Adopting the Python Array API Standard in NumPy's main namespace

Ralf Gommers & Aaron Meurer

3 April 2023

The proposal for NumPy 2.0

1. Full adoption of the array API standard in NumPy's main namespace
   a. Adding the missing functions,
   b. Making the backwards-incompatible changes where needed,
   c. Including in the `numpy.linalg` and `numpy.fft` submodules

2. Either keep `numpy.array_api` as a minimum reference implementation, or split it off (TBD depending on feedback)

Why now? What changed?

1. NEP 50 removes the main blocker (type promotion behavior / value-based casting) for why we didn't attempt this before. And NEP 50's changes are planned to become the default behavior in NumPy 2.0

2. Experience has shown that the separate array object in `numpy.array_api`, and wrapping to/from that object, is cumbersome.

Expected benefits

- Easier for CuPy/Dask/Numba/PyTorch/JAX/etc. to fully match or support NumPy
- Well-defined standard with more consistent behavior than NumPy itself has. Easier to teach, and organize docs around - functionality is orthogonal, hence overlapping NumPy functions can be de-emphasized.
- It will remove the “having to make a choice between the NumPy API and the Array API” issue for other libraries
- Some new features: `matrix_transpose`/`.mT`, `vecdot`, `matrix_norm`/`vector_norm` as gufuncs, namedtuple return values for linalg functions, new `svdvals` and `unique_*` functions that are easier to JIT, `copy` keyword for `reshape`, a new `isdtype` function, ...

Expected costs

- A number of (mostly minor) backwards-compatibility breaks (see next slides for details)
- Expanding the size of the main namespace with ~20 aliases

Note that https://numpy.org/devdocs/reference/array_api.html gives a very detailed per-object overview of changes.
"strictness" isn't relevant,
"compatible" is only about namespace size,
"breaking" is what we'll focus on here,

Breaking changes - raising errors

1. Making `.T` error for >2 dimensions
2. Making `cross` error on size-2 vectors
3. Making `solve` error on ambiguous input (only accept `x2` as vector if `x2.ndim == 1`
4. `outer` raises rather than flattens on >1-D inputs
5. In-place operators are disallowed when the left-hand side would be promoted.
6. Positional-only arguments for a number of non-ufunc functions (ufuncs already have positional-only arguments).

Breaking changes - return dtypes

1. `ceil`, `floor`, and `trunc` return an integer with integer input
2. `sum` and `prod` always upcast `float32` to `float64` when `dtype=None`.

Breaking changes - numerical behavior

1. The `rtol` default value for `pinv` changes

2. The definitions of `tol`/`rtol` and their default values change for `matrix_rank`. In addition, it does not support 1-D array input.

3. `argsort` and `sort` have a `stable` keyword argument instead of `kind`, with slightly different behavior

Breaking changes - other

1. The `diagonal` and `trace` functions that will be new in `numpy.linalg` operate on the last two rather than first two axes (for consistency, and to support stacking). Hence the linalg and main namespace functions of the same names will differ. Technically not breaking, but potential confusing.

    a. We could deprecate `np.trace` and `np.diagonal` to resolve it, but preferably not immediately.
    b. Side note: “stacking” is a confusing term, deep learning has won this one and the rest of the world calls this “batching”. Let’s change this terminology in NumPy?

Conclusion

- The benefits to array libraries and compilers that want to achieve compatibility with NumPy are significant
- The long-term benefits for the ecosystem as a whole - because of downstream libraries being able to support multiple array libraries as easily as possible - are significant too.
- The number of breaking changes needed is fairly limited, and the impact of those changes seems modest. Not painless, but a price worth paying.

Backup slides - background on the array API standard

Goals for and scope of the array API

Goal 1: enable writing code & packages that support multiple array libraries

Goal 2: make it easy for end users to switch between array libraries

In Scope

1. Syntax and semantics of functions and objects in the API
2. Casting rules, broadcasting, indexing, Python operator support
3. Data interchange & device support

Out of Scope

1. Execution semantics (e.g. task scheduling, parallelism, lazy eval)
2. Non-standard dtypes, masked arrays, I/O, subclassing array object, C API
3. Error handling & behaviour for invalid inputs to functions and methods

Array- and array-consuming libraries

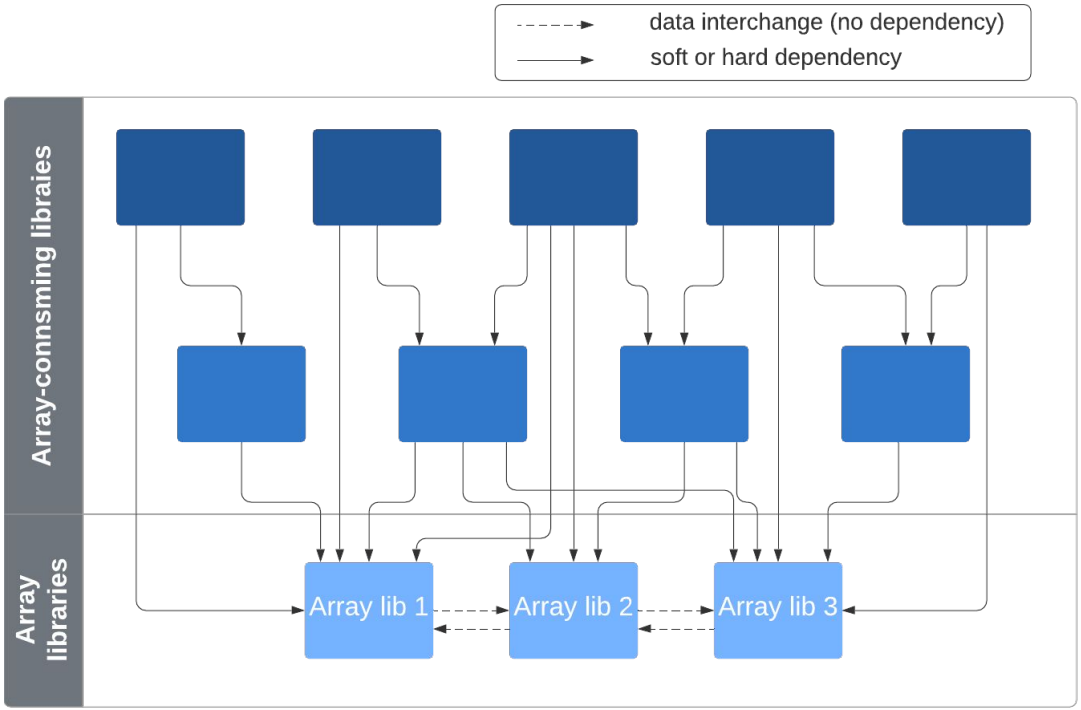
Data interchange between array libs

Using DLPack, will work for any two libraries if they support device the data resides on

```
x = xp.from_dlpack(x_other)
```

Portable code in array-consuming libs

```
def softmax(x):
    # grab standard namespace from
    # the passed-in array
    xp = get_array_api(x)
    x_exp = xp.exp(x)
    partition = xp.sum(x_exp, axis=1,
                       keepdims=True)
    return x_exp / partition
```



Use cases

Use cases inform the requirements for, and design choices made in, this array API standard. This section first discusses what types of use cases are considered, and then works out a few concrete use cases in more detail.

Types of use cases

- Packages that depend on a specific array library currently, and would like to support multiple of them (e.g. for GPU or distributed array support, for improved performance, or for reaching a wider user base).
- Writing new libraries/tools that wrap multiple array libraries.
- Projects that implement new types of arrays with, e.g., hardware-specific optimizations or auto-parallelization behavior, and need an API to put on top that is familiar to end users.
- End users that want to switch from one library to another without learning about all the small differences between those libraries.

Concrete use cases

- Use case 1: add hardware accelerator and distributed support to SciPy
- Use case 2: simplify einops by removing the backend system

API to xtensor

ation of array computations easier and more robust

Latest: github.com/data-apis/array-api/