Если умножение выполнить не удалось из-за того что два множителя несовместимы, то в консоль программа выводит сообщение об ошибке;
2. Поэлементое применение функции $f(x)$ к out_1, в результате чего получается выходной вектор y_1^i;
3. Для $k = 2 \ldots p$, где p – количество матриц весов, выполнять шаги 4-6.
4. Умножение матрицы весов M_k на вектор y_{k-1}^i $(1 \leq i \leq 5)$: $out_k = M_k \cdot x_i$. Если умножение выполнить не удалось из-за того что два множителя несовместимы, то в консоль программа выводит сообщение об ошибке;
5. Поэлементое применение функции $f(x)$ к out_k, в результате чего получается выходной вектор y_k^i;
6. Если $k = p$, то в качестве результата вернуть выходной вектор y_k^i, иначе увеличить k на единицу.

Стоит отметить, что для корректной работы программы во входных данных очень важно следить за тем, чтобы при умножении матрицы были совместимыми (число столбцов первого множителя равно числу строк второго множителя).

Так как в данном случае все условия для корректной работы программы выполняются, то получается следующий выходной вектор:

```
{
    "y":
    [
        [0.9997504849648627], [0.9998845678407059],
        [0.9999440920809696], [0.999971219155923],
        [0.9999841222683092]
    ]
}
```

Пример 2:

Пусть заданы следующие матрицы весов $M_1 = [[1], [2], [3]]$ размерности (3 x 1) и $M_2 = [[4, 5, 6, 7]]$ размерности (1 x 4), записанные в файл формата JSON:

```
{
    "W" :
    [
        [
            [1], [2], [3]
        ],
        [
            [4, 5, 6, 7]
        ]
    ]
}
```

Также в качестве входного вектора выступают значения $x = [[0.1], [0.2], [0.3], [0.4], [0.5]]$, которые записаны в файл формата JSON.

```
{
    "x" : [[0.1], [0.2], [0.3], [0.4], [0.5]]
}
```

В данном случае программа выдает ошибку во время попытки умножения матрицы M_2 на результирующую матрицу, полученную после умножения матрицы M_1 и применения функции активации $f(x)$. При умножении x_i на матрицу M_1, у результирующего вектора число строк стало равно числу строк матрицы M_1. Так как в данном случае у матрицы M_2 число столбцов не совпадает с числом строк результирующего вектора, то дальнейшее выполнение программы невозможно, и она выводит в консоль сообщение об ошибке.

5 Реализация метода обратного распространения ошибки для многослойной НС

5.1 Описание задачи №5

На входе:

а) Текстовый файл с описанием НС (формат см. в задании 4).

б) Текстовый файл с обучающей выборкой:

$$[x_1^1, x_2^1, \ldots, x_k^1] \to [y_1^1, y_2^1, \ldots, y_l^1]$$
$$\ldots$$
$$[x_1^n, x_2^n, \ldots, x_k^n] \to [y_1^n, y_2^n, \ldots, y_l^n]$$

Формат описания входного вектора x и выходного вектора y соответствует формату из задания 4.

в) Число итераций обучения (в строке параметров).

На выходе: Текстовый файл с историей N итераций обучения методом обратного распространения ошибки:

$$1 : \text{1-я ошибка}$$
$$2 : \text{2-я ошибка}$$
$$\ldots$$
$$N : \text{N-я ошибка}$$

5.2 Примеры исполнения программы

Пример:

Пусть заданы следующие матрицы весов $M_1 = [[1], [2], [3]]$ размерности (3 x 1) и $M_2 = [[4, 5, 6]]$ размерности (1 x 3), записанные в файл формата JSON:

```
{
    "W" :
    [
        [
            [1], [2], [3]
        ],
        [
            [4, 5, 6]
        ]
```

```
    ]
}
```

В другом JSON-файле записана обучающая выборка в следующем виде:

```
{
    "in" :
    [
        [0.1], [0.2], [0.3], [0.4]
    ],
    "out" :
    [
        [0.9], [0.8], [0.7], [0.6]
    ]
}
```

Обозначим обучающую выборку входных параметров как вектор x, а выборку выходных параметров – d. Также в отдельном файле записаны дополнительные параметры: количество итераций $m = 25$ и коэффициент скорости обучения ($0 < lrate < 1$) $lrate = 0.6$.

Обратное распространение ошибки состоит из следующих шагов:

1. Для $r = 1, \dots m$ выполнять шаги 2 - 6.
2. Прямое распространение сигнала. Входные сигналы передаются по сети от нейрона к нейрону. Каждый нейрон вычисляет свою активационную функцию от входных данных и передает результат следующему нейрону. При выполнении данного шага применяется алгоритм, реализованный в задаче №4. Однако стоит отметить, что дополнительно запоминаются промежуточные выходные векторы y_j^i ($0 \leq j \leq p$, $1 \leq i \leq k$, p – количество слоев (матриц весов), k – количество элементов в векторе x), полученные после каждого слоя. Стоит отметить, что вектор $y_0^i = x_i^s$ ($1 \leq s \leq n$), т.е. после выполнения прямого распространения сигнала должна получится последовательность векторов $y_0^i = x_i^s, y_1^i, \dots y_p^i$.
3. Вычисление ошибки. После прямого распространения сигнала вычисляется разница между полученными и ожидаемыми выходными значениями сети. Эта разница является ошибкой, которую необходимо минимизировать. В программной реализации i-я ошибка e вычисляется, как $e = y_p^i - d_i$.

4. Обратное распространение ошибки. Ошибка сети распространяется в обратном направлении, от выходных нейронов к входным. Каждый нейрон вычисляет градиент ошибки по отношению к его входным данным и весам связей. Градиент ошибки $grad$ относительно последнего слоя вычисляется по формуле $grad_p = e \cdot f^{'}(y_p^i)$.
5. Обновление весов. Используя градиент ошибки, веса связей между нейронами корректируются с помощью заданного обновления весового коэффициента. Целью является минимизация ошибки и улучшение результатов сети. Обновление весов происходит следующем образом:
   а) Обновление весов последнего p-го слоя M_p:
   $M_p = M_p - lrate \cdot y_{p-1}^i \cdot grad_p$;
   б) Для $t = p - 1 \ldots 1$ выполнять следующее: вычислить
   $grad_t = M_{t+1} \cdot grad_{t+1} \cdot f^{'}(y_t^i)$, затем выполнить обновление весов матрицы M_t: $M_t = M_t - y_{t-1}^i \cdot grad_t \cdot lrate$. Причем размерность матриц весов должна сохраняться, чтобы корректно посчитать последующие значения обучающей выборки.
6. Далее после прохождения по всем значениям x_i^s. ($1 \leq i \leq k$, $1 \leq s \leq n$) вычисляется среднее арифметическое значений ошибки $\mu = \frac{e_1 + \cdots + e_k}{k}$ и сохраняется в отдельный список. После чего количество итераций увеличивается на единицу.

Таким образом результатом работы программы является сообщение следующего вида, которое записывается в текстовый файл:

При i = 1 значения функции ошибок: 0.24965137935543213
При i = 2 значения функции ошибок: 0.24944708478343486
При i = 3 значения функции ошибок: 0.2494469476387328
При i = 4 значения функции ошибок: 0.24944681042609376
При i = 5 значения функции ошибок: 0.24944667314546723
При i = 6 значения функции ошибок: 0.2494465357968028
При i = 7 значения функции ошибок: 0.24944639838004992
При i = 8 значения функции ошибок: 0.24944626089515787
При i = 9 значения функции ошибок: 0.24944612334207614
При i = 10 значения функции ошибок: 0.24944598572075405
При i = 11 значения функции ошибок: 0.24944584803114084
При i = 12 значения функции ошибок: 0.2494457102731858

При i = 13 значения функции ошибок: 0.24944557244683785
При i = 14 значения функции ошибок: 0.24944543455204632
При i = 15 значения функции ошибок: 0.24944529658876027
При i = 16 значения функции ошибок: 0.24944515855692864
При i = 17 значения функции ошибок: 0.24944502045650035
При i = 18 значения функции ошибок: 0.2494448822874244
При i = 19 значения функции ошибок: 0.2494447440496496
При i = 20 значения функции ошибок: 0.24944460574312483
При i = 21 значения функции ошибок: 0.24944446736779877
При i = 22 значения функции ошибок: 0.2494443289236201
При i = 23 значения функции ошибок: 0.24944419041053753
При i = 24 значения функции ошибок: 0.24944405182849969
При i = 25 значения функции ошибок: 0.24944391317745518