More Efficient Sampling for Tensor Decomposition

Osman Asif Malik*

Abstract

Recent papers have developed alternating least squares (ALS) methods for CP and tensor ring decomposition with a per-iteration cost which is sublinear in the number of input tensor entries for low-rank decomposition. However, the per-iteration cost of these methods still has an exponential dependence on the number of tensor modes. In this paper, we propose sampling-based ALS methods for the CP and tensor ring decompositions whose cost does not have this exponential dependence, thereby significantly improving on the previous state-of-the-art. We provide a detailed theoretical analysis and also apply the methods in a feature extraction experiment.

1 Introduction

Tensor decomposition has recently emerged as an important tool in machine learning and data mining (Papalexakis et al., 2016; Cichocki et al., 2016, 2017; Ji et al., 2019). Due to their multidimensional nature, tensors are inherently plagued by the curse of dimensionality. Indeed, simply storing an \( N \)-way tensor with each dimension equal to \( I \) requires \( I^N \) numbers. To make matters worse, tensors are fundamentally more difficult to decompose than matrices. Many tensor decompositions correspond to difficult non-convex optimization problems. In view of these difficulties, it may seem hopeless to construct efficient algorithms for decomposing large-scale high-dimensional tensors.

A popular approach for tensor decomposition is to formulate it as an optimization problem and solve it via alternating least squares (ALS). While ALS works well for smaller tensors, the per-iteration cost for an \( N \)-way tensor of size \( I \times \cdots \times I \) is \( \Omega(I^N) \) since each iteration requires solving a number of least squares problems where the data tensor features in the right hand side. To address this issue, several recent works have developed sampling-based ALS methods for the CP decomposition (Cheng et al., 2016; Larsen and Kolda, 2020) and tensor ring decomposition (Malik and Becker, 2021). When the target rank is small enough, they have a per-iteration cost which is sublinear in the number of input tensor entries while still retaining approximation guarantees for each least squares solve with high probability. However, the cost of these methods still has an exponential dependence on \( N \): \( \Omega(RN+1) \) for the CP decomposition and \( \Omega(R^{2N+2}) \) for the tensor ring decomposition, where \( R \) is the relevant notion of rank. Unlike matrix rank, both the CP and tensor ring ranks of a tensor can exceed the mode dimension \( I \), in which case the previous methods would no longer have sublinear per-iteration cost. This is not just an issue for tensors with many modes (i.e., large \( N \)): A popular preprocessing step when decomposing tensors is to increase the number of modes via reshaping and permutation operations which turns a few-mode tensor into one with many modes (Yuan et al., 2019b).

This begs the following question:

Can we construct ALS algorithms for tensor decomposition with a per-iteration cost which does not depend exponentially on \( N \) and which has guarantees for each least squares solve?

In this paper, we show that this is indeed possible for both the CP and tensor ring\(^1\) decompositions with high probability relative error guarantees. Like the previous works mentioned above, we also use approximate

---

*Applied Mathematics & Computational Research Division, Lawrence Berkeley National Laboratory, oamalik@lbl.gov

\(^1\)Our results are also relevant for the popular tensor train decomposition (Oseledets, 2010, 2011) since it is a special case of the tensor ring decomposition.
leverage score sampling. Unlike those previous works which use quite coarse approximations to the leverage scores, we are able to sample from a distribution which is much closer to the exact one. We do this by using ideas for fast leverage score estimation from Drineas et al. (2012) combined with the recently developed recursive sketch by Ahle et al. (2020). We also design sampling schemes for both the CP and tensor ring decompositions which allow us to avoid computing the whole sampling distribution which otherwise would cost $\Omega(I^{N-1})$. We provide a detailed theoretical analysis and also apply the method in a feature extraction experiment.

2 Related Work

CP Decomposition  
Cheng et al. (2016) propose SPALS, the first ALS algorithm for CP decomposition with a per-iteration cost sublinear in the number of input tensor entries. They use leverage scores sampling to speed up computation of the matricized-tensor-times-Khatri–Rao product, a key kernel which arises in the ALS algorithm for CP decomposition. Larsen and Kolda (2020) propose CP-ARLS-LEV which uses leverage score sampling to reduce the size of the least squares problems in the ALS algorithm for CP decomposition. In addition to several practical algorithmic improvements, their relative error guarantees improve on the weaker additive error guarantees provided by Cheng et al. (2016). Other papers that develop randomized algorithms for the CP decomposition include Wang et al. (2015), Battaglino et al. (2018), Yang et al. (2018) and Aggour et al. (2020).

Tensor Ring Decomposition  
Yuan et al. (2019a) develop a randomized method for the tensor ring decomposition which first compresses the input tensor by applying Gaussian sketches to each mode. The compressed tensor is then decomposed using standard deterministic decomposition algorithms. This decomposition can be combined with the sketches to get a decomposition of the original tensor. Ahmadi-Asl et al. (2020) develop several randomized variants of the deterministic TR-SVD algorithm by replacing the SVDs with their randomized counterpart. Malik and Becker (2021) propose TR-ALS-Sampled which is an ALS algorithm with a per-iteration cost sublinear in the number of input tensor entries. It uses leverage score sampling to reduce the size of the least squares problems in the standard ALS algorithm. Other works that develop randomized methods for tensor ring decomposition include Espig et al. (2012), Khoo et al. (2019).

Other Related Work  
Papers that develop randomized methods for other tensor decompositions include the works by Drineas and Mahoney (2007), Tsourakakis (2010), da Costa et al. (2016), Malik and Becker (2018), Sun et al. (2020), Minster et al. (2020) and Faurlbach et al. (2021) for the Tucker decomposition; Biagioni et al. (2015) and Malik and Becker (2020) for the tensor interpolative decomposition; Zhang et al. (2018) and Tarzanagh and Michailidis (2018) for t-product-based decompositions; and Huber et al. (2017) and Che and Wei (2019) for the tensor train decomposition. Papers that use skeleton approximation and other sampling-based techniques include those by Mahoney et al. (2008), Oseledets et al. (2008), Oseledets and Tyryshnikov (2010), Caiafa and Cichocki (2010) and Friedland et al. (2011).
where $\Sigma \in \mathbb{R}^{\text{rank}(A) \times \text{rank}(A)}$ and $U,V$ have rank($A$) columns. The $i$th canonical basis vector is denoted by $e_i$. We denote the indicator of a random event $A$ by $\text{Ind}(A)$, which is 1 if $A$ occurs and 0 otherwise. The notation $O$ means the same thing as $O$ except with log factors ignored. For indices $i_1 \in [I_1], \ldots, i_N \in [I_N]$, the notation $\Pi_{j=1}^{N} I_j \equiv 1 + \sum_{n=1}^{N} (i_n - 1) \prod_{j=1}^{n-1} I_j$ will be useful when working with unfolded tensors.

**Definition 1.** The classical mode-$n$ unfolding of $X$ is the matrix $X_{(n)} \in \mathbb{R}^{I_n \times \prod_{j \neq n} I_j}$ defined elementwise via

$$X_{(n)}(i_n, t_1 \cdots t_{n-1} t_{n+1} \cdots t_N) \equiv X(i_1, \ldots, i_N).$$

The mode-$n$ unfolding of $X$ is the matrix $X_{[n]} \in \mathbb{R}^{I_n \times \prod_{j \neq n} I_j}$ defined elementwise via

$$X_{[n]}(i_n, t_{n+1} \cdots t_N t_1 \cdots t_{n-1}) \equiv X(i_1, \ldots, i_N).$$

### 3.1 Tensor Decomposition

We first introduce the CP decomposition. Consider an $N$-way tensor $X \in \mathbb{R}^{I_1 \times \cdots \times I_N}$. A rank-$R$ CP decomposition of $X$ is of the form

$$X(i_1, \ldots, i_N) = \sum_{r=1}^{R} \prod_{j=1}^{N} A^{(j)}(i_j, r),$$

(1)

where each $A^{(j)} \in \mathbb{R}^{I_j \times R}$ is called a factor matrix. We use $\text{CP}(A^{(1)}, \ldots, A^{(N)})$ to denote the tensor in (1). The problem of computing a rank-$R$ CP decomposition of a data tensor $X$ can be formulated as

$$\arg\min_{A^{(1)}, \ldots, A^{(N)}} \| \text{CP}(A^{(1)}, \ldots, A^{(N)}) - X \|_F.$$  

(2)

Unfortunately, this problem is non-convex and difficult to solve exactly. ALS is the “workhorse” algorithm for solving this problem approximately (Kolda and Bader, 2009). With ALS, we consider the objective in (2), but only solve with respect to one of the factor matrices at a time while keeping the others fixed:

$$\arg\min_{A^{(n)}} \| \text{CP}(A^{(1)}, \ldots, A^{(N)}) - X \|_F.$$  

(3)

The problem in (3) can be rewritten as the linear least squares problem

$$\arg\min_{A^{(n)}} \| A^{[n]} A^{(n)}^T - X^{(n)} \|_F^2,$$  

(4)

where $A^{[n]} \in \mathbb{R}^{\prod_{j \neq n} I_j \times R}$ is a defined as

$$A^{[n]} \equiv A^{(n)} \circ \cdots \circ A^{(n+1)} \circ A^{(n-1)} \circ \cdots \circ A^{(1)}.$$  

(5)

By repeatedly updating each factor matrix one at a time via (4), we get the standard CP-ALS algorithm outlined in Algorithm 1. For further details on the CP decomposition, see Kolda and Bader (2009).

Next, we introduce the tensor ring decomposition. For $j \in [N]$, let $G^{(j)} \in \mathbb{R}^{R_{j-1} \times I_j \times R_j}$ be 3-way tensors with $R_0 = R_N$. A rank-$\langle R_1, \ldots, R_N \rangle$ tensor ring decomposition of $X$ is of the form

$$X(i_1, \ldots, i_N) = \sum_{r_1, \ldots, r_N} \prod_{n=1}^{N} G^{(n)}(r_{n-1}, i_n, r_n),$$  

(6)

where each $r_j$ in the sum goes from 1 to $R_n$ and $r_0 = r_N$, and each $G^{(j)}$ is called a core tensor. We use $\text{TR}(G^{(1)}, \ldots, G^{(N)})$ to denote the tensor in (6). Finding the best possible rank-$\langle R_1, \ldots, R_N \rangle$ tensor ring
Algorithm 1: CP-ALS

```
Input: \( X \in \mathbb{R}^{I_1 \times \cdots \times I_N} \), rank \( R \)
Output: Factor matrices \( A^{(1)}, \ldots, A^{(N)} \)
1 Initialize factor matrices \( A^{(2)}, \ldots, A^{(N)} \)
2 while termination criteria not met do
3     for \( n = 1, \ldots, N \) do
4         \( A^{(n)} = \arg \min_A \| A^{\neq n} A^\top - X^{(n)}_n \|_F \)
5 return \( A^{(1)}, \ldots, A^{(N)} \)
```

decomposition of a tensor \( X \) is difficult. With an ALS approach we can update a single core tensor at a time by solving the following problem:

\[
\arg \min_{G^{(n)}} \| \text{TR}(G^{(1)}, \ldots, G^{(N)}) - X \|_F.
\] (7)

To reformulate this problem into a linear least squares problem we will need the following definition.

**Definition 2.** By merging all cores except the \( n \)th, we get a subchain tensor \( G^{\neq n} \in \mathbb{R}^{R_n \times (\prod_{j \neq n} I_j) \times R_{n-1}} \) defined elementwise via

\[
G^{\neq n}(r_n, i_{n+1} \ldots i_N, i_n, \ldots i_{n-1}) \overset{\text{def}}{=} \sum_{r_1, \ldots, r_{n-2}, j} N \prod_{j=1}^{N} G^{(j)}(r_{j-1}, i_j, r_j).
\]

The problem in (7) can now be written as the linear least square problem

\[
G^{(n)} = \arg \min_{G} \| G^{\neq n} G^{(2)}_\top - X^{(n)}_n \|_F.
\] (8)

By repeatedly updating each core tensor one at a time via (8), we get the standard TR-ALS algorithm outlined in Algorithm 2. For further details on the TR decomposition, see Zhao et al. (2016).

Algorithm 2: TR-ALS

```
Input: \( X \in \mathbb{R}^{I_1 \times \cdots \times I_N} \), ranks \( R_1, \ldots, R_N \)
Output: Core tensors \( G^{(1)}, \ldots, G^{(N)} \)
1 Initialize core tensors \( G^{(2)}, \ldots, G^{(N)} \)
2 while termination criteria not met do
3     for \( n = 1, \ldots, N \) do
4         \( G^{(n)} = \arg \min_G \| G^{\neq n} G^{(2)}_\top - X^{(n)}_n \|_F \)
5 return \( G^{(1)}, \ldots, G^{(N)} \)
```

### 3.2 Recursive Sketching

Ahle et al. (2020) present two variants of their recursive sketch. The first one, which we will use, combines CountSketch and TensorSketch into a single sketch which can be applied efficiently to Kronecker structured vectors. CountSketch was first introduced by Charikar et al. (2004) and extended to the linear algebra setting by Clarkson and Woodruff (2017). Recall that a function \( h: [I] \to [J] \) is said to be \( k \)-wise independent if for any \( k \) distinct \( i_1, \ldots, i_k \in [I] \) the values \( h(i_1), \ldots, h(i_k) \) are independent random variables uniformly distributed in \([J]\) (Pagh, 2013).
Definition 3. Let \( h : [I] \rightarrow [J] \) and \( s : [I] \rightarrow \{-1, +1\} \) be 3- and 4-wise independent functions, respectively. The CountSketch matrix \( C \in \mathbb{R}^{J \times I} \) is defined elementwise via \( C(j, i) \defeq s(i) \cdot \text{Ind}\{h(i) = j\} \).

The TensorSketch was developed in a series of papers by Pagh (2013), Pham and Pagh (2013), Avron et al. (2014) and Diao et al. (2018).

Definition 4. Let \( h_1, h_2 : [I] \rightarrow [J] \) and \( s_1, s_2 : [I] \rightarrow \{-1, +1\} \) be 3- and 4-wise independent functions, respectively. Define \( h : [I] \times [I] \rightarrow [J] \) via

\[
h(i_1, i_2) \defeq (h_1(i_1) + h_2(i_2)) \mod J + 1.
\]

The degree-two TensorSketch matrix \( T \in \mathbb{R}^{J \times J^2} \) is defined elementwise via

\[
T(j, i_1i_2) \defeq s(i_1)s(i_2) \cdot \text{Ind}\{h(i_1, i_2) = j\}.
\]

We are now ready to describe the recursive sketch of Ahle et al. (2020). It is easiest to understand it if we consider its application to Kronecker structured vectors. Consider \( x = x_1 \otimes \cdots \otimes x_N \in \mathbb{R}^{I_1 \cdot \ldots \cdot I_N} \), where each \( x_j \in \mathbb{R}^{I_j}. \)

Suppose first that \( N = 2^q \) is a power of 2. The first step of the recursive sketch is to apply an independent CountSketch matrix \( C_n \in \mathbb{R}^{J \times I_n} \) to each \( x_n \):

\[
y_n^{(0)} \defeq C_n x_n \in \mathbb{R}^{J}, \quad n \in [N].
\]

The vectors \( y_n^{(0)} \) are then combined pairwise using independent degree-two TensorSketches \( T_n^{(1)} \in \mathbb{R}^{J \times J^2} \):

\[
y_n^{(1)} \defeq T_n^{(1)}(y_{2n-1}^{(0)} \otimes y_{2n}^{(0)}), \quad n \in [N/2] .
\]

This process is then repeated: At each step, pairs of length-J vectors are combined using independent TensorSketches of size \( J \times J^2 \). The \( m \)th step is

\[
y_n^{(m)} \defeq T_n^{(m)}(y_{2n-1}^{(m-1)} \otimes y_{2n}^{(m-1)}), \quad n \in [N/2^m] .
\]

When \( m = q \), we are left with a single vector \( y_1^{(q)} \in \mathbb{R}^{J} \). The mapping \( x \mapsto y_1^{(q)} \), which we denote by \( \Psi_{J, (I_j)}^{(q)} \), is the recursive sketch. If \( N \) is not a power of 2, we choose \( q \defeq \lceil \log_2 N \rceil \) and define the recursive sketch as \( \Psi_{J, (I_j)}^N \defeq x \mapsto \Psi_{J, (I_j)}^{2^q}(x \otimes e_1 \otimes (2^q - N)) \), where \( e_1 \) is of length \( I_\max \defeq \max_{j \in [N]} I_j \), and each \( I_j \defeq I_j \) for \( j \leq N \) and \( I_j \defeq I_\max \) if \( j > N \). We will refer to \( \Psi_{J, (I_j)}^N \) as a \((J, (I_j))^{N}\)-recursive sketch. The recursive sketch is in fact linear, and when \( N = 2^q \) we can write \( \Psi_{J, (I_j)}^N \) as a product of \( q + 1 \) matrices:

\[
\Psi_{J, (I_j)}^N = T^{(q)} T^{(q-1)} \cdots T^{(1)} C,
\]

where \( C \defeq \bigotimes_{j=1}^N C_j \) is a \( J^N \times \prod_{j=1}^N I_j \) matrix and \( T^{(m)} \defeq \bigotimes_{j=1}^{2^m-1} T_j^{(m)} \) is a \( J^{2^m} \times J^{2^m+1} \) matrix.

The recursive sketch is a subspace embedding with high probability.

Definition 5. A matrix \( \Psi \in \mathbb{R}^{J \times I} \) is called a \( \gamma \)-subspace embedding for a matrix \( A \in \mathbb{R}^{I \times R} \) if

\[
\| \Psi A x \|_2^2 - \| A x \|_2^2 \leq \gamma \| A x \|_2^2 \quad \text{for all} \ x \in \mathbb{R}^R.
\]

The recursive sketch has the remarkable feature that the embedding dimension required for subspace embedding guarantees does not depend exponentially on \( N \). See Theorem 1 in Ahle et al. (2020) or Theorem 16 in the supplement for a precise statement.

\(^2\)Ahle et al. (2020) consider the case when each \( x_n \) has the same length. We consider a slightly more general definition here since this allows us to work with tensors whose modes are of different size.
3.3 Leverage Score Sampling

Leverage score sampling is a popular technique for a variety of problems in numerical linear algebra. For an in-depth discussion, see Mahoney (2011) and Woodruff (2014).

Definition 6. Let \( A \in \mathbb{R}^{I \times R} \) and suppose \( U \in \mathbb{R}^{I \times \text{rank}(A)} \) contains the left singular vectors of \( A \). The \( i \)th leverage score of \( A \) is defined as \( \ell_i(A) \overset{\text{def}}{=} \|U(:,i)\|_2^2 \) for \( i \in [I] \).

Definition 7. Let \( q \in \mathbb{R}^I \) be a probability distribution and let \( f : [J] \rightarrow [I] \) be a random map such that each \( f(j) \) is independent and distributed according to \( q \). Define \( S \in \mathbb{R}^{J \times I} \) elementwise via

\[
S(j, i) \overset{\text{def}}{=} \text{Ind}\{f(j) = i\}/\sqrt{Jq(f(j))}.
\]

We call \( S \) a sampling matrix with parameters \((J, q)\), or \( S \sim D(J, q) \) for short. Let \( A \in \mathbb{R}^{I \times R} \) be nonzero and suppose \( \beta \in (0, 1] \). Define the distribution \( p \in \mathbb{R}^I \) via \( p(i) \overset{\text{def}}{=} \ell_i(A)/\text{rank}(A) \). We say that \( S \sim D(J, q) \) is a leverage score sampling matrix for \((A, \beta)\) if \( q(i) \geq \beta p(i) \) for all \( i \in [I] \).

For a least squares problem \( \min_{x} \|Ax - y\|_2 \) where \( A \in \mathbb{R}^{I \times R} \) has many more rows than columns, we can use sampling to reduce the size of the problem to \( \min_{x} \|SAx - Sy\|_2 \). We would ideally like to sample according to the distribution \( p \) in Definition 7, but this requires computing \( U \) in Definition 6 (e.g., via SVD or QR) which costs \( O(IR^2) \). This is the same cost as solving the full least squares problem and is therefore too expensive. However, as shown by Drineas et al. (2012), the leverage scores can be accurately estimated in less time. Theorem 8 is a variant of Lemma 9 by Drineas et al. (2012). They consider the case when \( \Psi \) is a fast Johnson–Lindenstrauss transform instead of a subspace embedding.

Theorem 8. Let \( A \in \mathbb{R}^{I \times R} \) where \( I > R \), \( \gamma \in (0, 1) \) and suppose \( \Psi A = U_1 \Sigma_1 V_1^\top \) is a compact SVD. Define

\[
\tilde{\ell}_i(A) \overset{\text{def}}{=} \|e_i^\top A V_1 \Sigma_1^{-1}\|_2^2.
\]

Suppose that \( \Psi \) is a \( \gamma \)-subspace embedding for \( A \). Then

\[
|\ell_i(A) - \tilde{\ell}_i(A)| \leq \frac{\gamma}{1 - \gamma} \ell_i(A) \quad \text{for all } i \in [I].
\]

A proof of Theorem 8 appears in Section 7.1 of the supplement.

4 Efficient Sampling for Tensor Decomposition

In this section we present our proposed sampling schemes for the CP and tensor ring decompositions of a tensor \( X \in \mathbb{R}^{I_1 \times \cdots \times I_N} \). We will refer to these methods as CP-ALS-ES and TR-ALS-ES, respectively, where “ES” is short for “Efficient Sampling.”

4.1 CP Decomposition

Each least squares solve on line 4 in Algorithm 1 involves all entries in \( X \). To reduce the size of this problem, we sample rows according to an approximate leverage score distribution computed as in Theorem 8 with \( \Psi \) chosen to be a recursive sketch. Theorem 9 shows that such a sampling approach yields relative error guarantees for the CP-ALS least squares problem. A proof is provided in Section 7.2 of the supplement.

Theorem 9. Let \( A^{\neq n} \) be defined as in (5). Define the vector \( v \overset{\text{def}}{=} [N, \cdots, n + 1, n - 1, \cdots, 1] \) and suppose \( \varepsilon, \delta \in (0, 1) \). Suppose the estimates \( \tilde{\ell}_i(A^{\neq n}) \) are computed as in Theorem 8, with \( \Psi \in \mathbb{R}^{I_1 \times \Pi_{j \neq 1} I_j} \) chosen to be a \((J_1, (I_{v(j)})_{j=1}^N^{N-1})\)-recursive sketch. Moreover, suppose \( S \in \mathbb{R}^{I_2 \times \Pi_{j \neq 2} I_j} \) is a sampling matrix with parameters \((J_2, q)\) where \( q(i) \propto \tilde{\ell}_i(A^{\neq n}) \). If

\[
J_1 \gtrsim NR^2/\delta,
\]

\[
J_2 \gtrsim R \max \left( \log(R/\delta), 1/(\varepsilon\delta) \right),
\]

then...
then \( \hat{A} \overset{\text{def}}{=} \arg \min_A \|SA^{\neq n}A^T - SX^T_{(n)}\|_F \) satisfies the following with probability at least \( 1 - \delta \):
\[
\|A^{\neq n}A^T - X^T_{(n)}\|_F \leq (1 + \varepsilon) \min_A \|A^{\neq n}A^T - X^T_{(n)}\|_F.
\]

The dependence on \( R \) in (12) is optimal in the sense that it cannot be improved when rows are sampled i.i.d. (Dereziński and Warmuth, 2018). It is a significant improvement over the current state-of-the-art sampling-based ALS method by Larsen and Kolda (2020) which requires \( O(R^{N-1}\log(max(R, I_n)/\delta)/\varepsilon^2) \) samples to achieve relative error guarantees. The method by Cheng et al. (2016) requires \( O(R^N\log(I_n/\delta)/\varepsilon^2) \) samples and only achieves weaker additive error guarantees.

In Sections 4.1.1 and 4.1.2 we discuss how to compute the approximate solution \( \hat{A} \) in Theorem 9 efficiently. In Section 4.1.3 we compare the complexity of our method to that of other CP decomposition methods.

4.1.1 Step 1: Computing \( \Psi A^{\neq n} \)

The columns of \( A^{\neq n} \) are Kronecker products, so applying the recursive sketch \( \Psi \) to \( A^{\neq n} \) efficiently is straightforward. Let \( q \overset{\text{def}}{=} \lceil \log_2(N - 1) \rceil \). First, independent CountSketches \( C_j \) with \( J_i \) rows and an appropriate number of columns are applied:
\[
Y_j^{(0)} \overset{\text{def}}{=} \begin{cases} C_j A^{(v(j))} & \text{if } 1 \leq n \leq N - 1, \\ C_j e_1 1_{1 \times R} & \text{if } N - 1 < n \leq 2^q, \end{cases}
\]
where \( v \) is defined as in Theorem 9 and \( 1_{1 \times R} \) is a length-\( R \) row vector of ones. Then, independent TensorSketches are applied recursively:
\[
Y_j^{(m)} = T_j^{(m)}(Y_{2j - 1}^{(m-1)} \odot Y_{2j}^{(m-1)}), \quad j \in [2^{q-m}],
\]
for \( m = 1, \ldots, q \), where each \( T_j^{(m)} \in \mathbb{R}^{J_1 \times J_2^m} \). The final output is \( Y_1^{(q)} = \Psi A^{\neq n} \).

4.1.2 Step 2: Drawing Samples Efficiently

Since a row index \( i \in \prod_{j \neq n} I_j \) of \( A^{\neq n} \) can be written as \( i = i_1 \cdots i_{n-1} i_{n+1} \cdots i_N \) where each \( i_j \in [I_j] \), we can sample an index \( i \in \prod_{j \neq n} I_j \) by sampling subindices \( i_j \in [I_j] \) for each \( j \neq n \). By sampling the subindices in sequence one after another we avoid computing all entries in \( q \) which otherwise would cost \( \Omega(\prod_{j \neq n} I_j) \). We use an abbreviated notation to denote the probability of drawing subsequences of indices. For example, \( \mathbb{P}(i_1) \) denotes the probability that the first index is \( i_1 \), and \( \mathbb{P}(i_j)_{j \leq m, j \neq n} \) denotes the probability that the first \( m \) indices (excluding the \( n \)th) are \( i_1, \ldots, i_{n-1}, i_{n+1}, \ldots, i_m \).

**Lemma 10.** Let \( \Psi A^{\neq n} = U_1 \Sigma_1 V_1^T \) be a compact SVD and define \( \Phi \overset{\text{def}}{=} V_1 \Sigma_1^{-1} (V_1 \Sigma_1^{-1})^T \). The normalization constant for the distribution \( q \) with \( q(i) \propto \ell_i(A^{\neq n}) \) is
\[
C \overset{\text{def}}{=} \sum_i \ell_i(A^{\neq n}) = \sum_{r, k} \Phi(r, k) \cdot \prod_{j \neq n} (A^{(j)}^T A^{(j)})(r, k).
\]
The marginal probability of drawing \( (i_j)_{j \leq m, j \neq n} \) is
\[
\mathbb{P}((i_j)_{j \leq m, j \neq n}) = \frac{1}{C} \sum_{r, k} \Phi(r, k) \left( \prod_{j \leq m, j \neq n} (A^{(j)}(i_j, r) A^{(j)}(i_j, k)) \left( \prod_{j > m, j \neq n} (A^{(j)}^T A^{(j)})(r, k) \right) \right).
\]
In (14) and (15), the summations are over \( i \in \prod_{j \neq n} I_j \) and \( r, k \in [R] \).

The proof of Lemma 10 is given in Section 7.3 of the supplement. We now describe the sampling procedure by first describing how to sample the first index \( i_1 \) (or \( i_2 \), if \( n = 1 \)), followed by all subsequent indices.
then

Moreover, suppose

We compute the conditional probability in (16) for all

Theorem 11. The least squares problem for TR-ALS on line 4 in Algorithm 2 also involves all entries in

Table 1: Comparison of leading order computational cost for various CP decomposition methods. We ignore

We compute the conditional probability in (16) for all

Once the $J_2$ samples in $[\prod_{j\neq n} I_j]$ have been drawn, the matrix $SA^{\neq n}$ can be computed without forming

4.1.3 Complexity Analysis

If $J_1$ and $J_2$ are chosen as in (11) and (12), and if we assume that $I_j = I$ for all $j \in [N]$ and ignore log

If $J_1$ and $J_2$ are chosen as in (11) and (12), and if we assume that $I_j = I$ for all $j \in [N]$ and ignore log

Table 1: Comparison of leading order computational cost for various CP decomposition methods. We ignore

4.2 Tensor Ring Decomposition

The least squares problem for TR-ALS on line 4 in Algorithm 2 also involves all entries in $X$. We use an

Theorem 11. Let $G^{\neq n}_{[2]}$ be the mode-2 unfolding of the subchain tensor $S^{\neq n}$ (see Definitions 1 and 2).

Define the vector $w \defeq [n-1, \ldots, 1, N, \ldots, n+1]$ and suppose $\varepsilon, \delta \in (0, 1)$. Suppose the estimates

Moreover, suppose $S \in \mathbb{R}^{J_2 \times \Pi_{j=1}^{N} I_j}$ is a sampling matrix with parameters $(J_2, q)$ where $q(i) \propto \hat{\ell}_i(G^{\neq n}_{[2]})$. If

then $G \defeq \arg \min_G \|SG^{\neq n}_{[2]}G^\top - SX^{\top}_{[n]}\|_F$ satisfies the following with probability at least $1 - \delta$:

$$
\|G^{\neq n}_{[2]}G^\top - X^{\top}_{[n]}\|_F \leq (1 + \varepsilon) \min_G \|G^{\neq n}_{[2]}G^\top - X^{\top}_{[n]}\|_F.
$$

8
Since $G_{[2]}^{\hat{\theta}_n}$ has $R_{n-1} R_n$ columns, the sample complexity in (18) has optimal rank dependence in the sense discussed in Section 4.1. This is a significant improvement over the current state-of-the-art sampling-based ALS method by Malik and Becker (2021) which requires $O((\prod_{j} R_j^2) \max \{ \log (R_{n-1} R_n/\delta), 1/\epsilon \delta \})$ samples to achieve relative error guarantees.

In Sections 4.2.1 and 4.2.2 we discuss how to compute the approximate solution $\tilde{G}$ in Theorem 11 efficiently. In Section 4.2.3 we compare the complexity of our method to that of other tensor ring methods.

4.2.1 Step 1: Computing $\Psi G_{[2]}^{\hat{\theta}_n}$

Although $G_{[2]}^{\hat{\theta}_n}$ has a more complicated structure than $A^\theta_n$, $\Psi$ can still be applied efficiently to $G_{[2]}^{\hat{\theta}_n}$. We describe a scheme for computing the column $\Psi G_{[2]}^{\hat{\theta}_n}(:,r_{n-1} r_n)$ below, and give a more detailed motivation in Section 7.5 of the supplement. Let $q \defeq \log_2(N-1)$. Define matrices $H^{(j)}$ for $j \in [2^q]$ as follows: Let $H^{(1)} \in \mathbb{R}^{I_{n-1} \times R_{n-2}}$ be a matrix with columns $H^{(1)}(:,k) \defeq G_{[2]}^{(n-1)}(:,r_{n-1} k)$ for $k \in [R_{n-2}]$. Let $H^{(j)} \defeq G_{[2]}^{(u(j))} \in \mathbb{R}^{I_{u(j)} \times R_{u(j)} R_{u(j)-1}}$ for $2 \leq j \leq N - 2$. Let $H^{(N-1)} \in \mathbb{R}^{I_{n+1} \times R_{n+1}}$ be a matrix with columns $H^{(N-1)}(:,k) \defeq G_{[2]}^{(n+1)}(:,k r_n)$ for $k \in [R_{n+1}]$. Let $H^{(j)} \defeq e_1 \in \mathbb{R}^{\max j \neq n I_j}$ be a column vector for $N \leq j \leq 2^q$. Next, define

$$
Y^{(0)}_j = C_j H^{(j)}, \quad j \in [2^q],
$$

$$
K^{(0)}_j = \begin{cases} R_{u(j)} & \text{if } 2 \leq j \leq N - 1, \\
1 & \text{if } j = 1 \text{ or } j \leq 2^q + 1. 
\end{cases}
$$

The TensorSketch matrices are then applied recursively as follows. For each $m = 1, \ldots, q$, compute

$$
Y^{(m)}_j(:,k_1 k_3) = \sum_{k_2 \in [K^{(m-1)}_j]} T^{(m)}_j (Y^{(m-1)}_{2j-1}(:,k_1 k_2) \otimes Y^{(m-1)}_{2j}(:,k_2 k_3))
$$

for each $k_1 \in [K^{(m-1)}_{2j-1}], k_3 \in [K^{(m-1)}_{2j+1}], j \in [2^q - m]$. For each $m = 1, \ldots, q$, also compute

$$
K^{(m)}_j = K^{(m-1)}_{2j-1}, \quad j \in [2^q - m + 1].
$$

We prove the following in Section 7.5 of the supplement.

**Lemma 12.** $Y^{(q)}_1$ satisfies $Y^{(q)}_1 = \Psi G_{[2]}^{\hat{\theta}_n}(:,r_{n-1} r_n)$.

The entire matrix $\Psi G_{[2]}^{\hat{\theta}_n}$ can be computed by repeating the steps above for each column $r_{n-1} r_n \in [R_{n-1} R_n]$.

4.2.2 Step 2: Drawing Samples Efficiently

The sampling approach for the tensor ring decomposition is similar to the approach for the CP decomposition which we described in Section 4.1.2.

**Lemma 13.** Let $\Psi G_{[2]}^{\hat{\theta}_n} = U_1 \Sigma_1 V_1^\top$ be a compact SVD and define $\Phi \defeq V_1 \Sigma_1^{-1} (V_1 \Sigma_1^{-1})^\top$. The normalization constant for the distribution $q$ with $q(i) \propto \hat{\ell}_i(G_{[2]}^{\hat{\theta}_n})$ is

$$
C \defeq \sum_i \hat{\ell}_i(G_{[2]}^{\hat{\theta}_n}) = \sum_{r_{1}, \ldots, r_N} \Phi (r_{n-1} r_n, k_{n-1} k_n) \prod_{j \neq n} (G^{(j)}_{[2]} G^{(j)\top}_{[2]} (r_j r_{j-1}, k_j k_{j-1})). \quad (22)
$$

9
The marginal probability of drawing \((i_j)_{j \leq m, j \neq n}\) is

\[
P((i_j)_{j \leq m, j \neq n}) = \frac{1}{C} \sum_{k_1, \ldots, k_N} \Phi(r_{n-1}r_n, k_{n-1}k_n) 
\cdot \left( \prod_{j \leq m} G_{[2]}^{(j)}(i_j, r_j r_{j-1}) G_{[2]}^{(j)}(i_j, k_j k_{j-1}) \right) 
\cdot \left( \prod_{j > m} (G_{[2]}^{(j)} \cdot G_{[2]}^{(j)})(r_j r_{j-1}, k_j k_{j-1}) \right). \tag{23}
\]

In (22) and (23), the summations are over \(i \in [\prod_{j \neq n} I_j]\) and \(r_j, k_j \in [R_j]\) for each \(j \in [N]\).

The proof of Lemma 13 is given in Section 7.6 of the supplement. The sampling procedure itself is the same as for the CP decomposition. The distribution \((P(i_1))_{i_1=1}^{I_1}\) is computed via (23) and an index \(i_1\) is sampled; if \(n = 1\), these computations are done for \(i_2\) instead of \(i_1\). All subsequent indices are then sampled conditionally on the previous indices. This is done by computing the conditional distribution in (16) by using (23). The expression in (23) can be computed efficiently despite the exponential number of terms in the summation; see Remark 19 in the supplement for details.

Once the \(J_2\) samples in \([\prod_{j \neq n} I_j]\) have been drawn, the matrix \(SG_{[2]}^{J_2}\) can be computed without forming \(G_{[2]}^{J_2}\). We describe this in detail in Remark 20 in the supplement. The matrix \(SX_{[2]}^\top\) can be computed by extracting only \(J_2\) rows from \(X_{[2]}^\top\).

### 4.2.3 Complexity Analysis

If \(J_1\) and \(J_2\) are chosen as in (17) and (18), and if we assume that \(R_j = R\) and \(I_j = I\) for all \(j \in [N]\) and ignore log factors, then the per-iteration complexity for our method TR-ALS-ES is \(O(N^3 R^3 / \delta + N^3 IR^8 / (\varepsilon \delta))\).

In Table 2, we compare this to the complexity of several other methods for tensor ring decomposition (see Section 2). rTR-ALS refers to the method by Yuan et al. (2019a) with each \(I\) compressed to \(K\) and with TR-ALS as the deterministic algorithm. TR-SVD-Rand refers to Algorithm 7 in Ahmadi-Asl et al. (2020). Our method is the only one that does not have an explicit exponential dependence on \(N\). See Section 8 in the supplement for a detailed complexity analysis.

| Method               | Complexity                           |
|----------------------|--------------------------------------|
| TR-ALS               | \#iter \cdot NI^N R^2                |
| rTR-ALS              | \(NI^N K + \#iter \cdot NK^N R^2\)  |
| TR-SVD               | \(I^{N+1} + N R^3\)                  |
| TR-SVD-Rand          | \(I^N R^2\)                         |
| TR-ALS-Sampled       | \#iter \cdot NI R^{2N+2} / (\varepsilon \delta) |
| **TR-ALS-ES (our)**  | \#iter \cdot N^3 R^8 (R + I / \varepsilon) / \delta |

Table 2: Comparison of leading order computational cost for various tensor ring decomposition methods. We ignore log factors and assume that \(R_j = R\) and \(I_j = I\) for all \(j \in [N]\). \#iter is the number of ALS iterations.

### 5 Experiments

The experiments are run in Matlab R2021a on a computer with an AMD Ryzen 7 5800X CPU and 32 GB of RAM. Our code is available at https://github.com/OsmanMalik/TD-ALS-ES. Additional details are in Section 9 of the supplement.

**Sampling Distribution Comparison** We first compare the sampling distributions used by our methods with those used by the previous state-of-the-art—CP-ARLS-LEV by Larsen and Kolda (2020) for the CP...
decomposition and TR-ALS-Sampled by Malik and Becker (2021) for the tensor ring decomposition—when solving the least squares problems in (3) and (7). We run standard CP-ALS and TR-ALS on a real data tensor to get realistic factor matrices and core tensors when defining the design matrices $A^{\neq n}$ and $G^{\neq n}$. We get the real data tensor $X \in \mathbb{R}^{16 \times \cdots \times 16}$ by reshaping a $4096 \times 4096$ gray scale image of a tabby cat into a 6-way tensor and then appropriately permuting the modes, a process called visual data tensorization (Yuan et al., 2019b). We then consider the least squares problems corresponding to an update of the 6th factor matrix or core tensor. As a performance measure, we compute the KL-divergence of the approximate distribution $q$ from the exact leverage score sampling distribution $p$ in Definition 7. Tables 3 and 4 report the results for different $J_1$ and ranks. The results show that our methods sample from a distribution much closer to the exact leverage score distribution when $J_1$ is as small as $J_1 = 1000$. See Figures 1–4 in the supplement for a graphical comparison.

| Method              | $R = 10$ | $R = 20$ |
|---------------------|----------|----------|
| CP-ARLS-LEV         | 0.2342   | 0.1853   |
| CP-ALS-ES ($J_1 = 1\times4$) | 0.0005   | 0.0006   |
| CP-ALS-ES ($J_1 = 1\times3$) | 0.0151   | 0.0070   |
| CP-ALS-ES ($J_1 = 1\times2$) | 0.1416   | 0.2173   |

Table 3: KL-divergence (lower is better) of the approximated sampling distribution from the exact one for a CP-ALS least squares problem (3) with target rank $R$.

| Method              | $R = 3$ | $R = 5$ |
|---------------------|----------|----------|
| TR-ALS-Sampled      | 0.3087   | 0.1279   |
| TR-ALS-ES ($J_1 = 1\times4$) | 0.0005   | 0.0007   |
| TR-ALS-ES ($J_1 = 1\times3$) | 0.0076   | 0.0070   |
| TR-ALS-ES ($J_1 = 1\times2$) | 0.1565   | 0.1831   |

Table 4: KL-divergence (lower is better) of the approximated sampling distribution from the exact one for a TR-ALS least squares problem (8) with target rank $(R, \ldots, R)$.

**Feature Extraction** Next, we run a benchmark feature extraction experiment on a downsampled variant of the COIL-100 image dataset (Nene et al., 1996) with a setup similar to that in Zhao et al. (2016) and Malik and Becker (2021). The data consists of 7200 color images of size $32 \times 32$ pixels, each belonging to one of 100 different classes. The data is arranged into a $32 \times 32 \times 3 \times 7200$ tensor which is decomposed using either a rank-25 CP decomposition or a rank-$(5, 5, 5)$ tensor ring decomposition. The mode-4 factor matrix or core tensor is then used as a feature matrix in a $k$-NN algorithm with $k = 1$ and 10-fold cross validation. We use $J_1 = J_2 = 1000$ in our methods. CP-ARLS-LEV and TR-ALS-Sampled both use 1000 samples as well. All methods are run for 20 iterations. Table 5 shows the decomposition time, decomposition error, and classification accuracy for our proposed methods compared to the deterministic baselines and previous state-of-the-art sampling-based ALS methods. Since this is a relatively small dataset, the deterministic CP-ALS method is faster than the two randomized CP methods. Our method is slightly slower than CP-ARLS-LEV. In view of the complexity analysis, this is not unexpected since we only expect our method to have an advantage in the regime of large $R$ and $N$. The tensor ring decomposition is more computationally demanding than the CP decomposition. Even on this small dataset, the deterministic TR-ALS method is slower than the two randomized methods. Our method takes longer to run than TR-ALS-Sampled, which is again not unexpected. All methods achieve good classification accuracy and similar decomposition errors.
Table 5: Run time, decomposition error and classification accuracy when using tensor decomposition for feature extraction.

| Method            | Time (s) | Err.  | Acc. (%) |
|-------------------|----------|-------|----------|
| CP-ALS            | 1.60     | 0.28  | 99.0     |
| CP-ARLS-LEV       | 4.23     | 0.29  | 98.1     |
| CP-ALS-ES (our)   | 5.37     | 0.30  | 98.7     |
| TR-ALS            | 92.31    | 0.29  | 99.5     |
| TR-ALS-Sampled    | 3.9      | 0.30  | 98.9     |
| TR-ALS-ES (our)   | 7.5      | 0.30  | 98.2     |

6 Discussion and Conclusion

We have shown that it is possible to construct ALS algorithms with guarantees for both the CP and tensor ring decompositions of an \( N \)-way tensor with a per-iteration cost which does not depend exponentially on \( N \). In the regime of high-dimensional tensors (i.e., with many modes), this is a substantial improvement over the previous state-of-the-art which had a per-iteration cost of \( \Omega(RN+1) \) and \( \Omega(R2N+2) \) for the CP and tensor ring decompositions, respectively, where \( R \) is the relevant notion of rank.

The high cost of CP-ARLS-LEV and TR-ALS-Sampled in Tables 1 and 2 come from the large number of samples that these methods require for theoretical guarantees. However, in practice many fewer samples will usually yield good results. In such cases, those methods are likely to be faster than the methods we propose in this paper.

The sampling formulas (15) and (23) are sums of products where each feature matrix or core tensor (except the \( n \)th) appears twice. This can exacerbate issues with ill-conditioned factor matrices or core tensors. A particular concern is that catastrophic cancellation can occur, which will prevent accurate computation of the probabilities. Addressing this issue is an interesting direction for future research.

Acknowledgments

This work was supported by the Laboratory Directed Research and Development Program of Lawrence Berkeley National Laboratory under U.S. Department of Energy Contract No. DE-AC02-05CH11231.

References

Aggour, K. S., Gittens, A., and Yener, B. (2020). Adaptive Sketching for Fast and Convergent Canonical Polyadic Decomposition. In International Conference on Machine Learning. PMLR.

Ahle, T. D., Kapralov, M., Knudsen, J. B., Pagh, R., Velingker, A., Woodruff, D. P., and Zandieh, A. (2020). Oblivious sketching of high-degree polynomial kernels. In Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 141–160. SIAM.

Ahmadi-Asl, S., Cichocki, A., Phan, A. H., Asante-Mensah, M. G., Mousavi, F., Oseledets, I., and Tanaka, T. (2020). Randomized algorithms for fast computation of low-rank tensor ring model. Machine Learning: Science and Technology.

Avron, H., Nguyen, H. L., and Woodruff, D. P. (2014). Subspace Embeddings for the Polynomial Kernel. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, pages 2258–2266, Cambridge, MA, USA. MIT Press.

Bader, B. W. and Kolda, T. G. (2006). Algorithm 862: MATLAB tensor classes for fast algorithm prototyping. ACM Transactions on Mathematical Software (TOMS), 32(4):635–653.
Bader, B. W., Kolda, T. G., and others (2021). MATLAB Tensor Toolbox, Version 3.2.1. Available online at https://www.tensortoolbox.org.

Battaglino, C., Ballard, G., and Kolda, T. G. (2018). A practical randomized CP tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 39(2):876–901.

Biagioni, D. J., Beylkin, D., and Beylkin, G. (2015). Randomized interpolative decomposition of separated representations. *Journal of Computational Physics*, 281(C):116–134.

Caiafa, C. F. and Cichocki, A. (2010). Generalizing the column–row matrix decomposition to multi-way arrays. *Linear Algebra and its Applications*, 433(3):557–573.

Charikar, M., Chen, K., and Farach-Colton, M. (2004). Finding Frequent Items in Data Streams. *Theoretical Computer Science*, 312(1):3–15.

Che, M. and Wei, Y. (2019). Randomized algorithms for the approximations of Tucker and the tensor train decompositions. *Advances in Computational Mathematics*, 45(1):395–428.

Cheng, D., Peng, R., Liu, Y., and Perros, I. (2016). SPALS: Fast alternating least squares via implicit leverage scores sampling. In *Advances in Neural Information Processing Systems*, pages 721–729.

Cichocki, A., Lee, N., Oseledets, I., Phan, A.-H., Zhao, Q., and Mandic, D. P. (2016). Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions. *Foundations and Trends® in Machine Learning*, 9(4-5):249–429.

Cichocki, A., Phan, A.-H., Zhao, Q., Lee, N., Oseledets, I., Sugiyama, M., and Mandic, D. P. (2017). Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives. *Foundations and Trends® in Machine Learning*, 9(6):431–673.

Clarkson, K. L. and Woodruff, D. P. (2017). Low-Rank Approximation and Regression in Input Sparsity Time. *Journal of the ACM*, 63(6):54:1–54:45.

da Costa, M. N., Lopes, R. R., and Romano, J. M. T. (2016). Randomized methods for higher-order subspace separation. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 215–219. IEEE.

Dereziński, M. and Warmuth, M. K. (2018). Reverse iterative volume sampling for linear regression. *The Journal of Machine Learning Research*, 19(1):853–891.

Diao, H., Song, Z., Sun, W., and Woodruff, D. (2018). Sketching for Kronecker Product Regression and P-splines. In *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*, pages 1299–1308.

Drineas, P., Kannan, R., and Mahoney, M. W. (2006a). Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157.

Drineas, P., Magdon-Ismail, M., Mahoney, M. W., and Woodruff, D. P. (2012). Fast approximation of matrix coherence and statistical leverage. *The Journal of Machine Learning Research*, 13(1):3475–3506.

Drineas, P. and Mahoney, M. W. (2007). A randomized algorithm for a tensor-based generalization of the singular value decomposition. *Linear algebra and its applications*, 420(2-3):553–571.

Drineas, P., Mahoney, M. W., and Muthukrishnan, S. (2006b). Sampling algorithms for l2 regression and applications. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, pages 1127–1136.

Drineas, P., Mahoney, M. W., Muthukrishnan, S., and Sarlós, T. (2011). Faster least squares approximation. *Numerische Mathematik*, 117(2):219–249.
Espig, M., Naraparaju, K. K., and Schneider, J. (2012). A note on tensor chain approximation. *Computing and Visualization in Science*, 15(6):331–344.

Fahrbach, M., Ghadiri, M., and Fu, T. (2021). Fast Low-Rank Tensor Decomposition by Ridge Leverage Score Sampling. *arXiv preprint arXiv:2107.10654*.

Friedland, S., Mehrmann, V., Miedlar, A., and Nkengla, M. (2011). Fast low rank approximations of matrices and tensors. *Electronic Journal of Linear Algebra*, 22:1031–1048.

Golub, G. H. and Van Loan, C. F. (2013). *Matrix Computations*. Johns Hopkins University Press, Baltimore, fourth edition.

Horn, R. A. and Johnson, C. R. (1994). *Topics in Matrix Analysis*. Cambridge University Press.

Huber, B., Schneider, R., and Wolf, S. (2017). A randomized tensor train singular value decomposition. In *Compressed Sensing and Its Applications*, pages 261–290. Springer.

Ji, Y., Wang, Q., Li, X., and Liu, J. (2019). A survey on tensor techniques and applications in machine learning. *IEEE Access*, 7:162950–162990.

Khoo, Y., Lu, J., and Ying, L. (2019). Efficient construction of tensor ring representations from sampling. *arXiv preprint arXiv:1711.00954*.

Kolda, T. G. and Bader, B. W. (2009). Tensor Decompositions and Applications. *SIAM Review*, 51(3):455–500.

Larsen, B. W. and Kolda, T. G. (2020). Practical Leverage-Based Sampling for Low-Rank Tensor Decomposition. *arXiv preprint arXiv:2006.16438*.

Mahoney, M. W. (2011). Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3(2):123–224.

Mahoney, M. W., Maggioni, M., and Drineas, P. (2008). Tensor-CUR decompositions for tensor-based data. *SIAM Journal on Matrix Analysis and Applications*, 30(3):957–987.

Malik, O. A. and Becker, S. (2018). Low-Rank Tucker Decomposition of Large Tensors Using TensorSketch. In *Advances in Neural Information Processing Systems*, pages 10096–10106.

Malik, O. A. and Becker, S. (2020). Fast randomized matrix and tensor interpolative decomposition using CountSketch. *Advances in Computational Mathematics*, 46(6):76.

Minster, R., Saibaba, A. K., and Kilmer, M. E. (2020). Randomized algorithms for low-rank tensor decompositions in the Tucker format. *SIAM Journal on Mathematics of Data Science*, 2(1):189–215.

Nene, S. A., Nayar, S. K., and Murase, H. (1996). Columbia Object Image Library (COIL-100). Technical Report CUCS-006-96, Columbia University.

Oseledets, I. and Tyrtyshnikov, E. (2010). TT-cross approximation for multidimensional arrays. *Linear Algebra and its Applications*, 432(1):70–88.

Oseledets, I. V. (2010). Approximation of $2^d \times 2^d$ matrices using tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2130–2145.

Oseledets, I. V. (2011). Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317.

Oseledets, I. V., Savostianov, D. V., and Tyrtyshnikov, E. E. (2008). Tucker dimensionality reduction of three-dimensional arrays in linear time. *SIAM Journal on Matrix Analysis and Applications*, 30(3):939–956.
Pagh, R. (2013). Compressed Matrix Multiplication. *ACM Transactions on Computation Theory, 5*(3):9:1–9:17.

Papalexakis, E. E., Faloutsos, C., and Sidiropoulos, N. D. (2016). Tensors for data mining and data fusion: Models, applications, and scalable algorithms. *ACM Transactions on Intelligent Systems and Technology (TIST), 8*(2):1–44.

Pham, N. and Pagh, R. (2013). Fast and Scalable Polynomial Kernels via Explicit Feature Maps. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’13*, pages 239–247, New York, NY, USA. ACM.

Sun, Y., Guo, Y., Luo, C., Tropp, J., and Udell, M. (2020). Low-rank Tucker approximation of a tensor from streaming data. *SIAM Journal on Mathematics of Data Science, 2*(4):1123–1150.

Tarzanagh, D. A. and Michailidis, G. (2018). Fast Randomized Algorithms for t-Product Based Tensor Operations and Decompositions with Applications to Imaging Data. *SIAM Journal on Imaging Sciences, 11*(4):2629–2664.

Tsourakakis, C. E. (2010). Mach: Fast randomized tensor decompositions. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, pages 689–700. SIAM.

Wang, Y., Tung, H.-Y., Smola, A. J., and Anandkumar, A. (2015). Fast and guaranteed tensor decomposition via sketching. In *Advances in Neural Information Processing Systems*, pages 991–999.

Woodruff, D. P. (2014). Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science, 10*(1-2):1–157.

Yang, B., Zamzam, A., and Sidiropoulos, N. D. (2018). ParaSketch: Parallel Tensor Factorization via Sketching. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 396–404. SIAM.

Yuan, L., Li, C., Cao, J., and Zhao, Q. (2019a). Randomized tensor ring decomposition and its application to large-scale data reconstruction. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2127–2131. IEEE.

Yuan, L., Zhao, Q., Gui, L., and Cao, J. (2019b). High-order tensor completion via gradient-based optimization under tensor train format. *Signal Processing: Image Communication, 73*:53–61.

Zhang, J., Saibaba, A. K., Kilmer, M. E., and Aeron, S. (2018). A randomized tensor singular value decomposition based on the t-product. *Numerical Linear Algebra with Applications, 25*:e2179.

Zhao, Q., Zhou, G., Xie, S., Zhang, L., and Cichocki, A. (2016). Tensor ring decomposition. *arXiv preprint arXiv:1606.05535.*
7 Missing Proofs

7.1 Proof of Theorem 8

We first state and prove Lemma 14. It is similar to Lemma 5 in Drineas et al. (2012) and Lemma 4.1 in Drineas et al. (2006b) which consider the case when \( \Psi \) is a fast Johnson–Lindenstrauss transform and a sampling matrix, respectively, instead of a subspace embedding.

**Lemma 14.** Consider a matrix \( A \in \mathbb{R}^{I \times R} \) where \( I > R \). Let \( A = U \Sigma V^T \) be a compact SVD of \( A \). Suppose \( \Psi \) is a \( \gamma \)-subspace embedding for \( A \) with \( \gamma \in (0, 1) \), and let \( \Psi U = Q \Lambda W^T \) be a compact SVD. Then, the following hold:

1. \( \text{rank}(\Psi A) = \text{rank}(\Psi U) = \text{rank}(A) \),
2. \( \|I - \Lambda^{-2}\|_2 \leq \gamma/(1 - \gamma) \),
3. \( (\Psi A)^\dagger = V \Sigma^{-1} (\Psi U)^\dagger \).

**Proof.** The proof follows similar arguments as those used in the proof of Lemma 4.1 in Drineas et al. (2006b). Since \( \Psi \) is a \( \gamma \)-subspace embedding for \( A \), we have

\[
(1 - \gamma)\|\Sigma V^T x\|_2^2 \leq \|\Psi U \Sigma V^T x\|_2^2 \leq (1 + \gamma)\|\Sigma V^T x\|_2^2 \quad \text{for all} \quad x \in \mathbb{R}^R.
\]

Let \( r \overset{\text{def}}{=} \text{rank}(A) \). Since \( \Sigma V^T \in \mathbb{R}^{r \times R} \) is full rank, and using unitary invariance of the spectral norm, it follows that

\[
(1 - \gamma)\|y\|_2^2 \leq \|\Psi U y\|_2^2 \leq (1 + \gamma)\|y\|_2^2 \quad \text{for all} \quad y \in \mathbb{R}^r.
\]

Using Theorem 8.6.1 in Golub and Van Loan (2013), this in turn implies that

\[
1 - \gamma \leq \sigma_i^2(\Psi U) \leq 1 + \gamma \quad \text{for all} \quad i \in [r].
\]

(24)

Consequently, \( \text{rank}(\Psi U) = r = \text{rank}(A) \). Moreover, since \( \text{rank}(\Psi A) = \text{rank}(\Psi U \Sigma V^T) \) and \( \Sigma V^T \) is full rank, it follows that \( \text{rank}(\Psi A) = \text{rank}(\Psi U) \). This completes the proof of (i).

Next, note that

\[
\|I - \Lambda^{-2}\|_2 = \max_{i \in [r]} \left| 1 - \frac{1}{\sigma_i^2(\Psi U)} \right| = \max_{i \in [r]} \left| \sigma_i^2(\Psi U) - 1 \right| \leq \frac{\gamma}{1 - \gamma},
\]

where the inequality follows from the bound in (24). This completes the proof of (ii).

We may write

\[
(\Psi A)^\dagger = (Q \Lambda W^T \Sigma V^T)^\dagger = V (\Lambda W^T \Sigma)^\dagger Q^T
\]

(25)

where the second inequality follows since \( Q \) and \( V \) have orthonormal columns. Since \( \text{rank}(\Psi U) = \text{rank}(A) \) due to (i), the matrix \( \Lambda W^T \in \mathbb{R}^{r \times r} \) is invertible, and therefore \( \Lambda W^T \Sigma \in \mathbb{R}^{r \times r} \) is invertible, and hence

\[
(\Lambda W^T \Sigma)^\dagger = \Sigma^{-1} W A^{-1}.
\]

(26)

Consequently, combining (25) and (26) we have

\[
(\Psi A)^\dagger = V \Sigma^{-1} W A^{-1} Q^T = V \Sigma^{-1} (\Psi U)^\dagger.
\]

This completes the proof of (iii).
We are now ready to prove the statement in Theorem 8.

**Proof of Theorem 8.** Our proof is similar to the proof of Lemma 9 in Drineas et al. (2012). Let \( \mathbf{A} = \mathbf{U} \Sigma \mathbf{V}^T \) be a compact SVD, \( r \overset{\text{def}}{=} \text{rank}(\mathbf{A}) \) and suppose \( i \in [I] \). From Definition 6, we have
\[
\ell_i(\mathbf{A}) = \| U(i, :) \|_2^2 = e_i^T \mathbf{U} \mathbf{U}^T e_i.
\]
Moreover,
\[
\ell_i(\mathbf{A}) = \| e_i^T \mathbf{A} \mathbf{V} \Sigma_i^{-1} \mathbf{U}^T \|_2^2 = \| e_i^T \mathbf{A} (\Psi \mathbf{A})^\dagger \|_2^2 = \| e_i^T \mathbf{U} (\Psi \mathbf{U})^\dagger (\Psi \mathbf{U})^T \mathbf{U}^T e_i, \tag{27}
\]
where the first equality follows from the definition of \( \ell_i(\mathbf{A}) \) in (10) and the unitary invariance of the spectral norm, and the third equality follows from Lemma 14 (iii). From (27) and (28), we have
\[
| \ell_i(\mathbf{A}) - \tilde{\ell}_i(\mathbf{A}) | = | e_i^T \mathbf{U} (I - (\Psi \mathbf{U})^\dagger (\Psi \mathbf{U})^T) \mathbf{U}^T e_i |
\leq \| e_i^T \mathbf{U} \|_2 \cdot \| (I - (\Psi \mathbf{U})^\dagger (\Psi \mathbf{U})^T) \mathbf{U}^T e_i \|_2
\leq \| e_i^T \mathbf{U} \|_2 \cdot \| I - (\Psi \mathbf{U})^\dagger (\Psi \mathbf{U})^T \|_2 \cdot \| \mathbf{U}^T e_i \|_2
= \| I - (\Psi \mathbf{U})^\dagger (\Psi \mathbf{U})^T \|_2 \cdot \ell_i(\mathbf{A}), \tag{29}
\]
where the first inequality follows from Cauchy–Schwarz inequality, and the second inequality follows from the definition of the matrix spectral norm. Let \( \Psi \mathbf{U} = \mathbf{Q} \mathbf{A} \mathbf{W}^T \) be a compact SVD. It follows that
\[
\| I - (\Psi \mathbf{U})^\dagger (\Psi \mathbf{U})^T \|_2 = \| I - \mathbf{W} \Lambda^{-2} \mathbf{W}^T \|_2. \tag{30}
\]
From Lemma 14 (i), it follows that \( \mathbf{W} \) is \( r \times r \), hence \( \mathbf{W} \mathbf{W}^T = \mathbf{I} \). Consequently, and using unitary invariance of the spectral norm,
\[
\| I - \mathbf{W} \Lambda^{-2} \mathbf{W}^T \|_2 = \| I - \Lambda^{-2} \|_2. \tag{31}
\]
Combining (29), (30) and (31), we get
\[
| \ell_i(\mathbf{A}) - \tilde{\ell}_i(\mathbf{A}) | \leq \| I - \Lambda^{-2} \|_2 \cdot \ell_i(\mathbf{A}) \leq \frac{\gamma}{1 - \gamma} \ell_i(\mathbf{A}),
\]
where the last inequality follows from Lemma 14 (ii). This completes the proof. \( \square \)

### 7.2 Proof of Theorem 9

We first state some results that we will need for this proof. Lemma 15 follows from Lemma 4.2.10 in Horn and Johnson (1994).

**Lemma 15.** For matrices \( \mathbf{M}_1, \ldots, \mathbf{M}_n \) and \( \mathbf{N}_1, \ldots, \mathbf{N}_n \) of appropriate sizes,
\[
(\mathbf{M}_1 \otimes \cdots \otimes \mathbf{M}_n) \cdot (\mathbf{N}_1 \otimes \cdots \otimes \mathbf{N}_n) = (\mathbf{M}_1 \mathbf{N}_1) \otimes \cdots \otimes (\mathbf{M}_n \mathbf{N}_n).
\]

Theorem 16 follows directly from Theorem 1 in Ahle et al. (2020) and its proof.\(^1\)

**Theorem 16.** Let \( \mathbf{A} \in \mathbb{R}^{I \times N} \). Let \( \Psi \in \mathbb{R}^{J \times I \times N} \) be the \( (J, (I_j)_{j=1}^N) \)-recursive sketch described in Section 3.2. If \( J \gtrsim N R^2 / (\gamma^2 \delta) \), then \( \Psi \) is a \( \gamma \)-subspace embedding for \( \mathbf{A} \) with probability at least \( 1 - \delta \).

It is easy to generalize Theorem 16 to the setting when \( \Psi \) is a \( (J, (I_j)_{j=1}^N) \)-recursive sketch where the \( I_j \) are not necessarily all equal.

---

\(^1\)Since we are not considering regularized least squares problems, the statistical dimension \( s_\lambda \) in Ahle et al. (2020) just becomes equal to the number of columns of \( \mathbf{A} \), which is \( R \) in our case. The statement of Theorem 1 in Ahle et al. (2020) uses \( \delta = 1/10 \), but the statement for general \( \delta \) is easy to infer from their proof of the theorem.
Corollary 17. Let \( A \in \mathbb{R}^{N_1 \times I_j} \). Let \( \Psi \in \mathbb{R}^{J \times \Pi_{j=1}^N I_j} \) be the \((J, (I_j)_{j=1}^N)\)-recursive sketch described in Section 3.2. If \( J \geq N R^2/(\gamma^2 \delta) \), then \( \Psi \) is a \( \gamma \)-subspace embedding for \( A \) with probability at least \( 1 - \delta \).

Proof. Let \( q \overset{\text{def}}{=} \lceil \log_2(N) \rceil \), \( I_{\max} \overset{\text{def}}{=} \max_{j \in [N]} I_j \), and \( \tilde{I}_j \overset{\text{def}}{=} I_j \) for \( j \leq N \) and \( \tilde{I}_j \overset{\text{def}}{=} I_{\max} \) for \( j > N \). Let \( 1_{1 \times R} \) denote a length-\( R \) row vector of all ones. From the definition of the recursive sketch in Section 3.2 and the factorization in (9), we have

\[
\Psi A = \Psi_{\tilde{J}, \tilde{I}_{\max}} (A \odot (e_1^{\otimes (2^q - N)} 1_{1 \times R}))
\]

where the last equality follows from Lemma 15. From the definition of matrix multiplication, we have

\[
\Psi A = \Psi_{\tilde{J}, \tilde{I}_{\max}} (A \odot (e_1^{\otimes (2^q - N)} 1_{1 \times R}))
\]

Due to (34), it follows that \( \hat{C}_j(i, j) = C_j(i, j) \) when \( i \in I_j \), and consequently

\[
\left( \bigotimes_{j=1}^{N} \hat{C}_j \right)(\cdot, \tilde{I}_N \cdots \tilde{I}_1) = \left( \bigotimes_{j=1}^{N} \hat{C}_j \right)(\cdot, \tilde{I}_N \cdots \tilde{I}_1) = \left( \bigotimes_{j=1}^{N} C_j \right)(\cdot, \tilde{I}_N \cdots \tilde{I}_1) \quad \text{for all } (i_1, \ldots, i_N) \in \mathcal{I}. \tag{37}
\]

Using (33) and (37), we can simplify (36) to

\[
\left( \bigotimes_{j=1}^{N} \hat{C}_j \right) \hat{A} = \sum_{(i_1, \ldots, i_N) \in \mathcal{I}} \left( \bigotimes_{j=1}^{N} C_j \right)(\cdot, \tilde{I}_N \cdots \tilde{I}_1) \hat{A}(\tilde{I}_N \cdots \tilde{I}_1, \cdot) = \left( \bigotimes_{j=1}^{N} C_j \right) A. \tag{38}
\]
Similarly, since the first column of each $\hat{C}_j$ and $C_j$ are the same,

$$
\left( \bigotimes_{j=N+1}^{2^q} \hat{C}_j e_1 \right) 1_{1 \times R} = \left( \bigotimes_{j=N+1}^{2^q} C_j e_1 \right) 1_{1 \times R}.
$$

Equations (32), (35), (38) and (39) together now imply that

$$
\Psi A = \hat{\Psi} \hat{A}.
$$

Moreover, it follows immediately from (33) that

$$
\|Ax\|_2 = \|\hat{A}x\|_2 \text{ for all } x \in \mathbb{R}.
$$

Theorem 16 implies that

$$
\mathbb{P}(\|\hat{\Psi} \hat{A}x\|_2^2 - \|\hat{A}x\|_2^2 \leq \gamma \|\hat{A}x\|_2^2 \text{ for all } x \in \mathbb{R}^R) \geq 1 - \delta.
$$

Due to (40) and (41), this implies that

$$
\mathbb{P}(\|\Psi Ax\|_2^2 - \|Ax\|_2^2 \leq \gamma \|Ax\|_2^2 \text{ for all } x \in \mathbb{R}^R) \geq 1 - \delta,
$$

which is what we wanted to show.

Theorem 18 is a well-known result. Since it has appeared in slightly different variants in the literature, we provide a proof sketch just to give the reader some idea of how to derive it.

**Theorem 18.** Let $A \in \mathbb{R}^{I \times R}$ be a matrix, and suppose $S \sim D(J, q)$ is a leverage score sampling matrix for $(A, \beta)$ where $\beta \in (0, 1]$, and that $\varepsilon, \delta \in (0, 1)$. Moreover, define $\text{OPT} \overset{\text{def}}{=} \min_X \|AX - Y\|_F$ and $\bar{X} \overset{\text{def}}{=} \text{arg min}_X \|SAX - SY\|_F$. If

$$
J > \frac{4R}{\beta} \max \left( \frac{4}{3(\sqrt{2} - 1)^2 \ln \left( \frac{4R}{\delta} \right)}, \frac{1}{\varepsilon \delta} \right),
$$

then the following holds with probability at least $1 - \delta$:

$$
\|AX - Y\|_F \leq (1 + \varepsilon)\text{OPT}.
$$

**Proof sketch.** Let $U \in \mathbb{R}^{I \times \text{rank}(A)}$ contain the left singular vectors of $A$, and define $Y^\perp \overset{\text{def}}{=} (I - UU^\top)Y$. According to a matrix version\(^2\) of Lemma 1 by Drineas et al. (2011), the statement in (43) holds if both

$$
\sigma_{\text{min}}^2(SU) \geq \frac{1}{\sqrt{2}}
$$

and

$$
\|U^\top S^\top SY^\perp\|_F^2 \leq \frac{\varepsilon}{2} \text{OPT}^2.
$$

To complete the proof, it is therefore sufficient to show that $S$ satisfies both (44) and (45) with probability at least $1 - \delta$. Using Lemma S2 in Malik and Becker (2021), which is the same as Theorem 2.11 in Woodruff (2014) but with a slightly smaller constant, one can show that the condition (44) is satisfied with probability at least $1 - \delta/2$ if

$$
J > \frac{16}{3(\sqrt{2} - 1)^2} \frac{R}{\beta} \ln \left( \frac{4R}{\delta} \right).
$$

\(^2\)See Lemma S1 in Malik and Becker (2021).
Next, using Lemma 8 in Drineas et al. (2006a), it follows that
\[ \mathbb{E}\|U^T S^T Y Y \|^2 \leq \frac{1}{J \beta} R \cdot \text{OPT}^2. \]

Markov’s inequality together with the assumption
\[ J > \frac{4R}{\beta \epsilon \delta} \]
then implies that
\[ \mathbb{P}\left( \|U^T S^T Y Y \|_F^2 > \frac{\varepsilon}{2} \text{OPT}^2 \right) \leq \frac{2R}{J \beta \delta} < \frac{\delta}{2}. \]

If (42) is satisfied, then both (46) and (47) are satisfied, and consequently (44) and (45) are both true with probability at least \( 1 - \delta \). \( \square \)

We are now ready to prove the statement in Theorem 9.

**Proof of Theorem 9.** Let \( E_1 \) denote the event that \( \Psi \) is a 1/3-subspace embedding for \( A^\# n \). Following the notation used in Theorem 8, let \( \Psi A^\# n = U_1 \Sigma_1 V_1^T \) be a compact SVD. Let \( E_2 \) denote the event that (13) is true.

According to Corollary 17, we can guarantee that \( \mathbb{P}(E_1) \geq 1 - \delta/2 \) if we choose \( J_1 \) as in (11). With \( \gamma = 1/3 \) in Theorem 8, the estimates \( \hat{\ell}_i(A^\# n) \) satisfy
\[ \frac{1}{2} \hat{\ell}_i(A^\# n) \leq \bar{\ell}_i(A^\# n) \leq \frac{3}{2} \ell_i(A^\# n). \]

Consequently,
\[ \sum_{i = 1}^{\Pi_{j \neq n} l_i} \hat{\ell}_i(A^\# n) \leq \frac{3}{2} \sum_{i = 1}^{\Pi_{j \neq n} l_i} \ell_i(A^\# n) = \frac{3}{2} \text{rank}(A^\# n). \]

Therefore, since \( q(i) \propto \hat{\ell}_i(A^\# n) \), it follows by combining (48) and (49) that
\[ q(i) = \frac{\hat{\ell}_i(A^\# n)}{\sum_{i = 1}^{\Pi_{j \neq n} l_i} \hat{\ell}_i(A^\# n)} \geq \frac{1}{3} \frac{\ell_i(A^\# n)}{\text{rank}(A^\# n)}. \]

In view of Definition 7, Theorem 8 therefore implies that \( S \sim D(J_2, q) \) is a leverage score sampling matrix for \( (A^\# n, 1/3) \) if the event \( E_1 \) is true. From Theorem 18, it then follows that \( \mathbb{P}(E_2 \mid E_1) \geq 1 - \delta/2 \) if \( J_2 \) is chosen as in (12). With the choices of \( J_1 \) and \( J_2 \) above we now have
\[ \mathbb{P}(E_2) \geq \mathbb{P}(E_1, E_2) = \mathbb{P}(E_1) \mathbb{P}(E_2 \mid E_1) \geq (1 - \delta/2)^2 \geq 1 - \delta \]
which is what we wanted to show. \( \square \)

### 7.3 Proof of Lemma 10

Recall that \( \Phi \overset{\text{def}}{=} V_1 \Sigma_1^{-1}(V_1 \Sigma_1^{-1})^T \), where \( \Psi A^\# n = U_1 \Sigma_1 V_1^T \) is a compact SVD. From (10) we have
\[ \tilde{\ell}_i(A^\# n) = e_i^T A^\# n \Phi A^\# n^T e_i = (A^\# n \Phi A^\# n^T)(i, i) = \sum_{r, k} \Phi(r, k) \cdot \prod_{j \neq n} A(j)(i, r) A(j)(i, k), \]
where the last equality follows from the definition of \( A^\# n \) in (5), and \( i = \hat{\ell}_1 \cdots \hat{\ell}_{n-1} \hat{\ell}_{n+1} \cdots \hat{\ell}_N \). Using (51), we can compute the normalization constant \( C \) as
\[ C \overset{\text{def}}{=} \sum_i \tilde{\ell}_i(A^\# n) = \sum_{r, k} \Phi(r, k) \cdot \prod_{j \neq n} \sum_{i_j} A(j)(i_j, r) A(j)(i_j, k) = \sum_{r, k} \Phi(r, k) \cdot \prod_{j \neq n} (A^{(j)T} A^{(j)})(r, k), \]

20
The strategy of this proof is similar to that for the proof of Theorem 9 given in Section 7.2. Let 
which proves (14).

To make notation a bit less cumbersome, we will use the abbreviated notation

\[ \sum_{\{i_j\}_{j>m,j\neq n}} \] to denote  \( \left\{ \sum_{i_{m+1}} \cdots \sum_{i_{n-1}} \sum_{i_n} \cdots \sum_{i_N} \right\} \) if \( n > m \), otherwise.

Similar abbreviated notation will also be used later on for other indices. We can again use (51) to compute the marginal probabilities of drawing \((i_j)_{j \leq m,j \neq n}\) as

\[
\mathbb{P}((i_j)_{j \leq m,j \neq n}) = \frac{1}{C} \sum_{\{i_j\}_{j>m,j\neq n}} \tilde{\ell}_i(A^\tilde{n}) \\
= \frac{1}{C} \sum_{\{i_j\}_{j>m,j\neq n}} \left( \sum_{r,k} \Phi(r,k) \prod_{j\neq n} A^{(j)}(i_j,r)A^{(j)}(i_j,k) \right) \\
= \frac{1}{C} \sum_{r,k} \Phi(r,k) \left( \prod_{j \leq m} A^{(j)}(i_j,r)A^{(j)}(i_j,k) \right) \left( \prod_{j > m} A^{(j)^\top}A^{(j)}(r,k) \right),
\]

which proves (15).

### 7.4 Proof of Theorem 11

The strategy of this proof is similar to that for the proof of Theorem 9 given in Section 7.2. Let \( E_1 \) denote the event that \( \Psi \) is a 1/3-subspace embedding for \( G_{[2]}^{\neq n} \). Following the notation used in Theorem 8, let \( \Psi G_{[2]}^{\neq n} = U_1 \Sigma_1 V_1^\top \) be a compact SVD. Let \( E_2 \) denote the event that (19) is true.

The matrix \( G_{[2]}^{\neq n} \) is of size \( \prod_{j \neq n} I_j \times R_{n-1}R_n \). According to Corollary 17, we can therefore guarantee that \( \mathbb{P}(E_1) \geq 1 - \delta/2 \) if we choose \( J_1 \) as in (17). Following the same line of reasoning as in the proof of Theorem 9, we can show that the choice \( \gamma = 1/3 \) in Theorem 8 combined with the fact \( q(i) \propto \tilde{\ell}_i(G_{[2]}^{\neq n}) \) implies that

\[
q(i) = \frac{\tilde{\ell}_i(G_{[2]}^{\neq n})}{\sum_{i=1}^{\prod_{i \neq n} I_i} \tilde{\ell}_i(G_{[2]}^{\neq n})} \geq \frac{1}{3 \text{rank}(G_{[2]}^{\neq n})}.
\]

In view of Definition 7, Theorem 8 therefore implies that \( S \sim \mathcal{D}(J_2,q) \) is a leverage score sampling matrix for \((G_{[2]}^{\neq n},1/3)\) if the event \( E_1 \) is true. From Theorem 18, it then follows that \( \mathbb{P}(E_2 \mid E_1) \geq 1 - \delta/2 \) if \( J_2 \) is chosen as in (18). With the choices of \( J_1 \) and \( J_2 \) above and the formula (50), we have that \( \mathbb{P}(E_2) \geq 1 - \delta \), which is what we wanted to show.

### 7.5 Proof of Lemma 12

It follows directly from Definitions 1 and 2 that

\[
G_{[2]}^{\neq n}(i_{n+1} \cdots i_{N+1} \cdots i_{n-1} R_{n-1} R_n) = \sum_{\{r_j\}_{j \neq n-1,n}} \prod_{j=1}^{N-1} G_{[2]}^{\neq n}(i_{w(j)}, R_{w(j)} R_{w(j)-1}), \quad (52)
\]

and therefore the columns of \( G_{[2]}^{\neq n} \) can be written as

\[
G_{[2]}^{\neq n}(\cdot, R_{n-1} R_n) = \sum_{\{r_j\}_{j \neq n-1,n}} \bigotimes_{j=1}^{N-1} G_{[2]}^{\neq n}(\cdot, R_{w(j)} R_{w(j)-1}).
\]
Let $q \overset{\text{def}}{=} \lceil \log_2 (N - 1) \rceil$. Using the definition of the recursive sketch in Section 3.2 and the factorization in (9), we have

$$
\Psi G^{[2]}_{[2]} (:, r_{n-1} T_n) = T^{(q)} T^{(q-1)} \cdots T^{(1)} C \sum_{\{r_j\}_{j=1}^{n-1}} \left( \bigotimes_{j=1}^{N-1} G_{[2]}^{(w(j))} (:, r_{w(j)^T w(j)-1}) \right) \otimes e_1^{\otimes (2^n - (N-1))}. \quad (53)
$$

The notation in the equation above is quite cumbersome. In particular, the ordering of the matrices $G_{[2]}^{(j)}$ in the Kronecker product is somewhat awkward. To alleviate the issue somewhat, we define $H^{(j)}$ for $j \in [2^n]$ as we did in Section 4.2.1:

- Let $H^{(1)} \in \mathbb{R}^{I_{n-1} \times R_n-2}$ be a matrix with columns $H^{(1)}(:, k) \overset{\text{def}}{=} G_{[2]}^{(n-1)} (:, r_{n-1} k)$ for $k \in [R_n-2]$.
- Let $H^{(j)} \overset{\text{def}}{=} G_{[2]}^{(w(j))} \in \mathbb{R}^{I_{w(j)} \times R_{w(j)} R_{w(j)-1}}$ for $2 \leq j \leq N - 2$.
- Let $H^{(N-1)} \in \mathbb{R}^{I_{n+1} \times R_{n+1}}$ be a matrix with columns $H^{(N-1)}(:, k) \overset{\text{def}}{=} G_{[2]}^{(n+1)} (:, r_{n} k)$ for $k \in [R_{n+1}]$.
- Let $H^{(j)} \overset{\text{def}}{=} e_1 \in \mathbb{R}^{max \mathbb{j} \neq n I_j}$ be a column vector for $N \leq j \leq 2^n$.

Moreover, we also define the numbers $K^{(0)}_j$ for $j \in [2^n + 1]$ as in Section 4.2.1:

$$
K^{(0)}_j \overset{\text{def}}{=} \begin{cases} 
R_{w(j)} & \text{if } 2 \leq j \leq N - 1, \\
1 & \text{otherwise}.
\end{cases}
$$

With this new notation, we can write (53) as

$$
\Psi G^{[2]}_{[2]} (:, r_{n-1} T_n) = T^{(q)} T^{(q-1)} \cdots T^{(1)} C \sum_{\{k_j\}_{j=1}^{2^n+1} \atop j=1} 2^n H^{(j)} (:, \overline{k_j k_{j+1}}), \quad (54)
$$

where each summation index $k_j$ goes over values $k_j \in [K^{(0)}_j]$. Using Lemma 15, Equation (54) can be written as

$$
\Psi G^{[2]}_{[2]} (:, r_{n-1} T_n) = T^{(q)} T^{(q-1)} \cdots T^{(1)} \sum_{\{k_j\}_{j=1}^{2^n+1} \atop j=1} 2^n \otimes C_j H^{(j)} (:, \overline{k_j k_{j+1}})
$$

$$
= T^{(q)} T^{(q-1)} \cdots T^{(1)} \sum_{\{k_j\}_{j=1}^{2^n+1} \atop j=1} 2^n \otimes Y^{(0)}_j (:, \overline{k_j k_{j+1}}), \quad (55)
$$

where $Y^{(0)}_j$ was defined in (20). Recalling that $T^{(1)} \overset{\text{def}}{=} \bigotimes_{j=1}^{2^n-1} T^{(1)}_j$, we may further rewrite (55) as

$$
\Psi G^{[2]}_{[2]} (:, r_{n-1} T_n) = T^{(q)} T^{(q-1)} \cdots T^{(2)} \left( \bigotimes_{j=1}^{2^n-1} T^{(1)}_j \right) \sum_{\{k_{j-1}\}_{j=1}^{2^n+1} \atop j=1} 2^{n-1} \otimes \left( Y^{(0)}_{2j-1} (:, \overline{k_{2j-1} k_{2j}}) \otimes Y^{(0)}_{2j} (:, \overline{k_{2j} k_{2j+1}}) \right)
$$

$$
= T^{(q)} T^{(q-1)} \cdots T^{(2)} \sum_{\{k_{j-1}\}_{j=1}^{2^n+1} \atop j=1} 2^{n-1} \otimes \sum_{k_{2j-1}} \otimes T^{(1)}_j \left( Y^{(0)}_{2j-1} (:, \overline{k_{2j-1} k_{2j}}) \otimes Y^{(0)}_{2j} (:, \overline{k_{2j} k_{2j+1}}) \right)
$$

$$
= T^{(q)} T^{(q-1)} \cdots T^{(2)} \sum_{\{k_{j-1}\}_{j=1}^{2^n+1} \atop j=1} 2^{n-1} \otimes Y^{(1)}_j (:, \overline{k_{2j-1} k_{2j+1}}),
$$

22
where the second equality follows from Lemma 15, and each $Y_j^{(1)}$ is defined as in (21). Defining $K_j^{(1)} \defeq K_{2j-1}^{(0)}$ for $j \in [2^{q-1} + 1]$, we can further rewrite the equation above as

$$
\Psi G^{\neq n}_{[2]}(:, \frac{r_n-1}{r_n}) = T^{(q)} T^{(q-1)} \ldots T^{(2)} \sum_{\{k_j\}_{j=1}^{2^{q-1}+1}} 2^{q-1} \bigotimes Y_j^{(1)}(:, k_j k_{j+1}),
$$

where each summation index $k_j$ now goes over the values $k_j \in [K_j^{(1)}]$. In general, for $m \in [q]$, we have

$$
T^{(q)} T^{(q-1)} \ldots T^{(m)} \sum_{\{k_j\}_{j=1}^{2^{m-1}+1}} 2^{q-m} \bigotimes Y_j^{(m-1)}(:, k_j k_{j+1}) = T^{(q)} T^{(q-1)} \ldots T^{(m+1)} \sum_{\{\ell_j\}_{j=1}^{2^{m-1}+1}} 2^{q-m} \bigotimes Y_j^{(m)}(:, \ell_j \ell_{j+1}),
$$

where the summation indices $k_j$ and $\ell_j$ take on values $k_j \in [K_j^{(m-1)}]$ and $\ell_j \in [K_j^{(m)}]$, respectively, where $K_j^{(m)} \defeq K_{2j-1}^{(m-1)}$ for $j \in [2^{q-m} + 1]$, and where each $Y_j^{(m)}$ is defined as in (21). Combining (56) and (57), it follows by induction that

$$
\Psi G^{\neq n}_{[2]}(:, \frac{r_n-1}{r_n}) = \sum_{k_1 \in [K_1^{(q)}]} \sum_{k_2 \in [K_2^{(q)}]} Y_1^{(q)}(:, k_1 k_2) = Y_1^{(q)},
$$

where the last equality follows since $K_1^{(q)} = K_2^{(q)} = 1$.

### 7.6 Proof of Lemma 13

Throughout the following computations, the summation indices go over $i = \frac{r_{i_n+1} \ldots i_{N+1} \ldots r_{i_n}}{r_{i_n-1}} \in [\prod_{j \neq n} I_j]$ with each $i_j \in [I_j]$, and $r_j, k_j \in [R_j]$ for each $j \in [N]$. Recall that $\Phi \defeq V_1 \Sigma_1^{-1} (V_1 \Sigma_1^{-1})^\top$, where $\Psi G^{\neq n}_{[2]} = U_1 \Sigma_1 V_1^\top$ is a compact SVD. From (10) we have

$$
\tilde{\ell}_i(G^{\neq n}_{[2]}) = e_i^\top G^{\neq n}_{[2]} \Phi G^{\neq n}_{[2]}^\top e_i = (G^{\neq n}_{[2]} \Phi G^{\neq n}_{[2]}^\top)(i, i) = \sum_{r_n-1, \frac{r_n}{k_{n-1} \ldots k_n}} G^{\neq n}_{[2]}(i, \frac{r_n-1}{r_n}) \Phi(\frac{r_n-1}{r_n}, \frac{k_{n-1} \ldots k_n}{k_{n-1} \ldots k_n}) G^{\neq n}_{[2]}(i, \frac{k_{n-1} \ldots k_n}{k_{n-1} \ldots k_n}).
$$

From Definitions 1 and 2 it follows that

$$
G^{\neq n}_{[2]}(i, \frac{r_n-1}{r_n}) = G^{\neq n}_{[2]}(i, \frac{r_n+1 \ldots r_n+1}{r_n+1 \ldots r_n}) = \sum_{\{j\}_{j \neq n-1, n, j \neq n}} \prod_{j \neq n} G^{(j)}_{[2]}(i_j, \frac{r_j r_{j-1}}{r_j r_{j-1}}).
$$

---

3 The only difference between (52) and (59) is that the terms in the product are arranged in a different order.
Using (58) and (59), we have

$$C \overset{\text{def}}{=} \sum_i \sum_{r, n = 1, n \neq n} \left( \sum_{\{r_j\}, j \neq n} \prod_{i, j, r_j, j = r_j} G_{[2]}^{(i, r_j, j = r_j)} \right) \Phi_{[2]}(r_{n-1}r_n, k_{n-1}k_n) \left( \sum_{\{k_j\}, j \neq n} \prod_{i, j, k_j} G_{[2]}^{(i, k_j, j = k_j)} \right)$$

$$= \sum_{r, n = 1, n \neq n} \Phi_{[2]}(r_{n-1}r_n, k_{n-1}k_n) \prod_{j \neq n} \left( \sum_{i, j} G_{[2]}^{(i, r_j, j = r_j)} G_{[2]}^{(i, k_j, j = k_j)} \right)$$

$$= \sum_{r, n = 1, n \neq n} \Phi_{[2]}(r_{n-1}r_n, k_{n-1}k_n) \prod_{j \neq n} \left( \prod_{i, j} \left( G_{[2]}^{(i, r_j, j = r_j)} \right) \left( G_{[2]}^{(i, k_j, j = k_j)} \right) \right),$$

which proves the expression in (22).

Moreover, using (58) and (59) we have that the marginal probability of drawing \((i_j)_{j \leq m, j \neq n}\) is

$$\mathbb{P}(i_j)_{j \leq m, j \neq n} = \frac{1}{C} \sum_{\{i_j\}, j > m, j \neq n} \hat{C}_{i_j} \left( G_{[2]}^{\neq n} \right)$$

$$= \frac{1}{C} \sum_{\{i_j\}, j > m, j \neq n} \sum_{r, n = 1, n \neq n} \Phi_{[2]}(r_{n-1}r_n, k_{n-1}k_n) \left( \sum_{\{r_j\}, j \neq n} \prod_{i, j} G_{[2]}^{(i, r_j, j = r_j)} \right) \left( \sum_{\{k_j\}, j \neq n} \prod_{i, j} G_{[2]}^{(i, k_j, j = k_j)} \right)$$

$$= \frac{1}{C} \sum_{r, n = 1, n \neq n} \Phi_{[2]}(r_{n-1}r_n, k_{n-1}k_n) \prod_{j \neq n} \left( \prod_{i, j, m \leq m, j \neq n} \left( G_{[2]}^{(i, r_j, j = r_j)} \right) \left( G_{[2]}^{(i, k_j, j = k_j)} \right) \right),$$

which proves (23).

### 8 Detailed Complexity Analysis

#### 8.1 CP-ALS-ES: Proposed Sampling Scheme for CP Decomposition

In this section we derive the computational complexity of the scheme proposed in Section 4.1.

**Computing \(\Psi A^{\neq n}\)** First, we consider the costs of computing \(\Psi A^{\neq n}\) as described in Section 4.1.1:

- Computing \(Y_{j}^{(0)}\) for all \(j \in [2^q]\): Each \(C_j A_{[2]}^{(0)}\) costs at most \(O(I_{v(j)} R)\) to compute, and each \(C_j (e_1 1_{[1] \times R})\) costs \(O(R)\) to compute. Since \(2^q \leq 2N\), the total cost for this step is therefore \(O(R \sum_{j \neq n} I_j)\).

- Computing \(Y_{j}^{(m)}\) for all \(m \in [q]\) and all \(j \in [2^q - m]\): A single \(J_1 \times J_2^2\) TensorSketch costs \(O(R J_1 \log J_1)\) to apply to a matrix of the form \(Y_{2j-1}^{(m-1)} \odot Y_{2j}^{(m-1)}\). Such a TensorSketch is applied a total of \(\sum_{m=1}^q 2^{q-m} = 2^q - 1 = O(N)\) times, so the total cost of this whole step is therefore \(O(N R J_1 \log J_1)\).

The cost for computing \(\Psi A^{\neq n}\) is therefore

$$O\left(R \left(N J_1 \log J_1 + \sum_{j \neq n} I_j\right)\right).$$

**Drawing \(J_2\) Samples** Second, we consider the cost of drawing \(J_2\) samples in \(\prod_{j \neq n} I_j\) from the distribution \(q\) as described in Section 4.1.2:
• One-time costs: Computing the SVD of $\Psi A^{\neq n}$ costs $O(J_1 R^2)$. Computing $\Phi = V_1 \Sigma_1^{-1} (V_1 \Sigma_1^{-1})^T$ costs $O(R^3)$. Moreover, we can compute all products $A^{(j)^T} A^{(j)}$ for $j \neq n$ upfront for a cost of $O(R^2 \sum_{j \neq n} I_j)$. The sum of these one-time costs is $O(R^2 (J_1 + R + \sum_{j \neq n} I_j))$.

• Cost of sampling $J_2$ indices: Since each $A^{(j)^T} A^{(j)}$ for $j \neq n$ has already been computed, the cost of computing the probability $P(i_m | (i_j)_{j < m, j \neq n})$ for a single set $(i_j)_{j \leq m, j \neq n}$ via (15) and (16) is $O(R^2 N)$. The total cost for computing the whole distribution for $i_m \in [I_m]$, for all $m \in [N] \setminus \{n\}$, is therefore $O(R^2 N \sum_{j \neq n} I_j)$. Since the main cost of sampling an index $i = i_1 \cdots i_n = i_{n+1} \cdots i_N$ is computing the distribution for each subindex, and we need to sample a total of $J_2$ samples, it follows that the total cost of drawing $J_2$ samples is $O(J_2 R^2 N \sum_{j \neq n} I_j)$.

In total, when including both one-time and per-sample costs, we get a total cost for drawing $J_2$ samples from $q$ of

$$O \left( R^2 \left( J_1 + R + J_2 N \sum_{j \neq n} I_j \right) \right). \quad (61)$$

**Sample Least Squares Problem** Finally, we consider the cost of constructing and solving the sampled least squares problem once the $J_2$ samples in $[\prod_{j \neq n} I_j]$ have been drawn:

• Once the $J_2$ samples in $[\prod_{j \neq n} I_j]$ are drawn, it costs $O(J_2 R N)$ to form $S A^{\neq n}$, and $O(J_2 I_n)$ to form $S X^\top_{(n)}$. This can be done implicitly without forming the matrices $S$, $A^{\neq n}$, and $X^\top_{(n)}$.

• The cost of computing the solution $\tilde{A}^\top = (S A^{\neq n})^\dagger S X^\top_{(n)}$ using a standard method (e.g., via QR decomposition) is $O(J_2 R^2 + J_2 R I_n)$; see Section 5.3.3 in Golub and Van Loan (2013) for details.

In total, the costs of constructing and solving the least squares problem is therefore

$$O(J_2 R(N + R + I_n)). \quad (62)$$

**Total Per-Iteration Cost for CP-ALS-ES** Recall that for each iteration of CP-ALS, we need to solve $N - 1$ least squares problems. Consequently, adding the costs in (60), (61), (62) and multiplying by $N - 1$, we get a total cost per iteration of

$$O \left( R N^2 J_1 \log J_1 + R^2 N \left( J_1 + R + J_2 N \sum_{j \neq n} I_j \right) + J_2 R N I_n \right).$$

If the sketch rates $J_1$ and $J_2$ are chosen according to (11) and (12), this per-iteration cost becomes

$$O \left( \frac{R^3 N^3}{\delta} \log \left( \frac{R^2 N}{\delta} \right) + \frac{R^4 N^2}{\delta} + \left( R^3 N^2 \sum_{j \neq n} I_j + R^2 N I_n \right) \max \left( \log \left( \frac{R}{\delta} \right), \frac{1}{\varepsilon \delta} \right) \right).$$

### 8.2 TR-ALS-ES: Proposed Sampling Scheme for Tensor Ring Decomposition

In this section we derive the computational complexity of the scheme proposed in Section 4.2.

**Computing $\Psi G^{\neq n}_{[2]}$** First, we consider the computation of $\Psi G^{\neq n}_{[2]}$ described in Section 4.2.1:

• Computing $Y^{(0)}_j$ for all $j \in [2^q]$; Recall that computing $C_j H^{(j)}$ costs $mz(H^{(j)})$. Consequently, the cost of computing all $Y^{(0)}_j$ for $N \leq j \leq 2^q$ is just $O(1)$. The total cost for this step is therefore $O(\sum_{j=1}^N I_j R_{j-1} R_j)$. 

25
• Computing $Y_j^{(m)}$ for all $m \in [q]$ and all $j \in [2^m-1]$: Computing $T_j^{(m)}(Y_{2j-1}^{(m-1)}(:, k_1 k_2) \otimes Y_{2j}^{(m-1)}(:, k_2 k_3))$ requires applying a $J_1 \times J_2^3$ TensorSketch to the Kronecker product of two vectors, which costs $O(J_1 \log J_1)$. This needs to be done for each $k_2 \in [K_{2j-1}^{(m-1)}]$, when computing the sum in (21). This sum, in turn, needs to be computed for all $k_1 \in [K_{2j-1}^{(m-1)}]$, $k_3 \in [K_{2j+1}^{(m-1)}]$ and $j \in [2^m-1]$. Doing this for each $m \in [q]$ brings the total cost of this step to

$$O\left(\sum_{m=1}^{q} \sum_{j=1}^{2^m-1} \sum_{k_1=1}^{K_{2j-1}^{(m-1)}} \sum_{k_2=1}^{K_{2j}^{(m-1)}} \sum_{k_3=1}^{K_{2j+1}^{(m-1)}} J_1 \log J_1 \right).$$

As we will see further down, this expression simplifies considerably if e.g. all $R_i$ are assumed to be equal.

Adding up the per-column costs above and multiplying them by the number of columns $R_n - R_n$, we get that the cost for computing $\Psi G_{[2]}^{[n]}$ is

$$O \left( R_{n-1} R_n \left( \sum_{j=1}^{N} I_j R_{j-1} R_j + \sum_{m=1}^{q} \sum_{j=1}^{2^m-1} \sum_{k_1=1}^{K_{2j-1}^{(m-1)}} \sum_{k_2=1}^{K_{2j}^{(m-1)}} \sum_{k_3=1}^{K_{2j+1}^{(m-1)}} J_1 \log J_1 \right) \right).$$  (63)

**Drawing $J_2$ Samples**  Second, we consider the cost of drawing $J_2$ samples in $[\prod_{j \neq n} I_j]$ from the distribution $q$ as described in Section 4.2.2:

• One-time costs: Computing the SVD of $\Psi G_{[2]}^{[n]}$ costs $O(J_1 (R_{n-1} R_n)^2)$. Computing $\Phi = V_1 \Sigma_1^{-1} (V_1 \Sigma_1^{-1})^\top$ costs $O((R_{n-1} R_n)^3)$. Moreover, we can compute all products $G_{[2]}^{(j)} G_{[2]}^{(j)}$ for $j \neq n$ upfront for a cost of $O(\sum_{j \neq n} (R_{j-1} R_j)^2 I_j)$. The sum of these one-time costs is $O(J_1 (R_{n-1} R_n)^2 + (R_{n-1} R_n)^3 + \sum_{j \neq n} (R_{j-1} R_j)^2 I_j)$.

• Cost of sampling $J_2$ indices: The main cost of drawing the samples is computing the sampling distributions. Even though the number of terms in the sum of (23) is exponential in $N$, the joint probability distribution can be computed efficiently. We discuss how to do this in Remark 19. The cost of doing this for one set of indices $(i_j)_{j \leq m, j \neq n}$ is given in (67). Repeating this for all $i_j \in [I_j]$, which is required to get the distribution for the $j$th index, brings the cost to

$$O \left( I_j R_n^2 \sum_{d=1}^{N-1} R_d^2 R_{d+1}^2 \right).$$

When this is repeated for all $N$ indices, and a total of $J_2$ times to get all samples, this brings the cost to

$$O \left( J_2 \left( \sum_{j=1}^{N} I_j \right) R_n^2 \sum_{d=1}^{N-1} R_d^2 R_{d+1}^2 \right).$$

Adding the one-time costs and the costs associated to computing the distributions, we get the following total cost for drawing $J_2$ samples:

$$O \left( J_1 (R_{n-1} R_n)^2 + (R_{n-1} R_n)^3 + J_2 \left( \sum_{j=1}^{N} I_j \right) R_n^2 \sum_{d=1}^{N-1} R_d^2 R_{d+1}^2 \right).$$  (64)
Sampled Least Squares Problem  Finally, we consider the cost of constructing and solving the sampled least squares problem once the $J_2$ samples in $\prod_{j \neq n} I_j$ have been drawn:

- Once $J_2$ samples in $\prod_{j \neq n} I_j$ are drawn, the sketched design matrix $SG^{\neq n}_{[2]}$ can be computed efficiently without having to form the full matrix $G^{\neq n}_{[2]}$. We provide further details in Remark 20. With this approach, the cost of forming $SG^{\neq n}_{[2]}$ is

$$O(J_2 R_n \sum_{j \in [N] \setminus \{n, n+1\}} R_{j-1} R_j).$$

Forming $SX^\top_{[n]}$ by sampling the appropriate rows costs $O(J_2 I_n)$.

- The cost of computing the solution $\tilde{G}^\top = (SG^{\neq n}_{[2]} \mid SX^\top_{[n]})$ using a standard method (e.g., via QR decomposition) is $O(J_2 (R_{n-1} R_n)^2 + J_2 R_{n-1} n I_n)$; see Section 5.3.3 in Golub and Van Loan (2013) for details.

In total, the cost of constructing and solving the least squares problem is therefore

$$O\left( J_2 \left( R_n \sum_{j \in [N] \setminus \{n, n+1\}} R_{j-1} R_j + (R_{n-1} R_n)^2 + R_{n-1} n I_n \right) \right).$$

Total Per-Iteration Cost for TR-ALS-ES  Recall that for each iteration of TR-ALS, we need to solve $N - 1$ least squares problems. Consequently, adding the costs in (63), (64), (65) and multiplying by $N - 1$, we get the total per-iteration cost. If we assume that $R_j = R$ and $I_j = I$ for all $j \in [N]$, the expression simplifies considerably and we get a total per-iteration cost of

$$O(N^2 R^5 J_1 \log J_1 + N^3 IR^6 J_2).$$

If the sketch rates $J_1$ and $J_2$ are chosen according to (17) and (18), this per-iteration cost becomes

$$O\left( \frac{N^3 R^9}{\delta} \log \left( \frac{NR^4}{\delta} \right) + N^3 IR^8 \cdot \max \left( \log \left( \frac{R^2}{\delta} \right), \frac{1}{\epsilon \delta} \right) \right).$$

Remark 19. At first sight, the joint probability computation in (23) looks expensive since the number of terms in the sum is exponential in $N$. However, since not all summation indices $r_j$ and $k_j$ appear in every term, the summation can be done more efficiently. In fact, the computation (23) can be viewed as the evaluation of a tensor ring, which can be done efficiently by contracting core tensors pairwise. To see this, define core tensors $c^{(j)}$ for $j \in [N]$ as follows:

- For $j \leq m$ and $j \neq n$, let $c^{(j)} \in \mathbb{R}^{R_{j-1} \times I_j \times R_j}$ be defined elementwise via

$$c^{(j)}(r_{j-1} k_{j-1}, i_j, r_j k_j) \overset{\text{def}}{=} G^{(j)}_{[2]}(i_j, r_j, r_{j-1}) G^{(j)}_{[2]}(i_j, k_j, k_{j-1}).$$

- For $m < j \leq N$ and $j \neq n$, let $c^{(j)} \in \mathbb{R}^{R_{j-1} \times 1 \times R_j}$ be defined elementwise via

$$c^{(j)}(r_{j-1} k_{j-1}, 1, r_j k_j) \overset{\text{def}}{=} (G^{(j)}_{[2]} G^{(j)}_{[2]} )^\top(r_j r_{j-1}, k_j k_{j-1}).$$

- For $j = n$, let $c^{(j)} = c^{(n)} \in \mathbb{R}^{R_{n-1} \times 1 \times R_n}$ be defined elementwise via

$$c^{(n)}(r_{n-1} k_{n-1}, 1, r_n k_n) \overset{\text{def}}{=} \frac{1}{C} \Phi(r_{n-1} r_n, k_{n-1} k_n).$$

27
We can now rewrite the expression in (23) as
\[ P((i_j)_{j \leq m}, j \neq n) = \text{TR}(\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(N)})_{\xi_1, \ldots, \xi_N}, \]
where
\[ \xi_j = \begin{cases} i_j & \text{if } j \leq m, j \neq n, \\ 1 & \text{otherwise}. \end{cases} \]

As discussed in Zhao et al. (2016), the value of an entry in a tensor ring can be computed via a sequence of matrix-matrix products follows by taking the matrix trace:
\[ \text{TR}(\mathbf{e}^{(1)}, \ldots, \mathbf{e}^{(N)})_{\xi_1, \ldots, \xi_N} = \text{trace} \left( (\mathbf{C}(1)(:, \xi_1, :)) \cdot (\mathbf{C}(2)(:, \xi_2, :) \cdot \cdots \cdot (\mathbf{C}(N)(:, \xi_N, :)) \right), \]
where each \( \mathbf{e}^{(j)}(:, \xi_j, :) \) is treated as a \( R_j \times R_j \) matrix. If the matrix product in (66) is done left to right, evaluating the right hand side costs
\[ O\left( R_N^2 \sum_{j=1}^{N-1} R_j^2 R_{j+1} \right). \]

**Remark 20.** As described in Malik and Becker (2021), it is possible to construct the sketched design matrix \( \mathbf{S}G^{[2]}_{\neq n} \) efficiently without first forming the full matrix \( \mathbf{G}^{[2]}_{\neq n} \). To see how, note that each row \( \mathbf{G}^{[2]}_{\neq n}(i,:) \) is the vectorization of the tensor slice \( \mathbf{G}^{[2]}_{\neq n}(i, :) \) due to Definition 1. From Definition 2, the tensor slice \( \mathbf{G}^{[2]}_{\neq n}(i, :) \) is given by
\[ \mathbf{G}^{[2]}_{\neq n}(i, :) = S^{(n+1)(i, n+1, :) \cdots S^{(N)(i, N, :) \cdot S^{(1)(i, 1, :) \cdots S^{(n-1)(i, n-1, :).}} \]

Suppose \( v \in [\prod_{j \neq n} I_j]^{J_2} \) contains the \( J_2 \) sampled indices corresponding to the sketch \( \mathbf{S} \). Let \( \mathbf{G}^{[2]}_{\neq n} = \mathbb{R}^{R_n \times J_2 \times R_n - 1} \) be a tensor which we define as follows: For each \( j \in [J_2] \), let \( i = i_{n+1} \cdots i_N i_{n+1} \cdots i_{n-1} = v(j) \) and define
\[ \mathbf{G}^{[2]}_{\neq n}(i, :) = 1/\sqrt{J_2} = \mathbf{G}^{[1]}(i_{n+1}, :) \cdots S^{(N)(i, N, :) \cdot S^{(1)(i, 1, :) \cdots S^{(n-1)(i, n-1, :).}} \]

We now have \( \mathbf{SG}^{[2]}_{\neq n} = \mathbf{G}^{[2]}_{\neq n} \). If the matrix product in (68) is computed from left to right, it costs
\[ O\left( R_n \sum_{j \in [N \setminus \{n, n+1\}] R_j - 1 R_j \right). \]
Since this needs to be computed for each \( j \in [J_2] \), the total cost for computing \( \mathbf{SG}^{[2]}_{\neq n} \) via this scheme is
\[ O\left( J_2 R_n \sum_{j \in [N \setminus \{n, n+1\}] R_j - 1 R_j \right). \]

We refer the reader to Malik and Becker (2021) for further details.

### 8.3 Complexity Analysis of Competing Methods

In this section we provide a few notes on how we computed the computational complexity of the other methods we compare with in Tables 1 and 2.

#### 8.3.1 CP-ALS

The standard way to implement CP-ALS is given in Figure 3.3 in Kolda and Bader (2009). The leading order cost per least squares solve for that algorithm is
\[ O(NIR^2 + R^3 + NIN^{-1}R + INR). \]
Since \( N \) least squares problems need to be solved each iteration, the per-iteration cost is
\[ O(N^2IR^2 + N^2R^3 + N^2IN^{-1}R + NiNR). \]
When \( N \) is large, this becomes \( O(N(N + I)IN^{-1}R) \) which is what we report in Table 1.
8.3.2 SPALS

Cheng et al. (2016) only give the sampling complexity for case when \( N = 3 \) in their paper. For arbitrary \( N \), and without any assumptions on the rank of the factor matrices or the Khatri–Rao product design matrix, their scheme requires \( J \gtrsim R^N \log(I_n/\delta)/\varepsilon^2 \) samples when solving for the \( n \)th factor matrix in order to achieve the additive error guarantees in Theorem 4.1 of their paper.\(^4\)

SPALS requires a one-time upfront cost of \( \text{nnz}(\mathbf{X}) \) in order to compute the second term in Equation (5) in Cheng et al. (2016). In SPALS, the \( n \)th factor is updated via

\[
A^{(n)} = \mathbf{X}_{(n)} \mathbf{S}^\top \left( \bigotimes_{j=N \atop j \neq n}^{1} A^{(j)} \right) \left( \bigotimes_{j=1 \atop j \neq n}^{N} A^{(j)\top} A^{(j)} \right)^{-1},
\]

where \( \mathbf{S} \) is a sampling matrix. When this is computed in the appropriate order, and if log factors are ignored and we assume that \( I_n = I \) for all \( n \in [N] \), then the cost of computing \( A^{(n)} \) is

\[
\tilde{O}(NIR^2 + (N+I)R^N+1/\varepsilon^2).
\]

Notice that the cost of computing the sampling distribution is dominated by the cost above. Since \( N \) factor matrices need to be updated per iteration, the total per-iteration cost is

\[
\tilde{O}(N^2IR^2 + N(N+I)R^N+1/\varepsilon^2).
\]

When \( N \) is large, this becomes \( \tilde{O}(N(N+I)R^N+1/\varepsilon^2) \), which is what we report in Table 1.

8.3.3 CP-ARLS-LEV

From Theorems 5.2 and 5.3 in Larsen and Kolda (2020), the sampling complexity for CP-ARLS-LEV required to achieve relative error guarantees when solving for the \( n \)th factor matrix is \( J \gtrsim R^{N-1} \log(\max(R, I_n)/\delta)/\varepsilon^2 \). Solving the sampled least squares problem, which has a design matrix of size \( J \times R \) and \( I_n \) right hand sides via e.g. QR decomposition (see Section 5.3.3 in Golub and Van Loan (2013)) will therefore cost \( O((R + I_n)R^N \log(\max(R, I_n)/\delta)/\varepsilon^2) \). Each iteration requires solving \( N \) such least squares problems. If we assume that \( I_n = I \) for all \( n \in [N] \) and ignore log factors, the per-iteration cost becomes

\[
\tilde{O}(N(R + I)R^N/\varepsilon^2),
\]

which is what we report in Table 1.

8.3.4 Methods for Tensor Ring Decomposition

The complexities we report in Table 2 for other methods where taken directly from Table 1 in Malik and Becker (2021).

9 Additional Experiment Details

9.1 Details on Algorithm Implementations

For CP-ALS, we use the implementation available in Tensor Toolbox version 3.2.1 (Bader and Kolda, 2006; Bader et al., 2021). We wrote our own implementation of CP-ARLS-LEV in Matlab based on Algorithm 6.3 in Larsen and Kolda (2020). We do not use any hybrid-deterministic sampling, but we do combine repeated rows. The implementation of our own CP-ALS-ES is also done in Matlab, with key functions written in C and incorporated via Matlab’s MEX interface.

For TR-ALS and TR-ALS-Sampled, we use the implementations by Malik and Becker (2021). Our own TR-ALS-ES is implemented by appropriately modifying the code for TR-ALS-Sampled.

\(^4\)If the Khatri–Rao product design matrix is full rank, which happens if all factor matrices are full rank, then \( J \gtrsim R^{N-1} \log(I_n/\delta)/\varepsilon^2 \) samples will suffice.
9.2 Datasets

The photo used for the sampling distribution comparison was taken by Sebastian Müller on Unsplash and is available at https://unsplash.com/photos/l54ZALpH2_I. We converted this figure to gray scale by averaging the three color channels. We also cropped the image slightly to make the width and height a power of 2. The tensorization is done following the ideas for visual data tensorization discussed in Yuan et al. (2019b). Please see our code for precise details.

The COIL-100 dataset was created by Nene et al. (1996) and is available for download at https://www.cs.columbia.edu/CAVE/software/softlib/coil-100.php.

9.3 Sampling Distribution Plots

We have included figures below that compare the sampling distributions used by our methods with those used by the previous state-of-the-art methods in the least squares problem considered in the first experiment in Section 5. For a rank-10 CP decomposition of the tabby cat tensor, Figure 1 shows the exact leverage score distribution ($p$ in Definition 7), the sampling distribution used by CP-ARLS-LEV, and a realization (for $J_1 = 1000$) of the distribution our CP-ALS-ES uses. Figure 2 shows the same things as Figure 1, but for a rank-20 CP decomposition. For a rank-$(3,\ldots,3)$ tensor ring decomposition of the tabby cat tensor, Figure 3 shows the exact leverage score distribution, the sampling distribution used by TR-ALS-Sampled, and a realization (for $J_1 = 1000$) of the distribution our TR-ALS-ES uses. Figure 4 shows the same things as Figure 2, but for a rank-$(5,\ldots,5)$ tensor ring decomposition.

Notice that the sampling distribution that our methods use follow the exact leverage score sampling distribution closely. The distributions used by CP-ARLS-LEV and TR-ALS-Sampled are less accurate. In particular, when $R > I = 16$ for the CP decomposition (Figure 2) or when $R_{n-1}R_n > I = 16$ for the tensor ring decomposition (Figure 4), CP-ARLS-LEV and TR-ALS-Sampled sample from a uniform distribution. This is not an anomaly, but rather a direct consequence from how those methods estimate the leverage scores. Our proposed methods, by contrast, handle those cases well.

![Figure 1: Comparison of the exact leverage score distribution, the sampling distribution used by CP-ARLS-LEV, and a realization (for $J_1 = 1000$) of the distribution used by our CP-ALS-ES. The least squares problem corresponds to solving for the 6th factor matrix in a rank-10 CP decomposition of the 6-way tabby cat tensor.](image)

9.4 Feature Extraction Experiments

We provide some further details on the feature extraction experiments in Section 5 in this section. For a rank-25 CP decomposition, the 4th factor matrix is of size $7200 \times 25$. We directly use this factor matrix as the feature matrix we feed to the $k$-NN method in Matlab. For the rank-$(5,\ldots,5)$ tensor ring decomposition, the
Figure 2: Same as Figure 1, but for a rank-20 decomposition.

Figure 3: Comparison of the exact leverage score distribution, the sampling distribution used by TR-ALS-Sampled, and a realization (for $J_1 = 1000$) of the distribution used by our TR-ALS-ES. The least squares problem corresponds to solving for the 6th core tensor in a rank-$(3, \ldots, 3)$ tensor ring decomposition of the 6-way tabby cat tensor.

4th core tensor is of size $5 \times 7200 \times 5$. We turn this into a $7200 \times 25$ matrix via a classical mode-2 unfolding which we then use as the feature matrix in the $k$-NN algorithm.
Figure 4: Same as Figure 4, but for a rank-$(5, \ldots, 5)$ decomposition.