Rank Structured Matrices: Key Tools

Scribes: Marc Aurele Gilles, David Eriksson, Mateo Díaz

September 28, 2017

1 Introduction

In the last lecture we covered the Fast Multipole Method (FMM). This lecture will cover a generalization called rank structured matrices. Recall that FMM is valid in the following setting:

- We have a collection of points $\{x_i\}_{i=1}^n$ on a bounded domain $\Omega \subset \mathbb{R}^m$.
- We are given a kernel function $K : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$.

Last time we considered the kernel $K(x, y) = \log|x - y|$. Using the collection of points $\{x_i\}_{i=1}^n$ and the kernel $K(x, y)$, we defined $[K]_{i,j} = K(x_i, x_j)$. We showed that FMM allows us to compute the vector product Kq in $\mathcal{O}(n)$, rather than $\mathcal{O}(n^2)$. However, the FMM method has several limitations:

1. It only allows us to compute matrix-vector multiplication, not to solve linear system i.e., it does not help to find q such that $Kq = f$.
2. It requires a lot of analytical work for each different kernel. Indeed, we need to analytically compute a multipole expansion for a specific kernel $K(x, y)$ to make FMM work.

We will resolve both these issues using rank structured matrix computations. Before we describe these methods we will introduce some notation.

1.1 Notation

We will consider two disjoints subsets $\Omega_1, \Omega_2 \subset \Omega$ and associated sets of indices $I_1 = \{i \mid x_i \in \Omega_1\}$ and $I_2 = \{i \mid x_i \in \Omega_2\}$.

1. We denote the sub-block of K which corresponds the interaction between the targets in Ω_1 and the sources in Ω_2 by $K(I_1, I_2) \in \mathbb{R}^{m_1 \times m_2}$ where $m_1 = |I_1|$, $m_2 = |I_2|$. Typically, $m_1 \ll m_2$.
2. We say that some matrix A is compressible if $A \approx UV^T$ for some $U \in \mathbb{R}^{m_1 \times k}$, $V \in \mathbb{R}^{m_2 \times k}$ with $k \ll m_1$. In other words, A is low rank or approximately low rank.

Note that since the sets are disjoints, $K(I_1, I_2)$ represents an off-diagonal block of K.

1.2 Admissibility

There are two large classes of rank structured algorithms based on the assumptions on the matrix K. Consider two subsets Ω_1, Ω_2.

1. **Weak admissibility:** K is weakly admissibility if $K(I_1, I_1^c)$ is compressible.
2. **Strong admissibility:** K is strongly admissibility if the distance between the two sets Ω_1 and Ω_2 being greater than some distance D implies that the off-diagonal block $K(I_1, I_2)$ is compressible (i.e. low-rank), see Figure 1.

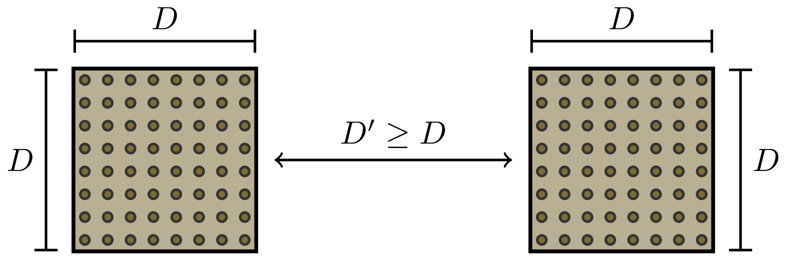

Figure 1: Two boxes in $\mathbb{R}^2$ with side length D are assumed to be numerically low rank if the distance between them is at least D, while weak admissibility only requires them to be non-overlapping [4]

Note that strong admissibility is actually a weaker condition than weak admissibility. Strong admissibility can also be stated as: all points outside of distance D away from the Ω_1 have "low rank interaction" with Ω_1. FMM is a special case of strong admissibility: we can use a multipole expansion only when sources and targets are far away, but compute all individual interactions for neighbors. There are tradeoffs between those two classes in accuracy, speed and assumptions you have to make. The two main features of rank structured matrix computations are:

1. It provides us with a way to apply approximate inverses, i.e., it allows us to compute Bf where $B \approx K^{-1}$.
2. It replaces the analytical work by a numerical approximation using the fact that some blocks are low rank with good accuracy.

This type of algorithm can only be applied efficiently when there are low-rank blocks.

Next we are going to describe two linear algebra tools: a block diagonal factorization and interpolative decomposition, which we will combine to form the essential building block of rank structured computation: skeletonization. The algorithms we will describe are called "skeletonization based algorithms", and were introduces by Martinssen and Rokhlin in [3]. In particular, we are looking at making a multiplicative factorization which goes back to Greengard and Ho [2].

2 A block diagonal factorization

The first piece in performing skeletonization is block elimination. Assume that the matrix $A \in \mathbb{R}^{N\times N}$, and consider a partition of the indices given by $[n] = I \cup J \cup R$ such that

$$A = \begin{bmatrix} A_{II} & A_{IJ} & 0 \\ A_{IJ} & A_{JJ} & A_{JR} \\ 0 & A_{RJ} & A_{RR} \end{bmatrix}.$$

Suppose we want to solve a linear system with this matrix. We could try to use LU, or Cholesky. We can handle the blocks separately if we can reduce the matrix to block diagonal form. Note that if we assume A_{II} is non-singular (which is typically the case in applications), we can define the matrices:

$$L = \begin{bmatrix} I & & \\ A_{JI}A_{II}^{-1} & I & \\ & & I \end{bmatrix} \qquad U = \begin{bmatrix} I & A_{II}^{-1}A_{IJ} & \\ & I & \\ & & I \end{bmatrix},$$

where the empty slots are zero entries. With this notation in hand, note that

$$\hat{A} := LAU = \begin{bmatrix} A_{II} & & \\ & S_{JJ} & A_{JR} \\ & A_{RJ} & A_{RR} \end{bmatrix}.$$

The matrices L and U have explicit inverses that are very easy to compute given L, U:

$$L^{-1} = \begin{bmatrix} I & & \\ -A_{JI}A_{II}^{-1} & I & \\ & & I \end{bmatrix} \qquad U^{-1} = \begin{bmatrix} I & -A_{II}^{-1}A_{IJ} & \\ & I & \\ & & I \end{bmatrix}.$$

That is, L and U differ from their inverses only by a negative sign in the $(1,2)$ and $(2,1)$ blocks respectively. Furthermore, the matrix $\hat{A}$ is block diagonal. This implies that we only need to factor the blocks separately which is computationally cheaper than working with the full matrix. Thus we have a factorization which is cheap to invert:

$$A = L^{-1}\hat{A}U^{-1}.$$

This gives us a tool to transform a matrix where all indices "talk" to each other, from one where only blocks talk to each other. If the large blocks of the matrix A have the same structure has A then we can recurse this algorithm and break down our blocks further into smaller blocks. The next section will show how to handle the case where the corner blocks A_{IR} are non-zeros.

3 Interpolative Decomposition

3.1 Definition

Suppose we want to compute a low rank representation of a block of the matrix. We can use an SVD, which provides the optimal rank r representation of a matrix, but the SVD is expensive to compute. In order to get computationally cheaper algorithms, we need to define a "goodness" measure of a low rank approximation of a function. Given $|I| \times |J|$ matrix A_{IJ}, and a tolerance ϵ, an ϵ-accurate interpolative decomposition (ID) of A_{IJ} is a partition of the points in J into two sets:

1. skeleton points S
2. redundant points R

with $S \subset J$, $R = J/S$, and matrix T such that:

$$\|A_{IR} - A_{IS}T\|_2 \leq \epsilon \|A_{IJ}\|_2. \tag{1}$$

In other words, it allows us to write the redundant columns of A (A_{IR}) as a linear combination of the skeleton columns of A (A_{IS}) up to some accuracy. The matrix T gives us the interpolation coefficients. Written this way, $S = I$. is a trivial solution for any accuracy, so we want $|S|$ to be as small as possible. Another way to state this is:

$$A_{IJ}\Pi \approx A_{IS} \begin{bmatrix} I & T \end{bmatrix},$$

where the matrix A_{IS} is tall and skinny, and Π is a permutation of the columns. In other words, we represent the columns of A_{IJ} by a subset A_{IS}.

As stated earlier, the k truncated SVD of A provides the optimal rank k approximation of A. This does not provide the same guarantee, but there is no asymptotic difference. For example, if the the SVD would provide us with an ϵ-approximation of rank k, then this may provide us with an ϵ-approximation of rank $k + 5$. The next section will show that this can be computed at a lower computational cost.

3.2 Computation

To compute the (ID), we use the QR pivoted column factorization. Recall that

$$A = QR$$

where Q is unitary and R upper triangular if a QR factorization of A. Now assume $|I| > |J|$. A pivoted QR factorization is defined by three matrices Q, R, Π such that

$$A\Pi = Q \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}$$

where Π is a permutation of the columns. We are going to choose Π such that R_{11} is as well conditioned as possible. Finding the optimal Π is NP-hard, but there are heuristics that work well [1].

Consider the factorization where we have split the columns $Q = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix}$

$$A\Pi = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} = \begin{bmatrix} Q_1 R_{11} & Q_1 R_{12} + Q_2 R_{22} \end{bmatrix}.$$

Let us get back to the matrices with indices I, J, S, R with $J = S \cup R$, $S \cap R = \emptyset$. We are seeking an ID of A_{IJ}. We can set $A_{IS} = Q_1 R_{11}$. Note that

$$A_{IJ}\Pi = \begin{bmatrix} A_{IS} & Q_1 R_{12} + Q2R22 \end{bmatrix} = \begin{bmatrix} A_{IS} & Q_1 R_{11} R_{11}^{-1} R_{12} + Q_2 R_{22} \end{bmatrix} = \begin{bmatrix} A_{IS} & A_{IS} R_{11}^{-1} R_{12} + Q_2 R_{22} \end{bmatrix}$$

We can let $T = R_{11}^{-1} R_{12}$, and if $\|R_{22}\| = \|QR_{22}\| < \epsilon$, we may ignore it. Now note that up that this gives us, up to permutations, a partition of the matrix $A_{IJ}\Pi = \begin{bmatrix} A_{IS} & A_{IR} \end{bmatrix}$ such that

$$A_{IJ}\Pi = \begin{bmatrix} A_{IS} & A_{IR} \end{bmatrix} = \begin{bmatrix} A_{IS} & A_{IS}T + Q_2 R_{22} \end{bmatrix}.$$

Thus

$$\|A_{IR} - A_{IS}T\| = \|R_{22}\| < \epsilon,$$

which up to a constant is the definition of ID in (1).

Note that the condition that $\|QR_{22}\| \approx 0$ is approximately equivalent to saying that A_{IJ} is compressible, which is the strong admissibility assumption. In practice, we don't want to compute the full QR, but we can a this trick so that at some point we are confident that we have a good ID and ignore the rest of the computation. These factorizations have high-performance implementations in e.g., Python, LAPACK, and MATLAB. Next, we put block diagonalization and ID together to perform skeletonization.

4 Skeletonization

Assume we have a large domain Ω and a small box B in the corner of Ω. We denote the neighbors of B by N, and the points far away from B by F. This is illustrated in Figure 2.

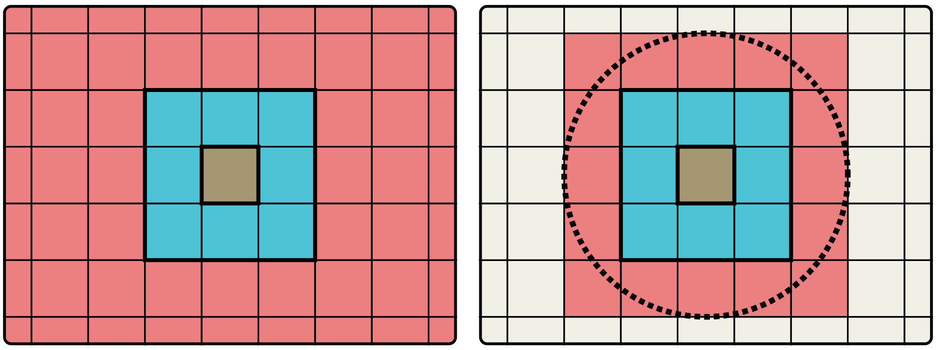

Figure 2: The near-field N are the blue boxes and the far-field F are the red boxes [4]

The idea is to use what interpolative decomposition to transform the matrix K into a matrix with zero blocks on the bottom left and top right corner. Once the matrix has this structure we can use the block diagonalization factorization that we covered in §2 to turn it into a block diagonal matrix with a small block and a giant block.

We will assume strong admissibility, i.e. the interactions between B and N are not low rank, but the interactions between B and F are. Up to some permutation P we can write the kernel matrix K in the following way:

$$P^*KP = \begin{bmatrix} K_{BB} & K_{BN} & K_{BF} \\ K_{NB} & K_{NN} & K_{NF} \\ K_{FB} & K_{FN} & K_{FF} \end{bmatrix},$$

where we have assumed that $|B| < |N| \ll |F|$, and that K_{FN} is compressible. Assuming that K is symmetric we compute an ϵ-accurate ID of K_{FB} matrix with $B = S \cup R$

$$K_{FB} = \begin{bmatrix} K_{FR} & K_{FS} \end{bmatrix} \approx K_{FS} \begin{bmatrix} T & I \end{bmatrix},$$

where we have assumed without loss of generality that $B = \begin{bmatrix} R & S \end{bmatrix}$. This forms a partition of the points in the box B into two sets, the set of redundant points F, and the set of skeletons points S. Physically, the skeleton points are typically the points close to the boundary of the box, and the points R are the points in the interior of the box. However, it is difficult to prove any such statement rigorously.

Using our ID, we can write

$$P^*KP = \begin{bmatrix} K_{RR} & K_{SR} & K_{RN} & K_{RF} \\ K_{SR} & K_{SS} & K_{SN} & K_{RN} \\ K_{NR} & K_{NS} & K_{NN} & K_{NF} \\ K_{FR} & K_{FS} & K_{FN} & K_{FF} \end{bmatrix} \approx \begin{bmatrix} K_{RR} & K_{SR} & K_{RN} & TK_{FS} \\ K_{SR} & K_{SS} & K_{SN} & K_{RN} \\ K_{NR} & K_{NS} & K_{NN} & K_{NF} \\ TK_{FS} & K_{FS} & K_{FN} & K_{FF} \end{bmatrix}.$$

If we define

$$U_T = \begin{bmatrix} I & -T^* & & \\ & I & & \\ & & I & \\ & & & I \end{bmatrix} \qquad L_T = \begin{bmatrix} I & & & \\ -T & I & & \\ & & I & \\ & & & I \end{bmatrix}$$

we have that

$$U_T P^* K P L_T \approx \begin{bmatrix} X_{RR} & X_{RS} & X_{RN} & 0 \\ X_{SR} & X_{SS} & X_{SN} & K_{SF} \\ X_{NR} & X_{NS} & K_{NN} & K_{Nf} \\ 0 & K_{FS} & K_{FN} & K_{FF} \end{bmatrix},$$

where X denotes blocks of the matrix that have been updated. Once again, L_T and U_T are very easy to invert: we only need to flip the sign in the $(2,1)$ and $(1,2)$ block respectively. We now have a matrix with zero blocks in the bottom left and top right corner and can use the block diagonalization algorithm described in §2. That is, using X_{RR} as the "pivot" we can define matrices L and U such that

$$LU_T P^* K P L_P T U \approx \begin{bmatrix} \hat{X}_{RR} & 0 & 0 & 0 \\ 0 & \hat{X}_{SS} & \hat{X}_{SN} & K_{SF} \\ 0 & \hat{X}_{NS} & \hat{X}_{NN} & K_{NF} \\ 0 & K_{FS} & K_{FN} & K_{FF} \end{bmatrix},$$

where $\hat{X}$ denote blocks that have changed. All the matrices $L, U, L_T U_T$ are cheap to apply and invert. We can use a factorization of the X_{RR} block. Factoring the large block is easier than factoring the original matrix. This factorization is called the skeletonized matrix K with respect to the block B, and is denoted $Z(K;B)$.

$$Z(K,B) := LU_T P^* K P L_P T U \approx \begin{bmatrix} \hat{X}_{RR} & 0 & 0 & 0 \\ 0 & \hat{X}_{SS} & \hat{X}_{SN} & K_{SF} \\ 0 & \hat{X}_{NS} & \hat{X}_{NN} & K_{NF} \\ 0 & K_{FS} & K_{FN} & K_{FF} \end{bmatrix}$$

In the next lecture, we will show how to use this decomposition recursively to build a fast algorithm to apply and invert large matrices.

References

[1] Gene Howard Golub et al. *Milestones in matrix computation: the selected works of Gene H. Golub with commentaries.* Oxford University Press, 2007.

[2] Kenneth L Ho and Leslie Greengard. "A fast direct solver for structured linear systems by recursive skeletonization". In: *SIAM Journal on Scientific Computing* 34.5 (2012), A2507–A2532.

[3] Per-Gunnar Martinsson and Vladimir Rokhlin. "A fast direct solver for boundary integral equations in two dimensions". In: *Journal of Computational Physics* 205.1 (2005), pp. 1–23.

[4] Victor Minden et al. "A recursive skeletonization factorization based on strong admissibility". In: *SIAM Journal of Multiscale Modeling and Simulation* (2017).