Solving Attention Kernel Regression Problem via Pre-conditioner

Zhao Song
Adobe Research

Junze Yin
Boston University

Lichen Zhang
Massachusetts Institute of Technology

Abstract

Attention mechanism is the key of large language models, and attention matrix serves as algorithmic and computational bottleneck for such scheme. In this paper, we define two problems, motivated by designing fast algorithms for *proxy* of attention matrix and solving regressions against them. Given an input matrix $A \in \mathbb{R}^{n \times d}$ with $n \gg d$ and a response vector b, we first consider the matrix exponential of the matrix $A^\top A$ as a proxy, and we in turn design algorithms for two types of regression problems: $\min_{x \in \mathbb{R}^d} \|(A^\top A)^j x - b\|_2$ and $\min_{x \in \mathbb{R}^d} \|A(A^\top A)^j x - b\|_2$ for any positive integer j. Studying algorithms for these regressions are essential, as matrix exponential can be approximated term-by-term via these smaller problems. The second proxy is applying exponential entrywise to the Gram matrix, denoted by $\exp(AA^\top)$ and solving the regression $\min_{x \in \mathbb{R}^n} \|\exp(AA^\top)x - b\|_2$. We call this problem the *attention kernel regression* problem, as the matrix $\exp(AA^\top)$ could be viewed as a kernel function with respect to A. We design fast algorithms for these regression problems, based on sketching and preconditioning. We hope these efforts would provide an alternative perspective of studying efficient approximation of attention matrices.

1 INTRODUCTION

Many numerical linear algebra tasks admit efficient, randomized solutions (Woolfe et al., 2008; Klein et al., 1994; Wang and Eppstein, 2006; Azar et al., 2001). These problems include the approximating leverage scores, least squares regression, and low rank approximation, and they have numerous applications in areas such as recommendation systems (Drineas et al., 2002), data mining (Azar et al., 2001), web search (Achlioptas et al., 2001; Kleinberg, 1999), information retrieval (Papadimitriou et al., 1998), learning mixtures of distributions (Achlioptas and McSherry, 2005; Kannan et al., 2008), and clustering (Drineas et al., 2004; McSherry, 2001). The deployment of randomization and approximation enables these problems to be solved much more rapidly than their deterministic and exact counterpart.

In the modern big data era in which the size of the matrices grows rapidly, these randomized approaches become even more appealing as most of them could leverage the structure of the input matrix, and subsequently solve the problem in time nearly linear of the data. A prominent randomized paradigm for numerical linear algebra is the *sketching* approach (Sarlos, 2006; Clarkson and Woodruff, 2017). Roughly speaking, given a tall and skinny matrix $A \in \mathbb{R}^{n \times d}$ with $n \gg d$, one draws a random matrix $S \in \mathbb{R}^{m \times n}$ from a structured family and computes the product SA. Based on the choice of m, the random matrix S can either preserve the column norms of A (Johnson and Lindenstrauss, 1984) or the entire subspace spanned by A (Sarlos, 2006). Moreover, the structure of S oftentimes enables the matrix product SA to be computed very efficiently (Ailon and Chazelle, 2006; Clarkson and Woodruff, 2017; Lu et al., 2013; Nelson and Nguyên, 2013; Kane and Nelson, 2014). Such an approach has found many applications including linear regression (Clarkson and Woodruff, 2017), low rank approximation (Clarkson and Woodruff, 2017; Boutsidis et al., 2014) and kernel approximation (Avron et al., 2014; Ahle et al., 2020; Song et al., 2021a).

In this paper, we study the efficient computation and approximation of *attention matrices.* These matrices are fundamental objects of deep learning models utilized in a wide range of domains, including natural language processing (Jiao et al., 2019), computer vision (Guo et al., 2022), speech recognition (Chorowski et al., 2015; Wang et al., 2021), and robotics (Liu et al., 2022). Attention mechanisms enable models to focus selectively on specific portions of input data and dynamically adjust weights for distinct features and

---

Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS) 2024, Valencia, Spain. PMLR: Volume 238. Copyright 2024 by the author(s).

context information. The attention matrix is a crucial component of attention mechanisms, capturing the relationships between input elements and the query vector. By computing the attention matrix, models can learn to attend to pertinent information and disregard irrelevant information, leading to improved performance and interpretability. Furthermore, the attention matrix provides valuable insights into how models make decisions and reason about input data, aiding in debugging and enhancing models. Therefore, understanding the attention matrix is crucial for comprehending the behavior and limitations of deep learning models and developing more potent attention mechanisms. Recent studies have shown that attention mechanisms can be applied to other problems beyond traditional deep learning, such as graph neural networks (Zhang et al., 2020; Gao et al., 2022), reinforcement learning (Bramlage and Cortese, 2022), and meta-learning (Sokar et al., 2021).

As indicated by Alman and Song (2023), efficient approximation of attention matrices requires the structural condition that the entries are bounded given standard complexity theoretical assumptions. On the flip side, they also show these matrices can be approximated in $O(n^{1+o(1)})$ time if certain parameters are desirable. While this seems to close the algorithmic study of approximating attention matrices, it doesn't bar the study of *proxy* for these objects, i.e., components of attention matrices that either admit different structural assumptions, bypassing bounded entries; or different type of matrices which serve as a replacement for the standard attention. We hence introduce two kinds of proxy: the first uses matrix exponential, and the second applies exponential function entrywise to a Gram matrix. We then consider solving regressions against these proxies.

Definition 1.1. *Given an input matrix* $A \in \mathbb{R}^{n \times d}$*, a response* $b \in \mathbb{R}^n$*, the goal is to solve*

$$\min_{x \in \mathbb{R}^d} \|AA^\top Ax - y\|_2^2$$

Definition 1.2. *Given an input matrix* $A \in \mathbb{R}^{n \times d}$*, a response* $b \in \mathbb{R}^d$*, the goal is to solve*

$$\min_{x \in \mathbb{R}^d} \|A^\top AA^\top Ax - y\|_2^2$$

Both of these problems serve as primitives for solving regressions for matrix exponential. As we will see later, by induction, algorithms for these problems naturally extend to higher even and odd powers:

$$\min_{x \in \mathbb{R}^d} \|(A^\top A)^j x - b\|_2 \tag{1}$$

and

$$\min_{x \in \mathbb{R}^d} \|A(A^\top A)^j x - b\|_2, \tag{2}$$

where d and j are arbitrary natural numbers. These sub-problems subtly relate to regression against matrix exponential, as each term of the matrix exponential can be approximated via solving these smaller problems for each power.

The next problem concerns applying entrywise exponential to a Gram matrix. As it could be interpreted as applying an exponential kernel function to the data A, we call it *attention kernel regression* or *exponential regression.*

Definition 1.3 (Attention Kernel Regression (or Exponential Regression))**.** *Given an input matrix* $A \in \mathbb{R}^{n \times d}$*, a response* $b \in \mathbb{R}^n$*, the goal is to solve*

$$\min_{x \in \mathbb{R}^n} \|\exp(AA^\top)x - b\|_2^2$$

where $\exp(\cdot)$ *is applied entrywise to the matrix* $AA^\top$*.*

The attention kernel regression problem owing its name to the attention mechanism, as it could be viewed as a simplification of approximating one critical component of the attention matrix.

1.1 Our Result

We present the informal version of our main result below.

Theorem 1.4 (Informal Version of Theorem G.2)**.** *Let* $A \in \mathbb{R}^{n \times d}$*,* $b \in \mathbb{R}^d$*, and* κ *denote the condition number of* A*. Let* $\epsilon_{\mathrm{final}}, \delta_{\mathrm{final}} \in (0, 0.1)$*.*

For the regression problem shown in Eq. (1), there exists an algorithm (Algorithm 1) that runs in time

$$O((nd + d^3) \cdot j \cdot \log(\kappa/\epsilon_{\mathrm{final}}) \cdot \log^2(jn/\delta_{\mathrm{final}}))$$

and outputs a vector $x' \in \mathbb{R}^d$ *such that* $\|(A^\top A)^j x' - b\|_2 \leq \epsilon_{\mathrm{final}} \|b\|_2$ *holds with probability* $1 - \delta_{\mathrm{final}}$*.*

Theorem 1.5 (Informal Version of Theorem G.3)**.** *Let* $A \in \mathbb{R}^{n \times d}$*,* $b \in \mathbb{R}^n$*, and* κ *denote the condition number of* A*. Let* $\epsilon_{\mathrm{final}}, \delta_{\mathrm{final}} \in (0, 0.1)$*.*

For the regression problem shown in Eq. (2), there exists an algorithm (Algorithm 2) that runs in time

$$O((nd + d^3) \cdot j \cdot \log(\kappa/\epsilon_{\mathrm{final}}) \cdot \log^2(jn/\delta_{\mathrm{final}}))$$

and outputs a vector $x' \in \mathbb{R}^d$ *such that* $\|A(A^\top A)^j x' - b\|_2 \leq \epsilon_{\mathrm{final}} \|b\|_2$ *holds with probability* $1 - \delta_{\mathrm{final}}$*.*

Before proceeding, we highlight the significant speedup obtained by our results. Note that if we try to compute the product $(A^\top A)^j$ directly, it will take $O(j \cdot nd^2)$ time. One could also utilize the squaring trick to compute this product in $O(\log j \cdot nd^2)$ time. In contrast, our algorithm runs in time $O(j \cdot nd)$ for $n \gg d$. This is a significant improvement as long as $j \leq \log j \cdot d$, which is often the case when j is orders smaller than d.

Our result regarding the attention kernel regression requires some extra assumptions, in particular, we focus on the regime where $n \leq d$. Now we are ready to state our result.

Theorem 1.6 (Informal Version of Theorem H.7)**.** *Let $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$, and κ denote the condition number of A. Let $\epsilon_{\mathrm{final}}, \delta_{\mathrm{final}} \in (0, 0.1)$. We can find x' such*

$$\| \exp(AA^\top)x' - b\|_2 \leq \epsilon_{\mathrm{final}} \cdot \|b\|_2$$

Let $m = O(\epsilon_{\mathrm{final}}^{-2} \beta \log^3(\kappa nd/(\epsilon_{\mathrm{final}} \delta_{\mathrm{final}})))$ be sketching dimension and β be a parameter to be specified later. Then, the vector $x' \in \mathbb{R}^n$ can be computed in time

$$O(mn + \epsilon_{\mathrm{final}}^{-2} nd + m^3).$$

The parameter β is an upper bound on the rank of the following sequence of matrices:

$$\{A, A^{\otimes 2}, A^{\otimes 3}, \dots, A^{\otimes q}\},$$

for $q = O(r^2 + \log(n/\epsilon_{\mathrm{final}})$ and r is the radius of rows of A, $A^{\otimes l} \in \mathbb{R}^{n \times d^l}$ is the matrix where each row of the matrix is the self-tensoring of corresponding row of A for l times.

Note that if we compute the kernel $\exp(AA^\top)$, it would take $O(n^2 d)$ time. If we pose a naive upper bound $\beta \leq n$, then as long as $d \geq \epsilon_{\mathrm{final}}^{-6} n \log^9(\kappa nd/\epsilon_{\mathrm{final}} \delta_{\mathrm{final}})$, our algorithm is faster than exact computation.

1.2 Related Work

Least-Squares Regression. The fitting method referred to as "least-squares" has only recently been named as such in literature (Golub and Van Loan, 1980). However, it is not a new method and has been extensively studied in the statistical literature for a long time under various names, such as "orthogonal regression," "errors-in-variables," and "measurement errors." In fact, the problem of univariate fitting, $n = 1, d = 1$, was first discussed in 1877 by Adcock (Adcock, 1877), and subsequent contributions were made by Pearson (Pearson, 1901), Koopmans (Koopmans, 1937), and York (York, 1966). The method has been rediscovered several times, often independently, and around 50 years ago, it was extended by Gleser (Gleser, 1981) to multivariate problems of dimension $n > 1$ and $d > 1$.

In more recent times, the least-squares method has gained attention beyond the field of statistics. Golub and Van Loan (Golub and Van Loan, 1980) were the first to study this problem in the field of numerical analysis, and they developed an algorithm based on the singular value decomposition. Staar (Staar, 1982) independently arrived at the same concept through geometrical insight into the properties of the singular value decomposition. Van Huffel and Vandewalle (Van Huffel and Vandewalle, 1988) extended Golub and Van Loan's algorithm to cover all cases where their algorithm fails to produce a solution. They described the properties of these non-generic total least-squares problems and proved that their proposed generalization still satisfies the least-squares criteria if additional constraints are imposed on the solution space. This approach, which appears to be different from the multivariate EIV regression analysis method studied by Gleser, is actually equivalent to it. Gleser's method is based on an eigenvalue-eigenvector analysis, while the least-squares method uses the singular value decomposition, which is more robust numerically in terms of algorithmic implementation. Moreover, the total least-squares algorithm can compute the minimum norm solution whenever the least-squares solution is not unique.

Attention Matrix. The attention matrix is a square matrix that represents correlations between words or tokens in natural language text. It has rows and columns that correspond to each token, and its entries denote the degree of correlation between them. This matrix is employed to determine the significance of each input token in a sequence when generating an output. In an attention mechanism, each input token is assigned a weight or score, which indicates its relevance or importance to the current output being produced. These scores are calculated based on a similarity function that compares the current output state to the input states.

There are several methods that attempt to estimate the heavy entries of the attention matrix by constraining the attention to local neighbors of queries using techniques such as *locality-sensitive hashing* (LSH) (Kitaev et al., 2020; Chen et al., 2021b; Sun et al., 2021) or k-means clustering (Daras et al., 2020). Another approach is to use random feature maps of Gaussian or exponential kernels to approximate the attention matrix (Choromanski et al., 2020). Recently, Chen et al. (Chen et al., 2021a) demonstrated that combining LSH-based and random feature-based methods is more effective at approximating the attention matrix.

The computation of inner product attention (Zandieh et al., 2023; Alman and Song, 2023; Gao et al., 2023a,d; Brand et al., 2023; Deng et al., 2023a; Li et al., 2023; Gao et al., 2023b) is also a crucial task in contemporary machine learning. It is necessary for training large language models (LLMs) such as Transformer (Vaswani et al., 2017), GPT-1 (Radford et al., 2018), BERT (Devlin et al., 2018), GPT-2 (Radford et al., 2019), GPT-3 (Brown et al., 2020), and ChatGPT, which are capable to handle natural language more effectively than conventional algorithms or smaller models. Alman and Song (2023) provides both algorithm and hardness for static attention computation. Brand et al. (2023) provides both algorithm and hardness for dynamically maintaining the attention matrix. Gao et al. (2023b) shows how to compute the attention matrix with differential privacy guarantees. Li et al. (2023) studies the regression for matrix hyperbolic functions. Gao et al. (2023a); Deng et al. (2023a) consider algorithms for softmax attention.

Sketching. Sketching techniques are powerful tools for speeding up machine learning algorithms and optimization. The central idea is to break down a large input matrix into a much smaller matrix while preserving important characteristics. This enables downstream tasks to be performed on top of the smaller matrix. Many previous works have developed sketching algorithms with strong theoretical guarantees. For example, the Johnson-Lindenstrauss lemma in (Johnson and Lindenstrauss, 1984) shows that under a certain high-dimensional space, the projecting points onto the lower-dimensional subspace may preserve the pairwise distances between the points. This property supports the development of faster algorithms for problems, such as approximate nearest neighbor search.

Typically, there are two methods for employing sketching matrices. The first one is called sketch-and-solve, which applies sketching and then subsequently solves the problem in a one-shot manner. The second one is called iterate-and-sketch: sketching can be employed during each iteration of the algorithm to speed up the iteration.

Sketch-and-solve has led to faster algorithms in several domains: for low-rank approximation and linear regression (Nelson and Nguyên, 2013; Clarkson and Woodruff, 2017), sketches are applied to reduce the problem size, so it is much easier to solve the smaller regression problem to get an approximated solution to the original problem; for approximating kernel matrices, as in Laparra et al. (2015); for fast computation of vector and matrix tensors, a long line of works (Avron et al., 2014; Pagh, 2013; Pham and Pagh, 2013; Diao et al., 2018) present a technique, called `TensorSketch` which can compress tensors down to much smaller core tensors, enabling faster algorithms for problems such as tensor regression (Reddy et al., 2022; Diao et al., 2019, 2018; Song et al., 2021a), CP decomposition (Ma and Solomonik, 2021); for column subset selection, sketching the data matrix speeds up column selection with provable approximation guarantees (Sobczyk and Gallopoulos, 2022; Song et al., 2019; Jiang et al., 2020, 2021a). Moreover, it can be used for finding an approximate solution with ℓ_∞ guarantees (Price et al., 2017; Song et al., 2023c), designing an efficient neural network training method (Qin et al., 2023b).

Beyond the classic sketch-and-solve paradigm, sketching has also been adapted to many iterative optimization algorithms. Notable examples include but not limited to non-convex optimization (Zhang, 2022; Alman et al., 2023; Song et al., 2024, 2021b), discrepancy minimization (Song et al., 2022b; Deng et al., 2022), John Ellipsoid computation (Song et al., 2022c), Frank-Wolfe algorithm (Song et al., 2022a; Xu et al., 2021), linear programming (Jiang et al., 2021c; Cohen et al., 2021; Song and Yu, 2021; Liu et al., 2023), reinforcement learning (Xu et al., 2023), dynamic kernel estimation (Qin et al., 2022b), empirical risk minimization (Qin et al., 2023c; Lee et al., 2019), federated learning (Song et al., 2023a), semidefinite programming (Gu and Song, 2022), rational database (Qin et al., 2022a), matrix completion (Gu et al., 2024), matrix sensing (Qin et al., 2023d), submodular maximization (Qin et al., 2023a), trace estimation (Jiang et al., 2021b), and projection maintenance (Song et al., 2023b).

Subspace Embedding. Given a matrix $A \in \mathbb{R}^{n \times d}$, we say a family of random matrices is an (n, d, ϵ, δ) subspace embedding if, for any fixed orthonormal basis U of A, with probability at $1-\delta$, any matrix $S \in \mathbb{R}^{m \times n}$ where m is a function of n, d, ϵ, δ, from this distribution has the property that

$$(1-\epsilon)I \preceq U^\top S^\top S U \preceq (1+\epsilon)I$$

where $A \preceq B$ denotes the matrix $B - A$ is PSD. Alternatively, one could interpret the result as the singular values of SU lie in the range of $[1-\epsilon, 1+\epsilon]$. In the context of least-squares regression, it is usually more convenient to state in the following way: for any vector $x \in \mathbb{R}^d$, $\|SAx\|_2 = (1 \pm \epsilon)\|Ax\|_2$. Subspace embedding is first introduced in Sarlos (2006). Subsequently, structured family of random matrices that can be applied to A efficiently was proposed (Clarkson and Woodruff, 2017; Nelson and Nguyên, 2013; Lu et al., 2013; Meng and Mahoney, 2013). For a more comprehensive review of subspace embedding, we refer readers to the book by Woodruff (Woodruff, 2014).

Roadmap. In Section 2, we introduce the basic mathematical notations. In Section 3, we give an overview of the techniques that we use in this paper. In Section 4, we present our algorithms and main results. In Section 5, we give a conclusion of this paper.

2 PRELIMINARY

In this section, we introduce notations used throughout the paper.

We use $\mathbb{R}$ to denote the set of all real numbers. We use $\mathbb{Z}$ to denote the set of all integers and use $\mathbb{Z}_+$ to denote the set containing all positive integers. For any $n \in \mathbb{Z}_+$, we define $[n] := \{1, 2, 3, \dots, n\}$.

For all $d \in \mathbb{Z}_+$, we use $\mathbb{R}^d$ to denote the set of all vectors with length d and real entries and use $\mathbb{Z}_+^d$ to denote the set containing all vectors with length d and entries of positive integers. For a vector $x \in \mathbb{R}^n$, we use $\|x\|_1$ to denote the ℓ_1 norm, use $\|x\|_2$ to denote the ℓ_2 norm, i.e., $\|x\|_2 = \left(\sum_{i=1}^n x_i^2\right)^{\frac{1}{2}}$, and use $\|x\|_\infty$ to denote the ℓ_∞ norm.

For all $m, n \in \mathbb{Z}_+$, we use $\mathbb{R}^{m \times n}$ to denote the set containing all matrices with m rows, n columns, and real entries. For a matrix $A \in \mathbb{R}^{m \times n}$, we use $\|A\|$ to denote the spectral norm of A, i.e., $\|A\| = \max_{\|x\|_2 = 1} \|Ax\|_2$. We use $A^\top$ to denote the transpose of A. We use $\sigma_{\min}(A)$ to denote the minimum singular value of A, i.e., $\sigma_{\min}(A) = \min_{\|x\|_2=1} \|Ax\|_2$. Accordingly, we use $\sigma_{\max}(A)$ to denote the maximum singular value of A, so $\sigma_{\max}(A) = \|A\|$. Furthermore, we use $A^\dagger$ to denote the pseudo-inverse of A and use A^{-1} to denote the true inverse of A. The true inverse exists if $m = n$ and $\operatorname{rank}(A) = m$. For all $m_1, m_2, n \in \mathbb{Z}_+$, for all the matrices $A \in \mathbb{R}^{m_1 \times n}$ and $B \in \mathbb{R}^{m_2 \times n}$, we use $A \oplus B$ to denote the matrix $\begin{bmatrix} A \\ B \end{bmatrix}$. Correspondingly, for all $m_1, m_2, \dots, m_p, p, n \in \mathbb{Z}_+$, for all the matrices $A_1 \in \mathbb{R}^{m_1 \times n}$, $A_2 \in \mathbb{R}^{m_2 \times n}$, $\cdots$, $A_p \in \mathbb{R}^{m_p \times n}$, we use $\oplus_{i=1}^p A_i$ to denote $A_1 \oplus A_2 \oplus \cdots \oplus A_p$.

We write $x = y \pm \epsilon$ if $x \in [y - \epsilon, y + \epsilon]$. $\mathbf{1}_n$ represents an n-dimensional vector, whose entries are all 1, and $\mathbf{0}_n$ represents a n-dimensional vector, whose entries are all 0. For a symmetric matrix $B \in \mathbb{R}^{n \times n}$, we say B is positive semidefinite (denoted as $B \succeq 0$) if, for any vectors x in $\mathbb{R}^n$, the inequality $x^\top B x \geq 0$ always holds. We also call B a PSD matrix for simplicity.

3 TECHNIQUE OVERVIEW

In Section 3.1, we present the techniques we use to show the properties of a particular case of the odd power algorithm. In Section 3.2, we not only show the techniques of showing the correctness and the runtime of a particular case of the even power algorithm but also introduce a way to bound the forward error of the PSD matrix. In Section 3.3, we offer the methods to generalize the particular case to all the even power cases. In Section 3.4, we elucidate the methods of generalizing the particular case to all the odd power cases. In Section 3.5, we introduce the techniques which are used for analyzing the exponential regression.

To give an algorithm for Theorem 1.4 and 1.5, we need to analyze the simpler case with only 3 and 4 matrices involved:

$$\min_{x \in \mathbb{R}^d} \|AA^\top Ax - b\|_2 \tag{3}$$

and

$$\min_{x \in \mathbb{R}^d} \|A^\top AA^\top Ax - b\|_2. \tag{4}$$

Suppose we are provided with algorithms for these smaller problems, we can complete the picture for Theorem 1.4 and 1.5 via induction. It remains to show how to solve the simpler cases.

3.1 A Particular Case for Odd Power Algorithm

To solve Eq. (3), our approach can be briefly described as follows: we first solve the regression $\min_{y \in \mathbb{R}^d} \|Ay - b\|_2$ then solve $\|A^\top Ax - y\|_2$. To understand the error, suppose x' is the final approximate solution to $\min_{x \in \mathbb{R}^d} \|A^\top Ax - y'\|_2$, we can decompose the error as follows:

$$\begin{aligned} \|AA^\top Ax' - b\|_2 &= \|A(A^\top Ax' - y') + Ay' - b\|_2 \\ &\leq \|A\| \|A^\top Ax' - y'\|_2 + \|Ay' - b\|_2. \end{aligned}$$

For the second term, it can be bounded as long as we utilize a fast solver with relative error dependence in terms of the optimal cost, the value of Eq. (3). For the first term, we deploy a fast, high accuracy solver for PSD regression, with the guarantee that

$$\|A^\top Ax' - y'\|_2 \leq \epsilon \cdot \|y'\|_2,$$

so it remains to get a good handle on $\|y'\|_2$. Note that it is the solution to the regression $\|Ay' - b\|_2$, and we can convert the error on the regression cost to the error on the quality of y'. More specifically, we have

$$\|y' - y_*\|_2 \leq O(\sqrt{\epsilon}) \cdot \frac{1}{\sigma_{\min}(A)} \cdot \|Ay_* - b\|_2$$

for y_* being the optimal solution to the regression. Finally, note that $y_* = A^\dagger b$, so we can simply bound its norm as $\|y_*\|_2 \leq \frac{1}{\sigma_{\min}(A)} \cdot \|b\|_2$. Put it all together, we have

$$\begin{aligned} \|y'\|_2 &\leq \|y' - y_*\|_2 + \|y_*\|_2 \\ &\leq \frac{1}{\sigma_{\min}(A)} (\sqrt{\epsilon} \cdot \|Ay_* - b\|_2 + \|b\|_2). \end{aligned}$$

By setting ϵ properly, we can conclude

$$\|AA^\top Ax' - b\|_2 \leq (1 + \epsilon) \cdot \|AA^\top Ax_* - b\|_2 + \epsilon \cdot \|b\|_2.$$

For the running time, since we would need to set ϵ perhaps polynomially small to account for the error induced by backward-to-forward conversion and minimum singular values of A, fast solvers with runtime depends inverse polynomially on ϵ is infeasible. We resort to solvers with runtime depends on $\log(1/\epsilon)$ (Avron et al., 2010; Brand et al., 2021), which in turn provides the following final runtime bound:

$$O((nd + d^3) \log(\kappa/\epsilon) \log^2(n/\delta)),$$

where κ is the condition number of A and δ is the failure probability of the algorithm.

3.2 A Particular Case for Even Power Algorithm

To solve Eq. (4), our algorithm is similar, except that we need to perform two rounds of regressions on PSD matrices: $\min_y \|A^\top Ay - b\|_2$ and $\min_x \|A^\top Ax - y\|_2$. It is more complicated to analyze this than to analyze the prior algorithm because, to the best of our knowledge, there was no past literature proving the forward error for PSD matrices.

We thus focus on the techniques for deriving the forward error for PSD matrices. Let us consider the regression problem

$$\min_{x \in \mathbb{R}^d} \|A^\top Ax - b\|_2, \tag{5}$$

and we will use x_* to denote the exact solution to the above problem and x' to denote an approximate solution, i.e.,

$$\|A^\top Ax' - b\|_2 \leq \epsilon \cdot \|b\|_2.$$

We can bound the gap $\|x' - x^*\|_2$ as follows:

$$\|x' - x_*\|_2 \leq \|A^\top A(x' - x_*)\|_2 \cdot \|(A^\top A)^\dagger\|.$$

The second term is simply $\sigma_{\min}^{-2}(A)$, and since $A^\top A$ is full rank, $A^\top Ax_* = b$, so the first term is $\epsilon \cdot \|b\|_2$. This gives us a final bound of

$$\|x' - x_*\|_2 \leq \epsilon \cdot \frac{1}{\sigma_{\min}^2(A)} \cdot \|b\|_2,$$

coupling with the analysis of error and runtime in the preceding section, we obtain the desired results.

3.3 General Case for Even Power Algorithm

Given an algorithm and analysis for the base case, we can now generalize to any even powers. We start by setting up the induction hypothesis, for which we assume for all $i \in [k]$, we have

1. $\|(A^\top A)^i b_i - b_0\|_2 \leq \epsilon_i \|b_0\|_2$,
2. $\|b_i\|_2 \leq 2\sigma_{\min}^{-2i}(A)\|b_0\|_2$,
3. $\epsilon_i \leq 0.5\epsilon_{i-1}$,
4. The running time is $C \cdot ((nd + d^3) \cdot k \cdot \log(\kappa(A)/\epsilon_k) \cdot \log(1/\delta_k))$,
5. The failure probability is $\delta_1 + \delta_2 + \cdots + \delta_k$.

We want to show that these five statements also hold for $i = k + 1$. To prove the first statement, we need to analyze $\|(A^\top A)^{k+1} b_i - b_0\|_2$:

$$\begin{aligned} &\|(A^\top A)^{k+1} b_{k+1} - b_0\|_2 \\ \leq\ &\|(A^\top A)^k\| \cdot \|A^\top Ab_{k+1} - b_k\|_2 + \|(A^\top A)^k b_k - b_0\|_2. \end{aligned} \tag{6}$$

The three terms can be bounded as follows:

- $\|(A^\top A)^k b_k - b_0\|_2$ can be bounded by $\epsilon_k\|b_0\|_2$ based on induction hypothesis for $i = k$.
- $\|(A^\top A)^k\|$ can be bounded by $\sigma_{\max}^{2k}(A)$ based on Fact A.3.
- $\|A^\top Ab_{k+1} - b_k\|_2$ can be bounded by $0.1\epsilon_{k+1}\kappa^{-2k(A)}\|b_k\|_2$ based on our two matrices version of PSD regression (see Lemma E.2).

For the second statement, the goal is to bound $\|b_{k+1}\|_2$. By using the property of the spectral norm, we can get

$$\|b_{k+1}\|_2 \leq \|((A^\top A)^{k+1})^{-1}\| \cdot \|(A^\top A)^{k+1} b_{k+1}\|_2. \tag{7}$$

These two terms can be bounded similarly, we include the reasoning here:

- $\|((A^\top A)^{k+1})^{-1}\|$ can be bounded by $2\sigma_{\min}^{-2(k+1)}(A)$ based on Fact A.3.
- $\|(A^\top A)^{k+1} b_{k+1}\|_2$ can be bounded $2\|b_0\|_2$ based on the first statement and triangle inequality.

The remaining statements are rather straightforward: the third one requires us to choose ϵ_{k+1} in a particular fashion, and the running time can be computed via plugging ϵ_{k+1}. The failure probability follows from a union bound, and conditioning all prior algorithms succeed.

3.4 General Case for Odd Power Algorithm

For general odd powers in the form of $\min_{x \in \mathbb{R}^d} \|A(A^\top A)^j x - b\|_2$, our argument is similar to preceding section, with a minor modification that we start by solving a linear regression on A instead of a PSD regression on $A^\top A$. Similar to Section 3.1, we will have to convert error on the cost to error on the solution, and the remaining are similar to that of Section 3.3.

3.5 Attention Kernel Regression

We start by recalling the definition of attention kernel regression: given an input matrix $A \in \mathbb{R}^{n\times d}$ and a response $b \in \mathbb{R}^n$, our task is in two-folds: 1). compute or approximate $\exp(AA^\top)$; 2). solve the regression on top of the (approximated) kernel matrix. A simple, exact algorithm would be computing the Gram matrix $AA^\top$, applying entrywise exponential activation and solving the regression. This would take $O(n^2 d)$ time, inefficient if d is large.

The key is to identify the kernel structure of $\exp(AA^\top)$, where the (i,j)-th entry is $\exp(\langle a_i, a_j\rangle)$. We can apply sketches for polynomial kernels to efficiently approximate this kernel matrix, and then solve the regression with high precision. Although the sketch dimension is super linear in n and the result naturally follows from Song et al. (2021a), we also design a novel algorithm if the data is highly structured and the sketching dimension is sublinear in n. We illustrate how to compute a quick preconditioner such kernel and solve the corresponding regression. For simplicity, we let $G := \exp(AA^\top)$. The goal is to find $x' \in \mathbb{R}^n$ such that

$$\|Gx' - y\|_2 \leq \epsilon \|y\|_2,$$

for $\epsilon \in (0,1)$ be an accuracy parameter. The idea is to compute a sketch $W_g(A) \in \mathbb{R}^{m\times n}$ such that $W_g(A)^\top W_g(A)$ is an ϵ-spectral approximation to G. Since m is small, we can apply another shallow sketches to the matrix $W_g(A)^\top$ to improve the dimension from n to s where $s = o(n)$, then compute an SVD of the sketched matrix $SW_g(A)^\top = U\Sigma V^\top$. By choosing the matrix $R = U\Sigma^{-2}$ as a preconditioner, we can then proceed to implement the preconditioned gradient descent, and obtains an improved running time and convergence rate. We briefly explain why the matrix R is a good preconditioner, note that

$$\begin{aligned} SW_g(A)^\top W_g(A) S^\top R &= U\Sigma V^\top V \Sigma U^\top U \Sigma^{-2} \\ &= U\Sigma^2\Sigma^{-2} \\ &= U, \end{aligned}$$

which has condition number 1. As S produces an ϵ-subspace embedding, the condition number of the matrix $W_g(A)^\top W_g(A) S^\top R$ is also good. We can thus perform gradient descent on this matrix, yields a good convergence and running time. Put it together, we obtain a running time of

$$\begin{aligned} &\epsilon^{-2} n\beta \cdot \mathrm{poly}(\log(nd/\epsilon\delta)) \cdot \log(\kappa/\epsilon) \\ &+ (nd + (\epsilon^{-2}\beta)^\omega) \cdot \log(nd/\epsilon\delta). \end{aligned}$$

4 MAIN RESULT

We now state our main algorithms and results. For the even/odd power regression, note that directly computing the product $(A^\top A)^j$ requires $O(j \cdot nd^2)$ time. The squaring trick reduces this to $O(\log j \cdot nd^2)$ time. While the dependence on n is nearly optimal, the dependence on d makes these impractical for large-scale problems. Our new algorithms achieve a running time of $O(j \cdot nd)$ for $n \gg d$. So when j is smaller than $d \cdot \log j$, this is a significant improvement.

Algorithm 1 Algorithm for solving even power regression $\min_{x\in\mathbb{R}^d} \|(A^\top A)^j x - b\|_2$.

```
procedure EvenPowers($A \in \mathbb{R}^{n\times d}, b \in \mathbb{R}^d, n \in \mathbb{Z}_+, d \in \mathbb{Z}_+, j \in \mathbb{Z}_+, \epsilon_{\mathrm{final}} \in (0,1), \delta_{\mathrm{final}} \in (0,1)$)
    $b_0 \leftarrow b$
    for $k = 1 \to j$ do
        $\delta_k \leftarrow \delta_{\mathrm{final}}/k$
        $\epsilon_k \leftarrow \epsilon_{\mathrm{final}} \cdot 0.5^{j-k}$
        $b_k \leftarrow$ FastPSDRegression($A \in \mathbb{R}^{n\times d}, b_{k-1} \in \mathbb{R}^d, n, d, \epsilon_k/\kappa(A)^{2k}, \delta_k$)
    end for
    $x' \leftarrow b_j$
    return $x'$
end procedure
```

Algorithm 2 Algorithm for solving odd power regression $\min_{x\in\mathbb{R}^d} \|A(A^\top A)^j x - b\|_2$.

```
procedure OddPowers($A \in \mathbb{R}^{n\times d}, b \in \mathbb{R}^n, n \in \mathbb{Z}_+, d \in \mathbb{Z}_+, j \in \mathbb{Z}_+, \epsilon_{\mathrm{final}} \in (0,1), \delta_{\mathrm{final}} \in (0,1)$)
    $\epsilon_1 \leftarrow 0.1\epsilon_{\mathrm{final}}$
    $\delta_1 \leftarrow \delta_{\mathrm{final}}/2$
    $b_1 \leftarrow$ FastLinearRegression($A \in \mathbb{R}^{n\times d}, b \in \mathbb{R}^n, n, d, \epsilon_1, \delta_1$)    ▷ $b_1 \in \mathbb{R}^d$
    $\epsilon_{\mathrm{even}} \leftarrow \epsilon_{\mathrm{final}}/\kappa(A)$
    $\delta_{\mathrm{even}} \leftarrow \delta_{\mathrm{final}}/2$
    $x' \leftarrow$ EvenPowers($A \in \mathbb{R}^{n\times d}, b_1 \in \mathbb{R}^d, n, d, j, \epsilon_{\mathrm{even}}, \delta_{\mathrm{even}}$)
    return $x'$
end procedure
```

Theorem 4.1 (Main Result for Matrix Exponential Proxy and Even/Odd Power Regression)**.** *Let $A \in \mathbb{R}^{n\times d}$ be a matrix and $b \in \mathbb{R}^d$ be a vector. Let $\epsilon_{\mathrm{final}} \in (0, 0.1)$ denote the accuracy parameter. Let $\delta_{\mathrm{final}} \in (0, 0.1)$ denote the failure probability. Let κ denote the condition number of A. Then, there exist algorithms (Algorithm 1 and 2) that run in time*

$$O((nd + d^3) \cdot j \cdot \log(\kappa/\epsilon_{\mathrm{final}}) \cdot \log^2(jn/\delta_{\mathrm{final}}))$$

and outputs a vector $x' \in \mathbb{R}^d$ such that

$$\|(A^\top A)^j x' - b\|_2 \leq \epsilon_{\mathrm{final}} \|b\|_2 \text{ (Algorithm 1)}$$
$$\|A(A^\top A)^j x' - b\|_2 \leq \epsilon_{\mathrm{final}} \|b\|_2 \text{ (Algorithm 2).}$$

holds with probability at least $1 - \delta_{\mathrm{final}}$.

Algorithm 3 Algorithm for solving attention kernel regression $\min_{x\in\mathbb{R}^n} \|\exp(AA^\top)x - b\|_2$.

```
procedure PreconditionedGradientDescent($A \in \mathbb{R}^{n\times d}, b \in \mathbb{R}^n, \beta \in [1,n], \epsilon_{\mathrm{final}} \in (0,1), \delta_{\mathrm{final}} \in (0,1)$)    ▷ Theorem H.7
    $\epsilon_0 \leftarrow 0.01$
    $m \leftarrow O(\beta \log^2(nd/\epsilon_{\mathrm{final}}\delta_{\mathrm{final}}) \log(n/\delta_{\mathrm{final}})/\epsilon_{\mathrm{final}}^2)$
    if $m \geq n$ then
        $x \leftarrow$ Theorem 6.6 of Song et al. (2021a)
        return $x$
    end if
    $s \leftarrow \Omega(m \log(mn/\epsilon_0\delta_{\mathrm{final}}) \log(n/\delta_{\mathrm{final}})/\epsilon_0^2)$
    Let $W_g(X) \in \mathbb{R}^{m\times n}$ be the approximate Gaussian kernel in Theorem H.6
    Let $S \in \mathbb{R}^{s\times n}$ be an $\mathsf{SRHT}$ matrix
    Compute the SVD of $SW_g(X)^\top = U\Sigma V^\top$
    $R \leftarrow U\Sigma^{-2} \in \mathbb{R}^{s\times m}$
    $z_0 \leftarrow \mathbf{0}_m \in \mathbb{R}^m$
    while $\|W_g(X)^\top W_g(X) S^\top R z_t - y\|_2 \geq \epsilon_{\mathrm{final}}$ do
        $z_{t+1} \leftarrow z_t - (R^\top S W_g(X)^\top W_g(X) S^\top R)^\top \cdot$
        $(R^\top S W_g(X)^\top W_g(X) S^\top R z_t - R^\top S y)$
    end while
    return $S^\top R z_t$
end procedure
```

We defer the details of the proofs to Appendix G.3 and G.

Theorem 4.2 (Main Result for Attention Kernel Regression). *Let* $A \in \mathbb{R}^{n\times d}$ *and* $b \in \mathbb{R}^n$, $\epsilon_{\mathrm{final}}, \delta_{\mathrm{final}} \in (0,1)$ *be accuracy parameter and failure probability, respectively. Suppose for all* $i \in [n]$, $\|a_i\|_2 \leq 1$, *where* a_i *is the* i*-th row of matrix* A. *Further, let* β *be an upper bound on the rank of the following sequence of matrices:*

$$\{A, A^{\otimes 2}, \ldots, A^{\otimes q}\}$$

for $q = O(\log(n/\epsilon_{\mathrm{final}}))$ *and* $A^{\otimes l} \in \mathbb{R}^{n\times d^l}$ *is the matrix formed by taking each row of* A *and computing the* l*-fold self-tensoring. Then, there exists a randomized algorithm (Algorithm 3) that succeeds with probability at least* $1 - \delta_{\mathrm{final}}$, *computes a vector* $x' \in \mathbb{R}^n$ *such that*

$$\|\exp(AA^\top)x' - b\|_2 \leq \epsilon_{\mathrm{final}} \cdot \|b\|_2,$$

moreover, let $m = O(\epsilon_{\mathrm{final}}^{-2}\beta \log^3(\kappa nd/(\epsilon_{\mathrm{final}}\delta_{\mathrm{final}})))$, *then the vector* x' *can be computed in time*

$$O(mn + \epsilon_{\mathrm{final}}^{-2} nd + m^3).$$

For the complete algorithm and detailed analysis, we refer readers to Appendix H.

5 Conclusion

Large language models have demonstrated remarkable performance in various tasks. One significant aspect, from a computational standpoint, is the computation of the attention matrix. Earlier research had thoroughly examined the feasibility and limitations of approximating the attention matrix. In this study, we introduce several new problems concerning computing or approximating proxies for the attention matrix, via matrix exponential or entrywise exponential for the Gram matrix of data. We provide fast algorithms for these problems, based on sketching and preconditioning.

We note that while our algorithm for regression against of product of matrices runs in nearly linear time, the runtime dependence on the number of matrices j is still linear. In contrast, the squaring method only depends logarithmically on j. Unfortunately, our algorithm has fundamental limits on improving dependence on j due to the alternating solve nature. It will be interesting to devise an algorithm that both runs in nearly linear time and has better dependence on j.

Acknowledgements

Lichen Zhang was supported in part by NSF CCF-1955217 and NSF DMS-2022448.

References

Dimitris Achlioptas and Frank McSherry. On spectral learning of mixtures of distributions. In *Learning Theory: 18th Annual Conference on Learning Theory, COLT 2005, Bertinoro, Italy, June 27-30, 2005. Proceedings 18*, pages 458–469. Springer, 2005.

Dimitris Achlioptas, Amos Fiat, Anna R Karlin, and Frank McSherry. Web search via hub synthesis. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 500–509. IEEE, 2001.

R. J. Adcock. Note on the method of least squares. *Analyst*, 4:183–184, 1877.

Thomas D Ahle, Michael Kapralov, Jakob BT Knudsen, Rasmus Pagh, Ameya Velingker, David P Woodruff, and Amir Zandieh. Oblivious sketching of high-degree polynomial kernels. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 141–160. SIAM, 2020.

Nir Ailon and Bernard Chazelle. Approximate nearest neighbors and the fast johnson-lindenstrauss transform. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 557–563, 2006.

Josh Alman and Zhao Song. Fast attention requires bounded entries. *Advances in Neural Information Processing Systems*, 2023.

Josh Alman, Jiehao Liang, Zhao Song, Ruizhe Zhang, and Danyang Zhuo. Bypass exponential time preprocessing: Fast neural network training via weight-data correlation preprocessing. In *Advances in Neural Information Systems*, 2023.

Haim Avron, Petar Maymounkov, and Sivan Toledo. Blendenpik: Supercharging lapack's least-squares solver. *SIAM J. Sci. Comput.*, 2010.

Haim Avron, Huy Nguyen, and David Woodruff. Subspace embeddings for the polynomial kernel. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2258–2266, 2014.

Yossi Azar, Amos Fiat, Anna Karlin, Frank McSherry, and Jared Saia. Spectral analysis of data. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 619–626, 2001.

Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Near-optimal column-based matrix reconstruction. *SIAM Journal on Computing*, 43(2): 687–717, 2014.

Lennart Bramlage and Aurelio Cortese. Generalized attention-weighted reinforcement learning. *Neural Networks*, 145:10–21, 2022.

Jan van den Brand, Binghui Peng, Zhao Song, and Omri Weinstein. Training (overparametrized) neural networks in near-linear time. In *ITCS*, 2021.

Jan van den Brand, Zhao Song, and Tianyi Zhou. Algorithm and hardness for dynamic attention maintenance in large language models. *arXiv preprint arXiv:2304.02207*, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention. *Advances in Neural Information Processing Systems*, 34:17413–17426, 2021a.

Beidi Chen, Zichang Liu, Binghui Peng, Zhaozhuo Xu, Jonathan Lingjie Li, Tri Dao, Zhao Song, Anshumali Shrivastava, and Christopher Re. Mongoose: A learnable lsh framework for efficient neural network training. In *International Conference on Learning Representations*, 2021b.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. Attention-based models for speech recognition. *Advances in neural information processing systems*, 28, 2015.

Kenneth L Clarkson and David P Woodruff. Low-rank approximation and regression in input sparsity time. *Journal of the ACM (JACM)*, 63(6):1–45, 2017.

Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. *Journal of the ACM (JACM)*, 68(1):1–39, 2021.

Giannis Daras, Nikita Kitaev, Augustus Odena, and Alexandros G Dimakis. Smyrf-efficient attention using asymmetric clustering. *Advances in Neural Information Processing Systems*, 33:6476–6489, 2020.

Yichuan Deng, Zhao Song, and Omri Weinstein. Discrepancy minimization in input-sparsity time. *arXiv preprint arXiv:2210.12468*, 2022.

Yichuan Deng, Zhihang Li, and Zhao Song. Attention scheme inspired softmax regression. *arXiv preprint arXiv:2304.10411*, 2023a.

Yichuan Deng, Sridhar Mahadevan, and Zhao Song. Randomized and deterministic attention sparsification algorithms for over-parameterized feature dimension. *arXiv preprint arXiv:2304.04397*, 2023b.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

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.

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.

Petros Drineas, Iordanis Kerenidis, and Prabhakar Raghavan. Competitive recommendation systems. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 82–90, 2002.

Petros Drineas, Alan Frieze, Ravi Kannan, Santosh Vempala, and Vishwanathan Vinay. Clustering large graphs via the singular value decomposition. *Machine learning*, 56:9–33, 2004.

Yeqi Gao, Lianke Qin, Zhao Song, and Yitan Wang. A sublinear adversarial training algorithm. *arXiv preprint arXiv:2208.05395*, 2022.

Yeqi Gao, Sridhar Mahadevan, and Zhao Song. An over-parameterized exponential regression. *arXiv preprint arXiv:2303.16504*, 2023a.

Yeqi Gao, Zhao Song, and Xin Yang. Differentially private attention computation. *arXiv preprint arXiv:2305.04701*, 2023b.

Yeqi Gao, Zhao Song, Xin Yang, and Ruizhe Zhang. Fast quantum algorithm for attention computation. *arXiv preprint arXiv:2307.08045*, 2023c.

Yeqi Gao, Zhao Song, and Junze Yin. An iterative algorithm for rescaled hyperbolic functions regression. *arXiv preprint arXiv:2305.00660*, 2023d.

Leon Jay Gleser. Estimation in a multivariate" errors in variables" regression model: large sample results. *The Annals of Statistics*, pages 24–44, 1981.

Gene H Golub and Charles F Van Loan. An analysis of the total least squares problem. *SIAM journal on numerical analysis*, 17(6):883–893, 1980.

Yuzhou Gu and Zhao Song. A faster small treewidth sdp solver. *arXiv preprint arXiv:2211.06033*, 2022.

Yuzhou Gu, Zhao Song, Junze Yin, and Lichen Zhang. Low rank matrix completion via robust alternating minimization in nearly linear time. In *International Conference on Learning Representations*, 2024.

Meng-Hao Guo, Tian-Xing Xu, Jiang-Jiang Liu, Zheng-Ning Liu, Peng-Tao Jiang, Tai-Jiang Mu, Song-Hai Zhang, Ralph R Martin, Ming-Ming Cheng, and Shi-Min Hu. Attention mechanisms in computer vision: A survey. *Computational Visual Media*, 8(3): 331–368, 2022.

Shuli Jiang, Dongyu Li, Irene Mengze Li, Arvind V Mahankali, and David Woodruff. An efficient protocol for distributed column subset selection in the entrywise ℓ_p norm, 2020.

Shuli Jiang, Dennis Li, Irene Mengze Li, Arvind V Mahankali, and David Woodruff. Streaming and distributed algorithms for robust column subset selection. In *International Conference on Machine Learning*, pages 4971–4981. PMLR, 2021a.

Shuli Jiang, Hai Pham, David Woodruff, and Richard Zhang. Optimal sketching for trace estimation. *Advances in Neural Information Processing Systems*, 34: 23741–23753, 2021b.

Shunhua Jiang, Zhao Song, Omri Weinstein, and Hengjie Zhang. A faster algorithm for solving general lps. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 823–832, 2021c.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

Daniel M Kane and Jelani Nelson. Sparser johnson-lindenstrauss transforms. *Journal of the ACM (JACM)*, 61(1):1–23, 2014.

Ravindran Kannan, Hadi Salmasian, and Santosh Vempala. The spectral method for general mixture models. *SIAM Journal on Computing*, 38(3):1141–1156, 2008.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Philip Klein, Serge Plotkin, Clifford Stein, and Eva Tardos. Faster approximation algorithms for the unit capacity concurrent flow problem with applications to routing and finding sparse cuts. *SIAM Journal on Computing*, 23(3):466–487, 1994.

Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46 (5):604–632, 1999.

Tjalling Charles Koopmans. *Linear regression analysis of economic time series*, volume 20. De erven F. Bohn nv, 1937.

Valero Laparra, Diego Marcos Gonzalez, Devis Tuia, and Gustau Camps-Valls. Large-scale random features for kernel regression. In *2015 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 17–20. IEEE, 2015.

Yin Tat Lee, Zhao Song, and Qiuyi Zhang. Solving empirical risk minimization in the current matrix multiplication time. In *Conference on Learning Theory*, pages 2140–2157. PMLR, 2019.

Zhihang Li, Zhao Song, and Tianyi Zhou. Solving regularized exp, cosh and sinh regression problems. *arXiv preprint arXiv:2303.15725*, 2023.

S. Cliff Liu, Zhao Song, Hengjie Zhang, Lichen Zhang, and Tianyi Zhou. Space-efficient interior point method, with applications to linear programming and maximum weight bipartite matching. In *International Colloquium on Automata, Languages and Programming (ICALP)*, pages 88:1–88:14, 2023.

Xiaoyun Liu, Daniel Esser, Brandon Wagstaff, Anna Zavodni, Naomi Matsuura, Jonathan Kelly, and Eric Diller. Capsule robot pose and mechanism state detection in ultrasound using attention-based hierarchical deep learning. *Scientific Reports*, 12(1):21130, 2022.

Yichao Lu, Paramveer Dhillon, Dean P Foster, and Lyle Ungar. Faster ridge regression via the subsampled randomized hadamard transform. *Advances in neural information processing systems*, 26, 2013.

Linjian Ma and Edgar Solomonik. Fast and accurate randomized algorithms for low-rank tensor decompositions. *Advances in Neural Information Processing Systems*, 34:24299–24312, 2021.

Frank McSherry. Spectral partitioning of random graphs. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 529–537. IEEE, 2001.

Xiangrui Meng and Michael 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, 2013.

Jelani Nelson and Huy L Nguyên. Osnap: Faster numerical linear algebra algorithms via sparser subspace embeddings. In *2013 ieee 54th annual symposium on foundations of computer science*, pages 117–126. IEEE, 2013.

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

Christos H Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 159–168, 1998.

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

Ninh Pham and Rasmus 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*, pages 239–247, 2013.

Eric Price, Zhao Song, and David P. Woodruff. Fast Regression with an ℓ_∞ Guarantee. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*, Leibniz International Proceedings in Informatics (LIPIcs), pages 59:1–59:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.

Lianke Qin, Rajesh Jayaram, Elaine Shi, Zhao Song, Danyang Zhuo, and Shumo Chu. Adore: Differentially oblivious relational database operators. *arXiv preprint arXiv:2212.05176*, 2022a.

Lianke Qin, Aravind Reddy, Zhao Song, Zhaozhuo Xu, and Danyang Zhuo. Adaptive and dynamic multi-resolution hashing for pairwise summations. In *2022 IEEE International Conference on Big Data (Big Data)*, pages 115–120. IEEE, 2022b.

Lianke Qin, Zhao Song, and Yitan Wang. Fast submodular function maximization. *arXiv preprint arXiv:2305.08367*, 2023a.

Lianke Qin, Zhao Song, and Yuanyuan Yang. Efficient sgd neural network training via sublinear activated neuron identification. *arXiv preprint arXiv:2307.06565*, 2023b.

Lianke Qin, Zhao Song, Lichen Zhang, and Danyang Zhuo. An online and unified algorithm for projection matrix vector multiplication with application to empirical risk minimization. In *AISTATS*, 2023c.

Lianke Qin, Zhao Song, and Ruizhe Zhang. A general algorithm for solving rank-one matrix sensing. *arXiv preprint arXiv:2303.12298*, 2023d.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1 (8):9, 2019.

Aravind Reddy, Zhao Song, and Lichen Zhang. Dynamic tensor product regression. In *NeurIPS*, 2022.

Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06)*, pages 143–152. IEEE, 2006.

Aleksandros Sobczyk and Efstratios Gallopoulos. pylspack: Parallel algorithms and data structures for sketching, column subset selection, regression, and leverage scores. *ACM Transactions on Mathematical Software*, 48(4):1–27, 2022.

Ghada Sokar, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Self-attention meta-learner for continual learning. *arXiv preprint arXiv:2101.12136*, 2021.

Zhao Song and Zheng Yu. Oblivious sketching-based central path method for linear programming. In *International Conference on Machine Learning*, pages 9835–9847. PMLR, 2021.

Zhao Song, David Woodruff, and Peilin Zhong. Towards a zero-one law for column subset selection. *Advances in Neural Information Processing Systems*, 32, 2019.

Zhao Song, David Woodruff, Zheng Yu, and Lichen Zhang. Fast sketching of polynomial kernels of polynomial degree. In *International Conference on Machine Learning*, pages 9812–9823. PMLR, 2021a.

Zhao Song, Shuo Yang, and Ruizhe Zhang. Does preprocessing help training over-parameterized neural networks? *Advances in Neural Information Processing Systems*, 34:22890–22904, 2021b.

Zhao Song, Zhaozhuo Xu, Yuanyuan Yang, and Lichen Zhang. Accelerating frank-wolfe algorithm using low-dimensional and adaptive data structures. *arXiv preprint arXiv:2207.09002*, 2022a.

Zhao Song, Zhaozhuo Xu, and Lichen Zhang. Speeding up sparsification using inner product search data structures. *arXiv preprint arXiv:2204.03209*, 2022b.

Zhao Song, Xin Yang, Yuanyuan Yang, and Tianyi Zhou. Faster algorithm for structured john ellipsoid computation. *arXiv preprint arXiv:2211.14407*, 2022c.

Zhao Song, Yitan Wang, Zheng Yu, and Lichen Zhang. Sketching for first order method: Efficient algorithm for low-bandwidth channel and vulnerability. In *ICML*, 2023a.

Zhao Song, Xin Yang, Yuanyuan Yang, and Lichen Zhang. Sketching meets differential privacy: fast algorithm for dynamic kronecker projection maintenance. In *International Conference on Machine Learning (ICML)*, pages 32418–32462. PMLR, 2023b.

Zhao Song, Mingquan Ye, Junze Yin, and Lichen Zhang. A nearly-optimal bound for fast regression with ℓ_∞ guarantee. In *International Conference on Machine Learning (ICML)*, pages 32463–32482. PMLR, 2023c.

Zhao Song, Lichen Zhang, and Ruizhe Zhang. Training multi-layer over-parametrized neural network in subquadratic time. In *Innovations in Theoretical Computer Science (ITCS)*, 2024.

J Staar. Concepts for reliable modelling of linear systems with application to on-line identification of multivariable state space descriptions. *Doct. Th., Kat. Univ. Leuven, Belgium*, 1982.

Zhiqing Sun, Yiming Yang, and Shinjae Yoo. Sparse attention with learning to hash. In *International Conference on Learning Representations*, 2021.

Sabine Van Huffel and Joos Vandewalle. Analysis and solution of the nongeneric total least squares problem. *SIAM journal on matrix analysis and applications*, 9 (3):360–372, 1988.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Joseph Wang and David Eppstein. Fast approximation of centrality. *Graph algorithms and applications*, 5(5): 39, 2006.

Yuanchao Wang, Wenji Du, Chenghao Cai, and Yanyan Xu. Explaining the attention mechanism of end-to-end speech recognition using decision trees. *arXiv preprint arXiv:2110.03879*, 2021.

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

Franco Woolfe, Edo Liberty, Vladimir Rokhlin, and Mark Tygert. A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25(3):335–366, 2008.

Zhaozhuo Xu, Zhao Song, and Anshumali Shrivastava. Breaking the linear iteration cost barrier for some well-known conditional gradient methods using maxip data-structures. *Advances in Neural Information Processing Systems*, 34:5576–5589, 2021.

Zhaozhuo Xu, Zhao Song, and Anshumali Shrivastava. A tale of two efficient value iteration algorithms for solving linear mdps with large action space. In Francisco Ruiz, Jennifer Dy, and Jan-Willem van de

Meent, editors, *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, Proceedings of Machine Learning Research. PMLR, 2023.

Derek York. Least-squares fitting of a straight line. *Canadian Journal of Physics*, 44(5):1079–1086, 1966.

Amir Zandieh, Insu Han, Majid Daliri, and Amin Karbasi. Kdeformer: Accelerating transformers via kernel density estimation. In *International Conference on Machine Learning*, 2023.

Lichen Zhang. Speeding up optimizations via data structures: Faster search, sample and maintenance. Master's thesis, Carnegie Mellon University, 2022.

Mengqi Zhang, Shu Wu, Meng Gao, Xin Jiang, Ke Xu, and Liang Wang. Personalized graph neural networks with attention mechanism for session-aware recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3946–3957, 2020.

Checklist

1. For all models and algorithms presented, check if you include:
    (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
    (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
    (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Not Applicable]

2. For any theoretical claim, check if you include:
    (a) Statements of the full set of assumptions of all theoretical results. [Yes]
    (b) Complete proofs of all theoretical results. [Yes]
    (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
    (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Not Applicable]
    (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Not Applicable]
    (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Not Applicable]
    (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Not Applicable]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
    (a) Citations of the creator If your work uses existing assets. [Not Applicable]
    (b) The license information of the assets, if applicable. [Not Applicable]
    (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
    (d) Information about consent from data providers/curators. [Not Applicable]
    (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
    (a) The full text of instructions given to participants and screenshots. [Not Applicable]
    (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
    (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Appendix

Roadmap. In Section A, we introduce the notations, basic definitions, and facts that we use. In Section B, we present the background of the exp of the inner product kernel. In Section C, we discuss the background of standard sketching. In Section D, we introduce the background of the high precision sketching. In Section E, we show the fast linear regression algorithm (see Algorithm 4), solving the regression problem containing one matrix, and analyze its properties, including its correctness and running time. In Section F, we offer a new algorithm (see Algorithm 6), which solves the regression problem containing three matrices, and analyzes its correctness and running time. In Section F.2, we propose an algorithm (see Algorithm 7), which solves the regression problem containing four matrices, and analyzes its properties. In Section G, we summarize and utilize the patterns of the previous sections to analyze the odd and even power regressions. In Section H, we analyze the attention kernel.

A PRELIMINARY

In Section A.1, we introduce the basic notations. In Section A.2, we introduce some basic definitions and useful facts. In Section A.3, we present some background about attention computation.

A.1 Notations

We use $\mathbb{R}$ to denote the set of all real numbers. We use $\mathbb{Z}$ to denote the set of all integers and use $\mathbb{Z}_+$ to denote the set containing all positive integers. For any $n \in \mathbb{Z}_+$, we define $[n] := \{1, 2, 3, \ldots, n\}$.

For all $d \in \mathbb{Z}_+$, we use $\mathbb{R}^d$ to denote the set of all vectors with length d and real entries and use $\mathbb{Z}_+^d$ to denote the set containing all vectors with length d and entries of positive integers. For a vector $x \in \mathbb{R}^n$, we use $\|x\|_1$ to denote the ℓ_1 norm, use $\|x\|_2$ to denote the ℓ_2 norm, i.e., $\|x\|_2 = \left(\sum_{i=1}^n x_i^2\right)^{\frac{1}{2}}$, and use $\|x\|_\infty$ to denote the ℓ_∞ norm.

For all $m, n \in \mathbb{Z}_+$, we use $\mathbb{R}^{m \times n}$ to denote the set containing all matrices with m rows, n columns, and real entries. For a matrix $A \in \mathbb{R}^{m \times n}$, we use $\|A\|$ to denote the spectral norm of A, i.e., $\|A\| = \max_{\|x\|_2=1} \|Ax\|_2$. We use $A^\top$ to denote the transpose of A. We use $\sigma_{\min}(A)$ to denote the minimum singular value of A, i.e., $\sigma_{\min}(A) = \min_{\|x\|_2=1} \|Ax\|_2$. Accordingly, we use $\sigma_{\max}(A)$ to denote the maximum singular value of A, so $\sigma_{\max}(A) = \|A\|$. Furthermore, we use $A^\dagger$ to denote the pseudo-inverse of A and use A^{-1} to denote the true inverse of A. The true inverse exists if $m = n$ and $\mathrm{rank}\,(A) = m$. For all $m_1, m_2, n \in \mathbb{Z}_+$, for all the matrices $A \in \mathbb{R}^{m_1 \times n}$ and $B \in \mathbb{R}^{m_2 \times n}$, we use $A \oplus B$ to denote the matrix $\begin{bmatrix} A \\ B \end{bmatrix}$. Correspondingly, for all $m_1, m_2, \ldots, m_p, p, n \in \mathbb{Z}_+$, for all the matrices $A_1 \in \mathbb{R}^{m_1 \times n}$, $A_2 \in \mathbb{R}^{m_2 \times n}$, $\cdots$, $A_p \in \mathbb{R}^{m_p \times n}$, we use $\oplus_{i=1}^p A_i$ to denote $A_1 \oplus A_2 \oplus \cdots \oplus A_p$.

We write $x = y \pm \epsilon$ if $x \in [y - \epsilon, y + \epsilon]$. $\mathbf{1}_n$ represents an n-dimensional vector, whose entries are all 1, and $\mathbf{0}_n$ represents a n-dimensional vector, whose entries are all 0. For a symmetric matrix $B \in \mathbb{R}^{n \times n}$, we say B is positive semidefinite (denoted as $B \succeq 0$) if, for any vectors x in $\mathbb{R}^n$, the inequality $x^\top B x \geq 0$ always holds. We also call B a PSD matrix for simplicity.

A.2 Definitions and Facts

In this section, we introduce the basic definitions and facts.

Definition A.1. *We use $\kappa(A)$ to denote the condition number of A, i.e.,*

$$\kappa(A) := \sigma_{\max}(A)/\sigma_{\min}(A).$$

Definition A.2 (Hadamard matrix)**.** *A Hadamard matrix is a square matrix of size n with entries of either 1 or -1, where each row of the matrix is orthogonal to every other row.*

Fact A.3. *We have*

- *For any matrix A, $\|(A^\top A)^\dagger\| = \sigma_{\min}(A)^{-2}$.*
- *For any matrix A, $\|A^\top A\| = \sigma_{\max}(A)^2$.*

- *For any matrix A, for any positive integer k, $\|((A^\top A)^k)^\dagger\| \leq \sigma_{\min}(A)^{-2k}$.*
- *For any orthonormal column basis $U \in \mathbb{R}^{n \times d}$ $(n \geq d)$, we have $\|Ux\|_2 = \|x\|_2$.*
- *For any matrix A, we have $\|Ax\| \leq \|A\| \cdot \|x\|_2 = \sigma_{\max}(A) \cdot \|x\|_2$.*
- *For any matrix A, we have $\|Ax\|_2 \geq \sigma_{\min}(A) \cdot \|x\|_2$.*
- *For any matrix A, $\kappa(A) = \kappa(A^\dagger)$.*
- *For any matrix A, B, $\kappa(A) \leq \kappa(AB) \cdot \kappa(B)$.*
- *For any matrix A, b, $\kappa(AB) \leq \kappa(A) \cdot \kappa(B)$.*

Definition A.4 (Tensor Product)**.** *Let $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$. We use $x \otimes y$ to denote the tensor product of x and y, and it is defined as*

$$x \otimes y := \mathrm{vec}(xy^\top).$$

We use $x^{\otimes p}$ to represent x tensoring with itself for p times.

A.3 Attention Backgrounds

In this section, we introduce the important background of attention mechanism and matrix. Given matrices $Q, K, V \in \mathbb{R}^{d \times d}$ as weights, and $X \in \mathbb{R}^{n \times d}$ which be viewed as the embedding of a length-n sentence where each length-d vector is corresponding to a word. The attention matrix is defined as

$$D(X)^{-1} \exp(XQK^\top X^\top)XV \tag{8}$$

where $D(X) = \mathrm{diag}(\exp(XQK^\top X^\top)\mathbf{1}_n)$ and $\exp(\cdot)$ is applied entrywise.

In Alman and Song (2023); Brand et al. (2023), they simplify the computation by treating XQ, XK, and XV as $Q, K, V \in \mathbb{R}^{n \times d}$ so they obtain

$$D^{-1} \exp(QK^\top)V \tag{9}$$

where $D = \mathrm{diag}(\exp(QK^\top)\mathbf{1}_n)$.

In Gao et al. (2023c), they simplify the attention by treating V as identity, namely

$$D^{-1} \exp(QK^\top), \tag{10}$$

where D, Q, K are defined same as above.

In addition, in Deng et al. (2023b), the attention is simplified into the form of

$$D^{-1} \exp(KK^\top), \tag{11}$$

where D, Q, K, for $(K = Q)$, are defined same as above.

In this work, we provide a simplification of Eq. (8) from a different perspective, by ignoring the factor of D^{-1} and exp, so that we can get

$$XQK^\top X^\top XV$$

Further, we merge $QK^\top$ into one matrix W and consider one column of V a time,

$$XWX^\top Xv \tag{12}$$

where v is a column of V.

Thus, we can obtain the following definition of attention formulation.

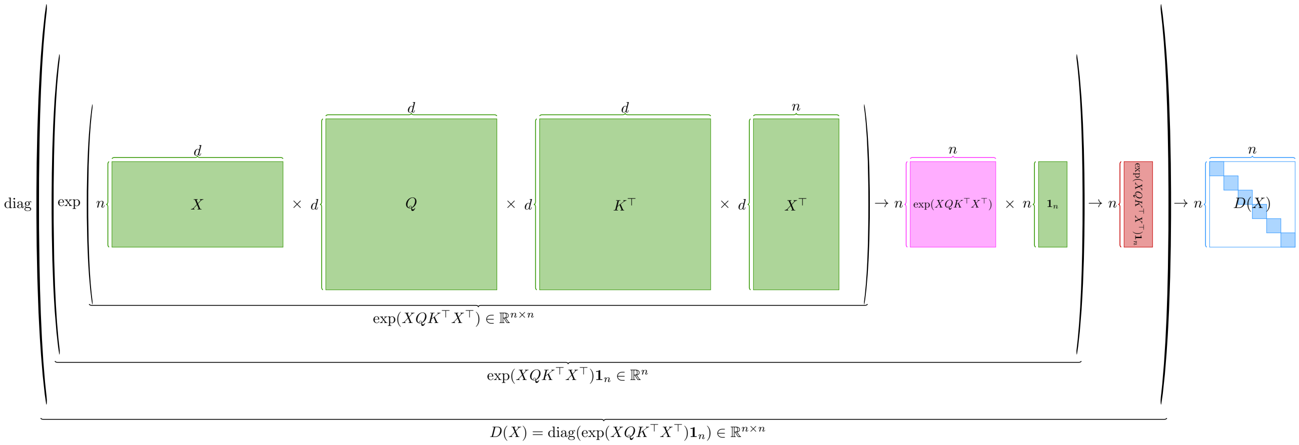
diag
exp
n
d
X
$\times$ d
Q
$K^\top$
$X^\top$
$\exp(XQK^\top X^\top)$
$\mathbf{1}_n$
$\exp(XQK^\top X^\top)\mathbf{1}_n$
$D(X)$
$\exp(XQK^\top X^\top) \in \mathbb{R}^{n \times n}$
$\exp(XQK^\top X^\top)\mathbf{1}_n \in \mathbb{R}^n$
$D(X) = \mathrm{diag}(\exp(XQK^\top X^\top)\mathbf{1}_n) \in \mathbb{R}^{n \times n}$

Figure 1: The visualization of the matrix $D(X) \in \mathbb{R}^{n \times n}$. Given $Q, K, V \in \mathbb{R}^{d \times d}$ and $X \in \mathbb{R}^{n \times d}$, we first compute $XQK^\top X^\top \in \mathbb{R}^{n \times n}$. Then, we find $\exp(XQK^\top X^\top) \in \mathbb{R}^{n \times n}$. After that, we multiply $\exp(XQK^\top X^\top) \in \mathbb{R}^{n \times n}$ with the vector $\mathbf{1}_n \in \mathbb{R}^n$. Finally, we use $\mathrm{diag}(\cdot)$ to transform $\exp(XQK^\top X^\top)\mathbf{1}_n \in \mathbb{R}^n$ into a diagonal matrix, which is $D(X) \in \mathbb{R}^{n \times n}$. In this figure, green matrices/vectors represent the terms that are given; the purple matrix represents the term after one operation; the red vector represents the term after two operations; the blue matrix represents the term after three operations.

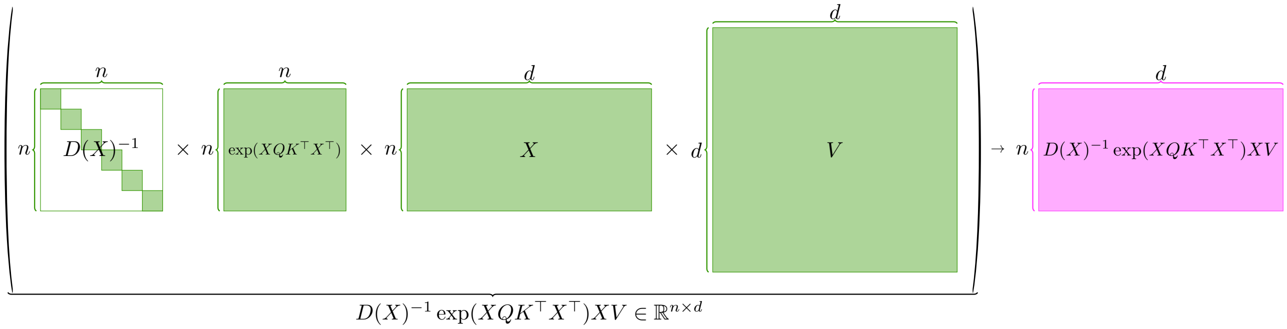
n
d
$D(X)^{-1}$
$\exp(XQK^\top X^\top)$
X
V
$D(X)^{-1}\exp(XQK^\top X^\top)XV$
$D(X)^{-1}\exp(XQK^\top X^\top)XV \in \mathbb{R}^{n \times d}$

Figure 2: The visualization of the attention computation (see Eq. (8)). Since we present the visualization of how we get $D(X) \in \mathbb{R}^{n \times n}$ and $\exp(XQK^\top X^\top) \in \mathbb{R}^{n \times n}$ in Figure 1, we regard them as given. Moreover, we are also given $V \in \mathbb{R}^{d \times d}$ and $X \in \mathbb{R}^{n \times d}$. We compute their product, namely $D(X)^{-1}\exp(XQK^\top X^\top)XV$. In this figure, green matrices represent the terms that are given, and the purple matrix represents the term after one operation.

Definition A.5. *Given $X \in \mathbb{R}^{n \times d}$, $W \in \mathbb{R}^{d \times d}$ and $y \in \mathbb{R}^n$, the goal is to solve the following regression problem*

$$\min_{v \in \mathbb{R}^d} \|XWX^\top X \cdot v - y\|_2^2$$

We show that it is in fact equivalent to the odd power regression of degree 3, defined as follows.

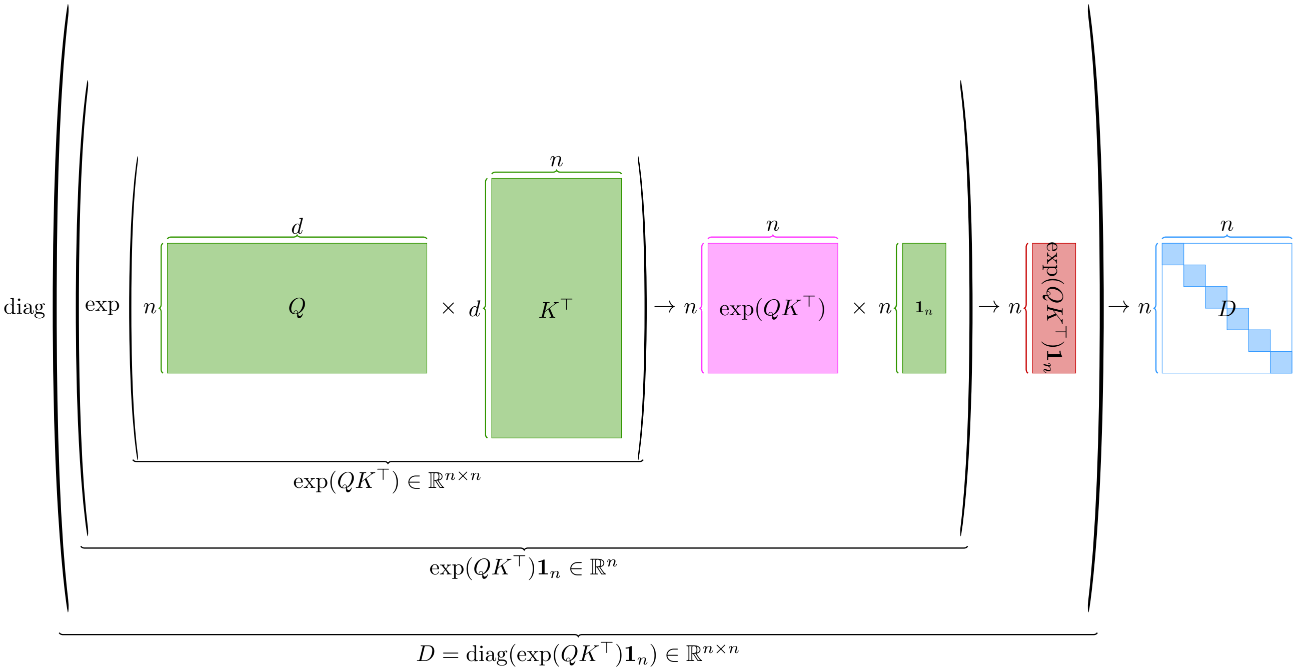
diag
exp
n
d
Q
$\times$
d
n
$K^\top$
$\to n$
n
$\exp(QK^\top)$
$\times$ n
$\mathbf{1}_n$
$\to n$
$\exp(QK^\top)\mathbf{1}_n$
$\to n$
n
D
$\exp(QK^\top) \in \mathbb{R}^{n \times n}$
$\exp(QK^\top)\mathbf{1}_n \in \mathbb{R}^n$
$D = \mathrm{diag}(\exp(QK^\top)\mathbf{1}_n) \in \mathbb{R}^{n \times n}$

Figure 3: The visualization of the matrix $D \in \mathbb{R}^{n \times n}$. Given $Q, K, V \in \mathbb{R}^{n \times d}$, we first compute $QK^\top \in \mathbb{R}^{n \times n}$. Then, we find $\exp(QK^\top) \in \mathbb{R}^{n \times n}$. After that, we multiply $\exp(QK^\top) \in \mathbb{R}^{n \times n}$ with the vector $\mathbf{1}_n \in \mathbb{R}^n$. Finally, we use $\mathrm{diag}(\cdot)$ to transform $\exp(QK^\top)\mathbf{1}_n \in \mathbb{R}^n$ into a diagonal matrix, which is $D \in \mathbb{R}^{n \times n}$. In this figure, green matrices/vectors represent the terms that are given; the purple matrix represents the term after one operation; the red vector represents the term after two operations; the blue matrix represents the term after three operations.

Definition A.6. *Given $X \in \mathbb{R}^{n \times d}$, $y \in \mathbb{R}^n$. The goal is to solve*

$$\min_{v \in \mathbb{R}^d} \|XX^\top Xv - y\|_2^2$$

The equivalence can be established as such.

Lemma A.7. *Solving problem A.5 is equivalent to problem A.6.*

Proof. Let $W = UU^\top$. We define $\widetilde{X}$ and $\widetilde{v}$ as follows

$$\widetilde{X} := XU$$
$$\widetilde{v} := U^{-1}v$$

Then we have $X = \widetilde{X}U^{-1}$. Consequentially,

$$\min_{v \in \mathbb{R}^d} \|XWX^\top X \cdot v - y\|_2^2$$

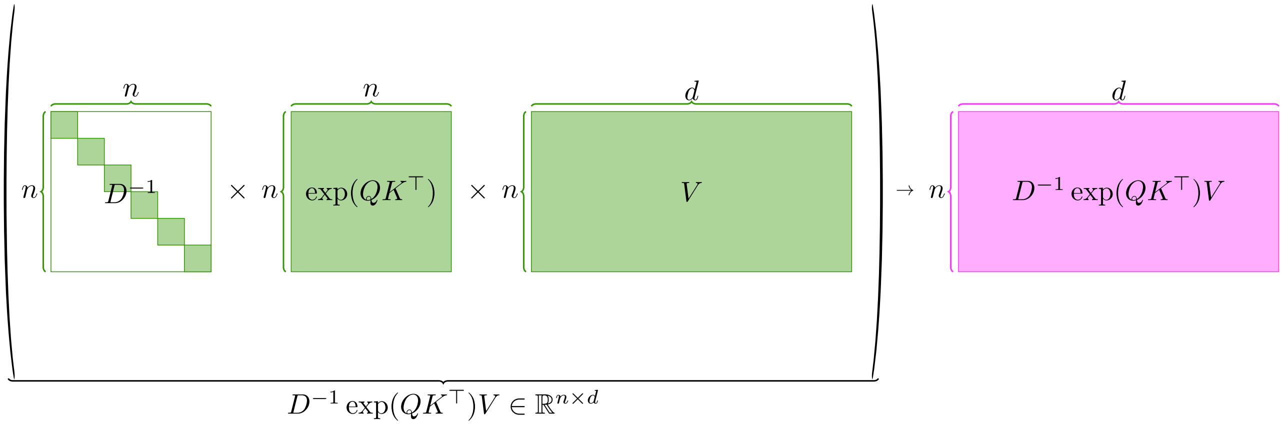

Figure 4: The visualization of the simplified version of attention computation in Alman and Song (2023); Brand et al. (2023) (see Eq. (9)). Since we present the visualization of how we get $D \in \mathbb{R}^{n\times n}$ and $\exp(QK^\top) \in \mathbb{R}^{n\times n}$ in Figure 3, we regard them as given. Moreover, we are also given $V \in \mathbb{R}^{n\times d}$. We compute their product, namely $D^{-1}\exp(QK^\top)V \in \mathbb{R}^{n\times d}$. In this figure, green matrices represent the terms that are given, and the purple matrix represents the term after one operation.

is equivalent to

$$\min_{v\in\mathbb{R}^d} \|\widetilde{X}\widetilde{X}^\top\widetilde{X}U^{-1}v - y\|_2^2$$

and can be rewritten as

$$\min_{\widetilde{v}\in\mathbb{R}^d} \|\widetilde{X}\widetilde{X}^\top\widetilde{X}\widetilde{v} - y\|_2^2$$

Here we use that U is full rank. □

B PRELIMINARY ABOUT EXPONENTIAL OF INNER PRODUCT KERNEL

In Section B.1, we provide a formal definition of the attention kernel. In Section B.2, we analyze the properties of the attention kernel.

B.1 Definition of Attention Kernel

Here, we start to present the definitions of the Gaussian Kernel and the Attention Kernel. We will focus on $\exp(\langle\cdot,\cdot\rangle)$ inner product. It is similar to the Gaussian kernel, let us first review the definition of the Gaussian Kernel

Definition B.1 (Gaussian Kernel). *Let $X \in \mathbb{R}^{d\times n}$. Let x_i be the i-th column of X. We say G is the Gaussian kernel if for any two points $x, y \in \mathbb{R}^d$,*

$$G(x, y) = \exp(-\|x - y\|_2^2/2).$$

We say G is the Gaussian kernel on X if

$$G_{i,j} = \exp(-\|x_i - x_j\|_2^2/2).$$

Here we abuse the notation and let G be an $n \times n$ real matrix.

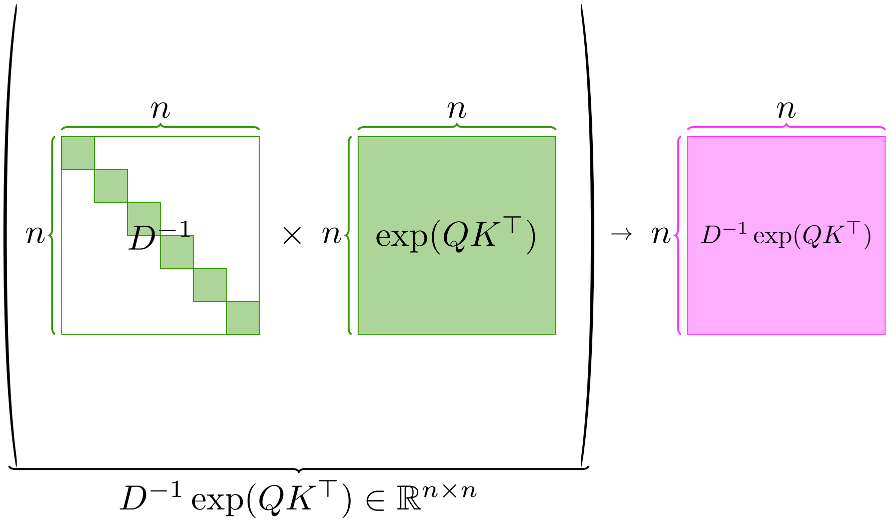

Figure 5: The visualization of the simplified version of attention computation in Gao et al. (2023c) (see Eq. (10)). Since we present the visualization of how we get $D \in \mathbb{R}^{n\times n}$ and $\exp(QK^\top) \in \mathbb{R}^{n\times n}$ in Figure 3, we regard them as given. We compute their product, namely $D^{-1}\exp(QK^\top) \in \mathbb{R}^{n\times n}$. In this figure, green matrices represent the terms that are given, and the purple matrix represents the term after one operation.

We define the attention kernel as follows

Definition B.2 (Attention Kernel)**.** *Let $X \in \mathbb{R}^{d\times n}$. Let x_i be the i-th column of X. We say G is the Gaussian kernel if for any two points $x, y \in \mathbb{R}^d$,*

$$G(x, y) = \exp(\langle x, y\rangle).$$

We say G is the Gaussian kernel on X if

$$G_{i,j} = \exp(\langle x_i, x_j\rangle).$$

Here we abuse the notation and let G be an $n \times n$ real matrix.

B.2 Properties of Attention Kernel

After we define the attention kernel, we start to analyze its properties.

Fact B.3. *Let B be a PSD matrix in $\mathbb{R}^{n\times n}$.*

Then we have

$$B_{i,i}B_{j,j} \geq B_{i,j}^2, \quad \forall i, j \in [n] \times [n]$$

Proof. Let $x = a \cdot e_i + b \cdot e_j$.

For all arbitrary $a, b \in \mathbb{R}$, we can get

$$0 \leq x^\top Bx$$

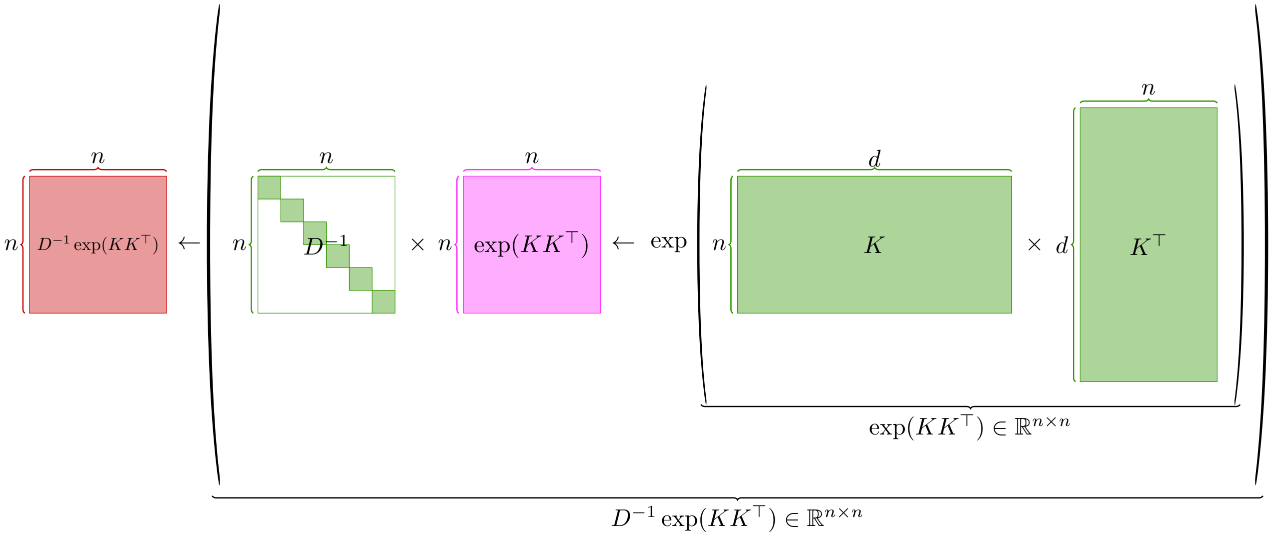

Figure 6: The visualization of the simplified version of attention computation in Deng et al. (2023b) (see Eq. (11)). Since we present the visualization of how we get $D \in \mathbb{R}^{n \times n}$ in Figure 3, we regard it as given. We are also given that $K \in \mathbb{R}^{n \times d}$, which implies that $K^\top \in \mathbb{R}^{d \times n}$. First, we compute their product, namely $KK^\top \in \mathbb{R}^{n \times n}$. Then, we find $\exp(KK^\top) \in \mathbb{R}^{n \times n}$. Finally, we multiply $D^{-1} \in \mathbb{R}^{n \times n}$ with $\exp(KK^\top) \in \mathbb{R}^{n \times n}$, which gives us $D^{-1}\exp(KK^\top) \in \mathbb{R}^{n \times n}$. In this figure, green matrices represent the terms that are given; the purple matrix represents the term after one operation; the red matrix represents the term after two operations.

$$= \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} B_{i,i} & B_{i,j} \\ B_{j,i} & B_{j,j} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}, \tag{13}$$

where the first step follows from the fact that B is a PSD matrix and the second step follows from expanding the equation.

Note that for all arbitrary $a, b \in \mathbb{R}$, Eq. (13) holds.

Therefore,

$$\det\left(\begin{bmatrix} B_{i,i} & B_{i,j} \\ B_{j,i} & B_{j,j} \end{bmatrix} \right) \geq 0$$

which is equivalent to say $B_{i,i}B_{j,j} \geq B_{i,j}^2$. $\square$

Lemma B.4. *Let $X \in \mathbb{R}^{n \times d}$. We define attention kernel*

$$A := \exp(XX^\top) \in \mathbb{R}^{n \times n}$$

where $\exp(\cdot)$ is applied entrywise. Let $\epsilon \in (0, 0.1)$, $r > 0$ and a matrix $B \in \mathbb{R}^{n \times n}$ satisfy the following conditions

- *Condition 1. $\epsilon \leq \frac{1}{4}\exp(-4r)$.*
- *Condition 2. $A_{i,j} \in [\exp(-r), \exp(r)]$ for $i, j \in [n] \times [n]$.*

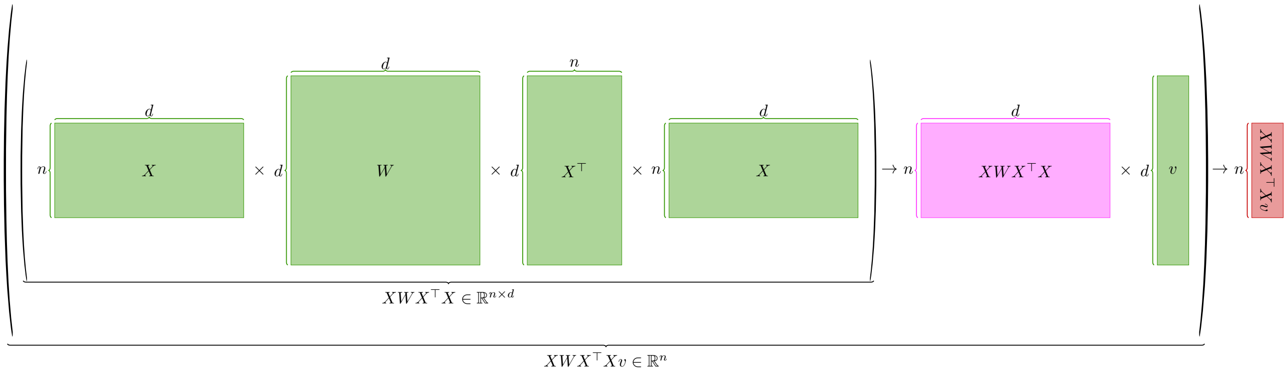

Figure 7: The visualization of the simplified version of attention computation that we analyze in this paper (see Eq. (12)). We are given that $X \in \mathbb{R}^{n \times d}$, $W \in \mathbb{R}^{d \times d}$, and $v \in \mathbb{R}^d$. First, we compute the product of the matrices, namely $XWX^\top X \in \mathbb{R}^{n \times d}$. Then, we multiply $XWX^\top X \in \mathbb{R}^{n \times d}$ with the vector $v \in \mathbb{R}^d$, which gives us $XWX^\top Xv \in \mathbb{R}^n$ In this figure, green matrices represent the terms that are given; the purple matrix represents the term after one operation; the red matrix represents the term after two operations.

- *Condition 3.* $(1-\epsilon) \cdot A \preceq B \preceq (1+\epsilon) \cdot A$.

Then, we have

$$B_{i,j} \in [(1-\sqrt{\epsilon})\exp(-r), (1+\epsilon)\exp(r)].$$

Proof. $B - (1-\epsilon) \cdot A$ is a PSD matrix.

Thus, we have

$$\begin{aligned} |B_{i,j} - (1-\epsilon) \cdot A_{i,j}| \leq & ~ \sqrt{(B_{i,i} - (1-\epsilon) \cdot A_{i,i})(B_{j,j} - (1-\epsilon) \cdot A_{j,j})} \\ \leq & ~ 2\epsilon \sqrt{A_{i,i} A_{j,j}} \\ \leq & ~ 2\epsilon \exp(r) \end{aligned} \tag{14}$$

where the 1st step is by Lemma B.3, the 2nd step is due to $B_{i,i} \leq (1+\epsilon) A_{i,i}$ (By condition 3 in lemma statement), and the 3rd step is because of the definition of $A_{i,j}$ from the lemma statement.

Based on the second condition from the lemma statement, we have

$$(1-\epsilon) \cdot A_{i,j} \in [(1-\epsilon) \cdot \exp(-r), (1-\epsilon) \cdot \exp(r)], \tag{15}$$

Combining Eq. (14) and Eq. (15), we have

$$B_{i,j} \in [(1-\epsilon)\exp(-r) - 2\epsilon\exp(r), (1+\epsilon)\exp(r)] \tag{16}$$

By using $(1+\epsilon) \cdot A - B$ is a PSD matrix, we may do a symmetric argument as follows:

$$B_{i,j} \in [(1+\epsilon)\exp(-r) - 2\epsilon\exp(r), (1+3\epsilon)\exp(r)]. \tag{17}$$

The intersection of Eq. (16) and Eq. (17) gives us:

$$B_{i,j} \in [(1+\epsilon)\exp(-r) - 2\epsilon\exp(r), (1+\epsilon)\exp(r)]$$

We know that

$$\begin{aligned}(1+\epsilon)\exp(-r)-2\epsilon\exp(r) \geq & ~ \exp(-r)-2\epsilon\exp(r) \\ = & ~ \exp(-r)-2\sqrt{\epsilon}\cdot\sqrt{\epsilon}\cdot\exp(r) \\ \geq & ~ \exp(-r)-2\cdot\sqrt{\epsilon}\cdot\frac{1}{2}\exp(-2r)\cdot\exp(r) \\ = & ~ \exp(-r)-\sqrt{\epsilon}\exp(-r) \\ = & ~ (1-\sqrt{\epsilon})\cdot\exp(-r)\end{aligned}$$

where the first step follows from $\epsilon \geq 0$, the second step follows from simple algebra, the third step follows from $\sqrt{\epsilon} \leq \frac{1}{2}\exp(-2r)$ (by condition 1 in lemma statement), the fourth step follows from simple algebra, and the last step follows from simple algebra.

Thus, we have

$$B_{i,j} \in [(1-\sqrt{\epsilon})\exp(-r),(1-\epsilon)\exp(r)].$$

This completes the proof. ☐

C PRELIMINARY ABOUT STANDARD SKETCHING

In Section C.1, we provide two types of fast sketching matrices. In Section C.2, we introduce the formal definition of subspace embedding. In Section C.3, we interpret and analyze the property of subspace embedding by different matrices. In Section C.4, we provide the definition of the Frobenius norm approximate matrix product.

C.1 Sketching Matrices

We define subsampled randomized Hadamard transform (SRHT) as follows.

Definition C.1 (Subsampled Randomized Hadamard Transform (SRHT), see Lu et al. (2013); Woodruff (2014))**.** *Let H be a Hadamard matrix in $\mathbb{R}^{d\times d}$ (see Definition A.2). Let D be a diagonal matrix in $\mathbb{R}^{d\times d}$, which satisfies that each diagonal entry is either -1 or 1 with the same probability. Let P be a matrix in $\{0,1\}^{m\times d}$, where each row of P contains only one 1 at a random entry. We define the matrix $S \in \mathbb{R}^{m\times d}$ as*

$$S := \frac{1}{\sqrt{m}}PHD$$

and call S the SRHT *matrix.*

We define TensorSRHT as follows:

Definition C.2 (Tensor Subsampled Randomized Hadamard Transform (TensorSRHT), Definition 2.9 in Song et al. (2021a))**.** *Let P be a matrix in $\{0,1\}^{m\times d}$, where each row of P contains only one 1 at a random entry. P can be regarded as the sampling matrix. Let H be the Hadamard matrix in $\mathbb{R}^{d\times d}$ (see Definition A.2). Let D_1 and D_2 be diagonal matrices in $\mathbb{R}^{d\times d}$, which satisfy that each diagonal entry is either -1 or 1 with the same probability. D_1 and D_2 are independent. Then, we define* TensorSRHT *as a function $S : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^m$, which is defined as*

$$S := \frac{1}{\sqrt{m}}P\cdot(HD_1 \otimes HD_2).$$

C.2 Subspace Embedding

We define subspace embedding as follows.

Definition C.3 (Subspace Embedding $\mathsf{SE}(n,d,\epsilon,\delta)$, Sarlos (2006))**.** *Given a matrix $A \in \mathbb{R}^{n\times d}$, we say S is an (n,d,ϵ,δ) subspace embedding, if*

$$\Pr[(1-\epsilon)\cdot\|Ax\|_2 \leq \|SAx\|_2 \leq (1+\epsilon)\cdot\|Ax\|_2] \geq 1-\delta.$$

Equivalently,

$$\Pr[(1-\epsilon)\cdot(A^\top A) \preceq A^\top S^\top SA \preceq (1+\epsilon)\cdot(A^\top A)] \geq 1-\delta.$$

C.3 Subspace Embedding by Different Sketches

We specify several classes of subspace embedding that obtain a small sketching dimension and can be applied to A fast.

Lemma C.4. *Given a matrix $A \in \mathbb{R}^{n \times d}$, S is an (n, d, ϵ, δ) subspace embedding for*

- *Let $S \in \mathbb{R}^{m \times n}$ denote a SRHT matrix with $m = O(\epsilon^{-2} d \log(n/\delta))$ rows. In addition, SA can be computed in $O(nd \log(n/\delta))$ time.*
- *Let $S \in \mathbb{R}^{m \times n}$ denote OSNAP matrix with $m = O(\epsilon^{-2} d \log(d/\delta))$ rows and column sparsity $s = O(\epsilon^{-1} \log(d/\delta))$. In addition, SA can be computed in $O(s \cdot \mathrm{nnz}(A))$ time.*

C.4 Approximate Matrix Product

Here, we define the Frobenius Norm Approximate Matrix Product as follows.

Definition C.5 (Frobenius Norm Approximate Matrix Product)**.** *Let $A \in \mathbb{R}^{n \times d_1}, B \in \mathbb{R}^{n \times d_2}$ be two matrices. We say S is Frobenius Norm Approximate Matrix Product $\mathsf{FAMP}(n, \epsilon, \delta)$ with respect to A, B if*

$$\Pr[\|A^\top B - A^\top S^\top S B\|_F \leq \epsilon \|A\|_F \|B\|_F] \geq 1 - \delta.$$

D PRELIMINARY ABOUT HIGH PRECISION SKETCHING

In Section D.1, we explain more useful facts regarding matrix spectrum. In Section D.2, we analyze the property of well-conditioned PSD regression. In Section D.3, we analyze the forward error for overconstrained systems. In Section D.4, we study the forward error for PSD matrices.

D.1 Norm Preservation and Spectrum

Fact D.1. *The following two conditions are equivalent:*

- *For all unit vector x, $1 - \epsilon \leq \|Ax\|_2^2 \leq 1 + \epsilon$;*
- *$\|A^\top A - I\| \leq \epsilon$.*

Proof. We know that,

$$\|A^\top A - I\| \leq \epsilon$$

is equivalent to

$$-\epsilon \leq x^\top (A^\top A - I) x \leq \epsilon, \quad \forall \|x\|_2 = 1$$

move $-x^\top x$ to both sides we have

$$1 - \epsilon \leq x^\top A^\top A x \leq 1 + \epsilon, \quad \forall \|x\|_2 = 1$$

which is equivalent to

$$1 - \epsilon \leq \|Ax\|_2^2 \leq 1 + \epsilon, \quad \forall \|x\|_2 = 1.$$

□

D.2 Well-Conditioned PSD Regression

We state a lemma for solving well-conditioned PSD regression, due to Brand et al. (2021).

Lemma D.2 (Well-conditioned PSD regression, Lemma B.2 in Brand et al. (2021))**.** *Consider the regression problem*

$$\min_{x \in \mathbb{R}^d} \|Bx - y\|_2^2.$$

Suppose $B \in \mathbb{R}^{d \times d}$ is a PSD matrix with

$$\frac{3}{4} \leq \|Bx\|_2 \leq \frac{5}{4}, \quad \forall x \text{ such that } \|x\|_2 = 1.$$

Using gradient descent update

$$x_{t+1} = x_t - B^\top (Bx_t - y).$$

Then, after t iterations, we obtain

$$\|B(x_t - x^*)\|_2 \leq c^t \|B(x_0 - x^*)\|_2$$

for some constant $c \in (0, 0, 9]$.

Proof. The gradient at time t is $B^\top(Bx_t - y)$ and

$$x_{t+1} = x_t - B^\top (Bx_t - y), \tag{18}$$

so we have

$$\begin{aligned} \|Bx_{t+1} - Bx^*\|_2 &= \|B(x_t - B^\top(Bx_t - y)) - Bx^*\|_2 \\ &= \|B(x_t - x^*) - BB^\top Bx_t + BB^\top Bx^*\|_2 \\ &= \|B(x_t - x^*) - BB^\top B(x_t - x^*)\|_2 \\ &= \|(I - BB^\top)B(x_t - x^*)\|_2 \\ &\leq \|(I - BB^\top)\| \cdot \|B(x_t - x^*)\|_2 \\ &\leq \frac{9}{16} \|B(x_t - x^*)\|_2, \end{aligned}$$

where the first step follows from Eq. (18), the second step follows from $B^\top Bx^* = B^\top y$, the third step follows from simple algebra, the fourth step follows from simple algebra, the fifth step follows from $\|Ax\|_2 \leq \|A\| \cdot \|x\|_2$, and the last step follows from the fact that the eigenvalue of $BB^\top$ belongs to $[\frac{9}{16}, \frac{25}{16}]$ by our assumption.

Thus we complete the proof. □

D.3 Forward Error for Overconstrained System

Given a fast regression solver with error guarantees on the cost, it can be conveniently converted to guarantees on the solution. We include the proof of a lemma in Gu et al. (2024) for completeness.

Lemma D.3 (Lemma 5.5 in Gu et al. (2024))**.** *Given a matrix $A \in \mathbb{R}^{n \times d}$, a vector $b \in \mathbb{R}^n$. Suppose there is a vector $x' \in \mathbb{R}^d$ such that*

$$\|Ax' - b\|_2 \leq (1 + \epsilon_1) \min_{x \in \mathbb{R}^d} \|Ax - b\|_2$$

Let x_* denote the exact solution to the regression problem, then it holds that

$$\|x' - x_*\|_2 \leq O(\sqrt{\epsilon_1}) \cdot \frac{1}{\sigma_{\min}(A)} \cdot \|Ax_* - b\|_2.$$

Proof. Note that

$$\|Ax' - Ax_*\|_2 = \|Ax' - b - (Ax_* - b)\|_2,$$

so we can perform the following decomposition:

$$\begin{aligned}\|A(x' - x_*)\|_2^2 &= \|Ax' - b - (Ax_* - b)\|_2^2 \\ &= \|Ax' - b\|_2^2 - \|Ax_* - b\|_2^2 \\ &\leq (1 + \epsilon_1)^2 \|Ax_* - b\|_2^2 - \|Ax_* - b\|_2^2 \\ &\leq 4\epsilon_1 \cdot \|Ax_* - b\|_2^2, \end{aligned} \tag{19}$$

where the first step follows from simple algebra, the second step follows from the Pythagorean theorem, the third step follows from the assumption in Lemma statement, and the fourth step follows from simple algebra.

Assuming A has full column rank, then $A^\dagger A = I$. Therefore, we have

$$\begin{aligned}\|x' - x_*\|_2 &= \|A^\dagger A(x' - x_*)\|_2 \\ &\leq \|A(x' - x_*)\|_2 \cdot \|A^\dagger\| \\ &\leq 2\sqrt{\epsilon_1} \cdot \|Ax_* - b\|_2 \cdot \|A^\dagger\| \\ &= \frac{2\sqrt{\epsilon_1}}{\sigma_{\min}(A)} \cdot \|Ax_* - b\|_2,\end{aligned}$$

where the first step follows from $A^\dagger A = I$, the second step follows from $\|ABx\|_2 \leq \|Ax\|_2 \|B\|$, the third step follows from Eq. (19), and the last step follows from $\|A^\dagger\| = \|A^{-1}\| = \frac{1}{\sigma_{\min}(A)}$. ☐

D.4 Forward Error for PSD Matrices

When the matrix in question is PSD, we can obtain a similar error guarantees, in terms $\|b\|_2$ instead of the cost $\|Ax_* - b\|_2$.

Lemma D.4 (PSD version of Lemma D.3)**.** *Given a matrix $A \in \mathbb{R}^{n \times d}$, a vector $b \in \mathbb{R}^d$. Suppose there is a vector $x' \in \mathbb{R}^d$ such that*

$$\|A^\top A x' - b\|_2 \leq \epsilon_2 \|b\|_2$$

Let x_* denote the exact solution to the regression problem, then it holds that

$$\|x' - x_*\|_2 \leq \epsilon_2 \cdot \frac{1}{\sigma_{\min}(A)^2} \cdot \|b\|_2.$$

Proof. Note that

$$\|A^\top A x' - A^\top A x_*\|_2 = \|(A^\top A x' - b) - (A^\top A x_* - b)\|_2,$$

so we can perform the following decomposition:

$$\begin{aligned}\|A^\top A(x' - x_*)\|_2^2 &= \|(A^\top A x' - b) - (A^\top A x_* - b)\|_2^2 \\ &= \|A^\top A x' - b\|_2^2 - \|A^\top A x_* - b\|_2^2 \\ &\leq \epsilon_2^2 \|b\|_2^2 \end{aligned} \tag{20}$$

where the first step follows from simple algebra, the second step follows from the Pythagorean theorem, the third step follows from the assumption in lemma statement.

Assuming A has full column rank, then $(A^\top A)^\dagger (A^\top A) = I$. Therefore, we have

$$\begin{aligned}\|x' - x_*\|_2 &= \|(A^\top A)^\dagger A^\top A(x' - x_*)\|_2 \\ &\leq \|A^\top A(x' - x_*)\|_2 \cdot \|(A^\top A)^\dagger\| \\ &\leq \epsilon_2 \cdot \|b\|_2 \cdot \|(A^\top A)^\dagger\| \\ &= \epsilon_2 \cdot \sigma_{\min}(A)^{-2} \|b\|_2\end{aligned}$$

where the first step follows from $(A^\top A)^\dagger (A^\top A) = I$, the second step follows from $\|ABx\|_2 \leq \|Ax\|_2 \|B\|$, the third step follows from Eq. (20), and the last step follows from Fact A.3. ☐

E FAST REGRESSION VIA SKETCHING AND PRECONDITIONING

E.1 Linear Regression

In this section, we present the fast linear regression algorithm and analyze its error guarantees and efficiency.

Algorithm 4 Algorithm for solving $\min_x \|Ax - b_1\|_2$.

```
procedure FastLinearRegression($A \in \mathbb{R}^{n \times d}, b_1 \in \mathbb{R}^n, n \in \mathbb{Z}_+, d \in \mathbb{Z}_+, \epsilon_1 \in (0,1), \delta_1 \in (0,1)$)
    Compute a subspace embedding $S_1$ and apply it to $A$   ▷ Let $S_1$ denote a $\mathsf{SE}(n, d, \epsilon_{\mathrm{ose}} = 0.1, \delta_{\mathrm{ose}} = \delta_1/2)$
    Compute $R$ such that $S_1AR$ are orthonormal columns via QR decomposition $R \in \mathbb{R}^{d \times d}$
    $T_1 \leftarrow \Theta(\log(1/\epsilon_1))$
    $z_0 \leftarrow \arg\min_x \|SARx - Sb_1\|_2$
    $t \leftarrow 0$
    while $\|ARz_t - b_1\|_2 \geq \epsilon_1$ do
        $z_{t+1} \leftarrow z_t - (R^\top A^\top)(ARz_t - b_1)$
        $t \leftarrow t + 1$
    end while
    return $Rz_t$
end procedure
```

Lemma E.1 (Dense and High Accuracy Regression, Lemma 5.4 in Gu et al. (2024))**.** *Given a matrix* $A \in \mathbb{R}^{n \times d}$ *and a vector* $b_1 \in \mathbb{R}^n$*, let* $\epsilon_1 \in (0, 0.1)$ *and* $\delta_1 \in (0, 0.1)$*, there exists an algorithm that takes time*

$$O((nd + d^3) \cdot \log(1/\epsilon_1) \cdot \log^2(n/\delta_1))$$

and outputs $x' \in \mathbb{R}^d$ *such that*

$$\|Ax' - b_1\|_2 \leq (1 + \epsilon_1) \min_{x \in \mathbb{R}^d} \|Ax - b_1\|_2$$

holds with probability $1 - \delta_1$*.*

Proof. Let us analyze Algorithm 4, first its convergence then its runtime. Note that the S we choose is an $(\epsilon_{\mathrm{ose}}, \delta_{\mathrm{ose}})$-oblivious subspace embedding. Since $SA = QR^{-1}$ where Q is orthonormal, we know the singular values of AR are between $[1 - \epsilon_{\mathrm{ose}}, 1 + \epsilon_{\mathrm{ose}}]$. Let $AR = U\Sigma V^\top$ be the SVD of AR and z^* denote the optimal solution to the regression $\min_{x \in \mathbb{R}^d} \|ARx - b_1\|_2$. Let us consider

$$\begin{aligned} AR(z_{t+1} - z^*) &= AR(z_t + R^\top A^\top (b_1 - ARz_t) - z^*) \\ &= AR(z_t - z^*) + ARR^\top A^\top b_1 - ARR^\top A^\top ARz_t \\ &= AR(z_t - z^*) + ARR^\top A^\top ARz^* - ARR^\top A^\top ARz_t \\ &= (AR - ARR^\top A^\top AR)(z_t - z^*) \\ &= (U\Sigma V^\top - U\Sigma^3 V^\top)(z_t - z^*), \end{aligned} \tag{21}$$

where the first step follows from the definition of z_{t+1} from Algorithm 4, the second step follows from simple algebra, the third step follows from $b_1 = ARz^*$, the fourth step follows from simple algebra, the last step follows from the SVD, $AR = U\Sigma V^\top$.

Therefore,

$$\begin{aligned} \|AR(z_{t+1} - z^*)\|_2 &= \|(U\Sigma V^\top - U\Sigma^3 V^\top)(z_t - z^*)\|_2 \\ &= \|(\Sigma - \Sigma^3)V^\top (z_t - z^*)\|_2 \\ &\leq O(\epsilon_{\mathrm{ose}}) \cdot \|V^\top (z_t - z^*)\|_2 \\ &\leq \frac{O(\epsilon_{\mathrm{ose}})}{1 - \epsilon_{\mathrm{ose}}} \|\Sigma V^\top (z_t - z^*)\|_2 \\ &= O(\epsilon_{\mathrm{ose}}) \cdot \|\Sigma V^\top (z_t - z^*)\|_2 \end{aligned}$$

$$\begin{aligned}
&= O(\epsilon_{\rm ose}) \cdot \|U\Sigma V^\top (z_t - z^*)\|_2 \\
&= O(\epsilon_{\rm ose}) \cdot \|AR(z_t - z^*)\|_2,
\end{aligned}$$

where the first step follows from Eq. (21), the second step follows from $U^\top U = I$, the third step follows from $\|AB\| \leq \|A\| \cdot \|B\|$, the fourth step follows from $(1-\epsilon_{\rm ose}) \leq \|\Sigma\|$, the fifth step follows from $\epsilon_{\rm ose} \in (0, 0.1)$, the sixth step follows from $U^\top U = I$, and the last step follows from the SVD, $AR = U\Sigma V^\top$.

This means the error shrinks by a factor of $O(\epsilon_{\rm ose})$ per iteration. After $T = O(\log(1/\epsilon_1))$ iterations, we have

$$\|AR(z_T - z^*)\|_2 \leq O(\epsilon_1) \cdot \|AR(z_0 - z^*)\|_2, \tag{22}$$

and recall for initial solution z_0, we have

$$\|ARz_0 - b_1\|_2 \leq (1+\epsilon_{\rm ose}) \cdot \|ARz^* - b_1\|_2.$$

The above equation implies that

$$\|ARz_0 - b_1\|_2^2 - \|ARz^* - b_1\|_2^2 \leq O(\epsilon_{\rm ose}) \|ARz^* - b_1\|_2^2. \tag{23}$$

We can wrap up the proof as follows:

$$\begin{aligned}
\|ARz_T - b_1\|_2^2 &= \|AR(z_T - z^*)\|_2^2 + \|ARz^* - b_1\|_2^2 \\
&\leq O(\epsilon_1^2) \cdot \|AR(z_0 - z^*)\|_2^2 + \|ARz^* - b_1\|_2^2 \\
&= O(\epsilon_1^2) \cdot (\|ARz_0 - b_1\|_2^2 - \|ARz^* - b_1\|_2^2) + \|ARz^* - b_1\|_2^2 \\
&\leq O(\epsilon_1^2) \cdot (O(\epsilon_{\rm ose}) \|ARz^* - b_1\|_2^2) + \|ARz^* - b_1\|_2^2 \\
&= (1 + O(\epsilon_1^2)) \cdot \|ARz^* - b_1\|_2^2,
\end{aligned}$$

where the first step follows from the Pythagorean theorem, the second step follows from Eq. (22), the third step follows from the Pythagorean theorem again, the fourth step follows from Eq. (23), and the fifth step follows from $\epsilon_{\rm ose} \leq 1$.

It remains to show the runtime. Applying S to A takes $O(nd \log n)$ time, the QR decomposition takes $O(m_{\rm sk} d^2) = O(d^3 \log^2(n/\delta_{\rm ose}))$ time.

Inverting $d \times d$ matrix Q takes $O(d^3)$ time. To solve for z_0, we need to multiply SA with R in $O(m_{\rm sk} d^2)$ time and the solve takes $O(m_{\rm sk} d^2)$ time as well. To implement each iteration, we multiply from right to left which takes $O(nd)$ time. Putting things together gives the desired runtime. ☐

E.2 Fast PSD Regression Solver

We can extend the algorithm in the preceding section for solving PSD regression.

Algorithm 5 Algorithm for solving $\min_{x \in \mathbb{R}^d} \|A^\top A x - b_2\|_2$.

```
procedure FastPSDRegression($A \in \mathbb{R}^{n \times d}, b_2 \in \mathbb{R}^d, n \in \mathbb{Z}_+, d \in \mathbb{Z}_+, \epsilon_2 \in (0,1), \delta_2 \in (0,1)$)   ▷ Lemma E.2
    Apply $S_2$ to matrix $A$   ▷ Let $S_2$ denote a $\mathsf{SE}(n, d, \epsilon_{\rm ose} = 0.1, \delta_{\rm ose} = \delta_2/2)$
    Compute $R$ such that $S_2 A R$ orthonormal columns via QR decomposition   ▷ $R \in \mathbb{R}^{d \times d}$
    $z_0 \leftarrow \mathbf{0}_d$
    $t \leftarrow 0$
    while $\|A^\top A R z_t - b_2\|_2 \geq \epsilon_2$ do
        $z_{t+1} \leftarrow z_t - (R^\top A^\top A R)^\top (R^\top A^\top A R z_t - R^\top b_2)$
        $t \leftarrow t + 1$
    end while
    return $R z_t$
end procedure
```

Lemma E.2 (Formal Version of Lemma 4.2, Lemma B.1 in Brand et al. (2021))**.** *Given a matrix $A \in \mathbb{R}^{n \times d}$, let κ denote the condition number of A, i.e. $\kappa = \sigma_{\max}(A)/\sigma_{\min}(A)$, consider the following regression problem*

$$\min_{x \in \mathbb{R}^d} \|A^\top A x - b_2\|_2. \tag{24}$$

There is an algorithm that runs in time

$$O((nd + d^3) \cdot \log(\kappa/\epsilon_2) \cdot \log^2(n/\delta_2)).$$

and outputs $x' \in \mathbb{R}^d$ such that

$$\|A^\top A x' - b_2\|_2 \leq \epsilon_2 \|b_2\|_2$$

holds with probability $1 - \delta_2$.

Proof. Let $S_2 \in \mathbb{R}^{s_2 \times n}$ be a $\mathsf{SE}(n, d, \epsilon_{\mathrm{ose}} = 0.1, \delta)$ (Definition C.3) for A, then with probability $1 - \delta$, the following holds for any $x \in \mathbb{R}^d$

$$\|S_2 A x\|_2 = (1 \pm \epsilon_{\mathrm{ose}}) \|Ax\|_2. \tag{25}$$

Suppose $R \in \mathbb{R}^{d \times d}$ is computed so that SAR has orthonormal columns, e.g., via QR decomposition. We use R as a preconditioner for matrix A. Formally, for any $x \in \mathbb{R}^d$ satisfying $\|x\|_2 = 1$, we have

$$\begin{aligned} \|ARx\|_2 &= (1 \pm \epsilon_{\mathrm{ose}}) \|S_2 A R x\|_2 \\ &= (1 \pm \epsilon_{\mathrm{ose}}), \end{aligned} \tag{26}$$

where the first step follows from Eq. (25) and the second step follows from the fact that $S_2 AR$ has orthonormal columns. Taking the squares on both sides, we have

$$\|ARx\|_2^2 = (1 \pm 3\epsilon_{\mathrm{ose}}).$$

By Fact D.1, the above equation implies

$$\|R^\top A^\top A R - I\| \leq 3\epsilon_{\mathrm{ose}}$$

Hence, using the definition of spectral norm, we know for any $\|x\|_2 = 1$,

$$\|R^\top A^\top A R x\|_2 \leq (1 + 3\epsilon_{\mathrm{ose}}),$$

Similarly, we can prove the other direction

$$\|R^\top A^\top A R x\|_2 \geq (1 - 3\epsilon_{\mathrm{ose}}).$$

We choose $\epsilon_{\mathrm{ose}} = 0.1$, and consider the regression problem

$$\min_{z \in \mathbb{R}^n} \|R^\top A^\top A R z - R^\top b_2\|_2. \tag{27}$$

By Lemma D.2, using gradient descent, after $T_2 = \log(1/\epsilon_2)$ iterations, we can find z_t satisfying

$$\|R^\top A^\top A R (z_t - z^*)\|_2 \leq \epsilon \|R^\top A^\top A R (z_0 - z^*)\|_2, \tag{28}$$

where

$$z^* = (R^\top A^\top A R)^{-1} R^\top b_2 \tag{29}$$

is the optimal solution to Eq. (27).

We are going to show that

$$x_t = R z_t \tag{30}$$

is an $2\kappa\epsilon$-approximate solution to the original regression problem (Eq. (24)), i.e.,

$$\|A^\top A x_t - b_2\|_2 \leq \kappa\epsilon_2\|b_2\|_2.$$

Loading Eq. (29) into Eq. (28), we obtain

$$\|R^\top A^\top A R z_t - R^\top b_2\|_2 \leq \epsilon_2\|R^\top A^\top A R z_0 - R^\top b_2\|_2$$

Loading the definition of $z_0 = 0$ and Eq. (30), we have

$$\begin{aligned} \|R^\top A^\top A x_t - R^\top b_2\|_2 &\leq \epsilon_2 \cdot \|R^\top b_2\|_2 \\ &\leq \epsilon_2 \cdot \sigma_{\max}(R) \cdot \|b_2\|_2, \end{aligned} \tag{31}$$

where the second step follows from the definition of $\sigma_{\max}(R)$. On the other hand, we have

$$\begin{aligned} \|R^\top A^\top A x_t - R^\top b_2\|_2 &= \|R^\top (A^\top A x_t - b_2)\|_2 \\ &\geq \sigma_{\min}(R^\top)\|A^\top A x_t - b_2\|_2, \end{aligned} \tag{32}$$

where the first step follows from simple algebra and the second step follows from the definition of $\sigma_{\min}(R^\top)$. Putting it all together, we have

$$\begin{aligned} \|A^\top A x_t - b_2\|_2 &\leq \epsilon_2\kappa(R^\top)\|b_2\|_2 \\ &= \epsilon_2\kappa(R)\|b_2\|_2 \\ &\leq \epsilon_2\kappa(AR)\kappa(A)\|b_2\|_2 \\ &\leq 2\epsilon_2\kappa(A)\|b_2\|_2, \end{aligned}$$

where the first step follows from Eq. (31) and Eq. (32), the second step follows from R is a square matrix and thus $\kappa(R) = \kappa(R^\top)$, the third step follows from Fact A.3, and the last step follows from Eq. (26).

For the running time, the preconditioning time is $\widetilde{O}(nd + d^3)$, the number of iteration for gradient descent is $\log(\kappa/\epsilon_2)$, the running time per iteration is $\widetilde{O}(nd)$, so the total running time is

$$O((nd + d^3) \cdot \log(\kappa/\epsilon_2) \cdot \log^2(n/\delta_2)).$$

$\square$

F EVEN AND ODD POWER REGRESSION: BASE CASE

We showcase the algorithms for solving even and odd power regression when there are only 4 (for even) and 3 (for odd) matrices are involved. This simple case serves as a basis, both algorithmically and analytically, for the more complicated algorithm.

F.1 Three Matrices

Our algorithm will be first solving an overconstrained regression, then followed with a PSD regression.

Lemma F.1. *Let $A \in \mathbb{R}^{n \times d}$ be a matrix and $b \in \mathbb{R}^n$ be a vector. Let κ denote the condition number of A (see Definition A.1), i.e. $\kappa = \sigma_{\max}(A)/\sigma_{\min}(A)$ Consider the regression problem (defined in Definition 1.1):*

$$\min_{x \in \mathbb{R}^d} \|AA^\top A x - b\|_2.$$

There exists an algorithm (Algorithm 6) that runs in time

$$O((nd + d^3) \cdot \log(\kappa/\epsilon_3) \cdot \log^2(n/\delta_3))$$

and outputs a vector $x' \in \mathbb{R}^d$ such that

$$\|AA^\top A x' - b\|_2 \leq (1 + \epsilon_3) \min_{x \in \mathbb{R}^d} \|AA^\top A x - b\|_2 + \epsilon_3\|b\|_2$$

with probability $1 - \delta_3$.

Algorithm 6 Algorithm for solving the regression $\min_x \|AA^\top Ax - b\|_2$.

```
procedure FastAttentionRegression($A \in \mathbb{R}^{n \times d}, b \in \mathbb{R}^n, n \in \mathbb{Z}_+, d \in \mathbb{Z}_+, \epsilon_3 \in (0,1), \delta_3 \in (0,1)$)
    $\epsilon_1 \leftarrow 0.1\epsilon_3$
    $\delta_1 \leftarrow \delta_3/2$
    $b_2 \leftarrow$ FastLinearRegression($A \in \mathbb{R}^{n \times d}, b \in \mathbb{R}^n, n, d, \epsilon_1, \delta_1$)    ▷ $b_2 \in \mathbb{R}^d$
    $\epsilon_2 \leftarrow \epsilon_3/\kappa(A)$
    $\delta_2 \leftarrow \delta_3/2$
    $x' \leftarrow$ FastPSDRegression($A \in \mathbb{R}^{n \times d}, b_2 \in \mathbb{R}^d, n, d, \epsilon_2, \delta_2$)
    return $x'$
end procedure
```

Proof. We define OPT as

$$\mathrm{OPT} := \min_{x \in \mathbb{R}^d} \|AA^\top Ax - b\|_2.$$

First, we use Algorithm 4 to solve

$$\min_{y \in \mathbb{R}^d} \|Ay - b\|_2. \tag{33}$$

Let y_* denote the exact solution to this regression problem. By Lemma E.1, we can get $y' \in \mathbb{R}^d$ such that the following holds with probability $1 - \delta_1$,

$$\begin{aligned} \|Ay' - b\|_2 &\leq (1 + \epsilon_1) \cdot \min_{y \in \mathbb{R}^d} \|Ay - b\|_2 \\ &\leq (1 + \epsilon_1) \cdot \mathrm{OPT}, \end{aligned} \tag{34}$$

where the last step follows from the $A^\top Ax$ might not be able to do a better job than minimizer y (in terms of minimizing cost).

This step takes time

$$O((nd + d^3) \cdot \log(1/\epsilon_1) \cdot \log^2(n/\delta_1)).$$

By triangle inequality, we can show

$$\begin{aligned} \|y'\|_2 &= \|y' - y_* + y_*\|_2 \\ &\leq \|y' - y_*\|_2 + \|y_*\|_2. \end{aligned} \tag{35}$$

To bound the first term of Eq. (35), we have

$$\begin{aligned} \|y' - y_*\|_2 &\leq O(\sqrt{\epsilon_1}) \cdot \sigma_{\min}(A)^{-1} \|Ay_* - b\|_2 \\ &\leq O(\sqrt{\epsilon_1}) \cdot \sigma_{\min}(A)^{-1}\,\mathrm{OPT}, \end{aligned} \tag{36}$$

where the first step follows from Lemma D.3 and the second step follows from Eq. (34).

To bound the second term of Eq. (35), we have

$$\begin{aligned} \|y_*\|_2 &= \|A^\dagger b\|_2 \\ &\leq \|A^\dagger\| \cdot \|b\|_2 \\ &\leq \sigma_{\min}(A)^{-1} \cdot \|b\|_2, \end{aligned} \tag{37}$$

where the first step follows from $y_* = A^\dagger b$, the second step follows from $\|Ax\|_2 \leq \|A\| \|x\|_2$, and the third step follows from $\|A^\dagger\| = \sigma_{\min}(A)^{-1}$.

By plugging Eq. (36) and Eq. (37) into Eq. (35), we have

$$\|y'\|_2 \leq O(\sqrt{\epsilon_1}) \cdot \sigma_{\min}(A)^{-1} \operatorname{OPT} + \sigma_{\min}(A)^{-1} \|b\|_2. \tag{38}$$

Let $b_2 = y' \in \mathbb{R}^d$ and $x' \in \mathbb{R}^d$. Then, using Algorithm 5, we solve

$$\min_{x' \in \mathbb{R}^d} \|A^\top A x' - b_2\|_2.$$

By Lemma E.2, we find an $x' \in \mathbb{R}^d$ such that

$$\|A^\top A x' - b_2\|_2 \leq \epsilon_2 \|b_2\|_2 \tag{39}$$

holds with probability $1 - \delta_2$.

This step takes time

$$O((nd + d^3) \cdot \log(\kappa/\epsilon_2) \cdot \log^2(n/\delta_2)).$$

Correctness.

To bound $\|AA^\top A x' - b\|_2$, we have

$$\begin{aligned}
\|AA^\top A x' - b\|_2 &= \|AA^\top A x' - Ay' + Ay' - b\|_2 \\
&\leq \|AA^\top A x' - Ay'\|_2 + \|Ay' - b\|_2 \\
&\leq \|AA^\top A x' - Ay'\|_2 + (1 + \epsilon_1) \cdot \operatorname{OPT},
\end{aligned} \tag{40}$$

where the first step follows from adding and subtracting the same thing, the second step follows from triangle inequality, and the third step follows from Eq. (34). Let's consider the first term of Eq. (40). We have

$$\begin{aligned}
\|AA^\top A x' - Ay'\|_2 &= \|A(A^\top A x' - y')\|_2 \\
&\leq \|A\| \cdot \|A^\top A x' - y'\|_2 \\
&\leq \|A\| \cdot \epsilon_2 \|y'\|_2 \\
&\leq \|A\| \cdot \epsilon_2 \left(O(\sqrt{\epsilon_1}) \cdot \sigma_{\min}(A)^{-1} \operatorname{OPT} + \sigma_{\min}(A)^{-1} \|b\|_2\right) \\
&= \|A\| \cdot \epsilon_2 O(\sqrt{\epsilon_1}) \cdot \sigma_{\min}(A)^{-1} \operatorname{OPT} + \|A\| \cdot \epsilon_2 \sigma_{\min}(A)^{-1} \|b\|_2,
\end{aligned} \tag{41}$$

where the first step follows from simple algebra, the second step follows from $\|Ax\|_2 \leq \|A\| \|x\|_2$, the third step follows from Eq. (39), the fourth step follows from Eq. (38), and the last step follows from simple algebra.

Then, to bound the first term of Eq. (41), we have

$$\begin{aligned}
\|A\| \cdot \epsilon_2 O(\sqrt{\epsilon_1}) \cdot \sigma_{\min}(A)^{-1} \operatorname{OPT} &\leq \sigma_{\max}(A) \sigma_{\min}(A)^{-1} \cdot \epsilon_2 O(\sqrt{\epsilon_1}) \cdot \operatorname{OPT} \\
&= \sigma_{\max}(A) \sigma_{\min}(A)^{-1} \cdot O(\sqrt{\epsilon_1} \epsilon_2) \cdot \operatorname{OPT} \\
&= O(\sqrt{\epsilon_1} \epsilon_2) \kappa(A) \operatorname{OPT} \\
&= O(\sqrt{\epsilon_1} \epsilon_3) \operatorname{OPT},
\end{aligned} \tag{42}$$

where the first step follows from $\|A\| \leq \sigma_{\max}(A)$, the second step follows from the property of $O(\cdot)$, the third step follows from the definition of $\kappa(A)$ (see Definition A.1), and the last step follows from $\epsilon_2 = \epsilon_3 / \kappa(A)$. Similarly, to bound the second term of Eq. (41), we get

$$\begin{aligned}
\|A\| \cdot \epsilon_2 \sigma_{\min}(A)^{-1} \|b\|_2 &\leq \sigma_{\max}(A) \sigma_{\min}(A)^{-1} \cdot \epsilon_2 \|b\|_2 \\
&= \kappa(A) \cdot \epsilon_2 \|b\|_2 \\
&= \epsilon_3 \|b\|_2,
\end{aligned} \tag{43}$$

where the first step follows from $\|A\| \leq \sigma_{\max}(A)$, the second step follows from the definition of $\kappa(A)$ (see Definition A.1), and the last step follows from $\epsilon_2 = \epsilon_3 / \kappa(A)$.

Plugging Eq. (42) and Eq. (43) into Eq. (41), we get

$$\|AA^\top Ax' - Ay'\|_2 \leq O(\sqrt{\epsilon_1}\epsilon_3)\,\mathrm{OPT} + \epsilon_3\|b\|_2. \tag{44}$$

Therefore, by plugging Eq. (44) into (40), we have

$$\begin{aligned}\|AA^\top Ax' - b\|_2 &\leq O(\sqrt{\epsilon_1}\epsilon_3)\,\mathrm{OPT} + \epsilon_3\|b\|_2 + (1+\epsilon_1)\cdot \mathrm{OPT}\\ &\leq (1+\epsilon_3)\cdot \mathrm{OPT} + \epsilon_3\|b\|_2,\end{aligned}$$

where the last step follows from $O(\epsilon_1) \leq 1/10$ and $\epsilon_1 < \epsilon_3/10$. Therefore, we complete bounding $\|AA^\top Ax' - b\|_2$.

Running time.

The overall running time is

$$O((nd + d^3)\log(\kappa/\epsilon_3)\cdot \log^2(n/\delta_3))$$

Failure probability.

By taking a union over two events, the failure probability is at most $\delta_1 + \delta_2 = \delta_3$. ☐

F.2 Four Matrices

For four matrices, the algorithm will be alternating two PSD regressions.

Algorithm 7 Algorithm for solving $\min_{x\in\mathbb{R}^d} \|A^\top AA^\top Ax - b_4\|_2$.

```
procedure FourMatrices($A \in \mathbb{R}^{n\times d}, b_4 \in \mathbb{R}^d, n \in \mathbb{Z}_+, d \in \mathbb{Z}_+, \epsilon_4 \in (0,1), \delta_4 \in (0,1)$)
    $\epsilon_2 \leftarrow 0.1\epsilon_4/\kappa(A)^2$
    $\delta_2 \leftarrow \delta_4/2$
    $b_2 \leftarrow$ FastPSDRegression($A \in \mathbb{R}^{n\times d}, b_4 \in \mathbb{R}^d, n, d, \epsilon_2, \delta_2$)
    $x' \leftarrow$ FastPSDRegression($A \in \mathbb{R}^{n\times d}, b_2 \in \mathbb{R}^d, n, d, \epsilon_2, \delta_2$)
    return $x'$
end procedure
```

Lemma F.2. *Let $A \in \mathbb{R}^{n\times d}$ be a matrix and $b_4 \in \mathbb{R}^d$ be a vector. Let κ denote the condition number of A. Consider the regression problem*

$$\min_{x\in\mathbb{R}^d} \|A^\top AA^\top Ax - b_4\|_2.$$

There exists an algorithm that runs in time

$$O((nd + d^3)\cdot \log(\kappa/\epsilon_4)\cdot \log^2(n/\delta_4))$$

and outputs a vector $x' \in \mathbb{R}^d$ such that

$$\|A^\top AA^\top Ax' - b_4\|_2 \leq \epsilon_4\|b_4\|_2$$

holds with probability $1-\delta_4$.

Proof. First, we use Algorithm 5 to solve

$$\min_{y\in\mathbb{R}^d} \|A^\top Ay - b_4\|_2.$$

Let y_* denote the exact solution to this regression problem. By Lemma E.2, we get

$$\|A^\top Ay' - b_4\|_2 \leq \epsilon_2\|b_4\|_2. \tag{45}$$

This step takes time

$$O((nd + d^3) \cdot \log(\kappa/\epsilon_2) \cdot \log^2(n/\delta_2)).$$

By triangle inequality, we can show that

$$\begin{aligned}\|y'\|_2 &\leq \|y' - y_* + y_*\|_2 \\ &\leq \|y' - y_*\|_2 + \|y_*\|_2. \end{aligned} \tag{46}$$

To bound the first term of Eq. (46), we have

$$\|y' - y_*\|_2 \leq \epsilon_2 \cdot \sigma_{\min}(A)^{-2} \cdot \|b_4\|_2. \tag{47}$$

where the last step follows from Lemma D.4. To bound the second term of Eq. (46), we have

$$\begin{aligned}\|y_*\|_2 &= \|(A^\top A)^\dagger b_4\|_2 \\ &= \|(A^\top A)^\dagger\| \cdot \|b_4\|_2 \\ &\leq \sigma_{\min}(A)^{-2} \cdot \|b_4\|_2, \end{aligned} \tag{48}$$

where the first step follows from the definition of y_*, the second step follows from $\|Ax\|_2 \leq \|A\|\|x\|_2$, the last step follows from Fact A.3. Then, plugging Eq. (47) and Eq. (48) into Eq. (46), we have

$$\begin{aligned}\|y'\|_2 &\leq (\epsilon_2 + 1)\sigma_{\min}(A)^{-2}\|b_4\|_2 \\ &\leq 2\sigma_{\min}(A)^{-2}\|b_4\|_2, \end{aligned} \tag{49}$$

where the first step follows from simple algebra and the second step follows from $\epsilon_2 < 1$. Let $b_2 = y' \in \mathbb{R}^d$ and $x' \in \mathbb{R}^d$. Then, using Algorithm 5 again, we solve

$$\min_{y \in \mathbb{R}^d} \|A^\top A x' - b_2\|_2.$$

By Lemma E.2, we can find $x' \in \mathbb{R}^d$ such that

$$\|A^\top A x' - b_2\|_2 \leq \epsilon_2 \|b_2\|_2 \tag{50}$$

holds with probability $1 - \delta_2$. This step takes time

$$O((nd + d^3) \cdot \log(\kappa/\epsilon_2) \cdot \log^2(n/\delta_2)).$$

Correctness.

To bound $\|A^\top A A^\top A x' - b_4\|_2$, we have

$$\begin{aligned}\|A^\top A A^\top A x' - b_4\|_2 &= \|A^\top A A^\top A x' - A^\top A y' + A^\top A y' - b_4\|_2 \\ &\leq \|A^\top A A^\top A x' - A^\top A y'\|_2 + \|A^\top A y' - b_4\|_2 \\ &\leq \|A^\top A A^\top A x' - A^\top A y'\|_2 + \epsilon_2 \|b_4\|_2 \end{aligned} \tag{51}$$

where the first step follows from adding and subtracting the same thing, the second step follows from triangle inequality, and the third step follows from Eq. (45). Let's consider the first term of Eq. (51). We have

$$\begin{aligned}\|A^\top A A^\top A x' - A^\top A y'\|_2 &= \|A^\top A(A^\top A x' - y')\|_2 \\ &\leq \|A^\top A\| \cdot \|A^\top A x' - y'\|_2 \\ &\leq \|A^\top A\| \cdot \epsilon_2 \|y'\|_2 \\ &\leq \|A^\top A\| \cdot \epsilon_2 \cdot (2\sigma_{\min}(A)^{-2}\|b_4\|_2) \\ &\leq \sigma_{\max}(A)^2 \cdot \epsilon_2 \cdot (2\sigma_{\min}(A)^{-2}\|b_4\|_2) \\ &\leq 2\kappa(A)\epsilon_2\|b_4\|_2, \end{aligned} \tag{52}$$

where the first step follows from simple algebra, the second step follows from $\|Ax\|_2 \leq \|A\|\|x\|_2$, the third step follows from Eq. (50), the fourth step follows from Eq. (49), the fifth step follows from Fact A.3, and the last step follows from the definition of κ (see Definition A.1).

Therefore, we complete bounding $\|A^\top A A^\top A x' - b_4\|_2$.

Running time.

The total running time is

$$O((nd + d^3) \cdot \log(\kappa/\epsilon_4) \cdot \log^2(n/\delta_4)).$$

Failure probability.

By taking a union over two events, the failure probability is at most $\delta_2 + \delta_2 = \delta_4$. □

G EVEN AND ODD POWER REGRESSION: COMPLETE ALGORITHM

We provide the complete algorithm and analysis for our even and odd power regressions via induction.

G.1 Induction Hypothesis and Step

In this section, we present our induction hypothesis and prove the induction step.

Lemma G.1 (Induction Hypothesis)**.** *Let $C > 1000$ denote a sufficiently large constant. If for all $i \in [k]$, we have*

- $\|(A^\top A)^i b_i - b_0\|_2 \leq \epsilon_i \|b_0\|_2$
- $\|b_i\|_2 \leq 2\sigma_{\min}(A)^{-2i}\|b_0\|_2$
- $\epsilon_i \leq 0.5\epsilon_{i-1}$
- *The running time is* $C \cdot ((nd + d^3) \cdot k \cdot \log(\kappa(A)/\epsilon_k) \cdot \log(1/\delta_k))$
- *The failure probability is* $\delta_1 + \delta_2 + \cdots + \delta_k$

Then for $i = k+1$, we have

- $\|(A^\top A)^{k+1} b_{k+1} - b_0\|_2 \leq \epsilon_{k+1}\|b_0\|_2$
- $\|b_{k+1}\|_2 \leq 2\sigma_{\min}(A)^{-2(k+1)}\|b_0\|_2$
- $\epsilon_{k+1} \leq 0.5\epsilon_k$
- *The running time is* $C \cdot ((nd + d^3) \cdot (k+1) \cdot \log(\kappa(A)/\epsilon_{k+1}) \cdot \log(1/\delta_{k+1}))$
- *The failure probability is* $\delta_1 + \delta_2 + \cdots + \delta_{k+1}$

Proof. **Proof of Part 1.**

Running our two matrices version of PSD regression, we can obtain b_{k+1} which is the approximate solution of

$$\min_{x \in \mathbb{R}^d} \|A^\top A x - b_k\|_2$$

then we have

$$\|A^\top A b_{k+1} - b_k\|_2 \leq 0.1\epsilon_{k+1}\kappa(A)^{-2k}\|b_k\|_2 \quad (53)$$

The running time for this additional step is

$$0.1C \cdot ((nd + d^3) \cdot \log(\kappa(A)^k/\epsilon_{k+1}) \cdot \log^2(n/\delta_{k+1})).$$

We have

$$\begin{aligned}
\|(A^\top A)^{k+1} b_{k+1} - b_0\|_2 &= \|(A^\top A)^{k+1} b_{k+1} - (A^\top A)^k b_k + (A^\top A)^k b_k - b_0\|_2 \\
&\leq \|(A^\top A)^{k+1} b_{k+1} - (A^\top A)^k b_k\|_2 + \|(A^\top A)^k b_k - b_0\|_2 \\
&= \|(A^\top A)^k (A^\top A b_{k+1} - b_k)\|_2 + \|(A^\top A)^k b_k - b_0\|_2 \\
&\leq \|(A^\top A)^k\| \cdot \|A^\top A b_{k+1} - b_k\|_2 + \|(A^\top A)^k b_k - b_0\|_2 \\
&\leq \|(A^\top A)^k\| \cdot \|A^\top A b_{k+1} - b_k\|_2 + \epsilon_k \|b_0\|_2 \\
&= \sigma_{\max}(A)^{2k} \cdot \|A^\top A b_{k+1} - b_k\|_2 + \epsilon_k \|b_0\|_2 \\
&\leq \sigma_{\max}(A)^{2k} \cdot 0.1 \epsilon_{k+1} \kappa(A)^{-2k} \|b_k\|_2 + \epsilon_k \|b_0\|_2 \\
&\leq 0.2 \epsilon_{k+1} \|b_0\|_2 + \epsilon_k \|b_0\|_2 \\
&\leq \epsilon_{k+1} \|b_0\|_2, \qquad (54)
\end{aligned}$$

where the first step follows from adding and subtracting the same thing, the second step follows from the triangle inequality, the third step follows from simple algebra, the fourth step follows from $\|Ax\|_2 \leq \|A\| \|x\|_2$, the fifth step follows from the assumption in the Lemma statement, the sixth step follows from Fact A.3, the seventh step follows from Eq. (53), the eighth step follows from the assumption in the Lemma statement, and the last step follows from $\epsilon_k \leq 0.5 \epsilon_{k+1}$.

Proof of Part 2.

We have

$$\begin{aligned}
\|(A^\top A)^{k+1} b_{k+1}\|_2 &\leq \|(A^\top A)^{k+1} b_{k+1} - b_0\|_2 + \|b_0\|_2 \\
&\leq (1 + \epsilon_{k+1}) \|b_0\|_2 \\
&\leq 2 \|b_0\|_2, \qquad (55)
\end{aligned}$$

where the first step follows triangle inequality, the second step follows from Part 1, and the third step follows from $\epsilon_{k+1} \leq 1$.

Thus,

$$\begin{aligned}
\|b_{k+1}\|_2 &\leq \|((A^\top A)^{k+1})^{-1} \cdot (A^\top A)^{k+1} b_{k+1}\|_2 \\
&\leq \|((A^\top A)^{k+1})^{-1}\| \cdot \|(A^\top A)^{k+1} b_{k+1}\|_2 \\
&\leq \|((A^\top A)^{k+1})^{-1}\| \cdot 2 \|b_0\| \\
&\leq 2 \sigma_{\min}(A)^{-2(k+1)} \|b_0\|_2,
\end{aligned}$$

where the first step follows from $((A^\top A)^{k+1})^{-1} \cdot (A^\top A)^{k+1} = I$, the second step follows from $\|Ax\|_2 = \|A\| \|x\|_2$, the third step follows from Eq. (55), and the last step follows from Fact A.3.

Proof of Part 3.

We can choose ϵ to satisfy these conditions. Thus, it automatically holds.

Proof of Part 4.

The proof follows by adding the time from the previous step and this step.

Proof of Part 5.

It follows from taking a union bound. ☐

G.2 Even Number of Matrices Regression

In this section, we present and prove our main result for even power regression.

Theorem G.2. *Let $A \in \mathbb{R}^{n \times d}$ be a matrix and $b \in \mathbb{R}^d$ be a vector. Let κ denote the condition number of A. Consider the regression problem*

$$\min_{x \in \mathbb{R}^d} \|(A^\top A)^j x - b\|_2.$$

Let $\epsilon_{\mathrm{final}} \in (0, 0.1)$ *denote the accuracy parameter. Let* $\delta_{\mathrm{final}} \in (0, 0.1)$ *denote the failure probability. There exists an algorithm that runs in time*

$$O((nd + d^3) \cdot j \cdot \log(\kappa/\epsilon_{\mathrm{final}}) \cdot \log^2(jn/\delta_{\mathrm{final}}))$$

and outputs a vector $x' \in \mathbb{R}^d$ *such that*

$$\|(A^\top A)^j x' - b\|_2 \leq \epsilon_{\mathrm{final}} \|b\|_2$$

holds with probability $1 - \delta_{\mathrm{final}}$.

Proof. The proof is a standard induction.

Base case:

When $j = 0$, we have

$$\min_{x \in \mathbb{R}^d} \|A^\top A (A^\top A)^0 x - b\|_2 = \min_{x \in \mathbb{R}^d} \|A^\top A x - b\|_2.$$

The base case follows from Lemma E.2.

Inductive case:

This follows from Lemma G.1. □

G.3 Odd Number of Matrices Regression

In this section, we analyze the odd power algorithm with its correctness and running time.

Theorem G.3. *Let* $A \in \mathbb{R}^{n \times d}$ *be a matrix and* $b \in \mathbb{R}^n$ *be a vector. Let* κ *denote the condition number of* A*. Consider the regression problem*

$$\min_{x \in \mathbb{R}^d} \|A(A^\top A)^j x - b\|_2.$$

Let $\epsilon_{\mathrm{final}} \in (0, 0.1)$ *denote the accuracy parameter. Let* $\delta_{\mathrm{final}} \in (0, 0.1)$ *denote the failure probability. There exists an algorithm that runs in time*

$$O((nd + d^3) \cdot j \cdot \log(\kappa/\epsilon_{\mathrm{final}}) \cdot \log^2(jn/\delta_{\mathrm{final}}))$$

and outputs a vector $x' \in \mathbb{R}^d$ *such that*

$$\|A(A^\top A)^j x' - b\|_2 \leq \epsilon_{\mathrm{final}} \|b\|_2$$

holds with probability $1 - \delta_{\mathrm{final}}$.

Proof. For convenience, we use b_{odd} to denote b. We define OPT as

$$\mathrm{OPT} := \min_{x \in \mathbb{R}^d} \|A(A^\top A)^j x - b_{\mathrm{odd}}\|_2.$$

First, we use Algorithm 4 to solve

$$\min_{y \in \mathbb{R}^d} \|Ay - b_{\mathrm{odd}}\|_2. \tag{56}$$

By Lemma E.1, we can get $y' \in \mathbb{R}^d$ such that the following holds with probability $1 - \delta_1$,

$$\begin{aligned} \|Ay' - b_{\mathrm{odd}}\|_2 &\leq (1 + \epsilon_1) \cdot \min_{y \in \mathbb{R}^d} \|Ay - b_{\mathrm{odd}}\|_2 \\ &\leq (1 + \epsilon_1) \cdot \mathrm{OPT}, \end{aligned} \tag{57}$$

where the last step follows from the $A^\top Ax$ might not be able to do a better job than minimizer y (in terms of minimizing cost). This step takes time

$$O((nd + d^3) \cdot \log(1/\epsilon_1) \cdot \log^2(n/\delta_1)).$$

By triangle inequality, we can show

$$\begin{aligned} \|y'\|_2 &= \|y' - y_* + y_*\|_2 \\ &\leq \|y' - y_*\|_2 + \|y_*\|_2. \end{aligned} \tag{58}$$

To bound the first term of Eq. (58), we have

$$\begin{aligned} \|y' - y_*\|_2 &\leq O(\sqrt{\epsilon_1}) \cdot \sigma_{\min}(A)^{-1} \|Ay_* - b_{\mathrm{odd}}\|_2 \\ &\leq O(\sqrt{\epsilon_1}) \cdot \sigma_{\min}(A)^{-1} \,\mathrm{OPT}, \end{aligned} \tag{59}$$

where the first step follows from Lemma D.3 and the second step follows from Eq. (57).

To bound the second term of Eq. (58), we have

$$\begin{aligned} \|y_*\|_2 &= \|A^\dagger b_{\mathrm{odd}}\|_2 \\ &\leq \|A^\dagger\| \cdot \|b_{\mathrm{odd}}\|_2 \\ &\leq \sigma_{\min}(A)^{-1} \cdot \|b_{\mathrm{odd}}\|_2, \end{aligned} \tag{60}$$

where the first step follows from $y_* = A^\dagger b_{\mathrm{odd}}$, the second step follows from $\|Ax\|_2 \leq \|A\| \|x\|_2$, and the third step follows from $\|A^\dagger\| = \sigma_{\min}(A)^{-1}$.

By plugging Eq. (59) and Eq. (60) into Eq. (58), we have

$$\|y'\|_2 \leq O(\sqrt{\epsilon_1}) \cdot \sigma_{\min}(A)^{-1} \,\mathrm{OPT} + \sigma_{\min}(A)^{-1} \|b_{\mathrm{odd}}\|_2. \tag{61}$$

Let $b_{\mathrm{even}} = y' \in \mathbb{R}^d$ and $x' \in \mathbb{R}^d$. Then, using Algorithm 1, we solve

$$\min_{x' \in \mathbb{R}^d} \|(A^\top A)^j x' - b_{\mathrm{even}}\|_2.$$

By Lemma E.2, we find an $x' \in \mathbb{R}^d$ such that

$$\|(A^\top A)^j x' - b_{\mathrm{even}}\|_2 \leq \epsilon_{\mathrm{even}} \|b_{\mathrm{even}}\|_2 \tag{62}$$

holds with probability $1 - \delta_2$.

This step takes time

$$O((nd + d^3) \cdot j \cdot \log(\kappa/\epsilon_{\mathrm{even}}) \cdot \log^2(n/\delta_{\mathrm{even}})).$$

Correctness.

To bound $\|A(A^\top A)^j x' - b_{\mathrm{odd}}\|_2$, we have

$$\begin{aligned} \|A(A^\top A)^j x' - b_{\mathrm{odd}}\|_2 &= \|A(A^\top A)^j x' - Ay' + Ay' - b_{\mathrm{odd}}\|_2 \\ &\leq \|A(A^\top A)^j x' - Ay'\|_2 + \|Ay' - b_{\mathrm{odd}}\|_2 \\ &\leq \|A(A^\top A)^j x' - Ay'\|_2 + (1 + \epsilon_1) \cdot \mathrm{OPT}, \end{aligned} \tag{63}$$

where the first step follows from adding and subtracting the same thing, the second step follows from the triangle inequality, and the third step follows from Eq. (57). Let's consider the first term of Eq. (63). We have

$$\|A(A^\top A)^j x' - Ay'\|_2 = \|A((A^\top A)^j x' - y')\|_2$$

$$
\begin{aligned}
&\leq \|A\| \cdot \|(A^\top A)^j x' - y'\|_2 \\
&\leq \|A\| \cdot \epsilon_{\mathrm{even}} \|y'\|_2 \\
&\leq \|A\| \cdot \epsilon_{\mathrm{even}} \left(O(\sqrt{\epsilon_1}) \cdot \sigma_{\min}(A)^{-1} \operatorname{OPT} + \sigma_{\min}(A)^{-1} \|b_{\mathrm{odd}}\|_2 \right) \\
&= \|A\| \cdot \epsilon_{\mathrm{even}} O(\sqrt{\epsilon_1}) \cdot \sigma_{\min}(A)^{-1} \operatorname{OPT} + \|A\| \cdot \epsilon_{\mathrm{even}} \sigma_{\min}(A)^{-1} \|b_{\mathrm{odd}}\|_2, \qquad (64)
\end{aligned}
$$

where the first step follows from simple algebra, the second step follows from $\|Ax\|_2 \leq \|A\| \|x\|_2$, the third step follows from Eq. (62), the fourth step follows from Eq. (61), and the last step follows from simple algebra.

Then, to bound the first term of Eq. (64), we have

$$
\begin{aligned}
\|A\| \cdot \epsilon_{\mathrm{even}} O(\sqrt{\epsilon_1}) \cdot \sigma_{\min}(A)^{-1} \operatorname{OPT} &\leq \sigma_{\max}(A) \sigma_{\min}(A)^{-1} \cdot \epsilon_{\mathrm{even}} O(\sqrt{\epsilon_1}) \cdot \operatorname{OPT} \\
&= \sigma_{\max}(A) \sigma_{\min}(A)^{-1} \cdot O(\sqrt{\epsilon_1} \epsilon_{\mathrm{even}}) \cdot \operatorname{OPT} \\
&= O(\sqrt{\epsilon_1} \epsilon_{\mathrm{even}}) \kappa(A) \operatorname{OPT} \\
&= O(\sqrt{\epsilon_1} \epsilon_{\mathrm{final}}) \operatorname{OPT}, \qquad (65)
\end{aligned}
$$

where the first step follows from $\|A\| \leq \sigma_{\max}(A)$, the second step follows from the property of $O(\cdot)$, the third step follows from the definition of $\kappa(A)$ (see Definition A.1), and the last step follows from $\epsilon_{\mathrm{even}} = \epsilon_{\mathrm{final}} / \kappa(A)$. Similarly, to bound the second term of Eq. (64), we get

$$
\begin{aligned}
\|A\| \cdot \epsilon_{\mathrm{even}} \sigma_{\min}(A)^{-1} \|b_{\mathrm{odd}}\|_2 &\leq \sigma_{\max}(A) \sigma_{\min}(A)^{-1} \cdot \epsilon_{\mathrm{even}} \|b_{\mathrm{odd}}\|_2 \\
&= \kappa(A) \cdot \epsilon_{\mathrm{even}} \|b_{\mathrm{odd}}\|_2 \\
&= \epsilon_{\mathrm{final}} \|b_{\mathrm{odd}}\|_2, \qquad (66)
\end{aligned}
$$

where the first step follows from $\|A\| \leq \sigma_{\max}(A)$, the second step follows from the definition of $\kappa(A)$ (see Definition A.1), and the last step follows from $\epsilon_{\mathrm{even}} = \epsilon_{\mathrm{final}} / \kappa(A)$.

Plugging Eq. (65) and Eq. (66) into Eq. (64), we get

$$
\|A(A^\top A)^j x' - Ay'\|_2 \leq O(\sqrt{\epsilon_1} \epsilon_{\mathrm{final}}) \operatorname{OPT} + \epsilon_{\mathrm{final}} \|b_{\mathrm{odd}}\|_2. \qquad (67)
$$

Therefore, by plugging Eq. (67) into (63), we have

$$
\begin{aligned}
\|A(A^\top A)^j x' - b_{\mathrm{odd}}\|_2 &\leq O(\sqrt{\epsilon_1} \epsilon_{\mathrm{final}}) \operatorname{OPT} + \epsilon_{\mathrm{final}} \|b_{\mathrm{odd}}\|_2 + (1 + \epsilon_1) \cdot \operatorname{OPT} \\
&\leq (1 + \epsilon_{\mathrm{final}}) \cdot \operatorname{OPT} + \epsilon_{\mathrm{final}} \|b_{\mathrm{odd}}\|_2,
\end{aligned}
$$

where the last step follows from $O(\epsilon_1) \leq 1/10$ and $\epsilon_1 < \epsilon_{\mathrm{final}}/10$. Therefore, we complete bounding $\|A(A^\top A)^j x' - b_{\mathrm{odd}}\|_2$.

Running time.

The overall running time is

$$
O((nd + d^3) \cdot j \cdot \log(\kappa/\epsilon_{\mathrm{final}}) \cdot \log^2(n/\delta_{\mathrm{final}})).
$$

Failure probability.

By taking a union over two events, the failure probability is at most $\delta_1 + \delta_{\mathrm{even}} = \delta_{\mathrm{final}}$. □

H ATTENTION KERNEL REGRESSION

In this section, we demonstrate algorithms for solving attention kernel regressions, inspired by algorithms for Gaussian kernel regression. We develop novel algorithms when the sketch dimension is smaller than n.

H.1 Fast Subspace Embedding for Tensors

We need the following tools from prior results. The first result regards a class of subspace embeddings whenever the target matrix has certain tensoring structure.

Theorem H.1 (Theorem 3 in Ahle et al. (2020))**.** *For every positive integers p, d, n, every $\epsilon, \beta > 0$, there exists a distribution on linear sketches $\Pi^p \in \mathbb{R}^{m \times d^p}$ such that*

1. *If*

$$m = \widetilde{\Omega}(p\beta^2\epsilon^{-2}),$$

then Π^p is an $(n, d^p, \epsilon, 1/\operatorname{poly}(n))$-$\mathsf{SE}$, where β is an upper bound on the rank of the target matrix.

2. *If*

$$m = \widetilde{\Omega}(p\epsilon^{-2}),$$

then Π^p has the $\mathsf{FAMP}(n, \epsilon, 1/\operatorname{poly}(n))$ (Definition C.5).

Moreover, in the setting of 1., for any $X \in \mathbb{R}^{d \times n}$, if $A \in \mathbb{R}^{d^p \times n}$ is the matrix whose columns are obtained by a p-fold self-tensoring of each column of X and $\operatorname{rank}(A) = \beta$, then the matrix $\Pi^p A$ can be computed in time

$$\widetilde{O}(pnm + p^{3/2}\beta\epsilon^{-1}\operatorname{nnz}(X)).$$

The next result provides a formal statement on the quality of SRHT matrix.

Lemma H.2 (Theorem 2.4 in Woodruff (2014))**.** *Let T be an SRHT matrix defined in Definition C.1. If*

$$m = O(\epsilon^{-2}\beta\log(nd/\delta)),$$

then T is an (n, d, ϵ, δ)-SE where β is an upper bound on the rank of the target matrix.

We review an algorithm due to Song et al. (2021a).

Algorithm 8 Algorithm of Song et al. (2021a).

```
procedure TensorSketchViaLimRand($x \in \mathbb{R}^d$, $p \in (1, \infty)$, $S \in \mathbb{R}^{m \times m^2}$, $T \in \mathbb{R}^{m \times d}$)    ▷ Theorem H.4 and H.5
    $q \leftarrow 2^{\lfloor \log_2 p \rfloor}$
    $w_0 \leftarrow Tx$    ▷ $T$ can be SRHT (Definition C.1)
    for $l = 1$ to $\log_2 q$ do
        $w_l \leftarrow S(w_{l-1}^{\otimes 2})$    ▷ $S$ can be TensorSRHT (Definition C.2)
    end for
    ▷ Let $b$ be the binary representation of $p$, and let $E = \{i : b_i = 1, i \in \{0, \ldots, \log_2 p\}\}$
    ▷ Let $z = w_j$, where $j$ is the lowest bit of $b$ where $b_j = 1$
    for $i$ in $E \setminus \{j\}$ do
        $z = S(z \otimes w_i)$
    end for
    return $z$    ▷ $z \in \mathbb{R}^m$
end procedure
```

Before stating the guarantees of this algorithm, we need some auxiliary definitions.

Definition H.3. *Let $S \in \mathbb{R}^{m^2} \to \mathbb{R}^m$ and $T : \mathbb{R}^d \to \mathbb{R}^m$ be base sketches. Let $X \in \mathbb{R}^{d \times n}$ be an input matrix. We define $\mathcal{Z}(S, T, X)$ to be the matrix for which we apply Algorithm 8 on each column of X, with base sketches S and T.*

Theorem H.4 (Theorem 4.8 in Song et al. (2021a))**.** *Let*

$$S : \mathbb{R}^{m^2} \to \mathbb{R}^m$$

be an (n, d, ϵ, δ)-SE for degree-2 tensors and

$$T : \mathbb{R}^d \to \mathbb{R}^m$$

be an (n, d, ϵ, δ)-SE. Let p be a positive integer, $Z = \mathcal{Z}(S, T, X)$ be the matrix as defined in Definition H.3. Then for any $y \in \mathbb{R}^n$, we have

$$(1 - \epsilon)^{3p} \cdot \|X^{\otimes p}y\|_2 \leq \|Zy\|_2 \leq (1 + \epsilon)^{3p} \cdot \|X^{\otimes p}y\|_2.$$

The next theorem states Algorithm 8 can be quickly applied to a structured matrix. We slightly modify their guarantee to depend on the rank of the matrix.

Theorem H.5. *Let $p \in \mathbb{Z}_+$ and let $\epsilon, \delta \in (0,1)$. For every $X \in \mathbb{R}^{d \times n}$, there exists a distribution over oblivious linear sketches $\Pi : \mathbb{R}^{d^p} \to \mathbb{R}^m$ such that if*

$$m = \Theta(\epsilon^{-2} \beta p^2),$$

where β is an upper bound on the rank of $X^{\otimes p}$. Then, we have

$$(\Pi X^{\otimes^p})^\top \Pi X^{\otimes^p} \approx_\epsilon (X^{\otimes^p})^\top X^{\otimes^p}.$$

Moreover, using Algorithm 8,

$$\Pi X^{\otimes^p} = \mathcal{Z}(S, T, X)$$

can be computed in time

$$O(nd + \epsilon^{-2} n \beta p^2).$$

Utilize this result, we prove a result regarding exponential kernel.

Theorem H.6. *For every $r > 0$, every positive integers n, d, and every $X \in \mathbb{R}^{d \times n}$ such that $\|x_i\|_2 \leq r$ for all $i \in [n]$, where x_i is the i-th column of X, suppose $K \in \mathbb{R}^{n \times n}$ is the attention kernel matrix i.e.,*

$$K_{j,k} = e^{\langle x_j, x_k \rangle}$$

for all $j, k \in [n]$. There exists an algorithm which computes $W_g(X) \in \mathbb{R}^{m \times n}$ in time

$$\widetilde{O}(q^3 \epsilon^{-2} n \beta + nd \log(nd/\epsilon\delta))$$

such that for every $\epsilon > 0$,

$$\Pr_{W_g}[(1-\epsilon)K \preceq (W_g(X))^\top W_g(X) \preceq (1+\epsilon)K] \geq 1 - \frac{1}{\mathrm{poly}(n)},$$

where

$$m = \widetilde{\Theta}(q^3 \beta / \epsilon^2)$$

and

$$q = \Theta(r^2 + \log(n/\epsilon))$$

and β is an upper bound on the rank of the following sequence of matrices:

$$\{X, X^{\otimes 2}, \ldots, X^{\otimes q}\}$$

Proof. Note that the Taylor series expansion for kernel K gives

$$K = \sum_{l=0}^{\infty} \frac{(X^{\otimes l})^\top X^{\otimes l}}{l!}.$$

Let

$$q = C \cdot (r^2 + \log(n/\epsilon))$$

for a sufficiently large constant C.

Let

$$Q = \sum_{l=0}^{q} \frac{(X^{\otimes l})^\top X^{\otimes l}}{l!}$$

be the first q terms of K. By triangle inequality, we have:

$$\begin{aligned} \|K - Q\| \leq & \sum_{l>q} \| \frac{(X^{\otimes l})^\top X^{\otimes l}}{l!} \| \\ \leq & \sum_{l>q} \| \frac{(X^{\otimes l})^\top X^{\otimes l}}{l!} \|_F \\ \leq & \sum_{l>q} \frac{n \cdot r^{2l}}{l!} \\ \leq & \frac{\epsilon}{2} \cdot \|K\|, \end{aligned}$$

where the first step follows from the definition of Q, the second step follows from $\|A\| \leq \|A\|_F$ for all matrix A, the third step follows from the upper bounding the Frobenious norm, and the last step follows from the choice of q and $\|K\| \leq n \exp(r)$.

For each term $(X^{\otimes l})^\top X^{\otimes l}$ in Q, we run Algorithm 8 to approximate $X^{\otimes l}$.

Let $Z_l \in \mathbb{R}^{m_l \times n}$ be the resulting matrix $\mathcal{Z}(S, T, X)$, where

$$m_l = \Omega(\epsilon^{-2} \beta l^2 \log^2(nd/\epsilon\delta) \log(n/\delta)).$$

Then by Theorem H.5, we get

$$(1 - \epsilon/2)(X^{\otimes l})^\top X^{\otimes l} \preceq (\Pi^l X^{\otimes l})^\top \Pi^l X^{\otimes l} \preceq (1 + \epsilon/2)(X^{\otimes l})^\top X^{\otimes l} \tag{68}$$

with probability at least $1 - \frac{\delta}{q+1}$.

Moreover, Z_l can be computed in time

$$O(\epsilon^{-2} n \beta l^2 \cdot \log^2(nd/\epsilon\delta) \cdot \log(\frac{n}{\delta})).$$

Our algorithm will simply compute Z_l from $l = 0$ to q, normalize each Z_l by $\frac{1}{\sqrt{l!}}$.

More precisely, the approximation $W_g(X)$ will be

$$W_g(X) = (\oplus_{l=0}^{q} \frac{Z_l}{\sqrt{l!}}),$$

Notice $W_g(X) \in \mathbb{R}^{m \times n}$.

The following holds for $W_g(X)^\top W_g(X)$:

$$W_g(X)^\top W_g(X) = \sum_{l=0}^{q} \frac{Z_l^\top Z_l}{l!}.$$

By combining terms in Eq. (68) and using a union bound over all $0 \leq l \leq q$, we obtain that with probability at least $1 - \delta$, we have the following:

$$(1 - \epsilon/2) \cdot Q \preceq W_g(X)^\top W_g(X) \preceq (1 + \epsilon/2) \cdot Q.$$

Thus, we conclude that

$$(1 - \epsilon) \cdot K \preceq W_g(X)^\top W_g(X) \preceq (1 + \epsilon) \cdot K.$$

Note the target dimension of W_g is

$$\begin{aligned} m = & \sum_{i=0}^{q} m_i \\ = & \Omega(\epsilon^{-2} n q^3 \cdot \log^2(nd/\epsilon\delta) \cdot \log(n/\delta)), \end{aligned}$$

where the first step follows from the construction of $W_g(X)$ and the second step follows from simple algebra.

Also, by Theorem H.5, the time to compute $W_g(X)$ is

$$\begin{aligned} t = & \sum_{j=0}^{q} t_j \\ = & O(\epsilon^{-2} n \beta q^3 \cdot \log^2(nd/\epsilon\delta) \cdot \log(n/\delta)). \end{aligned}$$

Notice that we will have to add the term $nd \log(nd/\epsilon\delta)$ due to line 2 of Algorithm 8 when applying the SRHT to X. However, we only need to perform this operation once for the term with the highest degree or for the terms with lower degree that can be formed by combining nodes computed with the highest degree. Therefore, the final runtime is:

$$O(\epsilon^{-2} n \beta q^3 \cdot \log^2(nd/\epsilon\delta) \cdot \log(n/\delta) + nd \log(nd/\epsilon\delta)).$$

$\square$

H.2 Main Result

Theorem H.7 (Formal version of Theorem 1.6)**.** *Let $G \in \mathbb{R}^{n \times n}$ be the Attention kernel matrix (Definition B.2) for $X \in \mathbb{R}^{d \times n}$ and factor $G = Z^\top Z$. Let κ denote the condition number of Z. If we assume that for all $i \in [n]$, $\|x_i\|_2 \leq 1$, then Algorithm 3, with probability at least $1 - \delta$, computes an $\widehat{x}$ satisfying the following:*

$$\|G\widehat{x} - y\|_2 \leq \epsilon \|y\|_2.$$

Moreover, let

$$m = O(\epsilon^{-2} \beta \log^2(nd/\epsilon\delta) \log(n/\delta)),$$

where β is an upper bound on the rank of the following sequence of matrices:

$$\{X, X^{\otimes 2}, \ldots, X^{\otimes q}\},$$

the vector $\widehat{x} \in \mathbb{R}^n$ can be computed in time

$$O(mn + \epsilon^{-2} nd + m^3).$$

Proof. Throughout the proof, we will set $\widehat{\epsilon} = \epsilon/4$. If $m \geq n$, then we can invoke Theorem 6.6 of Song et al. (2021a), which readily provides the desired result. So we assume $m < n$. By Theorem H.6, we can compute an ϵ-approximation to Z and $W_g(X)$ in time

$$O(\epsilon^{-2} n \beta \cdot \mathrm{poly}(\log(nd/\epsilon\delta)) + nd \log(nd/\epsilon\delta))$$

If we solve the problem:

$$\min_{x \in \mathbb{R}^n} \|W_g(X)^\top W_g(X) x - y\|_2$$

with solution $\widehat{x}$, then we have

$$\|W_g(X)^\top W_g(X) \widehat{x} - y\|_2 \leq (1 + \widehat{\epsilon}) \min_{x \in \mathbb{R}^n} \|Z^\top Z x - y\|_2.$$

This means the optimal solution for the sketched problem gives a $\widehat{\epsilon}$-approximation to the optimal solution to the original problem. We will now show that Algorithm 3 computes the desired solution. By Lemma H.2, with probability at least $1-\delta$, for any $x \in \mathbb{R}^m$, we have:

$$\|\underbrace{S}_{s\times n}\underbrace{W_g(X)^\top}_{n\times m} x\|_2 = (1\pm\epsilon_0)\cdot\|\underbrace{W_g(X)^\top}_{n\times m} x\|_2.$$

Note that from Algorithm 3, we have

$$\underbrace{S}_{s\times n}\underbrace{W_g(X)^\top}_{n\times m} = \underbrace{U}_{s\times m}\underbrace{\Sigma}_{m\times m}\underbrace{V^\top}_{m\times m} \tag{69}$$

$$\underbrace{R}_{s\times m} = \underbrace{U}_{s\times m}\underbrace{\Sigma^{-2}}_{m\times m} \tag{70}$$

We know that

$$\begin{aligned}\kappa(R^\top S W_g(X)^\top) &= \kappa(\Sigma^{-2}U^\top U\Sigma V^\top)\\ &= \kappa(\Sigma^{-1}V^\top)\\ &= \kappa(\Sigma^{-1})\\ &= \kappa(\Sigma)\\ &\leq 2\kappa(W_g(X)),\end{aligned} \tag{71}$$

where the first step follows from Eq. (69) and Eq. (70), the second step follows from $U^\top U = I$, the third step follows from Fact A.3 and V is an orthonormal basis, the fourth step follows from the Fact A.3, and the last step follows from S is a $(n, d, \epsilon_0, \delta)$-$\mathsf{SE}$ (see Lemma H.2).

We have

$$\begin{aligned}\kappa(R^\top S) &\leq \kappa(R^\top S W_g(X)^\top)\cdot\kappa(W_g(X))\\ &\leq 2\kappa(W_g(X))^2\end{aligned} \tag{72}$$

where the first step follows from Fact A.3, the second step follows from Eq. (71).

For any unit vector $x \in \mathbb{R}^n$, from the above formulation, we know that

$$\begin{aligned}\|SW_g(X)^\top W_g(X)S^\top Rx\|_2 &= \|U\Sigma V^\top V\Sigma U^\top U\Sigma^{-2}x\|_2\\ &= \|U\Sigma\Sigma\Sigma^{-2}x\|_2\\ &= \|Ux\|_2\\ &= \|x\|_2\\ &= 1,\end{aligned} \tag{73}$$

where the first step follows from Eq. (69), the second step follows from $V^\top V = I$ and $U^\top U = I$, the third step follows from simple algebra, the fourth step follows Fact A.3, the last step follows from $\|x\|_2 = 1$.

We need to obtain a bound on $\|W_g(X)^\top W_g(X)S^\top Rx\|_2$:

$$\begin{aligned}\|W_g(X)^\top W_g(X)S^\top Rx\|_2 &= (1\pm\epsilon_0)^{-1}\cdot\|SW_g(X)^\top W_g(X)S^\top Rx\|_2\\ &= (1\pm\epsilon_0)^{-1}\\ &= 1\pm 2\epsilon_0,\end{aligned}$$

where first step follows from S is $(n, d, \epsilon_0, \delta)$-$\mathsf{SE}$ (see Lemma H.2) of $W_g(X)^\top$, the second step follows from Eq. (73), the third step follows from $\epsilon_0 \in (0, 0.1)$.

Now, pick $\epsilon_0 = 0.1$ and solve the following regression problem:

$$\min_{z\in\mathbb{R}^n}\|\underbrace{R^\top}_{m\times s}\underbrace{S}_{s\times n}\underbrace{W_g(X)^\top}_{n\times m}\underbrace{W_g(X)}_{m\times n}\underbrace{S^\top}_{n\times s}\underbrace{R}_{s\times m} z - \underbrace{R^\top}_{m\times s}\underbrace{S}_{s\times n} y\|_2. \tag{74}$$

For convenience, we define $\Phi \in \mathbb{R}^{m \times m}$ as follows

$$\Phi := R^\top S W_g(X)^\top W_g(X) S^\top R.$$

Notice that Algorithm 3 implements gradient descent. Using Lemma D.2, after $t = \log(1/\widehat{\epsilon})$ iterations, we have

$$\|\Phi \cdot (z_t - z^*)\|_2 \leq \widehat{\epsilon} \cdot \|\Phi \cdot (z_0 - z^*)\|_2, \tag{75}$$

where

$$z^* = \Phi^{-1} R^\top S y \tag{76}$$

is the optimal solution to Eq. (74). Define

$$x_t := S^\top R z_t \tag{77}$$

We will show the following for x_t (in Eq. (77)):

$$\|W_g(X)^\top W_g(X) x_t - y\|_2 \leq \kappa \widehat{\epsilon} \|y\|_2.$$

We get

$$\begin{aligned} \|R^\top S W_g(X)^\top W_g(X) x_t - R^\top S y\|_2 &= \|\Phi \cdot z_t - R^\top S y\|_2 \\ &= \|\Phi \cdot (z_t - z^*)\|_2 \\ &\leq \widehat{\epsilon} \cdot \|\Phi(z_0 - z^*)\|_2 \\ &= \widehat{\epsilon} \cdot \|\Phi z^*\|_2 \\ &= \widehat{\epsilon} \cdot \|R^\top S y\|_2 \\ &\leq \widehat{\epsilon} \cdot \sigma_{\max}(R^\top S) \cdot \|y\|_2, \end{aligned} \tag{78}$$

where the first step follows follows from definition of Φ and Eq. (77), the second step follows from Eq. (76), the third step follows from Eq. (75), the fourth step follows from $z_0 = \mathbf{0}_m$, the fifth step follows from the definition of z^*, the last step follows from Fact A.3.

On the other hand,

$$\begin{aligned} \|R^\top S W_g(X)^\top W_g(X) x_t - R^\top S y\|_2 &= \|R^\top S (W_g(X)^\top W_g(X) x_t - y)\|_2 \\ &\geq \sigma_{\min}(R^\top S) \cdot \|W_g(X)^\top W_g(X) x_t - y\|_2, \end{aligned} \tag{79}$$

where the first step follows from simple algebra and the second step follows from Fact A.3.

Putting everything together, we get

$$\begin{aligned} \|W_g(X)^\top W_g(X) x_t - y\|_2^2 &\leq \widehat{\epsilon} \kappa(R^\top S) \|y\|_2 \\ &\leq 2\kappa(W_g(X))^2 \widehat{\epsilon} \|y\|_2, \end{aligned}$$

where the first step follows from Eq. (78) and Eq. (79), the second step follows from Eq. (72).

This means by setting the number of iterations to

$$t = \log(\kappa(W_g(X))/\epsilon),$$

we obtain

$$\|W_g(X)^\top W_g(X) x_t - y\|_2 \leq 2\widehat{\epsilon} \|y\|_2. \tag{80}$$

Now, recall that for any $x, y \in \mathbb{R}^n$, we have,

$$\|W_g(X)^\top W_g(X) x - y\|_2 \leq (1 + \widehat{\epsilon}) \|Z^\top Z x - y\|_2.$$

As a consequence,

$$\begin{aligned}\|Z^\top Z x_t - y\|_2 &\leq (1+\widehat{\epsilon})\|W_g(X)^\top W_g(X) x_t - y\|_2 \\ &\leq (1+\widehat{\epsilon}) 2\widehat{\epsilon}\|y\|_2 \\ &\leq \epsilon\|y\|_2,\end{aligned}$$

the second step follows from Eq. (80), and the third step follows from $\widehat{\epsilon} \leq 0.1\epsilon$. Now we analyze the runtime.

- Computing $W_g(X)$, by Theorem H.6, takes time

$$\epsilon^{-2} n\beta \cdot \mathrm{poly}(\log(nd/\epsilon\delta)) + nd\log(nd/\epsilon\delta).$$

- Applying S to $W_g(X)$, using the FFT algorithm, takes time

$$\epsilon^{-2} n\beta \cdot \mathrm{poly}(\log(nd/\epsilon\delta)).$$

- The SVD of $SW_g(X)^\top$ can be computed in time

$$(\epsilon^{-2}\beta)^\omega \cdot \mathrm{poly}(\log(nd/\epsilon/\delta))$$

The cost of each iteration is bounded by the cost of taking a matrix-vector product, which is at most $\widetilde{O}(n\beta/\epsilon^2)$, and there are $O(\log(\kappa/\epsilon))$ iterations in total. Thus, we obtain a final runtime of

$$\epsilon^{-2} n\beta \cdot \mathrm{poly}(\log(nd/\epsilon\delta)) \cdot \log(\kappa/\epsilon) + (nd + (\epsilon^{-2}\beta)^\omega) \cdot \log(nd/\epsilon\delta).$$

$\square$