Christian B. Mendl, Richard M. Milbradt

Tutorial 4 (Canonical Polyadic (CP) decomposition for optimized matrix-matrix multiplication)
Using the outer product of vectors, the SVD of an $n \times n$ matrix $A = USV^\dagger$ can be represented as

$$A = \sum_{j=1}^{n} \sigma_j \, u_j \circ v_j^*,$$

where u_j and v_j are the column vectors of U and V, respectively. Note that the summation only needs to include the non-zero singular values, i.e., run from 1 to $r = \operatorname{rank}(A)$. Similarly, a tensor $T \in \mathbb{C}^{n_1 \times n_2 \times n_3}$ can be decomposed as

$$T = \sum_{j=1}^{r} u_j \circ v_j \circ w_j =: [\![U, V, W]\!],$$

with $U \in \mathbb{C}^{n_1 \times r}$, $V \in \mathbb{C}^{n_2 \times r}$, $W \in \mathbb{C}^{n_3 \times r}$ (not necessarily isometries). This is denoted the Canonical Polyadic (CP) decomposition. The minimal possible r defines the rank of the tensor.

(a) Does the CP decomposition always exist?

One usage of the CP decomposition is optimizing matrix-matrix multiplication, $C = AB$, with $A, B \in \mathbb{C}^{n \times n}$. A literal implementation of $c_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk}$ (for $i, k = 1, \dots, n$) has runtime $\mathcal{O}(n^3)$, but it turns out that this can be further improved. In the following, we assume that $n = 2^k$ is a power of 2, and use a recursive block partitioning into four blocks of size $2^{k-1} \times 2^{k-1}$ each, such that

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}.$$

(b) Working with this block partitioning, write out the algorithmic complexity of the naive matrix multiplication. What is the computationally most expensive step in this algorithm?

(c) We can enumerate the block indexing of the submatrices as $(1,1) \to 1$, $(1,2) \to 2$, $(2,1) \to 3$, $(2,2) \to 4$, allowing for a one-hot encoding e_1 to e_4. Based on that, assemble a tensor of degree 3, with the first two dimensions corresponding to the input block indices and the third to the output index.

(d) How does the rank of this tensor affect the complexity of the matrix-matrix multiplication?

Remark: It turns out that the tensor in (c) has rank 7. This leads to the *Strassen algorithm*, exploiting that only 7 block matrix multiplications are necessary at each level of recursion to perform the same operation. The algorithmic complexity is thus reduced from $\mathcal{O}(n^3)$ to $\mathcal{O}(7^k) = \mathcal{O}(n^{\log_2 7}) \approx \mathcal{O}(n^{2.80735})$.[1]

Solution

(a) Yes, this is possible. You can easily replace all three vectors with basis vectors multiplied by a constant.

(b) The naive algorithm is

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned}$$

This requires $8\left(\frac{n}{2}\right)^3 = \mathcal{O}(n^3)$ operations. The most expensive step is the matrix multiplication, which takes $\mathcal{O}(m^3)$ when multiplying two $m \times m$ matrixes.

[1] The up to date smallest-known exponent is $\mathcal{O}(n^{2.37286})$, see arxiv.org/abs/2010.05846.

(c) We can encode the 8 different matrix multiplications as follows:

$$C_{ik} \mathrel{+}= A_{ij} B_{jk} \to e_{ij} \circ e_{jk} \circ e_{ik}$$

This gives for the expression in part (b):

$$\begin{aligned}\mathcal{T} &= e_{11} \circ e_{11} \circ e_{11} + e_{12} \circ e_{21} \circ e_{11} \\ &+ e_{11} \circ e_{12} \circ e_{12} + e_{12} \circ e_{22} \circ e_{12} \\ &+ e_{21} \circ e_{11} \circ e_{21} + e_{22} \circ e_{21} \circ e_{21} \\ &+ e_{21} \circ e_{12} \circ e_{22} + e_{22} \circ e_{22} \circ e_{22}.\end{aligned}$$

(d) This tensor has rank less than 8 (actually rank 7), according to the following CP decomposition with 7 terms:

$$\begin{aligned}\mathcal{T} &= (e_{11} + e_{22}) \circ (e_{11} + e_{22}) \circ (e_{11} + e_{22}) \\ &+ (e_{21} + e_{22}) \circ e_{11} \circ (e_{21} - e_{22}) \\ &+ e_{11} \circ (e_{12} - e_{22}) \circ (e_{12} + e_{22}) \\ &+ e_{22} \circ (e_{21} - e_{11}) \circ (e_{11} + e_{21}) \\ &+ (e_{11} + e_{12}) \circ e_{22} \circ (-e_{11} + e_{12}) \\ &+ (e_{21} - e_{11}) \circ (e_{11} + e_{12}) \circ e_{22} \\ &+ (e_{12} - e_{22}) \circ (e_{21} + e_{22}) \circ e_{11}.\end{aligned}$$

Hence, we can then compute the matrix multiplication as follows:

$$\begin{aligned}M_1 &= (A_{11} + A_{22})\,(B_{11} + B_{22}) \\ M_2 &= (A_{21} + A_{22})\,B_{11} \\ M_3 &= A_{11}\,(B_{12} - B_{22}) \\ M_4 &= A_{22}\,(-B_{11} + B_{21}) \\ M_5 &= (A_{11} + A_{22})\,B_{22} \\ M_6 &= (-A_{11} + A_{21})\,(B_{11} + B_{12}) \\ M_7 &= (A_{12} - A_{22})\,(B_{21} + B_{22})\end{aligned}$$

The entries of C are then

$$\begin{aligned}C_{11} &= M_1 + M_4 - M_5 + M_7 \\ C_{12} &= M_3 + M_5 \\ C_{11} &= M_2 + M_4 \\ C_{11} &= M_1 - M_2 + M_3 + M_6\end{aligned}$$

The algorithmic complexity is thus reduced from $\mathcal{O}(n^3)$ to $\mathcal{O}(7^k) = \mathcal{O}(n^{\log_2 7}) \approx \mathcal{O}(n^{2.80735})$ when the shown steps are performed recursively.