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

ПРАКТИЧЕСКИЕ ЗАДАНИЯ ПО КУРСУ ”НЕЙРОННЫЕ СЕТИ”

ОТЧЕТ О ПРАКТИКЕ

студента 5 курса 531 группы
направления 100501 — Компьютерная безопасность
факультета КНиИТ
Улитина Ивана Владимировича

Проверил
доцент _____ И. И. Слеповичев

Саратов 2023

СОДЕРЖАНИЕ

1 Задание 1: Создание ориентированного графа

1.1 Описание

На входе: 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, n_i — порядковый номер дуги в списке всех заходящих в вершину b_i дуг.

На выходе: Ориентированный граф с именованными вершинами и линейно упорядоченными дугами (в соответствии с порядком из текстового файла). Сообщение об ошибке в формате файла, если ошибка присутствует.

1.2 Пример исполнения программы

Рассмотрим пример, созданный для программы в файле 'test1.txt', со следующим содержимым:

```
(A, D, 1), (A, D, 2), (B, E, 1), (C, E, 2), (D, G, 1), (E, F, 1),
↪ (F, G, 2)
```

Запускаем программу с помощью консоли следующим образом:

```
python task1.py input=tests\task1\test1.txt
↪ output=tests\task1\task1_res.xml
```

В качестве результата получаем файл 'task1_res.xml' с содержимым:

```
<graph>
<vertex>A</vertex>
<vertex>D</vertex>
<vertex>B</vertex>
<vertex>E</vertex>
<vertex>C</vertex>
<vertex>G</vertex>
<vertex>F</vertex>
<arc>
  <from>A</from>
  <to>D</to>
  <order>1</order>
</arc>
<arc>
```

```
    <from>A</from>
    <to>D</to>
    <order>2</order>
  </arc>
  <arc>
    <from>D</from>
    <to>G</to>
    <order>1</order>
  </arc>
  <arc>
    <from>B</from>
    <to>E</to>
    <order>1</order>
  </arc>
  <arc>
    <from>E</from>
    <to>F</to>
    <order>1</order>
  </arc>
  <arc>
    <from>C</from>
    <to>E</to>
    <order>2</order>
  </arc>
  <arc>
    <from>F</from>
    <to>G</to>
    <order>2</order>
  </arc>
</graph>
```

2 Задание 2: Создание функции по графу

2.1 Описание

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

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

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

2.2 Пример исполнения программы

Рассмотрим пример, созданный для программы в файле 'test2.xml', со следующим содержимым:

```
<graph>
<vertex>A</vertex>
<vertex>D</vertex>
<vertex>B</vertex>
<vertex>E</vertex>
<vertex>C</vertex>
<vertex>G</vertex>
<vertex>F</vertex>
<arc>
  <from>A</from>
  <to>D</to>
  <order>1</order>
</arc>
<arc>
  <from>A</from>
  <to>D</to>
  <order>2</order>
</arc>
<arc>
  <from>D</from>
  <to>G</to>
  <order>1</order>
</arc>
<arc>
  <from>B</from>
  <to>E</to>
```

```
    <order>1</order>
  </arc>
  <arc>
    <from>E</from>
    <to>F</to>
    <order>1</order>
  </arc>
  <arc>
    <from>C</from>
    <to>E</to>
    <order>2</order>
  </arc>
  <arc>
    <from>F</from>
    <to>G</to>
    <order>2</order>
  </arc>
</graph>
```

Запускаем программу с помощью консоли следующим образом:

```
python task2.py input=tests\task2\task2.xml
↪  output=tests\task2\task2_res.txt
```

В качестве результата получаем файл 'task2_res.xml' с содержимым:

```
G(D(A(), A()), F(E(B(), C())))
```

3 Задание 3: Вычисление значение функции на графе

3.1 Описание

На входе:

1. Текстовый файл с описанием графа в виде списка дуг (смотри задание 1).
2. 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 и файлу.

3.2 Пример исполнения программы

Рассмотрим пример, созданный для программы в файлах 'graph.txt' и 'operations.txt', со следующим содержимым:

graph.txt

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

operations.txt

```
{
    "v1" : 3,
    "v2" : 2,
    "v3" : 5,
    "v4" : "*",
    "v5" : "+",
    "v6" : "+",
    "v7" : "exp"
}
```

Запускаем программу с помощью консоли следующим образом:

```
python task3.py graph=tests\task3\graph.txt
↪  ops=tests\task3\operations.txt output=tests\task3\res.txt
```

В качестве результата получаем файл ’res.txt’ с содержимым:

```
442413.3920089205
```

4 Задание 4: Построение многослойной нейронной сети

4.1 Описание

На входе:

1. Файл с набором матриц весов межнейронных связей:

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

2. Файл с входным вектором в формате:

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

На выходе:

1. Сериализованная многослойная нейронная сеть с полносвязной межслойной структурой. Файл с выходным вектором – результатом вычислений НС в формате:

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

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

4.2 Пример исполнения программы

Рассмотрим пример, созданный для программы в файлах 'x4.txt' и 'w.txt', со следующим содержимым:

x4.txt

```
[2, 2, 8]
```

w.txt

```
[[[0.47519493033675375, 0.015705490366171526, 0.9433818257724572],
[0.48092032736144574, 0.13929695479782134, 0.6869903232566065],
[0.436988975888717, 0.20037642195993755, 0.1756140627552794 7]],
[[0.042224071742743785, 0.15331022315027187, 0.464635658411239],
[0.6000159964796773, 0.22606113281552231, 0.5301212736820182],
[0.19651133783303198, 0.7498835958139106, 0.2872155697845659 7]],
[[0.11837615025116721, 0.00927217999098906, 0.7504596929897048],
[0.5675946231090779, 0.9748635791740536, 0.30501309542663524],
[0.8574872089946126, 0.3047120321509168, 0.3376899733092712]]]
```

Запускаем программу с помощью консоли следующим образом:

```
task4.py x=tests\task4\x4.txt w=tests\task4\w.txt
↪  y=tests\task4\out.txt
```

В качестве результата получаем файл ’out.txt’ с содержимым:

```
[0.6599800423450157, 0.7982164099813447, 0.7427805995966905]
```

5 Задание 5: Реализация метода обратного распространения ошибки для многослойной НС

5.1 Описание

На входе:

1. Текстовый файл с описанием НС (формат см. в задании 4).
2. 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.
3. 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 Пример исполнения программы

Рассмотрим пример, созданный для программы в файлах 'x5.txt', 'w.txt' и 'y.txt', со следующим содержимым:

x5.txt

```
[
    [-4, 1, 5],
    [7, -1, -4],
    [4, 14, 10],
    [-8, -18, 6],
]
```

w.txt

```
[[[0.47519493033675375, 0.015705490366171526, 0.9433818257724572],
[0.48092032736144574, 0.13929695479782134, 0.6869903232566065],
[0.436988975888717, 0.20037642195993755, 0.1756140627552794]],
```

```
[[0.042224071742743785, 0.15331022315027187, 0.464635658411239],
[0.6000159964796773, 0.22606113281552231, 0.5301212736820182],
[0.19651133783303198, 0.7498835958139106, 0.2872155697845697]],
[[0.11837615025116721, 0.00927217999098906, 0.7504596929897048],
[0.5675946231090779, 0.9748635791740536, 0.30501309542663524],
[0.8574872089946126, 0.3047120321509168, 0.3376899733092712]]]
```

y.txt

```
[
    [0, 0, 0],
    [1, 1, 1],
    [1, 1, 1],
    [0, 0, 0],
]
```

Запускаем программу с помощью консоли следующим образом:

```
python task5.py x=tests\task5\x5.txt y=tests\task5\y.txt
↪  w=tests\task5\w.txt epochs=10 loss=tests\task5\results.txt
```

В качестве результата получаем файл ’result.txt’ с содержимым:

result.txt

```
Ошибка на эпохе 1 равна 0.2832965082336704
Ошибка на эпохе 2 равна 0.28011549588401924
Ошибка на эпохе 3 равна 0.27719508418488775
Ошибка на эпохе 4 равна 0.2745690438749986
Ошибка на эпохе 5 равна 0.2722617577309298
Ошибка на эпохе 6 равна 0.2702545573996806
Ошибка на эпохе 7 равна 0.26849787858212437
Ошибка на эпохе 8 равна 0.2669367574826795
Ошибка на эпохе 9 равна 0.265525581639051
```