Sparse Matrices

Bast Practices with Applications to Python

Jason DeBacker

January 16, 2025

Roadmap

Intro

Storing sparse matrices

Computation with sparse matrices

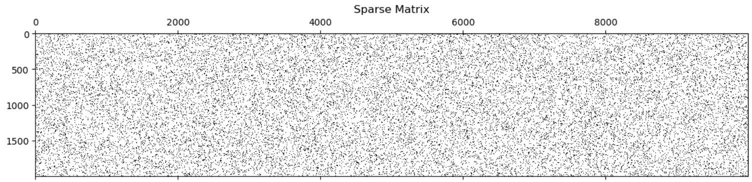
Implementation in Python

What is a sparse matrix?

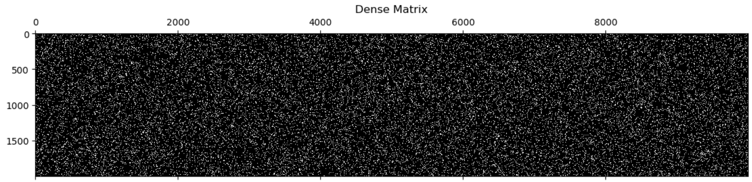
- It's a matrix with a lot of zeros.
- How many zeros? ¯_(ツ)_/¯
  - → Generally, number of non-zero elements equals number of rows or columns
  - → A measure of sparsity: $\frac{\text{number of non-zero elements}}{\text{number of rows} \times \text{number of columns}}$

What is a sparse matrix?

- Sparse (10% of elements are non-zero):



- Dense:



What is special about sparse matrices?

- In some sense, nothing. It’s an object with data, like any other.
- On the other hand, we might be able to exploit the sparsity to save memory and computation time.
- **Related**: Large matrices are often sparse, so saving memory and/or compute can be important

Roadmap

Intro

Storing sparse matrices

Computation with sparse matrices

Implementation in Python

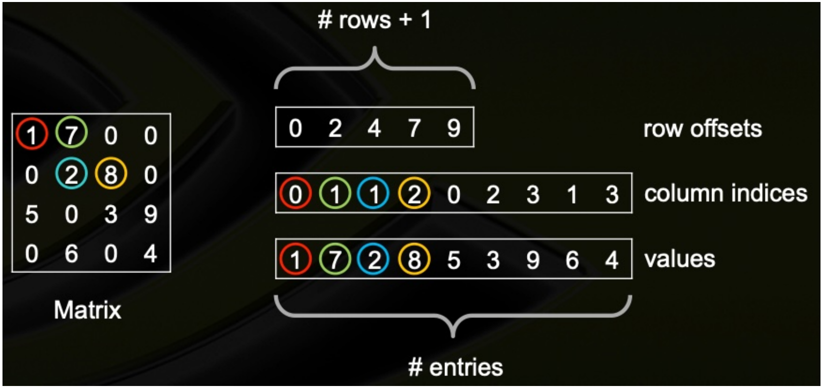
How to store a sparse matrix

- There are several conventions for storing sparse matrices:
  - $\rightarrow$ Compressed Sparse Row (CSR)
  - $\rightarrow$ Compressed Sparse Column (CSC)
  - $\rightarrow$ List of Lists (LIL)
  - $\rightarrow$ Dictionary of Keys (DOK)
  - $\rightarrow$ Coordinate List (COO)
- What's common across all these: they only store the non-zero elements with pointers to where they are in the matrix

Example: Compressed Sparse Row (CSR)

- CSR is one of the most common ways to store a sparse matrix
- CSR works as follows:
  - → CSR requires three arrays.
  - → The first array stores the cumulative count of nonzero values in all current and previous rows.
  - → The second array stores column index values for each nonzero value.
  - → The third array stores all nonzero values.

Example: Compressed Sparse Row (CSR)



Sparse Matrix Storage

- In this example with CSR, the matrix is not that sparse (7 of 16 values were zeros).
- But with large, sparse matrices, storing the matrix in a format like CSR can save a lot of memory

Roadmap

Intro

Storing sparse matrices

Computation with sparse matrices

Implementation in Python

- Similar to how we can economize on storage space, we can also economize on computation
- e.g., we might not need to loop over every element or the matrix if we know the product of certain elements will be zero
- Excellent examples of operations with Coordinate-wise vs. CSR in these notes
- Summary:
  - → CSR – good compression of matrix
  - → CSR – row access is easiest
  - → CSC – column access is easiest
  - → COO – column and row equally easy, but more compute in matrix operations that with CSR or CSC

Roadmap

Intro

Storing sparse matrices

Computation with sparse matrices

Implementation in Python

SciPy sparse matrices

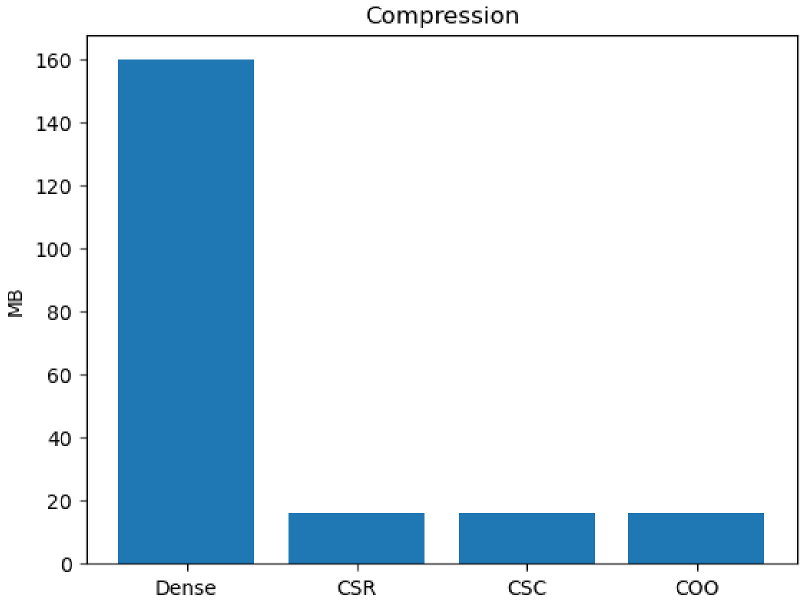
- `scipy.sparse` is a module in the SciPy library that provides tools for working with sparse matrices
- It provides classes for the different sparse matrix formats (CSR, CSC, COO, etc.)
  - → e.g., to cast a Numpy array called `data` as a CSR format sparse array, do `scipy.sparse.csr_matrix(data)`
  - → you can do similar for CSC, COO, etc.
- The module also provides functions for matrix operations
- To go back to a dense Numpy matrix us the `toarray()` method

WARNING from SciPy:

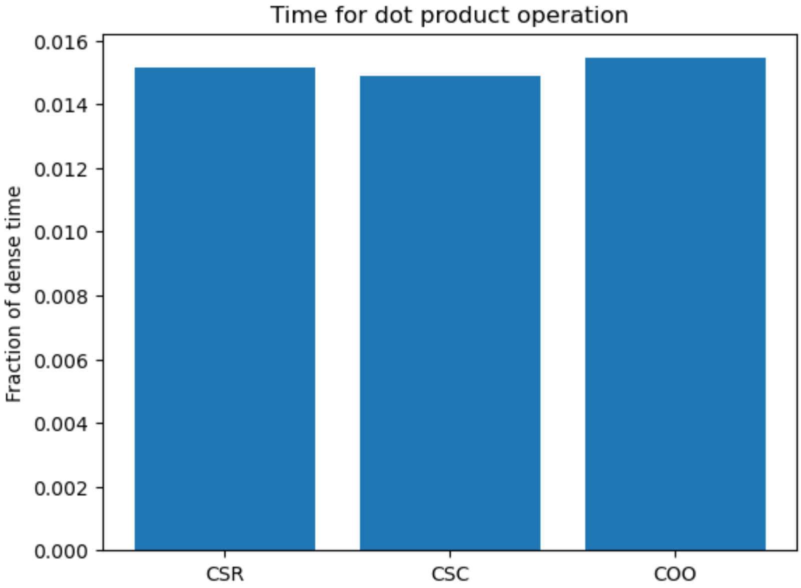
Warning:

Despite their similarity to NumPy arrays, it is strongly discouraged to use NumPy functions directly on these arrays because NumPy typically treats them as generic Python objects rather than arrays, leading to unexpected (and incorrect) results. If you do want to apply a NumPy function to these arrays, first check if SciPy has its own implementation for the given sparse array class, or convert the sparse array to a NumPy array (e.g., using the `toarray` method of the class) before applying the method.

Compression with Scipy



Computation with Scipy (matrix mult.)



Regression models and sparse matrices

Library	Compatible with scipy sparse arrays?
statsmodels	No
linearmodels	No (but does use for HDFE)
sklearn	Yes

Sparse data with Pandas

- Pandas has a sparse data structure object
- It’s not a sparse matrix, but it can be useful for dataframes with a lot of missing values
- The sparse data structure is a subclass of the regular Pandas data structure
- It only stores the non-missing values (or non-zero values – you can specify what to consider as ”missing”)

Implementing sparse data structures Pandas

- Supposed you have a DataFrame named `df` with no missing values, but many zeros
- You can cast it as a sparse DataFrame with
  ```
  sdf = df.astype(pd.SparseDtype("int", 0))
  ```
- With this, the 0's will not be stored in the DataFrame `sdf`