Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №7

По дисциплине: “Языки программирования”

Тема: “Изучение NumPy. Сравнение производительности с классическими библиотеками Python”

Вариант №9

Выполнил: студент 2 курса группы ПО-7
Крупенков Михаил Дмитриевич

Проверила: Дряпко А. В.

Брест 2021

Постановка задачи

Задание 1

1. Для написания кода использовать библиотеки классического Python, NumPy и SciPy.
2. Код демонстрируется в Jupyter Notebook
3. По каждому заданию должно быть предоставлено не менее 3-х вариантов решения, среди которых:

- чистый NumPy (максимально оптимизованный, векторизованный)
- любой не векторизованный вариант
- любой другой вариант, желательно конкурентноспособный

4. Все варианты решения должны быть протестированы на скорость выполнения при помощи %timeit
5. Полученные результаты отразить в отчете и сделать выводы о производительности и комфорте использования NumPy в различных задачах.name:

Задание 1

Подсчитать произведение ненулевых элементов на диагонали прямоугольной матрицы.

Пример: x = np.array([[1, 0, 1], [2, 0, 2], [3, 0, 3], [4, 4, 4]])

```
py: 422 ns ± 52.4 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
np: 7.01 µs ± 85.1 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

Задание 2

Дана матрица x и два вектора одинаковой длины i и j. Построить вектор np.array([X[i[0], j[0]], X[i[1], j[1]], . . . , X[i[N-1], j[N-1]]]).
Пример:

x = [[9 4 2], [6 0 0], [9 9 3]]

i: [1 2 1]

j: [1 0 1]

```
py: 485 ns ± 3.72 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
np: 2.18 µs ± 40.9 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

Задание 3

Даны два вектора x и y. Проверить, задают ли они одно и то же мультимножество.

Пример: x = np.array([1,2, 2, 4]), y = np.array([4, 2, 1, 2])

```
py: 344 ns ± 3.81 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
np: 5.57 µs ± 476 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

Задание 4

Найти максимальный элемент в векторе x среди элементов, перед которыми стоит нулевой.

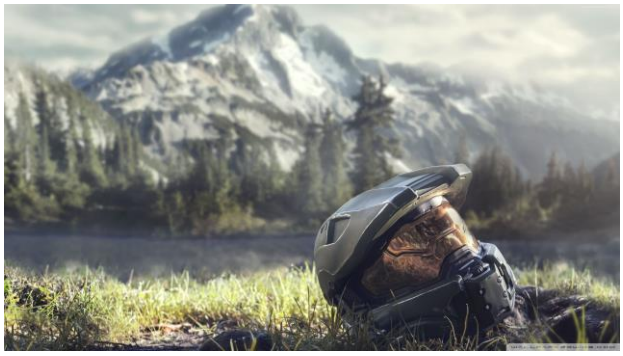
Пример: x = np.array([6, 2, 0, 3, 0, 0, 5, 7, 0])

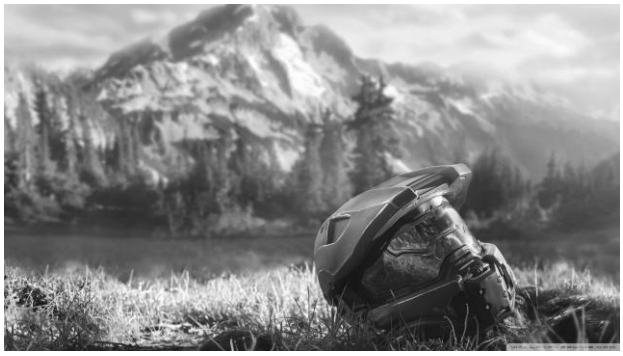
```
py: 637 ns ± 53.8 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
np: 3.03 µs ± 24.9 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

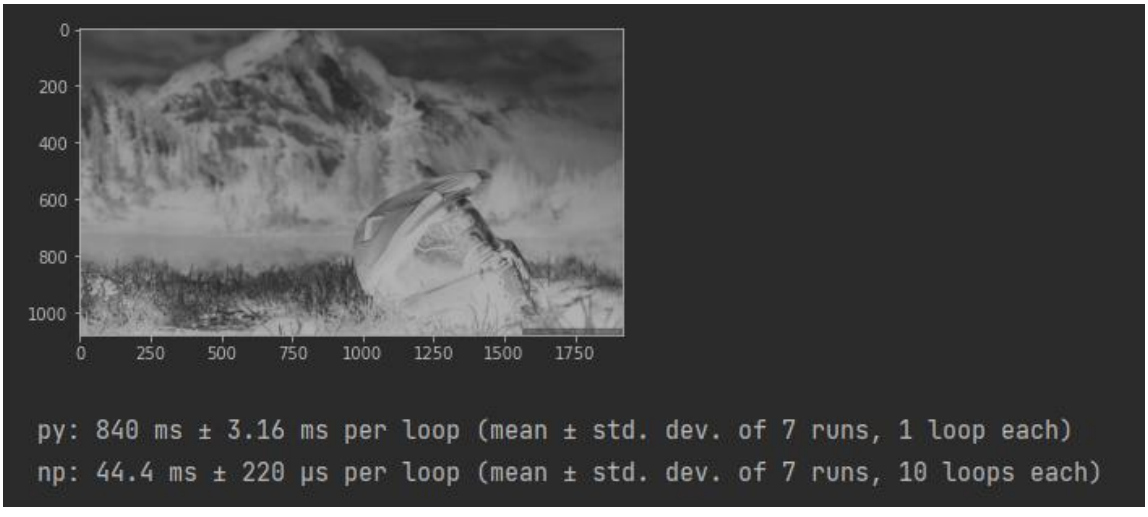
Задание 5

Дан трёхмерный массив, содержащий изображение, размера (height, width, numChannels), а также вектор длины numChannels. Сложить каналы изображения с указанными весами, и вернуть результат в виде матрицы размера (height, width). Считать реальное изображение можно при помощи функции scipy.misc.imread (если изображение не в формате png, установите пакет pillow: conda install pillow).
Преобразуйте цветное изображение в оттенки серого, использовав коэффициенты np.array([0.299, 0.587, 0.114]).

Пример:







Задание 6

Реализовать кодирование длин серий (Run-length encoding). Дан вектор x. Необходимо вернуть кортеж из двух векторов одинаковой длины. Первый содержит числа, а второй - сколько раз их нужно повторить.

Пример: x = np.array([2, 2, 2, 3, 3, 3, 5]).

Ответ: (np.array([2, 3, 5]), np.array([3, 3, 1])).

```
py: 1.85 µs ± 27 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)
np: 2.21 µs ± 30.1 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

Задание 7

Даны две выборки объектов - X и Y. Вычислить матрицу евклидовых расстояний между объектами.
Сравнить с функцией scipy.spatial.distance.euclidean.

Пример:

x: [2 7 6 6 9 6 3 4 9]

y: [1 0 0 7 2 2 4 3 0]

Ответ: 15.329709716755891

```
py:    2.07 µs ± 84.6 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
scipy: 9.8 µs ± 403 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
np:    4.44 µs ± 25.3 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

Задание 8

Реализовать функцию вычисления логарифма плотности многомерного нормального распределения. Входные параметры: точки X, размер (N, D), мат. ожидание m, вектор длины D, матрица ковариаций C, размер (D, D). Разрешается использовать библиотечные функции для подсчета определителя матрицы, а также обратной матрицы, в том числе в невекторизованном варианте. Сравнить с scipy.stats.multivariate_normal(m, C).logpdf(X) как по скорости работы, так и по точности вычислений

```
np:    195 µs ± 834 ns per loop (mean ± std. dev. of 7 runs, 10000 loops each)
scipy: 149 µs ± 2.9 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)
```

Вывод: я преисполнился