Sketching for Kronecker Product Regression and P-splines

Huaian Diao
hadiao@nenu.edu.cn
Northeast Normal University

Zhao Song
zhaos@utexas.edu
Harvard U & UT-Austin

Wen Sun
wensun@cs.cmu.edu
Carnegie Mellon University

David P. Woodruff
dwoodruf@cs.cmu.edu
Carnegie Mellon University

Abstract

TensorSketch is an oblivious linear sketch introduced in (Pagh, 2013) and later used in (Pham and Pagh, 2013) in the context of SVMs for polynomial kernels. It was shown in (Avron et al., 2014) that TensorSketch provides a subspace embedding, and therefore can be used for canonical correlation analysis, low rank approximation, and principal component regression for the polynomial kernel. We take TensorSketch outside of the context of polynomials kernels, and show its utility in applications in which the underlying design matrix is a Kronecker product of smaller matrices. This allows us to solve Kronecker product regression and non-negative Kronecker product regression, as well as regularized spline regression. Our main technical result is then in extending TensorSketch to other norms. That is, TensorSketch only provides input sparsity time for Kronecker product regression with respect to the 2-norm. We show how to solve Kronecker product regression with respect to the 1-norm in time sublinear in the time required for computing the Kronecker product, as well as for more general p-norms.

1 INTRODUCTION

In the overconstrained least squares regression problem, we are given an $n \times d$ matrix A called the "design matrix", $n \gg d$, and an $n \times 1$ vector b, and the goal is to find an x which minimizes $\|Ax - b\|_2$. There are many variants to this problem, such as regularized versions of the problem in which one seeks an x so as to minimize $\|Ax - b\|_2^2 + \|Lx\|_2^2$ for a matrix L, or regression problems which seek to minimize more robust loss functions, such as ℓ_1-regression $\|Ax - b\|_1$.

In the era of big data, large scale matrix computations have attracted considerable interest. To obtain reasonable computational and time complexities for large scale matrix computations, a number of randomized approximation algorithms have been developed. For example, in (Woodruff, 2014), it was shown how to output a vector $x' \in \mathbb{R}^d$ for which $\|Ax' - b\|_2 \leq (1+\epsilon) \min_{x \in \mathbb{R}^d} \|Ax - b\|_2$ in $\texttt{nnz}(A) + \text{poly}(d/\epsilon)$ time, where $\texttt{nnz}(A)$ denotes the number of non-zero entries of matrix A. We refer the reader to the recent surveys (Kannan and Vempala, 2009; Mahoney, 2011; Woodruff, 2014) for a detailed treatment of this topic.

In this work we focus on regression problems for which the design matrix is a *Kronecker product matrix*, that is, it has the form $A \otimes B$ for $A \in \mathbb{R}^{n_1 \times d_1}$ and $B \in \mathbb{R}^{n_2 \times d_2}$. Also, $b \in \mathbb{R}^{n_1 n_2}$, and one seeks to solve the problem $\min_{x \in \mathbb{R}^{d_1 d_2}} \|(A \otimes B)x - b\|_2$ in the standard setting, which can also be generalized to regularized and robust variants. One can also ask the question when the design matrix is a Kronecker product of more than two matrices.

Kronecker product matrices have many applications in statistics, linear system theory, signal processing, photogrammetry, multivariate data fitting, etc.; see (Golub and Van Loan, 2013; Van Loan and Pitsianis, 1993; Van Loan, 1992). The linear least squares problem involving the Kronecker product arises in many applications, such as structured linear total least norm on blind deconvolution problems (Oh and Yun, 2005), constrained linear least squares problems with a Kronecker product structure, the bivariate problem of surface fitting and multidimensional density smoothing (Eilers and Marx, 2006).

One way to solve Kronecker product regression is to form the matrix $C = A \otimes B$ explicitly, where $A \in \mathbb{R}^{n_1 \times d_1}, B \in \mathbb{R}^{n_2 \times d_2}$, and then apply a randomized algorithm to C. However, this takes at least $\texttt{nnz}(A) \cdot \texttt{nnz}(B)$ time and space. It is natural to ask if

Full version is at https://arxiv.org/abs/1712.09473v1
Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84.

it is possible to solve the Kronecker product regression problem in time faster than computing $A \otimes B$. This is in fact the case, as Fausett and Fulton (Fausett and Fulton, 1994) show that one can solve Kronecker product regression in $O(n_1 d_1^2 + n_2 d_2^2)$ time; indeed, the solution vector $x = \mathrm{vec}((B^\perp)^\top D^{-1} A^\perp)$, where $D = \mathrm{vec}(b)$ and $\mathrm{vec}(E)$ for a matrix E denotes the operation of stacking the columns of E into a single long vector. While such a computation does not involve computing $A \otimes B$, it is more expensive than what one would like.

A natural question is if one can approximately solve Kronecker product regression in $\mathtt{nnz}(A) + \mathtt{nnz}(B) + \mathrm{poly}(d/\epsilon)$ time, which, up to the $\mathrm{poly}(d/\epsilon)$ term, would match the description size of the input. Another natural question is Kronecker product regression with regularization. Such regression problems arise frequently in the context of splines (Eilers and Marx, 2006). Finally, what about Kronecker product regression with other, more robust, norms such as the ℓ_1-norm?

1.1 Our Contributions

We first observe that the random linear map TensorSketch, introduced in the context of problems for the polynomial kernel by Pagh (2013), Pham and Pagh (2013), is exactly suited for this task. Namely, in (Avron et al., 2014) it was shown that for a d-dimensional subspace C of $\mathbb{R}^n$, represented as an $n \times d$ matrix, there is a distribution on linear maps S with $O(d^2/\epsilon^2)$ rows such that with constant probability, simultaneously for all $x \in \mathbb{R}^d$, $\|SCx\|_2 = (1 \pm \epsilon)\|Cx\|_2$. That is, S provides an *Oblivious Subspace Embedding (OSE)* for the column span of C. Further, it is known that if b is an n-dimensional vector, then one has that $\|S[C, b]x\|_2 = (1 \pm \epsilon)\|[C, b]x\|_2$ for all $x \in \mathbb{R}^{d+1}$. Consequently, to solve the regression problem $\min_{x \in \mathbb{R}^d} \|Cx - b\|_2$, one can instead solve the much smaller problem $\min_{x \in \mathbb{R}^d} \|SCx - Sb\|_2$, and the solution x' to the latter problem will satisfy $\|Cx' - b\|_2 \leq (1 + \epsilon) \min_{x \in \mathbb{R}^d} \|Cx - b\|_2$.

Importantly, if $n = n_1 \cdot n_1$ and there is a basis for the column span of C of the form $A_1 \otimes A_1, A_2 \otimes A_2, \ldots, A_d \otimes A_d$, then given $A_1, \ldots, A_d$, it holds that SC can be computed in $\mathtt{nnz}(A)$ time, where A is the $n_1 \times d$ matrix whose i-th column is A_i. Further, given a vector b with $\mathtt{nnz}(b)$ non-zero entries, one can compute Sb in $\mathtt{nnz}(b)$ time. Thus, one obtains a $(1 + \epsilon)$-approximate solution to the regression $\min_{x \in \mathbb{R}^d} \|Cx - b\|_2$ in $\mathtt{nnz}(A) + \mathtt{nnz}(b) + \mathrm{poly}(d/\epsilon)$ time.

While not immediately useful for our problem, we show that via simple modifications, the claim about TensorSketch above can be generalized to the case when there is a basis for the column span of C of the form $A_i \otimes B_j$ for arbitrary vectors $A_1, \ldots, A_{d_1} \in \mathbb{R}^{n_1}$ and vectors $B_1, \ldots, B_{d_2} \in \mathbb{R}^{n_2}$. That is, we observe that in this case SC can be computed in $\mathtt{nnz}(A) + \mathtt{nnz}(B)$ time, where A is the $n_1 \times d_1$ matrix whose i-th column is A_i, and B is the $n_2 \times d_2$ matrix whose i-th column is B_i. In this case $C = A \otimes B$, which is exactly the case of Kronecker product regression. Using the above connection to regression, we obtain an algorithm for Kronecker product regression in $\mathtt{nnz}(A) + \mathtt{nnz}(B) + \mathtt{nnz}(b) + \mathrm{poly}(d_1 d_2/\epsilon)$ time. Using the fact that $\|SCx - Sb\|_2 = (1 \pm \epsilon)\|Cx - b\|_2$ for all x, we have in particular that this holds for all non-negative x, and so also obtain the same reduction in problem size for non-negative Kronecker product regression, which occurs often in image and text data; see e.g., (Chen and Plemmons, 2010).

The above observation allows us to extend many existing variants of least squares regression to the case when C is a Kronecker product of two matrices. For example, the results in (Avron et al., 2016) for the ridge regression problem $\min_{x \in \mathbb{R}^d} \|Ax - b\|_2^2 + \lambda \|x\|_2^2$ immediately hold when C is a Kronecker product matrix, since the conditions needed for the oblivious embedding in (Avron et al., 2016) come down to a subspace embedding and an approximate matrix product condition, both of which are known to hold for TensorSketch (Avron et al., 2014). More interestingly we are able to extend the results in (Avron et al., 2016) for Kronecker ridge regression to the case when the regularizer is a general matrix, namely, to the problem $\min_{x \in \mathbb{R}^d} \|Ax - b\|_2^2 + \lambda \|Lx\|_2^2$, where L is an arbitrary matrix. Such problems occur in the context of spline regression (Eilers and Marx, 1996, 2006; Eilers et al., 2015). The number of rows of TensorSketch depends on a generalized notion of statistical dimension depending on the generalized singular values γ_i of $[A; L]$, and may be much smaller than d_1 or d_2. In (Avron et al., 2016), only results for L equal to the identity were obtained.

Finally, our main technical result is to extend our results to least absolute deviation Kronecker product regression $\min_{x \in \mathbb{R}^d} \|Cx - b\|_1$, which, since it involves the 1-norm, is often considered more robust than least squares regression (Rousseeuw and Leroy, 2005) and has been widely used in applications such as computer vision (Zheng et al., 2012). We in fact extend this to general p-norms but focus the discussion on $p = 1$. We give the first algorithms for this problem that are faster than computing $C = A \otimes B$ explicitly, which would take at least $\mathtt{nnz}(A) \cdot \mathtt{nnz}(B) \geq n_1 n_2$ time. Namely, and for simplicitly focusing on the case when $n_1 = n_2$, for which the goal is to do better than n_1^2 time, we show how to output an $x \in \mathbb{R}^{d_1 \times d_2}$ for which $\|Cx' - b\|_1 \leq (1 + \epsilon) \min_{x \in \mathbb{R}^{d_1 d_2}} \|Cx - b\|_1$ in time $n^{3/2} \mathrm{poly}(d_1 d_2/\epsilon)$. While this is more expensive than solving Kronecker product least squares, the 1-norm

may lead to significantly more robust solutions. From a technical perspective, TensorSketch when applied to a vector actually destroys the 1-norm of a vector, preserving only its 2-norm, so new ideas are needed. We show how to use multiple TensorSketch matrices to implicitly obtain very crude estimates to the so-called ℓ_1-leverage scores of C, which can be interpreted as probabilities of sampling rows of C in order to obtain an ℓ_1-subspace embedding of the column span of C (see, e.g., (Woodruff, 2014)). However, since C has $n_1 n_2$ rows, we cannot even afford to write down the ℓ_1-leverage scores of C. We show how to implicitly represent such leverage scores and to sample from them without ever explicitly writing them down. Balancing the phases of our algorithm leads to our overall time.

1.2 Notation

We consider Kronecker Product of q 2-d matrices $A_1 \otimes A_2 \otimes ... \otimes A_q$, where each $A_i \in \mathbb{R}^{n_i \times d_i}$. We denote $\mathcal{A} = A_1 \otimes A_2 \otimes ... \otimes A_q$, $n = \prod_{i=1}^q n_i$, and $d = \prod_{i=1}^q d_i$. We denote $[n]$ as the set $\{1, 2, 3, \cdots, n\}$. The ℓ_p norm for a vector $x \in \mathbb{R}^d$ is defined as $\|x\|_p = (\sum_{i=1}^d |x_i|^p)^{1/p}$, where x_i stands for the i'th entry of the vector x. For any matrix M, we use $M_{i,*}$ to represent the i'th row of M and $M_{*,j}$ as the j'th column of M.

We define a *Well-Conditioned Basis* and *Statistical Dimension* as follow (similar definitions can be found in Clarkson (2005); Dasgupta et al. (2009); Sohler and Woodruff (2011); Meng and Mahoney (2013); Song et al. (2017a,b) and Avron et al. (2016)):

Definition 1.1 (Well-Conditioned Basis)**.** *Let A be an $n \times m$ matrix with rank d, let $p \in [1, \infty)$, and let $\|\cdot\|_q$ be the dual norm of $\|\cdot\|_p$, i.e., $1/p + 1/q = 1$. Then an $n \times d$ matrix U is an (α, β, p)-well-conditioned basis for the column space of A, if the columns of U span the column space of A and (1) $\|U\|_p \leq \alpha$, and (2) for all $z \in \mathbb{R}^d$, $\|z\|_q \leq \beta \|Uz\|_p$.*

Consider the classic Ridge Regression: $\min_x \|Ax - b\|_2^2 + \lambda \|x\|_2^2$. The *Statistical Dimension* is defined as:

Definition 1.2 (Statistical Dimension)**.** *Let A be an $n \times m$ matrix with rank d and singular values $\sigma_i, i \in [d]$. For $\lambda \geq 0$, the statistical dimension $\mathtt{sd}_\lambda(A)$ is defined as the quantity $\mathtt{sd}_\lambda(A) = \sum_{i \in [d]} 1/(1 + \lambda/\sigma_i^2)$.*

2 BACKGROUND: TensorSketch

We briefly introduce TensorSketch (Pagh, 2013; Avron et al., 2014) and how to apply TensorSketch to the Kronecker product of multiple matrices efficiently without explicitly computing the tensor product.[1]

We want to find a oblivious subspace embedding S such that for any $x \in \mathbb{R}^n$, we have $\|S\mathcal{A}x\|_2 = (1 \pm \epsilon)\|\mathcal{A}x\|_2$, where the notation $a = (1 \pm \epsilon b)$ stands for $(1 - \epsilon)b \leq a \leq (1 + \epsilon)b$, for any $a, b \in \mathbb{R}$. Consider the $(i_1, i_2, ..., i_q)$'th column of $\mathcal{A}$ ($i_j \in [d_j]$): $A_{1_{*,i_1}} \otimes A_{2_{*,i_2}} \otimes \cdots \otimes A_{q_{*,i_q}}$. Assume the sketching target dimension is m. TensorSketch is defined using q 3-wise indepedent hash functions $h_i : [n_i] \to [m]$, and q 4-wise independent sign functions $s_i : [n_i] \to \{-1, 1\}$, $\forall i \in [q]$. Define hash function $H : [n_1] \times [n_2] \times \cdots \times [n_q] \to [m]$ as $H(i_1, i_2, ..., i_q) = ((\sum_{k=1}^q h_k(i_k)) \mod m)$, and sign function $S : [n_1] \times [n_2] \times \cdots \times [n_q] \to \{-1, 1\}$ as $S(i_1, i_2, ..., i_q) = \prod_{k=1}^q s_k(i_k)$. Applying TensorSketch to the Kronecker product of vectors $A_{1_{*,i_1}}, ..., A_{q_{*,i_q}}$ is equivalent to applying CountSketch (Charikar et al., 2004) defined with H and S to the vector $(A_{1_{*,i_1}} \otimes ... \otimes A_{q_{*,i_q}})$. To apply TensorSketch to $\mathcal{A}$, we just need to apply CountSketch defined with H and S to the columns of $\mathcal{A}$ one by one.

Applying TensorSketch to the Kronecker product of $(A_{1_{*,i_1}}, ..., A_{q_{*,i_q}})$ naively would require at least $O(n)$ time. Pagh (2013) shows that one can apply TensorSketch to the Kronecker product of $(A_{1_{*,i_1}}, ..., A_{q_{*,i_q}})$ without explicitly computing the Kronecker product of these vectors using the Fast Fourier Transformation. Particularly, Pham and Pagh (2013) show that one only needs $O(\sum_{j=1}^q \mathtt{nnz}(A_{j_{*,i_j}}) + qm \log(m))$ time to compute $S(A_{1_{*,i_1}} \otimes ... \otimes A_{q_{*,i_q}})$ where $A_{i_{*,j}}$ stands for the j'th column of A_i. As $\mathcal{A}$ has d columns, computing $S\mathcal{A}$ takes $O(d(\sum_{i=1}^q \mathtt{nnz}(A_i)) + dqm \log(m))$ time, which is much smaller than $O(\prod_{i=1}^q n_i)$, which is the least amount of time one needs for explicitly computing $\mathcal{A}$. In the rest of the paper, we assume that we compute $S\mathcal{A}$ using the above efficient procedure without explicitly computing $A_1 \otimes \ldots \otimes A_q$.

3 TENSOR PRODUCT LEAST SQUARES REGRESSION

Consider the tensor product least squares regression problem $\min_x \|\mathcal{A}x - b\|_2$ where $\mathcal{A} \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$. Let $S \in \mathbb{R}^{m \times n}$ be the matrix form of TensorSketch of Section 2. We propose a TensorSketch -type Algorithm 1 for the tensor product regression problem. The following theorem shows that the solution obtained from Alg. 1 is a good approximation of the optimal solution of the original tensor product regression.

Let us define OPT to be the optimal cost of the optimization problem, e.g., $\mathrm{OPT} = \min_x \|\mathcal{A}x - b\|_2$. The following theorem shows that Alg. 1 computes a good approximate solution.

Theorem 3.1. (Tensor regression) Suppose $\widetilde{x}$ is the output of Algorithm 1 with TensorSketch $S \in \mathbb{R}^{m \times n}$, where $m = 8(d_1 d_2 \cdots d_q + 1)^2(2 + 3^q)/(\epsilon^2 \delta)$. Then the following approximation $\|(A_1 \otimes A_2 \otimes \cdots \otimes A_q)\widetilde{x} - b\|_2 \leq (1 + \epsilon)\,\mathrm{OPT}$, holds with probability at least $1 - \delta$.

[1] We refer readers to (Avron et al., 2014) for more details about TensorSketch.

Algorithm 1 Tensor product regression

procedure TREGRESSION(A, b, ϵ, δ)
  $m \leftarrow (d_1 d_2 \cdots d_q + 1)^2 (2 + 3^q)/(\epsilon^2 \delta)$
  Choose S to be an $m \times (n_1 n_2 \cdots n_q)$ TENSORSKETCH matrix
  Compute $S(A_1 \otimes A_2 \otimes \cdots \otimes A_q)$ and Sb
  $\widetilde{x} \leftarrow \min_x \|S(A_1 \otimes A_2 \otimes \cdots \otimes A_q)x - Sb\|_2$
  return $\widetilde{x}$
end procedure

The proof of Theorem 3.1 can be found in Appendix C.1. Theorem 3.1 shows that we can achieve an ϵ-close solution by solving a much smaller regression problem with a number of samples of order $O(\text{poly}(d/\epsilon))$, which is independent of the large dimension n. Using the technique we introduced in Sec. 2, we can also compute $S\mathcal{A}$ without explicitly computing the tensor product.

We can extend Theorem 3.1 to the nonnegative tensor product regression problem $\min_{x \geq 0} \|(A_1 \otimes A_2 \otimes \cdots \otimes A_q)x - b\|_2$, where $A_i \in \mathbb{R}^{n_i \times d_i}$, $i = 1, \ldots, q$ and $b \in \mathbb{R}^{n_1 n_2 \cdots n_q}$. Suppose x is the optimal solution. Similarly, let $S \in \mathbb{R}^{m \times (n_1 n_2 \cdots n_q)}$ be the matrix form of TENSORSKETCH of Section 2. If $\widetilde{x}$ is the optimal solution to $\min_{x \geq 0} \|S(A_1 \otimes A_2 \otimes \cdots \otimes A_q)x - Sb\|_2$., we have the following:

Corollary 3.2. (Sketch for tensor nonnegative regression) Suppose $\tilde{x} = \min_{x \geq 0} \|S\mathcal{A}x - Sb\|_2$ with TENSORSKETCH $S \in \mathbb{R}^{m \times n}$, where $m = 8(d_1 d_2 \cdots d_q + 1)^2 (2 + 3^q)/(\epsilon^2 \delta)$. Then the following approximation $\|(A_1 \otimes A_2 \otimes \cdots \otimes A_q)\widetilde{x} - b\|_2 \leq (1 + \epsilon)\,\text{OPT}$ holds with probability at least $1 - \delta$, where $\text{OPT} = \min_{x \geq 0} \|(A_1 \otimes A_2 \otimes \cdots \otimes A_q)x - b\|_2$.

The proof of Corollary 3.2 can be found in Appendix C.1.

4 P-SPLINES

B-splines are local basis functions, consisting of low degree (e.g., quadratic, cubic) polynomial segments. The positions where the segments join are called the knots. B-splines have local support and are thus suitable for smoothing and interpolating data with complex patterns. Unfortunately, control over smoothness is limited: one can only change the number and positions of the knots. If there are no reasons to assume that smoothness is non-uniform, the knots will be equally spaced and the only tuning parameter is their (discrete) number. In contrast P-spline (Eilers and Marx, 1996) equally spaces B-splines, discards the derivative completely, and controls smoothness by regularizing the sum of squares of differences of coefficients. Specifically Eilers and Marx Eilers and Marx (1996) proposed the P-spline recipe: (1) use a (quadratic or cubic) B-spline basis with a large number of knots, say 10-50; (2) introduce a penalty on (second or third order) differences of the B-spline coefficients; (3) minimize the resulting penalized likelihood function; (4) tune smoothness with the weight of the penalty, using cross-validation or AIC to determine the optimal weight.

We give a brief overview of B-Splines and P-Splines below. Let b and u, each vectors of length n, represent the observed and explanatory variables, respectively. Once a set of knots is chosen, the B-spline basis A follows from u. If there are d basis functions then A is $n \times d$. In the case of normally distributed observations the model is $b = Ax + e$, with independent errors e. In the case of B-spline regression the sum of squares of residuals $\|b - Ax\|_2$ is minimized and the normal equations $A^\top A\hat{x} = A^\top b$ are obtained; the explicit solution $\hat{x} = (A^\top A)^{-1} A^\top b$ results. The P-spline approach minimizes the penalized least-squares function

$$\|b - Ax\|_2^2 + \lambda \|Lx\|_2^2, \tag{1}$$

where $L \in \mathbb{R}^{p \times n}$ is a matrix that forms differences of order ℓ, i.e., $L_\ell x = \Delta^\ell x$. Examples of this matrix, for $\ell = 1$ and $\ell = 2$ are :

$$L_1 = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}, L_2 = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \end{bmatrix}.$$

The parameter λ determines the influence of the penalty. If λ is zero, we are back to B-spline regression; increasing λ makes $\hat{x}$, and hence $\hat{b} = A\hat{x}$, smoother.

Let x^* denote $\text{argmin}_{x \in \mathbb{R}^d} \|Ax - b\|_2^2 + \lambda \|Lx\|_2^2$, and OPT denote $\|Ax^* - b\|_2^2 + \lambda \|Lx^*\|_2^2$. In general $x^* = (A^\top A + \lambda L^\top L)^{-1} A^\top b = A^\top (AA^\top + \lambda LL^\top)^{-1} b$, so x^* can be found in $O(\texttt{nnz}(A) \min(n, d))$ time using an iterative method (e.g., LSQR). Our first goal in this section is to design faster algorithms that find an approximate $\widetilde{x}$ in the following sense:

$$\|A\widetilde{x} - b\|_2^2 + \lambda \|L\widetilde{x}\|_2^2 \leq (1 + \epsilon)\,\text{OPT}\,. \tag{2}$$

4.1 Sketching for P-Spline

We first introduce a new definition of *Statistical Dimension* that extends the statistical dimension defined for Ridge Regression (i.e., L is an identity matrix in Eq. 1) (Avron et al., 2016) to P-Spline regression.

The problem (1) can also be analyzed by generalized singular value decomposition (GSVD) Golub and Van Loan (2013). For matrices $A \in \mathbb{R}^{n \times d}$ and $L \in \mathbb{R}^{p \times d}$ with $\text{rank}(L) = p$ and $\text{rank}\left(\begin{bmatrix} A \\ L \end{bmatrix}\right) = d$, the GSVD of (A, L) is given by the pair of factorizations $A = U \begin{bmatrix} \Sigma & 0_{p \times (n-p)} \\ 0_{(n-p) \times p} & I_{d-p} \end{bmatrix} RQ^\top$ and $L =$

$V \begin{bmatrix} \Omega & 0_{p\times(n-p)} \end{bmatrix} RQ^\top$, where $U \in \mathbb{R}^{m\times n}$ has orthonormal columns, $V \in \mathbb{R}^{p\times p}$, $Q \in \mathbb{R}^{d\times d}$ are orthogonal, $R \in \mathbb{R}^{d\times d}$ is upper triangular and nonsingular, and Σ and Ω are $p\times p$ diagonal matrices: $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_p)$ and $\Omega = \mathrm{diag}(\mu_1, \mu_2, \ldots, \mu_p)$ with $0 \le \sigma_1 \le \sigma_2 \le \ldots \le \sigma_p < 1$ and $1 \ge \mu_1 \ge \mu_2 \ge \ldots \ge \mu_p > 0$, satisfying $\Sigma^\top\Sigma + \Omega^\top\Omega = I_p$. The *generalized singular values* γ_i of (A, L) are defined by the ratios $\gamma_i = \sigma_i/\mu_i$ $(i = [p])$.

In this section we design an algorithm that is aimed at the case when $n \gg d$. The general strategy is to design a distribution on matrices of size m-by-n (m is a parameter), sample an S from that distribution, and solve $\widetilde{x} \equiv \mathrm{argmin}_{x\in\mathbb{R}^d} \|S(Ax-b)\|_2^2 + \lambda\|Lx\|_2^2$.

The following lemma defines conditions on the distribution of S that guarantees Eq. (2) holds with constant probability (which can be boosted to high probability by repetition and taking the minimum objective).

Lemma 4.1. *Let $x^* \in \mathbb{R}^d$, $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$ as above. Let $U_1 \in \mathbb{R}^{n\times d}$ denote the first n rows of an orthogonal basis for $\begin{bmatrix} A \\ \sqrt{\lambda}L \end{bmatrix} \in \mathbb{R}^{(n+p)\times d}$. Let sketching matrix $S \in \mathbb{R}^{m\times n}$ have a distribution such that with constant probability*

$$\text{(I)}\ \ \|U_1^\top S^\top S U_1 - U_1^\top U_1\|_2 \le 1/4,$$

and

$$\text{(II)}\ \ \|U_1^\top (S^\top S - I)(b - Ax^*)\|_2 \le \sqrt{\epsilon\,\mathrm{OPT}\,/2}.$$

Let $\widetilde{x}$ denote $\mathrm{argmin}_{x\in\mathbb{R}^d} \|S(Ax-b)\|_2^2 + \lambda\|Lx\|_2^2$. Then with probability at least $9/10$,

$$\|A\widetilde{x} - b\|_2^2 + \lambda\|L\widetilde{x}\|_2^2 \le (1+\epsilon)\,\mathrm{OPT}\,.$$

Define the *statistical dimension* for P-Splines as follows:

Definition 4.2 (Statistical Dimension for P-Splines)**.** *For S-Spline in Eq. (1), the statistical dimension is defined as $\mathtt{sd}_\lambda(A, L) = \sum_i 1/(1 + \lambda/\gamma_i^2) + d - p$.*

The following theorem shows that there is a sparse subspace embedding matrix $S \in \mathbb{R}^{m\times n}$ (e.g., CountSketch), with $m \ge K(\mathtt{sd}_\lambda(A, L)/\epsilon + \mathtt{sd}_\lambda(A, L)^2)$, that satisfies Property (I) and (II) of Lemma 4.1, and hence achieves an $\epsilon-$approximation solution to problem 1:

Theorem 4.3. (P-Spline regression) There is a constant $K > 0$ such that for $m \ge K(\epsilon^{-1}\,\mathtt{sd}_\lambda(A, L) + \mathtt{sd}_\lambda(A, L)^2)$ and $S \in \mathbb{R}^{m\times n}$ a sparse embedding matrix (e.g., CountSketch) with SA computable in $O(\mathtt{nnz}(A))$ time, Property (I) and (II) of Lemma 4.1 apply, and with constant probability the corresponding $\widetilde{x} = \mathrm{argmin}_{x\in\mathbb{R}^d} \|S(Ax-b)\|_2 + \lambda\|Lx\|_2^2$ is an ϵ-approximate solution to $\min_{x\in\mathbb{R}^d} \|b - Ax\|_2^2 + \lambda\|Lx\|_2^2$.

Note $\mathtt{sd}_\lambda(A, L)$ is upper bounded by d. The above theorem shows that the statistical dimension allows us to design smaller sketch matrices whose size only depends on $O(\mathrm{poly}(\mathtt{sd}_\lambda(A, L)/\epsilon))$ instead of $O(\mathrm{poly}(d/\epsilon))$, without sacrificing the approximation accuracy.

Algorithm 2 P-Spline Tensor product regression

procedure PTRegression($A, b, K, L, \epsilon, \delta$)
  $m \leftarrow K(\epsilon^{-1}\,\mathtt{sd}_\lambda(\mathcal{A}, L) + \mathtt{sd}_\lambda(\mathcal{A}, L)^2)$
  Choose S to be a $m\times n$ TensorSketch matrix
  Compute $S(A_1 \otimes A_2 \otimes \cdots \otimes A_q)$ and Sb
  $\widetilde{x} = \mathrm{argmin}_{x\in\mathbb{R}^d} \|S(Ax-b)\|_2^2 + \lambda\|Lx\|_2^2$.
  return $\widetilde{x}$
end procedure

4.2 Tensor Sketching for Multi-Dimensional P-Spline

Tensor products allow a natural extension of one-dimensional P-spline smoothing to multi-dimensional P-Spline. We focus on 2-dimensional P-Spline but our results can be generalized to the multi-dimensional setting. Assume that in addition to u we have a second explanatory variable v. We have data triples $(u_i, v_j, b_{(i-1)\cdot n_2+j})$ for $i = 1, \ldots, n_1$ and $j = 1, \ldots, n_2$. We seek a smooth surface $f(u, v)$ which gives a good approximation to the response b. Let A_1, $n \times d_1$, be a B-spline basis along u, and A_2, $n \times d_2$, be a B-spline basis along v. We form the tensor product basis as $A_1 \otimes A_2$. When n_1 and n_2 are large, we do not compute $A_1 \otimes A_2$. We apply TensorSketch here to avoid explicitly forming $A_1 \otimes A_2$ to speed up computation.

Let $X = [x_{kl}]$ be a $d_1 \times d_2$ matrix of coefficients. Then, for given X, the value fit at (u, v) is $f(u, v) = \sum_k \sum_l A_{2,k}(v)A_{1,l}(u)x_{kl}$ and so X may be chosen using least squares by minimizing $\sum_{i,j}[b_{(i-1)\cdot n_2+j} - f(u_i, v_i)]^2 = \sum_i \left[b_{(i-1)\cdot n_2+j} - \sum_k \sum_l A_{2,k}(v_i)A_{1,l}(u_i)x_{kl}\right]^2$. Using Kronecker product, the above minimization can be written in the form $\min \|b - \mathcal{A}x\|_2$, where $\mathcal{A} = A_1 \otimes A_2 \in \mathbb{R}^{n_1 n_2 \times d_1 d_2}$ and $x = \mathrm{vec}(X)$. Again the P-spline approach minimizes the penalized least-squares function $\|b - (A_1 \otimes A_2)x\|_2^2 + \lambda\|Lx\|_2^2$. Consider the tensor p-spline regression problem $\min_x \|(A_1 \otimes A_2 \otimes \cdots \otimes A_q)x - b\|_2^2 + \lambda\|Lx\|_2^2$, where $L \in \mathbb{R}^{p\times n}$, $A_i \in \mathbb{R}^{n_i \times d_i}$, $i = 1, \ldots, q$ and $b \in \mathbb{R}^n$. Let $S \in \mathbb{R}^{m\times n}$ be the matrix form of TensorSketch of Section 2. Algorithm 2 summarizes the procedure for efficiently solving multi-dimensional P-Spline.

Let $\mathcal{A} = A_1 \otimes A_2 \otimes \cdots \otimes A_q$. Replacing the matrix A in Theorem 4.3 with $\mathcal{A}$, we have the following corollary for multi-dimensional P-Spline:

Corollary 4.4 (P-Spline tensor regression)**.** *Suppose $\lambda \le \sigma_1^2/\epsilon$. There is a constant $K > 0$ such that for $m \ge K(\epsilon^{-1}\,\mathtt{sd}_\lambda(\mathcal{A}, L) + \mathtt{sd}_\lambda(\mathcal{A}, L)^2)$ and $S \in \mathbb{R}^{m\times n}$ a TensorSketch matrix with $S\mathcal{A}$ computable in*

$O(\texttt{nnz}(\mathcal{A}))$ *time, Property (I) and Property (II) of Lemma 4.1 apply, and with constant probability the corresponding* $\widetilde{x} = \text{argmin}_{x\in\mathbb{R}^d}\|S(\mathcal{A}x-b)\|_2+\lambda\|Lx\|_2^2$ *is an* ϵ*-approximate solution to* $\min_{x\in\mathbb{R}^d}\|b-\mathcal{A}x\|_2^2+\lambda\|Lx\|_2^2$.

5 TENSOR PRODUCT ABSOLUTE DEVIATION REGRESSION

We extend our previous results for the ℓ_2 norm (i.e., least squares regression) to general ℓ_p norms, with a focus on $p = 1$ (i.e., absolute deviation regression). Specifically we consider $\min_x \|\mathcal{A}x - b\|_p$, where $\mathcal{A} = A_1 \otimes A_2 \otimes \cdots \otimes A_q$. We will show in this section that with probability at least 2/3, we can quickly find an $\tilde{x}$ for which $\|\mathcal{A}\tilde{x} - b\|_1 \leq (1+\epsilon)\min_x \|\mathcal{A}x - b\|_1$.

As in Clarkson and Woodruff (2013), for each i, in $O(\texttt{nnz}(A_i)\log(d_i)) + O(r_i^3)$ time we can replace the input matrix $A_i \in \mathbb{R}^{n_i\times d_i}$ with a new matrix with the same column space of A_i and full column rank. We therefore assume $\mathcal{A}$ has full rank in what follows.

Suppose S is the TENSORSKETCH matrix defined in Section 2. Let $w_i \in \mathbb{N}$ and assume $w_i \mid n_i$. Split A_i into n_i/w_i matrices $A_1^{(i)}, \ldots, A_{n_i/w_i}^{(i)}$, each $w_i \times d_i$, so that $A_j^{(i)}$ is the submatrix of A_i indexed by the j-th block of w_i rows. Note $\mathcal{A}$ can be written as:

$$\begin{bmatrix} A_1^{(1)} \otimes A_1^{(2)} \otimes \cdots \otimes A_1^{(q-1)} \otimes A_1^{(q)} \\ A_1^{(1)} \otimes A_1^{(2)} \otimes \cdots \otimes A_1^{(q-1)} \otimes A_2^{(q)} \\ \vdots \\ A_{n_1/w_1}^{(1)} \otimes A_{n_2/w_2}^{(2)} \otimes \cdots \otimes A_{n_{q-1}/w_{q-1}}^{(q-1)} \otimes A_{n_q/w_q}^{(q)}. \end{bmatrix}.$$

For each $A_{i_1}^{(1)} \otimes A_{i_2}^{(2)} \otimes \cdots \otimes A_{i_q}^{(q)}$, we can use the TENSORSKETCH matrix $S_{i_1 i_2 \ldots i_q} \in \mathbb{R}^{m\times\prod_{i=1}^q w_i}$, where we set $m \geq 100\prod_{i=1}^q d_i^2(2+3^q)/\epsilon^2$, such that with probability at least .99, $\|S_{i_1 i_2\ldots i_q}A_{i_1}^{(1)}\otimes A_{i_2}^{(2)}\otimes\cdots\otimes A_{i_q}^{(q)}x\|_2 = (1\pm\epsilon)\|A_{i_1}^{(1)} \otimes A_{i_2}^{(2)} \otimes \cdots \otimes A_{i_q}^{(q)}x\|_2$ simultaneously for all $x \in \mathbb{R}^d$ as $S_{i_1 i_2\ldots i_q}$ is an oblivious subspace embedding (Lemma. B.3) . Now we use Algorithm 4 from (Liang et al., 2014) to boost the success probability by computing $t = O(\log(1/\delta))$ independent TENSORSKETCH products $S_{i_1 i_2\ldots i_q}^{(j)}A_{i_1}^{(1)}\otimes A_{i_2}^{(2)}\otimes\cdots\otimes A_{i_q}^{(q)}$, $j = [t]$, each with only constant success probability, and then running a cross validation procedure similar to that in Algorithm 4 of Liang et al. (2014), to find one which succeeds with probability at least $1-\delta$:

Lemma 5.1 ((Liang et al., 2014)). *For* $\delta, \epsilon \in (0,1)$, *let* $t = O(\log 1/\delta)$ *and* $m \geq 100\prod_{i=1}^q d_i^2(2+3^q)/\epsilon^2$. *Running algorithm Algorithm 4 in (Liang et al., 2014) with parameters* t, m, *we can obtain a* TENSORSKETCH $S \in \mathbb{R}^{m\times\prod_{i=1}^q w_i}$ *such that with probability at least* $1-\delta$, $\|SA_{i_1}^{(1)} \otimes A_{i_2}^{(2)} \otimes \cdots \otimes A_{i_q}^{(q)}x\|_2 = (1\pm\epsilon)\|A_{i_1}^{(1)} \otimes A_{i_2}^{(2)} \otimes \cdots \otimes A_{i_q}^{(q)}x\|_2$ *for all* $x \in \mathbb{R}^d$.

After computing a TENSORSKETCH $S_{i_1 i_2\ldots i_q} \in \mathbb{R}^{m\times\prod_{i=1}^q w_i}$ using lemma 5.1 for each row-block $A_{i_1}^{(1)} \otimes \ldots \otimes A_{i_q}^{(q)}$, we compose a single TENSORSKETCH for $\mathcal{A}$ as $S = \text{diag}(S_{1,\cdots,1}, \cdots, S_{i_1,\cdots,i_q}, \cdots, S_{(n_1/w_1),\cdots,(n_q/w_q)}) \in \mathbb{R}^{m\prod_{i=1}^q n_i/w_i\times\prod_{i=1}^q n_i}$, which is defined as:

$$\begin{bmatrix} S_{1,\cdots,1} & & & & \\ & \ddots & & & \\ & & S_{i_1,\cdots,i_q} & & \\ & & & \ddots & \\ & & & & S_{(n_1/w_1),\cdots,(n_q/w_q)} \end{bmatrix},$$

where each block on the diagonal is from Lemma 5.1. Note that $\mathcal{A}$ has in total $\prod_{i=1}^q(n_i/w_i)$ many blocks. Using Lemma 5.1 with a union bound over all blocks of $\mathcal{A}$, we have the following theorem which shows S is an oblivious subspace embedding for $\mathcal{A}$ in ℓ_2 norm:

Theorem 5.2 (ℓ_2 OSE for tensor matrices). *Given* $\delta, \epsilon \in (0,1)$, *let* $S \in \mathbb{R}^{m\prod_{i=1}^q n_i/w_i\times\prod_{i=1}^q n_i}$ *denote the matrix that has* $\prod_{i=1}^q n_i/w_i$ *diagonal block matrices where each diagonal block* $S_{i_1 i_2\ldots i_q} \in \mathbb{R}^{m\times\prod_{i=1}^q w_i}$ *is from Lemma 5.1. With probability at least* $1-\prod_{i=1}^q(n_i/w_i)\delta$, $\|S\mathcal{A}x\|_2 = (1\pm\epsilon)\|\mathcal{A}x\|_2, \forall x \in \mathbb{R}^d$.

It is known that for any matrix $A \in \mathbb{R}^{n\times r}$, we can compute a change of basis $U \in \mathbb{R}^{r\times r}$ such that AU is an (α,β,p) well-conditioned basis of A (see Definition 1.1), in time polynomial with respect to n, r (Dasgupta et al., 2009). Specifically, Theorem 5 in (Dasgupta et al., 2009) shows that we can compute a change of basis U for which AU is a well-conditioned basis of A in time $O(nr^5\log(n))$ time. However we cannot afford to directly use the results from (Dasgupta et al., 2009) to compute a well-conditioned basis for $\mathcal{A}$, which requires time at least $\Omega(n)$. Instead we compute a well-conditioned basis for $\mathcal{A}$ through a sketch $S\mathcal{A}$ where S is the TENSORSKETCH from Theorem 5.2.

Specifically we define the following procedure $\mathsf{Condition}(\mathcal{A})$ for computing a well-conditioned basis for $\mathcal{A}$. Given $\mathcal{A} = A_1 \otimes \cdots \otimes A_q \in \mathbb{R}^{n\times r}$, 1) Compute $S\mathcal{A}$; 2) Compute a $d\times d$ change of basis matrix U so that $S\mathcal{A}U$ is an (α,β,p)-well-conditioned basis of the column space of $S\mathcal{A}$; 3) Output $\mathcal{A}U/(d\gamma_p)$, where $\gamma_p \equiv \sqrt{2}t^{1/p-1/2}$ for $p \leq 2$, and $\gamma_p \equiv \sqrt{2}w^{1/2-1/p}$ for $p \geq 2$, where $t = \prod_{i=1}^q m_i$ and $w = \prod_{i=1}^q w_i$. The following Lemma 5.3 (the proof can be found in the Appendix) is the analogue of that in Clarkson et al. (2013) proved for the Fast Johnson Lindenstauss Transform.

Lemma 5.3. *For any* $p \geq 1$. $\mathsf{Condition}(\mathcal{A})$ *computes* $\mathcal{A}U/(d\gamma_p)$ *which is an* $(\alpha, \beta\sqrt{3}d(tw)^{|1/p-1/2|}, p)$*-well-conditioned basis of* $\mathcal{A}$, *with probability at least* $1-\prod_{i=1}^q(n_i/w_i)\delta$.

Lemma 5.3 indicates that Condition($\mathcal{A}$) computes a $(\alpha, \beta \cdot \mathrm{poly}(\max(d, \log n)), p)$-well-conditioned basis. A well-conditioned basis can be used to solve ℓ_p regression problems, via sampling a subset of rows of the well-conditioned basis $\mathcal{A}U$ with probabilities proportional to the p-th power of the ℓ_p norm of the rows (Woodruff, 2014). However the first issue for sampling is that we cannot afford to compute $\mathcal{A}U$ as this requires $O(\mathtt{nnz}(\mathcal{A})d)$ time. To fix this, we apply a Gaussian sketch matrix $G \in \mathbb{R}^{d \times \log(n)}$ with i.i.d normal random variables (Drineas et al., 2011) on the right hand side of $\mathcal{A}U$. Note that $\mathcal{A}UG$ can be computed efficiently by first compuing UG and then $\mathcal{A}(UG)$ in time $O(d^2 \log(n) + nnz(\mathcal{A}) \log(n))$. The second issue is that even with $\mathcal{A}UG$, computing the ℓ_p norm of each row of $\mathcal{A}UG$ takes $O(n)$ time, but we want sublinear time. This leads us to the following sampling technique.

The high level idea is that since $\mathcal{A}UG$ only has $O(\log(n))$ columns, we can afford to sample columns of $\mathcal{A}UG$ with probability proportional to the p-th power of the ℓ_p norms of the columns, if we can efficiently estimate the ℓ_p norms of the columns (note that naïvely computing the ℓ_p norm of a column also takes $O(n)$ time). Let us denote the first of column of UG as $e \in \mathbb{R}^{\log(n)}$. We focus on how to efficiently estimate the ℓ_p norm of the first column of $\mathcal{A}UG$, which is $\mathcal{A}e$, and all the left columns can be estimated in the same way. We first reshape the vector $\mathcal{A}e$ into a 2-d matrix

$$M = A_1 \otimes ... \otimes A_{q_1} E (A_{q_1+1} \otimes ... \otimes A_q)^\top, \quad (3)$$

where $M \in \mathbb{R}^{(n_1 n_2 ... n_{q_1}) \times (n_{q_1+1} ... n_q)}$, E is obtained from reshaping e into a $(d_1 d_2 ... d_{q_1}) \times (d_{q_1+1} .. d_q)$ matrix and $q_1 \in [1, q]$ is chosen such that $(n_1 n_2 ... n_{q_1}) \approx (n_{q_1+1} ... n_q)$. Namely we reshape the column $\mathcal{A}e$ into a (nearly) square matrix. Focusing now on $p = 1$, note that $\|\mathcal{A}e\|_1 = \sum_{i=1}^{n_{q_1+1} ... n_q} \|M_{*,i}\|_1$. Hence to estimate $\|\mathcal{A}e\|_1$, we only need to estimate the ℓ_1 norm of the columns of M.

Let us apply a sketch matrix $R \in \mathbb{R}^{O(\log(n)) \times \prod_{i=1}^{q_1} n_i}$, whose entries are sampled i.i.d. from the Cauchy distribution, to the left hand side of M. For the i'th column of M, let us define random variables $z_l^i = R_{l,*}^\top M_i$, for $l \in [O(\log(n))]$, where $R_{l,*}$ is the l'th row of R. Due to the 1-stability property of the Cauchy distribution, we have that $\{z_l^i\}_{l=1}^{O(\log n)}$ are $O(\log n)$ independent Cauchys scaled by $\|M_i\|_1$. Applying a Chernoff bound to independent half-Cauchys (see Claims 1, 2 and Lemmas 1, 2 in Indyk (2006)), we have $0.5\|M_i\|_1 \leq \mathtt{median}_{l \in [O(\log n)]} \{|z_j^i|\} \leq 1.5\|M_i\|_1$ with probability at least $1 - 2e^{-cO(\log(n))}$, with constant $c \geq 0.07$. Denote the median of $\{|z_l^i|\}_{l=1}^{O(\log(n))}$ as λ_i. By a union bound over all columns of M, we have that

Algorithm 3 ℓ_1 tensor product regression

1: **procedure** L1TRREGRESSION(A, b, ϵ, δ)
2: Construct a TENSORSKETCH $S \in \mathbb{R}^{(m \prod_{i=1}^q \frac{n_i}{w_i}) \times n}$.
3: Run Condition($\mathcal{A}$) using S to compute $U/(d\gamma_p)$.
4: Generate a Gaussian matrix $G \in \mathbb{R}^{d \times O(\log(n))}$ and a Cauchy sketch matrix $R \in \mathbb{R}^{\log(n) \times n}$.
5: **for** for each column e in UG **do**
6: Reshape $\mathcal{A}e$ to M (Eq. 3).
7: Compute λ_i and $\lambda_e = \sum_i \lambda_i$ (Eq 4 and 5).
8: **end for**
9: **for** $i \in [\sqrt{\prod_{i=1}^q w_i}\, \mathrm{poly}(d)]$ **do**
10: Sample a column $(AUG)_{*,e}$ with probability proportional to λ_e. . ▷ $e \in [O(\log(n))]$
11: Reshape $(AUG)_{*,e}$ to M, sample a column $M_{*,j}$ with probability proportional to λ_j.
12: Sample an entry $M_{k,j}$ with probability proportional to $|M_{k,j}|$.
13: Convert (k, j) back to the corresponding row index in $\mathcal{A}$, denoted as r_i.
14: **end for**
15: $\widetilde{x} \leftarrow \min_x \|D(A_1 \otimes A_2 \otimes \cdots \otimes A_q)x - Db\|_1$, where D is a diagonal matrix to select the $r_1, r_2, ..., r_N$-th rows of $\mathcal{A}$ and b with $N = \sqrt{\prod_{i=1}^q w_i}\, \mathrm{poly}(d)$.
16: **return** $\widetilde{x}$ ▷ $\widetilde{x} \in \mathbb{R}^{d_1 d_2 \cdots d_q}$
17: **end procedure**

with probability at least $1 - n_{[q]\setminus[q_1]} 2e^{-cO(\log(n))}$:

$$\lambda_i = (1 \pm 0.5)\|M_i\|_1, \forall i \in [n_{[q]\setminus[q_1]}], \quad (4)$$

$$\lambda_e = \sum_{i=1}^{n_{[q]\setminus[q_1]}} \lambda_i = (1 \pm 0.5)\|\mathcal{A}e\|_1 \quad (5)$$

where $n_{[q]\setminus[q_1]} = \prod_{i=q_1+1}^q n_i$.

Note that since $\mathcal{A}UG$ only has $O(\log n)$ columns, we can afford to compute the ℓ_1 norm of all the columns using the above procedure. Let us denote the ℓ_1 norms of the columns of AUG by $\lambda_1, \lambda_2, ..., \lambda_{O(\log n)}$. We can sample a column $(\mathcal{A}UG)_{*,i}$ with probability proportional to λ_i (Line 10 in Alg. 3). Once we sample a column $\mathcal{A}UG_{*,i}$, we need to sample an entry $j \in [n]$ with probability proportional to the absolute value of the entry $|(\mathcal{A}UG)_{j,i}|$. As we cannot afford to compute $|(\mathcal{A}UG)_{j,i}|$ for all $j \in [n]$, we use the reshaped 2-d matrix M of $(\mathcal{A}UG)_{*,i}$. Note that sampling an entry in M with probability proportional to the absolute values of entries of M is equivalent to sampling an entry $j \in [n]$ from $(\mathcal{A}UG)_{*,i}$ with probability proportional to the absolute value of the entries of $(\mathcal{A}UG)_{*,i}$. We first sample a column $M_{*,j}$ from all the columns of M with probability proportional to λ_j for $j \in [\prod_{i=q_1+1}^q n_i]$. We then sample an entry $M_{k,j}$ with probability proportional to $|M_{k,j}|$ for $k \in [\prod_{i=1}^{q_1} n_i]$. Noting that $k \in [\prod_{i=1}^{q_1} n_i]$ and $j \in [\prod_{i=q_1+1}^q n_i]$, the pair (k, j) uniquely deter-

mines a corresponding row index r in $\mathcal{A}$, for some $r \in [\prod_{i=1}^{q} n_i]$. Hence we successfully sample a row from $\mathcal{A}UG$ without ever computing the ℓ_p norm of the rows. The above sampling procedure is summarized in Line 10 to Line 13 in Alg. 3. We use the above procedure to sample $\sqrt{\prod_{i=1}^{q} w_i}\,\mathrm{poly}(d)$ rows of $\mathcal{A}$. Let D be a diagonal matrix that selects the corresponding sampled rows from $\mathcal{A}$. We can now solve a smaller ADL problem as $\min_x \|D\mathcal{A} - Db\|_1$. Note that our analysis focuses on the ℓ_1 norm. We can extend the analysis to general ℓ_p norms by using a sketching matrix $R \in \mathbb{R}^{O(\log n)\times\prod_{i=1}^{q_1} n_i}$ with entries sampled i.i.d. from a p-stable distribution for $p \in [1, 2]$.

We now present our main theorem and defer the proofs to the appendix:

Theorem 5.4. (Main result) Given $\epsilon \in (0, 1)$, $\mathcal{A} \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$, Alg. 3 computes $\widehat{x}$ such that with probability at least $1/2$, $\|\mathcal{A}\hat{x} - b\|_1 \leq (1 + \epsilon)\min_{x\in\mathbb{R}^d} \|\mathcal{A}x - b\|_1$. For the special case when $q = 2$, $n_1 = n_2$, the algorithm's running time is $O({n_1}^{3/2}\,\mathrm{poly}(\prod_{i=1}^{2} d_i/\epsilon))$.

For the special case where $q = 2$ and $n_1 = n_2$, we can see from theorem 5.4 our algorithm computes $\widehat{x}$ in time $O({n_1}^{3/2}\,\mathrm{poly}(\hat{d}))$, which is faster than $O({n_1}^2)$— the time needed for forming $A_1 \otimes A_2$. Note that we can run Alg. 3 $O(\log(1/\delta))$ times independently and pick the best solution among these independent runs to boost the success probability to be $1 - \delta$, for $\delta \in [0, 1)$.

6 NUMERICAL EXPERIMENTS

We generate matrices A_1, A_2 and b with all entries sampled i.i.d from a normal distribution. The baseline we compared to is directly solving regression without sketching. We let $\mathtt{T}_1$ be the time for directly solving the regression problem, and $\mathtt{T}_2$ be the time of our algorithm. The time ratio is $r_t = \mathtt{T}_2/\mathtt{T}_1$. The relative residual percentage is defined by $r_e = \frac{100\left|\|(A_1\otimes A_2)\widetilde{x}-b\|_2 - \|(A_1\otimes A_2)x^*-b\|_p\right|}{\|(A_1\otimes A_2)x^*-b\|_p}$, where $\widetilde{x}$ is the output of our algorithms and x^* is the optimal solution. Throughout the simulations, we use a moderate input matrix size in order to accommodate the brute force algorithm and to compare to the exact solution.

Example 6.1 (ℓ_2 Regression)**.** *We create a design matrix with moderate size by fixing $n_1 = n_2 = 300$ and $d_1 = d_2 = 15$. Thus $\mathcal{A} \in \mathbb{R}^{90000\times 225}$. We do 10 rounds to compute the mean values of r_e and r_t, which is reported in Table 1 (Left).*

From Table 1 (left), we can see that when the number m of sampled rows in Algorithm 1 increases from 8000 to 12000, the mean values of r_e decrease while the mean values of r_t increase. In general we can see that we can achieve around 1% relative error while being 5 times faster than the direct method.

Table 1: Examples 6.1 and 6.2: the values of r_e and r_t with respect to different sampling parameters m.

	m	r_e	r_t		m	r_e	r_t
ℓ_2	8000	1.79%	0.11	ℓ_1	8000	1.89%	0.06
	12000	1.24%	0.18		12000	1.33%	0.11
	16000	1.01%	0.25		16000	0.992%	0.18

Table 2: Example 6.3: the mean values of r_e and r_t with respect to different sampling parameters m and regularization parameters λ.

λ	m	r_e	r_t
1	2000	4.43e-2%	0.52
	4000	2.99e-2%	0.70
	6000	7.92e-2%	0.91
0.1	2000	6.98e-2%	0.47
	4000	4.07e-2%	0.72
	6000	5.41e-2%	0.94
0.01	2000	3.78e-2%	0.46
	4000	1.07e-1%	0.71
	6000	2.97e-2%	0.95

Example 6.2 (ℓ_1 Regression)**.** *We set $n_1 = n_2 = 300$, $d_1 = d_2 = 15$. For $\min_x \|Ax - b\|_1$, we solve it by a Linear Programming solver in Gurobi (Gurobi Optimization, 2016). We tested different numbers of sampled rows m (the number of rows in D in Alg. 3). The results are summarized in Table 1 (Right).*

As we can see from Table 1 (right), our method is around 10 times faster than directly solving the problem, with relative error only around 1%.

Example 6.3 (P-Spline Regression)**.** *For P-spline regression, L_3 is fixed. We use 30 knots and cubic B-splines. The data $u = (u_i) \in \mathbb{R}^{n_1}, v = (v_j) \in \mathbb{R}^{n_2}$ and $b \in \mathbb{R}^{n_1 n_2}$ are generated i.i.d from the normal distribution. We compute the B-spline basis matrices $A_1 \in \mathbb{R}^{n_1\times d_1}$ and $A_2 \in \mathbb{R}^{n_2\times d_2}$ separately, if there are d_1 and d_2 basis functions for the B-spline. We set $n_1 = n_2$, $d_1 = d_2$, with $n_1 n_2 = 10^4$ and $d_1 d_2 = 529$. The original and sketched P-spline regression problem are both solved by* Regularization Tools *(Hansen, 1994) via computing their GSVDs. We test different choices of λ. The results are shown in Table 2.*

From Table 2, sampling only 20% of the rows can give around 0.05% relative error, while the computation time is half of the time of directly solving the p-spline.

ACKNOWLEDGEMENT

Wen Sun is supported in part by Office of Naval Research contract N000141512365 and H.-A. Diao was supported in part by the Fundamental Research Funds for the Central Universities under the grant 2412017FZ007.

References

H. Avron, H. Nguyen, and D. Woodruff. Subspace embeddings for the polynomial kernel. In *Advances in Neural Information Processing Systems(NIPS)*, pages 2258–2266, 2014.

H. Avron, K. L. Clarkson, and D. P. Woodruff. Sharper bounds for regression and low-rank approximation with regularization. *CoRR*, abs/1611.03225, 2016.

L. Carter and M. N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979.

M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *Theor. Comput. Sci.*, 312(1):3–15, 2004.

D. Chen and R. J. Plemmons. Nonnegativity constraints in numerical analysis. In *The birth of numerical analysis*, pages 109–139. World Sci. Publ., Hackensack, NJ, 2010.

K. Clarkson, P. Drineas, M. Magdon-Ismail, M. Mahoney, X. Meng, and D. P. Woodruff. The fast Cauchy transform and faster robust linear regression. In *SODA*, 2013.

K. L. Clarkson. Subgradient and sampling algorithms for ℓ_1 regression. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 257–266, 2005.

K. L. Clarkson and D. P. Woodruff. Low rank approximation and regression in input sparsity time. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 81–90. https://arxiv.org/pdf/1207.6365, 2013.

A. Dasgupta, P. Drineas, B. Harb, R. Kumar, and M. W. Mahoney. Sampling algorithms and coresets for ℓ_p regression. *SIAM J. Comput.*, 38(5):2060–2078, 2009.

P. Drineas, M. Magdon-Ismail, M. W. Mahoney, and D. P. Woodruff. Fast approximation of matrix coherence and statistical leverage. *CoRR*, abs/1109.3843, 2011.

P. H. Eilers and B. D. Marx. Multidimensional density smoothing with p-splines. In *Proceedings of the 21st international workshop on statistical modelling*, 2006.

P. H. C. Eilers and B. D. Marx. Flexible smoothing with B-splines and penalties. *Statist. Sci.*, 11(2): 89–121, 1996.

P. H. C. Eilers, B. D. Marx, and M. Durbán. Twenty years of P-splines. *SORT*, 39(2):149–186, 2015. ISSN 1696-2281.

D. W. Fausett and C. T. Fulton. Large least squares problems involving Kronecker products. *SIAM J. Matrix Anal. Appl.*, 15(1):219–227, 1994.

G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, 2013.

I. Gurobi Optimization. Gurobi optimizer reference manual, 2016. URL http://www.gurobi.com.

P. C. Hansen. Regularization tools: A matlab package for analysis and solution of discrete ill-posed problems. *Numerical Algorithms*, 6(1):1–35, 1994.

P. Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006.

R. Kannan and S. Vempala. Spectral algorithms. *Foundations and Trends in Theoretical Computer Science*, 4(3-4):157–288, 2009.

Y. Liang, M.-F. F. Balcan, V. Kanchanapally, and D. Woodruff. Improved distributed principal component analysis. In *Advances in Neural Information Processing Systems*, pages 3113–3121, 2014.

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

X. Meng and M. W. Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 91–100. ACM, https://arxiv.org/pdf/1210.3135, 2013.

J. Nelson and H. L. Nguyên. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 117–126. IEEE, https://arxiv.org/pdf/1211.1002, 2013.

S. Oh, S. Kwon and J. Yun. A method for structured linear total least norm on blind deconvolution problem. *Applied Mathematics and Computing*, 19: 151–164, 2005.

R. Pagh. Compressed matrix multiplication. *ACM Trans. Comput. Theory*, 5(3):9:1–9:17, 2013.

M. Patrascu and M. Thorup. The power of simple tabulation hashing. *J. ACM*, 59(3):14, 2012.

N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining(KDD)*, pages 239–247. ACM, 2013.

P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection*, volume 589. John wiley & sons, 2005.

C. Sohler and D. P. Woodruff. Subspace embeddings for the ℓ_1-norm with applications. In *Proceedings of the forty-third annual ACM symposium on Theory of computing (STOC)*, pages 755–764. ACM, 2011.

Z. Song, D. P. Woodruff, and P. Zhong. Low rank approximation with entrywise ℓ_1-norm error. In *Proceedings of the 49th Annual Symposium on the Theory of Computing (STOC)*. ACM, https://arxiv.org/pdf/1611.00898, 2017a.

Z. Song, D. P. Woodruff, and P. Zhong. Relative error tensor low rank approximation. *arXiv preprint arXiv:1704.08246*, 2017b.

C. Van Loan. *Computational frameworks for the fast Fourier transform*, volume 10 of *Frontiers in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1992.

C. F. Van Loan and N. Pitsianis. Approximation with Kronecker products. In *Linear algebra for large scale and real-time applications (Leuven, 1992)*, volume 232 of *NATO Adv. Sci. Inst. Ser. E Appl. Sci.*, pages 293–314. Kluwer Acad. Publ., Dordrecht, 1993.

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

Y. Zheng, G. Liu, S. Sugimoto, S. Yan, and M. Okutomi. Practical low-rank matrix approximation under robust l 1-norm. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1410–1417. IEEE, 2012.