Understanding Python, NumPy, and Tensors

Further Considerations

Alexander G. Ororbia II

COGS-621: Foundations of Scientific Computing

9/4/2025

```
import numpy as np
```

So, what are tensors?

- A building block of linear algebra
- A mathematical formalism for a multi-dimensional collection / object – houses items, notably numbers / values
  - In Python, there are often called arrays or n-dimensional arrays (*ndarrays*)

```
import numpy as np
```

So, what are tensors?

- A building block of linear algebra
- A mathematical formalism for a m… onal collection / object – houses items, not… s
  - In Python, there ar… nal arrays (*ndarrays*)

Let's revisit numbers and objects/collections of them.

Scalars

- A scalar is a single number (think of it as an “atomic” object)
- *Examples*: Integers, real numbers, rational numbers, etc.
- Denoted with italic font:

$$a, n, x$$

On number spaces / domains:

$\mathbb{R}\,\mathcal{R}$	All real (continuous) numbers
$\mathbb{Z}\,\mathcal{Z}$	All integers
$\mathbb{N}$	All natural (counting) numbers

Vectors

- An array of numbers arranged in order
- Each no. identified by an index
- Written in lower-case bold such as $\boldsymbol{x}$
  - its elements are in italics lower case, subscripted

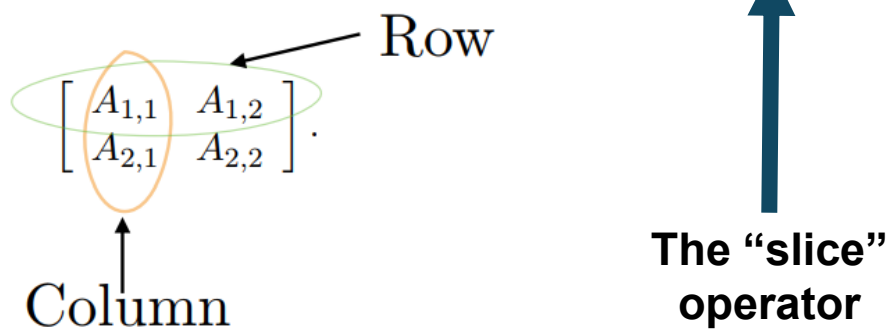
$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \qquad \mathcal{R}^n = \mathcal{R}^{n \times 1}$$

"All real numbers"

- If each element is in R then $\boldsymbol{x}$ is in R^n
- We can think of vectors as points in space
  - Each element gives coordinate along an axis

Matrices

- 2-D array of numbers
  - So each element identified by two indices
- Denoted by bold typeface $\boldsymbol{A}$
  - Elements indicated by name in italic but not bold
    - $A_{1,1}$ is the top left entry and $A_{m,n}$ is the bottom right entry
    - We can identify nos in vertical column j by writing : for the horizontal coordinate

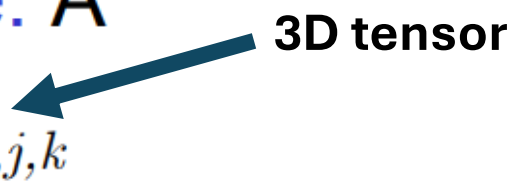


    - $A_{i:}$ is i^{th} row of A, $A_{:j}$ is j^{th} column of $\boldsymbol{A}$
- If A has shape of height m and width n with real-values then $\boldsymbol{A} \in \mathbb{R}^{m \times n}$

Tensors

- Sometimes need an array with more than two axes
  - E.g., an RGB color image has three axes
- A tensor is an array of numbers arranged on a regular grid with variable number of axes

- Denote a tensor with this bold typeface: $\mathbf{A}$
- Element (i,j,k) of tensor denoted by $\mathbf{A}_{i,j,k}$

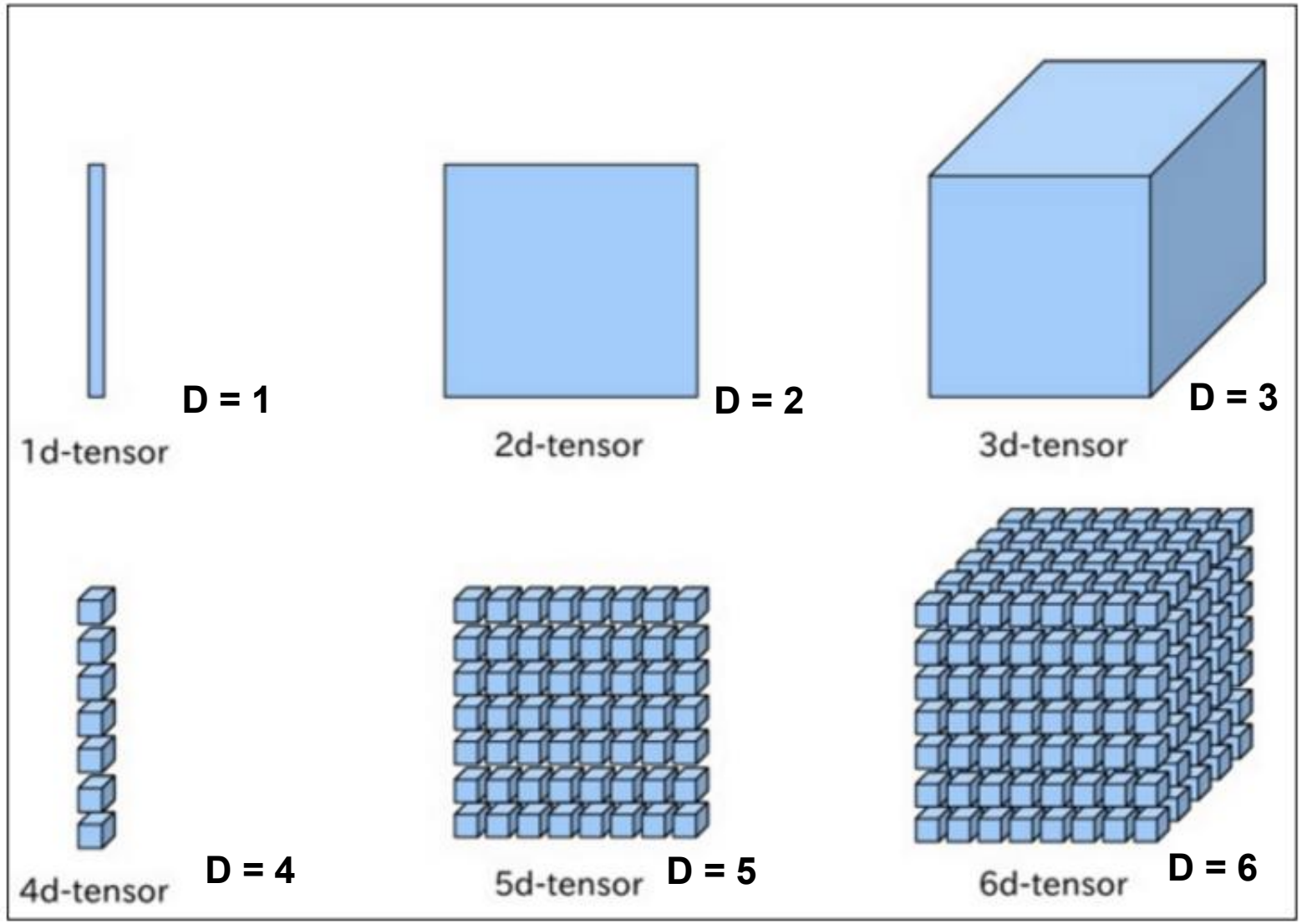


Types of notation accepted (just be consistent & mean what you write):

$$\mathcal{R}^{1\times1\times1} = \mathcal{R}^{1\times1} = \mathcal{R}^{1} = \mathcal{R}$$

Shapes of Tensors

Note: I often use *D* to denote dimensionality



The ndarray (the Python tensor)

Table 2-1. Basic Attributes of the ndarray Class

Attribute	Description
`shape`	A tuple that contains the number of elements (i.e., the length) for each dimension (axis) of the array
`size`	The total number of elements in the array
`ndim`	The number of dimensions (axes)
`nbytes`	The number of bytes used to store the data
`dtype`	The data type of the elements in the array

- The ndarray (object) contains a lot of useful metadata that you can access
  - Example: ndarray.shape → returns a tuple/pair with dimension values
- Array ordering – row (“C”) versus column (“F” for Fortran) major

When in doubt, read the manual for an object:

```
>>> data = np.array([[1., 2.], [3., 4.]], dtype=np.float32)
>>> help(data)

>>>
```

Data Types

Table 2-2. Basic Numerical Data Types Available in NumPy

dtype	Variants	Description
`int`	`int8, int16, int32, int64`	Integers
`uint`	`uint8, uint16, uint32, uint64`	Unsigned (nonnegative) integers
`bool`	`bool`	Boolean (True or False)
`float`	`float16, float32, float64, float128`	Floating-point numbers
`complex`	`complex64, complex128, complex256`	Complex-valued floating-point numbers

- You will primarily work with floating-point (like float32), integer (int32), and Boolean (bool) in this course
- Become comfortable with type-casting
- Become comfortable with type promotion

```
In [3]: type(data)
Out[3]: numpy.ndarray
In [4]: data
Out[4]: array([[1, 2],
               [3, 4],
               [5, 6]])
In [5]: data.ndim
Out[5]: 2
In [6]: data.shape
Out[6]: (3, 2)
In [7]: data.size
Out[7]: 6
In [8]: data.dtype
Out[8]: dtype('int64')
In [9]: data.nbytes
Out[9]: 48
```

Creating Different Kinds of (Initial) Arrays

- Useful ways to create (instantiate) ndarrays pre-filled with particular values

np.eye :
Creates diagonal array with values on main diagonal

Function Name	Type of Array
`np.array`	Create an array for which the elements are given by an array-like object, which, for example, can be a (nested) Python list, a tuple, an iterable sequence, or another `ndarray` instance.
`np.zeros`	Create an array with the specified dimensions and data type that is filled with zeros.
`np.ones`	Create an array with the specified dimensions and data type that is filled with ones.
`np.diag`	Create a diagonal array with specified values along the diagonal and zeros elsewhere.
`np.arange`	Create an array with evenly spaced values between the specified start, end, and increment values.
`np.linspace`	Create an array with evenly spaced values between specified start and end values, using a specified number of elements.
`np.logspace`	Create an array with values that are logarithmically spaced between the given start and end values.
`np.meshgrid`	Generate coordinate matrices (and higher-dimensional coordinate arrays) from one-dimensional coordinate vectors.
`np.fromfunction`	Create an array and fills it with values specified by a given function, which is evaluated for each combination of indices for the given array size.
`np.fromfile`	Create an array with the data from a binary (or text) file. NumPy also provides a corresponding function `np.tofile` with which NumPy arrays can be stored to disk and later read back using `np.fromfile`.
`np.genfromtxt`, `np.loadtxt`	Create an array from data read from a text file, for example, a comma-separated value (CSV) file. The `np.genfromtxt` function also supports data files with missing values.
`np.random.rand`	Generate an array with random numbers that are uniformly distributed between 0 and 1. Other types of distributions are also available in the `np.random` module.

Questions?