# TT-SVD

Сергей Роговой

December 2022

## Вступление

A simple nonrecursive form of the tensor decomposition in d dimensions is presented. It does not inherently suffer from the curse of dimensionality, it has asymptotically the same number of parameters as the canonical decomposition, but it is stable and its computation is based on low rank approximation of auxiliary unfolding matrices. The new form gives a clear and convenient way to implement all basic operations efficiently.

## Motivation

Tensors are natural multidimensional generalizations of matrices and have attracted tremendous interest in recent years. Multilinear algebra, tensor analysis, and the theory of tensor approximations play increasingly important roles in computational mathematics and numerical analysis. An efficient representation of a tensor (by tensor we mean only an array with d indices) by a small number of parameters may give us an opportunity and ability to work with d-dimensional problems, with d being as high as 10, 100, or even 1000 (such problems appear in quantum molecular dynamics, stochastic partial differential equations, and financial modelling). Problems of such sizes cannot be handled by standard numerical methods due to the curse of dimensionality, since everything (memory, amount of operations) grows exponentially in d.

## Why we need something new. Canonical decomp. problems

There is an effective way to represent a large class of important d-dimensional tensors by using the canonical decomposition of a given tensor A with elements $A(i_1, ..., i_d)$

$$A(i_1, i_2, ..., i_d) = \sum_{\alpha=1}^{r} U1(i_1, \alpha) U2(i_2, \alpha) ... Ud(i_d, \alpha).$$

The minimal number of summands r required to express A in form is called the tensor rank (or the canonical rank). The matrices $Uk = [Uk(ik, \alpha)]$ are called canonical factors. But canonical decomposition suffers from several drawbacks. The computation of the canonical rank is an NP-hard problem, and the approximation with a fixed canonical rank in the Frobenius norm can be ill-posed; thus the numerical algorithms for computing an approximate representation in such cases might fail. That is why it is a good idea to look at the alternatives for the canonical format, which may have a larger number of parameters but are much better suited for the numerical treatment.

## Why we need something new. Tucker decomp. problems

Representation of tensor $A(i_1, ..., i_d)$ as

$$A(i_1, ..., i_d) = G(\alpha_1, ..., \alpha_d)U_1(i_1, \alpha_1)...U_d(i_d, \alpha_d)$$

is called a Tucker decomposition. $G(\alpha_1, ..., \alpha_d)$ is called the core of Tucker decomposition, and $U_1(i_1, \alpha_1)...U_d(i_d, \alpha_d)$ tucker decomposition matrices. The Tucker format is stable but has exponential in d number of parameters, $O(dnr + r^d)$. It is suitable for "small" dimensions, especially for the three-dimensional. For large d it is not suitable.

## Tensor Train

We approximate a given tensor B by a tensor $A \approx B$ with elements

$$A(i_1, i_2, ..., i_d) = G_1(i_1)G_2(i_2)...G_d(i_d)$$

where $G_k(i_k)$ is an $r_{k-1} \times r_k$ matrix. Compare with the definition of a rank-1 tensor: it is a quite straightforward block generalization of the rank-1 tensor. As will be shown in this paper, one of the differences between and the canonical decomposition is that the ranks $r_k$ can be computed as the ranks of certain auxiliary matrices. Lets write in the index form. Matrix $G_k(i_k)$ is actually a three-dimensional array, and it can be treated as an $r_{k-1} \times n_k \times r_k$ array with elements $G_k(\alpha_{k-1}, n_k, \alpha_k) = G_k(i_k)_{\alpha_{k-\mathbf{1}}\alpha_k}$ . In the index form the decomposition is written as:

$$A(i_1, .., i_d) = \sum_{\alpha_{\mathbf{0}},..,\alpha_{d-\mathbf{1}},\alpha_d} G_1(\alpha_0, i_1, \alpha_1)G_2(\alpha_1, i_2, \alpha_2)...G_d(\alpha_{d-1}, i_d, \alpha_d)$$

## Tensor train network

This graphical representation means the following. There are two types of nodes. Rectangles contain spatial indices (i.e., the indices $i_k$ of the original tensor) and some auxiliary indices $\alpha_k$, and a tensor with these indices is associated with such kind of nodes. Circles contain only the auxiliary indices $\alpha_k$ and represent a link: if an auxiliary index is present in two cores, we connect it. The summation over the auxiliary indices is assumed; i.e., to evaluate an entry of a tensor, one has to multiply all tensors in the rectangles and then perform the summation over all auxiliary indices. This picture looks like a train with carriages and links between them, and that justifies the name tensor train decomposition, or simply TT-decomposition. The ranks $r_k$ will be called compression ranks or TT-ranks, three-dimensional tensors $G_k$—cores of the TT-decomposition (analogous to the core of the Tucker decomposition)

TENSOR-TRAIN DECOMPOSITION 2297

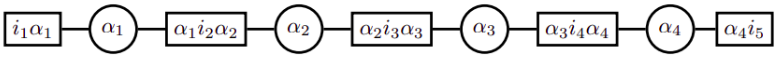


Fig. 1.1. *Tensor-train network.*

## Theorem 1

**Theorem 1** . If for each unfolding matrix $A_k = A_k(i_1, ..., i_k; i_{k+1}...i_d)$ of form of a d-dimensional tensor A, rank $A_k = r_k$ , then there exists a TT - decomposition with TT-ranks not higher than rk.

**Proof** Consider the unfolding matrix $A_1$. Its rank is equal to $r_1$; therefore it admits a skeleton decomposition $A_1 = UV^T$, or in the index form

$$A_1(i_1; i_2, ..., i_d) = \sum_{\alpha_1=1}^{r_1} U(i_1, \alpha_1)V(\alpha_1, i_2, ..., i_d).$$

The matrix V can be expressed as $V = A_1^T U(U^T U)^{-1} = A_1^T W$, or in the index form

$$V(\alpha_1, i_2, ..., i_d) = \sum_{i_1=1}^{n_1} A(i_1, ..., i_d)W(i_1, \alpha_1).$$

Now the matrix V can be treated as a (d - 1)-dimensional tensor V with $(\alpha_1 i_2)$ as one long index: $V = V(\alpha_1 i_2, i_3, ..., i_d)$. The process can be continued by induction. Consider V and separate the index $(\alpha_1, i_2)$ from others:

$$V(\alpha_1 i_2, i_3, ..., i_d) = \sum_{\alpha_2=1}^{r_2} G_2(\alpha_1, i_2, \alpha_2)V(\alpha_2 i_3, i_4, ..., i_d).$$

This yields the next core tensor $G_2(\alpha_1, i_2, \alpha_2)$ and so on, up to $G_d(\alpha_{d-1}, i_d,$ finally giving the TT-representation.

## Theorem 1

**Continue** Now consider its unfolding matrices $V_2, ..., V_d$. We will show that rank $V_k \leq r_k$ holds. Indeed, for the kth mode the TT-rank is equal to $r_k$; therefore A can be represented as

$$A(i_1, ..., i_d) = \sum_{\beta=1}^{r_k} F(i_1, ..., i_k, \beta) G(\beta, i_{k+1}, ..., i_d).$$

Using that, we obtain

$$V_k = V(\alpha_1 i_2, ..., i_k; i_{k+1} ..., i_d) = \sum_{i_1=1}^{n_1} \sum_{\beta=1}^{r_k} W(i_1, \alpha_1) F(i_1, ..., i_k, \beta) G(\beta, i_{k+1}, ..., i_d)$$

$= \sum_{\beta=1}^{r_k} H(\alpha_1 i_2, ..., i_k, \beta) G(\beta, i_{k+1}, ..., i_d)$
where $H(\alpha_1 i_2, ..., i_k, \beta) = \sum_{i_1=1}^{n_1} F(i_1, ..., i_k, \beta) W(i_1, \alpha_1)$. Row and column indices of $V_k$ are now separated and rank $V_k \leq r_k$.

## Theorem 2

**Theorem 2** Suppose that the unfoldings $A_k$ of the tensor A satisfy $A_k = R_k + E_k$, rank $R_k = r_k, ||Ek||_F = \epsilon_k, k = 1, ..., d1$ Then TT-SVD computes a tensor B in the TT-format with TT-ranks $r_k$ and $||A - B||_F \leq \sqrt{\sum_{k=1}^{d-1} \epsilon_k^2}$.

**Proof** The proof is by induction. For d = 2 the statement follows from the properties of the SVD. Consider an arbitrary d > 2. Then the first unfolding A1 is decomposed as $A_1 = U_1 \Sigma V_1 + E_1 = U_1 B_1 + E_1$ , where $U_1$ is of size $n_1 \times r_1$, has orthonormal columns, and $||E1|| = \epsilon_1$. The matrix $B_1$ is naturally associated with a (d - 1)-dimensional tensor $B_1$ with elements $B(\alpha_1 i_2, i_3, ..., i_d)$, which will be decomposed further in the TT-SVD algorithm. This means that B1 will be approximated by some other matrix $\hat{B}_1$. From the properties of the SVD it follows that $U_1^T E_1 = 0$ , and thus

$$||A - B||_F^2 = ||A_1 - U_1 \hat{B}_1||_F^2 = ||A_1 - U_1(\hat{B}_1 + B_1 - B_1)||_F^2 = ||A_1 - U_1 B_1||_F^2$$

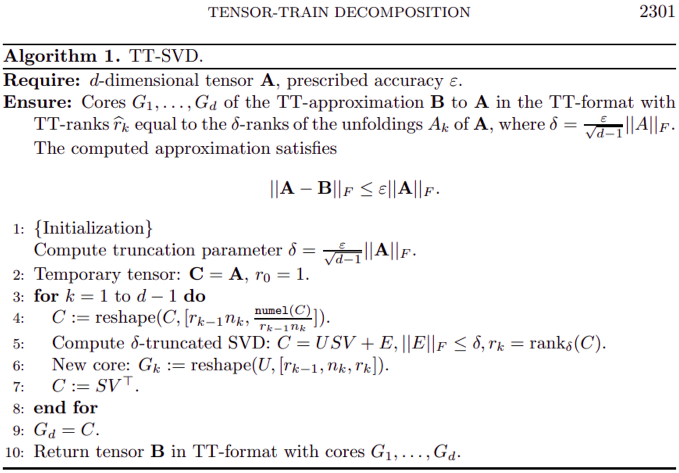
$+ ||\mathrm{U}_1(B_1 - \hat{B}_1)||_F^2$

and since $U_1$ has orthonormal columns, $||A - B||_F^2 \leq \epsilon_1^2 + ||B_1 - \hat{B}_1||_F^2$. The matrix $B_1$ is easily expressed from $A_1$, $B_1 = U_1^T A_1$, and thus it is not difficult to see from the orthonormality of columns of $U_1$ that the distance of the kth unfolding (k = 2,...,d - 1) of the (d - 1)-dimensional tensor $B_1$ to the $r_k$th rank matrix cannot be larger then $\epsilon_k$. Proceeding by induction, we have

$$||B_1 - \hat{B}_1||_F^2 \leq \sum_{k=2}^{d-1} \epsilon_k^2,$$

and together with, this completes the proof.

# TT-SVD algorithm



**Algorithm 1.** TT-SVD.

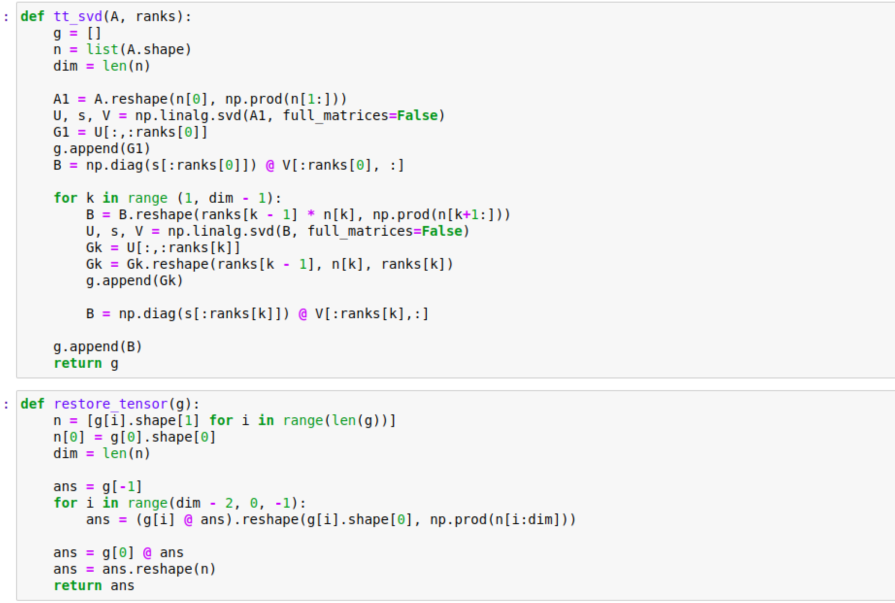
**Require:** $d$-dimensional tensor $\mathbf{A}$, prescribed accuracy $\varepsilon$.

**Ensure:** Cores $G_1, \dots, G_d$ of the TT-approximation $\mathbf{B}$ to $\mathbf{A}$ in the TT-format with TT-ranks $\widehat{r}_k$ equal to the $\delta$-ranks of the unfoldings $A_k$ of $\mathbf{A}$, where $\delta = \frac{\varepsilon}{\sqrt{d-1}} ||A||_F$. The computed approximation satisfies

$$||\mathbf{A} - \mathbf{B}||_F \leq \varepsilon ||\mathbf{A}||_F.$$

```
1: {Initialization}
   Compute truncation parameter $\delta = \frac{\varepsilon}{\sqrt{d-1}} ||\mathbf{A}||_F$.
2: Temporary tensor: $\mathbf{C} = \mathbf{A}$, $r_0 = 1$.
3: for $k = 1$ to $d - 1$ do
4:    $C := \text{reshape}(C, [r_{k-1} n_k, \frac{\texttt{numel}(C)}{r_{k-1} n_k}])$.
5:    Compute $\delta$-truncated SVD: $C = USV + E, ||E||_F \leq \delta, r_k = \text{rank}_\delta(C)$.
6:    New core: $G_k := \text{reshape}(U, [r_{k-1}, n_k, r_k])$.
7:    $C := SV^\top$.
8: end for
9: $G_d = C$.
10: Return tensor $\mathbf{B}$ in TT-format with cores $G_1, \dots, G_d$.
```

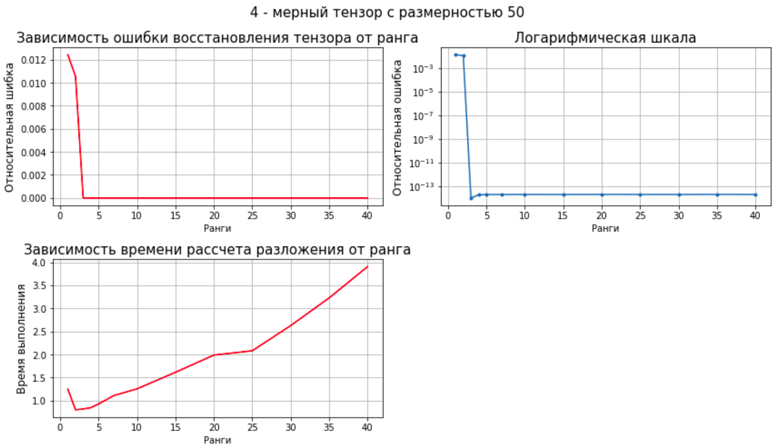
## Code



```
: def tt_svd(A, ranks):
      g = []
      n = list(A.shape)
      dim = len(n)

      A1 = A.reshape(n[0], np.prod(n[1:]))
      U, s, V = np.linalg.svd(A1, full_matrices=False)
      G1 = U[:,:ranks[0]]
      g.append(G1)
      B = np.diag(s[:ranks[0]]) @ V[:ranks[0], :]

      for k in range (1, dim - 1):
          B = B.reshape(ranks[k - 1] * n[k], np.prod(n[k+1:]))
          U, s, V = np.linalg.svd(B, full_matrices=False)
          Gk = U[:,:ranks[k]]
          Gk = Gk.reshape(ranks[k - 1], n[k], ranks[k])
          g.append(Gk)

          B = np.diag(s[:ranks[k]]) @ V[:ranks[k],:]

      g.append(B)
      return g
```

```
: def restore_tensor(g):
      n = [g[i].shape[1] for i in range(len(g))]
      n[0] = g[0].shape[0]
      dim = len(n)

      ans = g[-1]
      for i in range(dim - 2, 0, -1):
          ans = (g[i] @ ans).reshape(g[i].shape[0], np.prod(n[i:dim]))

      ans = g[0] @ ans
      ans = ans.reshape(n)
      return ans
```

## Graphs


4 - мерный тензор с размерностью 50
Зависимость ошибки восстановления тензора от ранга
Относительная шибка
0.012
0.010
0.008
0.006
0.004
0.002
0.000
0
5
10
15
20
25
30
35
40
Ранги
Логарифмическая шкала
Относительная ошибка
$10^{-3}$
$10^{-5}$
$10^{-7}$
$10^{-9}$
$10^{-11}$
$10^{-13}$
Ранги
Зависимость времени рассчета разложения от ранга
Время выполнения
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Ранги

## Thanks for attention