EPFL

HPC for numerical methods and data analysis

Fall Semester 2023

Prof. Laura Grigori

Assistant: Mariana Martinez

Session 10 – November 21, 2023

Deterministic rank revealing factorizations for low rank approximation

Exercise 0 (Optional): Numerical stability of randomized Nystrom

Just as in last week's exercise, consider the MNIST data set and the matrix A as defined in the project or in last week's exercise sheet (with a fixed value of c). **You'll have to be careful with this data set here and in your project because you have to make sure the data set is normalized, i.e. all the entries are between 0 and 1**. Take a relatively small sample of the training set. In this section you can use either last week's code or your code from your project. Consider two different types of sketching matrices, Ω_1, Ω_2. For different sketching dimensions of both types of matrices, compute the relative error of the low rank approximation in terms of the nuclear norm. Provide a graph that compares these errors. Comment on your findings.

Exercise 1: Tournament pivoting

The truncated SVD provides the best low rank approximation in terms of the Frobenius and L2 norms. Recall that the QR decomposition with column pivoting relies on a permutation matrix Π and computes the decomposition $A\Pi = QR$. There are many ways of building such permutation matrix. Let

$$\rho_i(W) = \frac{1}{\|W^{-1}[i,:]\|}$$
$$\chi_j(W) = \|W[:,j]\|$$

Consider the following theorem from `https://inria.hal.science/hal-02947991v2/document`:

Theorem 1 *Let A be an $m \times n$ matrix, $1 \leq k \leq \min(m,n)$. For any $f > 1$ there exists a permutation matrix Π such that the decomposition $A\Pi = QR$ verifies for all $(i,j) \in [1,k] \times [1, n-k]$,*

$$(R_{11}^{-1} R_{12})_{ij}^2 + \rho_i^2(R_1 1)\chi_j^2(R_2 2) \leq f^2.$$

Such factorization is called a strong RRQR factorization.

Recall tournament pivoting for deterministic column selection. (Refer to the image below).

a) Consider a matrix A partitioned into 4 column blocks. Each processor has one of these blocks.

b) Implement strong RRQR (there is a MATLAB implementation, you can use that as a template for building your own Python implementation).

c) For each block of columns, select k columns using strong RRQR, save their indices are given in I_{i0}. Be careful with these indices, you might have to deal with "global" and "local" indices.

d) In each processor output these indices $I_{00}, I_{10}, I_{20}, I_{30}$.

e) Test your method with three different matrices:

- $A = H_n D H_n^\top$, where H_n is the normalized Hadamard matrix of dimension n, D is a diagonal matrix of your choice. Pick n to be "small".
- Load the normalized MNIST data set and build A as in the project (or last week's exercises). Select a few columns and rows.
- Build A from the MNIST data set with 2^{11} data points.

f) Comment your results with the different matrices. Do you notice a problem when you take more rows and columns using the MNIST data set?

g) Using $I_{00}, I_{10}, I_{20}, I_{30}$ build a low rank approximation of A. Check the L2 norm of the error with respect to the error of the truncated SVD.

h) Check if the singular values of these selected columns approximate well the singular values of A.

i) Check if the diagonal elements of R_{11} approximate well the singular values of A.