Sparse Numerical Linear Algebra

Robert Klöfkorn

NUMN21/FMNN25

Sparse Numerical Linear Algebra

Problem: Number of matrix entries scale with N^2 where N is the number of unknowns.

dense matrix

$$\begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix}$$

Scales with N^2.

Sparse Numerical Linear Algebra

Problem: Number of matrix entries scale with N^2 where N is the number of unknowns.

dense matrix

$$\begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix}$$

Scales with N^2.

sparse matrix

$$\begin{pmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & \mathbf{0} & \\ & 1 & -2 & 1 & & \\ & & 1 & -2 & 1 & \\ & \mathbf{0} & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{pmatrix}$$

Scales with $3N$.

Density and Sparsity

In numerical analysis and scientific computing, a **sparse** matrix is a matrix in which most of the elements are zero.
By contrast, if most of the elements are nonzero, then the matrix is considered **dense**.

Definition 1
Let $A \in R^{n \times m}$ and
$A_{nz} := \{(i,j) \,|\, a_{ij} \neq 0 \text{ for } i = 1, \ldots, n \text{ and } j = 1, \ldots, m\}$.
We call

$$D := \frac{|A_{nz}|}{n\, m}$$

the density of A.
If $D < \frac{1}{2}$ we call A sparse.

Sparse Matrix Formats

- Dictionary of keys (DOK) where (row, column)-pairs are mapped to a value
- List of lists (LIL) where one list per row is stored, with each entry containing the column index and the value.
- Coordinate list (COO) stores a list of (row, column, value) tuples.
- Compressed sparse row (CSR, CRS or Yale format) where three (one-dimensional) arrays, that respectively contain nonzero values, the extents of rows, and column indices are stored. It is similar to COO, but compresses the row indices, hence the name. This format is very efficient for matrix-vector multiplication.
- Compressed sparse column (CSC or CCS) is similar to CSR but compresses the column indices.
- More formats for special structures like **banded**, **diagonal**, etc.

Sparse Matrix Formats

- Dictionary of keys (DOK) where (row, column)-pairs are mapped to a value
- List of lists (LIL) where one list per row is stored, with each entry containing the column index and the value.
- Coordinate list (COO) stores a list of (row, column, value) tuples.
- Compressed sparse row (CSR, CRS or Yale format) where three (one-dimensional) arrays, that respectively contain nonzero values, the extents of rows, and column indices are stored. It is similar to COO, but compresses the row indices, hence the name. This format is very efficient for matrix-vector multiplication.
- Compressed sparse column (CSC or CCS) is similar to CSR but compresses the column indices.
- More formats for special structures like **banded**, **diagonal**, etc.

Sparse Matrix Formats

- Dictionary of keys (DOK) where (row, column)-pairs are mapped to a value
- List of lists (LIL) where one list per row is stored, with each entry containing the column index and the value.
- Coordinate list (COO) stores a list of (row, column, value) tuples.
- Compressed sparse row (CSR, CRS or Yale format) where three (one-dimensional) arrays, that respectively contain nonzero values, the extents of rows, and column indices are stored. It is similar to COO, but compresses the row indices, hence the name. This format is very efficient for matrix-vector multiplication.
- Compressed sparse column (CSC or CCS) is similar to CSR but compresses the column indices.
- More formats for special structures like **banded**, **diagonal**, etc.

Sparse Matrix Formats

- Dictionary of keys (DOK) where (row, column)-pairs are mapped to a value
- List of lists (LIL) where one list per row is stored, with each entry containing the column index and the value.
- Coordinate list (COO) stores a list of (row, column, value) tuples.
- Compressed sparse row (CSR, CRS or Yale format) where three (one-dimensional) arrays, that respectively contain nonzero values, the extents of rows, and column indices are stored. It is similar to COO, but compresses the row indices, hence the name. This format is very efficient for matrix-vector multiplication.
- Compressed sparse column (CSC or CCS) is similar to CSR but compresses the column indices.
- More formats for special structures like **banded**, **diagonal**, etc.

Sparse Matrix Formats

- Dictionary of keys (DOK) where (row, column)-pairs are mapped to a value
- List of lists (LIL) where one list per row is stored, with each entry containing the column index and the value.
- Coordinate list (COO) stores a list of (row, column, value) tuples.
- Compressed sparse row (CSR, CRS or Yale format) where three (one-dimensional) arrays, that respectively contain nonzero values, the extents of rows, and column indices are stored. It is similar to COO, but compresses the row indices, hence the name. This format is very efficient for matrix-vector multiplication.
- Compressed sparse column (CSC or CCS) is similar to CSR but compresses the column indices.
- More formats for special structures like **banded**, **diagonal**, etc.

Sparse Matrix Formats

- Dictionary of keys (DOK) where (row, column)-pairs are mapped to a value
- List of lists (LIL) where one list per row is stored, with each entry containing the column index and the value.
- Coordinate list (COO) stores a list of (row, column, value) tuples.
- Compressed sparse row (CSR, CRS or Yale format) where three (one-dimensional) arrays, that respectively contain nonzero values, the extents of rows, and column indices are stored. It is similar to COO, but compresses the row indices, hence the name. This format is very efficient for matrix-vector multiplication.
- Compressed sparse column (CSC or CCS) is similar to CSR but compresses the column indices.
- More formats for special structures like **banded**, **diagonal**, etc.

Sparse Matrix Formats

- Dictionary of keys (DOK) where (row, column)-pairs are mapped to a value
- List of lists (LIL) where one list per row is stored, with each entry containing the column index and the value.
- Coordinate list (COO) stores a list of (row, column, value) tuples.
- Compressed sparse row (CSR, CRS or Yale format) where three (one-dimensional) arrays, that respectively contain nonzero values, the extents of rows, and column indices are stored. It is similar to COO, but compresses the row indices, hence the name. This format is very efficient for matrix-vector multiplication.
- Compressed sparse column (CSC or CCS) is similar to CSR but compresses the column indices.
- More formats for special structures like **banded**, **diagonal**, etc.

Scipy.sparse: Create CSR from Dense Matrix

dense2csr.py

```
from scipy.sparse import csr_matrix
import numpy as np

# create a dense matrix
D = np.array([[ -2.,  1.,  0.,  0.],
              [  1., -2.,  1.,  0.],
              [  0.,  1., -2.,  1.],
              [  0.,  0.,  1., -2.]])

# create sparse matrix from dense matrix
A = csr_matrix( D )

# let's see what we got
print ( A )
```

Scipy.sparse: Create CSR from Dense Matrix

dense2csr.py

```
from scipy.sparse import csr_matrix
import numpy as np

# create a dense matrix
D = np.array([[ -2.,  1.,  0.,  0.],
              [  1., -2.,  1.,  0.],
              [  0.,  1., -2.,  1.],
              [  0.,  0.,  1., -2.]])

# create sparse matrix from dense matrix
A = csr_matrix( D )

# let's see what we got
print ( A )
```

```
blade ~/work/Lehre/NUMN21/sparse $ python dense2csr.py
  (0, 0)        -2.0
  (0, 1)        1.0
  (1, 0)        1.0
  (1, 1)        -2.0
  (1, 2)        1.0
  (2, 1)        1.0
  (2, 2)        -2.0
  (2, 3)        1.0
  (3, 2)        1.0
  (3, 3)        -2.0
blade ~/work/Lehre/NUMN21/sparse $
```

Scipy.sparse: Create CSR from COO format

coo2csr.py

```
from scipy.sparse import csr_matrix
import numpy as np

# row indices
rowind = np.array([0, 0, 1, 1, 1, 2, 2, 2, 3, 3])

# column indices
colind = np.array([0, 1, 0, 1, 2, 1, 2, 3, 2, 3])

# matrix values
values = np.array([-2.,1.,1.,-2.,1.,1.,-2.,1.,1.,-2.])

# create sparse from COO format
# all arrays have the same length
A = csr_matrix( (values, (rowind, colind) ) )

# let's see what we got
print (A)
```

Scipy.sparse: Solving $Ax = b$

csrsolve.py

```
from scipy.sparse import csr_matrix
from scipy.sparse.linalg import spsolve,cg
import numpy as np
# row indices, column indices and values
rowptr = np.array([0, 0, 1, 1, 1, 2, 2, 2, 3, 3])
colind = np.array([0, 1, 0, 1, 2, 1, 2, 3, 2, 3])
values = np.array([-2.,1.,1.,-2.,1.,1.,-2.,1.,1.,-2.])
# create sparse from COO format
A = csr_matrix( (values, (rowptr, colind)) )

# setup right hand side
b = np.array([1.,2.,2.,1.])

# solve Ax = b
x = spsolve( A, b )

# solve Ax = b, returns tuple with solution and iteration count
x = cg( A, b )[0]

# lets check the result and print || Ax - b ||
print("Solution is correct? ", np.allclose( A@x, b))
```

Further Reading

scipy.sparse
https://docs.scipy.org/doc/scipy/reference/sparse.html
Accessed NUMN21/FMNN25.