Subquadratic Kronecker Regression with Applications to Tensor Decomposition

Matthew Fahrbach[*]
Google Research
fahrbach@google.com

Gang Fu
Google Research
thomasfu@google.com

Mehrdad Ghadiri
Georgia Tech
ghadiri@gatech.edu

Abstract

Kronecker regression is a highly-structured least squares problem $\min_{\mathbf{x}} \|\mathbf{K}\mathbf{x}-\mathbf{b}\|_2^2$, where the design matrix $\mathbf{K} = \mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(N)}$ is a Kronecker product of factor matrices. This regression problem arises in each step of the widely-used alternating least squares (ALS) algorithm for computing the Tucker decomposition of a tensor. We present the first *subquadratic-time* algorithm for solving Kronecker regression to a $(1+\varepsilon)$-approximation that avoids the exponential term $O(\varepsilon^{-N})$ in the running time. Our techniques combine leverage score sampling and iterative methods. By extending our approach to block-design matrices where one block is a Kronecker product, we also achieve subquadratic-time algorithms for (1) Kronecker ridge regression and (2) updating the factor matrices of a Tucker decomposition in ALS, which is not a pure Kronecker regression problem, thereby improving the running time of all steps of Tucker ALS. We demonstrate the speed and accuracy of this Kronecker regression algorithm on synthetic data and real-world image tensors.

1 Introduction

Tensor decomposition has a rich multidisciplinary history with countless applications in data mining, machine learning, and signal processing [35, 55, 58, 31]. The most widely-used tensor decompositions are the CP decomposition and the Tucker decomposition. Similar to the singular value decomposition of a matrix, both decompositions have natural analogs of *low-rank* structure. Unlike matrix factorization, however, computing the rank of a tensor and the best rank-one tensor are NP-hard [27]. Therefore, most low-rank tensor decomposition algorithms decide on the rank structure in advance, and then optimize the variables of the decomposition to fit the data. While conceptually simple, this approach is extremely effective in practice for many applications.

The *alternating least squares* (ALS) algorithm is the main workhorse for low-rank tensor decomposition, e.g., it is the first algorithm mentioned in the MATLAB Tensor Toolbox [7]. For both CP and Tucker decompositions, ALS cyclically optimizes disjoint blocks of variables while keeping all others fixed. As the name suggests, each step solves a linear regression problem. The *core tensor* update step in ALS for Tucker decompositions is notoriously expensive but highly structured. In fact, the design matrix of this regression problem is the Kronecker product of the factor matrices of the Tucker decomposition $\mathbf{K} = \mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(N)}$. Our work builds on a line of Kronecker regression algorithms [17, 18, 47] to give the first *subquadratic-time* algorithm for solving Kronecker regression to a $(1+\varepsilon)$-approximation while avoiding an exponential term of $O(\varepsilon^{-N})$ in the running time.

We combine leverage score sampling, iterative methods, and a novel way of multiplying sparsified Kronecker product matrices to fully exploit the Kronecker structure of the design matrix. We also extend our approach to block-design matrices where one block is a Kronecker product, achieving

[*]Authors are listed alphabetically. A preliminary version of this work that focuses on efficient sketching for Tucker decompositions appears in `arXiv:2107.10654` [21].

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

subquadratic-time algorithms for (1) Kronecker ridge regression and (2) updating the factor matrix of a Tucker decomposition in ALS, which is not a pure Kronecker regression problem. Putting everything together, this work improves the running time of all steps of ALS for Tucker decompositions and runs in time that is sublinear in the size of the input tensor, linear in the error parameter ε^{-1}, and subquadratic in the number of columns of the design matrix in each step. Our algorithms support L2 regularization in the Tucker loss function, so the decompositions can readily be used in downstream learning tasks, e.g., using the factor matrix rows as embeddings for clustering [67]. Regularization also plays a critical role in the more general tensor completion problem to prevent overfitting when data is missing and has applications in differential privacy [10, 8].

The current-fastest Kronecker regression algorithm of Diao et al. [18] uses leverage score sampling and achieves the following running times for $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ with $I_n \geq R_n$, for all $n \in [N]$, where $R = \prod_{n=1}^{N} R_n$ and $\omega < 2.373$ denotes the matrix multiplication exponent [4]:

1. $\tilde{O}(\sum_{n=1}^{N}(\text{nnz}(\mathbf{A}^{(n)}) + R_n^{\omega}) + R^{\omega}\varepsilon^{-1})$ by sampling $\tilde{O}(R\varepsilon^{-1})$ rows of $\mathbf{K}$ by their leverage scores.
2. $\tilde{O}(\sum_{n=1}^{N}(\text{nnz}(\mathbf{A}^{(n)}) + R_n^{\omega}\varepsilon^{-1}) + R\varepsilon^{-N})$ by sampling $\tilde{O}(R_n\varepsilon^{-1})$ rows from each factor matrix $\mathbf{A}^{(n)}$ and taking the Kronecker product of the sampled factor matrices.

Note that the second approach is linear in R, but the error parameter has an exponential cost in the number of factor matrices. In this work, we show that the running time of the first approach can be improved to subquadratic in R without increasing the running time dependence on ε in the dominant term, simultaneously improving on both approaches.

Theorem 1.1. *For $n \in [N]$, let $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, $I_n \geq R_n$, and $\mathbf{b} \in \mathbb{R}^{I_1 \cdots I_n}$. There is a $(1+\varepsilon)$-approximation algorithm for solving $\arg\min_{\mathbf{x}} \|(\mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(N)})\mathbf{x} - \mathbf{b}\|_2^2$ that runs in time*

$$\tilde{O}\left(\sum_{n=1}^{N}\left(\text{nnz}(\mathbf{A}^{(n)}) + R_n^{\omega}N^2\varepsilon^{-2}\right) + \min_{S\subseteq[N]} \text{MM}\left(\prod_{n\in S} R_n, R\varepsilon^{-1}, \prod_{n\in[N]\setminus S} R_n\right)\right), \tag{1}$$

where $\text{MM}(a,b,c)$ is the running time of multiplying an $a \times b$ matrix with a $b \times c$ matrix.

If we do not use fast matrix multiplication (Gall and Urrutia [24] and Alman and Williams [4]), the last term in (1) is $\tilde{O}(R^2\varepsilon^{-1})$, which is *already an improvement* over the standard $\tilde{O}(R^3\varepsilon^{-1})$ running time. With fast matrix multiplication, $\text{MM}(\prod_{n\in S} R_n, R\varepsilon^{-1}, \prod_{n\in[N]\setminus S} R_n)$ is subquadratic in R for any nontrivial subset $S \notin \{\emptyset, [N]\}$, which is an improvement over $\tilde{O}(R^{\omega}\varepsilon^{-1}) \approx \tilde{O}(R^{2.373}\varepsilon^{-1})$. If there exists a "balanced" subset S such that $\prod_{n\in S} R_n \approx \sqrt{R}$, our running time goes as low as $\tilde{O}(R^{1.626}\varepsilon^{-1})$ using [24]. For ease of notation, we denote the subquadratic improvement by the constant $\theta^* > 0$, where $R^{2-\theta^*} = \min_{S\subseteq[N]} \text{MM}(\prod_{n\in S} R_n, R, \prod_{n\in[N]\setminus S} R_n)$.

Updating the core tensor in the ALS algorithm for Tucker decomposition is a pure Kronecker product regression as described in Theorem 1.1, but updating the factor matrices is a regression problem of the form $\arg\min_{\mathbf{x}} \|\mathbf{KMx} - \mathbf{b}\|_2^2$, where $\mathbf{K}$ is a Kronecker product and $\mathbf{M}$ is a matrix without any particular structure. We show that such problems can be converted to block regression problems where one of the blocks is $\mathbf{K}$. We then develop sublinear-time leverage score sampling techniques for these block matrices, which leads to the following theorem that accelerates all of the ALS steps.

Theorem 1.2. *There is an ALS algorithm for L2-regularized Tucker decompositions that takes a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and returns N factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ and a core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_n}$ such that each factor matrix and core update is a $(1+\varepsilon)$-approximation to the optimum with high probability. The running times of each step are:*

- *Factor matrix $\mathbf{A}^{(k)}$: $\tilde{O}(\sum_{n=1}^{N}(\text{nnz}(\mathbf{A}^{(n)}) + R_n^{\omega}N^2\varepsilon^{-2}) + I_k R_{\neq k}^{2-\theta^*}\varepsilon^{-1} + I_k R\sum_{n=1}^{N} R_n + R_k^{\omega})$,*
- *Core tensor $\mathcal{G}$: $\tilde{O}(\sum_{n=1}^{N}(\text{nnz}(\mathbf{A}^{(n)}) + R_n^{\omega}N^2\varepsilon^{-2}) + R^{2-\theta^*}\varepsilon^{-1})$,*

where $R = \prod_{n=1}^{N} R_n$, $R_{\neq k} = R/R_k$, and $\theta^* > 0$ is a constant derived from fast rectangular matrix multiplication.

Table 1: Running times of `TuckerALS` factor matrix and core tensor updates with different Kronecker regression methods. The factor matrices are denoted by $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$. The input tensor has size $I = I_1 \cdots I_N$ and the core tensor has size $R = R_1 \cdots R_N$. Let $I_{\neq k} = I/I_k$ and $R_{\neq k} = R/R_k$. We use $\omega < 2.373$ to denote the matrix-multiplication exponent and the constant $\theta^* > 0$ for the optimally balanced fast rectangular matrix multiplication as stated in Theorem 4.5, i.e., $R^{2-\theta^*} = \min_{T \subseteq [N]} \text{MM}(\prod_{n \in T} R_n, R, \prod_{n \notin T} R_n)$.

Algorithm	Factor matrix $\mathbf{A}^{(k)}$	Core tensor $\mathcal{G}$
Naive	$O(I_k R R_{\neq k} + I_k R R_k + R_k^\omega + I R_{\neq k} + I_k R_k^2)$	$O(R^\omega + IR)$
This paper (Lemma 4.4)	$O(I_k R(\sum_{n=1}^N R_n) + R_k^\omega + I(\sum_{n \neq k} R_n) + I_k R_k^2)$	$O(R^2 + I\sum_{n=1}^N R_n)$
This paper (Theorem 1.2)	$\tilde{O}(I_k R_{\neq k}^{2-\theta^*} \varepsilon^{-1} + I_k R(\sum_{n=1}^N R_n) + R_k^\omega \varepsilon^{-2})$	$\tilde{O}(R^{2-\theta^*} \varepsilon^{-1})$
Diao et al. [18]	—	$\tilde{O}(R^\omega \varepsilon^{-2})$

For tensors of even modest order, the superlinear term in R is the bottleneck in many applications since R is exponential in the order of the tensor. It follows that our improvements are significant in both theory and practice as illustrated in our experiments in Section 6.

1.1 Our Contributions and Techniques

We present several new results about approximate Kronecker regression and the ALS algorithm for Tucker decompositions. Below is a summary of our contributions:

1. Our main technical contribution is the algorithm `FastKroneckerRegression` in Section 4. This Kronecker regression algorithm builds on the block-sketching tools introduced in Section 3, and combines iterative methods with a fast novel Kronecker-matrix multiplication for sparse vectors and matrices and fast rectangular matrix multiplication to achieve a running time that is *subquadratic* in the number of columns in the Kronecker matrix. A key insight is to use the original (non-sketched) Kronecker product as the preconditioner in the Richardson iterations when solving the sketched problem. This, by itself, improves the running time to quadratic. Then to achieve subqudratic running time, we exploit the singular value decomposition of Kronecker products and present a novel method for multiplying a sparsified Kronecker product matrix (Lemma 4.4 and Theorem 4.5).
2. We generalize our Kronecker regression techniques to work for Kronecker ridge regression and the factor matrix updates in ALS for Tucker decomposition. We show that a factor matrix update is equivalent to solving an *equality-constrained* Kronecker regression problem with a low-rank update to the preconditioner in the Richardson iterations. We can implement these new matrix-vector products nearly as fast by using the Woodbury matrix identity. Thus, we provably speed up each step of Tucker ALS, i.e., the core tensor and factor matrices.
3. We give a block-sketching toolkit in Section 3 that states we can sketch blocks of a matrix by their leverage scores, i.e., their leverage scores in isolation, not with respect to the entire block matrix. This is one of the ways we exploit the Kronecker product structure of the design matrix. This approach can be useful for constructing spectral approximations and for approximately solving block regression problems. One corollary is that we can use the "sketch-and-solve" method for any ridge regression problem (Corollary 3.5).
4. We compare `FastKroneckerRegression` with Diao et al. [18, Algorithm 1] on a synthetic Kronecker regression task studied in [17, 18] and as a subroutine in ALS for computing the Tucker decomposition of various image tensors [44, 50, 51]. Our results demonstrate the importance of reducing the running time dependence on the number of columns in the Kronecker product.

1.2 Related Work

Kronecker Regression. Diao et al. [17] recently gave the first Kronecker regression algorithm based on `TensorSketch` [53] that is faster than forming the Kronecker product. Diao et al. [18] improved this by removing the dependence on $O(\text{nnz}(\mathbf{b}))$ from the running time, where $\mathbf{b} \in \mathbb{R}^{I_1 \cdots I_N}$ is the response vector. Reddy, Song, and Zhang [56] recently initiated the study of *dynamic* Kronecker

regression, where the factor matrices $\mathbf{A}^{(n)}$ undergo updates and the solution vector can be efficiently queried. Marco, Martínez, and Viaña [47] studied the generalized Kronecker regression problem.

Ridge Leverage Scores. Alaoui and Mahoney [3] extended the notion of statistical leverage scores to account for L2 regularization. Sampling from approximate ridge leverage score distributions has since played a key role in sparse low-rank matrix approximation [16], the Nyström method [49], bounding statistical risk in ridge regression [48], and ridge regression [14, 48, 41, 33]. Fast recursive algorithms for computing approximate leverage scores [15] and for solving overconstrained least squares [40] are also closely related.

Tensor Decomposition. Cheng et al. [13] and Larsen and Kolda [38] used leverage score sampling to speed up ALS for CP decomposition.[2] Song et al. [59] gave a polynomial-time, relative-error approximation algorithm for several low-rank tensor decompositions, which include CP and Tucker. Frandsen and Ge [23] showed that if the tensor has an exact Tucker decomposition, then all local minima are globally optimal. Randomized low-rank Tucker decompositions based on sketching have become increasingly popular, especially in streaming applications: [45, 61, 11, 60, 31, 46, 44, 2]. The more general problem of low-rank tensor completion is also a fundamental approach for estimating the values of missing data [1, 43, 29, 28, 22]. Fundamental algorithms for tensor completion are based on ALS [68, 25, 42], Riemannian optimization [37, 34, 52], or projected gradient methods [65].

2 Preliminaries

Notation. The *order* of a tensor is the number of its dimensions. We denote scalars by normal lowercase letters $x \in \mathbb{R}$, vectors by boldface lowercase letters $\mathbf{x} \in \mathbb{R}^n$, matrices by boldface uppercase letters $\mathbf{X} \in \mathbb{R}^{m \times n}$, and higher-order tensors by boldface script letters $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$. We use normal uppercase letters to denote the size of an index set (e.g., $[N] = \{1, 2, \ldots, N\}$). The i-th entry of a vector $\mathbf{x}$ is denoted by x_i, the (i, j)-th entry of a matrix $\mathbf{X}$ by x_{ij}, and the (i, j, k)-th entry of a third-order tensor $\mathcal{X}$ by x_{ijk}.

Algorithm 1 `TuckerALS`

Input: $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, $(R_1, R_2, \ldots, R_N)$, λ

Initialize core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_n}$
Initialize factors $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ for $n \in [N]$
repeat
  for $n = 1$ to N **do**
    $\mathbf{K} \leftarrow \mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(n-1)} \otimes \mathbf{A}^{(n+1)} \otimes \cdots \otimes \mathbf{A}^{(N)}$
    $\mathbf{B} \leftarrow \mathbf{X}_{(n)}$
    for $i = 1$ to I_n **do**
      $\mathbf{y}^* \leftarrow \arg\min_{\mathbf{y}} \|\mathbf{K}\mathbf{G}_{(n)}^\intercal \mathbf{y} - \mathbf{b}_{i:}^\intercal\|_2^2 + \lambda\|\mathbf{y}\|_2^2$
      Update factor row $\mathbf{a}_{i:}^{(n)} \leftarrow \mathbf{y}^{*\intercal}$
  $\mathbf{K} \leftarrow \mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)} \otimes \cdots \otimes \mathbf{A}^{(N)}$
  $\mathbf{g}^* \leftarrow \arg\min_{\mathbf{g}} \|\mathbf{K}\mathbf{g} - \text{vec}(\mathcal{X})\|_2^2 + \lambda\|\mathbf{g}\|_2^2$
  Update core tensor $\mathcal{G} \leftarrow \text{vec}^{-1}(\mathbf{g}^*)$
until convergence
return $\mathcal{G}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}$

Linear Algebra. Let $\mathbf{I}_n$ denote the $n \times n$ identity matrix and $\mathbf{0}_{m \times n}$ denote the $m \times n$ zero matrix. The transpose of $\mathbf{A} \in \mathbb{R}^{m \times n}$ is $\mathbf{A}^\intercal$, the Moore–Penrose inverse (also called pseudoinverse) is $\mathbf{A}^+$, and the spectral norm is $\|\mathbf{A}\|_2$. The singular value decomposition (SVD) of $\mathbf{A}$ is a factorization of the form $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\intercal$, where $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ are orthogonal matrices, and $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ is a non-negative real diagonal matrix. The entries $\sigma_i(\mathbf{A})$ of $\mathbf{\Sigma}$ are the singular values of $\mathbf{A}$, and the number of non-zero singular values is equal to $r = \text{rank}(\mathbf{A})$. The *compact SVD* is a related decomposition where $\mathbf{\Sigma} \in \mathbb{R}^{r \times r}$ is a diagonal matrix containing the non-zero singular values. The Kronecker product of two matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{p \times q}$ is denoted by $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{(mp) \times (nq)}$.

Tensor Products. *Fibers* of a tensor are the vectors we get by fixing all but one index. If $\mathcal{X}$ is a third-order tensor, we denote the column, row, and tube fibers by $\mathbf{x}_{:jk}$, $\mathbf{x}_{i:k}$, and $\mathbf{x}_{ij:}$, respectively. The *mode-n unfolding* of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the matrix $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times (I_1 \cdots I_{n-1} I_{n+1} \cdots I_N)}$ that arranges the mode-n fibers of $\mathcal{X}$ as columns of $\mathbf{X}_{(n)}$ ordered lexicographically by index. The *vectorization* of $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the vector $\text{vec}(\mathcal{X}) \in \mathbb{R}^{I_1 I_2 \cdots I_N}$ formed by vertically stacking

[2] The design matrix in each step of ALS for CP decomposition is a Khatri–Rao product, not a Kronecker product. CP decomposition does not suffer from a bottleneck step like ALS for Tucker decomposition since it is a sparser decomposition, i.e., CP decomposition does not have a core tensor—just factor matrices.

the entries of $\mathcal{X}$ ordered lexicographically by index. For example, this transforms $\mathbf{X} \in \mathbb{R}^{m \times n}$ into a tall vector $\text{vec}(\mathbf{X})$ by stacking its columns. We use $\text{vec}^{-1}(\mathbf{x})$ to undo this operation when it is clear from context what the shape of the output tensor should be.

The *n-mode product* of tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and matrix $\mathbf{A} \in \mathbb{R}^{J \times I_n}$ is denoted by $\mathcal{Y} = \mathcal{X} \times_n \mathbf{A}$ where $\mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}$. This operation multiplies each mode-n fiber of $\mathcal{X}$ by the matrix $\mathbf{A}$. This operation is expressed elementwise as

$$(\mathcal{X} \times_n \mathbf{A})_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \ldots i_N} a_{j i_n}.$$

The Frobenius norm $\|\mathcal{X}\|_{\mathrm{F}}$ of a tensor $\mathcal{X}$ is the square root of the sum of the squares of its entries.

Tucker Decomposition. The *Tucker decomposition* decomposes tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ into a *core tensor* $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_N}$ and N *factor matrices* $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$. Given a regularization parameter $\lambda \in \mathbb{R}_{\geq 0}$, we compute a Tucker decomposition by minimizing the nonconvex loss function

$$L(\mathcal{G}, \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}; \mathcal{X}) = \|\mathcal{X} - \mathcal{G} \times_1 \mathbf{A}^{(1)} \cdots \times_N \mathbf{A}^{(N)}\|_{\mathrm{F}}^2 + \lambda \left(\|\mathcal{G}\|_{\mathrm{F}}^2 + \sum_{n=1}^{N} \|\mathbf{A}^{(n)}\|_{\mathrm{F}}^2 \right).$$

Entries of the reconstructed tensor $\widehat{\mathcal{X}} \stackrel{\text{def}}{=} \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \cdots \times_N \mathbf{A}^{(N)}$ are

$$\widehat{x}_{i_1 i_2 \ldots i_N} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1 r_2 \ldots r_N} a_{i_1 r_1}^{(1)} \cdots a_{i_N r_N}^{(N)}. \tag{2}$$

Equation (2) demonstrates that $\widehat{\mathcal{X}}$ is the sum of $R_1 \cdots R_N$ rank-1 tensors. The tuple $(R_1, R_2, \ldots, R_N)$ is the *multilinear rank* of the decomposition. The multilinear rank is typically chosen in advance and much smaller than the dimensions of $\mathcal{X}$.

Alternating Least Squares. We present `TuckerALS` in Algorithm 1 and highlight its connections to Kronecker regression. The core tensor update (Lines 10–12) is a ridge regression problem where the design matrix $\mathbf{K}_{\text{core}} \in \mathbb{R}^{I_1 \cdots I_N \times R_1 \cdots R_N}$ is a Kronecker product of the factor matrices. Each factor matrix update (Lines 5–9) also has Kronecker product structure, but there are additional subspace constraints we must account for. We describe these constraints in more detail in Section 5.

3 Row Sampling and Approximate Regression

Here we establish our sketching toolkit. The *λ-ridge leverage score* of the i-th row of $\mathbf{A} \in \mathbb{R}^{n \times d}$ is

$$\ell_i^{\lambda}(\mathbf{A}) \stackrel{\text{def}}{=} \mathbf{a}_{i:} (\mathbf{A}^{\intercal}\mathbf{A} + \lambda \mathbf{I})^{+} \mathbf{a}_{i:}^{\intercal}. \tag{3}$$

The matrix of *cross λ-ridge leverage scores* is $\mathbf{A}(\mathbf{A}^{\intercal}\mathbf{A} + \lambda \mathbf{I})^{+}\mathbf{A}^{\intercal}$. We denote its diagonal by $\boldsymbol{\ell}^{\lambda}(\mathbf{A})$ because it contains the λ-ridge leverage scores of $\mathbf{A}$. Ridge leverage scores generalize *statistical leverage scores* in that setting $\lambda = 0$ gives the leverage scores of $\mathbf{A}$. We denote the vector of statistical leverage scores by $\boldsymbol{\ell}(\mathbf{A})$. If $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\intercal}$ is the compact SVD of $\mathbf{A}$, then for all $i \in [n]$, we have

$$\ell_i^{\lambda}(\mathbf{A}) = \sum_{k=1}^{r} \frac{\sigma_k^2(\mathbf{A})}{\sigma_k^2(\mathbf{A}) + \lambda} u_{ik}^2, \tag{4}$$

where $r = \text{rank}(\mathbf{A})$. It follows that every $\ell_i^{\lambda}(\mathbf{A}) \leq 1$ since $\mathbf{U}$ is an orthogonal matrix. We direct the reader to Alaoui and Mahoney [3] or Cohen et al. [15] for further details.

The main results in this paper build on approximate leverage score sampling for block matrices. The λ-ridge leverage scores of $\mathbf{A} \in \mathbb{R}^{n \times d}$ can be computed by appending $\sqrt{\lambda}\mathbf{I}_d$ to the bottom of $\mathbf{A}$ to get $\overline{\mathbf{A}} \in \mathbb{R}^{(n+d) \times d}$ and considering the leverage scores of $\overline{\mathbf{A}}$, so we state the following results in terms of statistical leverage scores without loss of generality.

Definition 3.1. For any $\mathbf{A} \in \mathbb{R}^{n \times d}$, the vector $\hat{\boldsymbol{\ell}}(\mathbf{A}) \in \mathbb{R}^n$ is a *β-overestimate* for the leverage score distribution of $\mathbf{A}$ if, for all $i \in [n]$, it satisfies

$$\frac{\hat{\ell}_i(\mathbf{A})}{\|\hat{\boldsymbol{\ell}}(\mathbf{A})\|_1} \geq \beta \frac{\ell_i(\mathbf{A})}{\|\boldsymbol{\ell}(\mathbf{A})\|_1} = \beta \frac{\ell_i(\mathbf{A})}{\text{rank}(\mathbf{A})}.$$

Next we describe the approximate leverage score sampling algorithm in Woodruff [64, Section 2.4]. The core idea here is that if we sample $\tilde{O}(d/\beta)$ rows and reweight them appropriately, this smaller *sketched* matrix can be used instead of $\mathbf{A}$ to give provable guarantees for many problems.

Definition 3.2 (Leverage score sampling)**.** Let $\mathbf{A} \in \mathbb{R}^{n\times d}$ and $\mathbf{p} \in [0,1]^n$ be a β-overestimate for the leverage score distribution of $\mathbf{A}$ such that $\|\mathbf{p}\|_1 = 1$. $\texttt{SampleRows}(\mathbf{A}, s, \mathbf{p})$ denotes the following procedure. Initialize sketch matrix $\mathbf{S} = \mathbf{0}_{s\times n}$. For each row i of $\mathbf{S}$, independently and with replacement, select an index $j \in [n]$ with probability p_j and set $s_{ij} = 1/\sqrt{p_j s}$. Return sketch $\mathbf{S}$.

The main result in this section is that we can choose to sketch a single block of a matrix by the leverage scores of that block in isolation. This sketched submatrix can then be used with the other (non-sketched) block to give a spectral approximation to the original matrix or for approximate linear regression. The notation $\mathbf{A} \preccurlyeq \mathbf{B}$ is the Loewner order and means $\mathbf{B} - \mathbf{A}$ is positive semidefinite.

Lemma 3.3. *Let $\mathbf{A} = [\mathbf{A}_1; \mathbf{A}_2]$ be vertically stacked with $\mathbf{A}_1 \in \mathbb{R}^{n_1\times d}$ and $\mathbf{A}_2 \in \mathbb{R}^{n_2\times d}$. Let $\mathbf{p} \in [0,1]^{n_1}$ be a β-overestimate for the leverage score distribution of $\mathbf{A}_1$. If $s > 144d\ln(2d/\delta)/(\beta\varepsilon^2)$, the sketch $\mathbf{S}$ returned by $\texttt{SampleRows}(\mathbf{A}_1, s, \mathbf{p})$ guarantees, with probability at least $1-\delta$, that*

$$(1-\varepsilon)\mathbf{A}^\intercal\mathbf{A} \preccurlyeq (\mathbf{S}\mathbf{A}_1)^\intercal\mathbf{S}\mathbf{A}_1 + \mathbf{A}_2^\intercal\mathbf{A}_2 \preccurlyeq (1+\varepsilon)\mathbf{A}^\intercal\mathbf{A}.$$

Lemma 3.4 (Approximate block regression)**.** *Consider the problem $\arg\min_{\mathbf{x}\in\mathbb{R}^d}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2$ where $\mathbf{A} = [\mathbf{A}_1; \mathbf{A}_2]$ and $\mathbf{b} = [\mathbf{b}_1; \mathbf{b}_2]$ are vertically stacked and $\mathbf{A}_1 \in \mathbb{R}^{n_1\times d}$, $\mathbf{A}_2 \in \mathbb{R}^{n_2\times d}$, $\mathbf{b}_1 \in \mathbb{R}^{n_1}$, $\mathbf{b}_2 \in \mathbb{R}^{n_2}$. Let $\mathbf{p} \in [0,1]^{n_1}$ be a β-overestimate for the leverage score distribution of $\mathbf{A}_1$. Let $s \geq 1680d\ln(40d)/(\beta\varepsilon)$ and let $\mathbf{S}$ be the output of $\texttt{SampleRows}(\mathbf{A}_1, s, \mathbf{p})$. If*

$$\tilde{\mathbf{x}}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^d}\Big(\|\mathbf{S}(\mathbf{A}_1\mathbf{x}-\mathbf{b}_1)\|_2^2 + \|\mathbf{A}_2\mathbf{x}-\mathbf{b}_2\|_2^2\Big),$$

then, with probability at least $9/10$, we have

$$\|\mathbf{A}\tilde{\mathbf{x}}^* - \mathbf{b}\|_2^2 \leq (1+\varepsilon)\min_{\mathbf{x}\in\mathbb{R}^d}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2.$$

We defer the proofs of these results to Appendix A. The key idea behind Lemma 3.4 is that leverage scores do not increase if rows are appended to the matrix. This then allows us to prove a sketched submatrix version of Drineas et al. [19, Lemma 8] for approximate matrix multiplication and satisfy the structural conditions for approximate least squares in Drineas et al. [20]. One consequence is that we can "sketch and solve" ridge regression, which was shown in [63, Theorem 1] and [6, Theorem 2].

Corollary 3.5. *For any $\mathbf{A} \in \mathbb{R}^{n\times d}$, $\mathbf{b} \in \mathbb{R}^d$, $\lambda \geq 0$, consider*

$$\arg\min_{\mathbf{x}\in\mathbb{R}^d}(\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_2^2).$$

Let $\mathbf{p} \in [0,1]^{n_1}$ be a β-overestimate for the leverage scores of $\mathbf{A}$ and $s \geq 1680d\ln(40d)/(\beta\varepsilon)$. If $\mathbf{S}$ is the output of $\texttt{SampleRows}(\mathbf{A}, s, \mathbf{p})$, then, with probability at least $9/10$, the sketched solution

$$\tilde{\mathbf{x}}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^d}(\|\mathbf{S}(\mathbf{A}\mathbf{x}-\mathbf{b})\|_2^2 + \lambda\|\mathbf{x}\|_2^2)$$

gives a $(1+\varepsilon)$-approximation to the original problem.

Remark 3.6. The success probability of the sketch can be boosted from $9/10$ to $1-\delta$ by sampling a factor of $O(\log(1/\delta))$ more rows. See the discussion in Chen and Price [12, Section 2] about matrix concentration bounds for more details.

4 Kronecker Regression

Now we describe the key ingredients that allow us to design an approximate Kronecker regression algorithm whose running time is *subquadratic* in the number of columns in the design matrix.

1. The leverage score distribution of a Kronecker product matrix $\mathbf{K} = \mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(N)}$ is a *product distribution* of the leverage score distributions of its factor matrices. Therefore, we can sample rows of $\mathbf{K}$ from $\ell(\mathbf{K})$ with replacement in $\tilde{O}(N)$ time after a preprocessing step.

2. The normal matrix $\mathbf{K}^\intercal\mathbf{K} + \lambda\mathbf{I}$ in the ridge regression problem $\min_{\mathbf{x}} \|\mathbf{K}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_2^2$ is a $O(1)$-spectral approximation of the sketched matrix $(\mathbf{S}\mathbf{K})^\intercal\mathbf{S}\mathbf{K} + \lambda\mathbf{I}$ by Lemma 3.3. Thus we can use Richardson iteration with $(\mathbf{K}^\intercal\mathbf{K} + \lambda\mathbf{I})^+$ as the preconditioner to *solve the sketched instance*, which guarantees a $(1+\varepsilon)$-approximation. Using $(\mathbf{K}^\intercal\mathbf{K} + \lambda\mathbf{I})^+$ as the preconditioner allows us to *heavily exploit the Kronecker structure* with fast matrix-vector multiplications.
3. At this point, *Kronecker matrix-vector multiplications* are still the bottleneck, so we partition the factor matrices into two groups by their number of columns and use our novel way of multiplying sparsified Kronecker product matrices as well as fast rectangular matrix multiplication to get a subquadratic running time.

This first result shows how λ-ridge leverage scores of a Kronecker product matrix decompose according to the SVDs of its factor matrices. All missing proofs in this section are deferred to Appendix B.

Lemma 4.1. *Let $\mathbf{K} = \mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)} \otimes \cdots \otimes \mathbf{A}^{(N)}$, where each factor matrix $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$. Let $(i_1, i_2, \ldots, i_N)$ be the natural row indexing of $\mathbf{K}$ by its factors. Let the factor SVDs be $\mathbf{A}^{(n)} = \mathbf{U}^{(n)}\mathbf{\Sigma}^{(n)}\mathbf{V}^{(n)\intercal}$. For any $\lambda \geq 0$, the λ-ridge leverage scores of $\mathbf{K}$ are*

$$\ell^{\lambda}_{(i_1,\ldots,i_N)}(\mathbf{K}) = \sum_{\mathbf{t}\in T} \frac{\prod_{n=1}^{N} \sigma^2_{t_n}(\mathbf{A}^{(n)})}{\prod_{n=1}^{N} \sigma^2_{t_n}(\mathbf{A}^{(n)}) + \lambda} \left(\prod_{n=1}^{N} u^{(n)}_{i_n t_n}\right)^2, \tag{5}$$

where the sum is over $T = [R_1] \times [R_2] \times \cdots \times [R_N]$. For statistical leverage scores, this simplifies to $\ell_{(i_1,\ldots,i_N)}(\mathbf{K}) = \prod_{n=1}^{N} \ell_{i_n}(\mathbf{A}^{(n)})$.

This proof repeatedly uses the mixed-product property for Kronecker products and the definition of λ-ridge leverage scores in Equation (3).

4.1 Iterative Methods

Now we state a result for the convergence rate of preconditioned Richardson iteration [57], which uses the notation $\|\mathbf{x}\|^2_{\mathbf{M}} = \mathbf{x}^\intercal\mathbf{M}\mathbf{x}$.

Lemma 4.2 (Preconditioned Richardson iteration)**.** *Let $\mathbf{M}$ be any matrix such that $\mathbf{A}^\intercal\mathbf{A} \preccurlyeq \mathbf{M} \preccurlyeq \kappa \cdot \mathbf{A}^\intercal\mathbf{A}$ for some $\kappa \geq 1$. Let $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \mathbf{M}^+(\mathbf{A}^\intercal\mathbf{A}\mathbf{x}^{(k)} - \mathbf{A}^\intercal\mathbf{b})$. Then,*

$$\|\mathbf{x}^{(k)} - \mathbf{x}^*\|_{\mathbf{M}} \leq (1 - 1/\kappa)^k \|\mathbf{x}^{(0)} - \mathbf{x}^*\|_{\mathbf{M}},$$

where $\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^d} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$.

Remark 4.3. The ridge regression algorithm in Chowdhury et al. [14] is also based on sketching and preconditioned Richardson iteration. They consider short and wide matrices where $d \gg n$ and use the *sketched normal matrix as the preconditioner* to solve the original problem. One of our main technical contributions is to use the *original normal matrix as the preconditioner* to solve the sketched problem. Reversing this is advantageous because computing the pseudoinverse and matrix-vector products with the original Kronecker matrix is substantially less expensive due to its Kronecker structure. However, this still motivates the need for faster Kronecker matrix-vector multiplications.

4.2 Fast Kronecker-Matrix Multiplication

The next result is a simple but useful observation about extracting the rightmost factor matrix from the Kronecker product and recursively computing a new less expensive Kronecker-matrix multiplication.

Lemma 4.4. *Let $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times J_n}$, for $n \in [N]$, and $\mathbf{B} \in \mathbb{R}^{J_1 \cdots J_N \times K}$. There is an algorithm* `KronMatMul`$([\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}], \mathbf{B})$ *that computes $(\mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)} \otimes \cdots \otimes \mathbf{A}^{(N)})\mathbf{B} \in \mathbb{R}^{(I_1 \cdots I_N) \times K}$ in $O(K \sum_{n=1}^{N} J_1 \cdots J_n I_n \cdots I_N)$ time.*

The following theorem is more sophisticated. We write the statement in terms of rectangular matrix multiplication time $\mathrm{MM}(a, b, c)$, which is the time to multiply an $a \times b$ matrix by a $b \times c$ matrix.

Theorem 4.5. *Let $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, for $n \in [N]$, $I = I_1 \cdots I_N$, $R = R_1 \cdots R_N$, $\mathbf{b} \in \mathbb{R}^I$, $\mathbf{c} \in \mathbb{R}^R$, and $\mathbf{S} \in \mathbb{R}^{I \times I}$ be a diagonal matrix with $\tilde{O}(R\varepsilon^{-1})$ nonzeros. The vectors*

$$(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_N)^\intercal\mathbf{S}\mathbf{b} \quad \textit{and} \quad \mathbf{S}(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_N)\mathbf{c}$$

can be computed in time $\tilde{O}(\min_{T\subseteq[N]} \mathrm{MM}(\prod_{n\in T} R_n, R\varepsilon^{-1}, \prod_{n\notin T} R_n))$.

Algorithm 2 `FastKroneckerRegression`

Input: Factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, response vector $\mathbf{b} \in \mathbb{R}^{I_1 \cdots I_N}$, L2 regularization strength λ, error ε, failure probability δ

Set $R \leftarrow R_1 R_2 \cdots R_N$
for $n = 1$ to N **do**
    Compute a spectral approximation $\tilde{\mathbf{A}}^{(n)}$ with $\tilde{O}(R_n N^2 \varepsilon^{-2})$ rows by Lemma 3.3 such that

$$\mathbf{A}^{(n)\intercal}\mathbf{A}^{(n)} \preccurlyeq \tilde{\mathbf{A}}^{(n)\intercal}\tilde{\mathbf{A}}^{(n)} \preccurlyeq (1 + \log(1+\varepsilon/4)/N)\mathbf{A}^{(n)\intercal}\mathbf{A}^{(n)} \tag{6}$$

    Compute $\tilde{\mathbf{A}}^{(n)\intercal}\tilde{\mathbf{A}}^{(n)}$ and the SVD of $\tilde{\mathbf{A}}^{(n)\intercal}\tilde{\mathbf{A}}^{(n)} = \mathbf{V}^{(n)}(\mathbf{\Sigma}^{(n)\intercal}\mathbf{\Sigma}^{(n)})\mathbf{V}^{(n)\intercal}$
    Compute $(1 + \log(1+\varepsilon/2)/N)$-approximate leverage scores $\boldsymbol{\ell}(\mathbf{A}^{(n)})$ using Lemma B.4 by applying a random Johnson–Lindenstrauss projection
Initialize product distribution data structure $\mathcal{P}$ to sample indices from $(\boldsymbol{\ell}(\mathbf{A}^{(1)}), \cdots, \boldsymbol{\ell}(\mathbf{A}^{(N)}))$
Set $\mathbf{D} \leftarrow (\mathbf{\Sigma}^{(1)\intercal}\mathbf{\Sigma}^{(1)} \otimes \cdots \otimes \mathbf{\Sigma}^{(N)\intercal}\mathbf{\Sigma}^{(N)} + \lambda \mathbf{I}_R)^+$
Let $\mathbf{M}^+ = (\mathbf{V}^{(1)} \otimes \cdots \otimes \mathbf{V}^{(N)})\mathbf{D}(\mathbf{V}^{(1)} \otimes \cdots \otimes \mathbf{V}^{(N)})^\intercal$
Set $s \leftarrow \lceil 1680 R \ln(40R) \ln(1/\delta)/\varepsilon \rceil$
Set $\mathbf{S} \leftarrow \texttt{SampleRows}(\mathbf{K}, s, \mathcal{P})$
Let $\tilde{\mathbf{K}} = \mathbf{S}\mathbf{K}$ and $\tilde{\mathbf{b}} = \mathbf{S}\mathbf{b}$
Initialize $\mathbf{x} \leftarrow \mathbf{0}_R$
repeat
    $\mathbf{x} \leftarrow \mathbf{x} - (1 - \sqrt{\varepsilon})\mathbf{M}^+(\tilde{\mathbf{K}}^\intercal\tilde{\mathbf{K}}\mathbf{x} + \lambda\mathbf{x} - \tilde{\mathbf{K}}^\intercal\tilde{\mathbf{b}})$ using fast Kronecker-matrix multiplication
until convergence
return $\mathbf{x}$

The core idea behind Theorem 4.5 is that the factor matrices can be partitioned into two groups to achieve a good "column-product" balance, i.e., $\min_{T \subseteq [N]} \max\{\prod_{n \in T} R_n, \prod_{n \notin T} R_n\}$ is close to $\sqrt{R}$. Then we use the fact that $\text{nnz}(\mathbf{S}) = \tilde{O}(R\varepsilon^{-1})$ with a sparsity-aware `KronMatMul` to solve each part of this partition separately, and combine them with fast rectangular matrix multiplication. If we achieve perfect balance, the running time is $\tilde{O}(R^{1.626}\varepsilon^{-1})$ using results of Gall and Urrutia [24], which are explained in detail in van den Brand and Nanongkai [62, Appendix C]. If one of these two factor matrix groups has at most 0.9 of the "column-product mass," the running time is $\tilde{O}(R^{1.9}\varepsilon^{-1})$.

4.3 Main Algorithm

We are now ready to present our main algorithm for solving approximate Kronecker regression.

Theorem 4.6. *For any Kronecker product matrix* $\mathbf{K} = \mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(N)} \in \mathbb{R}^{I_1 \cdots I_N \times R_1 \cdots R_N}$, $\mathbf{b} \in \mathbb{R}^{I_1 \cdots I_N}$, $\lambda \geq 0$, $\varepsilon \in (0, 1/4]$, *and* $\delta > 0$, `FastKroneckerRegression` *returns* $\mathbf{x}^* \in \mathbb{R}^{R_1 \cdots R_N}$ *in*

$$\tilde{O}\Big(\textstyle\sum_{n=1}^N \big(\text{nnz}(\mathbf{A}^{(n)}) + R_n^\omega N^2 \varepsilon^{-2}\big) + \min_{S \subseteq [N]} \text{MM}\Big(\prod_{n \in S} R_n, R\varepsilon^{-1}, \prod_{n \in [N] \setminus S} R_n\Big)\Big),$$

time such that, with probability at least $1 - \delta$,

$$\|\mathbf{K}\mathbf{x}^* - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_2^2 \leq (1+\varepsilon) \min_{\mathbf{x}} \|\mathbf{K}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_2^2.$$

We defer the proof to Appendix B.2 and sketch how the ideas in Algorithm 2 come together. First, we do not compute the pseudoinverse $\tilde{\mathbf{K}}^+$ but instead use iterative Richardson iteration (Lemma 4.2), which allows us avoid a $\tilde{O}(R^\omega\varepsilon^{-1})$ running time. This technique by itself, however, only allows us to reduce the running time to $\tilde{O}(R^2\varepsilon^{-1})$ since all of the matrix-vector products (e.g., $\tilde{\mathbf{K}}^\intercal\tilde{\mathbf{b}}$, $\tilde{\mathbf{K}}\mathbf{x}$, and multiplication against $\mathbf{M}^+$) naively take $\Omega(R^2)$ time. To achieve subquadratic time, we need three more ideas: (1) compute an approximate SVD of each Gram matrix $\mathbf{A}^{(n)\intercal}\mathbf{A}^{(n)}$ in order to construct the decomposed preconditioner $\mathbf{M}^+$; (2) use fast Kronecker-vector multiplication (e.g., Lemma 4.4) to exploit the Kronecker structure of the decomposed preconditioner; (3) noting that Lemma 4.4 for the Kronecker-vector products $\tilde{\mathbf{K}}^\intercal\tilde{\mathbf{b}}$ and $\tilde{\mathbf{K}}^\intercal(\tilde{\mathbf{K}}\mathbf{x})$ is insufficient because the intermediate vectors can be large, we develop a novel multiplication algorithm in Theorem 4.5 that fully exploits the sparsity, Kronecker structure, and fast rectangular matrix multiplication of Gall and Urrutia [24].

5 Applications to Low-Rank Tucker Decomposition

Now we apply our fast Kronecker regression algorithm to `TuckerALS` and prove Theorem 1.2. We list the running times of different factor matrix and core update algorithms in Table 1, and we analyze these subroutines in Appendix C.3.

Core Tensor Update. The core update running time in Theorem 1.2 is a direct consequence of our algorithm for fast Kronecker regression in Theorem 4.6. The only difference is that we avoid recomputing the SVD and Gram matrix of each factor since these are computed at the end of each factor matrix update and stored for future use.

Factor Matrix Update. The factor matrix updates require more work because of the $\mathbf{G}_{(n)}^{\intercal}\mathbf{y}$ term in Line 8 of `TuckerALS`. To overcome this, we substitute variables and recast each factor update as an equality-constrained Kronecker regression problem with an appended low-rank block to account for the L2 regularization of the original variables. To support this new low-rank block, we use the *Woodbury matrix identity* to extend the technique of using Richardson iterations with fast Kronecker matrix-vector multiplication for solving sketched regression instances.

The next result formalizes this substitution and reduces the problem to block Kronecker regression with a subspace constraint. This result relies on the fact that the least squares solution to $\|\mathbf{M}\mathbf{x}-\mathbf{z}\|_2^2$ with minimum norm is $\mathbf{M}^+\mathbf{z}$.

Lemma 5.1. *Let $\mathbf{A}\in\mathbb{R}^{n\times m}$, $\mathbf{M}\in\mathbb{R}^{m\times d}$, $\mathbf{b}\in\mathbb{R}^n$, and $\lambda\geq 0$. For any ridge regression problem of the form $\arg\min_{\mathbf{x}\in\mathbb{R}^d}(\|\mathbf{A}\mathbf{M}\mathbf{x}-\mathbf{b}\|_2^2+\lambda\|\mathbf{x}\|_2^2)$, we can solve*

$$\mathbf{z}_{\text{opt}}=\operatorname*{arg\,min}_{\mathbf{N}\mathbf{z}=\mathbf{0}}\|\mathbf{A}\mathbf{z}-\mathbf{b}\|_2^2+\lambda\left\|\mathbf{M}^+\mathbf{z}\right\|_2^2,$$

where $\mathbf{N}=\mathbf{I}_m-\mathbf{M}\mathbf{M}^+$, and return vector $\mathbf{M}^+\mathbf{z}_{\text{opt}}$ instead.

Proof. Let $\mathbf{z}=\mathbf{M}\mathbf{x}\in\mathbb{R}^m$. For any $\mathbf{x}\in\mathbb{R}^d$, $\mathbf{z}$ is in the column space of $\mathbf{M}$ and hence orthogonal to any vector in the left null space of $\mathbf{M}$. Therefore, we can optimize over $\mathbf{z}\in\mathbb{R}^m$ subject to $\mathbf{N}\mathbf{z}=\mathbf{0}$ instead because for any $\mathbf{x}\in\mathbb{R}^d$, $\mathbf{N}\mathbf{M}\mathbf{x}=(\mathbf{I}_m-\mathbf{M}\mathbf{M}^+)\mathbf{M}\mathbf{x}=(\mathbf{M}-\mathbf{M})\mathbf{x}=\mathbf{0}$. Using this substitution, we can also replace the term $\lambda\|\mathbf{x}\|_2^2$ by $\lambda\|\mathbf{M}^+\mathbf{z}\|_2^2$ because for any $\mathbf{z}$, the least squares solution to $\mathbf{z}=\mathbf{M}\mathbf{x}$ with minimum norm is $\mathbf{M}^+\mathbf{z}$ [54]. □

To solve this constrained regression problem, we can add a scaled version of the constraint matrix $\mathbf{N}$ as a block to the approximate regression problem and take the projection of the resulting solution.

Lemma 5.2 (Approximate equality-constrained regression)**.** *Let $\mathbf{M}\in\mathbb{R}^{n\times d}$, $\mathbf{N}\in\mathbb{R}^{m\times d}$, $\mathbf{b}\in\mathbb{R}^n$, and $0<\varepsilon<1/3$. To solve $\min_{\mathbf{N}\mathbf{x}=\mathbf{0}}\|\mathbf{M}\mathbf{x}-\mathbf{b}\|_2^2$ to a $(1+\varepsilon)$-approximation, it suffices to solve*

$$\min_{\mathbf{x}\in\mathbb{R}^d}\left\|\begin{bmatrix}\mathbf{M}\\ \sqrt{w}\mathbf{N}\end{bmatrix}\mathbf{x}-\begin{bmatrix}\mathbf{b}\\ \mathbf{0}\end{bmatrix}\right\|_2^2$$

to a $(1+\varepsilon/3)$-approximation with $w\geq(1+12/\varepsilon)\|\mathbf{M}\mathbf{N}^+\|_2^2$.

Letting $\mathbf{z}=\mathbf{G}_{(n)}^{\intercal}\mathbf{y}$ in Line 8 of `TuckerALS` and modifying `FastKroneckerRegression` to support additional low-rank updates to the preconditioner, we get the `FastFactorMatrixUpdate` algorithm, presented as Algorithm 3 in Appendix C.2. The analysis is similar to the proofs of Theorem 4.6. The factor matrix updates benefit in the same way as before from fast Kronecker matrix-vector products, and new low-rank block updates are supported via the Woodbury identity. We defer the proofs of the next two results to Appendix C.

Theorem 5.3. *For any $\lambda\geq 0$, $\varepsilon\in(0,1/3)$, and $\delta>0$, the* `FastFactorMatrixUpdate` *algorithm updates $\mathbf{A}_{(k)}\in\mathbb{R}^{I_k\times R_k}$ in* `TuckerALS` *with a $(1+\varepsilon)$-approximation, with probability at least $1-\delta$, in time*

$$\tilde{O}\Big(I_kR_{\neq k}^2\varepsilon^{-1}\log(1/\delta)+I_kR\textstyle\sum_{n=1}^N R_n+R_k^{\omega}\varepsilon^{-2}\Big).$$

Corollary 5.4. `FastFactorMatrixUpdate` *updates $\mathbf{A}^{(k)}\in\mathbb{R}^{I_k\times R_k}$ in $\tilde{O}(I_kR_{\neq k}^{2-\theta^*}\varepsilon^{-1}\log(1/\delta)+I_kR\sum_{n=1}^N R_n+R_k^{\omega}\varepsilon^{-2})$ time, where $\theta^*>0$ is the optimally balanced* MM *exponent in Theorem 4.5.*

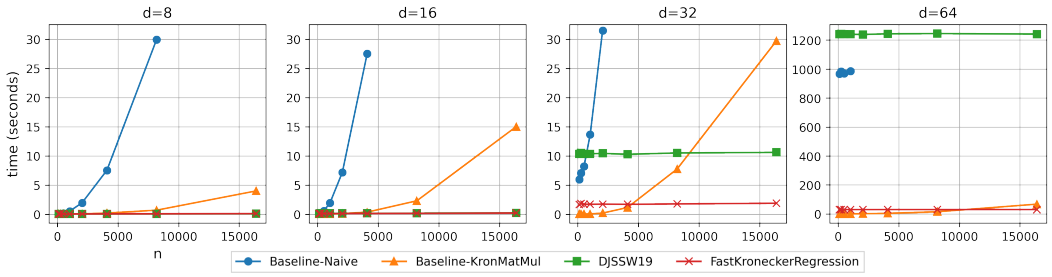

Figure 1: Running times of Kronecker regression algorithms with a design matrix of size $n^2 \times d^2$.

6 Experiments

All experiments were run using NumPy [26] with an Intel Xeon W-2135 processor (8.25MB cache, 3.70 GHz) and 128GB of RAM. The `FastKroneckerRegression`-based ALS experiments for low-rank Tucker decomposition on image tensors are deferred to Appendix D.2. All of our code is available at `https://github.com/fahrbach/subquadratic-kronecker-regression`.

Kronecker regression. We build on the numerical experiments in [17, 18] for Kronecker regression that use two random factor matrices. We generate matrices $\mathbf{A}^{(1)}, \mathbf{A}^{(2)} \in \mathbb{R}^{n \times d}$ where each entry is drawn i.i.d. from the normal distribution $\mathcal{N}(1, 0.001)$ and compare several algorithms for solving $\min_{\mathbf{x}} \|(\mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)})\mathbf{x} - \mathbf{1}_{n^2}\|_2^2 + \lambda\|\mathbf{x}\|_2^2$ as we increase n, d. The running times are plotted in Figure 1.

The algorithms we compare are: (1) a baseline that solves the normal equation $(\mathbf{K}^\intercal\mathbf{K} + \lambda\mathbf{I})^+\mathbf{K}^\intercal\mathbf{b}$ and fully exploits the Kronecker structure of $\mathbf{K}^\intercal\mathbf{K}$ before calling `np.linalg.pinv()`; (2) an enhanced baseline that combines the SVDs of $\mathbf{A}^{(n)}$ with Lemma 4.4, e.g., `KronMatMul`$([(\mathbf{U}^{(1)})^\intercal, (\mathbf{U}^{(2)})^\intercal], \mathbf{b})$, using only Kronecker-vector products; (3) the sketching algorithm of Diao et al. [18, Algorithm 1]; and (4) our `FastKroneckerRegression` algorithm in Algorithm 2. For both sketching algorithms, we use $\varepsilon = 0.1$ and $\delta = 0.01$. We reduce the number of row samples in both algorithms by $\alpha = 10^{-5}$ so that the algorithms are more practical and comparable to the earlier experiments in [17, 18]. Lastly, we set $\lambda = 10^{-3}$. We discuss additional parameter choice details and the full results in Appendix D.1.

The running times in Figure 1 demonstrate several different behaviors. The naive baseline quickly becomes impractical for moderately large values of n or d. `KronMatMul` is competitive for $n \leq 10^4$, especially since it is an exact method. The runtimes of the sketching algorithms are nearly-independent of n. Diao et al. [18] works well for small d, but deteriorates tremendously as d grows because it computes $((\mathbf{SK})^\intercal\mathbf{SK} + \lambda\mathbf{I})^+ \in \mathbb{R}^{d^2 \times d^2}$ and cannot exploit the Kronecker structure of $\mathbf{K}$, which takes $O(d^6)$ time. `FastKroneckerRegression`, on the other hand, runs in $O(d^4)$ time because it uses quadratic-time Kronecker-vector products in each Richardson iteration step (Line 14).

Table 2: Kronecker regression losses for $d = 64$. OPT denotes the loss of the `KronMatMul` algorithm, `DJSSW19` is Diao et al. [18, Algorithm 1], and Algorithm 2 is `FastKroneckerRegression`. We also record the relative error of each algorithm and the number of rows sampled from $\mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)}$.

n	OPT	Algorithm 2	Approx	`DJSSW19`	Approx	Rows sampled (%)
1024	0.031	0.032	1.051	0.035	1.138	0.0370
2048	0.123	0.126	1.026	1.577	12.792	0.0093
4096	0.507	0.520	1.026	275.566	543.776	0.0023
8192	2.073	2.136	1.030	333.430	160.809	0.0006
16384	8.238	8.608	1.045	546391.728	66329.791	0.0001

These experiments also show that combining sketching with iterative methods can give better *sketch efficiency*. Table 2 compares the loss of [18, Algorithm 1] and `FastKroneckerRegression` to an exact baseline OPT for $d = 64$. Both algorithms use the exact same sketch $\mathbf{SK}$ for each value of n. Our algorithm uses the original $(\mathbf{K}^\intercal\mathbf{K} + \lambda\mathbf{I})^+$ as a preconditioner to solve the sketched problem, whereas Diao et al. [18, Algorithm 1] computes $((\mathbf{SK})^\intercal\mathbf{SK} + \lambda\mathbf{I})^+(\mathbf{SK})^\intercal\mathbf{Sb}$ exactly and becomes numerically unstable for $n \geq 2048$ when $d \in \{16, 32, 64\}$. This raises the question about how to combine sketched information with the original data to achieve more efficient algorithms, even when solving sketched instances. We leave this question of sketch efficiency as an interesting future work.

References

[1] Evrim Acar, Daniel M Dunlavy, Tamara G Kolda, and Morten Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, 2011.

[2] Salman Ahmadi-Asl, Stanislav Abukhovich, Maame G Asante-Mensah, Andrzej Cichocki, Anh Huy Phan, Tohishisa Tanaka, and Ivan Oseledets. Randomized algorithms for computation of tucker decomposition and higher order svd (hosvd). *IEEE Access*, 9:28684–28706, 2021.

[3] Ahmed Alaoui and Michael W. Mahoney. Fast randomized kernel ridge regression with statistical guarantees. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[4] Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539. SIAM, 2021.

[5] Rosa I Arriaga and Santosh Vempala. An algorithmic theory of learning: Robust concepts and random projection. *Machine learning*, 63(2):161–182, 2006.

[6] Haim Avron, Kenneth L. Clarkson, and David P. Woodruff. Sharper Bounds for Regularized Data Fitting. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2017)*, volume 81, pages 27:1–27:22, 2017.

[7] Brett W. Bader and Tamara G. Kolda. Tensor toolbox for MATLAB, version 3.2.1. `https://www.tensortoolbox.org/`, 2021.

[8] Raghavendran Balu and Teddy Furon. Differentially private matrix factorization using sketching techniques. In *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security*, pages 57–62, 2016.

[9] Casey Battaglino, Grey Ballard, and Tamara G Kolda. A practical randomized cp tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 39(2):876–901, 2018.

[10] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3), 2011.

[11] Maolin Che and Yimin Wei. Randomized algorithms for the approximations of tucker and the tensor train decompositions. *Advances in Computational Mathematics*, 45(1):395–428, 2019.

[12] Xue Chen and Eric Price. Active regression via linear-sample sparsification. In *Conference on Learning Theory*, pages 663–695. PMLR, 2019.

[13] Dehua Cheng, Richard Peng, Yan Liu, and Ioakeim Perros. SPALS: Fast alternating least squares via implicit leverage scores sampling. *Advances in Neural Information Processing Systems*, 29:721–729, 2016.

[14] Agniva Chowdhury, Jiasen Yang, and Petros Drineas. An iterative, sketching-based framework for ridge regression. In *International Conference on Machine Learning*, pages 989–998. PMLR, 2018.

[15] Michael B Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 181–190, 2015.

[16] Michael B Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1758–1777. SIAM, 2017.

[17] Huaian Diao, Zhao Song, Wen Sun, and David Woodruff. Sketching for kronecker product regression and p-splines. In *International Conference on Artificial Intelligence and Statistics*, pages 1299–1308. PMLR, 2018.

[18] Huaian Diao, Rajesh Jayaram, Zhao Song, Wen Sun, and David Woodruff. Optimal sketching for kronecker product regression and low rank approximation. *Advances in neural information processing systems*, 32, 2019.

[19] Petros Drineas, Ravi Kannan, and Michael W Mahoney. Fast Monte Carlo algorithms for matrices I: Approximating matrix multiplication. *SIAM Journal on Computing*, 36(1):132–157, 2006.

[20] Petros Drineas, Michael W. Mahoney, Shan Muthukrishnan, and Tamás Sarlós. Faster least squares approximation. *Numerische Mathematik*, 117(2):219–249, 2011.

[21] Matthew Fahrbach, Mehrdad Ghadiri, and Thomas Fu. Fast low-rank tensor decomposition by ridge leverage score sampling. *arXiv preprint arXiv:2107.10654*, 2021.

[22] Marko Filipović and Ante Jukić. Tucker factorization with missing data with application to low-n-rank tensor completion. *Multidimensional systems and signal processing*, 26(3):677–692, 2015.

[23] Abraham Frandsen and Rong Ge. Optimization landscape of tucker decomposition. *Mathematical Programming*, pages 1–26, 2020.

[24] François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1029–1046. SIAM, 2018.

[25] Lars Grasedyck, Melanie Kluge, and Sebastian Kramer. Variants of alternating least squares tensor completion in the tensor train format. *SIAM Journal on Scientific Computing*, 37(5): A2424–A2450, 2015.

[26] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

[27] Christopher J Hillar and Lek-Heng Lim. Most tensor problems are NP-hard. *Journal of the ACM (JACM)*, 60(6):1–39, 2013.

[28] Prateek Jain and Sewoong Oh. Provable tensor factorization with missing data. *Advances in Neural Information Processing Systems*, 27, 2014.

[29] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674, 2013.

[30] M James. The generalised inverse. *The Mathematical Gazette*, 62(420):109–114, 1978.

[31] Jun-Gi Jang and U Kang. Fast and memory-efficient tucker decomposition for answering diverse time range queries. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 725–735, 2021.

[32] William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space 26. *Contemporary mathematics*, 26, 1984.

[33] Praneeth Kacham and David Woodruff. Sketching algorithms and lower bounds for ridge regression. In *International Conference on Machine Learning*, pages 10539–10556. PMLR, 2022.

[34] Hiroyuki Kasai and Bamdev Mishra. Low-rank tensor completion: a Riemannian manifold preconditioning approach. In *International Conference on Machine Learning*, pages 1012–1021. PMLR, 2016.

[35] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[36] Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. Tensorly: Tensor learning in Python. *Journal of Machine Learning Research (JMLR)*, 20(26), 2019.

[37] Daniel Kressner, Michael Steinlechner, and Bart Vandereycken. Low-rank tensor completion by Riemannian optimization. *BIT Numerical Mathematics*, 54(2):447–468, 2014.

[38] Brett W Larsen and Tamara G Kolda. Practical leverage-based sampling for low-rank tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 43(3):1488–1517, 2022.

[39] Brett W. Larsen and Tamara G. Kolda. Sketching matrix least squares via leverage scores estimates. *arXiv preprint arXiv:2201.10638*, 2022.

[40] Mu Li, Gary L Miller, and Richard Peng. Iterative row sampling. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 127–136. IEEE, 2013.

[41] Zhu Li, Jean-Francois Ton, Dino Oglic, and Dino Sejdinovic. Towards a unified analysis of random Fourier features. In *International Conference on Machine Learning*, pages 3905–3914. PMLR, 2019.

[42] Allen Liu and Ankur Moitra. Tensor completion made practical. In *Advances in Neural Information Processing Systems*, volume 33, pages 18905–18916, 2020.

[43] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. Tensor completion for estimating missing values in visual data. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):208–220, 2012.

[44] Linjian Ma and Edgar Solomonik. Fast and accurate randomized algorithms for low-rank tensor decompositions. *Advances in Neural Information Processing Systems*, 34, 2021.

[45] Osman Asif Malik and Stephen Becker. Low-rank Tucker decomposition of large tensors using TensorSketch. *Advances in Neural Information Processing Systems*, 31:10096–10106, 2018.

[46] Osman Asif Malik and Stephen Becker. A sampling-based method for tensor ring decomposition. In *International Conference on Machine Learning*, pages 7400–7411. PMLR, 2021.

[47] Ana Marco, José-Javier Martínez, and Raquel Viaña. Least squares problems involving generalized kronecker products and application to bivariate polynomial regression. *Numerical Algorithms*, 82(1):21–39, 2019.

[48] Shannon McCurdy. Ridge regression and provable deterministic ridge leverage score sampling. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[49] Cameron Musco and Christopher Musco. Recursive sampling for the Nyström method. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[50] Sérgio MC Nascimento, Kinjiro Amano, and David H Foster. Spatial distributions of local illumination color in natural scenes. *Vision research*, 120:39–44, 2016.

[51] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-20). 1996.

[52] Madhav Nimishakavi, Pratik Kumar Jawanpuria, and Bamdev Mishra. A dual framework for low-rank tensor completion. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[53] Rasmus Pagh. Compressed matrix multiplication. *ACM Transactions on Computation Theory (TOCT)*, 5(3):1–17, 2013.

[54] M. Planitz. Inconsistent systems of linear equations. *The Mathematical Gazette*, 63(425): 181–185, 1979.

[55] Stephan Rabanser, Oleksandr Shchur, and Stephan Günnemann. Introduction to tensor decompositions and their applications in machine learning. *arXiv preprint arXiv:1711.10781*, 2017.

[56] Aravind Reddy, Zhao Song, and Lichen Zhang. Dynamic tensor product regression. *arXiv preprint arXiv:2210.03961*, 2022.

[57] Yousef Saad. *Iterative methods for sparse linear systems*. SIAM, 2003.

[58] Nicholas D Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017.

[59] Zhao Song, David P Woodruff, and Peilin Zhong. Relative error tensor low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2772–2789. SIAM, 2019.

[60] Yiming Sun, Yang Guo, Charlene Luo, Joel Tropp, and Madeleine Udell. Low-rank tucker approximation of a tensor from streaming data. *SIAM Journal on Mathematics of Data Science*, 2(4):1123–1150, 2020.

[61] Abraham Traore, Maxime Berar, and Alain Rakotomamonjy. Singleshot : a scalable tucker tensor decomposition. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[62] Jan van den Brand and Danupon Nanongkai. Dynamic approximate shortest paths and beyond: Subquadratic and worst-case update time. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 436–455. IEEE, 2019.

[63] Shusen Wang, Alex Gittens, and Michael W Mahoney. Sketched ridge regression: Optimization perspective, statistical perspective, and model averaging. In *International Conference on Machine Learning*, pages 3608–3616. PMLR, 2017.

[64] David P. Woodruff. *Sketching as a Tool for Numerical Linear Algebra*. 2014.

[65] Rose Yu and Yan Liu. Learning from multiway data: Simple and efficient tensor regression. In *International Conference on Machine Learning*, pages 373–381. PMLR, 2016.

[66] Huamin Zhang and Feng Ding. On the kronecker products and their applications. *Journal of Applied Mathematics*, 2013, 2013.

[67] Guoxu Zhou, Andrzej Cichocki, and Shengli Xie. Decomposition of big tensors with low multilinear rank. *arXiv preprint arXiv:1412.1885*, 2014.

[68] Hua Zhou, Lexin Li, and Hongtu Zhu. Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, 108(502):540–552, 2013.

A Missing Analysis from Section 3

Here we show how to use leverage scores of the design matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ to create a smaller least squares problem whose solution vector gives a $(1+\varepsilon)$-approximation to the original regression problem. Our proof relies on several sketching and leverage score sampling results in randomized numerical linear algebra [19, 20, 64]. These prerequisite results are well-known, but scattered through the literature. They are the building blocks for proving our approximate block-regression results in Lemma 3.4 and Corollary 3.5.

A.1 Approximate Least Squares

We follow the outline of Larsen and Kolda [39] (originally written in [38, Appendix B]). Consider the overdetermined least squares problem defined by a matrix $\mathbf{A} \in \mathbb{R}^{n \times d}$ and response vector $\mathbf{b} \in \mathbb{R}^n$, where $n \geq d$ and $\text{rank}(\mathbf{A}) = d$. Define the optimal sum of squared residuals to be

$$\mathcal{R}^2 = \min_{\mathbf{x} \in \mathbb{R}^d} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2. \tag{7}$$

Assume for now $\mathbf{A}$ is full rank. Let the compact SVD of the design matrix be $\mathbf{A} = \mathbf{U}_\mathbf{A}\mathbf{\Sigma}_\mathbf{A}\mathbf{V}_\mathbf{A}^\intercal$. By definition, $\mathbf{U}_\mathbf{A} \in \mathbb{R}^{n \times d}$ is an orthonormal basis for the column space of $\mathbf{A}$. Let $\mathbf{U}_\mathbf{A}^\perp \in \mathbb{R}^{n \times (n-d)}$ be an orthonormal basis for the $(n-d)$-dimensional subspace that is orthogonal to the column space of $\mathbf{A}$. For notational simplicity, let $\mathbf{b}^\perp = \mathbf{U}_\mathbf{A}^\perp \mathbf{U}_\mathbf{A}^{\perp\intercal}\mathbf{b}$ denote the projection of $\mathbf{b}$ onto the orthogonal subspace $\mathbf{U}_\mathbf{A}^\perp$. The vector $\mathbf{b}^\intercal$ is important because its norm is equal to the norm of the residual vector. To see this, observe that $\mathbf{x}$ can be chosen so that $\mathbf{A}\mathbf{x}$ perfectly matches the part of $\mathbf{b}$ in the column space of $\mathbf{A}$, but cannot (by definition) match anything in the range of $\mathbf{U}_\mathbf{A}^\perp$:

$$\mathcal{R}^2 = \min_{\mathbf{x} \in \mathbb{R}^d} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 = \left\|\mathbf{U}_\mathbf{A}^\perp \mathbf{U}_\mathbf{A}^{\perp\intercal}\mathbf{b}\right\|_2^2 = \left\|\mathbf{b}^\perp\right\|_2^2. \tag{8}$$

We denote the solution to the least squares problem by $\mathbf{x}_{\text{opt}}$, hence we have $\mathbf{b} = \mathbf{A}\mathbf{x}_{\text{opt}} + \mathbf{b}^\perp$.

Now we build on a structural result of Drineas et al. [20] that establishes sufficient conditions on any sketching matrix $\mathbf{S} \in \mathbb{R}^{s \times n}$ such that the solution $\tilde{\mathbf{x}}_{\text{opt}}$ to the approximate least squares problem

$$\tilde{\mathbf{x}}_{\text{opt}} = \arg\min_{\mathbf{x} \in \mathbb{R}^d} \|\mathbf{S}(\mathbf{A}\mathbf{x} - \mathbf{b})\|_2^2 \tag{9}$$

gives a relative-error approximation to the original least squares problem. The two conditions we require of matrix $\mathbf{S}$ are:

$$\sigma_{\min}^2(\mathbf{S}\mathbf{U}_\mathbf{A}) \geq 1/\sqrt{2}, \text{ and} \tag{10}$$

$$\left\|\mathbf{U}_\mathbf{A}^\intercal \mathbf{S}^\intercal \mathbf{S}\mathbf{b}^\perp\right\|_2^2 \leq \varepsilon \mathcal{R}^2/2, \tag{11}$$

for some $\varepsilon \in (0, 1)$. While the algorithms we consider in this work are randomized, the following lemma is a deterministic statement. Failure probabilities enter our analysis later when we show our sketch matrices satisfy conditions (10) and (11) with sufficiently high probability.

Lemma A.1 (Drineas et al. [20, Lemma 1])**.** *Consider the overconstrained least squares approximation problem in (7), and let the matrix $\mathbf{U}_\mathbf{A} \in \mathbb{R}^{n \times d}$ contain the top d left singular vectors of $\mathbf{A}$. Assume the matrix $\mathbf{S}$ satisfies conditions (10) and (11) for some $\varepsilon \in (0, 1)$. Then, the solution $\tilde{\mathbf{x}}_{\text{opt}}$ to the approximate least squares problem (9) satisfies:*

$$\|\mathbf{A}\tilde{\mathbf{x}}_{\text{opt}} - \mathbf{b}\|_2^2 \leq (1+\varepsilon)\mathcal{R}^2, \textit{ and} \tag{12}$$

$$\|\tilde{\mathbf{x}}_{\text{opt}} - \mathbf{x}_{\text{opt}}\|_2^2 \leq \frac{1}{\sigma_{\min}^2(\mathbf{A})}\varepsilon\mathcal{R}^2. \tag{13}$$

Proof. Let us first rewrite the sketched least squares problem induced by $\mathbf{S}$ as

$$\begin{aligned}\min_{\mathbf{x} \in \mathbb{R}^d} \|\mathbf{S}\mathbf{A}\mathbf{x} - \mathbf{S}\mathbf{b}\|_2^2 &= \min_{\mathbf{y} \in \mathbb{R}^d} \left\|\mathbf{S}\mathbf{A}(\mathbf{x}_{\text{opt}} + \mathbf{y}) - \mathbf{S}\left(\mathbf{A}\mathbf{x}_{\text{opt}} + \mathbf{b}^\perp\right)\right\|_2^2 \qquad (14)\\ &= \min_{\mathbf{y} \in \mathbb{R}^d} \left\|\mathbf{S}\mathbf{A}\mathbf{y} - \mathbf{S}\mathbf{b}^\perp\right\|_2^2 \\ &= \min_{\mathbf{z} \in \mathbb{R}^d} \left\|\mathbf{S}\mathbf{U}_\mathbf{A}\mathbf{z} - \mathbf{S}\mathbf{b}^\perp\right\|_2^2. \qquad (15)\end{aligned}$$

Equation (14) is true because $\mathbf{b} = \mathbf{A}\mathbf{x}_{\text{opt}} + \mathbf{b}^{\perp}$, and (15) follows because the columns of $\mathbf{A}$ span the same subspace as the columns of $\mathbf{U}_{\mathbf{A}}$. Now, let $\mathbf{z}_{\text{opt}} \in \mathbb{R}^d$ be such that $\mathbf{U}_{\mathbf{A}}\mathbf{z}_{\text{opt}} = \mathbf{A}(\tilde{\mathbf{x}}_{\text{opt}} - \mathbf{x}_{\text{opt}})$ and note that $\mathbf{z}_{\text{opt}}$ minimizes (15). This fact follows from

$$\left\|\mathbf{S}\mathbf{A}(\tilde{\mathbf{x}}_{\text{opt}} - \mathbf{x}_{\text{opt}}) - \mathbf{S}\mathbf{b}^{\perp}\right\|_2^2 = \left\|\mathbf{S}\mathbf{A}\tilde{\mathbf{x}}_{\text{opt}} - \mathbf{S}\big(\mathbf{b} - \mathbf{b}^{\perp}\big) - \mathbf{S}\mathbf{b}^{\perp}\right\|_2^2 = \|\mathbf{S}\mathbf{A}\tilde{\mathbf{x}}_{\text{opt}} - \mathbf{S}\mathbf{b}\|_2^2.$$

Thus, by the normal equations, we have

$$(\mathbf{S}\mathbf{U}_{\mathbf{A}})^{\intercal}\mathbf{S}\mathbf{U}_{\mathbf{A}}\mathbf{z}_{\text{opt}} = (\mathbf{S}\mathbf{U}_{\mathbf{A}})^{\intercal}\mathbf{S}\mathbf{b}^{\perp}.$$

Taking the norm of both sides and observing that under condition (10) we have $\sigma_i((\mathbf{S}\mathbf{U}_{\mathbf{A}})^{\intercal}\mathbf{S}\mathbf{U}_{\mathbf{A}}) = \sigma_i^2(\mathbf{S}\mathbf{U}_{\mathbf{A}}) \geq 1/\sqrt{2}$, for all $i \in [d]$, it follows that

$$\|\mathbf{z}_{\text{opt}}\|_2^2/2 \leq \|(\mathbf{S}\mathbf{U}_{\mathbf{A}})^{\intercal}\mathbf{S}\mathbf{U}_{\mathbf{A}}\mathbf{z}_{\text{opt}}\|_2^2 = \left\|(\mathbf{S}\mathbf{U}_{\mathbf{A}})^{\intercal}\mathbf{S}\mathbf{b}^{\perp}\right\|_2^2. \tag{16}$$

Using condition (11), we observe that

$$\|\mathbf{z}_{\text{opt}}\|_2^2 \leq 2\left\|(\mathbf{S}\mathbf{U}_{\mathbf{A}})^{\intercal}\mathbf{S}\mathbf{b}^{\perp}\right\|_2^2 \leq \varepsilon\mathcal{R}^2. \tag{17}$$

To establish the first claim of the lemma, let us rewrite the squared norm of the residual vector as

$$\begin{aligned}
\|\mathbf{A}\tilde{\mathbf{x}}_{\text{opt}} - \mathbf{b}\|_2^2 &= \|\mathbf{A}\tilde{\mathbf{x}}_{\text{opt}} - \mathbf{A}\mathbf{x}_{\text{opt}} + \mathbf{A}\mathbf{x}_{\text{opt}} - \mathbf{b}\|_2^2 \\
&= \|\mathbf{A}\tilde{\mathbf{x}}_{\text{opt}} - \mathbf{A}\mathbf{x}_{\text{opt}}\|_2^2 + \|\mathbf{A}\mathbf{x}_{\text{opt}} - \mathbf{b}\|_2^2 &(18)\\
&= \|\mathbf{U}_{\mathbf{A}}\mathbf{z}_{\text{opt}}\|_2^2 + \mathcal{R}^2 &(19)\\
&\leq (1+\varepsilon)\mathcal{R}^2, &(20)
\end{aligned}$$

where (18) follows from the Pythagorean theorem since $\mathbf{b} - \mathbf{A}\mathbf{x}_{\text{opt}} = \mathbf{b}^{\perp}$, which is orthogonal to $\mathbf{A}$, and consequently $\mathbf{A}(\mathbf{x}_{\text{opt}} - \tilde{\mathbf{x}}_{\text{opt}})$; (19) follows from the definition of $\mathbf{z}_{\text{opt}}$ and $\mathcal{R}^2$; and (20) follows from (17) and the orthogonality of $\mathbf{U}_{\mathbf{A}}$.

To establish the second claim of the lemma, recall that $\mathbf{A}(\mathbf{x}_{\text{opt}} - \tilde{\mathbf{x}}_{\text{opt}}) = \mathbf{U}_{\mathbf{A}}\mathbf{z}_{\text{opt}}$. Taking the norm of both sides of this expression, we have

$$\begin{aligned}
\|\mathbf{x}_{\text{opt}} - \tilde{\mathbf{x}}_{\text{opt}}\|_2^2 &\leq \frac{\|\mathbf{U}_{\mathbf{A}}\mathbf{z}_{\text{opt}}\|_2^2}{\sigma_{\min}^2(\mathbf{A})} &(21)\\
&\leq \frac{\varepsilon\mathcal{R}^2}{\sigma_{\min}^2(\mathbf{A})}, &(22)
\end{aligned}$$

where (21) follows since $\sigma_{\min}(\mathbf{A})$ is the smallest singular value of $\mathbf{A}$ and $\text{rank}(\mathbf{A}) = d$; and (22) follows from (17) and the orthogonality of $\mathbf{U}_{\mathbf{A}}$. □

Next we present two results that are useful for proving our sketches $\mathbf{S}$ satisfy the structural conditions in Equations (10) and (11). The first result states $\mathbf{S}\mathbf{U}_{\mathbf{A}}$ is a subspace embedding for the column space of $\mathbf{U}_{\mathbf{A}}$. This result can be thought of as an approximate isometry and is noticeably stronger than the desired condition $\sigma_{\min}^2(\mathbf{S}\mathbf{U}_{\mathbf{A}}) \geq 1/\sqrt{2}$.

Theorem A.2 (Woodruff [64, Theorem 17])**.** *Consider $\mathbf{A} \in \mathbb{R}^{n\times d}$ and its compact SVD $\mathbf{A} = \mathbf{U}_{\mathbf{A}}\mathbf{\Sigma}_{\mathbf{A}}\mathbf{V}_{\mathbf{A}}^{\intercal}$. Let $\mathbf{p} \in [0,1]^n$ be a β-overestimate for the leverage score distribution of $\mathbf{A}$. Let $s > 144d\ln(2d/\delta)/(\beta\varepsilon^2)$. Let the matrix $\mathbf{S} \in \mathbb{R}^{s\times n}$ be the output of* $\texttt{SampleRows}(\mathbf{A}, s, \mathbf{p})$ *(Definition 3.2). Then, with probability at least $1-\delta$, simultaneously for all i, we have*

$$1 - \varepsilon \leq \sigma_i^2(\mathbf{S}\mathbf{U}_{\mathbf{A}}) \leq 1 + \varepsilon.$$

For the second structural condition, we use the following result about squared-distance sampling for approximate matrix multiplication in [19]. In our analysis of block leverage score sampling (e.g., ridge regression), it is possible (and beneficial) that $\beta > 1$ and that rows are sometimes not sampled. We modify the original theorem statement and provide a proof to show that the result is unaffected.

Theorem A.3 (Drineas et al. [19, Lemma 8]). *Let $\mathbf{A} \in \mathbb{R}^{n\times m}$, $\mathbf{B} \in \mathbb{R}^{n\times p}$, and s denote the number of samples. Let the vector $\mathbf{p} \in [0,1]^n$ contain probabilities such that, for all $i \in [n]$, we have*

$$p_i \geq \beta \frac{\|\mathbf{a}_{i:}\|_2^2}{\|\mathbf{A}\|_{\mathrm{F}}^2},$$

for some constant $\beta > 0$. We require that $\|\mathbf{p}\|_1 \leq 1$, but it is possible that $\mathbf{p}$ does not contain all of the probability mass (i.e., $\|\mathbf{p}\|_1 < 1$). Sample s row indices $(\xi^{(1)}, \xi^{(2)}, \ldots, \xi^{(s)})$ from $\mathbf{p}$, independently and with replacement, and form the approximate product

$$\frac{1}{s}\sum_{t=1}^{s} \frac{1}{p_{\xi^{(t)}}} \mathbf{a}_{\xi^{(t)}:}^{\intercal} \mathbf{b}_{\xi^{(t)}:} = (\mathbf{SA})^{\intercal}\mathbf{SB},$$

where $\mathbf{S} \in \mathbb{R}^{s\times n}$ is the sampling and rescaling matrix whose t-th row is defined by the entries

$$s_{tk} = \begin{cases} \frac{1}{\sqrt{sp_k}} & \text{if } k = \xi_t, \\ 0 & \text{otherwise}. \end{cases}$$

Disregard trials that occur with the remaining probability $1 - \|\mathbf{p}\|_1$. Then, we have

$$\mathbb{E}\Big[\|\mathbf{A}^{\intercal}\mathbf{B} - (\mathbf{SA})^{\intercal}\mathbf{SB}\|_{\mathrm{F}}^2\Big] \leq \frac{1}{\beta s}\|\mathbf{A}\|_{\mathrm{F}}^2\|\mathbf{B}\|_{\mathrm{F}}^2.$$

Proof. First we analyze the entry of $(\mathbf{SA})^{\intercal}\mathbf{SB}$ at index (i,j). Viewing the approximate product as a sum of outer products, we can write this entry in terms of scalar random variables X_t, for $t \in [s]$, as follows:

$$X_t = \begin{cases} \frac{a_{\xi^{(t)}i} b_{\xi^{(t)}j}}{sp_{\xi^{(t)}}} & \text{with probability } p_i \text{ for each } i \in [n], \\ 0 & \text{otherwise with probability } 1 - \|p\|_1 \end{cases} \implies [(\mathbf{SA})^{\intercal}\mathbf{SB}]_{ij} = \sum_{t=1}^{s} X_t.$$

The expected values of X_t and X_t^2 for all values of t are

$$\mathbb{E}[X_t] = \sum_{k=1}^{n} p_k \frac{a_{ki}b_{kj}}{sp_k} = \frac{1}{s}(\mathbf{A}^{\intercal}\mathbf{B})_{ij}, \text{ and}$$
$$\mathbb{E}\big[X_t^2\big] = \sum_{k=1}^{n} p_k \left(\frac{a_{ki}b_{kj}}{sp_k}\right)^2 = \frac{1}{s^2}\sum_{k=1}^{n} \frac{(a_{ki}b_{kj})^2}{p_k}.$$

Therefore, $\mathbb{E}[[(\mathbf{SA})^{\intercal}\mathbf{SB}]_{ij}] = \sum_{t=1}^{s}\mathbb{E}[X_t] = (\mathbf{A}^{\intercal}\mathbf{B})_{ij}$, which means the estimator is unbiased. Furthermore, since the estimated matrix entry is the sum of s i.i.d. random variables, its variance is

$$\begin{aligned}
\mathrm{Var}\Big([(\mathbf{SA})^{\intercal}\mathbf{SB}]_{ij}\Big) &= \sum_{t=1}^{s} \mathrm{Var}(X_t) \\
&= \sum_{t=1}^{s}\Big(\mathbb{E}\big[X_t^2\big] - \mathbb{E}[X_t]^2\Big) \\
&= \sum_{t=1}^{s}\frac{1}{s^2}\sum_{k=1}^{n}\left(\frac{(a_{ki}b_{kj})^2}{p_k} - (\mathbf{A}^{\intercal}\mathbf{B})_{ij}^2\right) \\
&= \frac{1}{s}\sum_{k=1}^{n}\left(\frac{(a_{ki}b_{kj})^2}{p_k} - (\mathbf{A}^{\intercal}\mathbf{B})_{ij}^2\right).
\end{aligned}$$

Now we apply this result to the expectation we want to bound:

$$\begin{aligned}
\mathbb{E}\Big[\|\mathbf{A}^\intercal\mathbf{B} - (\mathbf{SA})^\intercal\mathbf{SB}\|_{\mathrm{F}}^2\Big] &= \sum_{i=1}^{m}\sum_{j=1}^{p}\mathbb{E}\bigg[\Big([(\mathbf{SA})^\intercal\mathbf{SB}]_{ij} - (\mathbf{A}^\intercal\mathbf{B})_{ij}\Big)^2\bigg] \\
&= \sum_{i=1}^{m}\sum_{j=1}^{p}\mathbb{E}\bigg[\Big([(\mathbf{SA})^\intercal\mathbf{SB}]_{ij} - \mathbb{E}\Big[[(\mathbf{SA})^\intercal\mathbf{SB}]_{ij}\Big]\Big)^2\bigg] \\
&= \sum_{i=1}^{m}\sum_{j=1}^{p}\mathrm{Var}\Big([(\mathbf{SA})^\intercal\mathbf{SB}]_{ij}\Big) \\
&= \frac{1}{s}\sum_{i=1}^{m}\sum_{j=1}^{p}\sum_{k=1}^{n}\left(\frac{(a_{ki}b_{kj})^2}{p_k} - (\mathbf{A}^\intercal\mathbf{B})_{ij}^2\right) \\
&= \frac{1}{s}\sum_{k=1}^{n}\frac{(\sum_{i=1}^{m}a_{ki}^2)(\sum_{j=1}^{p}b_{kj}^2)}{p_k} - \frac{n}{s}\sum_{i=1}^{m}\sum_{j=1}^{p}(\mathbf{A}^\intercal\mathbf{B})_{ij}^2 \\
&= \frac{1}{s}\sum_{k=1}^{n}\frac{\|\mathbf{a}_{k:}\|_2^2\|\mathbf{b}_{k:}\|_2^2}{p_k} - \frac{n}{s}\|\mathbf{A}^\intercal\mathbf{B}\|_{\mathrm{F}}^2 \\
&\leq \frac{1}{s}\sum_{k=1}^{n}\frac{\|\mathbf{a}_{k:}\|_2^2\|\mathbf{b}_{k:}\|_2^2}{p_k}.
\end{aligned}$$

The last inequality uses the fact that the Frobenius norm of any matrix is nonnegative. Finally, by using the β-overestimate assumption on the sampling probabilities, we have

$$\begin{aligned}
\mathbb{E}\Big[\|\mathbf{A}^\intercal\mathbf{B} - (\mathbf{SA})^\intercal\mathbf{SB}\|_{\mathrm{F}}^2\Big] &\leq \frac{1}{s}\sum_{k=1}^{n}\frac{\|\mathbf{a}_{k:}\|_2^2\|\mathbf{b}_{k:}\|_2^2}{p_k} \\
&\leq \frac{1}{s}\sum_{k=1}^{n}\left(\|\mathbf{A}\|_{\mathrm{F}}^2\frac{\|\mathbf{a}_{k:}\|_2^2\|\mathbf{b}_{k:}\|_2^2}{\beta\|\mathbf{a}_{k:}\|_2^2}\right) \\
&= \frac{1}{s\beta}\|\mathbf{A}\|_{\mathrm{F}}^2\sum_{k=1}^{n}\|\mathbf{b}_{k:}\|_2^2 \\
&= \frac{1}{s\beta}\|\mathbf{A}\|_{\mathrm{F}}^2\|\mathbf{B}\|_{\mathrm{F}}^2,
\end{aligned}$$

which is the desired upper bound. □

A.2 Generalizing to Submatrix Sketching

Now that our main tools are in place, we extend the analysis of approximate least squares to work with sketched submatrices of the vertically stacked block design matrix.

Lemma 3.3. *Let $\mathbf{A} = [\mathbf{A}_1; \mathbf{A}_2]$ be vertically stacked with $\mathbf{A}_1 \in \mathbb{R}^{n_1\times d}$ and $\mathbf{A}_2 \in \mathbb{R}^{n_2\times d}$. Let $\mathbf{p} \in [0,1]^{n_1}$ be a β-overestimate for the leverage score distribution of $\mathbf{A}_1$. If $s > 144d\ln(2d/\delta)/(\beta\varepsilon^2)$, the sketch $\mathbf{S}$ returned by* `SampleRows`$(\mathbf{A}_1, s, \mathbf{p})$ *guarantees, with probability at least $1-\delta$, that*

$$(1-\varepsilon)\mathbf{A}^\intercal\mathbf{A} \preccurlyeq (\mathbf{SA}_1)^\intercal\mathbf{SA}_1 + \mathbf{A}_2^\intercal\mathbf{A}_2 \preccurlyeq (1+\varepsilon)\mathbf{A}^\intercal\mathbf{A}.$$

Proof. Write the compact SVD of $\mathbf{A_1}$ as $\mathbf{A}_1 = \mathbf{U}_{\mathbf{A}_1}\mathbf{\Sigma}_{\mathbf{A}_1}\mathbf{V}_{\mathbf{A}_1}^\intercal$. Theorem A.2 guarantees that with probability at least $1-\delta$,

$$1-\varepsilon \leq \sigma_i^2(\mathbf{SU}_{\mathbf{A}_1}) \leq 1+\varepsilon.$$

Therefore, we have

$$(1-\varepsilon)\mathbf{I}_d \preccurlyeq (\mathbf{SU}_{\mathbf{A}_1})^\intercal\mathbf{SU}_{\mathbf{A}_1} \preccurlyeq (1+\varepsilon)\mathbf{I}_d.$$

It follows that

$$\begin{aligned}
(1-\varepsilon)\mathbf{A}_1^\intercal\mathbf{A}_1 &= (1-\varepsilon)\mathbf{V}_{\mathbf{A}_1}\mathbf{\Sigma}_{\mathbf{A}_1}^\intercal\mathbf{I}_d\mathbf{\Sigma}_{\mathbf{A}_1}\mathbf{V}_{\mathbf{A}_1}^\intercal \\
&\preccurlyeq \mathbf{V}_{\mathbf{A}_1}\mathbf{\Sigma}_{\mathbf{A}_1}^\intercal\mathbf{U}_{\mathbf{A}_1}^\intercal\mathbf{S}^\intercal\mathbf{SU}_{\mathbf{A}_1}\mathbf{\Sigma}_{\mathbf{A}_1}\mathbf{V}_{\mathbf{A}_1}^\intercal \\
&= (\mathbf{SA}_1)^\intercal\mathbf{SA}_1.
\end{aligned}$$

Similarly, we have $(\mathbf{S}\mathbf{A}_1)^\intercal \mathbf{S}\mathbf{A}_1 \preccurlyeq (1+\varepsilon)\mathbf{A}_1^\intercal \mathbf{A}_1$. Writing $\mathbf{A}^\intercal\mathbf{A} = \mathbf{A}_1^\intercal\mathbf{A}_1 + \mathbf{A}_2^\intercal\mathbf{A}_2$ as the sum of outer products, we have

$$\begin{aligned}(1-\varepsilon)\mathbf{A}^\intercal\mathbf{A} &\preccurlyeq (1-\varepsilon)\mathbf{A}_1^\intercal\mathbf{A}_1 + \mathbf{A}_2^\intercal\mathbf{A}_2\\ &\preccurlyeq (\mathbf{S}\mathbf{A}_1)^\intercal\mathbf{S}\mathbf{A}_1 + \mathbf{A}_2^\intercal\mathbf{A}_2\\ &\preccurlyeq (1+\varepsilon)\mathbf{A}_1^\intercal\mathbf{A}_1 + \mathbf{A}_2^\intercal\mathbf{A}_2\\ &\preccurlyeq (1+\varepsilon)\mathbf{A}^\intercal\mathbf{A},\end{aligned}$$

which completes the proof. □

Lemma 3.4 (Approximate block regression)**.** *Consider the problem* $\arg\min_{\mathbf{x}\in\mathbb{R}^d}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2$ *where* $\mathbf{A} = [\mathbf{A}_1;\mathbf{A}_2]$ *and* $\mathbf{b} = [\mathbf{b}_1;\mathbf{b}_2]$ *are vertically stacked and* $\mathbf{A}_1 \in \mathbb{R}^{n_1\times d}$*,* $\mathbf{A}_2 \in \mathbb{R}^{n_2\times d}$*,* $\mathbf{b}_1 \in \mathbb{R}^{n_1}$*,* $\mathbf{b}_2 \in \mathbb{R}^{n_2}$*. Let* $\mathbf{p}\in[0,1]^{n_1}$ *be a* β*-overestimate for the leverage score distribution of* $\mathbf{A}_1$*. Let* $s \geq 1680d\ln(40d)/(\beta\varepsilon)$ *and let* $\mathbf{S}$ *be the output of* $\texttt{SampleRows}(\mathbf{A}_1, s, \mathbf{p})$*. If*

$$\tilde{\mathbf{x}}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^d}\Big(\|\mathbf{S}(\mathbf{A}_1\mathbf{x}-\mathbf{b}_1)\|_2^2 + \|\mathbf{A}_2\mathbf{x}-\mathbf{b}_2\|_2^2\Big),$$

then, with probability at least $9/10$*, we have*

$$\|\mathbf{A}\tilde{\mathbf{x}}^* - \mathbf{b}\|_2^2 \leq (1+\varepsilon)\min_{\mathbf{x}\in\mathbb{R}^d}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2.$$

Proof. Let $\delta = 1/10$ be the desired failure probability. Consider the augmented sketch matrix

$$\mathbf{S}' = \begin{bmatrix}\mathbf{S} & \mathbf{0}_{s\times n_2}\\ \mathbf{0}_{n_2\times n_1} & \mathbf{I}_{n_2}\end{bmatrix}. \tag{23}$$

It follows that

$$\mathbf{S}'\mathbf{A} = \begin{bmatrix}\mathbf{S}\mathbf{A}_1\\ \mathbf{A}_2\end{bmatrix}. \tag{24}$$

Let the compact SVD of $\mathbf{A}$ be $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\intercal$. We prove that each of the structural conditions about $\mathbf{S}'$ in (10) and (11) fail with probability at most $\delta/2$. Then we use a union bound and apply Lemma A.1.

Satisfying structural condition 1. It follows from (24) that

$$(\mathbf{S}'\mathbf{A})^\intercal\mathbf{S}'\mathbf{A} = (\mathbf{S}\mathbf{A}_1)^\intercal\mathbf{S}\mathbf{A}_1 + \mathbf{A}_2^\intercal\mathbf{A}_2.$$

Using Lemma 3.3, we know

$$(1-\varepsilon)\mathbf{A}^\intercal\mathbf{A} \preccurlyeq (\mathbf{S}'\mathbf{A})^\intercal\mathbf{S}'\mathbf{A} \preccurlyeq (1+\varepsilon)\mathbf{A}^\intercal\mathbf{A}. \tag{25}$$

Since $\mathbf{A}^\intercal\mathbf{A} = \mathbf{V}\mathbf{\Sigma}^\intercal\mathbf{I}_d\mathbf{\Sigma}\mathbf{V}^\intercal$ and $(\mathbf{S}'\mathbf{A}^\intercal)\mathbf{S}'\mathbf{A} = \mathbf{V}\mathbf{\Sigma}^\intercal\mathbf{U}^\intercal\mathbf{S}'^\intercal\mathbf{S}'\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\intercal$, it follows from (25) that

$$(1-\varepsilon)\mathbf{I}_d \preccurlyeq (\mathbf{S}'\mathbf{U})^\intercal\mathbf{S}'\mathbf{U} \preccurlyeq (1+\varepsilon)\mathbf{I}_d$$

since $\mathbf{\Sigma}$ and $\mathbf{V}^\intercal$ are positive definite. Therefore, the first structural condition (10) is true with probability at least $1-\delta/2$ as long as $1-\varepsilon \geq 1/\sqrt{2}$. This means the number of samples needs to be at least

$$s > \frac{144d\ln(4d/\delta)}{\beta(1-1/\sqrt{2})^2} > \frac{1680d\ln(4d/\delta)}{\beta}.$$

Satisfying structural condition 2. We show (11) holds with probability at least $1-\delta/2$ using a modification of Theorem A.3 and Markov's inequality. First observe that

$$\mathbf{U}^\intercal\mathbf{b}^\perp = \mathbf{U}^\intercal\Big(\mathbf{U}^\perp\mathbf{U}^{\perp\intercal}\mathbf{b}\Big) = \mathbf{0}_{\text{rank}(\mathbf{A})},$$

where $\mathbf{b}^\perp$ is defined as in Appendix A.1. Thus, the second structural condition can be seen as bounding how closely this sampled product approximates the zero vector. It follows that

$$\begin{aligned}\left\|\mathbf{U}^\intercal\mathbf{S}'^\intercal\mathbf{S}'\mathbf{b}^\perp\right\|_2^2 &= \left\|\mathbf{U}^\intercal\mathbf{b}^\perp - \mathbf{U}^\intercal\mathbf{S}'^\intercal\mathbf{S}'\mathbf{b}^\perp\right\|_2^2\\ &= \left\|\mathbf{U}^\intercal(\mathbf{I}_{n_1+n_2} - \mathbf{S}'^\intercal\mathbf{S}')\mathbf{b}^\perp\right\|_2^2\\ &= \left\|\mathbf{U}^\intercal\begin{bmatrix}\mathbf{I}_{n_1}-\mathbf{S}^\intercal\mathbf{S} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\mathbf{b}^\perp\right\|_2^2\\ &= \left\|\tilde{\mathbf{U}}^\intercal(\mathbf{I}_{n_1}-\mathbf{S}^\intercal\mathbf{S})\tilde{\mathbf{b}}^\perp\right\|_2^2\\ &= \left\|\tilde{\mathbf{U}}^\intercal\tilde{\mathbf{b}}^\perp - \tilde{\mathbf{U}}^\intercal\mathbf{S}^\intercal\mathbf{S}\tilde{\mathbf{b}}^\perp\right\|_2^2,\end{aligned}$$

where $\tilde{\mathbf{U}} \in \mathbb{R}^{n_1 \times d}$ and $\tilde{\mathbf{b}}^\perp \in \mathbb{R}^{n_1}$ denote the first n_1 rows of $\mathbf{U}$ and $\mathbf{b}^\perp$, respectively.

Now we bound the probability that a row index in $\tilde{\mathbf{U}}$ is sampled when constructing $\mathbf{S}$, which allows us to apply Theorem A.3:

$$\Pr(\text{row } i \in [n_1] \text{ is sampled}) \geq \beta \frac{\ell_i(\mathbf{A}_1)}{\|\boldsymbol{\ell}(\mathbf{A}_1)\|} \tag{26}$$

$$= \beta \frac{\ell_i(\mathbf{A}_1)}{\text{rank}(\mathbf{A}_1)} \cdot \frac{\|\boldsymbol{\ell}_{\mathbf{A}_1}(\mathbf{A})\|_1}{\ell_i(\mathbf{A})} \cdot \frac{\ell_i(\mathbf{A})}{\|\boldsymbol{\ell}_{\mathbf{A}_1}(\mathbf{A})\|_1} \tag{27}$$

$$\geq \beta \frac{\|\boldsymbol{\ell}_{\mathbf{A}_1}(\mathbf{A})\|_1}{\text{rank}(\mathbf{A}_1)} \cdot \frac{\ell_i(\mathbf{A})}{\|\boldsymbol{\ell}_{\mathbf{A}_1}(\mathbf{A})\|_1} \tag{28}$$

$$= \beta \frac{\|\tilde{\mathbf{U}}\|_{\text{F}}^2}{\text{rank}(\mathbf{A}_1)} \cdot \frac{\|\tilde{\mathbf{u}}_{i:}\|_2^2}{\|\tilde{\mathbf{U}}\|_{\text{F}}^2}. \tag{29}$$

We use $\|\boldsymbol{\ell}_{\mathbf{A}_1}(\mathbf{A})\|_1 = \sum_{i \in [n_1]} \ell_i(\mathbf{A})$ to denote the sum of leverage scores of $\mathbf{A}$ corresponding to the rows of $\mathbf{A}_1$. Equation (28) holds because leverage scores do not increase when rows are added to the matrix, i.e., $\ell_i(\mathbf{A}_1) \geq \ell_i(\mathbf{A})$. Equation (29) is true because the leverage scores of $\mathbf{A}$ corresponding to the rows in $\mathbf{A}_1$ are given by the submatrix $\tilde{\mathbf{U}}$ in the compact SVD of $\mathbf{A}$. Therefore, Theorem A.3 guarantees that

$$\begin{aligned}
\left\|\mathbf{U}^\intercal \mathbf{S}'^\intercal \mathbf{S}' \mathbf{b}^\perp\right\|_2^2 &= \left\|\tilde{\mathbf{U}}^\intercal \tilde{\mathbf{b}}^\perp - (\mathbf{S}\tilde{\mathbf{U}})^\intercal \mathbf{S}\tilde{\mathbf{b}}^\perp\right\|_2^2 \\
&\leq \frac{\text{rank}(\mathbf{A}_1)}{\beta \|\tilde{\mathbf{U}}\|_{\text{F}}^2 s} \cdot \|\tilde{\mathbf{U}}\|_{\text{F}}^2 \|\tilde{\mathbf{b}}^\perp\|_2^2 \\
&\leq \frac{\text{rank}(\mathbf{A}_1)}{\beta s} \cdot \|\mathbf{b}^\perp\|_2^2.
\end{aligned}$$

Since $\mathbf{b}^\perp$ is the residual vector, applying Markov's inequality gives us

$$\Pr\left(\left\|\mathbf{U}^\intercal \mathbf{S}'^\intercal \mathbf{S}' \mathbf{b}^\perp\right\|_2^2 \geq \frac{\varepsilon \|\mathbf{b}^\perp\|_2^2}{2}\right) \leq \frac{\text{rank}(\mathbf{A}_1)}{\beta s} \cdot \|\mathbf{b}^\perp\|_2^2 \cdot \frac{2}{\varepsilon \|\mathbf{b}^\perp\|_2^2} = \frac{2 \cdot \text{rank}(\mathbf{A}_1)}{\beta s \varepsilon}. \tag{30}$$

To upper bound (30) by a failure probability of $\delta/2$, the number of samples needs to be at least

$$s \geq \frac{4 \cdot \text{rank}(\mathbf{A}_1)}{\beta \delta \varepsilon}.$$

Conclusion. Since $d \geq \text{rank}(\mathbf{A}_1)$ and $\delta = 1/10$, it follows that

$$\max\left\{\frac{1680 d \ln(4d/\delta)}{\beta}, \frac{4d}{\beta \delta \varepsilon}\right\} \leq \frac{1680 d \ln(40d)}{\beta \varepsilon} \leq s$$

samples are sufficient for both structural conditions to hold at the same time with probability at least $1 - (\delta/2 + \delta/2) = 1 - \delta$ by a union bound. Finally, we may apply Lemma A.1 to achieve the $(1+\varepsilon)$-approximation guarantee. □

Corollary 3.5. *For any $\mathbf{A} \in \mathbb{R}^{n \times d}$, $\mathbf{b} \in \mathbb{R}^d$, $\lambda \geq 0$, consider*

$$\arg\min_{\mathbf{x} \in \mathbb{R}^d} (\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda \|\mathbf{x}\|_2^2).$$

Let $\mathbf{p} \in [0,1]^{n_1}$ be a β-overestimate for the leverage scores of $\mathbf{A}$ and $s \geq 1680 d \ln(40d)/(\beta\varepsilon)$. If $\mathbf{S}$ is the output of `SampleRows`$(\mathbf{A}, s, \mathbf{p})$*, then, with probability at least $9/10$, the sketched solution*

$$\tilde{\mathbf{x}}^* = \arg\min_{\mathbf{x} \in \mathbb{R}^d} (\|\mathbf{S}(\mathbf{A}\mathbf{x} - \mathbf{b})\|_2^2 + \lambda \|\mathbf{x}\|_2^2)$$

gives a $(1+\varepsilon)$-approximation to the original problem.

Proof. This is an immediate consequence of our results for approximate block regression in Lemma 3.4. Consider the augmented matrices

$$\mathbf{A}' = \begin{bmatrix} \mathbf{A} \\ \sqrt{\lambda}\mathbf{I}_d \end{bmatrix} \quad \text{and} \quad \mathbf{b}' = \begin{bmatrix} \mathbf{b} \\ \mathbf{0_d} \end{bmatrix}.$$

For any $\mathbf{x} \in \mathbb{R}^d$, we have

$$\|\mathbf{A}'\mathbf{x} - \mathbf{b}'\|_2^2 = \left\| \begin{bmatrix} \mathbf{A}\mathbf{x} - \mathbf{b} \\ \sqrt{\lambda}\mathbf{x} \end{bmatrix} \right\|_2^2 = \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_2^2.$$

Therefore, it suffices to approximately solve $\arg\min_{\mathbf{x}\in\mathbb{R}^d} \|\mathbf{A}'\mathbf{x} - \mathbf{b}'\|$, so we can use Lemma 3.4 to complete the proof. □

B Missing analysis for Section 4

Here we present an explicit formula for the λ-ridge leverage scores of a Kronecker matrix in terms of the singular value decompositions of its factor matrices. Then we show how to achieve fast Kronecker product-matrix multiplications in Appendix B.1 and prove our main Kronecker regression theorem in Appendix B.2.

Lemma 4.1. *Let $\mathbf{K} = \mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)} \otimes \cdots \otimes \mathbf{A}^{(N)}$, where each factor matrix $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$. Let $(i_1, i_2, \ldots, i_N)$ be the natural row indexing of $\mathbf{K}$ by its factors. Let the factor SVDs be $\mathbf{A}^{(n)} = \mathbf{U}^{(n)}\mathbf{\Sigma}^{(n)}\mathbf{V}^{(n)\intercal}$. For any $\lambda \geq 0$, the λ-ridge leverage scores of $\mathbf{K}$ are*

$$\ell^{\lambda}_{(i_1,\ldots,i_N)}(\mathbf{K}) = \sum_{\mathbf{t}\in T} \frac{\prod_{n=1}^N \sigma_{t_n}^2(\mathbf{A}^{(n)})}{\prod_{n=1}^N \sigma_{t_n}^2(\mathbf{A}^{(n)}) + \lambda} \left(\prod_{n=1}^N u^{(n)}_{i_n t_n} \right)^2, \tag{5}$$

where the sum is over $T = [R_1] \times [R_2] \times \cdots \times [R_N]$. For statistical leverage scores, this simplifies to $\ell_{(i_1,\ldots,i_N)}(\mathbf{K}) = \prod_{n=1}^N \ell_{i_n}(\mathbf{A}^{(n)})$.

Proof. For notational brevity, we prove the claim for $\mathbf{K} = \mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C}$. The order-$N$ version follows by the same argument.

First, the mixed property property of Kronecker products implies that

$$\mathbf{K}^\intercal\mathbf{K} = (\mathbf{A}^\intercal\mathbf{A}) \otimes (\mathbf{B}^\intercal\mathbf{B}) \otimes (\mathbf{C}^\intercal\mathbf{C}).$$

Let $\mathbf{A} = \mathbf{U_A}\mathbf{\Sigma_A}\mathbf{V_A^\intercal}$ be the SVD of $\mathbf{A}$ such that $\mathbf{U_A} \in \mathbb{R}^{I_1 \times I_1}$ and $\mathbf{V_A} \in \mathbb{R}^{R_1 \times R_1}$. The orthogonality of $\mathbf{U_A}$ implies that

$$\mathbf{A}^\intercal\mathbf{A} = \mathbf{V_A}\mathbf{\Sigma_A^2}\mathbf{V_A^\intercal},$$

where $\mathbf{\Sigma_A^2}$ denotes $\mathbf{\Sigma_A^\intercal}\mathbf{\Sigma_A}$. Similarly, let $\mathbf{B} = \mathbf{U_B}\mathbf{\Sigma_B}\mathbf{V_B^\intercal}$ and $\mathbf{C} = \mathbf{U_C}\mathbf{\Sigma_C}\mathbf{V_C^\intercal}$. It follows from the mixed-product property that

$$\begin{aligned}
\mathbf{K}^\intercal\mathbf{K} + \lambda\mathbf{I} &= \left(\mathbf{V_A}\mathbf{\Sigma_A^2}\mathbf{V_A^\intercal}\right) \otimes \left(\mathbf{V_B}\mathbf{\Sigma_B^2}\mathbf{V_B^\intercal}\right) \otimes \left(\mathbf{V_C}\mathbf{\Sigma_C^2}\mathbf{V_C^\intercal}\right) + \lambda\mathbf{I} \\
&= (\mathbf{V_A} \otimes \mathbf{V_B} \otimes \mathbf{V_C})\left(\mathbf{\Sigma_A^2} \otimes \mathbf{\Sigma_B^2} \otimes \mathbf{\Sigma_C^2}\right)(\mathbf{V_A^\intercal} \otimes \mathbf{V_B^\intercal} \otimes \mathbf{V_C^\intercal}) + \lambda\mathbf{I} \\
&= (\mathbf{V_A} \otimes \mathbf{V_B} \otimes \mathbf{V_C})\left(\mathbf{\Sigma_A^2} \otimes \mathbf{\Sigma_B^2} \otimes \mathbf{\Sigma_C^2} + \lambda\mathbf{I}\right)(\mathbf{V_A^\intercal} \otimes \mathbf{V_B^\intercal} \otimes \mathbf{V_C^\intercal}).
\end{aligned}$$

Since $(\mathbf{XY})^+ = \mathbf{Y}^+\mathbf{X}^+$ if $\mathbf{X}$ or $\mathbf{Y}$ is orthogonal, we have

$$\begin{aligned}
(\mathbf{K}^\intercal\mathbf{K} + \lambda\mathbf{I})^+ &= \left((\mathbf{V_A} \otimes \mathbf{V_B} \otimes \mathbf{V_C})\left(\mathbf{\Sigma_A^2} \otimes \mathbf{\Sigma_B^2} \otimes \mathbf{\Sigma_C^2} + \lambda\mathbf{I}\right)(\mathbf{V_A^\intercal} \otimes \mathbf{V_B^\intercal} \otimes \mathbf{V_C^\intercal})\right)^+ \\
&= (\mathbf{V_A^\intercal} \otimes \mathbf{V_B^\intercal} \otimes \mathbf{V_C^\intercal})^+\left(\mathbf{\Sigma_A^2} \otimes \mathbf{\Sigma_B^2} \otimes \mathbf{\Sigma_C^2} + \lambda\mathbf{I}\right)^+(\mathbf{V_A} \otimes \mathbf{V_B} \otimes \mathbf{V_C})^+ \\
&= (\mathbf{V_A} \otimes \mathbf{V_B} \otimes \mathbf{V_C})\left(\mathbf{\Sigma_A^2} \otimes \mathbf{\Sigma_B^2} \otimes \mathbf{\Sigma_C^2} + \lambda\mathbf{I}\right)^+(\mathbf{V_A^\intercal} \otimes \mathbf{V_B^\intercal} \otimes \mathbf{V_C^\intercal}).
\end{aligned}$$

Next, observe that

$$\begin{aligned}
\mathbf{K} &= (\mathbf{U_A}\mathbf{\Sigma_A}\mathbf{V_A^\intercal}) \otimes (\mathbf{U_B}\mathbf{\Sigma_B}\mathbf{V_B^\intercal}) \otimes (\mathbf{U_C}\mathbf{\Sigma_C}\mathbf{V_C^\intercal}) \\
&= (\mathbf{U_A} \otimes \mathbf{U_B} \otimes \mathbf{U_C})(\mathbf{\Sigma_A} \otimes \mathbf{\Sigma_B} \otimes \mathbf{\Sigma_C})(\mathbf{V_A^\intercal} \otimes \mathbf{V_B^\intercal} \otimes \mathbf{V_C^\intercal}).
\end{aligned}$$

Putting everything together, the λ-ridge cross leverage scores can be expressed as

$$\mathbf{K}(\mathbf{K}^\intercal\mathbf{K}+\lambda\mathbf{I})^+\mathbf{K}^\intercal = (\mathbf{U_A}\otimes\mathbf{U_B}\otimes\mathbf{U_C})\mathbf{\Lambda}(\mathbf{U_A}\otimes\mathbf{U_B}\otimes\mathbf{U_C})^\intercal, \tag{31}$$

where

$$\mathbf{\Lambda} = (\mathbf{\Sigma_A}\otimes\mathbf{\Sigma_B}\otimes\mathbf{\Sigma_C})\big(\mathbf{\Sigma_A^2}\otimes\mathbf{\Sigma_B^2}\otimes\mathbf{\Sigma_C^2}+\lambda\mathbf{I}\big)^+(\mathbf{\Sigma_A}\otimes\mathbf{\Sigma_B}\otimes\mathbf{\Sigma_C}).$$

Equation (31) is the eigendecomposition of $\mathbf{K}(\mathbf{K}^\intercal\mathbf{K}+\lambda\mathbf{I})^+\mathbf{K}^\intercal$. In particular, $\mathbf{\Lambda}\in\mathbb{R}^{I_1I_2I_3\times I_1I_2I_3}$ is a diagonal matrix of eigenvalues, where the (i_1,i_2,i_3)-th eigenvalue is

$$\lambda_{(i_1,i_2,i_3)} = \frac{\sigma_{i_1}^2(\mathbf{A})\sigma_{i_2}^2(\mathbf{B})\sigma_{i_3}^2(\mathbf{C})}{\sigma_{i_1}^2(\mathbf{A})\sigma_{i_2}^2(\mathbf{B})\sigma_{i_3}^2(\mathbf{C})+\lambda}. \tag{32}$$

The value of $\ell^\lambda_{(i_1,i_2,i_3),(j_1,j_2,j_3)}(\mathbf{K})$ follows from the definition of cross λ-ridge leverage scores in Equation (4).

Finally, the statistical leverage score property holds because setting $\lambda=0$ gives an expression that is the product of the leverage scores of the factor matrices. □

B.1 Fast Kronecker-Matrix Multiplication

Lemma 4.4. *Let $\mathbf{A}^{(n)}\in\mathbb{R}^{I_n\times J_n}$, for $n\in[N]$, and $\mathbf{B}\in\mathbb{R}^{J_1\cdots J_N\times K}$. There is an algorithm* `KronMatMul`$([\mathbf{A}^{(1)},\ldots,\mathbf{A}^{(N)}],\mathbf{B})$ *that computes $(\mathbf{A}^{(1)}\otimes\mathbf{A}^{(2)}\otimes\cdots\otimes\mathbf{A}^{(N)})\mathbf{B}\in\mathbb{R}^{(I_1\cdots I_N)\times K}$ in $O(K\sum_{n=1}^N J_1\cdots J_nI_n\cdots I_N)$ time.*

Proof. We prove the claim by induction on N. Our approach will be to show that we can extract the rightmost factor matrix out of the Kronecker product and solve a smaller instance recursively.

If $N=1$, this is a standard instance of matrix-matrix multiplication that takes $O(I_1J_1K)$ time to solve. Now let

$$\mathbf{X} = \mathbf{A}^{(1)}\otimes\cdots\otimes\mathbf{A}^{(N)}\in\mathbb{R}^{P\times Q}$$

and

$$\mathbf{Y} = \mathbf{A}^{(N+1)}\in\mathbb{R}^{R\times S}$$

Let $\mathbf{b}\in\mathbb{R}^{QS}$ be an arbitrary column of $\mathbf{B}$. We compute each of these K matrix-vector products separately. Now we show how to efficiently compute $\mathbf{c}=(\mathbf{X}\otimes\mathbf{Y})\mathbf{b}\in\mathbb{R}^{PR}$. The entry in $\mathbf{c}$ at the canonical index (p,r) is

$$c_{pr} = (\mathbf{x}_{p,:}\otimes\mathbf{y}_{r,:})\mathbf{b}.$$

Writing this out, we have

$$\begin{aligned} c_{pr} &= \sum_{q=1}^{Q}\sum_{s=1}^{S} x_{p,q}y_{r,s}b_{qs} \\ &= \sum_{q=1}^{Q} x_{p,q}\sum_{s=1}^{S} y_{r,s}b_{qs}. \end{aligned}$$

Therefore, for each (q,r) we can precompute

$$z_{q,r} = \sum_{s=1}^{S} y_{r,s}b_{qs}.$$

Computing all of $\mathbf{Z}\in\mathbb{R}^{Q\times R}$ takes $O(QRS)$ time. Now that we have $\mathbf{Z}$, we can compute the output $\mathbf{c}$ as:

$$\begin{aligned} c_{pr} &= \sum_{q=1}^{Q} x_{p,q}\sum_{s=1}^{S} y_{r,s}b_{qs} \\ &= \sum_{q=1}^{Q} x_{p,q}z_{q,r}. \end{aligned}$$

Therefore, we can write a natural matricized version of $\mathbf{c}$ as

$$\mathbf{C} = \mathbf{XZ} \in \mathbb{R}^{P \times R}.$$

This matrix $\mathbf{C}$ can be computed recursively since $\mathbf{X}$ is a Kronecker product. Translating back to the original dimensions as stated in the lemma, we have $P = I_1 \cdots I_N$, $Q = J_1 \cdots J_N$, $R = I_{N+1}$, and $S = J_{N+1}$. Computing $\mathbf{Z}$ takes time

$$O(QRS) = O(J_1 \cdots J_N I_{N+1} J_{N+1}) = O(J_1 \cdots J_{N+1} I_{N+1}).$$

By induction, the recursive solve for $\mathbf{XZ}$ takes time

$$O\left(I_{N+1} \sum_{n=1}^{N} J_1 \cdots J_n I_n \cdots I_N\right).$$

Adding the two running times together and accounting for all $\mathbf{K}$ columns of $\mathbf{B}$ gives us a total running time of

$$O\left(K\left(J_1 \cdots J_{N+1} I_{N+1} + I_{N+1} \sum_{n=1}^{N} J_1 \cdots J_n I_n \cdots I_N\right)\right) = O\left(K \sum_{n=1}^{N+1} J_1 \cdots J_n I_n \cdots I_{N+1}\right),$$

which completes the proof. □

Lemma B.1 (Zhang and Ding [66])**.** *For a matrix $\mathbf{C} = [\mathbf{c}_1, \ldots, \mathbf{c}_q] \in \mathbb{R}^{p \times q}$, let*

$$\text{vec}(\mathbf{C}) = \begin{bmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_q \end{bmatrix} \in \mathbb{R}^{pq}.$$

Let $\mathbf{A} \in \mathbb{R}^{m \times q}$ and $\mathbf{B} \in \mathbb{R}^{n \times p}$. Then $\text{vec}(\mathbf{BCA}^\intercal) = (\mathbf{A} \otimes \mathbf{B})\text{vec}(\mathbf{C})$.

Theorem B.2. *Let $\mathbf{A}_1 \in \mathbb{R}^{R_1 \times R_1}, \ldots, \mathbf{A}_N \in \mathbb{R}^{R_N \times R_N}$, and $\mathbf{c} \in \mathbb{R}^{R_1 \cdots R_N}$. Let $R = \prod_{n=1}^{N} R_n$. Then $(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_N)\mathbf{c}$ can be computed in time $O(R \sum_{n=1}^{N} R_n)$.*

Proof. Let $\mathbf{C}$ be an $R_N \times (R_1 \cdots R_{N-1})$ matrix such that $\mathbf{c} = \text{vec}(\mathbf{C})$ (see Lemma B.1). For $n \in [N]$, let $\mathbf{I}_n$ be the identity matrix of size $R_n \times R_n$. Then by Lemma B.1,

$$\begin{aligned}(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_N)\mathbf{c} &= (\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_N)\text{vec}(\mathbf{C}) \\ &= \text{vec}(\mathbf{I}_N \mathbf{A}_N \mathbf{C}(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_{N-1})^\intercal) \\ &= (\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_{N-1} \otimes \mathbf{I}_N)\text{vec}(\mathbf{A}_N \mathbf{C}) \end{aligned} \tag{33}$$

Because $\mathbf{A}_N$ is $R_N \times R_N$ and $\mathbf{C}$ is $R_N \times (R_1 \cdots R_{N-1})$, $\mathbf{A}_N \mathbf{C}$ can be computed in time $O(R_N R)$.

Now note that although Kronecker product is not commutative, $\mathbf{A} \otimes \mathbf{B}$ and $\mathbf{B} \otimes \mathbf{A}$ are permutation equivalent, i.e., there are permutation matrices that transform one to the other. Therefore, instead of computing $(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_{N-1} \otimes \mathbf{I}_N)\text{vec}(\mathbf{A}_N \mathbf{C})$, we can compute $(\mathbf{I}_N \otimes \mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_{N-1})\mathbf{c}_1$ where $\mathbf{c}_1$ is a permutation of $\text{vec}(\mathbf{A}_N \mathbf{C})$. We proceed with this multiplication using a technique similar to (33) which results in a cost of $O(R_{N-1} R)$. We continue until all the matrices in the Kronecker part are the identity. Then we can return a permutation of the final vector because identity multiplied by a vector is the vector itself. □

Theorem 4.5. *Let $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, for $n \in [N]$, $I = I_1 \cdots I_N$, $R = R_1 \cdots R_N$, $\mathbf{b} \in \mathbb{R}^I$, $\mathbf{c} \in \mathbb{R}^R$, and $\mathbf{S} \in \mathbb{R}^{I \times I}$ be a diagonal matrix with $\tilde{O}(R\varepsilon^{-1})$ nonzeros. The vectors*

$$(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_N)^\intercal \mathbf{S}\mathbf{b} \quad \textit{and} \quad \mathbf{S}(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_N)\mathbf{c}$$

can be computed in time $\tilde{O}(\min_{T \subseteq [N]} \text{MM}(\prod_{n \in T} R_n, R\varepsilon^{-1}, \prod_{n \notin T} R_n))$.

Proof. First note that although Kronecker product is not commutative, $\mathbf{A} \otimes \mathbf{B}$ and $\mathbf{B} \otimes \mathbf{A}$ are permutation equivalent, i.e., there are permutation matrices that transform one to the other. Therefore, without loss of generality we assume the minimized of $\min_{T \subseteq [N]} \text{MM}(\prod_{n \in T} R_n, R/\varepsilon, \prod_{n \notin T} R_n)$ is the set $[k]$ where $1 \leq k \leq N$. For a diagonal matrix $\mathbf{S}$ let S be the set corresponding to the indices

of the nonzero entries of $\mathbf{S}$, let $\mathbf{I}_S$ be a diagonal matrix where an entry is equal to one if its index is in S and it is zero otherwise.

Note that because $\mathbf{S}$ is an $(I_1 \cdots I_N) \times (I_1 \cdots I_N)$ matrix, each element of S (or nonzero entry of $\mathbf{S}$) is corresponding to a tuple $(i_1, \ldots, i_N) \in [I_1] \times \cdots \times [I_N]$. Let

$$S_1 = \{(i_1, \ldots, i_k) : \exists i_{k+1} \in [I_{k+1}], \ldots, i_N \in [I_N] \text{ such that } (i_1, \ldots, i_N) \in S\}$$
$$S_2 = \{(i_{k+1}, \ldots, i_N) : \exists i_1 \in [I_1], \ldots, i_k \in [I_k] \text{ such that } (i_1, \ldots, i_N) \in S\}$$

Let $\mathbf{B}_S$ be an $(I_1 \cdots I_k) \times (I_{k+1} \cdots I_N)$ matrix such that $Sb = \text{vec}(\mathbf{B}_S)$ (see Lemma B.1). Then by Lemma B.1,

$$\begin{aligned}(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_N)^\intercal \mathbf{S} b &= (\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_N)^\intercal \mathbf{I}_S \mathbf{S} b \\ &= (\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_N)^\intercal (\mathbf{I}_{S_1} \otimes \mathbf{I}_{S_2}) \mathbf{S} b \\ &= ((\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_k)^\intercal \mathbf{I}_{S_1}) \otimes ((\mathbf{A}_{k+1} \otimes \cdots \otimes \mathbf{A}_N)^\intercal \mathbf{I}_{S_2}) \text{vec}(\mathbf{B}_S) \\ &= \text{vec}\left(((\mathbf{A}_{k+1} \otimes \cdots \otimes \mathbf{A}_N)^\intercal \mathbf{I}_{S_2}) \mathbf{B}_S (\mathbf{I}_{S_1} (\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_k))\right).\end{aligned}$$

Note that the number of nonzero entries of $\mathbf{B}_S$ is equal to the number of nonzero entries of $\mathbf{S}$ which is $O(R/\varepsilon)$. Therefore $((\mathbf{A}_{k+1} \otimes \cdots \otimes \mathbf{A}_N)^\intercal \mathbf{I}_{S_2}) \mathbf{B}_S$ can be computed in $O(\frac{R}{\varepsilon} \prod_{n=k+1}^N R_n)$ because $(\mathbf{A}_{k+1} \otimes \cdots \otimes \mathbf{A}_N)^\intercal \mathbf{I}_{S_2}$ has $\prod_{n=k+1}^N R_n$ rows and $\mathbf{B}_S$ has $O(R/\varepsilon)$ nonzero entries. Moreover,

$$O\left(\frac{R}{\varepsilon} \prod_{n=k+1}^N R_n\right) = O\left(\text{MM}\left(1, \frac{R}{\varepsilon}, \prod_{n=k+1}^N R_n\right)\right) = O\left(\text{MM}\left(\prod_{n=1}^k R_n, \frac{R}{\varepsilon}, \prod_{n=k+1}^N R_n\right)\right).$$

Now note that $|S_1| \leq |S| = O(R/\varepsilon)$. Therefore multiplying $((\mathbf{A}_{k+1} \otimes \cdots \otimes \mathbf{A}_N)^\intercal \mathbf{I}_{S_2}) \mathbf{B}_S$ with $(\mathbf{I}_{S_1}(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_k))$ can be done in time $O(\text{MM}(\prod_{n=1}^k R_n, \frac{R}{\varepsilon}, \prod_{n=k+1}^N R_n))$ because $((\mathbf{A}_{k+1} \otimes \cdots \otimes \mathbf{A}_N)^\intercal \mathbf{I}_{S_2}) \mathbf{B}_S$ has $\prod_{n=k+1}^N R_n$ rows and $\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_k$ has $\prod_{n=1}^k R_n$ columns.

Now let $\mathbf{C}$ be an $(R_{k+1} \cdots R_N) \times (R_1 \cdots R_k)$ matrix such that $c = \text{vec}(\mathbf{C})$ (see Lemma B.1). Then we have

$$\begin{aligned}\mathbf{S}(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_N) c &= \mathbf{S}\mathbf{I}_S (\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_N) c \\ &= \mathbf{S}(\mathbf{I}_{S_1} \otimes \mathbf{I}_{S_2})(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_N) c \\ &= \mathbf{S}(\mathbf{I}_{S_1}(\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_k)) \otimes (\mathbf{I}_{S_2}(\mathbf{A}_{k+1} \otimes \cdots \otimes \mathbf{A}_N)) \text{vec}(\mathbf{C}) \\ &= \mathbf{S}\text{vec}((\mathbf{I}_{S_2}(\mathbf{A}_{k+1} \otimes \cdots \otimes \mathbf{A}_N)) C ((\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_k)^\intercal \mathbf{I}_{S_1}))\end{aligned}$$

We have $|S_2| \leq |S| = O(R/\varepsilon)$. Therefore $\mathbf{I}_{S_2}(\mathbf{A}_{k+1} \otimes \cdots \otimes \mathbf{A}_N)$ has $O(R/\varepsilon)$ nonzero entries. Moreover C is an $(R_{k+1} \cdots R_N) \times (R_1 \cdots R_k)$ matrix. Hence $(\mathbf{I}_{S_2}(\mathbf{A}_{k+1} \otimes \cdots \otimes \mathbf{A}_N))C$ can be computed in time $O(\text{MM}(\frac{R}{\varepsilon}, \prod_{n=k+1}^N R_n, \prod_{n=1}^k R_n)) = O(MM(\prod_{n=1}^k R_n, \frac{R}{\varepsilon}, \prod_{n=k+1}^N R_n))$.

Now note that we do not need to compute all entries of

$$(\mathbf{I}_{S_2}(\mathbf{A}_{k+1} \otimes \cdots \otimes \mathbf{A}_N)) C ((\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_k)^\intercal \mathbf{I}_{S_1}).$$

We only need to compute entries corresponding to nonzero entries of $\mathbf{S}$. Computing each such entry takes $O(\prod_{n=1}^k R_n)$ time because $(\mathbf{I}_{S_2}(\mathbf{A}_{k+1} \otimes \cdots \otimes \mathbf{A}_N))C$ and $((\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_k)^\intercal \mathbf{I}_{S_1})$ have $\prod_{n=1}^k R_n$ columns and rows, respectively. Moreover the number of nonzero entries of $\mathbf{S}$ is $\tilde{O}(R/\varepsilon)$. Therefore computing all entries of $(\mathbf{I}_{S_2}(\mathbf{A}_{k+1} \otimes \cdots \otimes \mathbf{A}_N))C((\mathbf{A}_1 \otimes \cdots \otimes \mathbf{A}_k)^\intercal \mathbf{I}_{S_1})$ that are corresponding to nonzero entries of $\mathbf{S}$ takes $O(\frac{R}{\varepsilon} \prod_{n=1}^k R_n) = O(\text{MM}(\prod_{n=1}^k, \frac{R}{\varepsilon}, 1)) = O(\text{MM}(\prod_{n=1}^k R_n, \frac{R}{\varepsilon}, \prod_{n=k+1}^N R_n))$ time. □

B.2 Main Algorithm

Lemma B.3 (Johnson–Lindenstrauss random projection [32, 5])**.** *Let $\mathbf{x} \in \mathbb{R}^d$. Assume the entries in $\mathbf{G} \in \mathbb{R}^{r \times d}$ are sampled independently from $N(0, 1)$. Then,*

$$\Pr\left((1-\varepsilon)\|\mathbf{x}\|_2^2 \leq \left\|\frac{1}{\sqrt{r}}\mathbf{G}\mathbf{x}\right\|_2^2 \leq (1+\varepsilon)\|\mathbf{x}\|_2^2\right) \geq 1 - 2e^{-(\varepsilon^2 - \varepsilon^3)r/4}.$$

Lemma B.4. *Let $\mathbf{A} \in \mathbb{R}^{n\times d}$ and $0 < \varepsilon \leq 1/4$. Given $\tilde{\mathbf{A}} \in \mathbb{R}^{k\times d}$ and $\mathbf{N} = \tilde{\mathbf{A}}^\intercal\tilde{\mathbf{A}} \in \mathbb{R}^{d\times d}$ such that*

$$\mathbf{A}^\intercal\mathbf{A} \preccurlyeq \tilde{\mathbf{A}}^\intercal\tilde{\mathbf{A}} \preccurlyeq (1+\varepsilon/4)\mathbf{A}^\intercal\mathbf{A},$$

with high probability all leverage scores of $\mathbf{A}$ can be computed to $(1+\varepsilon/2)$ approximation in $\tilde{O}(\text{nnz}(\mathbf{A}) + kd + d^\omega)$ time.

Proof. Let $\mathbf{M} = (1+\varepsilon/6)(\tilde{\mathbf{A}}^\intercal\tilde{\mathbf{A}})^{-1}$. Therefore, $(\mathbf{A}^\intercal\mathbf{A})^{-1} \preccurlyeq \mathbf{M} \preccurlyeq (1+\varepsilon/6)(\mathbf{A}^\intercal\mathbf{A})^{-1}$. Hence, for any $\mathbf{x} \in \mathbb{R}^d$, we have

$$\mathbf{x}^\intercal(\mathbf{A}^\intercal\mathbf{A})^{-1}\mathbf{x} \preccurlyeq \mathbf{x}^\intercal\mathbf{M}\mathbf{x} \preccurlyeq (1+\varepsilon/6)\mathbf{x}^\intercal(\mathbf{A}^\intercal\mathbf{A})^{-1}\mathbf{x}.$$

Now note that $\mathbf{M} = \mathbf{M}\mathbf{M}^{-1}\mathbf{M} = \frac{1}{1+\varepsilon/6}\mathbf{M}\tilde{\mathbf{A}}^\intercal\tilde{\mathbf{A}}\mathbf{M}$. Hence

$$\mathbf{x}^\intercal\mathbf{M}\mathbf{x} = \frac{1}{1+\varepsilon/6}\mathbf{x}^\intercal\mathbf{M}\tilde{\mathbf{A}}^\intercal\tilde{\mathbf{A}}\mathbf{M}\mathbf{x} = \frac{1}{1+\varepsilon/6}\|\tilde{\mathbf{A}}\mathbf{M}\mathbf{x}\|_2^2.$$

Using Lemma B.3 with $\varepsilon/20$ and $r = O(\log n)$, we can compute a random matrix $\mathbf{G}$ such that with high probability for all $\mathbf{a}_i$, we have

$$\|\tilde{\mathbf{A}}\mathbf{M}\mathbf{a}_i\|_2^2 \leq \frac{1}{1-\varepsilon/20}\|\mathbf{G}\tilde{\mathbf{A}}\mathbf{M}\mathbf{a}_i\|_2^2 \leq \frac{1+\varepsilon/20}{1-\varepsilon/20}\|\tilde{\mathbf{A}}\mathbf{M}\mathbf{a}_i\|_2^2 \leq (1+\varepsilon/6)\|\tilde{\mathbf{A}}\mathbf{M}\mathbf{a}_i\|_2^2.$$

Combining the above, we have

$$\mathbf{a}_i^\intercal(\mathbf{A}^\intercal\mathbf{A})^{-1}\mathbf{a}_i \preccurlyeq \mathbf{a}_i^\intercal\mathbf{M}\mathbf{a}_i \preccurlyeq \frac{1}{(1+\varepsilon/6)(1-\varepsilon/10)}\|\mathbf{G}\tilde{\mathbf{A}}\mathbf{M}\mathbf{a}_i\|_2^2,$$

and

$$\frac{1}{(1+\varepsilon/6)(1-\varepsilon/10)}\|\mathbf{G}\tilde{\mathbf{A}}\mathbf{M}\mathbf{a}_i\|_2^2 \preccurlyeq (1+\varepsilon/6)\mathbf{a}_i^\intercal\mathbf{M}\mathbf{a}_i \preccurlyeq (1+\varepsilon/6)^2\mathbf{a}_i^\intercal(\mathbf{A}^\intercal\mathbf{A})^{-1}\mathbf{a}_i.$$

Therefore, $\frac{1}{(1+\varepsilon/6)(1-\varepsilon/10)}\|\mathbf{G}\tilde{\mathbf{A}}\mathbf{M}\mathbf{a}_i\|_2^2$ is a $(1+\varepsilon/6)^2 \leq 1+\varepsilon/2$ approximation of the leverage score of $\mathbf{a}_i$.

Lastly, we discuss the running time. Note that given $\mathbf{N}$, we can compute $\mathbf{M}$ in $\tilde{O}(d^\omega)$ time. Moreover, since $\mathbf{G}$ has $\tilde{O}(1)$ rows, $\mathbf{G}\tilde{\mathbf{A}}$ and $\mathbf{G}\tilde{\mathbf{A}}\mathbf{M}$ can be computed in $\tilde{O}(kd+d^2)$ time. Finally, given $\mathbf{G}\tilde{\mathbf{A}}\mathbf{M}$, $\mathbf{G}\tilde{\mathbf{A}}\mathbf{M}\mathbf{a}_i$, for all $i \in [n]$, can be computed in $\tilde{O}(\text{nnz}(\mathbf{A}))$. □

Theorem 4.6. *For any Kronecker product matrix $\mathbf{K} = \mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(N)} \in \mathbb{R}^{I_1\cdots I_N \times R_1\cdots R_N}$, $\mathbf{b} \in \mathbb{R}^{I_1\cdots I_N}$, $\lambda \geq 0$, $\varepsilon \in (0, 1/4]$, and $\delta > 0$,* `FastKroneckerRegression` *returns $\mathbf{x}^* \in \mathbb{R}^{R_1\cdots R_N}$ in*

$$\tilde{O}\Big(\textstyle\sum_{n=1}^N\big(\text{nnz}(\mathbf{A}^{(n)}) + R_n^\omega N^2\varepsilon^{-2}\big) + \min_{S\subseteq[N]} \text{MM}\Big(\prod_{n\in S} R_n, R\varepsilon^{-1}, \prod_{n\in[N]\setminus S} R_n\Big)\Big),$$

time such that, with probability at least $1-\delta$,

$$\|\mathbf{K}\mathbf{x}^* - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_2^2 \leq (1+\varepsilon)\min_{\mathbf{x}}\|\mathbf{K}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_2^2.$$

Proof. By [15, Lemma 8], a spectral approximation $\tilde{\mathbf{A}}^{(n)}$ (with $\tilde{O}(R_nN^2\varepsilon^{-2})$ rows) of $\mathbf{A}^{(n)}$ can be computed in $\tilde{O}(\text{nnz}(\mathbf{A}^{(n)}) + R_n^\omega N^2\varepsilon^{-2})$ time because for any $0 < x < 1$, $\frac{x}{2} \leq \log(1+x) \leq x$. Given $\tilde{\mathbf{A}}^{(n)}$, we compute $\tilde{\mathbf{A}}^{(n)\intercal}\tilde{\mathbf{A}}^{(n)}$ in $\tilde{O}(R_n^\omega N^2\varepsilon^{-2})$ time. Finally, given $\tilde{\mathbf{A}}^{(n)\intercal}\tilde{\mathbf{A}}^{(n)} \in \mathbb{R}^{R_n\times R_n}$, its SVD can be computed in time $\tilde{O}(R_n^\omega)$.

Given $\tilde{\mathbf{A}}^{(n)}$ and $\tilde{\mathbf{A}}^{(n)\intercal}\tilde{\mathbf{A}}^{(n)}$, by Lemma B.4, it takes $\tilde{O}(\text{nnz}(\mathbf{A}^{(n)}) + R_n^\omega N^2\varepsilon^{-2})$ time to compute the (approximate) leverage scores $\boldsymbol{\ell}(\mathbf{A}^{(n)})$ because for any $0 < x < 1$, $x/2 \leq \log(1+x) \leq x$. Compute the cumulative density function of each leverage score distribution in $O(I_n)$ time. This allows us to sample from the product distribution $\mathcal{P} = \boldsymbol{\ell}(\mathbf{A}^{(1)}) \otimes \cdots \otimes \boldsymbol{\ell}(\mathbf{A}^{(N)})$ in $\tilde{O}(N)$ by sampling each coordinate independently with a binary search. Sampling from $\mathcal{P}$ is equivalent to sampling from $\boldsymbol{\ell}(\mathbf{K})$ by Lemma 4.1.

Note that

$$(1+\log(1+\varepsilon/4)/N)^N \le (e^{\log(1+\varepsilon/4)/N})^N = e^{\log(1+\varepsilon/4)} = 1+\varepsilon/4.$$

Therefore, because for all $n \in [N]$, we have

$$\mathbf{A}^{(n)\intercal}\mathbf{A}^{(n)} \preccurlyeq \tilde{\mathbf{A}}^{(n)\intercal}\tilde{\mathbf{A}}^{(n)} \preccurlyeq (1+\log(1+\varepsilon/4)/N)\mathbf{A}^{(n)\intercal}\mathbf{A}^{(n)},$$

it follows that

$$\begin{aligned}(\mathbf{A}^{(1)\intercal}\mathbf{A}^{(1)}) \otimes \cdots \otimes (\mathbf{A}^{(N)\intercal}\mathbf{A}^{(N)}) &\preccurlyeq (\tilde{\mathbf{A}}^{(1)\intercal}\tilde{\mathbf{A}}^{(1)}) \otimes \cdots \otimes (\tilde{\mathbf{A}}^{(N)\intercal}\tilde{\mathbf{A}}^{(N)}) \\ &\preccurlyeq (1+\varepsilon/4)(\mathbf{A}^{(1)\intercal}\mathbf{A}^{(1)}) \otimes \cdots \otimes (\mathbf{A}^{(N)\intercal}\mathbf{A}^{(N)}).\end{aligned}$$

Thus, the approximate leverage scores we get in Algorithm 2 for $\mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(N)}$ are within a factor of $(1+\varepsilon/4)$ of the true leverage scores.

Therefore our preconditioner given by the SVD, $\mathbf{V}_\mathbf{K}(\mathbf{\Sigma}_\mathbf{K}^\intercal\mathbf{\Sigma}_\mathbf{K} + \lambda\mathbf{I}_R)^+\mathbf{V}_\mathbf{K}^\intercal$, is a spectral approximation of $(\mathbf{K}^\intercal\mathbf{K} + \lambda\mathbf{I}_R)^+$. More specifically

$$\begin{aligned}\tilde{\mathbf{K}}^\intercal\tilde{\mathbf{K}} + \lambda\mathbf{I} &\preccurlyeq \frac{1}{1-\sqrt{\varepsilon}}(\mathbf{K}^\intercal\mathbf{K} + \lambda\mathbf{I}_R) \preccurlyeq \frac{1}{1-\sqrt{\varepsilon}}\mathbf{V}_\mathbf{K}(\mathbf{\Sigma}_\mathbf{K}^\intercal\mathbf{\Sigma}_\mathbf{K} + \lambda\mathbf{I}_R)^+\mathbf{V}_\mathbf{K}^\intercal \\ &\preccurlyeq \frac{1+\varepsilon/4}{1-\sqrt{\varepsilon}}(\mathbf{K}^\intercal\mathbf{K} + \lambda\mathbf{I}_R) \preccurlyeq \frac{(1+\varepsilon/4)(1+\sqrt{\varepsilon})}{1-\sqrt{\varepsilon}}\Big(\tilde{\mathbf{K}}^\intercal\tilde{\mathbf{K}} + \lambda\mathbf{I}\Big).\end{aligned}$$

Therefore by $O(\log(1/\varepsilon))$ iterations of Richardson (Lemma 4.2), we converge to the desired accuracy. Finally note that each iteration of Richardson can be done in time

$$\tilde{O}\left(\min_{S\subseteq[N]} \mathrm{MM}\left(\prod_{n\in S} R_n, R\varepsilon^{-1}, \prod_{n\in[N]\setminus S} R_n\right)\right),$$

using our novel sparse Kronecker-matrix multiplication procedure (Theorem 4.5), `KronMatMul` (Lemma 4.4), and the structure of the preconditioner which is a diagonal matrix multiplied from left and right by matrices with Kronecker structure. □

C Missing Analysis from Section 5

In this section we show how to extend the `FastKroneckerRegression` (Algorithm 2) to work for factor matrix updates in `TuckerALS`. This analysis is presented in Appendix C.1 and Appendix C.2. Then we compare the running times of different core tensor and factor matrix update algorithms in Appendix C.3 to establish baselines.

C.1 Factor Matrix Update Substitution: Reducing to Equality-Constrained Least Squares

Lemma 5.1. *Let $\mathbf{A} \in \mathbb{R}^{n\times m}$, $\mathbf{M} \in \mathbb{R}^{m\times d}$, $\mathbf{b} \in \mathbb{R}^n$, and $\lambda \ge 0$. For any ridge regression problem of the form $\arg\min_{\mathbf{x}\in\mathbb{R}^d}(\|\mathbf{AMx} - \mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_2^2)$, we can solve*

$$\mathbf{z}_{\text{opt}} = \operatorname*{arg\,min}_{\mathbf{Nz}=\mathbf{0}} \|\mathbf{Az} - \mathbf{b}\|_2^2 + \lambda\left\|\mathbf{M}^+\mathbf{z}\right\|_2^2,$$

where $\mathbf{N} = \mathbf{I}_m - \mathbf{MM}^+$, and return vector $\mathbf{M}^+\mathbf{z}_{\text{opt}}$ instead.

Lemma 5.2 (Approximate equality-constrained regression)**.** *Let $\mathbf{M} \in \mathbb{R}^{n\times d}$, $\mathbf{N} \in \mathbb{R}^{m\times d}$, $\mathbf{b} \in \mathbb{R}^n$, and $0 < \varepsilon < 1/3$. To solve $\min_{\mathbf{Nx}=\mathbf{0}}\|\mathbf{Mx} - \mathbf{b}\|_2^2$ to a $(1+\varepsilon)$-approximation, it suffices to solve*

$$\min_{\mathbf{x}\in\mathbb{R}^d} \left\|\begin{bmatrix}\mathbf{M}\\ \sqrt{w}\mathbf{N}\end{bmatrix}\mathbf{x} - \begin{bmatrix}\mathbf{b}\\ \mathbf{0}\end{bmatrix}\right\|_2^2$$

to a $(1+\varepsilon/3)$-approximation with $w \ge (1+12/\varepsilon)\|\mathbf{MN}^+\|_2^2$.

Proof. First note that for any $w \geq 0$, we have

$$\min_{\mathbf{x}\in\mathbb{R}^d} \left\| \begin{bmatrix} \mathbf{M} \\ \sqrt{w}\mathbf{N} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} \right\|_2^2 \leq \min_{\mathbf{N}\mathbf{x}=\mathbf{0}} \left\| \begin{bmatrix} \mathbf{M} \\ \sqrt{w}\mathbf{N} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} \right\|_2^2 = \min_{\mathbf{N}\mathbf{x}=\mathbf{0}} \|\mathbf{M}\mathbf{x} - \mathbf{b}\|_2^2. \tag{34}$$

Suppose $\hat{\mathbf{x}} \in \mathbb{R}^d$ such that

$$\left\| \begin{bmatrix} \mathbf{M} \\ \sqrt{w}\mathbf{N} \end{bmatrix} \hat{\mathbf{x}} - \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} \right\|_2^2 \leq (1+\varepsilon/3) \min_{\mathbf{x}\in\mathbb{R}^d} \left\| \begin{bmatrix} \mathbf{M} \\ \sqrt{w}\mathbf{N} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} \right\|_2^2. \tag{35}$$

Let $\mathbf{z} = (\mathbf{I} - \mathbf{N}^+\mathbf{N})\hat{\mathbf{x}}$. It follows that $\mathbf{N}\mathbf{z} = \mathbf{0}$ because $\mathbf{N} = \mathbf{N}\mathbf{N}^+\mathbf{N}$. Therefore,

$$\left\| \begin{bmatrix} \mathbf{M} \\ \sqrt{w}\mathbf{N} \end{bmatrix} \mathbf{z} - \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} \right\|_2^2 = \left\| \begin{bmatrix} \mathbf{M} \\ \sqrt{w}\mathbf{N} \end{bmatrix} (\mathbf{I} - \mathbf{N}^+\mathbf{N})\hat{\mathbf{x}} - \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} \right\|_2^2 = \|\mathbf{M}(\mathbf{I} - \mathbf{N}^+\mathbf{N})\hat{\mathbf{x}} - \mathbf{b}\|_2^2.$$

By the triangle inequality,

$$\|\mathbf{M}(\mathbf{I} - \mathbf{N}^+\mathbf{N})\hat{\mathbf{x}} - \mathbf{b}\|_2 \leq \|\mathbf{M}\hat{\mathbf{x}} - \mathbf{b}\|_2 + \|\mathbf{M}\mathbf{N}^+\mathbf{N}\hat{\mathbf{x}}\|_2.$$

Therefore,

$$\|\mathbf{M}(\mathbf{I} - \mathbf{N}^+\mathbf{N})\hat{\mathbf{x}} - \mathbf{b}\|_2^2 \leq \|\mathbf{M}\hat{\mathbf{x}} - \mathbf{b}\|_2^2 + \|\mathbf{M}\mathbf{N}^+\mathbf{N}\hat{\mathbf{x}}\|_2^2 + 2\|\mathbf{M}\hat{\mathbf{x}} - \mathbf{b}\|_2\|\mathbf{M}\mathbf{N}^+\mathbf{N}\hat{\mathbf{x}}\|_2.$$

Now we have two cases:

- Case 1: $2\|\mathbf{M}\mathbf{N}^+\mathbf{N}\hat{\mathbf{x}}\|_2 \leq \frac{\varepsilon}{3}\|\mathbf{M}\hat{\mathbf{x}} - \mathbf{b}\|_2$.
- Case 2: $2\|\mathbf{M}\mathbf{N}^+\mathbf{N}\hat{\mathbf{x}}\|_2 > \frac{\varepsilon}{3}\|\mathbf{M}\hat{\mathbf{x}} - \mathbf{b}\|_2$.

Note that by the consistency of operator norms, we have

$$\|\mathbf{M}\mathbf{N}^+\mathbf{N}\hat{\mathbf{x}}\|_2 \leq \|\mathbf{M}\mathbf{N}^+\|_2\|\mathbf{N}\hat{\mathbf{x}}\|_2.$$

Therefore, in the first case we have

$$\begin{aligned} \|\mathbf{M}(\mathbf{I} - \mathbf{N}^+\mathbf{N})\hat{\mathbf{x}} - \mathbf{b}\|_2^2 &\leq \left(1 + \frac{\varepsilon}{3}\right)\|\mathbf{M}\hat{\mathbf{x}} - \mathbf{b}\|_2^2 + \|\mathbf{M}\mathbf{N}^+\mathbf{N}\hat{\mathbf{x}}\|_2^2 \\ &\leq \left(1 + \frac{\varepsilon}{3}\right)\left(\|\mathbf{M}\hat{\mathbf{x}} - \mathbf{b}\|_2^2 + w\|\mathbf{N}\hat{\mathbf{x}}\|_2^2\right), \end{aligned}$$

where the last inequality follows from our choice of w.

In the second case we have

$$\begin{aligned} \|\mathbf{M}(\mathbf{I} - \mathbf{N}^+\mathbf{N})\hat{\mathbf{x}} - \mathbf{b}\|_2^2 &\leq \|\mathbf{M}\hat{\mathbf{x}} - \mathbf{b}\|_2^2 + \left(1 + \frac{12}{\varepsilon}\right)\|\mathbf{M}\mathbf{N}^+\mathbf{N}\hat{\mathbf{x}}\|_2^2 \\ &\leq \|\mathbf{M}\hat{\mathbf{x}} - \mathbf{b}\|_2^2 + w\|\mathbf{N}\hat{\mathbf{x}}\|_2^2. \end{aligned}$$

Therefore, in both cases

$$\|\mathbf{M}(\mathbf{I} - \mathbf{N}^+\mathbf{N})\hat{\mathbf{x}} - \mathbf{b}\|_2^2 \leq \left(1 + \frac{\varepsilon}{3}\right)(\|\mathbf{M}\hat{\mathbf{x}} - \mathbf{b}\|_2^2 + w\|\mathbf{N}\hat{\mathbf{x}}\|_2^2). \tag{36}$$

Moreover, $\varepsilon < 1/3$, so then $(1+\varepsilon/3)^2 \leq 1+\varepsilon$. Therefore by Equations (34), (35) and (36), we have

$$\begin{aligned} \|\mathbf{M}\mathbf{z} - \mathbf{b}\|_2^2 = \|\mathbf{M}(\mathbf{I} - \mathbf{N}^+\mathbf{N})\hat{\mathbf{x}} - \mathbf{b}\|_2^2 &\leq (1+\varepsilon) \min_{\mathbf{x}\in\mathbb{R}^d} \left\| \begin{bmatrix} \mathbf{M} \\ \sqrt{w}\mathbf{N} \end{bmatrix} \mathbf{x} - \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix} \right\|_2^2 \\ &\leq (1+\varepsilon) \min_{\mathbf{N}\mathbf{x}=0} \|\mathbf{M}\mathbf{x} - \mathbf{b}\|_2^2. \end{aligned}$$

Finally, note that $\mathbf{N}\mathbf{z} = \mathbf{0}$ and that $\mathbf{z}$ is a feasible solution. □

Application to factor matrix updates. Now we explain how the reduction to equality-constrained least squares in Lemma 5.1 applies to factor matrix updates in `TuckerALS`. For the factor matrix updates, we solve a regression problem of the form:

$$\arg\min_{\mathbf{y}\in\mathbb{R}^{R_n}} \|(\mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(n-1)} \otimes \mathbf{A}^{(n+1)} \otimes \cdots \otimes \mathbf{A}^{(N)})\mathbf{G}_{(n)}^\intercal \mathbf{y} - \mathbf{b}_{i:}^\intercal\|_2^2 + \lambda\|\mathbf{y}\|_2^2, \tag{37}$$

where $\mathbf{b}_{i:}$ is the i-th row of the mode-n unfolding of tensor $\mathcal{X}$. Note that for any $\mathbf{y}$, $\mathbf{G}_{(n)}^\intercal\mathbf{y}$ is a vector in the column space of $\mathbf{G}_{(n)}^\intercal$. Thus, $\mathbf{G}_{(n)}^\intercal\mathbf{y}$ is orthogonal to any vector in the left null space of $\mathbf{G}_{(n)}^\intercal$. Let $\mathbf{N}$ be a matrix in which the rows are a basis for the left null space of $\mathbf{G}_{(n)}^\intercal$. Then, solving the following is equivalent to solving (37):

$$\min_{\mathbf{Nz}=\mathbf{0}} \|(\mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(n-1)} \otimes \mathbf{A}^{(n+1)} \otimes \cdots \otimes \mathbf{A}^{(N)})\mathbf{z} - \mathbf{b}_{i:}^\intercal\|_2^2 + \lambda\|(\mathbf{G}_{(n)}^\intercal)^+\mathbf{z}\|_2^2, \tag{38}$$

where $(\mathbf{G}_{(n)}^\intercal)^+$ is the pseudoinverse of $\mathbf{G}_{(n)}^\intercal$.

To explain the $\|(\mathbf{G}_{(n)}^\intercal)^+\mathbf{z}\|_2^2$ term, consider a vector $\mathbf{y}$ that under the transformation $\mathbf{G}_{(n)}^\intercal$ goes to $\mathbf{z}$, i.e., $\mathbf{G}_{(n)}^\intercal\mathbf{y} = \mathbf{z}$. The set of solutions to this linear system is $(\mathbf{G}_{(n)}^\intercal)^+\mathbf{z} + (\mathbf{I} - (\mathbf{G}_{(n)}^\intercal)^+\mathbf{G}_{(n)}^\intercal)\mathbf{w}$, for all $\mathbf{w}$ by [30]. Moreover, $(\mathbf{G}_{(n)}^\intercal)^+\mathbf{z}$ is orthogonal to $(\mathbf{I} - (\mathbf{G}_{(n)}^\intercal)^+\mathbf{G}_{(n)}^\intercal)\mathbf{w}$ because

$$\begin{aligned}
((\mathbf{G}_{(n)}^\intercal)^+\mathbf{z})^\intercal(\mathbf{I} - (\mathbf{G}_{(n)}^\intercal)^+\mathbf{G}_{(n)}^\intercal)\mathbf{w} &= ((\mathbf{G}_{(n)}^\intercal)^+\mathbf{G}_{(n)}^\intercal(\mathbf{G}_{(n)}^\intercal)^+\mathbf{z})^\intercal(\mathbf{I} - (\mathbf{G}_{(n)}^\intercal)^+\mathbf{G}_{(n)}^\intercal)\mathbf{w} \\
&= ((\mathbf{G}_{(n)}^\intercal)^+\mathbf{z})^\intercal(\mathbf{G}_{(n)}^\intercal)^+\mathbf{G}_{(n)}^\intercal(\mathbf{I} - (\mathbf{G}_{(n)}^\intercal)^+\mathbf{G}_{(n)}^\intercal)\mathbf{w} \\
&= ((\mathbf{G}_{(n)}^\intercal)^+\mathbf{z})^\intercal(\mathbf{G}_{(n)}^\intercal)^+(\mathbf{G}_{(n)}^\intercal - \mathbf{G}_{(n)}^\intercal(\mathbf{G}_{(n)}^\intercal)^+\mathbf{G}_{(n)}^\intercal)\mathbf{w} \\
&= 0.
\end{aligned}$$

Therefore, by the Pythagorean theorem, we have

$$\|(\mathbf{G}_{(n)}^\intercal)^+\mathbf{z} + (\mathbf{I} - (\mathbf{G}_{(n)}^\intercal)^+\mathbf{G}_{(n)}^\intercal)\mathbf{w}\|_2^2 = \|(\mathbf{G}_{(n)}^\intercal)^+\mathbf{z}\|_2^2 + \|(\mathbf{I} - (\mathbf{G}_{(n)}^\intercal)^+\mathbf{G}_{(n)}^\intercal)\mathbf{w}\|_2^2,$$

so $\|(\mathbf{G}_{(n)}^\intercal)^+\mathbf{z} + (\mathbf{I} - (\mathbf{G}_{(n)}^\intercal)^+\mathbf{G}_{(n)}^\intercal)\mathbf{w}\|_2^2$ is minimized when $\mathbf{w} = \mathbf{0}$ [54]. Thus, for all y such that $\mathbf{G}_{(n)}^\intercal\mathbf{y} = \mathbf{z}$, it follows that $(\mathbf{G}_{(n)}^\intercal)^+\mathbf{z}$ minimizes $\|\mathbf{y}\|_2^2$ in (37), hence we can replace $\mathbf{y}$ by $(\mathbf{G}_{(n)}^\intercal)^+\mathbf{z}$.

Note that $\mathbf{N} = \mathbf{I} - \mathbf{G}_{(n)}^\intercal(\mathbf{G}_{(n)}^\intercal)^+$ works because for any $\mathbf{y}$, by definition of pseudoinverse:

$$(\mathbf{I} - \mathbf{G}_{(n)}^\intercal(\mathbf{G}_{(n)}^\intercal)^+)\mathbf{G}_{(n)}^\intercal\mathbf{y} = (\mathbf{G}_{(n)}^\intercal - \mathbf{G}_{(n)}^\intercal(\mathbf{G}_{(n)}^\intercal)^+\mathbf{G}_{(n)}^\intercal)\mathbf{y} = 0.$$

More generally, a vector $\mathbf{z}$ is in the image of $\mathbf{G}_{(n)}^\intercal$ if and only if $(\mathbf{I} - \mathbf{G}_{(n)}^\intercal(\mathbf{G}_{(n)}^\intercal)^+)\mathbf{z} = \mathbf{0}$. This is an alternate formulation of Lemma 5.1 and leads to the algorithm `FastFactorMatrixUpdate` below.

C.2 Fast Factor Matrix Update Algorithm

Theorem 5.3. *For any $\lambda \geq 0$, $\varepsilon \in (0, 1/3)$, and $\delta > 0$, the* `FastFactorMatrixUpdate` *algorithm updates $\mathbf{A}_{(k)} \in \mathbb{R}^{I_k \times R_k}$ in* `TuckerALS` *with a $(1+\varepsilon)$-approximation, with probability at least $1 - \delta$, in time*

$$\tilde{O}\Big(I_k R_{\neq k}^2 \varepsilon^{-1} \log(1/\delta) + I_k R \textstyle\sum_{n=1}^N R_n + R_k^\omega \varepsilon^{-2}\Big).$$

Proof. Each factor row matrix update in `TuckerALS` (Algorithm 1) has the form

$$\mathbf{a}_{i:}^{(n)} \leftarrow \arg\min_{\mathbf{y}\in\mathbb{R}^{R_n}} \Big(\|\mathbf{K}\mathbf{G}_{(n)}^\intercal\mathbf{y} - \mathbf{b}_{i:}^\intercal\|_2^2 + \lambda\|\mathbf{y}\|_2^2\Big).$$

Use Lemma 5.1 to reduce the factor matrix updates to solving the equality-constrained Kronecker regression problem

$$\mathbf{z}_{\text{opt}} = \arg\min_{\mathbf{Nz}=\mathbf{0}} \Big(\|\mathbf{K}\mathbf{z} - \mathbf{b}_{i:}{}^\intercal\|_2^2 + \lambda\|(\mathbf{G}_{(n)}^\intercal)^+\mathbf{z}\|_2^2\Big).$$

The correctness of this algorithm is analogous to the argument in the proof of Theorem 4.6, but now we have more sophisticated blocks in the data matrix and need to account for them.

Algorithm 3 `FastFactorMatrixUpdate`

Input: Tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, factors $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$, core $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$, index n, λ, error ε, probability δ

1: Set $R_{\neq n} \leftarrow R_1 \cdots R_{n-1} R_{n+1} \cdots R_N$
2: $\mathbf{K} = \mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(n-1)} \otimes \mathbf{A}^{(n+1)} \otimes \cdots \otimes \mathbf{A}^{(N)}$
3: Set $\mathbf{B} \leftarrow \mathbf{X}_{(n)}$
4: Initialize product distribution data structure $\mathcal{P}$ to sample indices from $(\boldsymbol{\ell}(\mathbf{A}^{(1)}), \cdots, \boldsymbol{\ell}(\mathbf{A}^{(N)}))$
5: Let $\mathbf{N} = \mathbf{I}_{R_{\neq n}} - \mathbf{G}_{(n)}^\intercal (\mathbf{G}_{(n)}^\intercal)^+$
6: Let $w \geq \left(1 + \frac{12}{\varepsilon}\right) \left\| \begin{bmatrix} \mathbf{K} \\ \sqrt{\lambda}(\mathbf{G}_{(n)}^\intercal)^+ \end{bmatrix} \mathbf{N}^+ \right\|_2^2$ as in Lemma 5.2, and construct the operator

$$\mathbf{M}^+ = \left(\begin{bmatrix} \mathbf{K} \\ \sqrt{\lambda}(\mathbf{G}_{(n)}^\intercal)^+ \\ \sqrt{w}\mathbf{N} \end{bmatrix}^\intercal \begin{bmatrix} \mathbf{K} \\ \sqrt{\lambda}(\mathbf{G}_{(n)}^\intercal)^+ \\ \sqrt{w}\mathbf{N} \end{bmatrix} \right)^+$$

using the Woodbury identity in Equation (39)
7: Set $s \leftarrow \lceil 1680 R_{\neq n} \ln(40 R_{\neq n}) \ln(I_n/\delta)/\varepsilon \rceil$
8: **for** $i = 1$ to I_n **do**
9: Set $\mathbf{S} \leftarrow \texttt{SampleRows}(\mathbf{K}, s, \mathcal{P})$
10: Set $\tilde{\mathbf{K}} \leftarrow \mathbf{S}\mathbf{K}$ and $\tilde{\mathbf{b}} \leftarrow \mathbf{S}\mathbf{b}_{i:}^\intercal$
11: Initialize $\mathbf{z} \leftarrow \mathbf{0}_{R_{\neq n}}$
12: **repeat**
13: Update $\mathbf{z} \leftarrow \mathbf{z} - (1-\sqrt{\varepsilon})\mathbf{M}^+((\tilde{\mathbf{K}}^\intercal\tilde{\mathbf{K}} + w\mathbf{I})\mathbf{z} + \lambda \mathbf{G}_{(n)}^+ (\mathbf{G}_{(n)}^\intercal)^+ \mathbf{z} - w\mathbf{G}_{(n)}^+ \mathbf{G}_{(n)}\mathbf{z} - \tilde{\mathbf{K}}^\intercal\tilde{\mathbf{b}})$
using fast Kronecker-matrix multiplication
14: **until** convergence
15: Update factor matrix row $\mathbf{a}_{i:}^{(n)} \leftarrow \mathbf{z}^\intercal \mathbf{G}_{(n)}^+$

We solve each row update independently. The construction of the sketched submatrix $\tilde{\mathbf{K}}$ guarantees that $(1-\sqrt{\varepsilon})^{-1}\mathbf{M}$ is a 3-spectral approximation to the sketched normal matrix

$$\tilde{\mathbf{M}} \stackrel{\text{def}}{=} \tilde{\mathbf{K}}^\intercal\tilde{\mathbf{K}} + \lambda \mathbf{G}_{(n)}^+ (\mathbf{G}_{(n)}^\intercal)^+ + w\mathbf{N}^\intercal\mathbf{N},$$

with probability at least $1 - \delta/I_n$, by Lemma 3.3. Thus, we can use the (non-sketched) matrix $\mathbf{M}^+$ as a preconditioner and exploit its Kronecker structure since this iterative method converges in $\tilde{O}(1)$ steps by Lemma 4.2.

It remains to show the main difference with Theorem 4.6: the time complexity of one Richardson iteration (Line 13). We show in Lemma C.1 how $\mathbf{M}^+\mathbf{x}$ can be computed in time

$$\tilde{O}(R_{\neq n}^2 \varepsilon^{-1} \log(I_n/\delta) + R \sum_{k=1}^N R_k + R_n^\omega)$$

using the Woodbury matrix identity since $\mathbf{M}^+$ is a rank-R_n update to $(\mathbf{K}^\intercal\mathbf{K} + w\mathbf{I})^+$.

The solution of each sketch is a $(1+\varepsilon)$-approximation to the optimal factor row by Lemma 3.4, and the success guarantee follows from a union bound over all I_n rows. □

Lemma C.1. ***Line 13*** *in* `FastFactorMatrixUpdate` *takes* $\tilde{O}(R_{\neq n}^2 \varepsilon^{-1} \log(I_n/\delta) + R \sum_{k=1}^N R_k)$ *time after preprocessing.*

Proof. Recall that $\mathbf{N} = \mathbf{I} - \mathbf{G}_{(n)}^\intercal (\mathbf{G}_{(n)}^\intercal)^+$ and consider the following equality. Letting

$$\mathbf{M} \stackrel{\text{def}}{=} \begin{bmatrix} \mathbf{K} \\ \sqrt{\lambda}(\mathbf{G}_{(n)}^\intercal)^+ \\ \sqrt{w}\mathbf{N} \end{bmatrix}^\intercal \begin{bmatrix} \mathbf{K} \\ \sqrt{\lambda}(\mathbf{G}_{(n)}^\intercal)^+ \\ \sqrt{w}\mathbf{N} \end{bmatrix},$$

we have

$$\begin{aligned}\mathbf{M} &= \mathbf{K}^\intercal\mathbf{K} + \lambda\mathbf{G}_{(n)}^+(\mathbf{G}_{(n)}^\intercal)^+ + w\mathbf{N}^\intercal\mathbf{N} \\ &= \mathbf{K}^\intercal\mathbf{K} + \lambda\mathbf{G}_{(n)}^+(\mathbf{G}_{(n)}^\intercal)^+ + w(\mathbf{I} - \mathbf{G}_{(n)}^\intercal(\mathbf{G}_{(n)}^\intercal)^+)^\intercal(\mathbf{I} - \mathbf{G}_{(n)}^\intercal(\mathbf{G}_{(n)}^\intercal)^+) \\ &= \mathbf{K}^\intercal\mathbf{K} + \lambda\mathbf{G}_{(n)}^+(\mathbf{G}_{(n)}^\intercal)^+ + w(\mathbf{I} - \mathbf{G}_{(n)}^+\mathbf{G}_{(n)} - \mathbf{G}_{(n)}^\intercal(\mathbf{G}_{(n)}^\intercal)^+ + \mathbf{G}_{(n)}^+\mathbf{G}_{(n)}\mathbf{G}_{(n)}^\intercal(\mathbf{G}_{(n)}^\intercal)^+) \\ &= (\mathbf{K}^\intercal\mathbf{K} + w\mathbf{I}) + \lambda\mathbf{G}_{(n)}^+(\mathbf{G}_{(n)}^\intercal)^+ \\ &\quad - w(\mathbf{G}_{(n)}^+\mathbf{G}_{(n)} + \mathbf{G}_{(n)}^\intercal(\mathbf{G}_{(n)}^\intercal)^+ - \mathbf{G}_{(n)}^+\mathbf{G}_{(n)}\mathbf{G}_{(n)}^\intercal(\mathbf{G}_{(n)}^\intercal)^+).\end{aligned}$$

For any matrix, we have the pseudoinverse identity $\mathbf{A}^+\mathbf{A}\mathbf{A}^\intercal = \mathbf{A}^\intercal$, so it follows that

$$\mathbf{G}_{(n)}^+\mathbf{G}_{(n)}\mathbf{G}_{(n)}^\intercal(\mathbf{G}_{(n)}^\intercal)^+ = \mathbf{G}_{(n)}^\intercal(\mathbf{G}_{(n)}^\intercal)^+.$$

Therefore,

$$\begin{aligned}\mathbf{M} &= (\mathbf{K}^\intercal\mathbf{K} + w\mathbf{I}) + \lambda\mathbf{G}_{(n)}^+(\mathbf{G}_{(n)}^\intercal)^+ - w(\mathbf{G}_{(n)}^+\mathbf{G}_{(n)} + \mathbf{G}_{(n)}^\intercal(\mathbf{G}_{(n)}^\intercal)^+ - \mathbf{G}_{(n)}^\intercal(\mathbf{G}_{(n)}^\intercal)^+) \\ &= (\mathbf{K}^\intercal\mathbf{K} + w\mathbf{I}) + \lambda\mathbf{G}_{(n)}^+(\mathbf{G}_{(n)}^\intercal)^+ - w(\mathbf{G}_{(n)}^+\mathbf{G}_{(n)}) \\ &= (\mathbf{K}^\intercal\mathbf{K} + w\mathbf{I}) + \mathbf{G}_{(n)}^+(\lambda(\mathbf{G}_{(n)}^\intercal)^+ - w\mathbf{G}_{(n)}).\end{aligned}$$

Applying the Woodbury matrix identity, we have

$$\begin{aligned}\mathbf{M}^+ &= (\mathbf{K}^\intercal\mathbf{K} + w\mathbf{I})^{-1} \\ &\quad + (\mathbf{K}^\intercal\mathbf{K} + w\mathbf{I})^{-1}\mathbf{G}_{(n)}^+(\mathbf{I} + (\lambda(\mathbf{G}_{(n)}^\intercal)^+ - w\mathbf{G}_{(n)})(\mathbf{K}^\intercal\mathbf{K} + w\mathbf{I})^{-1}\mathbf{G}_{(n)}^+)^{-1} \\ &\quad \cdot (\lambda(\mathbf{G}_{(n)}^\intercal)^+ - w\mathbf{G}_{(n)})(\mathbf{K}^\intercal\mathbf{K} + w\mathbf{I})^{-1}.\end{aligned} \tag{39}$$

First note that

$$\mathbf{I} + (\lambda(\mathbf{G}_{(n)}^\intercal)^+ - w\mathbf{G}_{(n)})(\mathbf{K}^\intercal\mathbf{K} + w\mathbf{I})^{-1}\mathbf{G}_{(n)}^+$$

is an $R_n \times R_n$ matrix. The time complexity of computing the factored SVD of $\mathbf{K}^\intercal\mathbf{K}$ is $O(\sum_{k\neq n}(I_kR_k^2 + R_k^\omega))$. In `TuckerALS`, these are computed at the end of each factor matrix update and therefore do not need to be computed in this step. After this, multiplying a vector by $(\mathbf{K}^\intercal\mathbf{K} + w\mathbf{I})^{-1}$ can be done in time $O(R_{\neq n}\sum_{m\neq n} R_m)$ by Lemma 4.4. Therefore, computing

$$\mathbf{I} + (\lambda(\mathbf{G}_{(n)}^\intercal)^+ - w\mathbf{G}_{(n)})(\mathbf{K}^\intercal\mathbf{K} + w\mathbf{I})^{-1}\mathbf{G}_{(n)}^+$$

takes a total of $O(R\sum_{k=1}^N R_k)$ time. Computing the inverse of this matrix takes $O(R_n^\omega)$ time. Moreover, this inverse can be used for all Richardson iteration steps and row updates. Finally, observe that multiply any vector with $\mathbf{G}_{(n)}^+$ or $(\lambda(\mathbf{G}_{(n)}^\intercal)^+ - w\mathbf{G}_{(n)})$ takes $O(R)$ time. Therefore, to evaluate $\mathbf{M}^+\mathbf{z}$ for any $\mathbf{z}$, we use Equation (39) and repeatedly apply matrix-vector multiplcations from right to left. The total running time per evaluation after preprocessing is

$$O\left(R\sum_{k=1}^N R_k\right).$$

Now we show that the vector

$$\tilde{\mathbf{K}}^\intercal\tilde{\mathbf{K}}\mathbf{z} + \lambda\mathbf{G}_{(n)}^+(\mathbf{G}_{(n)}^\intercal)^+\mathbf{z} + w\mathbf{N}^\intercal\mathbf{N}\mathbf{z} - \tilde{\mathbf{K}}^\intercal\tilde{\mathbf{b}}$$

can be computed fast enough. By the same argument above, this is equivalent to

$$(\tilde{\mathbf{K}}^\intercal\tilde{\mathbf{K}} + w\mathbf{I})\mathbf{z} + \lambda\mathbf{G}_{(n)}^+(\mathbf{G}_{(n)}^\intercal)^+\mathbf{z} - w\mathbf{G}_{(n)}^+\mathbf{G}_{(n)}\mathbf{z} - \tilde{\mathbf{K}}^\intercal\tilde{\mathbf{b}}.$$

We can compute $\tilde{\mathbf{K}}^\intercal\tilde{\mathbf{b}}$ in $\tilde{O}(R_{\neq n}^2\varepsilon^{-1}\log(I_n/\delta))$ time, $\lambda\mathbf{G}_{(n)}^+(\mathbf{G}_{(n)}^\intercal)^+\mathbf{z}$ and $w\mathbf{G}_{(n)}^+\mathbf{G}_{(n)}\mathbf{z}$ take $O(R)$ time, and $(\tilde{\mathbf{K}}^\intercal\tilde{\mathbf{K}} + w\mathbf{I})\mathbf{z}$ takes $\tilde{O}(R + R_{\neq n}^2\varepsilon^{-1}\log(I_n/\delta))$. Summing all of these running times completes the proof. □

C.3 Comparison of different factor matrix and core tensor update algorithms.

In this subsection, we prove the running times for the different algorithms in Table 1.

Naive factor matrix update. Consider the n-th factor matrix $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$. First let

$$\mathbf{K} = \mathbf{G}_{(n)}(\mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(n-1)} \otimes \mathbf{A}^{(n+1)} \otimes \cdots \otimes \mathbf{A}^{(N)})^\intercal$$

and

$$\mathbf{B} = \mathbf{X}_{(n)}.$$

For $i = 1$ to I_n we want to solve:

$$\mathbf{a}_{i:}^{(n)} \leftarrow \underset{\mathbf{y} \in \mathbb{R}^{1 \times R_n}}{\arg\min} \|\mathbf{y}\mathbf{K} - \mathbf{b}_{i:}\|_2^2 + \lambda \|\mathbf{y}\|_2^2$$

We use the normal equation for ridge regression:

$$\mathbf{y}_{\text{opt}}^\intercal = (\mathbf{K}\mathbf{K}^\intercal + \lambda \mathbf{I})^+ \mathbf{K}\mathbf{b}_{i:}^\intercal.$$

Let's start by computing $\mathbf{K}\mathbf{K}^\intercal + \lambda\mathbf{I}$ efficiently:

$$\begin{aligned}
&\mathbf{K}\mathbf{K}^\intercal \\
&= \mathbf{G}_{(n)}(\mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(n-1)} \otimes \mathbf{A}^{(n+1)} \otimes \cdots \otimes \mathbf{A}^{(N)})^\intercal \\
&\cdot (\mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(n-1)} \otimes \mathbf{A}^{(n+1)} \otimes \cdots \otimes \mathbf{A}^{(N)})\mathbf{G}_{(n)}^\intercal \\
&= \mathbf{G}_{(n)}\Big(\mathbf{A}^{(1)\intercal}\mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(n-1)\intercal}\mathbf{A}^{(n-1)} \otimes \mathbf{A}^{(n+1)\intercal}\mathbf{A}^{(n+1)} \otimes \cdots \otimes \mathbf{A}^{(N)\intercal}\mathbf{A}^{(N)}\Big)\mathbf{G}_{(n)}^\intercal.
\end{aligned} \tag{40}$$

The internal terms of the form $\mathbf{A}^{(k)\intercal}\mathbf{A}^{(k)}$ can be computed in $O(R_k^2 I_k)$ time using $O(R_k^2)$ space. Therefore, we can compute

$$\mathbf{A}^{(1)\intercal}\mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(n-1)\intercal}\mathbf{A}^{(n-1)} \otimes \mathbf{A}^{(n+1)\intercal}\mathbf{A}^{(n+1)} \otimes \cdots \otimes \mathbf{A}^{(N)\intercal}\mathbf{A}^{(N)}$$

in $O(R_{\neq n}^2 \cdot N)$ time and $O(R_{\neq n}^2)$ space, where $R_{\neq n} = \prod_{k \neq n} R_k$. Further, we can compute $\mathbf{K}\mathbf{K}^\intercal \in \mathbb{R}^{R_n \times R_n}$ in time

$$O(R_{\neq n}^2 R_n + R_n^2 R_{\neq n})$$

by multiplying the two rightmost matrices, and then the remaining two. Therefore, we can compute $(\mathbf{K}\mathbf{K}^\intercal + \lambda\mathbf{I})^+$ in $O(R_n^\omega)$ time.

Now we need to work on the term $\mathbf{K}\mathbf{b}_{i:}^\intercal \in \mathbb{R}^{r_n}$. We have the following:

$$\mathbf{K}\mathbf{b}_{i:}^\intercal = \mathbf{G}_{(n)}(\mathbf{A}^{(1)} \otimes \cdots \otimes \mathbf{A}^{(n-1)} \otimes \mathbf{A}^{(n+1)} \otimes \cdots \otimes \mathbf{A}^{(N)})^\intercal \mathbf{b}_{i:}^\intercal. \tag{41}$$

We need to compute the internal Kronecker product in $O(I_{\neq n} R_{\neq n} N)$ time and $O(I_{\neq n} R_{\neq n})$ space. The vector multiplication with $\mathbf{b}_{i:}^\intercal$ takes $O(R_{\neq n} I_{\neq n})$ time (and this happens I_n times). These can actually be combined so that the total max memory consumed is $O(R_{\neq n})$ by computing each entry of the matrix-vector product individually. Then the final matrix multiplication takes $O(R_n R_{\neq n})$ time.

Putting everything together, we can compute $(\mathbf{K}\mathbf{K}^\intercal + \lambda\mathbf{I})^+ \mathbf{K}\mathbf{b}_{i:}^\intercal$ with a final matrix-vector multiplication in $O(R_n^2)$ time. Thus, the overall running time of this factor matrix is:

$$O\left(\left(\sum_{k \neq n} R_k^2 I_k\right) + R_{\neq n}^2 N + R_{\neq n}^2 R_n + R_n^2 R_{\neq n} + R_n^\omega + I_{\neq n} R_{\neq n} N + R_{\neq n} I\right)$$

and the space complexity is $O(R_{\neq n}^2 + R_{\neq n}) = O(R_{\neq n}^2)$.

Naive core tensor update. We use the normal equation to solve:

$$\mathcal{G} \leftarrow \underset{\mathcal{G}'}{\arg\min} \|(\mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)} \otimes \cdots \otimes \mathbf{A}^{(N)})\text{vec}(\mathcal{G}') - \text{vec}(\mathcal{X})\|_2^2 + \lambda \|\text{vec}(\mathcal{G}')\|_2^2.$$

Let $\mathbf{K} = \mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)} \otimes \cdots \otimes \mathbf{A}^{(N)}$. We need to efficiently compute

$$\mathbf{g}_{\text{opt}} = (\mathbf{K}^\intercal\mathbf{K} + \lambda\mathbf{I})^+ \mathbf{K}^\intercal \text{vec}(\mathcal{X}).$$

Observe that

$$\mathbf{K}^\intercal\mathbf{K} = \mathbf{A}^{(1)\intercal}\mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)\intercal}\mathbf{A}^{(2)} \otimes \cdots \otimes \mathbf{A}^{(N)\intercal}\mathbf{A}^{(N)}.$$

Thus, we can compute $\mathbf{K}^\intercal\mathbf{K}$ in time $O((\sum_{n=1}^N R_n^2 I_n) + R^2 N)$ where $R = \prod_{n=1}^N R_n$. It follows that we can then compute $(\mathbf{K}^\intercal\mathbf{K} + \lambda\mathbf{I})^+$ in time $O(R^\omega)$ and $O(R^2)$ space.

Now we need to compute $\mathbf{K}^\intercal\mathbf{x} \in \mathbb{R}^R$. Computing $\mathbf{K}^\intercal$ explicitly requires $O(IR)$ space and $O(IRN)$ time. Computing the matrix-vector product one entry at a time requires $O(IRN)$ time and only $O(R)$ space. Putting everything together, the overall running time is:

$$O\left(\left(\sum_{n=1}^N R_n^2 I_n\right) + R^2 N + R^\omega + IRN\right).$$

The space complexity is $O(R^2)$.

Factor matrix update with `KronMatMul`. We use `KronMatMul` (Lemma 4.4) in (40) as follows:

$$\texttt{KronMatMul}\Big([\mathbf{A}^{(1)\intercal}\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(n-1)\intercal}\mathbf{A}^{(n-1)}, \mathbf{A}^{(n+1)\intercal}\mathbf{A}^{(n+1)}, \ldots \mathbf{A}^{(N)\intercal}\mathbf{A}^{(N)}], \mathbf{G}_{(n)}^\intercal\Big).$$

The resulting matrix has size $R_{\neq n} \times R_n$. The running time for this step is $O(R_{\neq n} R_n \sum_{k\neq n} R_k) = O(R\sum_{k\neq n} R_k)$ and the space complexity is $O(R)$. Previously this step was $O(R_{\neq n}^2 N + R_{\neq n}^2 R_n)$.

Now for the expensive matrix-vector product:

$$\texttt{KronMatMul}\Big([\mathbf{A}^{(1)\intercal}, \ldots, \mathbf{A}^{(n-1)\intercal}, \mathbf{A}^{(n+1)\intercal}, \ldots, \mathbf{A}^{(N)\intercal}], \mathbf{b}_{i:}^\intercal\Big)$$

The result is a vector of size $R_{\neq n}$. The running time of this subroutine is $O(I_{\neq n}\sum_{k\neq n} R_k)$, and it uses $O(I_{\neq n})$ space. It follows that we can compute $\mathbf{K}\mathbf{b}_{i:}^\intercal$ by multiplying this vector with $\mathbf{G}_{(n)}$, which takes $O(R_n R_{\neq n}) = O(R)$ time. Previously this took $O(I_{\neq n} R_{\neq n} N + R_{\neq n} I_{\neq n} + R)$ time.

Thus, the updated time complexity is

$$O\left(R\left(\sum_{n=1}^N R_n\right) + R_n^\omega + I_{\neq n}\left(\sum_{k\neq n} R_k\right) + R_n^2 I_n\right).$$

Core update with `KronMatMul`. The running time for $\mathbf{K}^\intercal\mathbf{x} \in \mathbb{R}^{I_1\cdots I_N}$ is $O((R_1 + \cdots + R_N)I)$ and space is $O(I)$. Therefore, the overall running time is $O(R^2 N + R^\omega + I\sum_{n=1}^N R_n)$. The overall space complexity is $O(R^2)$.

D Supplementary Material for Experiments

D.1 Kronecker Regression

Here we provide the full experimental results for the Kronecker regression numerical experiments in Section 6. In the tables below, `Baseline` is the naive baseline algorithm that computes the normal equation exactly (i.e., fully constructs $\mathbf{K}$ in the $\mathbf{K}^\intercal\mathbf{b}$ term), `KronMatMul` is an exact baseline that uses the fast Kronecker-vector product in Lemma 4.4, `DJSSW19` is the sketching algorithm of Diao et al. [18], and Algorithm 2 is our `FastKroneckerRegression` algorithm.

We note that the number of sketches used by both sketching algorithms is reduced by an $\alpha = 10^{-5}$ factor so that the number of total row samples is of the same order of magnitude as the fixed number of samples in the experiments of [17, 18]. Our choice of $\lambda = 10^{-3}$ for the L2 regularization strength does not affect the running times or significantly impact the relative errors.

Running times. The running times of all the algorithms are presented in Table 3. We denote entries where the algorithm ran out of memory by a dash.

Losses. We list the losses of all the Kronecker regression algorithms in Table 4. Entries where the algorithm ran out of memory by a dash are denoted by a dash. We note that `DJSSW19` begins to have numerical stability problems for $n \geq 2048$ for $d \in \{16, 32, 64\}$, though it is solving the same sketched ridge regression problem as Algorithm 2.

Table 3: Running times of Kronecker regression algorithms with an $n^2 \times d^2$ design matrix (seconds).

d	n	`Baseline`	`KronMatMul`	`DJSSW19`	Algorithm 2
8	128	0.0106	0.0008	0.0056	0.0074
	256	0.0349	0.0014	0.0051	0.0155
	512	0.1343	0.0037	0.0064	0.0172
	1024	0.5009	0.0115	0.0087	0.0192
	2048	1.9403	0.0466	0.014	0.0229
	4096	7.5253	0.1855	0.0236	0.042
	8192	29.9479	0.676	0.0404	0.0578
	16384	–	3.9717	0.0776	0.1019
16	128	0.1416	0.001	0.1154	0.0942
	256	0.1992	0.0022	0.1212	0.0988
	512	0.5485	0.0062	0.1216	0.0927
	1024	1.8994	0.0222	0.1184	0.0982
	2048	7.1677	0.0916	0.1286	0.1112
	4096	27.5237	0.3399	0.1365	0.114
	8192	–	2.2858	0.1599	0.1367
	16384	–	15.0028	0.2036	0.1869
32	128	5.9593	0.0019	10.3478	1.6544
	256	7.0358	0.0039	10.4735	1.8021
	512	8.1757	0.0118	10.3156	1.7045
	1024	13.6273	0.0474	10.3379	1.6802
	2048	31.4996	0.1739	10.4088	1.7193
	4096	–	1.129	10.2585	1.6732
	8192	–	7.7335	10.4856	1.7505
	16384	–	29.7196	10.5815	1.8613
64	128	966.1175	0.0058	1240.6471	29.6798
	256	982.5172	0.0102	1243.1648	29.6453
	512	970.6558	0.0262	1240.7365	29.3725
	1024	985.4097	0.09	1239.8602	29.5546
	2048	–	0.6687	1238.6558	29.4367
	4096	–	3.6422	1243.2635	29.7029
	8192	–	14.5137	1244.8347	29.8926
	16384	–	67.0037	1241.5569	29.9143

D.2 Low-rank Tucker Decomposition of Image Tensors

Here we compare different Kronecker regression algorithms in the core update of the alternating least squares (ALS) algorithm for Tucker decompositions. For the sketching-based algorithms, we increase the number of row samples to study how this affects the quality of the tensor decomposition. We record the quality of the tensor decomposition using the relative reconstruction error $\|\widehat{\mathcal{X}} - \mathcal{X}\|_{\mathrm{F}}^2 / \|\mathcal{X}\|_{\mathrm{F}}^2$. The number of row samples used is $m \in \{1024, 4096, 16384\}$, as in the experiments of [17, 18].

We compare against higher-order orthogonal iteration (HOOI) and ALS as baseline algorithms. We use the Tensorly [36] implementation of HOOI, which is an industry standard. We do not use L2 regularization so that we can compare against HOOI. We compare our Kronecker regression algorithm with [18, Algorithm 1], denoted by `DJSSW19`. The running times reported are the mean iteration times, where an iteration includes all factor matrix updates and the core tensor update. Trials that ran out of memory or failed to converge are denoted by a dash. All algorithms are run for five iterations.

Cardiac MRI. This dataset is a $256 \times 256 \times 14 \times 20$ tensor whose elements are MRI measurements indexed by (x, y, z, t) where (x, y, z) is a point in space and t corresponds to time.

Table 4: Losses of the Kronecker regression algorithms with a random $n^2 \times d^2$ design matrix.

d	n	`Baseline`	`KronMatMul`	`DJSSW19`	Algorithm 2
8	128	0.0033	0.0033	0.0035	0.0035
	256	0.0151	0.0151	0.0161	0.0161
	512	0.0606	0.0606	0.0635	0.0635
	1024	0.2531	0.2531	0.2636	0.2636
	2048	1.032	1.032	1.0599	1.0598
	4096	4.2118	4.2118	4.4741	4.4525
	8192	16.9013	16.9009	17.6642	17.5533
	16384	–	66.6618	96.9055	72.6006
16	128	0.0019	0.0019	0.0021	0.0021
	256	0.0077	0.0077	0.0081	0.0081
	512	0.0313	0.0313	0.0324	0.0324
	1024	0.1306	0.1306	0.1328	0.1328
	2048	0.5244	0.5244	0.5457	0.5375
	4096	2.0917	2.0915	2.2278	2.1381
	8192	–	8.2601	9.1014	8.5251
	16384	–	33.3611	42.8783	34.4764
32	128	0.0007	0.0007	0.0009	0.0009
	256	0.0037	0.0037	0.004	0.004
	512	0.0163	0.0163	0.0172	0.0172
	1024	0.0671	0.0671	0.0693	0.0692
	2048	0.2661	0.266	0.388	0.2701
	4096	–	1.0434	2.9236	1.0589
	8192	–	4.1777	4.7172	4.2537
	16384	–	16.874	24149.8906	17.4612
64	128	0.0002	0.0002	0.0004	0.0004
	256	0.0015	0.0015	0.0019	0.0019
	512	0.0078	0.0078	0.0087	0.0087
	1024	0.0307	0.0307	0.035	0.0323
	2048	–	0.1233	1.5768	0.1264
	4096	–	0.5068	275.5661	0.5198
	8192	–	2.0735	333.4299	2.1358
	16384	–	8.2375	546391.7285	8.6082

Table 5: Relative reconstruction errors for cardiac MRI tensor with different multilinear ranks.

			`FastKroneckerRegression`			`DJSSW19`		
rank	HOOI	ALS	1024	4096	16384	1024	4096	16384
1,1,1,1	0.648	0.648	0.649	0.648	0.648	0.648	0.648	0.648
4,2,2,1	0.569	0.570	0.574	0.571	0.570	0.573	0.571	0.570
4,4,2,2	0.511	0.511	0.533	0.514	0.512	0.561	0.515	0.513
8,2,2,1	0.569	0.577	0.584	0.579	0.577	0.589	0.581	0.577
8,4,4,1	0.448	0.452	0.491	0.459	0.453	0.488	0.458	0.454
8,4,4,2	0.448	0.451	0.492	0.475	0.455	0.585	0.466	0.455
8,8,2,2	0.465	0.467	0.498	0.485	0.471	0.556	0.480	0.470
8,8,4,4	0.350	0.351	–	0.371	0.356	0.679	0.476	0.409

Table 6: Average iteration time of ALS with sketching-based Kronecker regression for cardiac MRI tensor with different multilinear ranks (seconds).

			FastKroneckerRegression			DJSSW19		
rank	HOOI	ALS	1024	4096	16384	1024	4096	16384
1,1,1,1	1.187	1.307	1.328	1.334	1.321	1.397	1.313	1.346
4,2,2,1	1.429	1.368	1.345	1.326	1.349	1.395	1.350	1.383
4,4,2,2	1.458	1.463	1.539	1.511	1.536	1.413	1.497	1.719
8,2,2,1	2.401	1.339	1.421	1.415	1.347	1.368	1.300	1.425
8,4,4,1	1.664	1.435	1.575	1.562	1.676	1.573	1.693	2.737
8,4,4,2	1.745	1.614	1.782	1.754	2.137	1.820	2.667	6.496
8,8,2,2	1.741	1.466	1.621	1.810	2.079	1.751	2.525	6.133
8,8,4,4	1.784	1.835	2.131	2.745	5.210	9.199	35.977	128.538

We also investigate how sensitive the convergence rate of sketching-based ALS is to the choice of the error parameter ε. First, we reduce the number of sampled rows by $\alpha = 0.001$ to compensate for the large constant coefficient in Line 8 in Algorithm 2; otherwise, we do not see any quality degradation even for $\varepsilon = 0.99$. Then in Table 7 and Table 8, we compare the RRE at each step of ALS (without sampling) and when using `FastKroneckerRegression` as a subroutine for decreasing values of ε.

Table 7: Relative reconstruction errors for cardiac MRI tensor with multilinear rank $(4, 4, 2, 2)$ during ALS with and without using `FastKroneckerRegression` as a subroutine.

		FastKroneckerRegression				
Step	ALS	$\varepsilon = 0.8$	$\varepsilon = 0.4$	$\varepsilon = 0.2$	$\varepsilon = 0.1$	$\varepsilon = 0.05$
1	0.55883	0.56270	0.56044	0.55957	0.55942	0.55899
2	0.51292	0.51609	0.51511	0.51443	0.51377	0.51316
3	0.51096	0.51466	0.51206	0.51167	0.51139	0.51120
4	0.51081	0.51338	0.51287	0.51171	0.51127	0.51102
5	0.51079	0.51361	0.51286	0.51150	0.51126	0.51105

Table 8: Relative reconstruction errors for cardiac MRI tensor with multilinear rank $(8, 8, 4, 4)$ during ALS with and without using `FastKroneckerRegression` as a subroutine.

		FastKroneckerRegression				
Step	ALS	$\varepsilon = 0.8$	$\varepsilon = 0.4$	$\varepsilon = 0.2$	$\varepsilon = 0.1$	$\varepsilon = 0.05$
1	0.44961	0.45165	0.45084	0.45021	0.44987	0.44975
2	0.36573	0.36707	0.36650	0.36612	0.36609	0.36579
3	0.35488	0.35549	0.35571	0.35508	0.35516	0.35504
4	0.35162	0.35293	0.35238	0.35201	0.35184	0.35177
5	0.35081	0.35193	0.35149	0.35124	0.35100	0.35092

Hyperspectral. This dataset is a $1024 \times 1344 \times 33$ tensor of time-lapse hyperspectral radiance images capturing a 1-hour interval of a nature scene undergoing illumination changes [50]. These hyperspectral images and the COIL-100 dataset have both been used recently as benchmark tasks for low-rank tensor decomposition [44, 9, 67].

Table 9: Relative reconstruction errors for hyperspectral tensor with different multilinear ranks.

			FastKroneckerRegression			DJSSW19		
rank	HOOI	ALS	1024	4096	16384	1024	4096	16384
1,1,1	0.271	0.271	0.271	0.271	0.271	0.271	0.271	0.271
2,2,2	0.235	0.235	0.236	0.236	0.235	0.236	0.236	0.235
4,4,4	0.203	0.208	0.213	0.211	0.211	0.213	0.208	0.208
8,8,4	0.169	0.170	0.189	0.176	0.171	0.201	0.175	0.171
8,8,8	0.169	0.169	0.213	0.177	0.171	0.261	0.180	0.171
16,16,4	0.133	0.134	–	0.155	0.139	0.465	0.156	0.139

Table 10: Average iteration time of ALS with sketching-based Kronecker regression for the hyperspcetral image tensor with different multilinear ranks (seconds).

			FastKroneckerRegression			DJSSW19		
rank	HOOI	ALS	1024	4096	16384	1024	4096	16384
1,1,1	1.873	2.377	2.222	2.241	2.312	2.224	2.241	2.203
2,2,2	2.019	2.491	2.449	2.413	2.476	2.450	2.470	2.452
4,4,4	2.255	2.965	2.798	2.838	2.922	2.747	2.902	3.022
8,8,4	2.845	3.282	3.150	3.305	3.481	3.324	4.240	7.783
8,8,8	2.888	4.043	3.700	4.168	5.682	5.559	11.774	34.878
16,16,4	3.997	3.880	3.969	4.802	7.781	11.104	38.066	129.878

COIL-100. This dataset is a $7200 \times 120 \times 120 \times 3$ tensor that contains 7200 colored images of 100 objects (72 images per object). These objects have a wide variety of geometric characteristics and reflective properties. To construct this dataset, these objects were placed on a rotating table and pictures were taken at pose intervals of 5 degrees, hence 72 images per object [51].

Table 11: Relative reconstruction errors for the COIL-100 tensor with different multilinear ranks.

			FastKroneckerRegression			DJSSW19		
rank	HOOI	ALS	1024	4096	16384	1024	4096	16384
1,1,1,1	0.528	0.528	0.528	0.528	0.528	0.528	0.528	0.528
4,2,2,1	0.460	0.460	0.463	0.461	0.460	0.464	0.461	0.461
8,2,2,1	0.460	0.460	0.472	0.462	0.461	0.466	0.462	0.461
8,4,4,1	0.414	0.414	0.447	0.421	0.416	0.443	0.420	0.416
8,4,4,2	0.379	0.386	0.454	0.400	0.388	0.438	0.399	0.388
16,4,4,2	0.349	0.349	0.499	0.377	0.355	0.517	0.376	0.356

Table 12: Average iteration time of ALS with sketching-based Kronecker regression for the COIL-100 tensor with different multilinear ranks (seconds).

			FastKroneckerRegression			DJSSW19		
rank	HOOI	ALS	1024	4096	16384	1024	4096	16384
1,1,1,1	2.455	10.975	10.128	10.147	10.138	10.131	10.264	10.195
4,2,2,1	6.100	11.718	11.122	11.153	11.164	11.104	10.941	11.080
8,2,2,1	12.092	11.727	11.137	11.120	11.164	11.111	11.069	11.120
8,4,4,1	10.877	13.873	13.096	12.954	13.051	12.924	13.067	13.986
8,4,4,2	10.906	19.614	17.862	17.784	17.984	17.957	18.575	22.107
16,4,4,2	19.225	19.921	18.367	18.299	18.525	19.922	25.639	48.483