Язык Python. Часть 2

Лекция 13

Склеивание ndarray

- Есть картинка с цветовыми каналами
- Есть другая с такой же шириной
- Хотим их склеить в высоту
- Или с такой же высотой
- И хотим склеить в ширину

Склеивание ndarray

Склеивание ndarray

- Нужны функции или методы
- С разными параметрами
- concatenate
- Основной (и единственный обязательный) параметр - кортеж ndarray
- По умолчанию склеиваем по 0-й размерности

Склеивание ndarray

- Если двумерные, то по умолчанию "ставим друг на друга"
- Ширина должна совпадать
- Высота результата будет суммой высот
- shape-ы (a, x) и (b, x) дадут (a + b, x)

Склеивание ndarray

- Можем захотеть "поставить слева направо"
- Тогда высота должна совпадать
- Нужно указать параметр axis=1
- shape-ы (x, a) и (x, b) дадут (x, a + b)

Склеивание ndarray

- Если у нас картинка с цветовыми каналами
- Высота, ширина, значения каналов
- Можем склеить по нулевому измерению (по умолчанию)
- Ставим одну на другую картинки одного формата и одной ширины

Склеивание ndarray

- Можем склеить по первому измерению (axis=1)
- Ставим слева направо картинки одного формата и одной высоты
- Можем склеить по второму (axis=2)
- Добавляем к картинке новую информацию

Склеивание ndarray

- Например, альфа-канал (степень прозрачности)
- Новая информация может быть одним числом на пиксель
- То есть быть двумерной
- Тогда добавляем размерность

Пример

```
1 import numpy as np
2
3 a = np.array([[1, 2], [3, 4]])
4 b = np.array([[5, 6]])
5 print(a.shape, b.shape)
6 c = np.concatenate((a, b), axis=0)
7 print(c.shape)
8 print(c)
```

Пример

```
1 import numpy as np
2
3 a = np.array([[1, 2], [3, 4]])
4 b = np.array([[5, 6]])
5 print(a.shape, b.shape)
6
7 d = np.concatenate((a, b.T), axis=1)
8 print(d.shape)
9 print(d)
```

Пример

```
1 import numpy as np
2
3 a = np.arange(36).reshape((2, 3, 6))
4 b = np.arange(60).reshape((2, 5, 6)) + 100
5 print(a.shape, b.shape)
6
7 d = np.concatenate((a, b), axis=1)
8 print(d.shape)
9 print(d)
```

Пример

```
1 import numpy as np
2
3 a = np.arange(36).reshape((2, 3, 6))
4 b = np.arange(18).reshape((2, 3, 3)) + 100
5 print(a.shape, b.shape)
6
7 d = np.concatenate((a, b), axis=2)
8 print(d.shape)
9 print(d)
```

Пример

```
1 import numpy as np
2
3 a = np.arange(10)
4 b = np.arange(5) + 100
5 c = np.arange(7) + 200
6 print(a.shape, b.shape, c.shape)
7 d = np.concatenate((a, b, c))
8 print(d.shape)
9 print(d)
```

Склеивание ndarray

- axis может быть None
- Это НЕ значение по умолчанию
- Все параметры вытягиваются в вектора
- И склеиваются
- Результат - одномерный вектор

Пример

```
1 import numpy as np
2
3 a = np.array([[1, 2], [3, 4]])
4 b = np.array([[5, 6]])
5 print(a.shape, b.shape)
6
7 d = np.concatenate((a, b), axis=None)
8 print(d.shape)
9 print(d)
```

stack

- Есть несколько картинок
- Цветовые каналы
- Размеры картинок одинаковые
- Хотим получить массив картинок

stack

- То есть новое измерение
- Есть функция stack
- Принимает последовательность ndarray
- Одинакового размера

stack

- Есть и axis
- Номер новой размерности
- По умолчанию - 0
- Например, есть 5 картинок
- shape каждой - (600, 400, 3)
- Получим shape (5, 600, 400, 3)

stack

- Могли получить по отдельности значения по трем каналам
- В одной картинке
- Три ndarray
- shape каждого - (600, 400)
- Хотим получить shape (600, 400, 3)

stack

- Применим параметр axis
- В данном случае - 2
- И передадим список/кортеж из трех ndarray
- По каждому каналу

Пример

```
1 import numpy as np
2
3 a = np.arange(24).reshape(2, 3, 4)
4
5 d = np.stack((a, a + 10, a + 20))
6 print(d.shape)
7 print(d)
```

Пример

```
1 import numpy as np
2
3 a = np.arange(24).reshape(4, 6)
4
5 d = np.stack((a, a + 10, a + 20), axis=2)
6 print(d.shape)
7 print(d)
```

block

- Возьмем двумерную матрицу
- Проведем линии между двумя строками
- И между двумя столбцами
- Матрица поделится на 4 части

block

- Большая матрица - это как пазл
- Из четырех блоков
- Это может быть верным в самом буквальном смысле
- Допустим, есть монохромные изображения
- Один цветовой канал

block

- Первое изображение - (600, 400)
- Второе изображение - (600, 300)
- Третье изображение - (200, 400)
- Четвертое изображение - (200, 300)

block

- Можем склеить (через concatenate) первое со вторым
- И третье с четвертым
- Получим две картинки (600, 700) и (200, 700)
- Можем склеить и их
- И получить (800, 700)

block

- Но есть специальная функция - block
- Не обязачельно для четырех ndarray
- Она принимает параметр, похожий на параметр np.array
- Только в качестве элемента - ndarray

block

- Сам параметр-список может быть произвольной конфигурации
- И элементом может быть ndarray любой размерности
- Важно только, чтобы алгоритм склейки отработал
- И не сломался

block

- Начинаем с внутреннего уровня
- И все элементы склеиваем через concatenate и axis=-1
- Потом - -2 (предпоследнее измерение)
- И так далее, по количеству измерений параметра

Пример

```
1 import numpy as np
2
3 data1 = np.arange(6).reshape((2, 3))
4 data2 = np.arange(10).reshape((2, 5)) + 10
5
6 d = np.block([data1, data2])
7 print(d)
```

Пример

```
 1 import numpy as np
 2
 3 data1 = np.arange(6).reshape((2, 3))
 4 data2 = np.arange(14).reshape((2, 7)) + 10
 5 data3 = np.arange(9).reshape((3, 3))
 6 data4 = np.arange(6).reshape((3, 2)) + 100
 7 data5 = np.arange(15).reshape((3, 5)) + 1000
 8
 9 d = np.block([[data1, data2], [data3, data4, data5]])
10 print(d)
```

vstack

- Принимает последовательность ndarray
- Если размерность - 1, то увеличиваем за счет 1 в начале
- После этого - concatenate
- По нулевому измерению

Пример

```
1 import numpy as np
2
3 a = np.array([1, 2, 3])
4 b = np.array([4, 5, 6])
5 print(a.shape, b.shape)
6
7 data = np.vstack((a, b))
8 print(data.shape)
9 print(data)
```

Пример

```
1 import numpy as np
2
3 a = np.array([[1], [2], [3]])
4 b = np.array([[4], [5], [6]])
5 print(a.shape, b.shape)
6
7 data = np.vstack((a, b))
8 print(data.shape)
9 print(data)
```

hstack

- Принимает последовательность ndarray
- Если размерность - 1, concatenate по нулевому измерению
- Оно же - единственное
- Иначе - по первому

Пример

```
1 import numpy as np
2
3 a = np.array([1, 2, 3])
4 b = np.array([4, 5, 6])
5 print(a.shape, b.shape)
6
7 data = np.hstack((a, b))
8 print(data.shape)
9 print(data)
```

Пример

```
1 import numpy as np
2
3 a = np.array([[1], [2], [3]])
4 b = np.array([[4], [5], [6]])
5 print(a.shape, b.shape)
6
7 data = np.hstack((a, b))
8 print(data.shape)
9 print(data)
```

dstack

- depth wise stack
- Не по самому глубокому
- А по второму
- Считая с нуля

dstack

- Двумерную размерность (m, n) обрабатываем как (m, n, 1)
- Как stack по третьеиу измерению
- Одномерную размерность (n,) обрабатываем как (1, n, 1)
- Добавили пару скобок, чтобы сделать двумерным и дальше - stack по третьему измерению

Пример

```
1 import numpy as np
2
3 a = np.array([1, 2, 3])
4 b = np.array([4, 5, 6])
5 print(a.shape, b.shape)
6
7 data = np.dstack((a, b))
8 print(data.shape)
9 print(data)
```

Пример

```
1 import numpy as np
2
3 a = np.array([[1], [2], [3]])
4 b = np.array([[4], [5], [6]])
5 print(a.shape, b.shape)
6
7 data = np.dstack((a, b))
8 print(data.shape)
9 print(data)
```

Пример

```
1 import numpy as np
2
3 a = np.arange(24).reshape((2, 3, 4))
4 b = np.arange(30).reshape((2, 3, 5))
5 print(a.shape, b.shape)
6
7 data = np.dstack((a, b))
8 print(data.shape)
9 print(data)
```

column_stack, row_stack

- Принимает последовательность ndarray
- Одномерных или двумерных
- Все должны иметь одинаковую нулевую размерность
- Одномерные воспринимаем как колонки ("ставим вертикально")
- Дальше - как hstack
- row_stack - фактически, другое название для vstack

Пример

```
1 import numpy as np
2
3 a = np.array((1,2,3))
4 b = np.array((2,3,4))
5 print(a.shape, b.shape)
6
7 data = np.column_stack((a, b))
8 print(data.shape)
9 print(data)
```

Пример

```
1 import numpy as np
2
3 a = np.array((1,2,3))
4 b = np.array([(2,3), (4, 5), (5, 6)])
5 print(a.shape, b.shape)
6
7 data = np.column_stack((a, b))
8 print(data.shape)
9 print(data)
```

Обратные к stack-функциям

- stack-функции "ставят рядом" несколько структур
- И объединяют в одну
- А есть обратные
- Они - распиливают на несколько

split

- Первый параметр - всегда ndarray
- Второй может быть целым числом
- Или одномерный список отсортированных целых чисел
- Третий параметр - axis (размерность)

split

- Если число - задает, на сколько частей пилим
- Должно пилиться на равные части
- По указанному измерению
- Если не пилится, получим исключение