SIAM J. SCI. COMPUT.
Vol. 42, No. 6, pp. A3812–A3824

TENSOR COMPLETION VIA GAUSSIAN PROCESS–BASED INITIALIZATION[*]

YERMEK KAPUSHEV[†], IVAN OSELEDETS[†], AND EVGENY BURNAEV[†]

Abstract. In this paper, we consider the tensor completion problem representing the solution in the tensor train (TT) format. It is assumed that the tensor is of high order, and tensor values are generated by an unknown smooth function. The assumption allows us to develop an efficient initialization scheme based on Gaussian process regression and the TT-cross approximation technique. The proposed approach can be used in conjunction with any optimization algorithm that is usually utilized in tensor completion problems. We empirically justify that in this case the reconstruction error improves compared to the tensor completion with random initialization. As an additional benefit, our technique automatically selects rank thanks to using the TT-cross approximation technique.

Key words. tensor completion, Gaussian processes, tensor train, cross-approximation

AMS subject classifications. 68W25, 65F99, 60G15

DOI. 10.1137/19M1306518

1. Introduction.

In this paper we consider the tensor completion problem. We suppose that values of tensor $\mathcal{X}$ are generated by some smooth function, i.e.,

$$\mathcal{X}_{i_1,\dots,i_d} = f(x_{i_1}, \dots, x_{i_d}),$$

where $(x_{i_1}, \dots, x_{i_d})$ is a point on some multidimensional grid and $f(\cdot)$ is some unknown smooth function. However, the tensor values are known only at some small subset of the grid. The task is to complete the tensor, i.e., to reconstruct the tensor values at all points on the grid taking into account the properties of the *data generating process* $f(\cdot)$.

This problem statement differs from the traditional problem statement, which does not use any assumptions about the function $f(\cdot)$. Knowing some properties of the data generating function provides insights about how the tensor values relate to each other, and this, in turn, allows us to improve the results. In this work we assume that function $f(\cdot)$ is smooth.

There are a lot of practical applications that suit the statement, for example, modeling of physical processes, solutions of differential equations, and modeling probability distributions.

In this paper we propose modeling the smoothness of the data generating process by using Gaussian process (GP) regression. In GP regression the assumptions about the function that we approximate are controlled via the kernel function. The GP regression model is then used to construct the initial solution to the tensor completion problem.

In principle, such initialization can improve any other tensor completion technique. This means that using the proposed initialization state-of-the-art results can

[*] Submitted to the journal's Methods and Algorithms for Scientific Computing section December 12, 2019; accepted for publication (in revised form) August 24, 2020; published electronically December 14, 2020.
https://doi.org/10.1137/19M1306518
Funding: This work was supported by the Ministry of Education and Science of the Russian Federation through grant 14.756.31.0001.

[†] Center for Computational and Data-Intensive Science and Engineering, Skolkovo Institute of Science and Technology, Moscow, 121205, Russia (y.kapushev@skoltech.ru, i.oseledets@skoltech.ru, e.burnaev@skoltech.ru).

be obtained by employing some simple optimization procedure like stochastic gradient descent.

When the tensor order is high, the problem should be solved in some low-rank format because the number of elements of the tensor grows exponentially. The proposed approach is based on the tensor train (TT) format for its computational efficiency and ability to handle large dimensions [15].

The contributions of this paper are as follows:

– We introduce a new initialization algorithm which takes into account the tensor generating process. The proposed algorithm is described in section 3.
– The proposed initialization technique automatically selects the rank of the tensor; the details are given in subsection 3.2.
– We conducted empirical evaluations of the proposed approach and compared it with tensor completion techniques without our initialization. The results are given in section 4 and show the superiority of the proposed algorithm.

2. Tensor completion.

The formal problem statement is as follows. Suppose that $\mathcal{Y}$ is a d-way tensor, $\mathcal{Y} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ (by tensor here we mean a multidimensional array). Tensor values are known only at some subset of indices $\Omega \subset \{1, \ldots, n_1\} \times \cdots \times \{1, \ldots, n_d\}$. By P_Ω we denote the projection onto the set Ω, i.e.,

$$P_\Omega \mathcal{X} = \mathcal{Z}, \quad \mathcal{Z}(i_1, i_2, \ldots, i_d) = \begin{cases} \mathcal{X}(i_1, i_2, \ldots, i_d) & \text{if } (i_1, \ldots, i_d) \in \Omega, \\ 0 & \text{otherwise.} \end{cases}$$

We formulate the tensor completion as an optimization problem,

$$\begin{aligned} &\min_{\mathcal{X}} && f(\mathcal{X}) = \|P_\Omega \mathcal{X} - P_\Omega \mathcal{Y}\|_F^2 \\ &\text{subject to} && \mathcal{X} \in \mathcal{M}_r = \{\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_d} \mid \operatorname{rank}_{TT}(\mathcal{X}) = \mathbf{r}\}, \end{aligned} \tag{2.1}$$

where $\operatorname{rank}_{TT}(\mathcal{X})$ is a tensor train rank of $\mathcal{X}$ [16], which is a generalization of the matrix rank, and $\|\cdot\|_F$ is the Frobenius norm. A tensor $\mathcal{X}$ is said to be in TT format if its elements are represented as

$$\mathcal{X}(i_1, \ldots, i_d) = \sum_{j_1, j_2, \ldots j_d} \mathcal{G}^{(1)}_{1, i_1, j_1} \mathcal{G}^{(2)}_{j_1, i_2, j_2} \cdots \mathcal{G}^{(d)}_{j_{d-1}, i_d, 1},$$

where $\mathcal{G}^{(i)}$ is a 3-way tensor core with size $r_{i-1} \times n_i \times r_i$, $r_0 = r_d = 1$. Vector $\mathbf{r}_{TT} = (r_0, \ldots, r_d)$ is called TT-rank.

TT format assumes that the full tensor can be approximated by a set of 3-way core tensors; the total number of elements in core tensors is $\mathcal{O}(dnr^2)$, where $r = \max_{i=0,\ldots,d}\{r_i\}$, $n = \max_{i=1,\ldots,d}\{n_i\}$, which is much smaller than n^d.

In problem (2.1) we optimize the objective function straightforwardly with respect to tensor cores $\mathcal{G}^{(1)}, \ldots, \mathcal{G}^{(d)}$ while keeping their sizes fixed. Problem (2.1) is nonconvex, so optimization methods can converge to a local minimum. To get an efficient solution we impose two requirements:

1. Initial tensor $\mathcal{X}_0$ in TT format that is as close to the optimum as possible.
2. Availability of an efficient optimization procedure that will be launched from the obtained initial tensor.

These steps are independent, and one can apply any desired algorithm in each of them.

In this work we develop the initialization algorithm that allows obtaining accurate initial tensor for the case when the tensor of interest is generated by some smooth function. The experimental section demonstrates that our initialization can improve the results of many optimization procedures and shows the potential of our approach to be adapted to a large number of different tensor completion techniques.

3. Initialization.

We consider tensors that are generated by some function, i.e., tensor values are computed as follows:

$$\mathcal{Y}_{i_1,\dots,i_d} = f(x_{i_1}, \dots, x_{i_d}),$$

where $f(\cdot)$ is some unknown smooth function and $(x_{i_1}, \dots, x_{i_d})^\top \in \mathbb{R}^d$, $i_k = 1, \dots, n_k$, $n_1, \dots, n_d$, are tensor sizes. The set of points $\{(x_{i_1}, \dots, x_{i_d}) : i_k = 1, \dots, n_k; k = 1, \dots, d\}$ is a full factorial design of experiments, i.e., a multidimensional grid, and we also assume that the grid is uniform.

In this setting the tensor completion can be considered as a regression problem and can be solved by any regression technique that guarantees the smoothness of the solution. However, in the tensor completion problem we are interested in a tensor of values of $f(\cdot)$ at a predefined finite grid of points. The tensor should be in a low-rank format to be able to perform various operations with tensor efficiently (e.g., calculation of the norm of the tensor, dot product, and other).

These observations give us the solution—build regression model $\widehat{f}$ using the observed values of the tensor, then use the obtained approximation as a black-box for the TT cross-approximation algorithm [15]. The last step results in a tensor $\widehat{\mathcal{X}}$ in TT format, which is a low-rank format and allows efficient computations. The next step (which is optional) is to improve the obtained solution $\widehat{\mathcal{X}}$ by using it as initialization for any other tensor completion technique.

Let us write down the set of observed tensor values into a vector $\mathbf{y}$ and the corresponding indices into a matrix $\mathbf{X}$ (each row is a vector of indices $(i_1, i_2, \dots, i_d)$). Then the approach for tensor completion (in TT format) can be written as follows:

1. Construct initial tensor $\mathcal{X}_0$ in TT format:
   (a) Apply some regression technique using a given data set $(\mathbf{X}, \mathbf{y})$ to construct approximation of the function that generates tensor values.
   (b) Apply the TT-cross method (see subsection 3.2 or [15]) to the constructed approximation to obtain $\mathcal{X}_0$.
2. Apply some tensor completion technique using $\mathcal{X}_0$ as an initial value.

At step 1(a) the choice of the regression technique affects the result of the initialization, although it can be arbitrary. It is required to choose the regression algorithm such that it will capture the peculiarities of the tensor we would like to restore. In this work we suppose that the tensor generating function is smooth (which is a common situation when modeling physical processes). Therefore, we choose a regression technique that is good at approximating smooth functions. A reasonable choice, in this case, is to use Gaussian process (GP) regression [17]. GP models are a favorite tool in many engineering applications as they have proved to be efficient, especially for problems where it is required to model some smooth function [3]. The points $(x_{i_1}, \dots, x_{i_d})$ are not given; all we know is that at the point with multi-index $(i_1, \dots, i_d)$ on the grid the function value is equal to $\mathcal{X}_{i_1,\dots,i_d}$. To make the problem statement reasonable we assume that the indices are connected with the points as follows: $x_{i_k} = a_k i_k + b_k$, where $a_k, b_k \in \mathbb{R}$. So, as an input for the approximation we set a_k and b_k such that $x_{i_k} \in [0, 1]$.

At step 1(b) we use TT-cross because it allows us to efficiently approximate a black-box function by a low-rank tensor in TT format. Moreover, this approach can automatically select TT-rank, making it more desirable. More details on the technique are given in subsection 3.2.

The described approach has the following benefits:

1. Initial tensor $\mathcal{X}_0$ which is close to the optimal value in terms of the reconstruction error at observed values. It will push the optimization to faster convergence.
2. Better generalization ability—there are many degrees of freedom: a lot of different TT factors can give low reconstruction error at observed positions but can give a large error at other locations. An accurate approximation model will push the initial tensor to be closer to the original tensor in both the observed positions and unobserved ones.
3. TT-cross technique chooses rank automatically, so there is no need to tune the rank of the tensor manually.

The described approach leads to Algorithm 3.1. Steps 3 and 4 of the algorithm are described in subsections 3.1 and 3.2, respectively.

Algorithm 3.1. Initialization.

```
Input: $\mathbf{y}, \Omega$
Output: $\mathcal{Y}_0$ in TT format
    1. Construct the training set $(\mathbf{X}, \mathbf{y})$ from $\mathbf{y}, \Omega$
    2. Rescale inputs $\mathbf{X}$ to $[0, 1]$ interval
    3. Using $(\mathbf{X}, \mathbf{y})$ build GP model $\hat{f}(\mathbf{x})$        ▷ see subsection 3.1 for details
    4. Apply TT-cross to $\hat{f}(\mathbf{x})$ and obtain $\mathcal{Y}_0$          ▷ see subsection 3.2 for details
return $\mathcal{Y}_0$
```

3.1. Gaussian process regression. One of the most efficient tools for approximating smooth functions is the GP regression [6]. GP regression is a Bayesian approach where a prior distribution over continuous functions is assumed to be a Gaussian process, i.e.,

$$\mathbf{y} \,|\, \mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \, \mathbf{K}_f + \sigma^2_{noise}\mathbf{I}),$$

where $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ is a vector of outputs, $\mathbf{X} = (\mathbf{x}_1^\top, \mathbf{x}_2^\top, \ldots, \mathbf{x}_N^\top)^\top$ is a matrix of inputs, $\mathbf{x}_i \in \mathbb{R}^d$, σ^2_{noise} is a noise variance, $\boldsymbol{\mu} = (\mu(\mathbf{x}_1), \mu(\mathbf{x}_2), \ldots, \mu(\mathbf{x}_N))$ is a mean vector modeled by some function $\mu(\mathbf{x})$, $\mathbf{K}_f = \{k(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1}^N$ is a covariance matrix for some a priori selected covariance function k, and $\mathbf{I}$ is an identity matrix. An example of such a function is a squared exponential kernel

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\sum_{i=1}^{d}\left(\frac{\mathbf{x}^{(i)} - \mathbf{x}'^{(i)}}{\sigma_i}\right)^2\right),$$

where $\sigma_i, i = 1, \ldots, d$, are parameters of the kernel (hyperparameters of the GP model). The hyperparameters should be chosen according to the given data set.

Without loss of generality we make the standard assumption of zero-mean data. Now, for a new unseen data point $\mathbf{x}_*$ we have

$$\begin{aligned} \hat{f}(\mathbf{x}_*) &\sim \mathcal{N}\left(\mu(\mathbf{x}_*), \sigma^2(\mathbf{x}_*)\right), \\ \mu(\mathbf{x}_*) &= \mathbf{k}(\mathbf{x}_*)^\top \mathbf{K}_y^{-1}\mathbf{y}, \\ \sigma^2(\mathbf{x}_*) &= k(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}(\mathbf{x}_*)^\top \mathbf{K}_y^{-1}\mathbf{k}(\mathbf{x}_*), \end{aligned} \tag{3.1}$$

where $\mathbf{k}(\mathbf{x}_*) = (k(\mathbf{x}_*, \mathbf{x}_1), \ldots, k(\mathbf{x}_*, \mathbf{x}_N))^T$ and $\mathbf{K}_y = \mathbf{K}_f + \sigma^2_{noise}\mathbf{I}$.

Let us denote the vector of hyperparameters $\sigma_i, i = 1, \dots, d$, σ_f, and σ_{noise} by $\boldsymbol{\theta}$. To choose the hyperparameters of our model we consider the log-likelihood

$$\log p(\mathbf{y} \mid \mathbf{X}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{y}^T \mathbf{K}_y^{-1}\mathbf{y} - \frac{1}{2}\log|\mathbf{K}_y| - \frac{N}{2}\log 2\pi$$

and maximize it over the hyperparameters [17]. The runtime complexity of learning GP regression is $\mathcal{O}(N^3)$ as we need to calculate the inverse of $\mathbf{K}_y$, its determinant, and its derivatives of the log-likelihood. If the sample size N ($|\Omega|$ in our case) is large, the computational complexity becomes an issue. There are several ways to overcome it. If the data set has a factorial structure (multidimensional grid in a simple case), we can use the algorithm from [4]. If the structure is factorial with a small number of missing values, the method from [5] should be applied. For general unstructured cases, the approximate GP model can be built using, for example, the model described in [14], or using a subsample as a training set.

After tuning of the hyperparameters, we can use the mean of the posterior distribution (3.1) as a prediction of the model.

Note that the input points $\mathbf{X}$ in our case is a set of indices of the observed values $\mathbf{y}$. For the GP model we scale each index to the $[0, 1]$ interval.

3.1.1. Kernel choice and smoothness assumption. We approximate function f using the GP model. The GP model is a function from some reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ which is fully defined by the kernel function. In the ideal case the function f should be from the Hilbert space $\mathcal{H}$. However, for a given kernel function it can be difficult to identify what functions lie in the corresponding RKHS.

In practice GP models with popular kernels (radial basis function (RBF) kernel, Matérn kernel) provide good results, when $f \in C^k$, $k \geq 1$. So, in the experiments section we assume that the functions are from this class of functions and use the RBF kernel.

3.2. Tensor train cross-approximation. To approximate tensor $\widehat{\mathcal{X}}$ generated by $\hat{f}$ we use TT cross-approximation. First, let us consider the matrix case. Suppose that we are given a rank-r matrix $\mathbf{A}$ of size $m \times n$. A cross-approximation for the matrix is represented as

$$\mathbf{A} = \mathbf{C}\widehat{\mathbf{A}}^{-1}\mathbf{R},$$

where $\mathbf{C} = \mathbf{A}(:, J), \mathbf{R} = \mathbf{A}(I, :)$ are some r columns and rows of the matrix $\mathbf{A}$ and $\widehat{\mathbf{A}} = A(I, J)$ is the submatrix on the intersection of these rows and columns. To construct accurate approximation it is required to find submatrix $\widehat{\mathbf{A}}$ of large volume. It can be done in $\mathcal{O}(nr^2)$ operations [21].

Now for tensor $\widehat{\mathcal{X}} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ the procedure is the following. At the first step let us consider unfolding $\mathbf{X}_1$ of size $n_1 \times n_2 n_3 \times \cdots \times n_d$ and rank r_1. Using the row-column alternating algorithm from [21] we can find r_1 linearly independent columns of matrix $\mathbf{X}_1$; these columns form matrix $\mathbf{C}$. After that, applying the maxvol procedure [21] to the matrix $\mathbf{C}$, we can find the set of row indices $I_1 = \left[i_1^{\alpha_1}\right], \alpha_1 = 1, \dots, r_1$, matrix $\mathbf{R}$, and matrix $\widehat{\mathbf{A}}_1$ that will give the cross-approximation of unfolding $\mathbf{X}_1$:

$$\mathbf{X}_1 = \mathbf{C}\widehat{\mathbf{A}}_1^{-1}\mathbf{R}.$$

We set

$$\mathbf{G}_1 = \mathbf{C}\widehat{\mathbf{A}}_1^{-1},$$

where $\mathbf{G}_1$ is of size $n_1 \times r_1$. Next, let us form tensor $\mathcal{R}$ from r_1 rows of $\mathbf{X}_1$,

$$\mathcal{R}(\alpha_1, i_2, \dots, i_d) = \widehat{\mathcal{X}}(i_1^{\alpha_1}, i_2, \dots, i_d),$$

and reshape it into a tensor of size $r_1 n_2 \times n_3 \times \cdots \times n_d$. The next step is to apply the same procedure to the unfolding $\mathbf{R}_1$ of the tensor $\mathcal{R}$ and obtain the matrices $\mathbf{C}$, $\widehat{\mathbf{A}}_2$, and

$$\mathbf{G}_2 = \mathbf{C}\widehat{\mathbf{A}}_2^{-1}$$

of size $r_1 n_2 \times r_2$.

Repeating the described procedure d times, we will end up with matrices $\mathbf{G}_1$, $\mathbf{G}_2, \dots, \mathbf{G}_d$ of sizes $n_1 \times r_1, r_1 n_2 \times r_2, \dots, r_{d-1} n_d \times 1$. Then each matrix can be reshaped to the 3-way tensor of size $r_{d-1} \times n_d \times r_d$, $r_0 = r_d = 1$ and can be used as core tensors for the TT format. It turns out that such a representation is a TT decomposition of the initial tensor $\widehat{\mathcal{X}}$.

The exact ranks $r_1, \dots, r_d$ are not known to us in general. They can only be estimated from the above (e.g., by the maximum rank of the corresponding unfolding). If the rank is overestimated, then the calculation of matrices $\mathbf{G}_i$ is an unstable operation (because we obtain almost rank-deficient unfolding matrices). However, in [15] the authors suggest some simple modification that overcomes this issue. Therefore, we need to estimate the ranks from the above, but the estimate should not be much larger than the real rank. So, the approach is to start from some small rank, construct the tensor in TT format, and then apply recompression (see [16]). If there is a rank that is not reduced, then we underestimated that rank and should increase it and repeat the procedure.

3.3. Computational complexity. The computational complexity of the TT cross-approximation method is as follows. To perform the procedure we need to evaluate the GP model $\mathcal{O}(dnr^2)$ times at some subset of grid points and then perform $\mathcal{O}(dnr^3)$ operations to find all maximum volume submatrices. The complexity of evaluating the GP model at one point is $\mathcal{O}(Nd)$, where N is the number of observed tensor elements. The total complexity is thus $\mathcal{O}(Nd^2nr^2 + dnr^3)$.

4. Experimental results.

In this section we present the results of the application of our approach to two engineering problems and also test it on some artificial problems to investigate how its properties depend on smoothness.

The experimental setup is the following. We try the following optimization algorithms:

1. SGD – stochastic gradient descent [25];
2. Ropt – Riemannian optimization [19];
3. TTWopt – weighted tensor train optimization [25];
4. ALS – alternating least squares [9].

We run each algorithm with random initialization and with the proposed GP-based initialization and then compare results.

4.1. Functions generated from GP prior. In order to study the dependence of the solution on the smoothness of the generating function, we applied the proposed approach to the toy functions generated from GP prior with different kernels. The smoothness of the generated functions is the same as the kernel that we used to generate them. Thus, we can investigate the performance of the approach for different smoothnesses. We considered shift-invariant kernels, i.e., $k(\mathbf{x}, \mathbf{y}) = k(r)$, where $r = \|\mathbf{x} - \mathbf{y}\|$. The list of kernels is as follows:

- Exponential kernel,

$$k(r) = \exp\left(-\frac{r}{\sigma}\right).$$

The kernel is not differentiable at $\mathbf{x} = \mathbf{y} = 0$, and thus the functions generated with this kernel are from C^0.

- Matern$_{3/2}$,

$$k_{3/2}(r) = \left(1 + \frac{\sqrt{3}r}{\sigma}\right) \exp\left(-\frac{\sqrt{3}r}{\sigma}\right).$$

This kernel is 1-time differentiable.

- Matern$_{5/2}$,

$$k_{5/2}(r) = \left(1 + \frac{\sqrt{5}r}{\sigma} + \frac{5r^2}{3\sigma^2}\right) \exp\left(-\frac{\sqrt{5}r}{\sigma}\right).$$

This kernel is 2-times differentiable.

- Radial basis function (RBF) kernel,

$$k(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right).$$

This kernel is infinitely differentiable.

Note that despite the functions having been generated using different kernel functions, in the proposed approach we used the RBF kernel.

To compare how much one approach is better than the other, we calculate the relative mean squared error

$$\mathrm{MSE}_{\mathrm{rel}} = \frac{1}{|\Omega_{test}|} \left\| \frac{P_{\Omega_{test}} \hat{\mathcal{Y}} - P_{\Omega_{test}} \mathcal{Y}}{\hat{\sigma}} \right\|_F^2,$$

where Ω_{test} is some set of indices independent of the given observed set of indices Ω, $|\Omega_{test}|$ is a size of the set Ω_{test}, $\widehat{\mathcal{Y}}$ is an obtained approximation of the actual tensor $\mathcal{Y}$, and $\hat{\sigma}$ is a standard deviation of $P_{\Omega_{test}} \mathcal{Y}$. Such an error can be interpreted as the ratio of unexplained variance. For each optimization technique we calculate the difference between the error obtained using random initialization and the error obtained using GP-based initialization.

For each kernel function we generated several data sets with different numbers of observed points ($N \in \{100, 500, 1000, 2000, 5000\}$) and different dimensionalities ($d \in \{2, 3, \dots, 10, 11, 13, \dots, 19\}$). The quantity is illustrated in Figure 1 (note that we clamp the values to the $[-1, 1]$ interval to make the figures more illustrative). It shows the improvement of one initialization over another. We can see from the figure that in most of the cases GP-based initialization gives high improvement in the relative error. The only exception is the TTWopt method for which the benefit of the proposed initialization scheme takes place only in about half of the cases. Note also the white squares for TTWopt, which indicate that implementation of TTWopt crashed for the given data set (for some unknown reason).

4.2. Read-world functions. We compared the approaches on two real-world problems: the CMOS oscillator model and the cookie problem (see, respectively, subsections 4.3 and 4.4). In the CMOS oscillator problem we run each optimization 10 times with different random training sets and then calculate the average reconstruction error as well as standard deviation. The cookie problem is more computationally intensive because each evaluation of the tensor value takes more resources and time. Therefore, for the cookie problem we performed 10 runs, and the training set was the same during all runs.

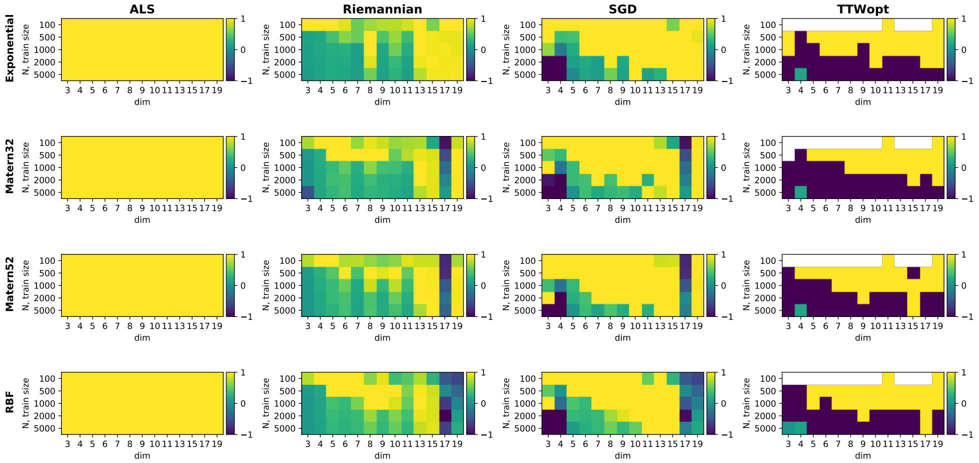

FIG. 1. *Improvement of GP-based initialization over random initialization for different tensors and optimization methods. We clamp the improvement value to the* $[-1, 1]$ *interval to make plots more illustrative.*

The quality of the methods is measured using mean squared error

$$\mathrm{MSE} = \frac{1}{|\Omega_{test}|} \| P_{\Omega_{test}} \widehat{\mathcal{Y}} - P_{\Omega_{test}} \mathcal{Y} \|_F^2 .$$

We also report the error of the initial tensor for random initializations and the proposed initializations for each problem.

Note that when we use GP-based initialization, the TT-rank $\mathbf{r}_{\mathrm{TT}}$ of the tensor is selected automatically by the TT-cross algorithm, and the max value of $\mathbf{r}_{\mathrm{TT}}$ can be larger than n. The optimization algorithms with random initialization do not have a procedure for automatic rank selection, so we ran them with different ranks (from 1 to $\min_k n_k$) and then chose the best one.

TTWopt implementation[1] does not support high-dimensional problems. For higher-dimensional problems the authors of TTWopt propose using SGD. The authors of TTWopt also propose truncated SVD-based initialization. The idea is to fill missing values using the mean value of the observed part of the tensor and then to apply truncated SVD to obtain TT cores. However, such an approach is only applicable to low-dimensional tensors as it requires one to calculate full matrices of large size.

For Ropt and ALS we used publicly available MATLAB codes.[2]

4.3. Cookie problem. Let us consider a parameter-dependent PDE [2, 20]:

$$\begin{aligned} -\mathrm{div}(a(x,p)\nabla u(x,p)) &= 1, \quad x \in D = [0,1]^2, \\ u(x,p) &= 0, \quad x \in \partial D, \end{aligned}$$

where

$$a(x,p) = \begin{cases} p_\mu & \text{if } x \in D_{s,t}, \mu = mt + s, \\ 1 & \text{otherwise}, \end{cases}$$

[1] https://github.com/yuanlonghao/T3C_tensor_completion
[2] https://anchp.epfl.ch/index-html/software/ttemps/

$D_{s,t}$ is a disk of radius $\rho = \frac{1}{4m+2}$, and m^2 is a number of disks which form an $m \times m$ grid. This is a heat equation where heat conductivity $a(x,p)$ depends on x (see Figure 2) and p is m^2-dimensional.

We are interested in average temperature over D: $u(p) = \int_{[0,1]^2} u(x,p)\mathrm{d}x$. If p takes 10 possible values, then there are $\mathbf{10^{m^2}}$ possible values of $u(p)$.

In this work we used the following setup for the Cookie problem: each parameter p lies in the interval $[0.01, 1]$, the number of levels for each p is 10, the number of cookies is $m^2 = 9$ and 16, the size of the observed set is $N = 5000$, and for the test set we used 10000 independently generated points.

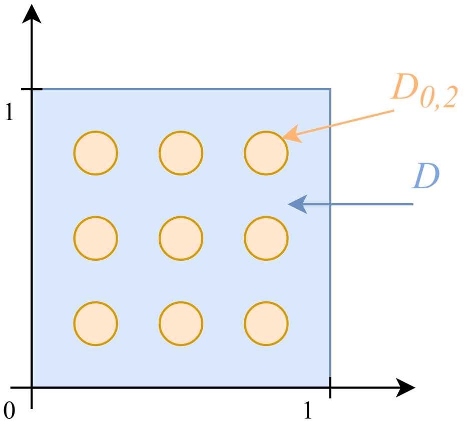

FIG. 2. *Illustration of the cookie problem with $m = 3$ (9 cookies).*

The results of tensor completion are presented in Table 1 (the variance of the initialization error for GP-based initialization is not presented as it is negligible in this case). One can see that GP-based initialization gives lower reconstruction errors both on the training set and test set except for the ALS technique. The ALS method with the proposed initialization overfits: the error on the training set is close to 0, whereas the test error is much more significant. The error on the training set is about 10^{-29}, which means that the training set was approximated with machine precision. It is not surprising if we recall that there are only 5000 observed values, while the number of free parameters that are used to construct TT is much higher.

4.4. CMOS ring oscillator. Let us consider the CMOS ring oscillator [24]. It is an electric circuit which consists of 7 stages of CMOS inverters. We are interested in the oscillation frequency of the oscillator. The characteristics of the electric circuit are described by 57 parameters. Each parameter can take one of 3 values, so the total size of the tensor is $3^{57} \approx 1.57 \times 10^{27}$. The number of observed values that were used during the experiments is $N = 5000$. For the test set we used 10000 independently generated points.

The results of the experiments are given in Table 2. The table demonstrates that utilizing GP-based initialization improves the results for all algorithms except for ALS. ALS, in this case, overfits again: the training error is extremely small, whereas the test error is much larger, though it is rather small compared to other techniques and ALS with random initialization.

All in all, the obtained results prove that GP-based initialization allows improving

Table 1
MSE errors for the cookie problem.

	Training set			
	$m = 3$			
	Random init		GP init	
	Error	Average N iters	Error	Average N iters
SGD	$(1.66 \pm 0.067) \times 10^{-2}$	1500	$(\mathbf{2.86} \pm \mathbf{0.18}) \times \mathbf{10^{-5}}$	150
Ropt	$(4.13 \pm 2.20) \times 10^{-8}$	1000	$(\mathbf{5.48} \pm \mathbf{1.10}) \times \mathbf{10^{-10}}$	1000
TTWopt	$(2.73 \pm 0.19) \times 10^{-4}$	100	$(\mathbf{9.21} \pm \mathbf{2.17}) \times \mathbf{10^{-7}}$	100
ALS	$(1.07 \pm 1.07) \times 10^{-4}$	100	$(\mathbf{2.39} \pm \mathbf{0.60}) \times \mathbf{10^{-30}}$	100
Init error	$(1.15 \pm 0.24) \times 10^{8}$		$\mathbf{2.66} \times \mathbf{10^{-4}}$	
	$m = 4$			
SGD	$(3.14 \pm 1.08) \times 10^{-2}$	1500	$(\mathbf{1.65} \pm \mathbf{0.13}) \times \mathbf{10^{-5}}$	150
Ropt	$(1.42 \pm 0.01) \times 10^{-2}$	1000	$(\mathbf{3.42} \pm \mathbf{0.50}) \times \mathbf{10^{-4}}$	1000
TTWopt	$(1.31 \pm 0.00) \times 10^{-4}$	100	$(\mathbf{1.80} \pm \mathbf{0.16}) \times \mathbf{10^{-6}}$	100
ALS	$(6.59 \pm 3.30) \times 10^{-5}$	100	$(\mathbf{1.33} \pm \mathbf{0.46}) \times \mathbf{10^{-29}}$	100
Init error	$(8.32 \pm 2.52) \times 10^{14}$		$\mathbf{3.14} \times \mathbf{10^{-4}}$	
	Test set			
	$m = 3$			
SGD	$(2.06 \pm 2.31) \times 10^{-1}$	—	$(\mathbf{9.97} \pm \mathbf{0.40}) \times \mathbf{10^{-5}}$	—
Ropt	$(\mathbf{1.48} \pm \mathbf{0.90}) \times \mathbf{10^{-7}}$	—	$(3.45 \pm 0.0165) \times 10^{-4}$	—
TTWopt	$(4.52 \pm 0.50) \times 10^{-4}$	—	$(\mathbf{5.27} \pm \mathbf{0.74}) \times \mathbf{10^{-6}}$	—
ALS	$(\mathbf{4.37} \pm \mathbf{7.73}) \times \mathbf{10^{-2}})$	—	$(3.78 \pm 1.08) \times 10^{0}$	—
Init error	$(1.12 \pm 0.23) \times 10^{8}$		$\mathbf{4.12} \times \mathbf{10^{-4}}$	
	$m = 4$			
SGD	$(2.40 \pm 2.76) \times 10^{1}$	—	$(\mathbf{1.15} \pm \mathbf{0.05}) \times \mathbf{10^{-4}}$	—
Ropt	$(1.47 \pm 0.003 \times 10^{-2}$	—	$(\mathbf{5.38} \pm \mathbf{0.07}) \times \mathbf{10^{-4}}$	—
TTWopt	$(2.42 \pm 0.00) \times 10^{-4}$	—	$(\mathbf{3.02} \pm \mathbf{0.17}) \times \mathbf{10^{-5}}$	—
ALS	$(\mathbf{3.57} \pm \mathbf{5.65}) \times \mathbf{10^{-1}}$	—	$(1.85 \pm 60.5) \times 10^{0}$	—
Init error	$(8.33 \pm 2.46) \times 10^{14}$		$\mathbf{5.37} \times \mathbf{10^{-4}}$	

Table 2
MSE errors for CMOS oscillator.

	Training set			
	Random init		GP init	
	Error	N iters	Error	N iters
SGD	$(7.77 \pm 15.25) \times 10^{5}$	1500	$(\mathbf{3.11} \pm \mathbf{4.87}) \times \mathbf{10^{-4}}$	150
Ropt	$(6.22 \pm 0.01) \times 10^{3}$	1000	$(\mathbf{9.50} \pm \mathbf{4.28}) \times \mathbf{10^{-5}}$	1000
ALS	$(9.95 \pm 0.26) \times 10^{-2}$	300	$(\mathbf{3.57} \pm \mathbf{0.45}) \times \mathbf{10^{-26}}$	300
Init error	$(1.67 \pm 3.19) \times 10^{12}$		$(\mathbf{3.95} \pm \mathbf{2.19}) \times \mathbf{10^{-4}}$	
	Test set			
SGD	$(3.45 \pm 9.68) \times 10^{8}$	—	$(\mathbf{4.65} \pm \mathbf{5.01}) \times \mathbf{10^{-4}}$	—
Ropt	$(6.23 \pm 0.0) \times 10^{3}$	—	$(\mathbf{9.68} \pm \mathbf{4.16}) \times \mathbf{10^{-5}}$	—
ALS	$(1.03 \pm 0.01) \times 10^{-1}$	—	$(\mathbf{4.09} \pm \mathbf{3.10}) \times \mathbf{10^{-4}}$	—
Init error	$(1.04 \pm 2.53) \times 10^{15}$		$(\mathbf{3.90} \pm \mathbf{2.15}) \times \mathbf{10^{-4}}$	

the tensor completion results in general. At least it provides better training error. As for the error on the test set, one should be more careful as the number of degrees of freedom is large and there exist many solutions that give a small error for the observed values but large errors for other values.

5. Related works.

One set of approaches to tensor completion is based on nuclear norm minimization. The nuclear norm of a matrix is defined as a sum of all singular values of the matrix. This objective function is a convex envelope of the rank function. For a tensor the nuclear norm is defined as a sum of singular values of matricizations of the tensor.

There are efficient off-the-shelf techniques for such types of problems that apply interior-point methods. However, they are second-order methods and scale poorly with the dimensionality of the problem. A special optimization technique was derived for nuclear norm minimization [8, 13, 18].

More often such techniques are applied to matrices or low-dimensional tensors as their straightforward formulation allows finding the full tensor. It becomes infeasible when we come to high-dimensional problems.

The second type of approach is based on low-rank tensor decomposition [1, 7, 12, 19, 23]. There are several tensor decompositions, and all these papers derive some optimization procedure for one of them, namely, CANDECOMP/PARAFAC decomposition, Tucker decomposition, or tensor train/matrix product states (MPS) decomposition. The simplest technique is alternating least squares (ALS) [10]. It just finds the solution iteratively, at each iteration minimizing the objective function with respect to one core while other cores are fixed.

Another approach, based on Riemannian optimization, tries to find the optimal solution on the manifold of low-rank tensors of the given structure [19]. The same can be done by using stochastic gradient descent [23]. Riemannian optimization, TTWopt, ALS, and its modifications (e.g., alternating direction fitting (ADF) [9]) try to find the TT representation of the actual tensor iteratively. At each iteration it optimizes TT cores such that the resulting tensor approximates well the tensor which coincides with the real tensor at observed indices and with the result of the previous iteration at other indices. All these approaches need to specify rank manually. The authors of [26] apply the Bayesian framework for CP decomposition which allows them to select the rank of the decomposition automatically.

In some papers the objective is modified by introducing special regularizers to better suit the problem [22]. For example, in [7, 26] to obtain better results for visual data a special prior regularizer was utilized.

Our proposed algorithm is an initialization technique for the tensor completion problems in TT format and can be used with most of the algorithms solving such problems. If the assumptions from section 3 (the tensor values are values of some rather smooth function of tensor indices) are satisfied, the initial value will be close to the optimal, providing better results. The question of a good initialization is rarely taken into account. In paper [11] a special initialization is proposed for visual data. The idea is to use some crude technique (like bilinear interpolation) to fill missing values and after that apply SVD-based TT decomposition. The drawback of the approach is that it can be applied only in the case of small-dimensional tensors as we need to fill all missing values. In [9] they propose special initialization for the ADF method. This is a general technique for the tensor completion and does not take into account the assumptions on the data generating function.

6. Conclusions.

We proposed a new initialization algorithm for high-dimensional tensor completion in TT format. The approach is designed mostly for cases when some smooth function generates the tensor values. It can be combined with any optimization procedure that is used for tensor completion. Additionally, the TT-rank of the initial tensor is adjusted automatically by the TT-cross method and defines the resulting rank of the tensor. So, the approach provides an automatic rank selection. Our experimental study confirms that the proposed initialization delivers lower reconstruction errors for many of the optimization procedures.

Acknowledgment. We would like to thank Zheng Zhang, who kindly provided us with the CMOS ring oscillator data set.

REFERENCES

[1] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup, *Scalable tensor factorizations for incomplete data*, Chemometrics Intell. Lab. Syst., 106 (2011), pp. 41–56.

[2] J. Ballani and L. Grasedyck, *Hierarchical tensor approximation of output quantities of parameter-dependent PDEs*, SIAM/ASA J. Uncertain. Quantif., 3 (2015), pp. 852–872, https://doi.org/10.1137/140960980.

[3] M. Belyaev, E. Burnaev, E. Kapushev, M. Panov, P. Prikhodko, D. Vetrov, and D. Yarotsky, *GTApprox: Surrogate modeling for industrial design*, Adv. Engrg. Software, 102 (2016), pp. 29–39.

[4] M. Belyaev, E. Burnaev, and Y. Kapushev, *Gaussian process regression for structured data sets*, in International Symposium on Statistical Learning and Data Sciences, Springer, 2015, pp. 106–115.

[5] M. Belyaev, E. Burnaev, and Y. Kapushev, *Computationally efficient algorithm for Gaussian process regression in case of structured samples*, Comput. Math. Math. Phys., 56 (2016), pp. 499–513.

[6] E. Burnaev, M. Panov, and A. Zaytsev, *Regression on the basis of nonstationary Gaussian processes with Bayesian regularization*, J. Commun. Technol. Electron., 61 (2016), pp. 661–671.

[7] Y.-L. Chen, C.-T. C. Hsu, and H.-Y. M. Liao, *Simultaneous tensor decomposition and completion using factor priors*, IEEE Trans. Pattern Anal. Mach. Intell., 36 (2014), pp. 577–591.

[8] S. Gandy, B. Recht, and I. Yamada, *Tensor completion and low-n-rank tensor recovery via convex optimization*, Inverse Problems, 27 (2011), 025010.

[9] L. Grasedyck, M. Kluge, and S. Krämer, *Alternating Directions Fitting (ADF) of Hierarchical Low Rank Tensors*, Preprint 149, DFG, 2013.

[10] L. Grasedyck, M. Kluge, and S. Krämer, *Alternating Least Squares Tensor Completion in the TT-Format*, preprint, https://arxiv.org/abs/1509.00311, 2015.

[11] C.-Y. Ko, K. Batselier, L. Daniel, W. Yu, and N. Wong, *Fast and accurate tensor completion with total variation regularized tensor trains*, IEEE Trans. Image Process., 29 (2020), pp. 6918–6931.

[12] D. Kressner, M. Steinlechner, and B. Vandereycken, *Low-rank tensor completion by Riemannian optimization*, BIT, 54 (2014), pp. 447–468.

[13] J. Liu, P. Musialski, P. Wonka, and J. Ye, *Tensor completion for estimating missing values in visual data*, IEEE Trans. Pattern Anal. Mach. Intell., 35 (2013), pp. 208–220.

[14] M. Munkhoeva, Y. Kapushev, E. Burnaev, and I. Oseledets, *Quadrature-based features for kernel approximation*, in Advances in Neural Information Processing Systems, 2018, pp. 9147–9156.

[15] I. Oseledets and E. Tyrtyshnikov, *TT-cross approximation for multidimensional arrays*, Linear Algebra Appl., 432 (2010), pp. 70–88.

[16] I. V. Oseledets, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317, https://doi.org/10.1137/090752286.

[17] C. E. Rasmussen, *Gaussian processes in machine learning*, in Advanced Lectures on Machine Learning, Springer, 2004, pp. 63–71.

[18] B. Recht, M. Fazel, and P. A. Parrilo, *Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization*, SIAM Rev., 52 (2010), pp. 471–501, https://doi.org/10.1137/070697835.

[19] M. Steinlechner, *Riemannian optimization for high-dimensional tensor completion*, SIAM J. Sci. Comput., 38 (2016), pp. S461–S484, https://doi.org/10.1137/15M1010506.

[20] C. Tobler, *Low-Rank Tensor Methods for Linear Systems and Eigenvalue Problems*, Ph.D. thesis, ETH Zurich, 2012.

[21] E. Tyrtyshnikov, *Incomplete cross approximation in the mosaic-skeleton method*, Computing, 64 (2000), pp. 367–380.

[22] T. Yokota, Q. Zhao, and A. Cichocki, *Smooth PARAFAC decomposition for tensor com-*

pletion, IEEE Trans. Signal Process., 64 (2016), pp. 5423–5436.

[23] L. Yuan, Q. Zhao, and J. Cao, *Completion of high order tensor data with missing entries via tensor-train decomposition*, in International Conference on Neural Information Processing, Springer, 2017, pp. 222–229.

[24] Z. Zhang, T.-W. Weng, and L. Daniel, *Big-data tensor recovery for high-dimensional uncertainty quantification of process variations*, IEEE Trans. Components Packaging Manufact. Technol., 7 (2017), pp. 687–697.

[25] L. Yuan, Q. Zhao, L. Gui, and J. Cao, *High-Dimension Tensor Completion via Gradient-Based Optimization under Tensor-Train Format*, preprint, https://arxiv.org/abs/1804.01983, 2018.

[26] Q. Zhao, L. Zhang, and A. Cichocki, *Bayesian CP factorization of incomplete tensors with automatic rank determination*, IEEE Trans. Pattern Anal. Mach. Intell., 37 (2015), pp. 1751–1763.