Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №10

за 3 семестр

По дисциплине: «Языки программирования»

Тема: «Изучение NumPy»

Выполнил:

Студент 2 курса

Группы ПО-6(1)

Мартынович Д. М.

Проверил:

Хацкевич М. В.

2021

Общие правила выполнения задания:

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:

* 1. чистый NumPy (максимально оптимизованный, векторизованный)
  2. любой не векторизованный вариант
  3. 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.

Список заданий:

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]])

Ответ:

**from** functools **import** reduce  
**import** numpy **as** np  
  
*# Решение с numpy*x = np.array([[1, 0, 1], [2, 0, 2], [3, 0, 3], [4, 4, 4]])  
%timeit np.prod(np.diag(x)[np.diag(x)!=0])  
  
*# Решения без numpy*x = [[1, 0, 1], [2, 0, 2], [3, 0, 3], [4, 4, 4]]  
%timeit reduce(**lambda** x, y: x\*y,[x[i][i] **for** i **in** range(len(x[0])) **if** x[i][i]!=0])  
  
x = [[1, 0, 1], [2, 0, 2], [3, 0, 3], [4, 4, 4]]  
**def** red(x):  
 ret = 1  
 **for** i **in** range(len(x[0])):  
 **if** x[i][i]!=0:  
 ret \*= x[i][i]  
 **return** ret  
%timeit red(x)  
print(red(x))

3.

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]

**import** numpy **as** np  
  
*# Решение с numpy*x = np.array([[9, 4, 2], [6, 0, 0], [9, 9, 3]])  
i = np.array([1, 2, 1])  
j = np.array([1, 0, 1])  
%timeit x[i, j]  
  
*# Решения без numpy*x = [[9, 4, 2], [6, 0, 0], [9, 9, 3]]  
i = [1, 2, 1]  
j = [1, 0, 1]  
%timeit [x[i[k]][j[k]] **for** k **in** range(len(i))]  
  
ret = []  
%timeit **for** k **in** range(len(i)): ret.append(x[i[k]][j[k]])  
print([x[i[k]][j[k]] **for** k **in** range(len(i))])

Ответ: [0 9 0]

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])

Ответ:

**import** numpy **as** np  
  
*# Решение с numpy*x = np.array([1, 2, 2, 4])  
y = np.array([4, 2, 1, 2])  
%timeit np.array\_equal(np.sort(x),np.sort(y))  
  
*# Решения без numpy*x = [1, 2, 2, 4]  
y = [4, 2, 1, 2]  
%timeit x.sort()==y.sort()  
  
**def** true(x, y):  
 ret = []  
 **for** i **in** range(len(x)):  
 **for** j **in** range(len(y)):  
 **if** x[i] == y[j]:  
 ret.append(**True**)  
 **break  
 elif** j+1 == len(y):  
 ret.append(**False**)  
 **return** ret   
%timeit all(true(x,y))

True.

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])

Ответ:

**import** numpy **as** np  
  
*# Решение с numpy*x = np.array([6, 2, 0, 3, 0, 0, 5, 7, 0])  
%timeit np.max(x[1:][(x==0)[:-1]])  
  
*# Решения без numpy*x = [6, 2, 0, 3, 0, 0, 5, 7, 0]  
%timeit max(x[i] **for** i **in** range(1, len(x)) **if not** x[i-1])  
  
**def** fun(x):  
 ret = []  
 **for** i **in** range(1, len(x)):  
 **if not** x[i-1]:  
 ret.append(x[i])  
 **return** ret  
%timeit max(fun(x))

5.

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:

Изображение выглядит как внешний, трава, гора

Автоматически созданное описание

Ответ:

Изображение выглядит как внешний, трава, поле

Автоматически созданное описание

**import** numpy **as** np  
**import** imageio  
**from** PIL **import** Image  
  
img = Image.open(**'![](../Desktop/halo.png)'**)  
  
*# Решение с numpy*color = np.array([0.2989, 0.587, 0.114])  
**def** fun(img, color):  
 arr = np.asarray(img, dtype=**'uint8'**)  
 new\_img = Image.fromarray(np.dot(arr[...,:3], color))  
 **return** new\_img  
  
%timeit fun(img, color)  
  
*# Решение без numpy*%timeit img.convert(**"L"**)  
color = [0.2989, 0.587, 0.114]  
img = imageio.imread(**'![](../Desktop/halo.png)'**)  
img = img.astype(**'float64'**)  
%timeit **for** i **in** range(len(img)): img[i] += color  
imageio.imwrite(**'![](../Desktop/halo.png)-gray.jpg'**, img[:,:,0])

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]).

**import** numpy **as** np  
**from** itertools **import** groupby  
  
*# Решение с numpy*x = np.array([2, 2, 2, 3, 3, 3, 5])  
**def** fun(seq):  
 ret = np.append(np.where(seq[1:] != seq[:-1]), len(seq) - 1)  
 count = np.diff(np.append(-1, ret))  
 **return** (ret,count)  
%timeit fun(x)  
  
x = [2, 2, 2, 3, 3, 3, 5]  
  
*# Решения без numpy***def** fun1(seq):  
 ret = ([seq[0]],[])  
 count = 1  
 j = 0  
 **for** i **in** range(1,len(seq)):  
 **if** seq[i] == ret[0][j]:  
 count += 1  
 **else** :  
 ret[0].append(seq[i])  
 ret[1].append(count)  
 j += 1  
 count = 1  
 ret[1].append(count)  
 **return** ret  
%timeit fun1(x)  
  
**def** fun2(seq):  
 ret = ([],[])  
 **for** x, group **in** groupby(seq):  
 ret[0].append(x)  
 ret[1].append(len(tuple(group)))  
 **return** ret  
%timeit fun2(x)

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

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:

**import** numpy **as** np  
**from** scipy.spatial **import** distance  
**import** math  
  
*# Решение с numpy*x = np.array([2, 7, 6, 6, 9, 6, 3, 4, 9])  
y = np.array([1, 0, 0, 7, 2, 2, 4, 3, 0])  
%timeit np.sqrt(np.dot(x-y, x-y))  
  
x = [2, 7, 6, 6, 9, 6, 3, 4, 9]  
y = [1, 0, 0, 7, 2, 2, 4, 3, 0]  
*# Решение с scipy*%timeit distance.euclidean(x, y)  
*# Решение с чистым python*%timeit math.dist(x, y)  
print(math.dist(x, y))

15.329709716755891

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

**import** numpy **as** np  
**import** scipy.sparse **as** sp  
**import** scipy.sparse.linalg **as** spln  
**from** scipy.stats **import** multivariate\_normal  
*# covariance matrix*sigma = np.array([[2.3, 0, 0, 0],  
 [0, 1.5, 0, 0],  
 [0, 0, 1.7, 0],  
 [0, 0, 0, 2]  
 ])  
*# mean vector*mu = np.array([2,3,8,10])  
  
*# input*x = np.array([2.1,3.5,8, 9.5])  
  
**def** lognormpdf(x,mu,S):  
 nx = len(S)  
 norm\_coeff = nx\*np.log(2\*np.pi)+np.linalg.slogdet(S)[1]  
 err = x-mu  
 numerator = spln.spsolve(S, err).T.dot(err)  
 **return** -0.5\*(norm\_coeff+numerator)  
%timeit lognormpdf(x, mu, sigma)  
print (lognormpdf(x, mu, sigma))  
%timeit multivariate\_normal(mu, sigma).logpdf(x)  
print (multivariate\_normal(mu, sigma).logpdf(x))