Exercise-Sheet-B

March 31, 2021

1 Exercise sheet B - Template

[36]:
```
# Some imports
import numpy as np
import scipy.sparse as sp
import scipy.sparse.linalg as spla
```

1.1 Exercise 1

Define the correct matrix **A** and **b** for use with, e.g., `np.linalg.lstsq` or `scipy.linalg.lstsq` (see documentation), then call the least-squares solver appropriately.

Note: Both examples from the exercise sheet work the same way, just the matrices are different.

[9]:
```
# UNCOMMENT AND COMPLETE
# A = np.array(...).astype('float32')
# b = np.array(...).astype('float32')
# ...
```

1.2 Exercise 2

Implement the Karczmarz algorithm from the lecture (see here) and solve both problems from Exercise 1 with this algorithm.

Below is some **helper code**, e.g., to normalize the system of linear equations. Say you have

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

and

$$\mathbf{b} = [3, 4]^\top$$

then `normalize_system` would work as follows:

[18]:
```
A = np.array([[1,0,1],[0,0,1]]).astype('float32')
b = np.array([3,4]).astype('float32')
ret = normalize_system(A,b)
A_norm = ret[0]
b_norm = ret[1]
print(A_norm)
print(b_norm)
```

```
[[0.70710677 0.         0.70710677]
 [0.         0.         1.        ]]
[2.1213205 4.       ]
```

Use `normalize_system` to create a normalized **A** and **b** that you use when calling your implementation of the Kaczmarz algorithm.

1.2.1 Termination criterion

As termination criterion compute

$$\mathbf{r}^{(k)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(k)}$$

and check

$$\|\mathbf{r}^{(k)}\|_2 \leq 10^{-6}$$

where the norm can be computed using `np.linalg.norm`.

Note: Inner products with `numpy` can easily be done either with `np.dot` or using the syntax `a @ b`.

1.2.2 Initialization

Initialize $\mathbf{x}^{(0)}$ as the vector of **all zeros**.

[1]:
```
def compute_row_norms(A):
    if sp.issparse(A):
        return spla.norm(A, axis=1)
    return np.linalg.norm(A, axis=1)


def normalize_matrix(A, row_norms):
    normalization_matrix = sp.diags(1 / row_norms)
    return normalization_matrix @ A


def normalize_system(A, b):
    if not sp.issparse(A):
        A = np.array(A)

    row_norms = compute_row_norms(A)
    A = normalize_matrix(A, row_norms=row_norms)
    b = np.array(b).ravel() / row_norms

    return A, b, row_norms
```

[25]:
```
def kaczmarz(A,b):
    # YOUR CODE GOES HERE
    # return x_k
    pass
```