Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный университет”

Кафедра ИИТ

Лабораторная работа №7

По дисциплине: “ЯП”

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

Выполнил:

Студент 2-го курса

Группы ПО-7

Смушко О.Р.

Проверил:

Бойко Д.О.

Брест, 2021

**Вариант 8**

**Цель работы:** Изучение NumPy. Сравнение производительности с классическими библиотеками Python

**Общие требования**

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.
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.

**Задание №1**

Подсчитать произведение ненулевых элементов на диагонали прямоугольной матрицы.

Python: 273 ns ± 0.921 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)

NumPy: 4.65 µs ± 16.9 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]]]).

Python: 370 ns ± 1.59 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)

NumPy: 545 ns ± 6.94 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)

**Задание №3**

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

Python: 223 ns ± 0.87 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)

NumPy: 2.9 µs ± 3.76 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.

Python: 512 ns ± 1.47 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)

NumPy: 1.41 µs ± 3.75 ns per loop (mean ± std. dev. of 7 runs, 1000000 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]).

Python: 26.4 ms ± 82.8 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)

NumPy: 1.43 ms ± 9.06 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)

**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.

Python: 1.18 µs ± 9.95 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)

NumPy: 903 ns ± 6.95 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)

**Задание №7**

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

Python: 197 ns ± 0.464 ns per loop (mean ± std. dev. of 7 runs, 1000000 loops each)

NumPy: 2.27 µs ± 369 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)

Scipy: 3.06 µs ± 44.3 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.

NumPy: 84 µs ± 453 ns per loop (mean ± std. dev. of 7 runs, 10000 loops each)

Value NumPy = -5.054836210528194

Scipy: 51.8 µs ± 136 ns per loop (mean ± std. dev. of 7 runs, 10000 loops each)

Value Scipy = -5.054836210528194

**Вывод:** В ходе лабораторной работы сравнил производительность NumPy с классическими библиотеками Python. Таким образом, чем больше количество данных, тем заметнее преимущество NumPy перед классической реализацией на Python. При небольшом количестве данных преимущество наблюдается за классическим Python.