Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2022.021636

ARTICLE

Tensor Train Random Projection

Yani Feng[1], Kejun Tang[2], Lianxing He[3], Pingqiang Zhou[1] and Qifeng Liao[1,*]

[1]School of Information Science and Technology, ShanghaiTech University, Shanghai, 201210, China

[2]Peng Cheng Laboratory, Shenzhen, 518055, China

[3]Innovation Academy for Microsatellites of Chinese Academy of Sciences, Shanghai, 201210, China

*Corresponding Author: Qifeng Liao. Email: liaoqf@shanghaitech.edu.cn

Received: 25 January 2022 Accepted: 25 March 2022

ABSTRACT

This work proposes a Tensor Train Random Projection (TTRP) method for dimension reduction, where pairwise distances can be approximately preserved. Our TTRP is systematically constructed through a Tensor Train (TT) representation with TT-ranks equal to one. Based on the tensor train format, this random projection method can speed up the dimension reduction procedure for high-dimensional datasets and requires fewer storage costs with little loss in accuracy, compared with existing methods. We provide a theoretical analysis of the bias and the variance of TTRP, which shows that this approach is an expected isometric projection with bounded variance, and we show that the scaling Rademacher variable is an optimal choice for generating the corresponding TT-cores. Detailed numerical experiments with synthetic datasets and the MNIST dataset are conducted to demonstrate the efficiency of TTRP.

KEYWORDS

Tensor Train; random projection; dimension reduction

1 Introduction

Dimension reduction is a fundamental concept in science and engineering for feature extraction and data visualization. Exploring the low-dimensional structures of high-dimensional data attracts broad attention. Popular dimension reduction methods include Principal Component Analysis (PCA) [1,2], Non-negative Matrix Factorization (NMF) [3], and t-distributed Stochastic Neighbor Embedding (t-SNE) [4]. A main procedure in dimension reduction is to build a linear or nonlinear mapping from a high-dimensional space to a low-dimensional one, which keeps important properties of the high-dimensional space, such as the distance between any two points [5].

The Random Projection (RP) is a widely used method for dimension reduction. It is well-known that the Johnson-Lindenstrauss (JL) transformation [6,7] can nearly preserve the distance between two points after a random projection f, which is typically called isometry property. The isometry property can be used to achieve the nearest neighbor search in high-dimensional datasets [8,9]. It can also be used to build the Restricted Isometry Property (RIP) in compressed sensing [10,11], where a sparse

signal can be reconstructed under a linear random projection [12]. The JL lemma [6] tells us that there exists a nearly isometry mapping f, which maps high-dimensional datasets into a lower dimensional space. Typically, a choice for the mapping f is the linear random projection.

$$f(\boldsymbol{x}) = \frac{1}{\sqrt{M}}\boldsymbol{R}x, \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^N$, and $\boldsymbol{R} \in \mathbb{R}^{M\times N}$ is a matrix whose entries are drawn from the mean zero and variance one Gaussian distribution, denoted by $\mathcal{N}(0,1)$. We call it Gaussian random projection (Gaussian RP). The storage of matrix $\boldsymbol{R}$ in Eq. (1) is $O(MN)$ and the cost of computing $\boldsymbol{Rx}$ in Eq. (1) is $O(MN)$. However, with large M and N, this construction is computationally infeasible. To alleviate the difficulty, the sparse random projection method [13] and the very sparse random projection method [14] are proposed, where the random projection is constructed by a sparse random matrix. Thus the storage and the computational cost can be reduced.

To be specific, Achlioptas [13] replaced the dense matrix $\boldsymbol{R}$ by a sparse matrix whose entries follows:

$$\boldsymbol{R}_{ij} = \sqrt{s}\cdot\begin{cases} +1, & \text{with probability}\dfrac{1}{2s}, \\ 0, & \text{with probability}1-\dfrac{1}{s}, \\ -1, & \text{with probability}\dfrac{1}{2s}. \end{cases} \tag{2}$$

This means that the matrix is sampled at a rate of $1/s$. Note that, if $s = 1$, the corresponding distribution is called the Rademacher distribution. When $s = 3$, the cost of computing $\boldsymbol{Rx}$ in Eq. (1) reduces down to a third of the original one but is still $O(MN)$. When $s = \sqrt{N} \gg 3$, Li et al. [14] called this case as the very sparse random projection (Very Sparse RP), which significantly speeds up the computation with little loss in accuracy. It is clear that the storage of very sparse random projection is $O(M\sqrt{N})$. However, the sparse random projection can typically distort a sparse vector [9]. To achieve a low-distortion embedding, Ailon et al. [9,15] proposed the Fast-Johnson-Lindenstrauss Transform (FJLT), where the preconditioning of a sparse projection matrix with a randomized Fourier transform is employed. To reduce randomness and storage requirements, Sun et al. [16] proposed the following format: $\boldsymbol{R} = (\boldsymbol{R}_1 \odot \ldots \odot \boldsymbol{R}_d)^{\mathrm{T}}$, where $\odot$ represents the Khatri-Rao product, $\boldsymbol{R}_i \in \mathbb{R}^{n_i\times M}$, and $N = \prod_{i=1}^{d} n_i$. Each $\boldsymbol{R}_i$ is a random matrix whose entries are i.i.d. random variables drawn from $\mathcal{N}(0,1)$. This transformation is called the Gaussian tensor random projection (Gaussian TRP) throughout this paper. It is clear that the storage of the Gaussian TRP is $O(M\sum_{i=1}^{d} n_i)$, which is less than that of the Gaussian random projection (Gaussian RP). For example, when $N = n_1 n_2 = 40000$, the storage of Gaussian TRP is only $1/20$ of Gaussian RP. Also, it has been shown that Gaussian TRP satisfies the properties of expected isometry with vanishing variance [16].

Recently, using matrix or tensor decomposition to reduce the storage of projection matrices is proposed in [17,18]. The main idea of these methods is to split the projection matrix into some small scale matrices or tensors. In this work, we focus on the low rank tensor train representation to construct the random projection f. Tensor decompositions are widely used for data compression [5,19–24]. The Tensor Train (TT) decomposition, also called Matrix Product States (MPS) [25–28], gives the following benefits—low rank TT-formats can provide compact representations of projection matrices and efficient basic linear algebra operations of matrix-by-vector products [29]. Based on these

benefits, we propose a Tensor Train Random Projection (TTRP) method, which requires significantly smaller storage and computational costs compared with existing methods (e.g., Gaussian TRP [16], Very Sparse RP [14] and Gaussian RP [30]). We note that constructing projection matrices using Tensor Train (TT) and Canonical Polyadic (CP) decompositions based on Gaussian random variables is proposed in [31]. The main contributions of our work are three-fold: first our TTRP is conducted based on a rank-one TT-format, which significantly reduces the storage of projection matrices; second, we provide a novel construction procedure for the rank-one TT-format in our TTRP based on i.i.d. Rademacher random variables; third, we prove that our construction of TTRP is unbiased with bounded variance.

The rest of the paper is organized as follows. The tensor train format is introduced in Section 2. Details of our TTRP approach are introduced in Section 3, where we prove that the approach is an expected isometric projection with bounded variance. In Section 4, we demonstrate the efficiency of TTRP with datasets including synthetic, MNIST. Finally Section 5 concludes the paper.

2 Tensor Train Format

Let lowercase letters (x), boldface lowercase letters ($\boldsymbol{x}$), boldface capital letters ($\boldsymbol{X}$), calligraphy letters ($\mathscr{X}$) be scalar, vector, matrix and tensor variables, respectively. $\boldsymbol{x}(i)$ represents the element i of a vector $\boldsymbol{x}$. $\boldsymbol{X}(i,j)$ means the element (i,j) of a matrix $\boldsymbol{X}$. The i-th row and j-th column of a matrix $\boldsymbol{X}$ is defined by $\boldsymbol{X}(i,:)$ and $\boldsymbol{X}(:,j)$, respectively. For a given d-th order tensor $\mathscr{X}$, $\mathscr{X}(i_1,i_2,\ldots,i_d)$ is its $(i_1,i_2,\ldots,i_d)$-th component. For a vector $\boldsymbol{x}\in\mathbb{R}^N$, we denote its ℓ^p norm as $\|\boldsymbol{x}\|_p=(\sum_{i=1}^{N}|\boldsymbol{x}(i)|^p)^{\frac{1}{p}}$, for any $p\geq 1$. The Kronecker product of matrices $\boldsymbol{A}\in\mathbb{R}^{I\times J}$ and $\boldsymbol{B}\in\mathbb{R}^{K\times L}$ is denoted by $\boldsymbol{A}\otimes\boldsymbol{B}$ of which the result is a matrix of size $(IK)\times(JL)$ and defined by

$$\boldsymbol{A}\otimes\boldsymbol{B}=\begin{bmatrix}\boldsymbol{A}(1,1)\boldsymbol{B} & \boldsymbol{A}(1,2)\boldsymbol{B} & \cdots & \boldsymbol{A}(1,J)\boldsymbol{B}\\ \boldsymbol{A}(2,1)\boldsymbol{B} & \boldsymbol{A}(2,2)\boldsymbol{B} & \cdots & \boldsymbol{A}(2,J)\boldsymbol{B}\\ \vdots & \vdots & \ddots & \vdots\\ \boldsymbol{A}(I,1)\boldsymbol{B} & \boldsymbol{A}(I,2)\boldsymbol{B} & \cdots & \boldsymbol{A}(I,J)\boldsymbol{B}\end{bmatrix}.$$

The Kronecker product conforms the following laws [32]:

$$(\boldsymbol{AC})\otimes(\boldsymbol{BD})=(\boldsymbol{A}\otimes\boldsymbol{B})(\boldsymbol{C}\otimes\boldsymbol{D}), \tag{3}$$

$$(\boldsymbol{A}+\boldsymbol{B})\otimes(\boldsymbol{C}+\boldsymbol{D})=\boldsymbol{A}\otimes\boldsymbol{C}+\boldsymbol{A}\otimes\boldsymbol{D}+\boldsymbol{B}\otimes\boldsymbol{C}+\boldsymbol{B}\otimes\boldsymbol{D}, \tag{4}$$

$$(k\boldsymbol{A})\otimes\boldsymbol{B}=\boldsymbol{A}\otimes(k\boldsymbol{B})=k\,(\boldsymbol{A}\otimes\boldsymbol{B})\,. \tag{5}$$

2.1 Tensor Train Decomposition

Tensor Train (TT) decomposition [29] is a generalization of Singular Value Decomposition (SVD) from matrices to tensors. TT decomposition provides a compact representation for tensors, and allows for efficient application of linear algebra operations (discussed in Section 2.2 and Section 2.3).

Given a d-th order tensor $\mathscr{G}\in\mathbb{R}^{n_1\times\cdots\times n_d}$ and a vector $\boldsymbol{g}$ (vector format of $\mathscr{G}$) , the tensor train decomposition [29] is

$$\boldsymbol{g}(i)=\mathscr{G}(\nu(i))=\mathscr{G}(i_1,i_2,\ldots,i_d)=\mathscr{G}_1(i_1)\mathscr{G}_2(i_2)\cdots\mathscr{G}_d(i_d), \tag{6}$$

where $\nu{:}\mathbb{N}\mapsto\mathbb{N}^d$ is a bijection, $\mathscr{G}_k\in\mathbb{R}^{r_{k-1}\times n_k\times r_k}$ are called TT-cores, $\mathscr{G}_k(i_k)\in\mathbb{R}^{r_{k-1}\times r_k}$ is a slice of $\mathscr{G}_k$, for $k=1,2,\ldots,d$, $i_k=1,\ldots,n_k$, and the "boundary condition" is $r_0=r_d=1$. The tensor $\mathscr{G}$ is said to be in the TT-format if each element of $\mathscr{G}$ can be represented by Eq. (6). The vector $[r_0,r_1,r_2,\ldots,r_d]$ is

referred to as TT-ranks. Let $\mathcal{G}_k(\alpha_{k-1}, i_k, \alpha_k)$ represent the element of $\mathcal{G}_k(i_k)$ in the position (α_{k-1}, α_k). In the index form, the decomposition (6) is rewritten as the following TT-format:

$$\mathcal{G}(i_1, i_2, \ldots, i_d) = \sum_{\alpha_0, \cdots, \alpha_d} \mathcal{G}_1(\alpha_0, i_1, \alpha_1)\mathcal{G}_2(\alpha_1, i_2, \alpha_2) \cdots \mathcal{G}_d(\alpha_{d-1}, i_d, \alpha_d).$$

To look more closely to Eq. (6), an element $\mathcal{G}(i_1, i_2, \ldots, i_d)$ is represented by a sequence of matrix-by-vector products. Fig. 1 illustrates the tensor train decomposition. It can be seen that the key ingredient in tensor train (TT) decomposition is the TT-ranks. The TT-format only uses $O(ndr^2)$ memory to $O(n^d)$ elements, where $n = \max\{n_1, \ldots, n_d\}$ and $r = \max\{r_0, r_1, \ldots, r_d\}$. Although the storage reduction is efficient only if the TT-rank is small, tensors in data science and machine learning typically have low TT-ranks. Moreover, one can apply the TT-format to basic linear algebra operations, such as matrix-by-vector products, scalar multiplications, etc. In later section, it is shown that this can reduce the computational cost significantly when the data have low rank structures (see [29] for details).

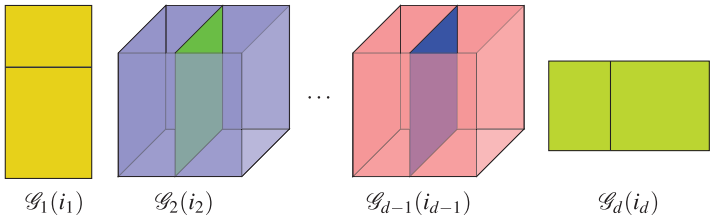


Figure 1: Tensor train format (TT-format): extract an element $\mathcal{G}(i_1, i_2, \ldots, i_d)$ via a sequence of matrix-byvector product

2.2 Tensorizing Matrix-by-Vector Products

The tensor train format gives a compact representation of matrices and efficient computation for matrix-by-vector products. We first review the TT-format of large matrices and vectors following [29]. Defining two bijections $\nu{:}\mathbb{N} \mapsto \mathbb{N}^d$ and $\mu : \mathbb{N} \mapsto \mathbb{N}^d$, a pair index $(i, j) \in \mathbb{N}^2$ is mapped to a multi-index pair $(\nu(i), \mu(j)) = (i_1.i_2, \ldots, i_d, j_1, j_2, \ldots, j_d)$. Then a matrix $\boldsymbol{R} \in \mathbb{R}^{M \times N}$ and a vector $\boldsymbol{x} \in \mathbb{R}^N$ can be tensorized in the TT-format as follows. Letting $M = \prod_{i=1}^{d} m_i$ and $N = \prod_{j=1}^{d} n_j$, an element (i, j) of $\boldsymbol{R}$ can be written as Eq. (7) (see [29,33] for more details):

$$\boldsymbol{R}(i, j) = \mathcal{R}(\nu(i), \mu(j)) = \mathcal{R}(i_1, \ldots, i_d; j_1, \ldots, j_d) = \mathcal{R}_1(i_1, j_1) \cdots \mathcal{R}_d(i_d, j_d), \tag{7}$$

where $(i_1, \ldots, i_d)$ enumerate the rows of $\boldsymbol{R}$, $(j_1, \ldots, j_d)$ enumerate the columns of $\boldsymbol{R}$, and $\mathcal{R}_k(i_k, j_k)$ is an $r_{k-1} \times r_k$ matrix (and $r_0 = r_d = 1$), i.e., (i_k, j_k) is treated as one "long index" for $k = 1, \ldots, d$. An element j of $\boldsymbol{x}$ can be written as Eq. (8):

$$\boldsymbol{x}(j) = \mathcal{X}(\mu(j)) = \mathcal{X}(j_1, \ldots, j_d) = \mathcal{X}_1(j_1) \cdots \mathcal{X}_d(j_d), \tag{8}$$

where $\mathcal{X}_k(j_k)$ is an $\hat{r}_{k-1} \times \hat{r}_k$ matrix and $\hat{r}_0 = \hat{r}_d = 1$ for $k = 1, \ldots, d$. We consider the matrix-by-vector product ($\boldsymbol{y} = \boldsymbol{R}\boldsymbol{x}$), and each element of $\boldsymbol{y}$ can be tensorized in the TT-format as Eq. (9):

$$
\begin{aligned}
\boldsymbol{y}(i) = \mathscr{Y}(i_1,\ldots,i_d) &= \sum_{j_1,\cdots j_d} \mathscr{R}(i_1,\ldots,i_d,j_1,\ldots,j_d)\mathscr{X}(j_1,\ldots,j_d) \\
&= \sum_{j_1,\cdots,j_d} (\mathscr{R}_1(i_1,j_1)\cdots\mathscr{R}_d(i_d,j_d))(\mathscr{X}_1(j_1)\cdots\mathscr{X}_d(j_d)) \\
&= \sum_{j_1,\cdots,j_d} (\mathscr{R}_1(i_1,j_1)\cdots\mathscr{R}_d(i_d,j_d)) \otimes (\mathscr{X}_1(j_1)\cdots\mathscr{X}_d(j_d)) \\
&= \sum_{j_1,\cdots,j_d} \underbrace{(\mathscr{R}_1(i_1,j_1)\otimes\mathscr{X}_1(j_1))}_{O(r_0 r_1 \hat{r}_0 \hat{r}_1)} \cdots \underbrace{(\mathscr{R}_d(i_d,j_d)\otimes\mathscr{X}_d(j_d))}_{O(r_{d-1} r_d \hat{r}_{d-1} \hat{r}_d)} \\
&= \underbrace{\mathscr{Y}_1(i_1)}_{O(n_1 r_0 r_1 \hat{r}_0 \hat{r}_1)} \cdots \underbrace{\mathscr{Y}_d(i_d)}_{O(n_d r_{d-1} r_d \hat{r}_{d-1} \hat{r}_d)},
\end{aligned}
\tag{9}
$$

where $\mathscr{Y}_k(i_k) = \sum_{j_k} \mathscr{R}_k(i_k,j_k) \otimes \mathscr{X}_k(j_k) \in \mathbb{R}^{r_{k-1}\hat{r}_{k-1}\times r_k\hat{r}_k}$, for $k = 1,\ldots,d$. The complexity of computing each TT-core $\mathscr{Y}_k \in \mathbb{R}^{r_{k-1}\hat{r}_{k-1}\times m_k\times r_k\hat{r}_k}$, is $O(m_k n_k r_{k-1} r_k \hat{r}_{k-1}\hat{r}_k)$ for $k = 1,\ldots,d$. Assuming that the TT-cores of $\boldsymbol{x}$ are known, the total cost of the matrix-by-vector product ($\boldsymbol{y} = \boldsymbol{R}\boldsymbol{x}$) in the TT-format can reduce significantly from the original complexity $O(MN)$ to $O(dmnr^2\hat{r}^2)$, $m = \max\{m_1, m_2, \ldots, m_d\}$, $n = \max\{n_1, n_2, \ldots, n_d\}$, $r = \max\{r_0, r_1, \ldots, r_d\}$, $\hat{r} = \max\{\hat{r}_0, \hat{r}_1, \ldots, \hat{r}_d\}$, where N is typically large and r is small. When $m_k = n_k$, $r_k = \hat{r}_k$, for $k = 1,\ldots,d$, the cost of such matrix-by-vector product in the TT-format is $O(dn^2r^4)$ [29]. Note that, in the case that r equals one, the cost of such matrix-by-vector product in the TT-format is $O(dmn\hat{r}^2)$.

2.3 Basic Operations in the TT-Format

In Section 2.2, the product of matrix $\boldsymbol{R}$ and vector $\boldsymbol{x}$ which are both in the TT-format, is conducted efficiently. In the TT-format, many important operations can be readily implemented. For instance, computing the Euclidean distance between two vectors in the TT-format is more efficient with less storage than directly computing the Euclidean distance in standard matrix and vector formats. In the following Eqs. (10)–(15), some important operations in the TT-format are discussed.

The subtraction of tensor $\mathscr{Y} \in \mathbb{R}^{m_1\times\cdots\times m_d}$ and tensor $\widehat{\mathscr{Y}} \in \mathbb{R}^{m_1\times\cdots\times m_d}$ in the TT-format is

$$
\begin{aligned}
\mathscr{Z}(i_1,\ldots,i_d) :&= \mathscr{Y}(i_1,\ldots,i_d) - \widehat{\mathscr{Y}}(i_1,\ldots,i_d) \\
&= \mathscr{Y}_1(i_1)\mathscr{Y}_2(i_2)\cdots\mathscr{Y}_d(i_d) - \widehat{\mathscr{Y}}_1(i_1)\widehat{\mathscr{Y}}_2(i_2)\cdots\widehat{\mathscr{Y}}_d(i_d) \\
&= \mathscr{Z}_1(i_1)\mathscr{Z}_2(i_2)\cdots\mathscr{Z}_d(i_d),
\end{aligned}
\tag{10}
$$

where

$$
\mathscr{Z}_k(i_k) = \begin{pmatrix} \mathscr{Y}_k(i_k) & 0 \\ 0 & \widehat{\mathscr{Y}}_k(i_k) \end{pmatrix}, \quad k = 2,\ldots,d-1,
$$

and

$$
\mathscr{Z}_1(i_1) = \begin{pmatrix} \mathscr{Y}_1(i_1) & -\widehat{\mathscr{Y}}_1(i_1) \end{pmatrix}, \quad \mathscr{Z}_d(i_d) = \begin{pmatrix} \mathscr{Y}_d(i_d) \\ \widehat{\mathscr{Y}}_d(i_d) \end{pmatrix},
$$

and TT-ranks of $\mathscr{Z}$ equal the sum of TT-ranks of $\mathscr{Y}$ and $\widehat{\mathscr{Y}}$.

The dot product of tensor $\mathscr{Y}$ and tensor $\widehat{\mathscr{Y}}$ in the TT-format [29] is

$$\begin{aligned}\langle \mathscr{Y}, \widehat{\mathscr{Y}} \rangle :&= \sum_{i_1,\ldots,i_d} \mathscr{Y}(i_1,\ldots,i_d)\,\widehat{\mathscr{Y}}(i_1,\ldots,i_d)\\ &= \sum_{i_1,\ldots,i_d} (\mathscr{Y}_1(i_1)\mathscr{Y}_2(i_2)\cdots\mathscr{Y}_d(i_d))(\widehat{\mathscr{Y}}_1(i_1)\widehat{\mathscr{Y}}_2(i_2)\cdots\widehat{\mathscr{Y}}_d(i_d))\\ &= \sum_{i_1,\ldots,i_d} (\mathscr{Y}_1(i_1)\,\mathscr{Y}_2(i_2)\cdots\mathscr{Y}_d(i_d)) \otimes (\widehat{\mathscr{Y}}_1(i_1)\widehat{\mathscr{Y}}_2(i_2)\cdots\widehat{\mathscr{Y}}_d(i_d))\\ &= \left(\sum_{i_1} \mathscr{Y}_1(i_1)\otimes\widehat{\mathscr{Y}}_1(i_1)\right)\left(\sum_{i_2} \mathscr{Y}_2(i_2)\otimes\widehat{\mathscr{Y}}_2(i_2)\right)\cdots\left(\sum_{i_d} \mathscr{Y}_d(i_d)\otimes\widehat{\mathscr{Y}}_d(i_d)\right)\\ &= \boldsymbol{V}_1\boldsymbol{V}_2\cdots\boldsymbol{V}_d, \end{aligned} \tag{11}$$

where

$$\boldsymbol{V}_k = \sum_{i_k} \mathscr{Y}_k(i_k)\otimes\widehat{\mathscr{Y}}_k(i_k), \quad k=1,\ldots,d. \tag{12}$$

Since $\boldsymbol{V}_1, \boldsymbol{V}_d$ are vectors and $\boldsymbol{V}_2,\ldots,\boldsymbol{V}_{d-1}$ are matrices, we compute $\langle \mathscr{Y}, \widehat{\mathscr{Y}} \rangle$ by a sequence of matrix-by-vector products:

$$\boldsymbol{v}_1 = \boldsymbol{V}_1, \tag{13}$$

$$\boldsymbol{v}_k = \boldsymbol{v}_{k-1}\boldsymbol{V}_k = \boldsymbol{v}_{k-1}\sum_{i_k} \mathscr{Y}_k(i_k)\otimes\widehat{\mathscr{Y}}_k(i_k) = \sum_{i_k} \boldsymbol{p}_k(i_k), \quad k=2,\ldots,d, \tag{14}$$

where

$$\boldsymbol{p}_k(i_k) = \boldsymbol{v}_{k-1}\left(\mathscr{Y}_k(i_k)\otimes\widehat{\mathscr{Y}}_k(i_k)\right), \tag{15}$$

and we obtain

$$\langle \mathscr{Y}, \widehat{\mathscr{Y}} \rangle = \boldsymbol{v}_d.$$

For simplify we assume that TT-ranks of $\mathscr{Y}$ are the same as that of $\widehat{\mathscr{Y}}$. In Eq. (15), let $\boldsymbol{B}:=\mathscr{Y}_k(i_k)\in\mathbb{R}^{r\times r}$, $\boldsymbol{C}:=\widehat{\mathscr{Y}}_k(i_k)\in\mathbb{R}^{r\times r}$, $\boldsymbol{x}:=\boldsymbol{v}_{k-1}\in\mathbb{R}^{1\times r^2}$, $\boldsymbol{y}:=\boldsymbol{p}_k(i_k)\in\mathbb{R}^{1\times r^2}$, for $k=2,\ldots,d-1$, and we use the reshaping Kronecker product expressions [34] for Eq. (15):

$$\boldsymbol{y}=\boldsymbol{x}(\boldsymbol{B}\otimes\boldsymbol{C}) \quad \Leftrightarrow \quad \boldsymbol{Y}=\boldsymbol{C}^T\boldsymbol{X}\boldsymbol{B},$$

where we reshape $\boldsymbol{x},\boldsymbol{y}$ into $X=[\boldsymbol{x}_1\ \boldsymbol{x}_2\ \cdots\ \boldsymbol{x}_r]\in\mathbb{R}^{r\times r}$, $Y=[\boldsymbol{y}_1\ \boldsymbol{y}_2\ \cdots\ \boldsymbol{y}_r]\in\mathbb{R}^{r\times r}$, respectively. Note that the cost of computing $\boldsymbol{Y}=\boldsymbol{C}^T\boldsymbol{X}\boldsymbol{B}$ is $O(r^3)$ while the disregard of Kronecker structure of $\boldsymbol{y}=\boldsymbol{x}(\boldsymbol{B}\otimes\boldsymbol{C})$ leads to an $O(r^4)$ calculation. Hence the complexity of computing $\boldsymbol{p}_k(\boldsymbol{i}_k)$ in (15) is $O(r^3)$, because of the efficient Kronecker product computation. Then the cost of computing $\boldsymbol{v}_k$ in (14) is $O(mr^3)$, and the total cost of the dot product $\langle \mathscr{Y}, \widehat{\mathscr{Y}} \rangle$ is $O(dmr^3)$.

The Frobenius norm of a tensor $\mathscr{Y}$ is defined by

$$\|\mathscr{Y}\|_F = \sqrt{\langle \mathscr{Y}, \mathscr{Y} \rangle}.$$

Computing the distance between tensor $\mathscr{Y}$ and tensor $\widehat{\mathscr{Y}}$ in the TT-format is computationally efficient by applying the dot product Eqs. (11) and (12),

$$\left\|\mathscr{Y}-\widehat{\mathscr{Y}}\right\|_F=\sqrt{\langle\mathscr{Y}-\widehat{\mathscr{Y}},\mathscr{Y}-\widehat{\mathscr{Y}}\rangle}. \tag{16}$$

The complexity of computing the distance is also $O(dmr^3)$. Algorithm 1 gives more details about computing Eq. (16) based on Frobenius norm $\left\|\mathscr{Y}-\widehat{\mathscr{Y}}\right\|_F$.

Algorithm 1: Distance based on Frobenius norm $W:=\left\|\mathscr{Y}-\widehat{\mathscr{Y}}\right\|_F=\sqrt{\langle\mathscr{Y}-\widehat{\mathscr{Y}},\mathscr{Y}-\widehat{\mathscr{Y}}\rangle}$

Input: TT-cores $\mathscr{Y}_k$ of tensor $\mathscr{Y}$ and TT-cores $\widehat{\mathscr{Y}_k}$ of tensor $\widehat{\mathscr{Y}}$, for $k=1,\ldots,d$.

1: Compute $\mathscr{Z}:=\mathscr{Y}-\widehat{\mathscr{Y}}$. ▷ $O(mr)$ by (10)
2: Compute $\boldsymbol{v_1}:=\sum_{i_1}\mathscr{Z}_1(i_1)\otimes\mathscr{Z}_1(i_1)$. ▷ $O(mr^2)$ by Eq.(13)
3: **for** $k=2{:}d-1$ **do**
4: Compute $\boldsymbol{p_k}(i_k)=\boldsymbol{v_{k-1}}(\mathscr{Z}_k(i_k)\otimes\mathscr{Z}_k(i_k))$. ▷ $O(r^3)$ by Eq. (15)
5: Compute $\boldsymbol{v_k}:=\sum_{i_k}\boldsymbol{p_k}(i_k)$. ▷ $O(mr^3)$ by Eq. (14)
6: **end for**
7: Compute $\boldsymbol{p_d}(i_d)=\boldsymbol{v_{d-1}}(\mathscr{Z}_d(i_d)\otimes\mathscr{Z}_d(i_d))$. ▷ $O(r^2)$ by Eq. (15)
8: Compute $\boldsymbol{v_d}:=\sum_{i_d}\boldsymbol{p_d}(i_d)$. ▷ $O(mr^2)$ by Eq. (14)

Output: Distance $W:=\sqrt{\langle\mathscr{Y}-\widehat{\mathscr{Y}},\mathscr{Y}-\widehat{\mathscr{Y}}\rangle}=\sqrt{\boldsymbol{v_d}}$.

In summary, just merging the cores of two tensors in the TT-format can perform the subtraction of two tensors instead of directly subtraction of two tensors in standard tensor format. A sequence of matrix-by-vector products can achieve the dot product of two tensors in the TT-format. The cost of computing the distance between two tensors in the TT-format, reduces from the original complexity $O(M)$ to $O(dmr^3)$, where $M=\prod_{i=1}^{d}m_i$, $r\ll M$, and $m=\max\{m_1,m_2,\ldots,m_d\}$.

3 Tensor Train Random Projection

Due to the computational efficiency of TT-format discussed above, we consider the TT-format to construct projection matrices. Our tensor train random projection is defined as follows.

Definition 3.1. (Tensor Train Random Projection). For a data point $\boldsymbol{x}\in\mathbb{R}^N$, our tensor train random projection (TTRP) is

$$f_{TTRP}(\boldsymbol{x}):=\frac{1}{\sqrt{M}}\boldsymbol{R}x, \tag{17}$$

where the tensorized versions (through the TT-format) of $\boldsymbol{R}$ and $\boldsymbol{x}$ are denoted by $\mathscr{R}$ and $\mathscr{X}$ (see Eqs. (7) and (8)), the corresponding TT-cores are denoted by $\{\mathscr{R}_k\in\mathbb{R}^{r_{k-1}\times m_k\times n_k\times r_k}\}_{k=1}^d$ and $\{\mathscr{X}_k\in\mathbb{R}^{\hat{r}_{k-1}\times n_k\times\hat{r}_k}\}_{k=1}^d$, respectively, we set $r_0=r_1=\ldots=r_d=1$, and $\boldsymbol{y}:=\boldsymbol{Rx}$ is specified by Eq. (9).

In Section 2.2, it is shown that for any TT-rank, the computation cost of $\boldsymbol{Rx}$ in Eq. (17) is $O(dmnr^2\hat{r}^2)$, and the storage cost is $O(dmnr^2)$, where $m=\max\{m_1,m_2,\ldots,m_d\}$, $n=\max\{n_1,n_2,\ldots,n_d\}$, $r=\max\{r_0,r_1,\ldots,r_d\}$ and $\hat{r}=\max\{\hat{r}_0,\hat{r}_1,\ldots,\hat{r}_d\}$. Note that our TTRP is based on the tensorized version of $\boldsymbol{R}$ with TT-ranks all equal to one, i.e., $r_0=r_1=\ldots=r_d=1$, which significantly

reduces the computational and storage costs. Therefore, our TTRP is constructed using a rank-one TT tensor. The comparisons for TTRP with different TT-ranks are investigated in Section 4. It turns out that the computational cost of computing $\boldsymbol{Rx}$ and the storage cost in Eq. (17) are $O(dmn\hat{r}^2)$ and $O(dmn)$, respectively. Moreover, from our analysis in the latter part of this section, we find that the Rademacher distribution introduced in Section 1 is an optimal choice for generating the TT-cores of $\boldsymbol{R}$. In the following, we prove that TTRP established by Eq. (17) is an expected isometric projection with bounded variance.

Theorem 3.1. Given a vector $\boldsymbol{x} \in \mathbb{R}^{\prod_{j=1}^{d} n_j}$, if $\boldsymbol{R}$ in Eq. (17) is composed of d independent TT-cores $\mathscr{R}_1, \ldots, \mathscr{R}_d$, whose entries are independent and identically random variables with mean zero and variance one, then the following equation holds

$$\mathbb{E}\,\|f_{TTRP}(\boldsymbol{x})\|_2^2 = \|\boldsymbol{x}\|_2^2.$$

Proof. Denoting $\boldsymbol{y} = \boldsymbol{Rx}$ gives

$$\mathbb{E}\,\|f_{TTRP}(\boldsymbol{x})\|_2^2 = \frac{1}{M}\mathbb{E}\,\|\boldsymbol{y}\|_2^2 = \frac{1}{M}\mathbb{E}\left[\sum_{i=1}^{M}\boldsymbol{y}^2(i)\right] = \frac{1}{M}\mathbb{E}\left[\sum_{i_1,\ldots,i_d}\mathscr{Y}^2(i_1,\ldots,i_d)\right]. \tag{18}$$

By the TT-format, $\mathscr{Y}(i_1,\ldots,i_d) = \mathscr{Y}_1(i_1)\cdots\mathscr{Y}_d(i_d)$, where $\mathscr{Y}_k(i_k) = \sum_{j_k}\mathscr{R}_k(i_k,j_k)\otimes\mathscr{X}_k(j_k)$, for $k = 1,\ldots,d$, it follows that

$$\begin{aligned}
\mathbb{E}\left[\mathscr{Y}^2(i_1,\ldots,i_d)\right] &= \mathbb{E}\left[(\mathscr{Y}_1(i_1)\cdots\mathscr{Y}_d(i_d))(\mathscr{Y}_1(i_1)\cdots\mathscr{Y}_d(i_d))\right] \\
&= \mathbb{E}\left[(\mathscr{Y}_1(i_1)\cdots\mathscr{Y}_d(i_d))\otimes(\mathscr{Y}_1(i_1)\cdots\mathscr{Y}_d(i_d))\right] &(19)\\
&= \mathbb{E}\left[(\mathscr{Y}_1(i_1)\otimes\mathscr{Y}_1(i_1))\cdots(\mathscr{Y}_d(i_d)\otimes\mathscr{Y}_d(i_d))\right] &(20)\\
&= \mathbb{E}[\mathscr{Y}_1(i_1)\otimes\mathscr{Y}_1(i_1)]\cdots\mathbb{E}[\mathscr{Y}_d(i_d)\otimes\mathscr{Y}_d(i_d)], &(21)
\end{aligned}$$

where Eq. (20) is derived using Eqs. (3) and (19), and then combining Eq. (20) and using the independence of TT-cores $\mathscr{R}_1,\ldots,\mathscr{R}_d$ give Eq. (21). The k-th term of the right hand side of Eq. (21), for $k = 1,\ldots,d$, can be computed by

$$\begin{aligned}
\mathbb{E}\left[\mathscr{Y}_k(i_k)\otimes\mathscr{Y}_k(i_k)\right] &= \mathbb{E}\left[\left[\sum_{j_k}\mathscr{R}_k(i_k,j_k)\otimes\mathscr{X}_k(j_k)\right]\otimes\left[\sum_{j_k}\mathscr{R}_k(i_k,j_k)\otimes\mathscr{X}_k(j_k)\right]\right] &(22)\\
&= \mathbb{E}\left[\left[\sum_{j_k}\mathscr{R}_k(i_k,j_k)\mathscr{X}_k(j_k)\right]\otimes\left[\sum_{j_k}\mathscr{R}_k(i_k,j_k)\mathscr{X}_k(j_k)\right]\right] &(23)\\
&= \sum_{j_k,j'_k}\mathbb{E}\left[\mathscr{R}_k(i_k,j_k)\mathscr{R}_k(i_k,j'_k)\right]\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j'_k) &(24)\\
&= \sum_{j_k}\mathbb{E}\left[\mathscr{R}_k^2(i_k,j_k)\right]\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k) &(25)\\
&= \sum_{j_k}\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k). &(26)
\end{aligned}$$

Here as we set the TT-ranks of $\mathscr{R}$ to be one, $\mathscr{R}_k(i_k,j_k)$ is scalar, and Eq. (22) then leads to Eq. (23). Using Eqs. (4), (5) and (23) gives Eq. (24), and we derive Eq. (26) from Eq. (24) by the assumption that

$\mathbb{E}[\mathscr{R}_k^2(i_k,j_k)]=1$ and $\mathbb{E}[\mathscr{R}_k(i_k,j_k)\mathscr{R}_k(i_k,j_k')]=0$, for $j_k,j_k'=1,\ldots,n_k, j_k\neq j_k', k=1,\ldots,d$. Substituting Eq. (26) into Eq. (21) gives

$$\begin{aligned}\mathbb{E}[\mathscr{Y}^2(i_1,\ldots,i_d)] &= \left[\sum_{j_1}\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\right]\cdots\left[\sum_{j_d}\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\right]\\ &= \sum_{j_1,\ldots,j_d}[\mathscr{X}_1(j_1)\cdots\mathscr{X}_d(j_d)]\otimes[\mathscr{X}_1(j_1)\cdots\mathscr{X}_d(j_d)]\\ &= \sum_{j_1,\ldots,j_d}\mathscr{X}^2(j_1,\ldots,j_d)\\ &= \|\boldsymbol{x}\|_2^2. \end{aligned}\tag{27}$$

Substituting Eq. (27) into Eq. (18), it concludes that

$$\begin{aligned}\mathbb{E}\,\|f_{TTRP}(\mathscr{X})\|_2^2 &= \frac{1}{M}\mathbb{E}\left[\sum_{i_1,\ldots,i_d}\mathscr{Y}^2(i_1,\ldots,i_d)\right]\\ &= \frac{1}{M}\times M\,\|\boldsymbol{x}\|_2^2\\ &= \|\boldsymbol{x}\|_2^2.\end{aligned}$$

Theorem 3.2. Given a vector $\boldsymbol{x}\in\mathbb{R}^{\prod_{j=1}^{d}n_j}$, if $\boldsymbol{R}$ in Eq. (17) is composed of d independent TT-cores $\mathscr{R}_1,\ldots,\mathscr{R}_d$, whose entries are independent and identically random variables with mean zero, variance one, with the same fourth moment Δ and $\mathscr{M}:=\max_{i=1,\ldots,N}|\boldsymbol{x}(i)|, m=\max\{m_1,m_2,\ldots,m_d\}, n=\max\{n_1,n_2,\ldots,n_d\}$, then

$$\mathrm{Var}\left(\|f_{TTRP}(\boldsymbol{x})\|_2^2\right)\leq\frac{1}{M}(\Delta+n(m+2)-3)^d N\mathscr{M}^4-\|\boldsymbol{x}\|_2^4.$$

Proof. By the property of the variance and using Theorem 3.1,

$$\begin{aligned}\mathrm{Var}\left(\|f_{TTRP}(\boldsymbol{x})\|_2^2\right) &= \mathbb{E}\left[\|f_{TTRP}(\boldsymbol{x})\|_2^4\right]-\left[\mathbb{E}[\|f_{TTRP}(\boldsymbol{x})\|_2^2]\right]^2\\ &= \mathbb{E}\left[\left\|\frac{1}{\sqrt{M}}\boldsymbol{y}\right\|_2^4\right]-\|\boldsymbol{x}\|_2^4\\ &= \frac{1}{M^2}\mathbb{E}\left[\|\boldsymbol{y}\|_2^4\right]-\|\boldsymbol{x}\|_2^4 \qquad (28)\\ &= \frac{1}{M^2}\left[\sum_{i=1}^{M}\mathbb{E}[\boldsymbol{y}^4(i)]+\sum_{i\neq j}\mathbb{E}[\boldsymbol{y}^2(i)\boldsymbol{y}^2(j)]\right]-\|\boldsymbol{x}\|_2^4, \qquad (29)\end{aligned}$$

where note that $\mathbb{E}[\boldsymbol{y}^2(i)\boldsymbol{y}^2(j)]\neq\mathbb{E}[\boldsymbol{y}^2(i)]\mathbb{E}[\boldsymbol{y}^2(j)]$ in general and a simple example can be found in Appendix A.

We compute the first term of the right hand side of Eq. (29),

$$\mathbb{E}[\boldsymbol{y}^4(i)]=\mathbb{E}\left[\mathscr{Y}(i_1,\ldots,i_d)\otimes\mathscr{Y}(i_1,\ldots,i_d)\otimes\mathscr{Y}(i_1,\ldots,i_d)\otimes\mathscr{Y}(i_1,\ldots,i_d)\right]\tag{30}$$

$$= \mathbb{E}\left[[\mathscr{Y}_1(i_1) \otimes \mathscr{Y}_1(i_1) \otimes \mathscr{Y}_1(i_1) \otimes \mathscr{Y}_1(i_1)] \cdots [\mathscr{Y}_d(i_d) \otimes \mathscr{Y}_d(i_d) \otimes \mathscr{Y}_d(i_d) \otimes \mathscr{Y}_d(i_d)]\right] \tag{31}$$

$$= \mathbb{E}[\mathscr{Y}_1(i_1) \otimes \mathscr{Y}_1(i_1) \otimes \mathscr{Y}_1(i_1) \otimes \mathscr{Y}_1(i_1)] \cdots \mathbb{E}[\mathscr{Y}_d(i_d) \otimes \mathscr{Y}_d(i_d) \otimes \mathscr{Y}_d(i_d) \otimes \mathscr{Y}_d(i_d)], \tag{32}$$

where $\boldsymbol{y}(i) = \mathscr{Y}(i_1, \ldots, i_d)$, applying Eq. (3) to Eq. (30) obtains Eq. (31), and we derive Eq. (32) from Eq. (31) by the independence of TT-cores $\{\mathscr{R}_k\}_{k=1}^d$.

Considering the k-th term of the right hand side of Eq. (32), for $k = 1, \ldots, d$, we obtain that

$$\begin{aligned}
&\mathbb{E}\left[\mathscr{Y}_k(i_k) \otimes \mathscr{Y}_k(i_k) \otimes \mathscr{Y}_k(i_k) \otimes \mathscr{Y}_k(i_k)\right] \\
&= \mathbb{E}\left[\left[\left[\sum_{j_k} \mathscr{R}_k(i_k, j_k) \otimes \mathscr{X}_k(j_k)\right] \otimes \left[\sum_{j_k} \mathscr{R}_k(i_k, j_k) \otimes \mathscr{X}_k(j_k)\right]\right.\right. \\
&\quad \left.\left. \otimes \left[\sum_{j_k} \mathscr{R}_k(i_k, j_k) \otimes \mathscr{X}_k(j_k)\right] \otimes \left[\sum_{j_k} \mathscr{R}_k(i_k, j_k) \otimes \mathscr{X}_k(j_k)\right]\right]\right] \qquad (33) \\
&= \mathbb{E}[[\sum_{j_k} \mathscr{R}_k(i_k, j_k) \mathscr{X}_k(j_k)] \otimes [\sum_{j_k} \mathscr{R}_k(i_k, j_k) \mathscr{X}_k(j_k)] \\
&\quad \otimes [\sum_{j_k} \mathscr{R}_k(i_k, j_k) \mathscr{X}_k(j_k)] \otimes [\sum_{j_k} \mathscr{R}_k(i_k, j_k) \mathscr{X}_k(j_k)]] \qquad (34) \\
&= \mathbb{E}\left[\sum_{j_k} \mathscr{R}_k^4(i_k, j_k) \mathscr{X}_k(j_k) \otimes \mathscr{X}_k(j_k) \otimes \mathscr{X}_k(j_k) \otimes \mathscr{X}_k(j_k)\right] \\
&\quad + \mathbb{E}\left[\sum_{j_k \neq j'_k} \mathscr{R}_k^2(i_k, j_k) \mathscr{R}_k^2(i_k, j'_k) \mathscr{X}_k(j_k) \otimes \mathscr{X}_k(j_k) \otimes \mathscr{X}_k(j'_k) \otimes \mathscr{X}_k(j'_k)\right] \\
&\quad + \mathbb{E}\left[\sum_{j_k \neq j'_k} \mathscr{R}_k^2(i_k, j_k) \mathscr{R}_k^2(i_k, j'_k) \mathscr{X}_k(j_k) \otimes \mathscr{X}_k(j'_k) \otimes \mathscr{X}_k(j_k) \otimes \mathscr{X}_k(j'_k)\right] \\
&\quad + \mathbb{E}\left[\sum_{j_k \neq j'_k} \mathscr{R}_k^2(i_k, j_k) \mathscr{R}_k^2(i_k, j'_k) \mathscr{X}_k(j_k) \otimes \mathscr{X}_k(j'_k) \otimes \mathscr{X}_k(j'_k) \otimes \mathscr{X}_k(j_k)\right] \qquad (35) \\
&= \Delta \sum_{j_k} \mathscr{X}_k(j_k) \otimes \mathscr{X}_k(j_k) \otimes \mathscr{X}_k(j_k) \otimes \mathscr{X}_k(j_k) + \sum_{j_k \neq j'_k} \mathscr{X}_k(j_k) \otimes \mathscr{X}_k(j_k) \otimes \mathscr{X}_k(j'_k) \otimes \mathscr{X}_k(j'_k) \\
&\quad + \sum_{j_k \neq j'_k} \mathscr{X}_k(j_k) \otimes \mathscr{X}_k(j'_k) \otimes \mathscr{X}_k(j_k) \otimes \mathscr{X}_k(j'_k) + \sum_{j_k \neq j'_k} \mathscr{X}_k(j_k) \otimes \mathscr{X}_k(j'_k) \otimes \mathscr{X}_k(j'_k) \otimes \mathscr{X}_k(j_k), \qquad (36)
\end{aligned}$$

where we infer Eq. (34) from Eq. (33) by scalar property of $\mathscr{R}_k(i_k, j_k)$, Eq. (35) is obtained by Eq. (4) and the independence of TT-cores $\{\mathscr{R}_k\}_{k=1}^d$, and denoting the fourth moment $\Delta := \mathbb{E}[\mathscr{R}_k^4(i_k, j_k)]$, we deduce Eq. (36) by the assumption $\mathbb{E}[\mathscr{R}_k^2(i_k, j_k)] = 1$, for $k = 1, \ldots, d$.

Substituting Eq. (36) into Eq. (32), it implies that

$$
\begin{aligned}
&\mathbb{E}\Big[\mathscr{Y}^4(i_1,\ldots,i_d)\Big]\\
=&\Big[\Delta\sum_{j_1}\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)+\sum_{j_1\neq j_1'}\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1')\otimes\mathscr{X}_1(j_1')\\
&+\sum_{j_1\neq j_1'}\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1')\otimes\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1')+\sum_{j_1\neq j_1'}\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1')\otimes\mathscr{X}_1(j_1')\otimes\mathscr{X}_1(j_1)\Big]\\
&\cdots\Big[\Delta\sum_{j_d}\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)+\sum_{j_d\neq j_d'}\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d')\otimes\mathscr{X}_d(j_d')\\
&+\sum_{j_d\neq j_d'}\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d')\otimes\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d')+\sum_{j_d\neq j_d'}\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d')\otimes\mathscr{X}_d(j_d')\otimes\mathscr{X}_d(j_d)\Big]\\
\le&\Delta^d\sum_{j_1,\ldots,j_d}\Big[\Big[\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\Big]\cdots\Big[\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\Big]\Big]\\
&+\Delta^{d-1}C_d^1\max_k\Big[\sum_{j_1,\ldots,j_k\neq j_k',\ldots,j_d}\Big[\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\Big]\cdots\\
&\Big[\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k')\otimes\mathscr{X}_k(j_k')\Big]\cdots\Big[\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\Big]\Big]\\
&+\Delta^{d-1}C_d^1\max_k\Big[\sum_{j_1,\ldots,j_k\neq j_k',\ldots,j_d}\Big[\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\Big]\cdots\\
&\Big[\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k')\otimes\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k')\Big]\cdots\Big[\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\Big]\Big]\\
&+\Delta^{d-1}C_d^1\max_k\Big[\sum_{j_1,\ldots,j_k\neq j_k',\ldots,j_d}\Big[\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\Big]\cdots\\
&\Big[\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k')\otimes\mathscr{X}_k(j_k')\otimes\mathscr{X}_k(j_k)\Big]\cdots\Big[\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\Big]\Big]+\cdots \qquad (37)\\
\le&\quad\Delta^d\sum_{j_1,\ldots,j_d}\mathscr{X}^4(j_1,\ldots,j_d)+3\Delta^{d-1}C_d^1\max_k[\sum_{j_1,\ldots,j_k\neq j'_k,\ldots,j_d}\mathscr{X}(j_1,\ldots,j_k,\ldots,j_d)^2\mathscr{X}(j_1,\ldots,j'_k,\ldots,j_d)^2]+\cdots \qquad (38)\\
\le&\Delta^d\|\boldsymbol{x}\|_4^4+3(n-1)\Delta^{d-1}C_d^1N\mathscr{M}^4+3^2(n-1)^2\Delta^{d-2}C_d^2N\mathscr{M}^4+\cdots+3^d(n-1)^dN\mathscr{M}^4\\
\le&(\Delta+3(n-1))^dN\mathscr{M}^4, \qquad (39)
\end{aligned}
$$

where denoting $\mathscr{M}:=\max_{i=1,\ldots,N}|\boldsymbol{x}(i)|$, $n=\max\{n_1,n_2,\ldots,n_d\}$, we derive Eq. (38) from Eq. (37) by Eq. (3).

Similarly, the second term $\mathbb{E}[\mathbf{y}^2(i)\mathbf{y}^2(j)]$ of the right hand side of Eq. (29), for $i\neq j$, $\nu(i)=(i_1,i_2,\ldots,i_d)\neq\nu(j)=(i_1',i_2',\ldots,i_d')$, is obtained by

$$\mathbb{E}[\boldsymbol{y}^2(i)\boldsymbol{y}^2(j)]$$
$$= \mathbb{E}\left[\mathscr{Y}_1(i_1)\otimes\mathscr{Y}_1(i_1)\otimes\mathscr{Y}_1(i'_1)\otimes\mathscr{Y}_1(i'_1)\right]\cdots\mathbb{E}\left[\mathscr{Y}_d(i_d)\otimes\mathscr{Y}_d(i_d)\otimes\mathscr{Y}_d(i'_d)\otimes\mathscr{Y}_d(i'_d)\right]. \tag{40}$$

If $i_k \neq i'_k$, for $k = 1,\ldots,d$, then the k-th term of the right hand side of Eq. (40) is computed by

$$\begin{aligned}
&\mathbb{E}\left[\mathscr{Y}_k(i_k)\otimes\mathscr{Y}_k(i_k)\otimes\mathscr{Y}_k(i'_k)\otimes\mathscr{Y}_k(i'_k)\right]\\
&=\mathbb{E}\left[\left[\sum_{j_k}\mathscr{R}_k(i_k,j_k)\mathscr{X}_k(j_k)\right]\otimes\left[\sum_{j_k}\mathscr{R}_k(i_k,j_k)\mathscr{X}_k(j_k)\right]\right.\\
&\left.\otimes\left[\sum_{j_k}\mathscr{R}_k(i'_k,j_k)\mathscr{X}_k(j_k)\right]\otimes\left[\sum_{j_k}\mathscr{R}_k(i'_k,j_k)\mathscr{X}_k(j_k)\right]\right]\\
&=\mathbb{E}\left[\sum_{j_k}\mathscr{R}_k^2(i_k,j_k)\mathscr{R}_k^2(i'_k,j_k)\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k)\right]\\
&+\mathbb{E}\left[\sum_{j_k\neq j'_k}\mathscr{R}_k^2(i_k,j_k)\mathscr{R}_k^2(i'_k,j'_k)\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j'_k)\otimes\mathscr{X}_k(j'_k)\right]\\
&=\sum_{j_k}\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k)+\sum_{j_k\neq j'_k}\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j'_k)\otimes\mathscr{X}_k(j'_k).
\end{aligned}\tag{41}$$

Supposing that $i_1 = i'_1,\ldots,i_k \neq i'_k,\ldots,i_d = i'_d$ and substituting Eqs. (36) and (41) into Eq. (40), we obtain

$$\begin{aligned}
&\mathbb{E}\Big[\mathbf{y}^2(i)\mathbf{y}^2(j)\Big]\\
&=\mathbb{E}\Big[\mathscr{Y}_1(i_1)\otimes\mathscr{Y}_1(i_1)\otimes\mathscr{Y}_1(i_1)\otimes\mathscr{Y}_1(i_1)\Big]\cdots\mathbb{E}\Big[\mathscr{Y}_k(i_k)\otimes\mathscr{Y}_k(i_k)\otimes\mathscr{Y}_k(i'_k)\otimes\mathscr{Y}_k(i'_k)\Big]\cdots\\
&\quad\mathbb{E}\Big[\mathscr{Y}_d(i_d)\otimes\mathscr{Y}_d(i_d)\otimes\mathscr{Y}_d(i_d)\otimes\mathscr{Y}_d(i_d)\Big]\\
&=\Big[\Delta\sum_{j_1}\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)+\sum_{j_1\neq j'_1}\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j'_1)\otimes\mathscr{X}_1(j'_1)\\
&\quad+\sum_{j_1\neq j'_1}\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j'_1)\otimes\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j'_1)+\sum_{j_1\neq j'_1}\mathscr{X}_1(j_1)\otimes\mathscr{X}_1(j'_1)\otimes\mathscr{X}_1(j'_1)\otimes\mathscr{X}_1(j_1)\Big]\\
&\quad\cdots\Big[\sum_{j_k}\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k)+\sum_{j_k\neq j'_k}\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j_k)\otimes\mathscr{X}_k(j'_k)\otimes\mathscr{X}_k(j'_k)\Big]\\
&\quad\cdots\Big[\Delta\sum_{j_d}\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)+\sum_{j_d\neq j'_d}\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j'_d)\otimes\mathscr{X}_d(j'_d)\\
&\quad+\sum_{j_d\neq j'_d}\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j'_d)\otimes\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j'_d)+\sum_{j_d\neq j'_d}\mathscr{X}_d(j_d)\otimes\mathscr{X}_d(j'_d)\otimes\mathscr{X}_d(j'_d)\otimes\mathscr{X}_d(j_d)\Big]
\end{aligned}$$

$$\leq n(\Delta + 3(n-1))^{d-1} N\mathscr{M}^4. \tag{42}$$

Similarly, if for $k \in S \subseteq \{1, \ldots, d\}$, $|S| = l$, $i_k \neq i'_k$, and for $k \in \bar{S}$, $i_k = i'_k$, then

$$\mathbb{E}[\mathbf{y}^2(i)\mathbf{y}^2(j)] \leq n^l(\Delta + 3(n-1))^{d-l} N\mathscr{M}^4. \tag{43}$$

Hence, combining Eqs. (42) and (43) gives

$$\begin{aligned}\sum_{i\neq j} \mathbb{E}[\mathbf{y}^2(i)\mathbf{y}^2(j)] \leq \quad & M[C_d^1(m-1)n(\Delta + 3(n-1))^{d-1} + \cdots + C_d^l(m-1)^l n^l(\Delta + 3(n-1))^{(d-l)} \\ & + \cdots + C_d^d(m-1)^d n^d] N\mathscr{M}^4,\end{aligned} \tag{44}$$

where $m = \max\{m_1, m_2, \ldots, m_d\}$.

Therefore, using Eqs. (39) and (44) deduces

$$\begin{aligned}\mathbb{E}[\|\boldsymbol{y}\|_2^4] \leq \quad & M[(\Delta + 3(n-1))^d + C_d^1(m-1)n(\Delta + 3(n-1))^{d-1} + \cdots + C_d^d(m-1)^d n^d] N\mathscr{M}^4 \\ & = M((m-1)n + \Delta + 3(n-1))^d N\mathscr{M}^4 \\ & = M(\Delta + n(m+2) - 3)^d N\mathscr{M}^4.\end{aligned} \tag{45}$$

In summary, substituting Eq. (45) into Eq. (28) implies

$$\begin{aligned}\mathrm{Var}(\|f_{TTRP}(\boldsymbol{x})\|_2^2) \leq \quad & \frac{M(\Delta + n(m+2) - 3)^d N\mathscr{M}^4}{M^2} - \|\boldsymbol{x}\|_2^4 \\ & \leq \frac{1}{M}(\Delta + n(m+2) - 3)^d N\mathscr{M}^4 - \|\boldsymbol{x}\|_2^4.\end{aligned} \tag{46}$$

One can see that the bound of the variance Eq. (46) is reduced as M increases, which is expected. When $M = m^d$ and $N = n^d$, we have

$$\mathrm{Var}(\|f_{TTRP}(\boldsymbol{x})\|_2^2) \quad \leq \left(\frac{\Delta + 2n - 3}{m} + n\right)^d N\mathscr{M}^4 - \|\boldsymbol{x}\|_2^4. \tag{47}$$

Note that as m increases, the upper bound in Eq. (47) tends to $(N^2\mathscr{M}^4 - \|\boldsymbol{x}\|_2^4) \geq 0$, and this upper bound vanishes as M increases if and only if $\boldsymbol{x}(1) = \boldsymbol{x}(2) = \ldots = \boldsymbol{x}(N)$. Also, the upper bound in Eq. (46) is affected by the fourth moment $\Delta = \mathbb{E}[\mathscr{R}_k^4(i_k, j_k)] = \mathrm{Var}(\mathscr{R}_k^2(i_k, j_k)) + [\mathbb{E}[\mathscr{R}_k^2(i_k, j_k)]]^2$. To keep the expected isometry, we need $\mathbb{E}[\mathscr{R}_k^2(i_k, j_k)] = 1$. Note that when the TT-cores follow the Rademacher distribution, i.e., $\mathrm{Var}(\mathscr{R}_k^2(i_k, j_k)) = 0$, the fourth moment Δ in Eq. (46) achieves the minimum. So, the Rademacher distribution is an optimal choice for generating the TT-cores, and we set the Rademacher distribution to be our default choice for constructing TTRP (Definition 3.1).

Proposition 3.1. (Hypercontractivity [35]) Consider a degree q polynomial $f(Y) = f(Y_1, \ldots, Y_n)$ of independent centered Gaussian or Rademacher random variables $Y_1, \ldots, Y_n$. Then for any $\lambda > 0$

$$\mathbb{P}(|f(Y) - \mathbb{E}[f(Y)]| \geq \lambda) \leq e^2 \cdot \exp\left[-\left(\frac{\lambda^2}{K \cdot \mathrm{Var}[f(Y)]}\right)^{\frac{1}{q}}\right],$$

where $\mathrm{Var}([f(Y)])$ is the variance of the random variable $f(Y)$ and $K > 0$ is an absolute constant.

Proposition 3.1 extends the Hanson-Wright inequality whose proof can be found in [35].

Proposition 3.2. Let $f_{TTRP}: \mathbb{R}^N \mapsto \mathbb{R}^M$ be the tensor train random projection defined by Eq. (17). Suppose that for $i = 1, \ldots, d$, all entries of TT-cores $\mathscr{R}_i$ are independent standard Gaussian or Rademacher random variables, with the same fourth moment Δ and $\mathscr{M} := \max_{i=1,\ldots,N} |\boldsymbol{x}(i)|$, $m = \max\{m_1, m_2, \ldots, m_d\}$, $n = \max\{n_1, n_2, \ldots, n_d\}$. For any $\boldsymbol{x} \in \mathbb{R}^N$, there exist absolute constants C and $K > 0$ such that the following claim holds

$$\mathbb{P}(|\ \|f_{TTRP}(\boldsymbol{x})\|_2^2 - \ \|\boldsymbol{x}\|_2^2| \geq \varepsilon\ \|\boldsymbol{x}\|_2^2) \leq C \exp\left[-\left(\frac{M\cdot\varepsilon^2}{K\cdot\left[(\Delta+n(m+2)-3)^d N - M\right]}\right)^{\frac{1}{2d}}\right].$$

Proof. According to Theorem 3.1, $\mathbb{E}\,\|f_{TTRP}(\boldsymbol{x})\|_2^2 = \|\boldsymbol{x}\|_2^2$. Since $\|f_{TTRP}(\boldsymbol{x})\|_2^2$ is a polynomial of degree $2d$ of independent standard Gaussian or Radamecher random variables, which are the entries of TT-cores $\mathscr{R}_i$, for $i = 1,\ldots,d$, we apply Proposition 3.1 and Theorem 3.2 to obtain

$$\begin{aligned}
\mathbb{P}(|\ \|f_{TTRP}(\boldsymbol{x})\|_2^2 - \|\boldsymbol{x}\|_2^2| \geq \varepsilon\,\|\boldsymbol{x}\|_2^2) &\leq e^2\cdot\exp\left[-\left(\frac{\varepsilon^2\,\|\boldsymbol{x}\|_2^4}{K\cdot\mathrm{Var}\left(\|f_{TTRP}(\boldsymbol{x})\|_2^2\right)}\right)^{\frac{1}{2d}}\right]\\
&\leq e^2\cdot\exp\left[-\left(\frac{\varepsilon^2}{K\cdot\left[\frac{1}{M}(\Delta+n(m+2)-3)^d N\frac{\mathscr{M}^4}{\|\boldsymbol{x}\|_2^4}-1\right]}\right)^{\frac{1}{2d}}\right]\\
&\leq e^2\cdot\exp\left[-\left(\frac{M\cdot\varepsilon^2}{K\cdot\left[(\Delta+n(m+2)-3)^d N - M\right]}\right)^{\frac{1}{2d}}\right]\\
&\leq C\exp\left[-\left(\frac{M\cdot\varepsilon^2}{K\cdot\left[(\Delta+n(m+2)-3)^d N - M\right]}\right)^{\frac{1}{2d}}\right],
\end{aligned}$$

where $\mathscr{M} = \max_{i=1,\ldots,N}|\boldsymbol{x}(i)|$ and then $\frac{\mathscr{M}^4}{\|\boldsymbol{x}\|_2^4} \leq 1$.

We note that the upper bound in Proposition 3.2 is not tight, as it involves the dimensionality of datasets (N). To give a tight bound independent of the dimensionality of datasets for the corresponding concentration inequality is our future work.

The procedure of TTRP is summarized in Algorithm 2. For the input of this algorithm, the TT-ranks of $\mathscr{R}$ (the tensorized version of the projection matrix $\boldsymbol{R}$ in Eq. (17)) are set to one, and from our above analysis, we generate entries of the corresponding TT-cores $\{\mathscr{R}_k\}_{k=1}^d$ through the Rademacher distribution. For a given data point $\boldsymbol{x}$ in the TT-format, Algorithm 2 gives the TT-cores of the corresponding output, and each element of $f_{TTRP}(\boldsymbol{x})$ in Eq. (17) can be represented as

$$f_{TTRP}(\boldsymbol{x})(i) = f_{TTRP}(\boldsymbol{x})(\nu(i)) = f_{TTRP}(\boldsymbol{x})(i_1,\ldots,i_d) = \frac{1}{\sqrt{M}}\mathscr{Y}_1(i_1)\cdots\mathscr{Y}_d(i_d),$$

where ν is a bijection from $\mathbb{N}$ to $\mathbb{N}^d$.

4 Numerical Experiments

We demonstrate the efficiency of TTRP using synthetic datasets and the MNIST dataset [36]. The quality of isometry is a key factor to assess the performance of random projection methods, which in our numerical studies is estimated by the ratio of the pairwise distance:

$$\frac{2}{n_0(n_0-1)}\sum_{n_0\geq i>j}\frac{\left\|f_{TTRP}(\boldsymbol{x}^{(i)}) - f_{TTRP}(\boldsymbol{x}^{(j)})\right\|_2}{\|\boldsymbol{x}^{(i)}-\boldsymbol{x}^{(j)}\|_2}, \tag{48}$$

where n_0 is the number of data points. Since the output of our TTRP procedure (see Algorithm 2) is in the TT-format, it is efficient to apply TT-format operations to compute the pairwise distance of Eq. (48) through Algorithm 1. In order to obtain the average performance of isometry, we repeat numerical experiments 100 times (different realizations for TT-cores) and estimate the mean and the variance for the ratio of the pairwise distance using these samples. The rest of this section is organized as follows. First, through a synthetic dataset, the effect of different TT-ranks of the tensorized version $\mathscr{R}$ of $\boldsymbol{R}$ in Eq. (17) is shown, which leads to our motivation of setting the TT-ranks to be one. After that, we focus on the situation with TT-ranks equal to one, and test the effect of different distributions of TT-cores. Finally, based on both high-dimensional synthetic and MNIST datasets, our TTRP are compared with related projection methods, including Gaussian TRP [16], Very Sparse RP [14] and Gaussian RP [30]. All results reported below are obtained in MATLAB with TT-Toolbox 2.2.2 (https://github.com/oseledets/TT-Toolbox).

Algorithm 2: Tensor train random projection

Input: TT-cores $\mathscr{R}_k(i_k, j_k)$ of $\boldsymbol{R}$, and TT-cores $\mathscr{X}_k$ of $\boldsymbol{x}$, for $k = 1, \ldots, d$.

1: **for** $k = 1 : d$ **do**

2: **for** $i_k = 1 : m_k$ **do**

3: Compute $\mathscr{Y}_k(i_k) = \sum_{j_k=1}^{n_k} (\mathscr{R}_k(i_k, j_k) \otimes \mathscr{X}_k(j_k))$. ▷ $O(n\hat{r}^2)$ by Eq. (9)

4: **end for**

5: **end for**

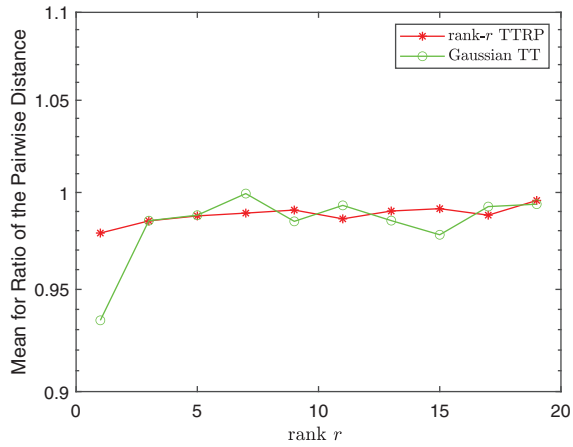
Output: TT-cores $\frac{1}{\sqrt{M}}\mathscr{Y}_1, \mathscr{Y}_2, \ldots, \mathscr{Y}_d$.

4.1 Effect of Different TT-Ranks

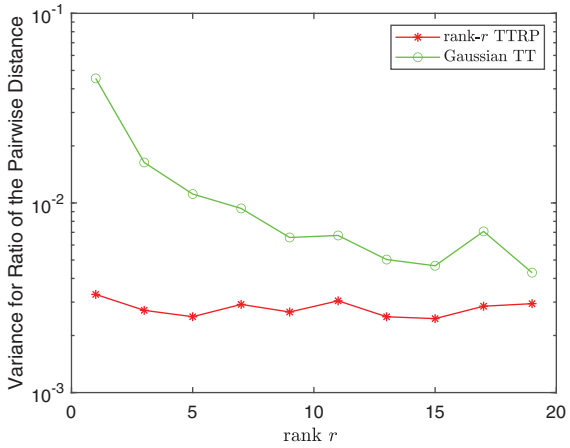
In Definition 3.1, we set the TT-ranks to be one. To explain our motivation of this settting, we investigate the effect of different TT-ranks—we herein consider the situation that the TT-ranks take $r_0 = r_d = 1, r_k = r, k = 1, \ldots, d-1$, where the rank $r \in \{1, 2, \ldots\}$, and we keep other settings in Definition 3.1 unchanged. For comparison, two different distributions are considered to generate the TT-cores in this part—the Rademacher distribution (our default optimal choice) and the Gaussian distribution, and the corresponding tensor train projection is denoted by rank-r TTRP and Gaussian TT, respectively. To keep the expected isometry of rank-r TTRP, the entries of TT-cores $\mathscr{R}_1$ and $\mathscr{R}_d$ are drawn from $1/r^{1/4}$ or $-1/r^{1/4}$ with equal probability, and each element of $\mathscr{R}_k, k = 2, .., d-1$ is uniformly and independently drawn from $1/r^{1/2}$ or $-1/r^{1/2}$. For Gaussian TT, $\mathscr{R}_k$ for $k = 2, \ldots, d-1$ and $k = 1, d$ are drawn from $\mathcal{N}(0, \frac{1}{r})$ and $\mathcal{N}(0, \frac{1}{\sqrt{r}})$, respectively.

Two synthetic datasets with dimension $N = 1000$ are generated (the size of the first one is $n_0 = 10$ and that of the second one is $n_0 = 50$), where each entry of vectors (each vector is a sample in the synthetic dataset) is independently generated through $\mathcal{N}(0, 1)$. In this test problem, we set the reduced dimension to be $M = 24$, and the dimensions of the corresponding tensor representations are set to $m_1 = 4, m_2 = 3, m_3 = 2$ and $n_1 = n_2 = n_3 = 10$ ($M = m_1 m_2 m_3$ and $N = n_1 n_2 n_3$). Fig. 2 shows the ratio of the pairwise distance of the two projection methods (computed through Eq. (48)). It can be seen that the estimated mean of ratio of the pairwise distance of rank-r TTRP is typically more close to one than that of Gaussian TT, i.e., rank-r TTRP has advantages for keeping the pairwise distances. Clearly, for a given rank in Fig. 2, the estimated variance of the pairwise distance of rank-r TTRP is smaller than that of Gaussian TT. Moreover, focusing on rank-r TTRP, the results of both the mean

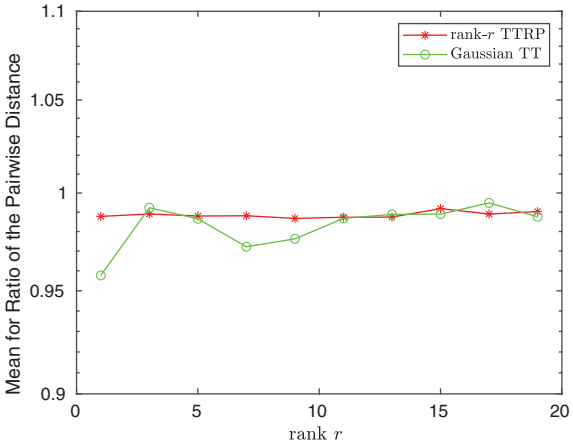
and the variance are not significantly different for different TT-ranks. In order to reduce the storage, we only focus on the rank-one case (as in Definition 3.1) in the rest of this paper.



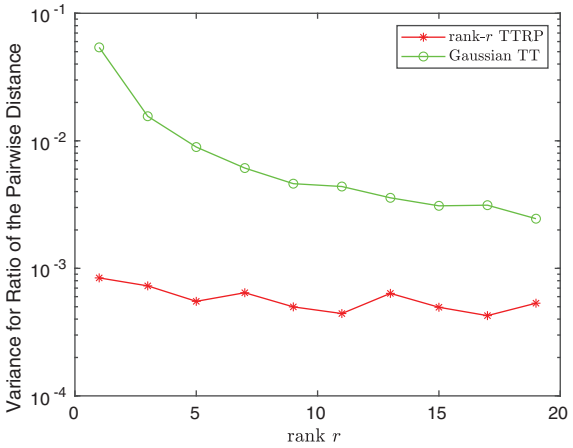
(a) Mean, $n_0 = 10$.



(b) Variance, $n_0 = 10$.



(c) Mean, $n_0 = 50$.



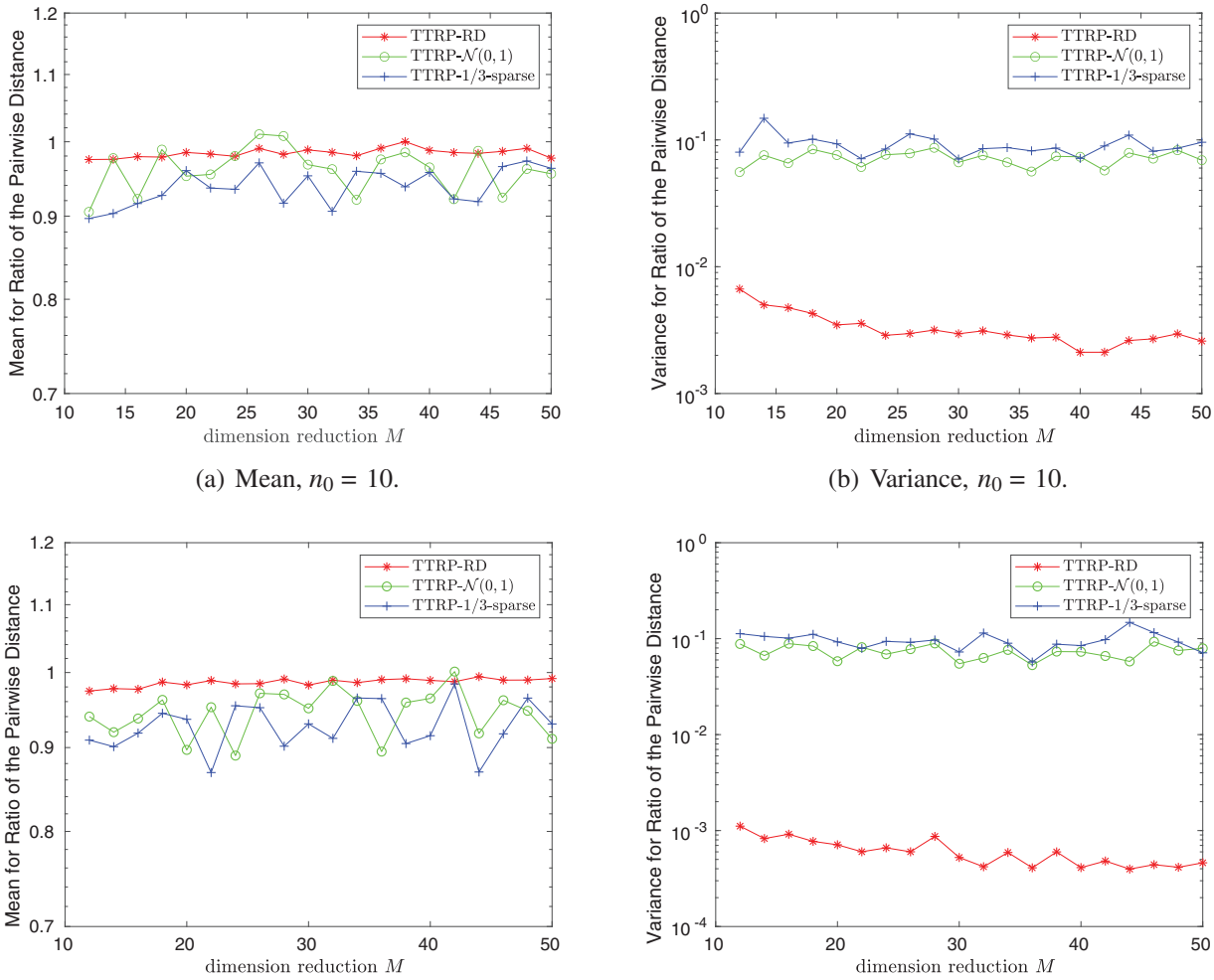
(d) Variance, $n_0 = 50$.

Figure 2: Effect of different ranks based on synthetic data ($M = 24, N = 1000, m_1 = 4, m_2 = 3, m_3 = 2, n_1 = n_2 = n_3 = 10$)

4.2 Effect of Different TT-Cores

Two synthetic datasets are tested to assess the effect of different distributions for TT-cores, whose sizes are $n_0 = 10$ and $n_0 = 50$, respectively, with dimension $N = 2500$, whose elements are sampled from the standard Gaussian distribution. The following three distributions are investigated to construct TTRP (see Definition 3.1), which include the Rademacher distribution (our default choice), the standard Gaussian distribution, and the 1/3-sparse distribution (i.e., $s = 3$ in Eq. (2)), while the corresponding projection methods are denoted by TTRP-RD, TTRP-$\mathcal{N}(0, 1)$, and TTRP-1/3-sparse, respectively. For this test problem, three TT-cores are utilized for $m_1 = M/2, m_2 = 2, m_3 = 1$ and $n_1 = 25, n_2 = 10, n_3 = 10$. Fig. 3 shows that the estimated mean of the ratio of the pairwise distance for TTRP-RD is very close to one, and the estimated variance of TTRP-RD is at least one order of

magnitude smaller than that of TTRP-$\mathcal{N}(0,1)$ and TTRP-1/3-sparse. These results are consist with Theorem 3.2. In the rest of this paper, we focus on our default choice for TTRP—the TT-ranks are set to one, and each element of TT-cores is independently sampled through the Rademacher distribution.



(a) Mean, $n_0 = 10$. (b) Variance, $n_0 = 10$.

(c) Mean, $n_0 = 50$. (d) Variance, $n_0 = 50$.

Figure 3: Three test distributions for TT-cores based on synthetic data ($N = 2500$)

4.3 Comparison with Gaussian TRP, Very Sparse RP and Gaussian RP

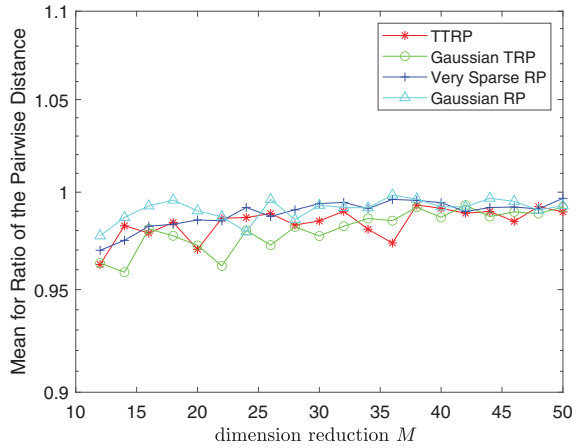
The storage of the projection matrix and the cost of computing $\boldsymbol{Rx}$ (see Eq. (17)) of our TTRP (TT-ranks equal one), Gaussian TRP [16], Very Sparse RP [14] and Gaussian RP [30], are shown in Table 1, where $\boldsymbol{R} \in \mathbb{R}^{M\times N}$, $M = \prod_{i=1}^{d} m_i$, $N = \prod_{j=1}^{d} n_j$, $m = \max\{m_1, m_2, \ldots, m_d\}$ and $n = \max\{n_1, n_2, \ldots, n_d\}$. Note that the matrix $\boldsymbol{R}$ in Eq. (17) is tensorized in the TT-format, and TTRP is efficiently achieved by the matrix-by-vector products in the TT-format (see Eq. (9)). From Table 1, it is clear that our TTRP has the smallest storage cost and requires the smallest computational cost for computing $\boldsymbol{Rx}$.

Table 1: The comparison of the storage and the computational costs

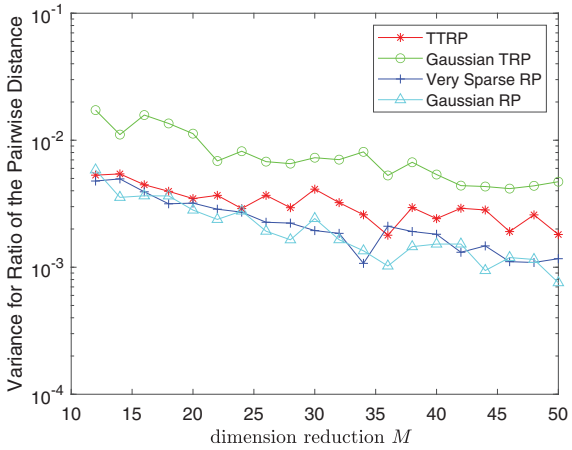
	Gaussian RP	Very sparse RP	Gaussian TRP	TTRP
Storage cost	$O(MN)$	$O(M\sqrt{N})$	$O(dMn)$	$O(dmn)$
Computational cost	$O(MN)$	$O(M\sqrt{N})$	$O(MN)$	$O(dmn\hat{r}^2)$

Two synthetic datasets with size $n_0 = 10$ are tested—the dimension of the first one is $N = 2500$ and that of the second one is $N = 10^4$; each entry of the samples is independently generated through $\mathcal{N}(0, 1)$. For TTRP and Gaussian TRP, the dimensions of tensor representations are set to: for $N = 2500$, we set $n_1 = 25, n_2 = 10, n_3 = 10, m_1 = M/2, m_2 = 2, m_3 = 1$; for $N = 10^4$, we set $n_1 = n_2 = 25, n_3 = n_4 = 4, m_1 = M/2, m_2 = 2, m_3 = 1, m_4 = 1$, where M is the reduced dimension. We again focus on the ratio of the pairwise distance (putting the outputs of different projection methods into Eq. (48)), and estimate the mean and the variance for the ratio of the pairwise distance through repeating numerical experiments 100 times (different realizations for constructing the random projections, e.g., different realizations of the Rademacher distribution for TTRP).

Fig. 4 shows that the performance of TTRP is very close to that of sparse RP and Gaussian RP, while the variance for Gaussian TRP is larger than that for the other three projection methods. Moreover, the variance for TTRP basically reduces as the dimension M increases, which is consistent with Theorem 3.2. To be further, more details are given for the case with $M = 24$ and $N = 10^4$ in Tables 2 and 3, where the value of storage is the number of nonzero entries that need to be stored. It turns out that TTRP with fewer storage costs achieves a competitive performance compared with Very Sparse RP and Gaussian RP. In addition, from Table 3, for $d > 2$, the variance of TTRP is clearly smaller than that of Gaussian TRP, and the storage cost of TTRP is much smaller than that of Gaussian TRP.

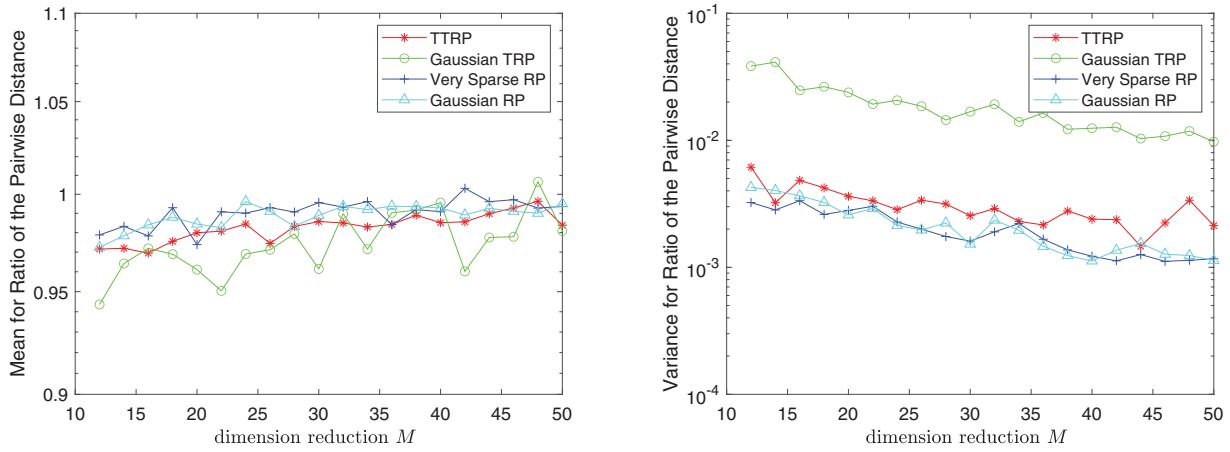


(a) Mean, $N = 2500$.



(b) Variance, $N = 2500$.

Figure 4: (Continued)



(c) Mean, $N = 10^4$. (d) Variance, $N = 10^4$.

Figure 4: Mean and variance for the ratio of the pairwise distance, synthetic data

Table 2: The comparison of mean and variance for the ratio of the pairwise distance, and storage, for Gaussian RP and Very Sparse RP ($M = 24$ and $N = 10^4$)

Gaussian RP			Very Sparse RP		
Mean	Variance	Storage	Mean	Variance	Storage
0.9908	0.0032	240000	0.9963	0.0025	2400

Table 3: The comparison of mean and variance for the ratio of the pairwise distance, and storage, for Gaussian TRP and TTRP ($M = 24$ and $N = 10^4$)

Dimensions for tensorization		Gaussian TRP			TTRP		
$[m_1, \ldots, m_d]$	$[n_1, \ldots, n_d]$	Mean	Variance	Storage	Mean	Variance	Storage
[6, 4]	[100, 100]	0.9908	0.0026	4800	0.9884	0.0026	1000
[4, 3, 2]	[25, 20, 20]	0.9747	0.0062	1560	0.9846	0.0028	200
[3, 2, 2, 2]	[10, 10, 10, 10]	0.9811	0.0123	960	0.9851	0.0035	90

Next the CPU times for projecting a data point using the four methods (TTRP, Gaussian TRP, Very Sparse RP and Gaussian RP) are assessed. Here, we set the reduced dimension $M = 1000$, and test four cases with $N = 10^4$, $N = 10^5$, $N = 2 \times 10^5$ and $N = 10^6$, respectively. The dimensions of the tensorized output is set to $m_1 = m_2 = m_3 = 10$ (such that $M = m_1 m_2 m_3$), and the dimensions of the corresponding tensor representations of the original data points are set to: for $N = 10^4$, $n_1 = 25$, $n_2 = 25$, $n_3 = 16$; for $N = 10^5$, $n_1 = 50$, $n_2 = 50$, $n_3 = 40$; for $N = 2 \times 10^5$, $n_1 = 80$, $n_2 = 50$, $n_3 = 50$; for $N = 10^6$, $n_1 = n_2 = n_3 = 100$. For each case, given a data point of which elements are sampled from the standard Gaussian distribution, the simulation of projecting it to the reduced dimensional space is repeated 100 times (different realizations for constructing the random projections), and the CPU time

is defined to be the average time of these 100 simulations. Fig. 5 shows the CPU times, where the results are obtained in MATLAB on a workstation with Intel(R) Xeon(R) Gold 6130 CPU. It is clear that the computational cost of our TTRP is much smaller than those of Gaussian TRP and Gaussian RP for different data dimension N. Note that as the data dimension N increases, the computational costs of Gaussian TRP and Gaussian RP grow rapidly, while the computational cost of our TTRP grows slowly. When the data dimension is large (e.g., $N = 10^6$ in Fig. 5), the CPU time of TTRP becomes smaller than that of Very Sparse RP, which is consistent with the results in Table 1.

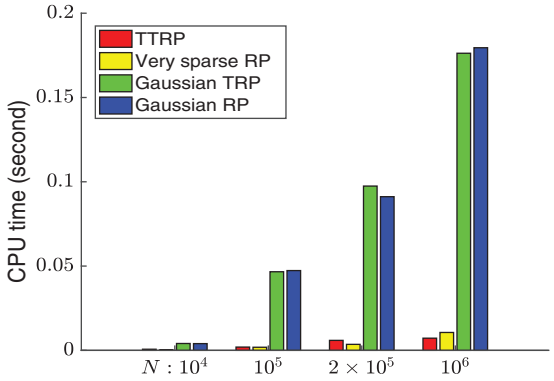
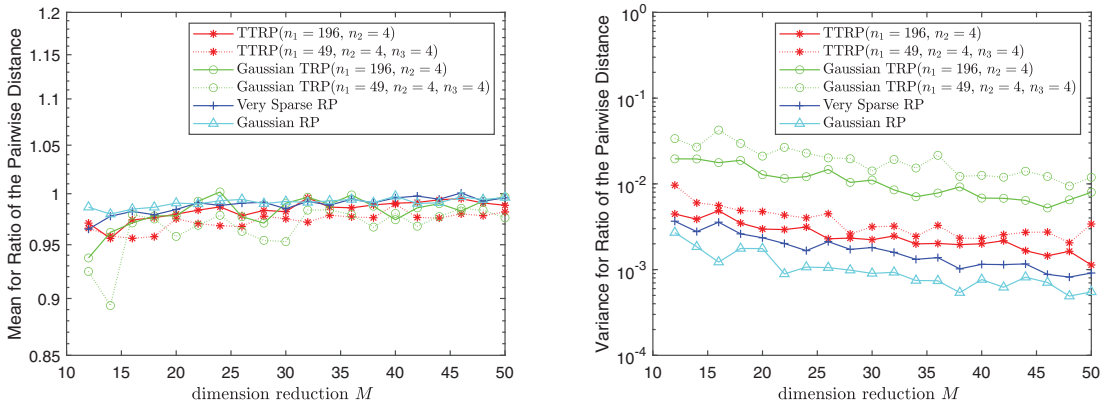


Figure 5: A comparison of CPU time for different random projections ($M = 1000$)

Finally, we validate the performance of our TTRP approach using the MNIST dataset [36]. From MNIST, we randomly take $n_0 = 50$ data points, each of which is a vector with dimension $N = 784$. We consider two cases for the dimensions of tensor representations: in the first case, we set $m_1 = M/2, m_2 = 2, n_1 = 196, n_2 = 4$, and in the second case, we set $m_1 = M/2, m_2 = 2$, $m_3 = 1, n_1 = 49, n_2 = 4, n_3 = 4$. Fig. 6 shows the properties of isometry and bounded variance of different random projections on MNIST. It can be seen that TTRP satisfies the isometry property with bounded variance. Although Gaussian RP has the smallest variance (note that it is not based on the tensor format), its computational and storage costs are large (see Fig. 5). It is clear that as the reduced dimension M increases, the variances of the four methods reduce, and the variance of our TTRP is close to that of Very Sparse RP.



(a) Mean for the ratio of the pairwise distance (b) Variance for the ratio of the pairwise distance

Figure 6: Isometry and variance quality for MNIST data ($N = 784$)

5 Conclusion

Random projection plays a fundamental role in conducting dimension reduction for high-dimensional datasets, where pairwise distances need to be approximately preserved. With a focus on efficient tensorized computation, this paper develops a tensor train random projection (TTRP) method. Based on our analysis for the bias and the variance, TTRP is proven to be an expected isometric projection with bounded variance. From the analysis in Theorem 3.2, the Rademacher distribution is shown to be an optimal choice to generate the TT-cores of TTRP. For computational convenience, the TT-ranks of TTRP are set to one, while from our numerical results, we show that different TT-ranks do not lead to significant results for the mean and the variance of the ratio of the pairwise distance. Our detailed numerical studies show that, compared with standard projection methods, our TTRP with the default setting (TT-ranks equal one and TT-cores are generated through the Rademacher distribution), requires significantly smaller storage and computational costs to achieve a competitive performance. From numerical results, we also find that our TTRP has smaller variances than tensor train random projection methods based on Gaussian distributions. Even though we have proven the properties of the mean and the variance of TTRP and the numerical results show that TTRP is efficient, the upper bound in Proposition 3.2 involves the dimensionality of datasets (N), and our future work is to give a tight bound independent of the dimensionality of datasets for the concentration inequality.

Acknowledgement: The authors thank Osman Asif Malik and Stephen Becker for helpful suggestions and discussions.

Funding Statement: This work is supported by the National Natural Science Foundation of China (No. 12071291), the Science and Technology Commission of Shanghai Municipality (No. 20JC1414300) and the Natural Science Foundation of Shanghai (No. 20ZR1436200).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Wold, S., Esbensen, K., Geladi, P. (1987). Principal component analysis. *Chemometrics and Intelligent Laboratory Systems, 2(1–3),* 37–52. DOI 10.1016/0169-7439(87)80084-9.
2. Vidal, R., Ma, Y., Sastry, S. S. (2016). *Generalized principal component analysis*. New York: Springer.
3. Sra, S., Dhillon, I. (2005). Generalized nonnegative matrix approximations with bregman divergences. *Proceedings of the 18th International Conference on Neural Information Processing Systems*, pp. 283–290. Vancouver, Canada.
4. van der Maaten, L., Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research, 9(11),* 2579–2605.
5. Pham, N., Pagh, R. (2013). Fast and scalable polynomial kernels via explicit feature maps. *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 239–247. Chicago, USA.
6. Johnson, W. B., Lindenstrauss, J. (1984). Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics, 26*. DOI 10.1090/conm/026/737400.
7. Dasgupta, S., Gupta, A. (2003). An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures and Algorithms, 22(1),* 60–65. DOI 10.1002/(ISSN)1098-2418.

8. Kleinberg, J. M. (1997). Two algorithms for nearest-neighbor search in high dimensions. *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pp. 599–608. El Paso, USA.
9. Ailon, N., Chazelle, B. (2006). Approximate nearest neighbors and the fast Johnson–Lindenstrauss transform. *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, pp. 557–563. Seattle, USA.
10. Baraniuk, R., Davenport, M., DeVore, R., Wakin, M. (2008). A simple proof of the restricted isometry property for random matrices. *Constructive Approximation, 28(3),* 253–263. DOI 10.1007/s00365-007-9003-x.
11. Krahmer, F., Ward, R. (2011). New and improved Johnson–Lindenstrauss embeddings via the restricted isometry property. *SIAM Journal on Mathematical Analysis, 43(3),* 1269–1281. DOI 10.1137/100810447.
12. Candès, E. J., Romberg, J., Tao, T. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory, 52(2),* 489–509. DOI 10.1109/TIT.2005.862083.
13. Achlioptas, D. (2003). Database-friendly random projections: Johnson–Lindenstrauss with binary coins. *Journal of Computer and System Sciences, 66(4),* 671–687. DOI 10.1016/S0022-0000(03)00025-4.
14. Li, P., Hastie, T. J., Church, K. W. (2006). Very sparse random projections. *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 287–296. Philadelphia, USA.
15. Ailon, N., Chazelle, B. (2009). The fast Johnson–Lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on Computing, 39(1),* 302–322. DOI 10.1137/060673096.
16. Sun, Y., Guo, Y., Tropp, J. A., Udell, M. (2021). Tensor random projection for low memory dimension reduction. arXiv preprint arXiv:2105.00105.
17. Jin, R., Kolda, T. G., Ward, R. (2021). Faster Johnson–Lindenstrauss transforms via Kronecker products. *Information and Inference: A Journal of the IMA, 10(4),* 1533–1562. DOI 10.1093/imaiai/iaaa028.
18. Malik, O. A., Becker, S. (2020). Guarantees for the kronecker fast Johnson–Lindenstrauss transform using a coherence and sampling argument. *Linear Algebra and its Applications, 602,* 120–137. DOI 10.1016/j.laa.2020.05.004.
19. Kolda, T. G., Bader, B. W. (2009). Tensor decompositions and applications. *SIAM Review, 51(3),* 455–500. DOI 10.1137/07070111X.
20. Acar, E., Dunlavy, D. M., Kolda, T. G., MArup, M. (2010). Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems, 106(1),* 41–56. DOI 10.1016/j.chemolab.2010.08.004.
21. Austin, W., Ballard, G., Kolda, T. G. (2016). Parallel tensor compression for large-scale scientific data. *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 912–922. Chicago, USA.
22. Ahle, T. D., Kapralov, M., Knudsen, J. B., Pagh, R., Velingker, A. et al. (2020). Oblivious sketching of high-degree polynomial kernels. *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 141–160. DOI 10.1137/1.9781611975994.9.
23. Tang, K., Liao, Q. (2020). Rank adaptive tensor recovery based model reduction for partial differential equations with high-dimensional random inputs. *Journal of Computational Physics, 409,* 109326. DOI 10.1016/j.jcp.2020.109326.
24. Cui, T., Dolgov, S. (2021). Deep composition of tensor-trains using squared inverse rosenblatt transports. *Foundations of Computational Mathematics, 21,* 1–60. DOI 10.1007/s10208-021-09537-5.
25. White, S. R. (1992). Density matrix formulation for quantum renormalization groups. *Physical Review Letters, 69(19),* 2863. DOI 10.1103/PhysRevLett.69.2863.
26. Perez-Garcia, D., Verstraete, F., Wolf, M., Cirac, J. (2007). Matrix product state representations. *Quantum Information & Computation, 7(5–6),* 401–430. DOI 10.26421/QIC.

27. Verstraete, F., Murg, V., Cirac, J. I. (2008). Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in Physics, 57(2),* 143–224. DOI 10.1080/14789940801912366.
28. Orús, R. (2014). A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics, 349,* 117–158. DOI 10.1016/j.aop.2014.06.013.
29. Oseledets, I. V. (2011). Tensor-train decomposition. *SIAM Journal on Scientific Computing, 33(5),* 2295–2317. DOI 10.1137/090752286.
30. Achlioptas, D. (2001). Database-friendly random projections. *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 274–281. Santa Barbara, USA.
31. Rakhshan, B., Rabusseau, G. (2020). Tensorized random projections. *International Conference on Artificial Intelligence and Statistics*, pp. 3306–3316. Palermo, Italy.
32. van Loan, C. F. (2000). The ubiquitous kronecker product. *Journal of Computational and Applied Mathematics, 123(1–2),* 85–100. DOI 10.1016/S0377-0427(00)00393-9.
33. Novikov, A., Podoprikhin, D., Osokin, A., Vetrov, D. P. (2015). Tensorizing neural networks. *Advances in Neural Information Processing Systems, 28,* 442–450.
34. Golub, G. H., van Loan, C. F. (2013). *Matrix computations*. Baltimore: The Johns Hopkins University Press.
35. Schudy, W., Sviridenko, M. (2012). Concentration and moment inequalities for polynomials of independent random variables. *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 437–446. Kyoto, Japan.
36. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86(11),* 2278–2324. DOI 10.1109/5.726791.

Appendix A

Example for $\mathbb{E}[\boldsymbol{y}^2(i)\boldsymbol{y}^2(j)] \neq \mathbb{E}[\boldsymbol{y}^2(i)]\mathbb{E}[\boldsymbol{y}^2(j)]$, $i \neq j$.

If all TT-ranks of tensorized matrix $\boldsymbol{R}$ in Eq. (17) are equal to one, then $\boldsymbol{R}$ is represented as a Kronecker product of d matrices [29],

$$\boldsymbol{R} = \boldsymbol{R}_1 \otimes \boldsymbol{R}_2 \otimes \cdots \otimes \boldsymbol{R}_d, \tag{49}$$

where $\boldsymbol{R}_i \in \mathbb{R}^{m_i \times n_i}$, for $i = 1, 2, .., d$, whose entries are i.i.d. mean zero and variance one.

We set $d = 2, m_1 = m_2 = n_1 = n_2 = 2$ in Eq. (49), then

$$\boldsymbol{y} = \boldsymbol{R}x = (\boldsymbol{R}_1 \otimes \boldsymbol{R}_2)\boldsymbol{x}, \tag{50}$$

where

$$\boldsymbol{R}_1 = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix}, \boldsymbol{R}_2 = \begin{bmatrix} c_1 & c_2 \\ d_1 & d_2 \end{bmatrix}.$$

Substituting $\boldsymbol{x}$: $= [\boldsymbol{x}(1)\ \boldsymbol{x}(2)\ \boldsymbol{x}(3)\ \boldsymbol{x}(4)]^T$ and $\boldsymbol{y}$: $= [\boldsymbol{y}(1)\ \boldsymbol{y}(2)\ \boldsymbol{y}(3)\ \boldsymbol{y}(4)]^T$ into Eq. (50), we obtain

$$\begin{bmatrix} \boldsymbol{y}(1) \\ \boldsymbol{y}(2) \\ \boldsymbol{y}(3) \\ \boldsymbol{y}(4) \end{bmatrix} = \begin{bmatrix} a_1c_1\boldsymbol{x}(1) + a_1c_2\boldsymbol{x}(2) + a_2c_1\boldsymbol{x}(3) + a_2c_2\boldsymbol{x}(4) \\ a_1d_1\boldsymbol{x}(1) + a_1d_2\boldsymbol{x}(2) + a_2d_1\boldsymbol{x}(3) + a_2d_2\boldsymbol{x}(4) \\ b_1c_1\boldsymbol{x}(1) + b_1c_2\boldsymbol{x}(2) + b_2c_1\boldsymbol{x}(3) + b_2c_2\boldsymbol{x}(4) \\ b_1d_1\boldsymbol{x}(1) + b_1d_2\boldsymbol{x}(2) + b_2d_1\boldsymbol{x}(3) + b_2d_2\boldsymbol{x}(4) \end{bmatrix}. \tag{51}$$

Applying Eq. (51) to $\mathrm{cov}(\boldsymbol{y}^2(1), \boldsymbol{y}^2(3))$ gives

$$
\begin{aligned}
&\mathrm{cov}(\boldsymbol{y}^2(1), \boldsymbol{y}^2(3)) \\
&= \mathrm{cov}\left((a_1c_1\boldsymbol{x}(1) + a_1c_2\boldsymbol{x}(2) + a_2c_1\boldsymbol{x}(3) + a_2c_2\boldsymbol{x}(4))^2, (b_1c_1\boldsymbol{x}(1) + b_1c_2\boldsymbol{x}(2) + b_2c_1\boldsymbol{x}(3) + b_2c_2\boldsymbol{x}(4))^2\right) \\
&= \mathrm{cov}\left(a_1^2c_1^2\boldsymbol{x}(1)^2 + a_1^2c_2^2\boldsymbol{x}(2)^2 + a_2^2c_1^2\boldsymbol{x}(3)^2 + a_2^2c_2^2\boldsymbol{x}(4)^2, b_1^2c_1^2\boldsymbol{x}(1)^2 + b_1^2c_2^2\boldsymbol{x}(2)^2 + b_2^2c_1^2\boldsymbol{x}(3)^2 + b_2^2c_2^2\boldsymbol{x}(4)^2\right) \\
&+ \mathrm{cov}\left(2a_1^2c_1c_2\boldsymbol{x}(1)\boldsymbol{x}(2) + 2a_2^2c_1c_2\boldsymbol{x}(3)\boldsymbol{x}(4), 2b_1^2c_1c_2\boldsymbol{x}(1)\boldsymbol{x}(2) + 2b_2^2c_1c_2\boldsymbol{x}(3)\boldsymbol{x}(4)\right) \\
&= \left(\boldsymbol{x}(1)^2 + \boldsymbol{x}(3)^2\right)^2\mathrm{var}(c_1^2) + \left(\boldsymbol{x}(2)^2 + \boldsymbol{x}(4)^2\right)^2\mathrm{var}(c_2^2) + 4(\boldsymbol{x}(1)\boldsymbol{x}(2) + \boldsymbol{x}(3)\boldsymbol{x}(4))^2\mathrm{var}(c_1c_2) \\
&= \left(\boldsymbol{x}(1)^2 + \boldsymbol{x}(3)^2\right)^2\mathrm{var}(c_1^2) + \left(\boldsymbol{x}(2)^2 + \boldsymbol{x}(4)^2\right)^2\mathrm{var}(c_2^2) + 4(\boldsymbol{x}(1)\boldsymbol{x}(2) + \boldsymbol{x}(3)\boldsymbol{x}(4))^2 > 0,
\end{aligned}
\tag{52}
$$

and then Eq. (52) implies $\mathbb{E}[\boldsymbol{y}^2(1)\boldsymbol{y}^2(3)] \neq \mathbb{E}[\boldsymbol{y}^2(1)]\,\mathbb{E}[\boldsymbol{y}^2(3)]$.

Generally, for some $i \neq j$, $\mathbb{E}[\boldsymbol{y}^2(i)\boldsymbol{y}^2(j)] \neq \mathbb{E}[\boldsymbol{y}^2(i)]\mathbb{E}[\boldsymbol{y}^2(j)]$.