Computing (with) tensor train decompositions

Dario Fasino[1] and Eugene Tyrtyshnikov[2]

[1] *DMIF, Università di Udine, Udine, Italy.* dario.fasino@uniud.it
[2] *INM-RAS, Moscow, Russia.* eugene.tyrtyshnikov@gmail.com

The *tensor train decomposition* is a representation technique which allows compact storage and efficient computations with arbitrary tensors [2]. Basically, a tensor train (TT) decomposition of a d-dimensional tensor $\mathbf{A}$ with size $n_1 \times n_2 \times \cdots \times n_d$ is a sequence $G_1, \ldots, G_d$ of 3-tensors (the *carriages*); the size of G_i is $r_{i-1} \times n_i \times r_i$ with $r_0 = r_d = 1$ (that is, G_1 and G_d are ordinary matrices) and

$$\mathbf{A}(i_1, i_2, \ldots, i_d) = \sum_{\alpha_1, \ldots, \alpha_{d-1}} G_1(i_1, \alpha_1) G_2(\alpha_1, i_2, \alpha_2) \cdots G_d(\alpha_{d-1}, i_d).$$

The index α_i runs from 1 to r_i, for $i = 1, \ldots, d-1$, and the numbers $r_1, \ldots, r_{d-1}$ are the *TT-ranks* of $\mathbf{A}$. We will use the notation $\mathbf{A} = \mathrm{TT}(G_1, \ldots, G_d)$.

We present a backward error analysis of two algorithms found in [3] which perform computations with tensors in TT-format. The first one produces an exact or approximate TT-decomposition $G_1, \ldots, G_d$ of a tensor $\mathbf{A}$ given in functional form, depending on a tolerance ε. If $\varepsilon = 0$ then the output of the algorithm is an exact TT-decomposition, that is, $\mathbf{A} = \mathrm{TT}(G_1, \ldots, G_d)$. If $\varepsilon > 0$ then $\mathrm{TT}(G_1, \ldots, G_d)$ is an $\mathcal{O}(\varepsilon)$-approximation of $\mathbf{A}$ which can realize significant savings in memory space. The computational core of the algorithm is a suitable (approximate) matrix factorization that, in the original paper, relies on SVD computations. We prove that analogous performances and backward stability can be obtained by means of QR factorizations.

The second algorithm computes the *contraction* (multilinear form) of a tensor in TT-format and vectors $v_1, \ldots, v_d$,

$$a = \sum_{i_1=1}^{n_1} \cdots \sum_{i_d=1}^{n_d} \sum_{\alpha_1, \ldots, \alpha_{d-1}} G_1(i_1, \alpha_1) G_2(\alpha_1, i_2, \alpha_2) \cdots G_d(\alpha_{d-1}, i_d) v_1(i_1) \cdots v_d(i_d).$$

By means of known error bounds for inner products in floating point arithmetic [1], we prove backward stability of the proposed algorithm under very general hypotheses on the evaluation order of the innermost summations. More precisely, if $\mathbf{A} = \mathrm{TT}(G_1, \ldots, G_d)$ and no underflows or overflows are encountered then the output $\widehat{a}$ computed by the algorithm in floating point arithmetic is the exact contraction of $\widehat{\mathbf{A}} = \mathrm{TT}(G_1 + \Delta G_1, \ldots, G_d + \Delta G_d)$ and $v_1, \ldots, v_d$ where $|\Delta G_i| \le (n_i + r_{i-1}) u |G_i| + \mathcal{O}(u^2)$ and u is the machine precision.

References

[1] C.-P. Jeannerod, S. M. Rump, *Improved error bounds for inner products in floating-point arithmetic*, SIAM J. Matrix Anal. Appl., 34 (2013) 338–344.

[2] I. Oseledets, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011) 2295–2317.

[3] I. Oseledets, E. Tyrtyshnikov, *TT-cross approximation for multidimensional arrays*, Lin. Alg. Appl., 432 (2010) 70–88.