Sparse Arrays for Scientific Python

Proposal for CZI's Essential Open Source Software for Science program

Stéfan J. van der Walt (Berkeley Institute for Data Science, UC Berkeley)
K. Jarrod Millman (Division of Biostatistics, UC Berkeley)
Ross Barnowski (Berkeley Institute for Data Science, UC Berkeley)
Dan Schult (Department of Mathematics, Colgate)

Proposal Purpose

To improve sparse structures in SciPy so they support array semantics; to deprecate SciPy's sparse matrices and numpy.matrix; to assist with sparse array adoption in downstream ecosystem packages.

```
Words in summary (500 max): 492
Words in work plan (750 max): 440
Words in deliverables (500 max): 207
Words in existing_support  (250 max): 230
Words in landscape (250 max): 247
Words in value_biomed (250 max): 244
Words in dei (250 max): 245
```

Purpose

To improve sparse structures in SciPy so they support array semantics; to deprecate SciPy's sparse matrices and `numpy.matrix`; to assist with sparse array adoption in downstream ecosystem packages.

Summary

Sparse data refers to datasets where a high percentage of the values are zero or empty. This happens when relationships across dimensions (e.g. rows and columns) don't exist or are neglected. Sparse datasets are ubiquitous in modern scientific computing, including network analysis, signal processing, image processing, machine learning, etc. There exist many sparse data formats which save memory by only storing non-zero values, yet still allow efficient computation and manipulation.

In the scientific Python OSS ecosystem, `scipy.sparse` provides 7 sparse matrix data structures for datasets that are 2D (like a matrix). Their interface mimics the `numpy.matrix` interface using *matrix semantics*, which is fundamentally different from the `numpy.ndarray` interface which uses *array semantics*. Semantic differences mean that matrix objects and array objects are not interchangeable and this can lead to confusion or silent errors when standard array semantics are used incorrectly with a `matrix` object. The NumPy community has a long-standing desire to remove `numpy.matrix` and related semantics in favor of the `ndarray` semantics that have become the de facto standard for data in Python. The major blocker to this removal is the use of matrix semantics in `scipy.sparse`. Constructing array semantics interfaces for sparse data in Python has been discussed for years.

Unlike the universally accepted strided memory model on which `ndarray` is built, sparse data can be stored in many formats with varied memory usage and computational efficiency. Performance scales with amount of data and number of dimensions differently across formats. Some formats, such as COOrdinate (`coo`), directly generalize to any number of dimensions, whereas others do not. Also, some users need flexibility to switch between formats while others need strict format control. This makes interface design critical, involving communication across developers of many libraries throughout the scientific Python ecosystem. Interface input and facilitation of acceptance from `numpy`, `scipy`, and other libraries is critical. While the long term goal is implementation of an n-dimensional sparse interface with array semantics, this cannot be expected from a two-year project. We will focus on commonly used cases with 1D to 4D data, using appropriate sparse data structures and on collecting community input and facilitating community acceptance.

We propose to introduce sparse array data structures that provides array semantics for straightforward interaction with sparse data. The scope of the project includes the design and implementation of 1D and 2D sparse array data structures, expanding on the `scipy.sparse` package. An extension of some formats to 3D or 4D is also in scope. We will also take a leading role in the community migration of sparse data applications to this array interface throughout the libraries SciPy, NetworkX, scikit-image and scikit-learn. Other libraries are likely to follow suit and we will help them as needed. This is a community effort. The inclusion of sparse arrays in `scipy.sparse` will remove the last barrier for the scientific Python community to move to sparse array semantics, officially deprecating `numpy.matrix`. The resulting adoption and availability of sparse data tools could be transformative.

Work Plan

The primary focus of this proposal is the extension and refactoring of sparse data structures within SciPy to fully support NumPy `ndarray` semantics. `scipy.sparse` is the fundamental package for handling sparse data within the scientific Python ecosystem, and is used by various downstream libraries, including `scikit-image`, `scikit-learn`, and `NetworkX`. Its lack of an array interface is the last remaining blocker for the deprecation of `numpy.matrix`, which has been pending for over 5 years (see: https://github.com/numpy/numpy/pull/10142).

The changes proposed impact many libraries in the scientific Python ecosystem, and as such community coordination and implementation of a transition plan will form a large part of the project.

Currently, the prototype `scipy.sparse` array implementation builds on the older sparse matrix objects. These assume that all data is two-dimensional, whereas arrays are inherently nested: for example, a two-dimensional array is a container of one-dimensional arrays. Thus, when working with arrays, one-dimensional results often arise from slicing or reduction operations. Currently, those results are returned as *dense* NumPy arrays. A first step towards more complete array support is the addition of a one-dimensional sparse container and associated tests.

While implementing one-dimensional sparse arrays, we will work with core developers of SciPy on a transition plan for moving from the current matrix interface to sparse arrays. Since matrix and array semantics differ, simply switching from one to the other will break existing code that depends on `scipy.sparse`. Considerations in the transition plan include what behaviors to change, and how to advertise those changes so that developers of downstream code has sufficient chance to respond. It is likely that, even after implementation, the transition plan may need to be adjusted based on developer and user feedback.

The transition plan, as well as the open discussions that go into its formulation, will be captured in a design document, possibly in in an enhancement proposal or a Scientific Python Ecosystem Coordination (SPEC) document. The last step of the transition plan will be the deprecation of `numpy.matrix`, appropriately coordinated with the release of `scipy`.

While developing the transition plan, we will help core ecosystem packages to adopt the new `scipy.sparse` array interface. Doing so will provide feedback on common use-cases and inform the design of the transition plan. `scikit-learn`, `scikit-image`, and `networkx` are explicitly listed in this proposal as prominent consumers of `scipy.sparse` with large user bases; however, our aim is to assist with adoption more broadly ecosystem-wide.

The final component of the work plan is to study the feasibility of extending `scipy.sparse` arrays to higher-dimensional data. The full design and implementation of an n-dimensional sparse data structure is beyond the scope of this proposal.

Deliverables

SciPy typically has two feature releases per year around December and June. The proposed work will be done with this release schedule in mind, especially during the first year of the project when many of the additions to scipy are expected to occur. Given that downstream libraries cannot take advantage of any new `scipy.sparse` features until they have been released, explicit work on downstream projects is expected to be more concentrated in year 2.

Year 1

- One-dimensional sparse data structure added to `scipy.sparse`.
- Add support for sparse array indexing.
- Formalized transition plan for migrating the `scipy.sparse` default interface from matrices to arrays.
  - Note: new `scipy.sparse` array features are expected to be made available to users within the first two releases of scipy in year 1. The full transition from matrix to array semantics as the `scipy.sparse` default will likely be a longer process.

Year 2

- Formalized plan for the removal of `numpy.matrix` (likely via a NumPy Enhancement Proposal).
- Incremental adoption of `scipy.sparse` array features in downstream libraries, depending on version requirements in individual libraries.
- Implementation of transition plan to remove matrix objects within `scipy.sparse`.
  - Note: it is likely that deprecation periods will extend beyond the timeframe of this project.
- Preliminary design/test suite for support of higher-dimensional sparse arrays.

Existing support

This proposal involves working specifically on the `scipy.sparse` subpackage within the SciPy package, related cleanup in the `numpy.matrix` subpackage of the NumPy package, and aiding in the adoption of the updated sparse interface of downstream packages like NetworkX, scikit-learn, and scikit-image. (Note: these five projects were listed explicitly due to their central location in the ecosystem and the fact that the proposal limits the number of listed projects to 5. However, aiding sparse array migration for *any* scientific Python project in the ecosystem is considered in scope from our perspective.)

Each of these packages has received some form of funded support in the recent past, but only NetworkX has previous funding related to sparse data structures. NetworkX is receiving ongoing funding from Cycle 4 of the CZI EOSS program (EOSS4-0000000138) which included goals of more tightly integrating with scientific Python, including `scipy.sparse`, and an investigation of backend-neutral graph API, which involves `scipy.sparse` data structures. In support of this work, NetworkX has been an early adopter of changes to the `scipy.sparse` interface. This benefits the proposed work in `scipy.sparse` in several ways:

- Network analysis often involves sparse data, and applications in NetworkX provide diverse examples to aid in the design and testing of sparse arrays.
- By setting the minimum required version of SciPy to the latest stable release, NetworkX contributes to greater adoption of the new `scipy.sparse` interface among downstream users.

Landscape

`scipy.sparse` is the standard scientific Python toolkit for sparse data. Its main limitation is that the fundamental data structures implement matrix semantics, meaning that all data is inherently 2D and not interoperable with NumPy.ndarray semantics. We recently took first steps to address these limitations by introducing sparse arrays to `scipy.sparse`. We also converted NetworkX to that interface. The goal of this proposal is to support and solidify this effort. We are well positioned to work with scipy.sparse, SciPy more generally, NumPy, NetworkX, scikit-image and scikit-learn. We have spoken with core developers to ensure our visions align.

The other main library for sparse array analysis in Python is PyData's sparse package which provides n-dimensional sparse data structures that support array semantics appropriate for Numba compiled code. PyData is actively maintained and used by several hundred packages (according to GitHub). But the data structures are specialized and not possible to incorporate in `scipy.sparse`, The emphasis is on inter-operability with specialized computational array back-ends. Thus, PyData-sparse does not compete with `scipy.sparse`. The libraries are mutually beneficial, providing effective general data manipulation in `scipy-sparse` along with speed-up for special cases in PyData-sparse, analogous to the relationship between e.g. NumPy and JAX. Our project should improve adoption of array-based analyses for sparse data. Users can then move appropriate special cases to PyData-sparse. PyData-sparse represents a parallel effort to `scipy.sparse` with separate goals and design principles.

Value to Biomedical

A flexible, performant n-dimensional sparse array is a foundational tool with large impact potential across many scientific computing domains. Array semantics (indexing, reductions, broadcasting, vectorization) are already foundational components of data analysis workflows. A sparse data structure that implements array semantics could be a "drop-in" replacement in many machine-learning workflows where non-sparse code exists but needs to be applied to sparse datasets. Allowing sparse array operations using code developed for traditional array operations reduces memory and CPU constraints on data analysis and machine-learning algorithms. In addition to these foundational considerations, several direct biomedical applications involve sparse data. For example, biomedical imaging involves volumetric image spaces and time-series analysis (3D or 4D) with sparse data. Collections of data across patients or imaging devices can increase the dimension further. Image segmentation involves extraction of image features. Sparse image representation exploits the idea that similar features can be identified using sparse representations. Sparsity constraints allow identification of such features. Sparse data representations have also been applied to shape prior modeling, nonrigid registration and functional connectivity modeling. Clinical data is often sparse because most medical tests and procedures do not apply to most patients. Clustering patients and providers enables machine-learning techniques to identify likely conditions and treatments. The ability to collect large datasets with many variables from varied sources often results in sparse data. Machine-learning techniques are more powerful when they take advantage of this sparsity. Sparse data structures using effective semantics are crucial parts of this workflow.

DEI

We believe that healthy communities are built when everyone's voice is heard, when their perspective is valued, and when their work is recognized. It is also evident that better technical and social solutions can be found through wide participation.

Common to all projects in this proposal is an explicit dedication both to onboarding contributors who are new to open source development, and to integrating existing contributors more tightly into project communities.

One mechanism used to implement these goals is mentored projects. Each of the participants listed in this proposal has served as a mentor: either as part of an established program (e.g. GSoC, GSoD, Outreachy) or separately, often supporting contributors from underrepresented communities. For example, NetworkX is currently mentoring a women through Outreachy and the PI has hired a women to work with us this summer to contribute to both the Scientific Python project and NetworkX.

The Scientific Python project team (comprising members of this proposal) has also undertaken to generate video content in an effort to reach potential contributors who may not yet be comfortable interacting through traditional OSS development channels like GitHub or mailing lists. These outreach videos include content related to getting started on projects within the scientific Python ecosystem, and interviews with mostly newer maintainers to highlight different ways people can become contributors and maintainers. We've also been successful with recruiting a diverse group of people to participate on the various teams associated with the project (e.g., https://scientific-python.org/about/#people).