NumPy Cheat Sheet

EPFL CS 233
Introduction to Machine Learning
(Version 1)

Array initialization

Create arrays from (potentially nested) Python lists:

Rank 1	`a = np.array([1, 2, 3])`
Rank 2	`b = np.array([[1, 2, 3], [4, 5, 6]])`
Force a datatype	`c = np.array([0, 1, 2], dtype=np.int32)`

Functions to create standard arrays (e.g. all zeros):

Zero-filled array	`a = np.zeros((2,2))`
One-filled array	`b = np.ones((1,2))`
Random array	`d = np.random.rand((2,2))`
Identity matrix	`c = np.eye(2)`
Increasing sequence	`e = np.linspace(2.0, 8.0, 10)`

Note that the first three functions require a shape *tuple* argument, hence the double parentheses.

Data Types

Basic data types:

Unsigned 32 bit integer	`np.uint32`
Signed 64 bit integer	`np.int64`
Single precision floating point	`np.float32`
Double precision floating point	`np.float64`
Boolean	`np.bool`

Array indexing, slicing

Basic indexing notation:

Select the element at the 3rd index	`a[3]`
Select the element at row 2, column 0	`b[2][0]`

Slicing:

Select elements at index 0 and 1	`a[0:2]`
Select all elements in column 1	`b[:,1:2]`
Select first two rows and last two columns	`b[:2,-2:]`

Indexing using a list of indices:

Select elements (1,1) and (2,1)	`b[[1,2],[1,1]]`

Indexing using masking:

Select elements less than 4	`b[b<4]`

Standard arithmetic operations

Elementwise arithmetic operations

Addition	`d = e + f`
Subtraction	`d = e - f`
Multiplication	`d = e * f`
Division	`d = e / f`
Square root	`d = np.sqrt(e)`
Exponentiation	`d = np.exp(e)`
Natural logarithm	`d = np.log(e)`
Cosine	`d = np.cos(e)`

Other functions

Dot/Matrix product	`d = np.dot(e, f)` `d = e @ f`
Compute sum of each column	`d = np.sum(b, axis=0)`
Compute max value of each row	`d = np.max(b, axis=1)`
Compute min value of array	`d = np.min(b)`
Transpose matrix	`d = e.T`

Note that * and @ are different. The former does element-wise multiplication, while the latter is a dot or matrix-matrix/vector product depending on the input shapes.

Inspecting arrays

Basic definitions:

Array dimensions	`a.shape`
Number of array dimensions	`a.ndim`
Number of array elements	`a.size`
Number of array elements in a row	`a.shape[0]`
Data type of array elements	`a.dtype`
Cast an array to a different type	`a.astype(np.float32)`

Reshaping, Copying

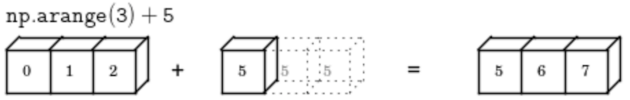
Reshape an array	`d = a.reshape(6, 1)`

Extend the dimensionality	`# shape (2, 3)`
	`a = np.array([[1, 2, 3], [4, 5, 6]])`
	`# shape (2, 1, 3)`
	`a[:, None, :]`

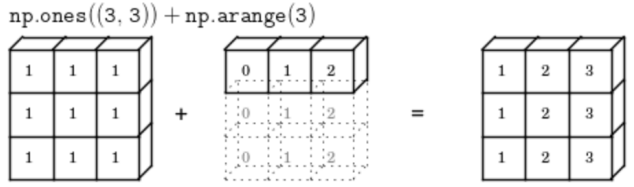
Flatten array to 1D	`a.flatten()` `a.reshape((-1, ))`
Create a deep copy of b	`c = np.copy(b)`

Broadcasting

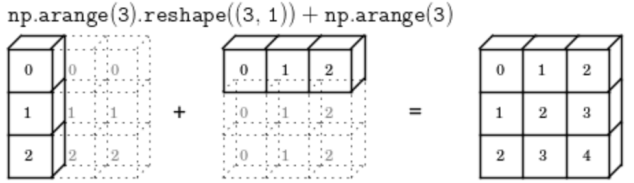
Broadcasting enables operations that combine arrays of different shapes.



A two dimensional array multiplied by a one dimensional array results in broadcasting if number of 1D array elements matches the number of 2D array columns.



Broadcasting can stretch both arrays to form an output array larger than either of the initial arrays.



Cheat-sheet by S. Arpa and J. Bednarik ([sami.arpa|jan.bednarik]@epfl.ch).
LaTeX template by Michelle Cristina de Sousa Baltazar.