Array-API Standards: One Ring to Rule them all!

WELCOME

By Prady0t

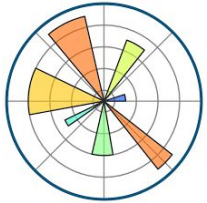
About Me

- Software Engineer @Pihex Labs
- Open Source Contributor @Pydata/sparse
- Likes working with Scientific Python Community.

Scientific Python

- What is Scientific Python? Well it mostly deals with Python libraries useful for scientific and technical computing. Some examples include: Numpy, Scipy, Matplotlib.
- Scientific Python ecosystem try to provide common API, spec and standards, interoperability and long term sustainability for the projects.

Different array libraries



We need different array libraries for different things we might be interested in.

Example: Numpy is used pretty much as a standard for numerical computing but it relies on CPU, whereas CuPy uses GPU.

Pydata/sparse library is used for sparse matrix calculation, and so on...

Different array libraries

- We can roughly divide the array libraries into two types:

  Array Producer Libraries: Libraries like Numpy, CuPy, JAX, Pydata/Sparse etc

  Array Consumer Libraries: Libraries like Scipy, Scikit-learn, Matplotlib etc

Array Producer Libraries

Array Producer Libraries are libraries that define or expose new types of arrays tailored for specific use cases, such as sparse data, GPU acceleration, or distributed computing.

They either implement new array classes or wrap existing array types to provide enhanced functionality.

Array Consumer Libraries

Array Consumer Libraries are libraries that operate on array-like objects without necessarily defining their own array types.

They consume arrays typically from NumPy or other array-producing libraries to perform computations, visualizations, analyses, or transformations.

```python
import numpy as np
from scipy import linalg

# NumPy array as producer
a = np.array([[1, 2], [3, 4]])

# SciPy consumes NumPy array directly
eigvals, eigvecs = linalg.eig(a)
print(type(a))          # <class 'numpy.ndarray'>
print(type(eigvecs))    # <class 'numpy.ndarray'>
```

The Problem?

You can already see the problem here. Consumer libraries are restricted by their respective producer libraries.

Scikit-learn cannot natively leverage GPUs because it uses NumPy arrays, which are CPU-based, and its algorithms are implemented in a way that assumes CPU execution. This makes integration with GPU backed array libraries like CuPy non-trivial.

The Solution?

What if Consumer libraries were able to use multiple Producer libraries in their backend?

Here comes Array API Standards.

Array API Standards

- Aims to provide a common, consistent API for array operations across different Python array libraries.
- Think of it as a set of agreed-upon functions and behaviors for array manipulation.
- Code written using the standard doesn't depend on a specific library's implementation details.
- Defines essential array creation, manipulation, arithmetic, and linear algebra operations.
- Developed through a collaborative effort by representatives from major array library projects.

Array API Standards

In terms of Producer Library:

- Defines the essential attributes and methods an array object must expose, such as shape, dtype, ndim

- A comprehensive set of mathematical and logical element-wise operations (e.g., abs, add, equal, exp, log, maximum, minimum, power, subtract, sin, cos, etc.)

- Set of functions are implemented as specified in Array API Standards

Array API Standards

In terms of Consumer Library:

- Instead of importing and calling functions directly, a compliant consumer library will write its code using the functions and methods defined by the Array API standard.

- Interoperability, basically array agnostic.

- Reduced Maintenance, consumer library only needs to maintain one code path that adheres to the Array API

- Simplified User Experience: Users of the consumer library can choose their preferred array backend

```python
import cupy as cp
from scipy import fft

# GPU array produced by CuPy
x = cp.linspace(0, 10, 1000)

# SciPy FFT can consume thanks to Array API / NEP 47
y = fft.fft(x)
print(type(y))  # <class 'cupy.ndarray'>
```

Any Questions?

Thank you!

THANK YOU