Fast Sketching of Polynomial Kernels of Polynomial Degree

Zhao Song 1, David P. Woodruff 2, Zheng Yu 3, Lichen Zhang 4

Abstract

Kernel methods are fundamental in machine learning, and faster algorithms for kernel approximation provide direct speedups for many core tasks in machine learning. The polynomial kernel is especially important as other kernels can often be approximated by the polynomial kernel via a Taylor series expansion. Recent techniques in oblivious sketching reduce the dependence in the running time on the degree q of the polynomial kernel from exponential to polynomial, which is useful for the Gaussian kernel, for which q can be chosen to be polylogarithmic. However, for more slowly growing kernels, such as the neural tangent and arc-cosine kernels, q needs to be polynomial, and previous work incurs a polynomial factor slowdown in the running time. We give a new oblivious sketch which greatly improves upon this running time, by removing the dependence on q in the leading order term. Combined with a novel sampling scheme, we give the fastest algorithms for approximating a large family of slow-growing kernels.

1. Introduction

Kernel methods are a powerful tool for solving non-parametric learning problems, such as kernel regression, support vector machines (SVM), principal component analysis (PCA), and many others. A typical burden for kernel methods is that they suffer from scalability, since computing a kernel matrix requires computing a quadratic (in the number of input points) number of entries in the matrix. A direction that has received less attention but still of particular interest is the regime where the dimension d of the data points is large. Typically, applying the kernel function to each pair of data points takes $O(d)$ time. This is especially undesirable in applications for natural language processing (Drikvandi & Lawal, 2020) and computational biology (Teodoro et al., 2002), where $d$ can be as large as $\text{poly}(n)$, with $n$ being the number of data points. To compute the kernel matrix, the algorithm does have to read the $d \times n$ input matrix. Therefore, algorithms that have a nearly linear dependence on $nd$ are of particular interest.

To accelerate the computation of kernel matrices from the naive $O(n^2d)$ time algorithm, a lot of work has focused on finding a good approximation to a kernel matrix efficiently (Rahimi & Recht, 2007; Alaoui & Mahoney, 2015; Musco & Musco, 2017; Ahle et al., 2020; Woodruff & Zandieh, 2020). All of these methods make use of randomized algorithmic primitives such as sampling or sketching. Roughly speaking, the idea is to randomly generate a “sketching matrix” with a small number of rows, multiply the sketching matrix with the input matrix, and show that the resulting matrix approximately preserves the length of vectors in the row or column space of the original matrix.

The polynomial kernel is of interest, since any kernel can be written as a sum of polynomial kernels through a Taylor expansion. The polynomial kernel is especially important as other kernels can be approximated by the polynomial kernel. Recent works of (Ahle et al., 2020; Woodruff & Zandieh, 2020) have improved the dependence on $p$ to polynomial. However, their algorithms mainly focus on optimizing the dependence on $n$ and reducing the exponential dependence on $p$ to polynomial. If $X \in \mathbb{R}^{d \times n}$ is the input matrix and is dense, then these algorithms have runtime $\tilde{O}(pnd + e^{-2}n^3p^2)$. Notice this is unsatisfactory when both $d$ and $p$ are large. Thus, a natural question to ask is:

Does there exist a sketch for polynomial kernels of degree $p$, such that the runtime is nearly linear in $nd$, and with an improved dependence on $p$?

1We use $\tilde{O}(\cdot), \tilde{\Omega}(\cdot), \tilde{\Theta}(\cdot)$ to suppress $\text{poly}(\log(nd/\epsilon \delta))$ factors.
Notice this is especially desirable for kernels such as the neural tangent kernel (NTK) (Jacot et al., 2018) and the arc-cosine kernel (Cho & Saul, 2009), whose Taylor series have a much slower decay rate \(1/n^e\) for some \(e\) compared to the Gaussian kernel (which is \(1/n!\)).

We list our contributions as follows:

- We develop an efficient algorithm that computes a sketch of the polynomial kernel of degree \(p\) in time linear in \(p^2\) and nearly linear in \(nd\).
- Our algorithm only uses two distinct sketches compared to the \(O(p)\) independent sketches of (Ahle et al., 2020). This enables us to use repeated powering to compute our sketch very efficiently.
- We characterize kernel matrices by considering their Taylor series, and provide different algorithmic schemes to solve them. Our characterization includes a family of interesting and popular kernels. We also use this sketch as a preconditioner for solving linear systems involving a kernel matrix, and we extend our sketch to solve kernel ridge regression, by composing it with another sketch that depends on the statistical dimension.

### 1.1. Related Work

**Kernel regression**

Classical regression has the form \(\min_w \|Y - Xw\|_2^2\), where \(X\) and \(Y\) are a given dataset and corresponding labels, respectively. Kernel regression (Buch, 2013; Zhang et al., 2015; Alaoui & Mahoney, 2015; Avron et al., 2017bc; Zandieh et al., 2020; Lee et al., 2020; Alman et al., 2020) allows for \(X\) to be a kernel matrix \(K \in \mathbb{R}^{n \times n}\), where each entry is the application of a kernel function to a pair of data points in \(X\). Kernel regression \(\min_w \|Y - Kw\|_2^2\) enables fitting non-linear data into a hyperplane by transforming the data into a high-dimensional space.

**Sketching techniques for tensor-related problems**

Sketching techniques have been used extensively in tensor-related problems, e.g., for linear-algebraic problems involving polynomial kernels (Avron et al., 2014; Ahle et al., 2020; Woodruff & Zandieh, 2020), for tensor low-rank approximation (Song et al., 2019), and for tensor regression (Haupt et al., 2017; Diao et al., 2018; 2019).

**Subspace embeddings**

An (oblivious) subspace embedding is a useful concept in randomized numerical linear algebra introduced by Sárlós (Sarlos, 2006). Many applications rely on subspace embeddings or their variants, such as linear regression, low-rank approximation (Clarkson & Woodruff, 2013; Nelson & Nguyen, 2013; Meng & Mahoney, 2013; Boutsidis & Woodruff, 2014; Boutsidis et al., 2016; Song et al., 2017; Andoni et al., 2018), tensor decomposition (Song et al., 2019), cutting plane methods (Jiang et al., 2020), and linear programming (Lee et al., 2019; Jiang et al., 2021; Song & Yu, 2021).

**Roadmap**

In Section 2, we introduce definitions, notations and some basic facts. In Section 3, we present a technical overview of our results. In Section 4, we propose an efficient algorithm to generate a sketch and apply it to a polynomial kernel of arbitrary positive integer degree \(p\). In Section 5, we analyze our algorithm with a specific sketching matrix. In Section 6, we present applications to the Gaussian kernel and a more general class of kernels, which can be characterized through the coefficients of their Taylor expansion. We also discuss how to use our sketch as a preconditioner for solving kernel linear systems, and solve sketched kernel ridge regression.

### 2. Preliminaries

For an integer \(n\), let \([n]\) denote the set \(\{1, 2, \cdots, n\}\). For two scalars \(a\) and \(b\), we say \(a \approx b\) if \((1 - \epsilon)b \leq a \leq (1 + \epsilon)b\). We say a square symmetric matrix \(A\) is positive semi-definite (PSD) if \(\forall x, x^\top Ax \geq 0\). For two PSD matrices \(A\) and \(B\), we define \(A \preceq B\) if \((1 - \epsilon)B \preceq A \preceq (1 + \epsilon)B\), where \(A \preceq B\) means \(B - A\) is PSD. For a matrix \(A\), we use \(\|A\|_F = (\sum_{i,j} A_{ij}^2)^{1/2}\) to denote its Frobenius norm and use \(\|A\|_{op}\) to denote its operator (spectral) norm. For a square symmetric matrix \(A\), we use \(\text{tr}[A]\) to denote the trace of \(A\). For a square matrix \(A\), we use \(\lambda_{\min}(A), \lambda_{\max}(A)\) to denote its smallest and largest eigenvalues, respectively. For a rectangular matrix \(A\), we use \(\sigma_{\min}(A), \sigma_{\max}(A)\) to denote its smallest and largest singular values, respectively, and we use \(\kappa = \sigma_{\max}(A)/\sigma_{\min}(A)\) to denote its condition number.

#### 2.1. Definitions

We define an oblivious subspace embedding ((Sarlos, 2006)) as follows:

**Definition 2.1 (Oblivious Subspace Embedding(OSE)).** Let \(\epsilon, \delta \in (0, 1)\) and \(d, n \geq 1\) be integers. An \((\epsilon, \delta, d, n)\)-Oblivious Subspace Embedding (OSE) is a distribution over \(m \times d\) matrices with the guarantee that for any fixed matrix \(A \in \mathbb{R}^{d \times n}\), we have

\[
Pr_{H \sim D}\left[\left(\|HA\|^2\|IA\|^2\right) \approx \epsilon (A^\top A)\right] \geq 1 - \delta
\]

We also introduce tensor products of vectors and Kronecker products of matrices.

**Definition 2.2 (Tensor product of vectors).** Given \(a \in \mathbb{R}^n\) and \(b \in \mathbb{R}^m\), we define the tensor product of \(a\) and \(b\), denoted by \(a \times b\), to be \(\text{vec}(ab^\top)\). We will use \(a^{\otimes p}\) to denote the tensoring of the vector \(a\) with itself a total of \(p\) times.

The Kronecker product of matrices is a natural extension of the tensor product of vectors:
Definition 2.3. Given $A_1 \in \mathbb{R}^{m_1 \times n_1}, A_2 \in \mathbb{R}^{m_2 \times n_2}, \ldots, A_k \in \mathbb{R}^{m_k \times n_k}$, we define $A_1 \times A_2 \times \cdots \times A_k$ to be the matrix in $\mathbb{R}^{m_1 m_2 \cdots m_k \times n_1 n_2 \cdots n_k}$ whose element at row $(i_1, \ldots, i_k)$ and column $(j_1, \ldots, j_k)$ is $A_1(i_1, j_1) \cdots A_k(i_k, j_k)$.

An important property of the Kronecker product is the so-called mixed product property:

Claim 2.4. For conforming matrices $A, B, C, D$, the following holds:

$$ (A \cdot B) \times (C \cdot D) = (A \times C) \cdot (B \cdot D). $$

One consequence is the following claim:

Claim 2.5. Let $A_1 \in \mathbb{R}^{m_1 \times n_1}, A_2 \in \mathbb{R}^{m_2 \times n_2}, \ldots, A_k \in \mathbb{R}^{m_k \times n_k}$ and $v_1 \in \mathbb{R}^{n_1}, v_2 \in \mathbb{R}^{n_2}, \ldots, v_k \in \mathbb{R}^{n_k}$. Then,

$$ (A_1 \times A_2 \times \cdots \times A_k)(v_1 \times v_2 \times \cdots \times v_k) = (A_1 v_1) \times (A_2 v_2) \times \cdots \times (A_k v_k). $$

We will extensively use the following notation:

Definition 2.6. Given $A_1 \in \mathbb{R}^{m_1 \times n_1}, A_2 \in \mathbb{R}^{m_2 \times n_2}, \ldots, A_k \in \mathbb{R}^{m_k \times n_k}$, we define $A_1 \otimes A_2 \otimes \cdots \otimes A_k$ to be the matrix in $\mathbb{R}^{m_1 m_2 \cdots m_k \times n_1 n_2 \cdots n_k}$ whose $j$th column is $A_1^j \times A_2^j \times \cdots \times A_k^j$ for every $j \in [n]$, where $A_i^j$ is the $j$th column of $A_i$ for every $i \in [k]$.

2.2. Sketching Matrices

We recall the Subsampled Randomized Hadamard Transform (SRHT), which is a Fast Johnson-Lindenstrauss transform (Ailon & Chazelle, 2006).

Definition 2.7 (Subsampled Randomized Hadamard Transform (SRHT), see (Lu et al., 2013; Woodruff et al., 2014)). The SRHT matrix $S \in \mathbb{R}^{m \times d}$ is defined as $S = \frac{1}{\sqrt{m}} PHD$, where $P \in \{0, 1\}^{m \times d}$ is a sampling matrix where each row contains exactly one 1 at a uniformly random coordinate, $H$ is the $d \times d$ Hadamard matrix, and $D$ is a $d \times d$ diagonal matrix with independent Rademacher random variables on its diagonal.

 Remark 2.8. Using the Fast Fourier Transform (FFT) (Cooley & Tukey, 1965), $S$ can be applied to a vector in time $O(d \log d)$.

We also introduce a sketching matrix for degree-2 tensors, which is a generalization of the SRHT.

Definition 2.9 (Tensor Subsampled Randomized Hadamard Transform (TensorSRHT) (Ahle et al., 2020)). We define the TensorSRHT $S : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^m$ as $S = \frac{1}{\sqrt{m}} P \cdot (HD_1 \times HD_2)$, where $P \in \{0, 1\}^{m \times d}$ is a sampling matrix where each row contains only one 1 at a uniformly random coordinate. $H$ is a $d \times d$ Hadamard matrix, and $D_1, D_2$ are two $d \times d$ independent diagonal matrices with diagonals that are each independently set to be a Rademacher random variable (uniform in $\{-1, 1\}$).

Remark 2.10. By leveraging the FFT algorithm in the sketch space, $S(x \otimes^2)$ can be computed in time $O(d \log d + m)$.

We will use the following properties of the SRHT and TensorSRHT.

Lemma 2.11 (Theorem 2.4 in (Woodruff et al., 2014)). Let $T$ be an SRHT matrix defined in Definition 2.7. If $m = O(n \log(nd/\delta)e^{-2})$, then $T$ is an $(\epsilon, \delta, d, n)$-OSE.

Lemma 2.12 (Lemma 21 in (Ahle et al., 2020)). Let $S$ be a TensorSRHT matrix defined in Definition 2.9. If $m = O(n \log^3(nd/\delta)e^{-2})$, then $S$ is an $(\epsilon, \delta, d, n)$-OSE for degree-2 tensors.

2.3. Kernels

We introduce several kernels that are widely-used in practice, e.g., see (Goldberg & Elhadad, 2008; Chang et al., 2010) for the polynomial kernel, and see (Ng et al., 2001) for the Gaussian kernel.

Definition 2.13 (Polynomial Kernel). Given two data points $x, y \in \mathbb{R}^d$, the degree-$p$ polynomial kernel, $P$, between $x$ and $y$ is defined as:

$$ P(x, y) = \langle x, y \rangle^p. $$

Let $X \in \mathbb{R}^{d \times n}$. The degree-$p$ polynomial kernel $P$, defined on matrix $X$, is the matrix $P_{i,j} = \langle x_i, x_j \rangle^p$, where $x_i, x_j$ are the $i$th and $j$th column of $X$, respectively.

Definition 2.14 (Gaussian Kernel). Given two data points $x, y \in \mathbb{R}^d$, the Gaussian kernel, $G$, between $x$ and $y$ is defined as:

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

Let $X \in \mathbb{R}^{d \times n}$. The Gaussian kernel $G$, defined on matrix $X$, is the matrix $G_{i,j} = \exp(-\|x_i - x_j\|^2/2)$, where $x_i, x_j$ are the $i$th and $j$th column of $X$, respectively.

3. Technical Overview

We first consider one way to compute the polynomial kernel $P$ via the identity $P = (X \otimes^p) \cdot (X \otimes^p)$. Our algorithm will try to compute $X \otimes^p$ quickly.

Suppose we want to compute the $p$-fold tensoring of a vector $x$, and assume for simplicity that $p$ is a power of 2. Our algorithm is inspired by that of (Ahle et al., 2020), which generates a complete binary tree with $2p - 1$ nodes, and thus $p$ leaves. For the $i$-th leaf node, it picks a sketch $T_i$ and applies it to $x$, obtaining $T_i^T x$. Each internal node $j$ then does the following: it picks a sketch $S_j$, and applies $S_j^T$ to the tensor product of its two children. $S_j^T$ is picked as a map from $\mathbb{R}^{m^2} \to \mathbb{R}^m$ for each $j$, so at each level of the binary tree, we reduce the number of vectors by half, while remaining in the low-dimensional space $\mathbb{R}^m$. One drawback of this algorithm is the usage of an independent sketch for each node of the tree, thus incurring a linear dependence on $p$ in the runtime.

---

1 A more standard definition is $P(x, y) = \langle x, y \rangle^p + c$; we can simulate this by creating an extra dimension on all data points.
Our algorithm instead uses a much smaller amount of randomness. We pick only a single $T$ and a single $S$, i.e.,

$$T^1 = T^2 = \cdots = T^p$$

for all leaf nodes, and we have $S^j = S$ for all internal nodes $j$. The challenge with this approach is of course that we have much less independence in our analysis, and consequently do not obtain the same guarantees for preserving the length of the tensor product of an arbitrary set of vectors as in previous work. We stress that we compute a sketch that preserves the column span of $X \otimes p$, and this is weaker than we would get had we used full independence, which gives a sketch that is an oblivious subspace embedding, meaning that it can preserve the column span of any matrix in $\mathbb{R}^{d \times n}$. However, the key point is that we can preserve the tensor product of a vector with itself $p$ times, and this will suffice for our applications.

This allows for a much faster way to compute $x \otimes p$: “square” a vector by computing the tensor product with itself, then apply a sketch, and repeat this process. By doing so, we reduce the dependence on $p$ in the first level of the tree from linear to logarithmic. However, this will incur a $p^2$ factor in the dimension of the sketch, and so we will pay more for $p$ in levels other than the first level. Fortunately, levels other than the first level apply sketches to lower dimensional vectors. By carefully balancing the complexity of applying $T$ and $S$, we achieve an improved running time, which is useful when the degree $p$ is large.

4. Fast Sketching Algorithm for the Polynomial Kernel

We introduce our algorithm that sketches a single vector $x \otimes p$ and extend it to each column of a matrix. In Section 4.1 we give some definitions. In Section 4.2 we prove several technical tools related to tensors. In Section 4.3 we show our sketch preserves the column space of the polynomial kernel. In Section 4.4 we prove our main result for this section.

4.1. Definitions

We define the sketching matrix formed by Algorithm 1 as follows:

**Definition 4.1.** Let $q$ be a power of 2 and $\Pi^q : \mathbb{R}^{d^q} \to \mathbb{R}^m$ be defined as the following matrix:

$$\Pi^q = Q^q \cdot T^q,$$

where $T^q = T \times T \times \cdots \times T$ and $Q^q = S^1 \cdot S^2 \cdot S^4 \cdot \cdots$, $S^q/2$, and $S^1 = S \times S \times \cdots \times S$.

We will design an algorithm that achieves the following goal:

**Case 1** If $p$ is a power of 2, then it computes $\Pi^p X \otimes p$ efficiently.

**Case 2** If $p$ is not a power of 2, then let $b$ be its binary representation and let $E = \{ i : b_i = 1, i \in \{0, \ldots, \log_2 p\}\}$. We will iterate through all indices in $E$ and continue tensoring two vectors where $b_i = 1$, and apply $S$ to them.

**Definition 4.2.** Let $S \in \mathbb{R}^{m^q} \to \mathbb{R}^m$ 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 $Z(S, T, X)$ to be the matrix for which we apply Algorithm 1 on each column of $X$, with base sketches $S$ and $T$.

4.2. Equivalence Results for Tensors

We provide two technical tools for handling tensors.

**Lemma 4.3** (The output guarantee of Algorithm 1). Let $p$ be a power of 2 and $\Pi^p$ be defined as in Definition 4.1. Let $x \in \mathbb{R}^d$. Then the output vector $z$ generated by Algorithm 1 satisfies $z = \Pi^p(x \otimes p)$.

**Proof.** If $p$ is a power of 2, then Algorithm 1 will output $z$ on line 8. We will exploit the fact that although Algorithm 1 only computes one vector at a time, we can view it as computing $p/2^i$ identical vectors in the $i$th iteration. On line 3, we can treat it as computing $p$ copies of $w_0$, and therefore, by Claim 2.5, we have

$$w_0^\otimes p = (Tx)^\otimes p$$

$$= (T \times T \times \cdots \times T)(x \times x \times \cdots \times x)$$

$$= T^p x^\otimes p.$$

We can apply the same line of reasoning to line 4 of the algorithm. In the $i$th iteration, we can treat it as

$$S(w_{i-1}^\otimes 2) \times S(w_{i-1}^\otimes 2) \times \cdots \times S(w_{i-1}^\otimes 2)$$

a total of $p/2^i$ times. Again, using Claim 2.5, we have

$$w_i^{\otimes p/2^i} = S(w_{i-1}^\otimes 2) \times S(w_{i-1}^\otimes 2) \times \cdots \times S(w_{i-1}^\otimes 2)$$

$$= (S(w_{i-1}^\otimes 2))^{\otimes p/2^i}$$

$$= (S \times S \times \cdots \times S)(w_{i-1}^\otimes 2)^{\otimes p/2^i}$$

$$= 2^p \times w_{i-1}^{\otimes p/2^i}.$$

Recursively applying this identity, we will end up with

$$z = S^1 \cdot S^2 \cdot \cdots \cdot S^p \cdot T^p x^\otimes p = \Pi^p(x^\otimes p). \square$$

Next, we wish to show that if $p$ is a power of 2, then $\Pi^p$ preserves the subspace spanned by the columns of $X^\otimes p$ within a factor of $1 + \frac{2}{2^p}$. Notice this is weaker than $\Pi^p$ being an
Fast Sketching of Polynomial Kernels of Polynomial Degree

Algorithm 1 Our algorithm for sketching the vector $x \otimes^p$ with limited randomness.

1: procedure TENSORSKETCHVIALIMRAND($x \in \mathbb{R}^d, p \in (1, \infty), S \in \mathbb{R}^{m \times m^2}, T \in \mathbb{R}^{m \times d}$) \hspace{1cm} \triangleright \text{Theorem 4.8 and 5.1}
2: \hspace{1cm} Let $q = 2^{\lfloor \log_2 p \rfloor}$
3: \hspace{1cm} Let $w_0 = Tx$
4: \hspace{1cm} for $l = 1$ to $\log_2 q$ do
5: \hspace{2cm} Compute $w_l = S(w_{l-1}^\top)$ \hspace{1cm} \triangleright \text{$S$ can be TensorSRHT (Definition 2.9)}
6: \hspace{1cm} end for
7: \hspace{1cm} Let $b$ be the binary representation of $p$, and let $E = \{i : b_i = 1, i \in \{0, \ldots, \log_2 p\}\}$
8: \hspace{1cm} Let $z = w_j$, where $j$ is the lowest bit of $b$ where $b_j = 1$
9: \hspace{1cm} for $i$ in $E \setminus \{j\}$ do
10: \hspace{2cm} $z = S(z \times w_i)$
11: \hspace{1cm} end for
12: \hspace{1cm} return $z$ \hspace{1cm} \triangleright $z \in \mathbb{R}^m$
13: end procedure

OSE, but sufficient for our application to polynomial kernels. We will show that

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

The following lemma outlines the main technique in our proof of this property. It establishes a one-to-one mapping between a vector in the column span of $X \otimes^p$ and a matrix. We then use this equivalence to inductively prove that $\Pi^p$ preserves the target subspace.

Lemma 4.4 (From Vector Tensor to Matrix Tensor). Let $X \in \mathbb{R}^{d \times n}$. Consider $u = X \otimes^p y$ for some $y \in \mathbb{R}^n$ and positive integer $p$. Then,

$$\|u\|_2 = \|X \otimes^{(p-1)} Y X^\top\|_F,$$

where $Y = \text{diag}(y) \in \mathbb{R}^{n \times n}$ is a diagonal matrix where the $i$-th entry on the diagonal is $y_i$, $\forall i \in [n]$.

Proof. We will use $X_{i,j}$ to denote the $i$-th row and $j$-th column of $X$. We observe that a fixed column $X_{*,j}$ is equivalent to taking the outer product $X_{*,j} \otimes (p-1)$, then flattening the matrix into a long vector. If we use a double index to indicate an entry of $X_{*,j}$, then we have $X_{*,j}(a,b) = X \otimes^{(p-1)} (a) X(b)$, where $a$ ranges from 1 to $d^{p-1}$ and $b$ ranges from 1 to $d$. Consider the $\ell_2$ norm $u$:

$$\|u\|_2^2 = \|X \otimes^p y\|_2^2$$

$$= \sum_{a=1}^{d^{p-1}} \left( \sum_{b=1}^d y_i(X_{*,j}(a,b)) \right)^2 .$$

On the other hand, we can write $X \otimes^{(p-1)}$ in its column form:

$$\begin{bmatrix}
X_{*,1}^{(p-1)} & X_{*,2}^{(p-1)} & \cdots & X_{*,n}^{(p-1)}
\end{bmatrix} .$$

Recall that $Y$ is a diagonal matrix and therefore, the product $X^{(p-1)} Y$ can be expressed as

$$\begin{bmatrix}
y_1X_{*,1}^{(p-1)} & y_2X_{*,2}^{(p-1)} & \cdots & y_nX_{*,n}^{(p-1)}
\end{bmatrix} .$$

Using the outer product definition of matrix product, we have

$$X^{(p-1)} Y X^\top = \sum_{i=1}^n y_i X_{*,i}^{(p-1)} X_{*,i}^\top .$$

This is a matrix of size $d^{p-1} \times d$. Therefore, we can write its Frobenius norm as

$$\|X^{(p-1)} Y X^\top\|_F^2 = \sum_{i=1}^n y_i \|X_{*,i}^{(p-1)}\|_F^2$$

$$= \sum_{a=1}^{d^{p-1}} \sum_{b=1}^d \left( \sum_{i=1}^n y_i \|X_{*,i}^{(p-1)}(a) X_{*,i}(b)\|_2^2 \right)^2$$

$$= \sum_{a=1}^{d^{p-1}} \sum_{b=1}^d \left( \sum_{i=1}^n y_i \|X_{*,i}^{(p-1)}(a,b)\|_2^2 \right)^2$$

$$= \|u\|_2^2 .$$

This completes the proof. \hfill \Box

4.3. Preserving the Polynomial Kernel Subspace

We prove the sketch generated by Algorithm 1 preserves the column space of the polynomial kernel.

Lemma 4.5 ($T^p$ Preserves Polynomial Kernel Subspace). Let $X \in \mathbb{R}^{d \times n}$, $y \in \mathbb{R}^n$ and $u = X \otimes^p y$, where $p$ is a positive integer. If $T$ is an $(\epsilon, \delta, d, n)$ OSE, then with probability at least $1 - \delta$,

$$\|TX \otimes^p y\|_2^2 = (1 \pm \epsilon) \|X \otimes^p y\|_2^2$$
Fast Sketching of Polynomial Kernels of Polynomial Degree

Proof. We proceed by induction on \( p \). For \( p = 1 \), by Definition 2.1, we have

\[
\|TXy\|_2^2 = (1 + \epsilon)\|Xy\|_2^2.
\]

For the inductive step, we will prove this for a general positive integer \( p > 1 \). We break \( p \) into \( p - 1 \) and 1. By Lemma 4.4, we have

\[
\|TX\otimes p y\|_2^2 = \|TX\otimes(p-1)YTX^T\|_F^2,
\]

Recall that \( T \) is an \((\epsilon, \delta, d, n)\) OSE for \( X \). This means right multiplying by \( T^T \) preserves the length of all columns of \( (TX)\otimes(p-1)YX^T \), and therefore we have

\[
\|TX\otimes(p-1)YX^T\|_F^2 = (1 + \epsilon)\|TX\otimes(p-1)YX^T\|_F^2.
\]

Using the inductive hypothesis, for any vector \( z \in \mathbb{R}^n \), we have

\[
\|TX\otimes(p-1)z\|_2^2 = (1 + \epsilon)\|X\otimes(p-1)z\|_2^2.
\]

Applying this to each column of \( YX^T \), we have

\[
\|TX\otimes(p-1)YX^T\|_F^2 = (1 + \epsilon)\|X\otimes(p-1)YX^T\|_2^2
\]

This enables us to conclude that

\[
\|TX\otimes p y\|_2^2 = (1 + \epsilon)^p \|X\otimes p y\|_2^2.
\]

\(\square\)

Remark 4.6. As we motivated in Section 3, if we view Algorithm 1 as a binary tree, Lemma 4.5 effectively proves that the bottom layer of the tree preserves the column space of \( X\otimes p \). We then pick \( S \) to be an OSE for degree-2 tensors, and inductively establish our embedding.

Lemma 4.7 (\( \Pi^p \) Preserves the Polynomial Kernel Subspace). Let \( S : \mathbb{R}^{m^d} \to \mathbb{R}^m \) be an \((\epsilon, \delta, d, n)\)-OSE for degree-2 tensors, and let \( T : \mathbb{R}^d \to \mathbb{R}^m \) be an \((\epsilon, \delta, d, n)\)-OSE (Definition 2.1). Let \( p \) be a power of 2. Let \( \Pi^p \) be the sketching matrix defined as in Definition 4.1. Then we have for any \( y \in \mathbb{R}^n \), with probability at least \( 1 - \delta \),

\[
(1 - \epsilon)^{2p} \|X\otimes p y\|_2 \leq \|\Pi^p X\otimes p y\|_2 \leq (1 + \epsilon)^{2p} \|X\otimes p y\|_2.
\]

Proof. We will prove this by induction on the number of iterations of Algorithm 1. Let \( k = \log_2 p \); we will induct on the parameter \( l \) from 1 to \( k \). Let \( \Upsilon^2 \) denote the sketching matrix at level \( l \): \( \Upsilon^2 = S^{p/2} : \cdots : S^{p/2^{l-1}} : S^{p/2} : T^p \); \forall l \in [k] \) and \( \Upsilon^0 = T^p \). We will prove the following statement: \( \forall l \in \{0, \ldots, k\} \), we have

\[
\|\Upsilon^l X\otimes p y\|_2 \leq (1 + \epsilon)^{2p} \|X\otimes p y\|_2.
\]

Note that when \( l = k \), we have \( \Upsilon^k = \Pi^p \), and therefore it gives us the desired result.

For the base case, note that Lemma 4.5 automatically gives our desired result. For the inductive step, we assume it holds for some \( l - 1 \), so

\[
\|\Upsilon^{l-1} X\otimes p y\|_2 \leq (1 + \epsilon)^{2p} \|X\otimes p y\|_2.
\]

Notice that \( \Upsilon^{l-1} = S^{p/2^{l-1}} \cdot \Upsilon^{l-1} \). Let \( Z \) be defined as the matrix

\[
Z = \begin{bmatrix}
(x_{1-1}^2) & (x_{2-1}^2) & \cdots & (x_{n-1}^2)
\end{bmatrix},
\]

where we use \( x_i^j \) to denote the \( j \)th column of \( X \) after the \( j \)th iteration. From Algorithm 1, we have that \( Z = \Upsilon^{l-1} X\otimes p \), and so the product \( S^{p/2^{l}} \cdot \Upsilon^{l-1} X\otimes p \) can be written as \((SZ)\otimes p^{2^{l-1}}\).

If \( p/2^l > 1 \), then similar to Lemma 4.5,

\[
\|S^{p/2^{l}} y\|_2 = (1 + \epsilon)^{p/2^l} \|S^{p/2^l} y\|_2.
\]

The third step uses the same reasoning as Lemma 4.5, i.e., we can pull out \( S \) by paying an extra \((1 + \epsilon)^{p/2^l - 1}\) factor. The last line uses the inductive hypothesis.

If \( p/2^l = 1 \), then we will end up with \( SZy \), and can simply use the fact that \( S \) is an OSE to argue that \( SZy \) preserves the length of \( Zy \). We then use the inductive hypothesis on \( Z \) to conclude the proof. \(\square\)

Below, we state and prove a theorem that establishes the correctness of Algorithm 1 without instantiating the sketching matrix \( T \) and \( S \). This enables us to use different sketches with various trade-offs.

4.4. Main Result

We prove the main result of this section, which establishes the correctness of Algorithm 1.

Theorem 4.8 (Main Result, Correctness Part). Let \( S : \mathbb{R}^{m^d} \to \mathbb{R}^m \) be an \((\epsilon, \delta, 0, d, n)\)-OSE for degree-2 tensors and \( T : \mathbb{R}^d \to \mathbb{R}^m \) be an \((\epsilon, \delta, d, n)\)-OSE. Let \( p \) be a positive integer. Let \( Z = Z(S, T, X) \) be the matrix as defined in Def. 4.2. Then for any \( y \in \mathbb{R}^n \), we have

\[
(1 - \epsilon)^{3p} \|X\otimes p y\|_2 \leq \|Zy\|_2 \leq (1 + \epsilon)^{3p} \|X\otimes p y\|_2.
\]
Fast Sketching of Polynomial Kernels of Polynomial Degree

Proof. Let $b$ be the binary representation of $p$, and let $E = \{i : b_i = 1, i \in \{0, 1, \ldots, \log_2 p\}\}$. If $p$ is a power of 2, by Lemma 4.7, we are done. So suppose $p$ is not a power of 2. Let $q = 2^{\lfloor \log_2 p \rfloor}$. Algorithm 1 computes $\Pi^q X^{\otimes q}$ and combines intermediate results with indices in $E$ to form the final result. We will again prove this by induction on the indices in $E$, from smallest to largest. For the base case, let $i_1$ be an index in $E$ and let $q_1 = 2^{i_1}$. Since $q_1$ is a power of 2, Lemma 4.7 establishes this case.

For the inductive step, suppose this holds for $i_1, i_2, \ldots, i_{j-1} \in E$, and let $q_1 = 2^{i_1}, q_2 = 2^{i_2}, \ldots, q_j-1 = 2^{i_{j-1}}$. We will prove this holds for $q_j = 2^{i_j}$. Let $Z$ denote the matrix after the $(j - 1)$th application of this recursive process. We will show that

$$\|S((\Pi^{q_1} X^{\otimes q_1} \otimes Z) y_2^2)\|_2 = (1 \pm \epsilon)^{j-1+\sum_{i=1}^{j-1} 2q_i} \|X^{\otimes j} y\|_2^2.$$

(1)

We first use the fact $S$ is an OSE to obtain

$$\|S((\Pi^{q_1} X^{\otimes q_1} \otimes Z) y_2^2)\|_2 = (1 \pm \epsilon)\|((\Pi^{q_1} X^{\otimes q_1} \otimes Z) y\|_2^2.$$

(2)

By Lemma 4.4, we have

$$\|((\Pi^{q_1} X^{\otimes q_1} \otimes Z) y\|_2^2 = \|\Pi^{q_1} X^{\otimes q_1}\|_F \|y\|_2^2,$$

$$= (1 \pm \epsilon)^{2q_1} \|X^{\otimes q_1} y\|_2^2 \|y\|_2^2 \|y\|_2^2,$$

$$= (1 \pm \epsilon)^{2q_j-1+\sum_{i=1}^{j-1} 2q_i} \|X^{\otimes q_j} y\|_2^2 \|y\|_2^2,$$

$$= (1 \pm \epsilon)^{j-1+\sum_{i=1}^{j-1} 2q_i} \|X^{\otimes j} y\|_2^2.$$

(3)

Combining Eq. (2) and Eq. (3), we obtain Eq. (1), which is our desired result.

5. Analysis of Sketching Matrices: SRHT and TensorSRHT

In this section, we analyze the runtime of Algorithm 1 with $T$ being an SRHT sketch (Definition 2.7) and $S$ being a TensorSRHT sketch (Definition 2.9).

5.1. Main Result

The goal of this section is to give a runtime analysis of Algorithm 1 using SRHT as $T$ and TensorSRHT as $S$.

Theorem 5.1 (Main Result, Running Time). Let $p \in \mathbb{N}_+$ and $\epsilon, \delta \in (0, 1)$. Then for every $X \in \mathbb{R}^{d \times n}$, there exists a distribution over oblivious linear sketches $\Pi : \mathbb{R}^p \rightarrow \mathbb{R}^m$ such that if $m = \Theta(\epsilon^{-2} n^2 p^2)$, we have

$$\|\Pi X^{\otimes p}\|^2 \approx \epsilon (X^{\otimes p})^\top X^{\otimes p}.$$

Moreover, using Algorithm 1, $\Pi X^{\otimes p} = Z(S, T, X)$ can be computed in time $O(nd + \epsilon^{-2} n^2 p^2)$.

Proof. We will use an SRHT for $T$ and a TensorSRHT for $S$. We pick both of these sketches to be $(\epsilon, \delta, d, n)$-OSEs, where $\epsilon = \frac{\epsilon}{\delta}$, $\delta = \frac{\delta}{\epsilon}$, $d = \Theta(n/\epsilon)$ and $n$ OSEs, we need $m = \Theta(n^2/\epsilon)$. We then enter a loop with $O(\log p)$ iterations, where in each iteration we apply $S$ to the tensor product of a column with itself resulting from the previous iteration. Since $S$ is a TensorSRHT, this takes $O(m) = \tilde{O}(p^2 n^2/\epsilon^2)$. We also compute each bit in the binary representation, which incurs an $O(\log p)$ factor in the final runtime. So it takes Algorithm 1 $O(nd + p^2 n^2/\epsilon^2)$ time to compute $Z(S, T, X)$. This completes the proof.

5.2. Discussion

We compare our result with the results obtained in (Ahle et al., 2020; Woodruff & Zandieh, 2020). The setting we are considering is 1) matrix $X$ is dense, i.e., $\text{nnz}(X) \approx nd$, and 2) $d \gg n$. In such a scenario, (Ahle et al., 2020) obtains a sketching dimension $m = \Omega(\epsilon^{-2} n^2 p^2)$ and the runtime of applying the sketch to $X$ is $\tilde{O}(p^2 n^2/\epsilon^2)$, so our result improves the dependence on the $nd$ term and pays only $n^2$ instead of $n^3$ on the second term. Another result from (Woodruff & Zandieh, 2020) has $m = \Theta(\epsilon^{-2} n)$ but the time to apply sketching is $\tilde{O}(p^2 \cdot nd + \text{poly}(\epsilon^{-1}, p) n^3)$, which is much worse in the leading $nd$ term, compared to our result. However, we also point out the results obtained in these two works are more general than ours in the sense that their sketches have the OSE property, while our sketch only preserves the column space of $X^{\otimes p}$. Nevertheless, the latter suffices for our applications. We use Table 1 to summarize and compare the different results.
Fast Sketching of Polynomial Kernels of Polynomial Degree

<table>
<thead>
<tr>
<th>Reference</th>
<th>Sketch Dimension</th>
<th>Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Ahle et al., 2020)</td>
<td>(\Theta(e^{-2}n^2p))</td>
<td>(O(pmd + e^{-2}n^3p^2))</td>
</tr>
<tr>
<td>(Woodruff &amp; Zandieh, 2020)</td>
<td>(\Theta(e^{-2}n))</td>
<td>(O(p^{2.5}nd + \text{poly}(e^{-1}, p)n^3))</td>
</tr>
<tr>
<td>Theorem 5.1</td>
<td>(\Theta(e^{-2}np^2))</td>
<td>(O(nd + e^{-2}n^2p^2))</td>
</tr>
</tbody>
</table>

Table 1: Comparison of different algorithms. We assume \(\text{nnz}(X) \approx nd\) and \(d \gg n\). We also assume there is no regularization, i.e., \(\lambda = 0\).

sketches can be composed with sketches that do. Therefore, we consider the case when there is no regularization \((\lambda = 0)\) and \(X^{\otimes p}\) is of full rank. In this case, the statistical dimension reduces to \(n\).

6. Applications

In this section, we introduce various applications using our sketch. In Section 6.1, we study approximating the Gaussian kernel using our algorithm. In Section 6.2 we extend the analysis to a class of slow-decaying kernels. In Section 6.3 we illustrate an efficient algorithm to solve kernel linear systems. In Section 6.4 we show how to solve kernel ridge regression using our sketch.

6.1. Gaussian Kernels

We provide the fastest algorithm to preserve the column space of a Gaussian kernel when \(d\) is large.

Theorem 6.1 (Gaussian Kernel, informal version of Theorem A.1). Let \(r \in \mathbb{R}_+\) and let \(X \in \mathbb{R}^{d \times n}\) be the data matrix such that \(\|x_i\|_2 \leq r\) for all \(i \in [n]\), where \(x_i\) is the \(i\)th column of \(X\). Suppose \(G \in \mathbb{R}^{n \times n}\) is the Gaussian kernel matrix given in Definition 2.14. Then there exists an algorithm that computes a matrix \(W_g(X) \in \mathbb{R}^{m \times n}\) in time \(\tilde{O}(e^{-2}n^3q^3 + nd)\), such that for every \(\epsilon > 0\),

\[
\Pr [W_g(X)^	op W_g(X) \approx_G G] \geq 1 - 1/\text{poly}(n),
\]

where \(m = \tilde{O}(q^3n/e^2)\) and \(q = \Theta(r^2 + \log(n/e))\).

We provide a sketch of the proof here, and further details can be found in the supplementary material. The Taylor expansion of the Gaussian kernel can be written as

\[
G = \sum_{l=0}^{\infty} \frac{(X^{\otimes l}D)^\top X^{\otimes l}D}{l!},
\]

where \(D\) is a diagonal matrix with \(D_{i,i} = \exp(-\|x_i\|_2^2/2)\). Let

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

If we set \(q = \Omega(r^2 + \log(n/e))\) and just use the first \(q\) terms of \(K\):

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

then we have that \(\|K - Q\|_\infty \leq \frac{\epsilon}{2}\). Our algorithm applies Algorithm 1 on each term of \(Q\). This gives us the desired runtime and dimension. The complete proof is in Appendix A.

6.2. General \(p\)-convergent Kernels

A key advantage of Algorithm 1 is its moderate dependence on the degree \(p\), which gives it more leverage when \(p\) is large. We introduce a characterization of kernels, based on the series of the coefficients in the Taylor expansion of the kernel. As we will later see in the proof of Theorem B.2, the decay rate of coefficients has a direct relation with the degree \(p\) we need for approximating a kernel.

Definition 6.2 (\(p\)-convergent kernel, informal version of Definition B.1). We say the kernel matrix \(K\) for data matrix \(X\) is \(p\)-convergent if its corresponding Taylor expansion series can be written as follows: \(K = \sum_{l=0}^{\infty} C_l \cdot (X^{\otimes l})^\top X^{\otimes l}\), where the coefficients \(C_l = (l+1)^{-\Theta(p)}\).

Theorem 6.3 (\(p\)-convergent Kernels, informal version of Theorem B.2). Let \(r \in \mathbb{R}_+\) and \(p > 1\) be an integer, and let \(X \in \mathbb{R}^{d \times n}\) be the data matrix such that \(\|x_i\|_2 \leq r\) for all \(i \in [n]\), where \(x_i\) is the \(i\)th column of \(X\). Suppose \(K\) is a \(p\)-convergent kernel. If \(p > 1\), let \(m = \tilde{O}(e^{-2}nq^3)\) and \(q = \Theta(r^2 + (n/e)^{1/p})\). There exists an algorithm that computes a matrix \(W_g(X) \in \mathbb{R}^{m \times n}\) in time \(\tilde{O}(e^{-2}n^2q^3 + nd)\) such that

\[
\Pr [W_g(X)^	op W_g(X) \approx_G G] \geq 1 - 1/\text{poly}(n).
\]

For the sake of illustration, suppose \(r = 1\). Then the first term in the running time becomes \(e^{-2}n^2q^3 + nd\). When \(p\) is large, Theorem 6.3 gives a fast algorithm for approximating the kernel, but the runtime becomes much slower when \(p \in (1, 3)\). Therefore, we propose a novel sampling scheme to deal with small values of \(p\). Roughly speaking, we exactly compute the first \(s\) terms in the Taylor expansion, while for the remaining \(q - s\) terms we sample only \(s\) of them proportional to their coefficient. Using a matrix Bernstein
Fast Sketching of Polynomial Kernels of Polynomial Degree

bound (Theorem B.4), we obtain an even faster algorithm. We apply our result to the neural tangent kernel (NTK), which is a 1.5-convergent kernel.

**Corollary 6.4** (Approximate NTK, informal version of Corollary C.3). Let \( r \in \mathbb{R}_+ \) and let \( X \in \mathbb{R}^{d \times n} \) be a data matrix for which \( \| x_i \|_2^2 \leq 1 \) 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 NTK matrix. Then there exists an algorithm that computes a matrix \( W_g(X) \) in time

\[
\tilde{O}(\epsilon^{-3}n^{11/3} + nd)
\]

such that

\[
\Pr [W_g(X)^TW_g(X) \approx K] \geq 1 - \delta.
\]

**Remark 6.5.** We remark that our definition of \( p \)-convergent kernels captures a wide range of kernels that have slow decay rate in their coefficients in their Taylor expansion, such as NTK and arc-cosine kernels. Typically, the coefficients are of the form \( 1/n^c \) for some \( c > 1 \). In contrast, Gaussian kernels enjoy a much faster decay rate, and therefore, designing algorithm for the Gaussian kernel is considerably simpler, since the number of terms we need to approximate it with in its Taylor expansion is small, and no sampling is necessary.

### 6.3. Kernel Linear Systems

Another interesting application of our sketching scheme is to constructing a preconditioner for solving PSD systems involving a kernel matrix (Cutajar et al., 2016). In order to apply algorithms such as Conjugate Gradient (Shewchuk, 1994), one has to obtain a good preconditioner for a potentially ill-conditioned kernel system.

**Theorem 6.6** (Sketching as a Preconditioner, informal version of Theorem D.1). Let \( G \in \mathbb{R}^{n \times n} \) be the Gaussian kernel matrix for \( X \in \mathbb{R}^{d \times n} \) and suppose \( \| x_i \|_2 \leq 1, \forall i \in [n] \), where \( x_i \) is the \( i \)-th column of \( X \). Let \( G = Z^\top Z \) and \( k \) denote the condition number of \( Z \). There exists an algorithm that, with probability at least \( 1 - \delta \), computes an \( \epsilon \)-approximate solution \( \hat{x} \) satisfying

\[
\| G\hat{x} - y \|_2 \leq \epsilon \| y \|_2
\]

in \( \tilde{O}(\epsilon^{-2}n^2 \log(\kappa/\epsilon) + n^\omega + nd) \) time, where \( \omega \) is the exponent of matrix multiplication (currently \( \omega \approx 2.373 \) (Williams, 2012; Le Gall, 2014)).

**Remark 6.7.** In certain NLP (Drikvandi & Lawal, 2020) and biological tasks (Teodoro et al., 2002) where \( d = n^c \) for a positive integer \( c \), Theorem 6.6 provides a fast algorithm for which the running time depends nearly linearly on \( nd \). We also remark that the algorithm we use for Theorem 6.6 is inspired by the idea of (Brand et al., 2021) (their situation involves \( c = 4 \)). It is also interesting that in their applications, regularization is not needed since solving a kernel system is equivalent to training an over-parametrized ReLU network without regularization.

### 6.4. Kernel Ridge Regression

Kernel ridge regression (KRR) is a popular method to model the relationship between data points and labels. Let \( K = A^\top A \) denote the kernel matrix. Instead of solving the ordinary regression \( \min_{x \in \mathbb{R}^n} \| K x - y \|_2^2 \), which is equivalent to solving a linear system, we focus on solving the following ridge regression problem:

\[
\min_{x \in \mathbb{R}^n} \| K x - y \|_2^2 + \lambda \| A x \|_2^2,
\]

for \( \lambda > 0 \). A relevant notion is the statistical dimension:

**Definition 6.8** (Statistical Dimension). Let \( \lambda > 0 \), and \( K \in \mathbb{R}^{n \times n} \) be a positive semi-definite matrix. We define the \( \lambda \)-statistical dimension of \( K \) to be

\[
s_\lambda(K) := \text{tr}[K(K + \lambda I_n)^{-1}].
\]

One drawback of our sketch is that we cannot obtain a dimension depending on \( s_\lambda(K) \) instead of \( n \), since it does not have the approximate matrix product property. To mitigate this effect, we propose the following composition of sketches.

**Theorem 6.9** (Kernel Ridge Regression, informal version of Theorem E.2). Let \( \epsilon \in (0, 1), p > 1 \) be an integer and \( X \in \mathbb{R}^{d \times n} \). If \( K \) is its degree-\( p \) polynomial kernel with statistical dimension \( s_\lambda(K) \), where \( \lambda < \epsilon^{-2}\lambda_{\text{max}}(K) \), then we can compute \( Z \) such that \( Z^\top Z \) is a \( (1 \pm \epsilon) \)-spectral approximation to \( K \), in \( \tilde{O}(\epsilon^{-2}p^2n^2 + nd) \) time. Moreover, there exists a matrix \( S \) with \( m = \tilde{O}(\epsilon^{-1}s_\lambda(K)) \) rows such that the optimal solution \( x^* \) to \( \| S(Z^\top Z x - b) \|_2^2 + \lambda \| Zx \|_2^2 \) satisfies

\[
\| Kx^* - y \|_2^2 + \lambda \| X^{\otimes p} x^* \|_2^2 \leq (1 + \epsilon) \cdot \text{OPT},
\]

where \( \text{OPT} \) is \( \min_{x \in \mathbb{R}^n} \| Kx - y \|_2^2 + \lambda \| X^{\otimes p} x \|_2^2 \). The time to solve the above KRR is

\[
\tilde{O}(\epsilon^{-2}p^2n(n + m^2) + n^\omega).
\]

**Acknowledgments:** D. Woodruff would like to thank partial support from NSF grant No. CCF-1815840, Office of Naval Research grant N00014-18-1-256, and a Simons Investigator Award.
Fast Sketching of Polynomial Kernels of Polynomial Degree

References


Fast Sketching of Polynomial Kernels of Polynomial Degree


