Polynomial Tensor Sketch for Element-wise Function of Low-Rank Matrix

Insu Han [1] **Haim Avron** [2] **Jinwoo Shin** [3 1]

Abstract

This paper studies how to sketch element-wise functions of low-rank matrices. Formally, given low-rank matrix $A = [A_{ij}]$ and scalar non-linear function f, we aim for finding an approximated low-rank representation of the (possibly high-rank) matrix $[f(A_{ij})]$. To this end, we propose an efficient sketching-based algorithm whose complexity is significantly lower than the number of entries of A, i.e., it runs without accessing all entries of $[f(A_{ij})]$ explicitly. The main idea underlying our method is to combine a polynomial approximation of f with the existing tensor sketch scheme for approximating monomials of entries of A. To balance the errors of the two approximation components in an optimal manner, we propose a novel regression formula to find polynomial coefficients given A and f. In particular, we utilize a coreset-based regression with a rigorous approximation guarantee. Finally, we demonstrate the applicability and superiority of the proposed scheme under various machine learning tasks.

1. Introduction

Given a low-rank matrix $A = [A_{ij}] \in \mathbb{R}^{n\times n}$ with $A = UV^\top$ for some matrices $U, V \in \mathbb{R}^{n\times d}$ with $d \ll n$ and a scalar non-linear function $f : \mathbb{R} \to \mathbb{R}$, we are interested in the following element-wise matrix function:*

$$f^{\odot}(A) \in \mathbb{R}^{n\times n}, \quad \text{where } f^{\odot}(A)_{ij} := [f(A_{ij})]\,.$$

Our goal is to design a fast algorithm for computing tall-and-thin matrices T_U, T_V in time $o(n^2)$such that $f^{\odot}(A) \approx T_U T_V^\top$. In particular, our algorithm should run without computing all entries of A or $f^{\odot}(A)$ explicitly. As an side, we obtain a $o(n^2)$-time approximation scheme of $f^{\odot}(A)\boldsymbol{x} \approx T_U T_V^\top \boldsymbol{x}$ for an arbitrary vector $\boldsymbol{x} \in \mathbb{R}^n$ due to the associative property of matrix multiplication, where the exact computation of $f^{\odot}(A)\boldsymbol{x}$ requires the complexity of $\Theta(n^2)$.

The matrix-vector multiplication $f^{\odot}(A)\boldsymbol{x}$ or the low-rank decomposition $f^{\odot}(A) \approx T_U T_V^\top$ is useful in many machine learning algorithms. For example, the Gram matrices of certain kernel functions, e.g., polynomial and radial basis function (RBF), are element-wise matrix functions where the rank is the dimension of the underlying data. Such matrices are the cornerstone of so-called kernel methods and the ability to multiply the Gram matrix to a vector suffices for most kernel learning. The Sinkhorn-Knopp algorithm is a powerful, yet simple, tool for computing optimal transport distances (Cuturi, 2013; Altschuler et al., 2017) and also involves the matrix-vector multiplication $f^{\odot}(A)\boldsymbol{x}$ with $f(x) = \exp(x)$. Finally, $f^{\odot}(UV^\top)$ can also describe the non-linear computation of activation in a layer of deep neural networks (Goodfellow et al., 2016), where U, V and f correspond to input, weight and activation function (e.g., sigmoid or ReLU) of the previous layer, respectively.

Unlike the element-wise matrix function $f^{\odot}(A)$, a traditional matrix function $f(A)$ is defined on their eigenvalues (Higham, 2008) and possesses clean algebraic properties, i.e., it preserves eigenvectors. For example, given a diagonalizable matrix $A = PDP^{-1}$, it is defined that $f(A) = Pf^{\odot}(D)P^{-1}$. A classical problem addressed in the literature is of approximating the trace of $f(A)$ (or $f(A)\boldsymbol{x}$) efficiently (Han et al., 2017; Ubaru et al., 2017). However, these methods are not applicable to our problem because element-wise matrix functions are fundamentally different from the traditional function of matrices, e.g., they do not guarantee the spectral property. We are unaware of any approximation algorithm that targets general element-wise matrix functions. The special case of element-wise kernel functions such as $f(x) = \exp(x)$ and $f(x) = x^k$, as been address in the literature, e.g., using random Fourier features (RFF) (Rahimi & Recht, 2008; Pennington et al., 2015), Nyström method (Williams & Seeger, 2001), sparse greedy approximation (Smola & Schökopf,

[1]School of Electrical Engineering, KAIST, Daejeon, Korea [2]School of Mathematical Sciences, Tel Aviv University, Israel [3]Graduate School of AI, KAIST, Daejeon, Korea. Correspondence to: Jinwoo Shin <jinwoos@kaist.ac.kr>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

*In this paper, we primarily focus on the square matrix A for simplicity, but it is straightforward to extend our results to the case of non-square matrices.

2000) and sketch-based methods (Pham & Pagh, 2013; Avron et al., 2014; Meister et al., 2019; Ahle et al., 2020), just to name a few examples. We aim for not only designing an approximation framework for general f, but also outperforming the previous approximations (e.g., RFF) even for the special case $f(x) = \exp(x)$.

The high-level idea underlying our method is to combine two approximation schemes: TENSORSKETCH approximating the element-wise matrix function of monomial x^k and a degree-r polynomial approximation $p_r(x) = \sum_{j=0}^{r} c_j x^j$ of function f, e.g., $f(x) = \exp(x) \approx p_r(x) = \sum_{j=0}^{r} \frac{1}{j!} x^j$. More formally, we consider the following approximation:

$$f^{\odot}(A) \overset{\text{(a)}}{\approx} \sum_{j=0}^{r} c_j \cdot (x^j)^{\odot}(A) \overset{\text{(b)}}{\approx} \sum_{j=0}^{r} c_j \cdot \textsc{TensorSketch}\left((x^j)^{\odot}(A)\right),$$

which we call POLY-TENSORSKETCH. This is a linear-time approximation scheme with respect to n under the choice of $r = \mathcal{O}(1)$. Here, a non-trivial challenge occurs: a larger degree r is required to approximate an arbitrary function f better, while the approximation error of TENSORSKETCH is known to increase exponentially with respect to r (Pham & Pagh, 2013; Avron et al., 2014). Hence, it is important to choose good c_j's for balancing two approximation errors in both (a) and (b). The known (truncated) polynomial expansions such as Taylor or Chebyshev (Mason & Handscomb, 2002) are far from being optimal for this purpose (see Section 3.2 and 4.1 for more details).

To address this challenge, we formulate an optimization problem on the polynomial coefficients (the c_js) by relating them to the approximation error of POLY-TENSORSKETCH. However, this optimization problem is intractable since the objective involves an expectation taken over random variables whose supports are exponentially large. Thus, we derive a novel tractable alternative optimization problem based on an upper bound the approximation error of POLY-TENSORSKETCH. This problem turns out to be an instance of generalized ridge regression (Hemmerle, 1975), and as such has a closed-form solution. Furthermore, we observe that the regularization term effectively forces the coefficients to be exponentially decaying, and this compensates the exponentially growing errors of TENSORSKETCH with respect to the rank, while simultaneously maintaining a good polynomial approximation to the scalar function f given entries of A. We further reduce its complexity by regressing only a subset of the matrix entries found by an efficient coreset clustering algorithm (Har-Peledx, 2008): if matrix entries are close to the selected coreset, then the resulting coefficient is provably close to the optimal one. Finally, we construct the regression with respect to Chebyshev polynomial basis (instead of monomials), i.e., $p_r(x) = \sum_{j=0}^{r} c_j t_j(x)$ for the Chebyshev polynomial $t_j(x)$ of degree j, in order to resolve numerical issues arising for large degrees.

We evaluate the approximation quality of our algorithm under the RBF kernels of synthetic and real-world datasets. Then, we apply the proposed method to classification using kernel SVM (Cristianini et al., 2000; Scholkopf & Smola, 2001) and computation of optimal transport distances (Cuturi, 2013) which require to compute element-wise matrix functions with $f(x) = \exp(x)$. Our experimental results confirm that our scheme is at the order of magnitude faster than the exact method with a marginal loss on accuracy. Furthermore, our scheme also significantly outperforms a state-of-the-art approximation method, RFF for the aforementioned applications. Finally, we demonstrate a wide applicability of our method by applying it to the linearization of neural networks, which requires to compute element-wise matrix functions with $f(x) = \texttt{sigmoid}(x)$.

2. Preliminaries

In this section, we provide backgrounds on the randomized sketching algorithms COUNTSKETCH and TENSORSKETCH that are crucial components of the proposed scheme.

First, COUNTSKETCH (Charikar et al., 2002; Weinberger et al., 2009) was proposed for an effective dimensionality reduction of high-dimensional vector $\boldsymbol{u} \in \mathbb{R}^d$. Formally, consider a random hash function $h : [d] \to [m]$ and a random sign function $s : [d] \to \{-1, +1\}$, where $[d] := \{1, \ldots, d\}$. Then, COUNTSKETCH transforms $\boldsymbol{u}$ into $C_{\boldsymbol{u}} \in \mathbb{R}^m$ such that $[C_{\boldsymbol{u}}]_j := \sum_{i:h(i)=j} s(i)u_i$ for $j \in [m]$. The algorithm takes $\mathcal{O}(d)$ time to run since it requires a single pass over the input. It is known that applying the same[†] COUNTSKETCH transform on two vectors preserves the dot-product, i.e., $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \mathbf{E}\left[\langle C_{\boldsymbol{u}}, C_{\boldsymbol{v}} \rangle\right]$.

TENSORSKETCH (Pham & Pagh, 2013) was proposed as a generalization of COUNTSKETCH to tensor products of vectors. Given $\boldsymbol{u} \in \mathbb{R}^d$ and a degree k, consider i.i.d. random hash functions $h_1, \ldots, h_k : [d] \to [m]$ and i.i.d. random sign functions $s_1, \ldots, s_k : [d] \to \{-1, +1\}$, TENSORSKETCH applied to $\boldsymbol{u}$ is defined as the m-dimensional vector $T_{\boldsymbol{u}} \in \mathbb{R}^m$ such that $j \in [m]$,

$$[T^{(k)}_{\boldsymbol{u}}]_j := \sum_{H(i_1,\ldots,i_k)=j} s_1(i_1) \ldots s_k(i_k) u_{i_1} \ldots u_{i_k},$$

where $H(i_1, \ldots, i_k) \equiv h_1(i_1) + \cdots + h_k(i_k) \pmod{m}$.[‡] In (Pham & Pagh, 2013; Avron et al., 2014; Pennington et al., 2015; Meister et al., 2019), TENSORSKETCH was used for approximating of the power of dot-product between vectors.

[†] i.e., use the same hash and sign functions.

[‡] Unless stated otherwise, we define $T^{(0)}_{\boldsymbol{u}} := 1$.

Algorithm 1 TENSORSKETCH (Pham & Pagh, 2013)

Input: $U \in \mathbb{R}^{n\times d}$, degree k, sketch dimension m
Draw k i.i.d. random hash functions $h_1, \dots, h_k : [d] \to [m]$
Draw k i.i.d. random sign functions $s_1, \dots, s_k : [d] \to \{+1, -1\}$
$C_U^{(i)} \leftarrow$ COUNTSKETCH on each row of U using h_i and s_i for $i = 1, \dots, k$.
$T_U^{(k)} \leftarrow \text{FFT}^{-1}\left(\text{FFT}(C_U^{(1)}) \odot \cdots \odot \text{FFT}(C_U^{(k)})\right)$
Output : $T_U^{(k)}$

In other words, let $T_{\boldsymbol{u}}^{(k)}, T_{\boldsymbol{v}}^{(k)}$ be the same TENSORSKETCH on $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^d$ with degree k. Then, it holds that

$$\langle \boldsymbol{u}, \boldsymbol{v}\rangle^k = \left\langle \boldsymbol{u}^{\otimes k}, \boldsymbol{v}^{\otimes k}\right\rangle = \mathbf{E}\left[\left\langle T_{\boldsymbol{u}}^{(k)}, T_{\boldsymbol{v}}^{(k)}\right\rangle\right],$$

where $\otimes$ denotes the tensor product (or outer-product) and $\boldsymbol{u}^{\otimes k} := \boldsymbol{u} \otimes \cdots \otimes \boldsymbol{u} \in \mathbb{R}^{d^k}$ ($k-1$ times). This can be naturally extended to matrices $U, V \in \mathbb{R}^{n\times d}$. Suppose $T_U^{(k)}, T_V^{(k)} \in \mathbb{R}^{n\times m}$ are the same TENSORSKETCH on each row of U, V, and it follows that $(UV^\top)^{\odot k} = \mathbf{E}\left[T_U^{(k)} T_V^{(k)\top}\right]$ where $A^{\odot k} := [A_{ij}^k]$. Pham & Pagh (2013) devised a fast way to compute TENSORSKETCH using the Fast Fourier Transform (FFT) as described in Algorithm 1.

In Algorithm 1, $\odot$ is the element-wise multiplication (also called the Hadamard product) between two matrices of the same dimension and $\text{FFT}(C), \text{FFT}^{-1}(C)$ are the Fast Fourier Transform and its inverse applied to each row of a matrix $C \in \mathbb{R}^{n\times m}$, respectively. The cost of the FFTs is $\mathcal{O}(nm \log m)$ and so the total cost of Algorithm 1 is $\mathcal{O}(nk(d + m \log m))$ time since it runs FFT and COUNTSKETCH k times. Avron et al. (2014) proved a tight bound on the variance (or error) of TENSORSKETCH as follows.

Theorem 1 (Avron et al. (2014)) *Given $U, V \in \mathbb{R}^{n\times d}$, let $T_U^{(k)}, T_V^{(k)} \in \mathbb{R}^{n\times m}$ be the same* TENSORSKETCH *of U, V with degree $k \geq 0$ and sketch dimension m. Then, it holds*

$$\mathbf{E}\left[\left\|\left(UV^\top\right)^{\odot k} - T_U^{(k)} T_V^{(k)\top}\right\|_F^2\right] \leq \frac{(2+3^k)\left(\sum_i (\sum_j U_{ij}^2)^k\right)\left(\sum_i (\sum_j V_{ij}^2)^k\right)}{m}. \quad (2.1)$$

The above theorem implies the error of TENSORSKETCH becomes small for large sketch dimension m, but can grow fast with respect to degree k. Recently, Ahle et al. (2020) proposed another method whose error bound is tighter with respect to k, but can be worse with respect to another factor so-called statistical dimension (Avron et al., 2017).[§]

Algorithm 2 POLY-TENSORSKETCH

Input: $U, V \in \mathbb{R}^{n\times d}$, degree r, coefficient $c_0, \dots, c_r$, sketch dimension m
Draw r i.i.d. random hash functions $h_1, \dots, h_r : [d] \to [m]$
Draw r i.i.d. random sign functions $s_1, \dots, s_r : [d] \to \{+1, -1\}$
$T_U^{(1)}, T_V^{(1)} \leftarrow$ COUNTSKETCH of U, V using h_1 and s_1, respectively.
$F_U, F_V \leftarrow \text{FFT}(T_U^{(1)}),\ \text{FFT}(T_V^{(1)})$
$\Gamma \leftarrow c_0 T_U^{(0)} T_V^{(0)\top} + c_1 T_U^{(1)} T_V^{(1)\top}$
for $j = 2$ to r **do**
  $C_U, C_V \leftarrow$ COUNTSKETCH of U, V using h_j and s_j, respectively.
  $F_U, F_V \leftarrow \text{FFT}(C_U) \odot F_U,\ \text{FFT}(C_V) \odot F_V$
  $T_U^{(j)}, T_V^{(j)} \leftarrow \text{FFT}^{-1}(F_U),\ \text{FFT}^{-1}(F_V)$
  $\Gamma \leftarrow \Gamma + c_j T_U^{(j)} T_V^{(j)\top}$
end for
Output : Γ

3. Linear-time Approximation of Element-wise Matrix Functions

Given a scalar function $f : \mathbb{R} \to \mathbb{R}$ and matrices $U, V \in \mathbb{R}^{n\times d}$ with $d \ll n$, our goal is to design an efficient algorithm to find $T_U, T_V \in \mathbb{R}^{n\times d'}$ with $d' \ll n$ in $o(n^2)$ time such that

$$f^{\odot}(UV^\top) \approx T_U T_V^\top \in \mathbb{R}^{n\times n}.$$

Namely, we aim for finding a low-rank approximation of $f^{\odot}(UV^\top)$ without computing all n^2 entries of $UV^\top$. We first describe the proposed approximation scheme in Section 3.1 and provide further ideas for minimizing the approximation gap in Section 3.2.

3.1. POLY-TENSORSKETCH Transform

Suppose we have a polynomial $p_r(x) = \sum_{j=0}^r c_j x^j$ approximating $f(x)$, e.g., $f(x) = \exp(x)$ and its (truncated) Taylor series $p_r(x) = \sum_{j=0}^r \frac{1}{j!} x^j$. Then, we consider the following approximation scheme, coined POLY-TENSORSKETCH :

$$f^{\odot}(UV^\top) \overset{\text{(a)}}{\approx} \sum_{j=0}^r c_j (UV^\top)^{\odot j} \overset{\text{(b)}}{\approx} \sum_{j=0}^r c_j T_U^{(j)} T_V^{(j)\top}, \quad (3.1)$$

where $T_U^{(j)}, T_V^{(j)} \in \mathbb{R}^{n\times m}$ are the same TENSORSKETCH of U, V with degree j and sketch dimension m, respectively.

[§]In our experiments, TENSORSKETCH and the method by Ahle et al. (2020) show comparable approximation errors, but the latter one requires up to 4 times more computation.

Namely, our main idea is to combine (a) a polynomial approximation of a scalar function with (b) the randomized tensor sketch of a matrix. Instead of running Algorithm 1 independently for each $j \in [r]$, we utilize the following recursive relation to amortize operations:

$$T_U^{(j)} = \mathrm{FFT}^{-1}\left(\mathrm{FFT}(C_U) \odot \mathrm{FFT}(T_U^{(j-1)})\right),$$

where C_U is the CountSketch on each row of U whose randomness is independently drawn from that of $T_U^{(j-1)}$. Since each recursive step can be computed in $\mathcal{O}(n(d + m\log m))$ time, computing all $T_U^{(j)}$ for $j \in [r]$ requires $\mathcal{O}(nr(d + m\log m))$ operations. Hence, the overall complexity of Poly-TensorSketch, formally described in Algorithm 2, is $\mathcal{O}(n)$ if $r, d, m = \mathcal{O}(1)$.

Observe that multiplication of Γ (i.e., the output of Algorithm 2) and an arbitrary vector $\boldsymbol{x} \in \mathbb{R}^d$ can be done in $\mathcal{O}(nmr)$ time due to $\Gamma\boldsymbol{x} = \sum_{j=0}^r c_j T_U^{(j)} T_V^{(j)\top}\boldsymbol{x}$. Hence, for $r, m, d = \mathcal{O}(1)$, Poly-TensorSketch can approximate $f^{\odot}(UV^\top)\boldsymbol{x}$ in $\mathcal{O}(nr(d + m\log m)) = \mathcal{O}(n)$ time. As for the error, we prove the following error bound.

Proposition 2 *Given $U, V \in \mathbb{R}^{n\times d}$, $f : \mathbb{R} \to \mathbb{R}$, suppose that $|f(x) - \sum_{j=0}^r c_j x^j| \le \varepsilon$ in a closed interval containing all entries of $UV^\top$ for some $\varepsilon > 0$. Then, it holds that*

$$\mathbf{E}\left[\left\|f^{\odot}(UV^\top) - \Gamma\right\|_F^2\right] \le 2n^2\varepsilon^2 + \sum_{j=1}^r \frac{2rc_j^2(2+3^j)\left(\sum_i(\sum_k U_{ik}^2)^j\right)\left(\sum_i(\sum_k V_{ik}^2)^j\right)}{m}, \tag{3.2}$$

where Γ is the output of Algorithm 2.

The proof of Proposition 2 is given in the supplementary material. Note that even when ε is close to 0, the error bound (3.2) may increase exponentially with respect to the degree r. This is because the approximation error of TensorSketch grows exponentially with respect to the degree (see Theorem 1). Therefore, in order to compensate, it is desirable to use exponentially decaying (or even truncated) coefficient $\{c_j\}$. However, we are now faced with the challenge of finding exponentially decaying coefficients, while not hurting the approximation quality of f. Namely, we want to balance the two approximation components of Poly-TensorSketch: TensorSketch and a polynomial approximation for f. In the following section, we propose a novel approach to find the optimal coefficients minimizing the approximation error of Poly-TensorSketch.

3.2. Optimal Coefficient via Ridge Regression

A natural choice for the coefficients is to utilize a polynomial series such as Taylor, Chebyshev (Mason & Handscomb, 2002) or other orthogonal basis expansions (see Szego (1939)). However, constructing the coefficient in this way focuses on the error of the polynomial approximation of f, and ignores the error of TensorSketch which depends on the decay of the coefficient, as reflected in the error bound (3.2). To address the issue, we study the following optimization to find the optimal coefficient:

$$\min_{\boldsymbol{c}\in\mathbb{R}^{r+1}} \mathbf{E}\left[\left\|f^{\odot}(UV^\top) - \Gamma\right\|_F^2\right], \tag{3.3}$$

where Γ is the output of Poly-TensorSketch and $\boldsymbol{c} = [c_0, \dots, c_r]$ is a vector of the coefficient. However, it is not easy to solve the above optimization directly as its objective involves an expectation over random variables with a huge support, i.e., uniform hash and binary sign functions. Instead, we aim for minimizing an upper bound of the approximation error (3.3). To this end, we define the following notation.

Definition 3 *Let $X \in \mathbb{R}^{n^2\times(r+1)}$ be the matrix[¶] whose k-th column corresponds to the vectorization of $(UV^\top)_{ij}^{k-1}$ for all $i, j \in [n]$, $\boldsymbol{f} \in \mathbb{R}^{n^2}$ be the vectorization of $f^{\odot}(UV^\top)$ and $W \in \mathbb{R}^{(r+1)\times(r+1)}$ be a diagonal matrix such that $W_{11} = 0$ and for $i = 2, \dots, r+1$*

$$W_{ii} = \sqrt{\frac{r(2+3^i)(\sum_j(\sum_k U_{jk}^2)^i)(\sum_j(\sum_k V_{jk}^2)^i)}{m}}.$$

Using the above notation, we establish the following error bound.

Lemma 4 *Given $U, V \in \mathbb{R}^{n\times d}$ and $f : \mathbb{R} \to \mathbb{R}$, consider $X, \boldsymbol{f}$ and W defined in Definition 3. Then, it holds*

$$\mathbf{E}\left[\left\|f^{\odot}(UV^\top) - \Gamma\right\|_F^2\right] \le 2\|X\boldsymbol{c} - \boldsymbol{f}\|_2^2 + 2\|W\boldsymbol{c}\|_2^2, \tag{3.4}$$

where Γ is the output of Algorithm 2.

The proof of Lemma 4 is given in the supplementary material. Observe that the error bound (3.4) is a quadratic form of $\boldsymbol{c} \in \mathbb{R}^{r+1}$, where it is straightforward to obtain a closed-form solution for minimizing (3.4):

$$\begin{aligned}\boldsymbol{c}^* &:= \operatorname*{arg\,min}_{\boldsymbol{c}\in\mathbb{R}^{r+1}} \|X\boldsymbol{c} - \boldsymbol{f}\|_2^2 + \|W\boldsymbol{c}\|_2^2 \\ &= \left(X^\top X + W^2\right)^{-1} X^\top \boldsymbol{f}.\end{aligned} \tag{3.5}$$

This optimization task is also known as generalized ridge regression (Hemmerle, 1975). The solution (3.5) minimizes the regression error (i.e., the error of polynomial), while it is regularized by W, i.e., W_{ii} is a regularizer of c_i. Namely, if W_{ii} grows exponentially with respect to i, then c_i^* may

[¶] X is known as the Vandermonde matrix.

decay exponentially (this compensates the error of TENSORSKETCH with degree i). By substituting $\boldsymbol{c}^*$ into the error bound (3.4), we obtain the following multiplicative error bound of POLY-TENSORSKETCH.

Theorem 5 *Given $U, V \in \mathbb{R}^{n\times d}$ and $f : \mathbb{R} \to \mathbb{R}$, consider X defined in Definition 3. Then, it holds*

$$\mathbf{E}\left[\left\|f^{\odot}(UV^\top) - \Gamma\right\|_F^2\right] \le \left(\frac{2}{1+\frac{m\sigma^2}{rC}}\right)\|f^{\odot}(UV^\top)\|_F^2, \tag{3.6}$$

where Γ is the output of Algorithm 2 with the coefficient $\boldsymbol{c}^*$ in (3.5), $\sigma \ge 0$ is the smallest singular value of X and $C = \max(5\|U\|_F^2\|V\|_F^2, (2+3^r)(\sum_j(\sum_k U_{jk}^2)^r)(\sum_j(\sum_k V_{jk}^2)^r))$.

The proof of Theorem 5 is given in the supplementary material. Observe that the error bound (3.6) is bounded by $2\|f^{\odot}(UV^\top)\|_F^2$ since $m, r, \sigma, C \ge 0$. On the other hand, we recall that the error bound (3.2), i.e., POLY-TENSORSKETCH without using the optimal coefficient $\boldsymbol{c}^*$, can grow exponentially with respect to r.[‖] We indeed observe that $\boldsymbol{c}^*$ is empirically superior to the coefficient of the popular Taylor and Chebyshev series expansions with respect to the error of POLY-TENSORSKETCH (see Section 4.1 for more details). We also remark that the error bound (3.6) of POLY-TENSORSKETCH is even better than that (2.1) of TENSORSKETCH even for the case of monomial $f(x) = x^k$. This is primarily because the former is achievable by paying an additional cost for computing the optimal coefficient vector (3.5). In what follows, we discuss the extra cost.

3.3. Reducing Complexity via Coreset Regression

To obtain the optimal coefficient $\boldsymbol{c}^*$ in (3.5), one can check that $\mathcal{O}(r^2n^2 + r^3)$ operations are required because of computing $X^\top X$ for $X \in \mathbb{R}^{n^2\times(r+1)}$ (see Definition 3). This hurts the overall complexity of POLY-TENSORSKETCH. Instead, we choose a subset of entries in $UV^\top$ and approximately find the coefficient $\boldsymbol{c}^*$ based on the selected entries.

More formally, suppose $\overline{U} \in \mathbb{R}^{k\times d}$ is a matrix containing certain k rows of $U \in \mathbb{R}^{n\times d}$ and we use the following approximation:

$$\begin{aligned}\boldsymbol{c}^* &= \left(X^\top X + W^2\right)^{-1} X^\top \boldsymbol{f} \\ &\approx \left(\overline{X}^\top D\overline{X} + W^2\right)^{-1} \overline{X}^\top D\overline{\boldsymbol{f}},\end{aligned} \tag{3.7}$$

where $\overline{X}$, D and $\overline{\boldsymbol{f}}$ are defined as follows.[**]

[‖]Note that σ is decreasing with respect to r, and the error bound (3.6) may decrease with respect to r.

[**]Note that $\overline{X}$, $\overline{\boldsymbol{f}}$ are defined similar to X, f in Definition 3.

Algorithm 3 Greedy k-center

Input: $\boldsymbol{u}_1, \dots, \boldsymbol{u}_n \in \mathbb{R}^d$, number of clusters k
$a \leftarrow$ uniformly random in $\{1, 2, \dots, n\}$ and $S \leftarrow \{a\}$
$\Delta_i \leftarrow \|\boldsymbol{u}_i - \boldsymbol{u}_a\|_2$ for all $i \in [n]$
for $i = 2$ to k **do**
  $a \leftarrow \arg\max_i \Delta_i$ and $S \leftarrow S \cup \{a\}$
  $\Delta_i \leftarrow \min(\Delta_i, \|\boldsymbol{u}_i - \boldsymbol{u}_a\|_2)$ for all i
end for
$P(i) \leftarrow \arg\min_{j\in S} \|\boldsymbol{u}_i - \boldsymbol{u}_j\|_2$ for all i
Output: $\{\boldsymbol{u}_i : i \in S\}$, P

Algorithm 4 Coefficient approximation via coreset

Input: $U, V \in \mathbb{R}^{n\times d}$, a function f, number of clusters k, degree r
$\{\overline{\boldsymbol{u}}_i\}_{i=1}^k, P_1 \leftarrow$ Algorithm 3 with rows in U and k
$\{\overline{\boldsymbol{v}}_i\}_{i=1}^k, P_2 \leftarrow$ Algorithm 3 with rows in V and k
$\varepsilon_U \leftarrow \sum_i \left\|\boldsymbol{u}_i - \overline{\boldsymbol{u}}_{P_1(i)}\right\|_2$, $\varepsilon_V \leftarrow \sum_i \left\|\boldsymbol{v}_i - \overline{\boldsymbol{v}}_{P_2(i)}\right\|_2$
if $\varepsilon_U \sum_i \|\boldsymbol{v}_i\|_2 < \varepsilon_V \sum_j \|\boldsymbol{u}_j\|_2$ **then**
  $\overline{X}, D$ and $\overline{\boldsymbol{f}} \leftarrow$ matrices and vector defined in Definition 6 with $\overline{U}, V$ and P_1
else
  $\overline{X}, D$ and $\overline{\boldsymbol{f}} \leftarrow$ matrices and vector defined in Definition 6 with $U, \overline{V}$ and P_2
end if
Output : $\left(\overline{X}^\top D\overline{X} + W^2\right)^{-1} \overline{X}^\top D\overline{\boldsymbol{f}}$

Definition 6 *Let $\overline{X} \in \mathbb{R}^{kn\times(r+1)}$ be the matrix whose ℓ-th column is the vectorization of $(\overline{U}V^\top)_{ij}^{\ell-1}$ for $i \in [k], j \in [n]$ and $\overline{\boldsymbol{f}} \in \mathbb{R}^{kn}$ be the vectorization of $f^{\odot}(\overline{U}V^\top)$. Given a mapping $P : [n] \to [k]$, let $D \in \mathbb{R}^{kn\times kn}$ be a diagonal matrix where $D_{ii} = |\{s : P(s) = \lceil i/n \rceil\}|$.*

Computing (3.7) requires $\mathcal{O}(r^2kn + r^3) = \mathcal{O}(n)$ for $r, k = \mathcal{O}(1)$. In what follows, we show that if rows of U are close to those of $\overline{U}$ under the mapping P, then the approximation (3.7) becomes tighter. To this end, we say $\overline{U}$ is a ε-coreset of U for some $\varepsilon > 0$ if there exists a mapping $P : [n] \to [k]$ such that $\sum_{i=1}^n \left\|\boldsymbol{u}_i - \overline{\boldsymbol{u}}_{P(i)}\right\|_2 \le \varepsilon$, where $\boldsymbol{u}_i, \overline{\boldsymbol{u}}_i$ are the i-th rows of $U, \overline{U}$, respectively. Using the notation, we now present the following error bound of POLY-TENSORSKETCH under the approximated coefficient vector in (3.7).

Theorem 7 *Given $U, V \in \mathbb{R}^{n\times d}$ and $f : \mathbb{R} \to \mathbb{R}$, consider ε-coreset $\overline{U} \in \mathbb{R}^{k\times d}$ of U with $\overline{X}, D, \overline{\boldsymbol{f}}$ and P defined in Definition 6. For the approximated coefficient vector $\mathbf{c}$ in (3.7), assume that $(f(x) - \sum_{j=0}^r c_j x^j)^2$ is a L-Lipschitz*

function.[††] *Then, it holds*

$$\mathbf{E}\left[\left\|f^{\odot}(UV^\top)-\Gamma\right\|_2^2\right] \leq \frac{2}{\left(1+\frac{m\overline{\sigma}^2}{rC}\right)}\left\|\overline{\boldsymbol{f}}\right\|_2^2+2\varepsilon L\left(\sum_{i=1}^{n}\|\boldsymbol{v}_i\|_2\right), \quad (3.8)$$

where Γ *is the output of Algorithm 2 with* $\boldsymbol{c}$*,* $\boldsymbol{v}_i$ *is the* i*-th row of* V*,* $\overline{\sigma}$ *is the smallest singular value of* $D^{1/2}\overline{X}$ *and* $C := \max(5\|U\|_F^2\|V\|_F^2, (2+3^r)(\sum_j(\sum_k U_{jk}^2)^r)(\sum_j(\sum_k V_{jk}^2)^r))$.

The proof of Theorem 7 is given in the supplementary material. Observe that when ε is small, the error bound (3.8) becomes closer to that under the optimal coefficient (3.6). To find a coreset with small ε, we use the greedy k-center algorithm (Har-Peledx, 2008) described in Algorithm 3. We remark that the greedy k-center runs in $\mathcal{O}(ndk)$ time and it marginally increases the overall complexity of POLY-TENSORSKETCH in our experiments. Moreover, we indeed observe that various real-world datasets used in our experiments are well-clustered, which leads to a small ε (see supplementary material for details).

Algorithm 4 summarizes the proposed scheme for computing the approximated coefficient vector $\boldsymbol{c}$ in (3.7) with the greedy k-center algorithm. Here, we run the greedy k-center on both rows in U and V, and choose the one with a smaller value in the second term in (3.8), i.e., $\varepsilon_U\sum_i\|\boldsymbol{v}_i\|_2$ or $\varepsilon_V\sum_i\|\boldsymbol{u}_i\|_2$ (see also line 4-8 in Algorithm 4). We remark that Algorithm 4 requires $\mathcal{O}(nk(d+r^2)+r^3)$ operations, hence, applying this to POLY-TENSORSKETCH results in $\mathcal{O}(nk(d+r^2)+r^3)+\mathcal{O}(nr(d+m\log m))$ time in total. If one chooses $m,k,r=\mathcal{O}(1)$, the overall running time becomes $\mathcal{O}(nd)$ which is linear in the size of input matrix. For example, we choose $r,k,m=10$ in our experiments.

Chebyshev polynomial regression for avoiding a numerical issue. Recall that X contains $(UV^\top)_{ij}^r$ (see Definition 3). If entries in $UV^\top$ are greater (or smaller) than 1 and degree r is large, X can have huge (or negligible) values. This can cause a numerical issue for computing the optimal coefficient (3.5) (or (3.7)) using X. To alleviate the issue, we suggest to construct a matrix $X'\in\mathbb{R}^{n^2\times(r+1)}$ whose entries are the output of the Chebyshev polynomials: $(X')_{ij}=t_j((UV^\top)_{k\ell})$ where $k,\ell\in[n]$ corresponds to index $i\in[n^2]$, $j\in[r+1]$ and $t_j(x)$ is the Chebyshev polynomial of degree j (Mason & Handscomb, 2002). Now, the value of $t_j(x)$ is always in $[-1,1]$ and does not monotonically increase or decrease with respect to the degree j. Then, we find the optimal coefficient $\boldsymbol{c}'\in\mathbb{R}^{r+1}$ based on Chebyshev polynomials as follows:

$$\boldsymbol{c}'=\left(X'^\top X'+R^\top W^\top WR\right)^{-1}X'^\top\boldsymbol{f}, \quad (3.9)$$

where $R\in\mathbb{R}^{(r+1)\times(r+1)}$ satisfies that $\boldsymbol{c}^*=R\boldsymbol{c}'$ and can be easily computed. Specifically, it converts $\boldsymbol{c}'\in\mathbb{R}^{r+1}$ into the coefficient based on monomials, i.e., $\sum_{j=0}^r c_j^*x^j=\sum_{j=0}^r c_j't_j(x)$. We finally remark that to find t_j, one needs to find a closed interval containing all $(UV^\top)_{ij}$. To this end, we use the interval $[-a,a]$ where $a=(\max_i\|\boldsymbol{u}_i\|_2)(\max_j\|\boldsymbol{v}_j\|_2)$ and $\boldsymbol{u}_i$ is the i-row of the matrix U. It takes $\mathcal{O}(nd)$ time and contributes marginally to the overall complexity of POLY-TENSORSKETCH.

4. Experiments

In this section, we report the empirical results of POLY-TENSORSKETCH for the element-wise matrix functions under various machine learning applications.[‡‡] All results are reported by averaging over 100 and 10 independent trials for the experiments in Section 4.1 and those in other sections, respectively. Our implementation and experiments are available at `https://github.com/insuhan/polytensorsketch`.

4.1. RBF Kernel Approximation

Given $U=[\boldsymbol{u}_1,\ldots,\boldsymbol{u}_n]^\top\in\mathbb{R}^{n\times d}$, the RBF kernel $K=[K_{ij}]$ is defined as $K_{ij}:=\exp(-\|\boldsymbol{u}_i-\boldsymbol{u}_j\|_2^2/\gamma)$ for $i,j\in[n]$ and $\gamma>0$. It can be represented using the element-wise matrix exponential function:

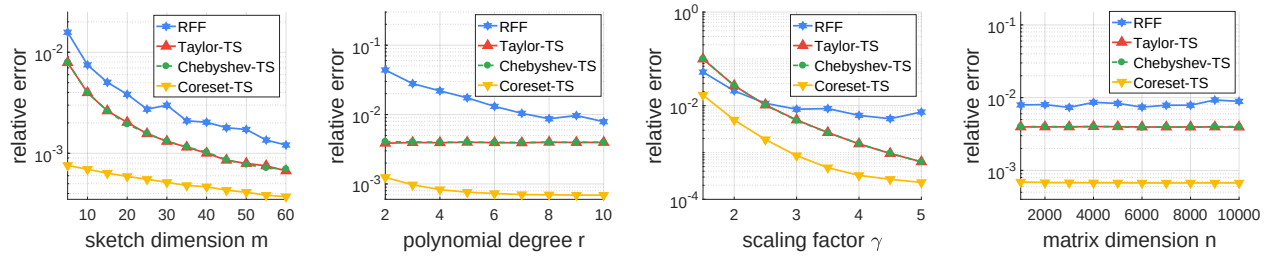
$$K=Z\exp^{\odot}(2UU^\top/\gamma)Z, \quad (4.1)$$

where Z is the n-by-n diagonal matrix with $Z_{ii}=\exp(-\|\boldsymbol{u}_i\|_2^2/\gamma)$. One can approximate the element-wise matrix function $\exp^{\odot}(2UU^\top/\gamma)\approx\Gamma$ where Γ is the output of POLY-TENSORSKETCH with $f(x)=\exp(2x/\gamma)$ and $K\approx Z\Gamma Z$. The RBF kernel has been used in many applications including classification (Pennington et al., 2015), covariance estimation (Wang et al., 2015), Gaussian process (Rasmussen, 2003), determinantal point processes (Affandi et al., 2014) where they commonly require multiplications between the kernel matrix and vectors.

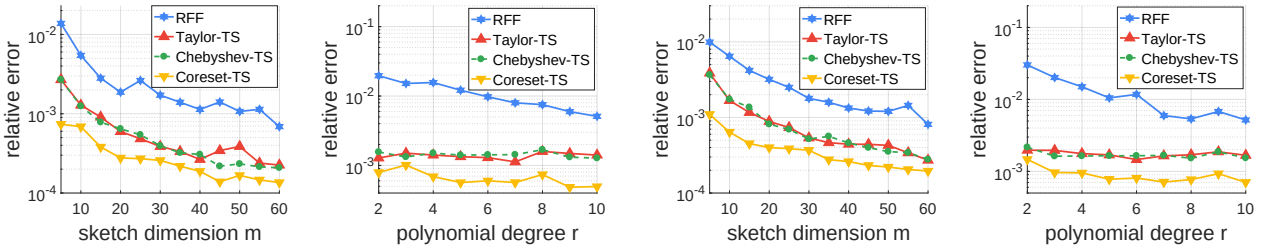
For synthetic kernels, we generate random matrices $U\in\mathbb{R}^{1,000\times50}$ whose entries are drawn from the normal distribution $\mathcal{N}(0,1/50)$. For real-world kernels, we use `segment` and `usps` datasets. We report the average of relative error for n^2 entries of K under varying parameters, i.e., sketch dimension m and polynomial degree r. We choose $m=10$, $r=10$ and $k=10$ as the default configuration.

[††] For some constant $L>0$, a function f is called L-Lipschitz if $|f(x)-f(y)|\leq L|x-y|$ for all x,y.

[‡‡] The datasets used in Section 4.1 and 4.2 are available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/` and `http://archive.ics.uci.edu/`.



(a) synthetic dataset



(b) `segment` dataset (c) `usps` dataset

Figure 1: Kernel approximation under (a) synthetic, (b) `segment` and (c) `usps` dataset. We set sketch dimension $m = 10$, polynomial degree $r = 10$ and coreset size $k = 10$ unless stated otherwise. All test methods have comparable running times.

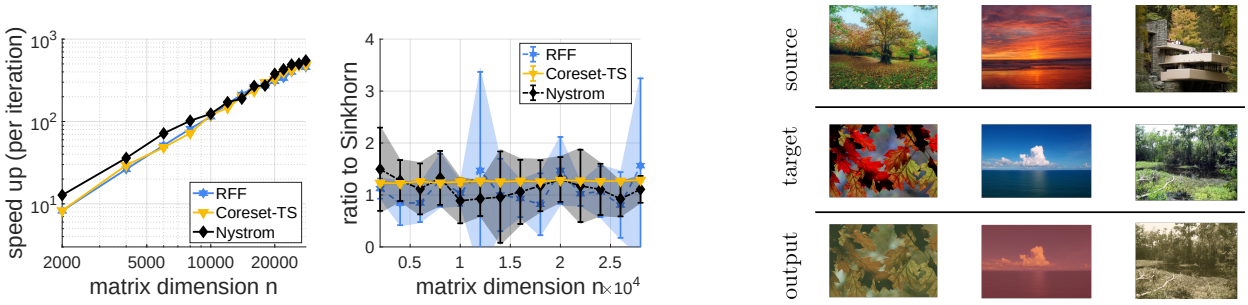
Table 1: Classification error, kernel approximation error and speedup for classification (training time) with kernel SVM under various real-world datasets.

Dataset	Statistics		Classification error (%)			Kernel approximation error		Speedup
	n	d	Exact	RFF	Coreset-TS	RFF	Coreset-TS	Coreset-TS
`segment`	2,310	19	3.20	3.28	3.22	1.7×10^{-3}	3.21×10^{-4}	8.26
`satimage`	4,335	36	23.54	24.11	23.74	6.5×10^{-3}	3.54×10^{-3}	10.06
`anuran`	7,195	22	37.26	40.33	37.36	6.1×10^{-3}	3.1×10^{-4}	49.45
`usps`	7,291	256	2.11	6.50	5.36	1.2×10^{-3}	1.61×10^{-4}	27.15
`grid`	10,000	12	3.66	18.27	15.99	2.34×10^{-3}	1.02×10^{-3}	4.23
`mapping`	10,845	28	15.04	18.57	17.35	6.09×10^{-3}	2.84×10^{-4}	4.17
`letter`	20,000	16	2.62	11.38	10.30	3.06×10^{3}	1.29	40.60

We compare our algorithm, denoted by Coreset-TS, (i.e., POLY-TENSORSKETCH using the coefficient computed as described in Section 3.3) with the random Fourier feature (RFF) (Rahimi & Recht, 2008) of the same computational complexity, i.e., its embedding dimension is chosen to mr so that the rank of their approximated kernels is the same and their running times are comparable. We also benchmark POLY-TENSORSKETCH using coefficients from Taylor and Chebyshev series expansions which are denoted by Taylor-TS and Chebyshev-TS, respectively. As reported in Figure 1, we observe that Coreset-TS consistently outperforms the competitors under all tested settings and datasets, where its error is often at orders of magnitude smaller than that of RFF. In particular, observe that the error of Coreset-TS tends to decreases with respect to the polynomial degree r, which is not the case for Taylor-TS and Chebyshev-TS (suboptimal versions of our algorithm). We additionally perform the POLY-TENSORSKETCH using the optimal coefficient and compare it with Coreset-TS in the supplementary material.

4.2. Classification with Kernel SVM

Next, we aim for applying our algorithm (Coreset-TS) to classification tasks using the support vector machine (SVM) based on RBF kernel. Given the input data $U \in \mathbb{R}^{n \times d}$, our algorithm can find a feature $T_U \in \mathbb{R}^{n \times m'}$ such that $K \approx T_U T_U^\top$ where K is the RBF kernel of U (see Section 4.1). One can expect that a linear SVM using T_U shows a similar performance compared to the kernel SVM using K. However, for $m' \ll n$, the complexity of a linear SVM is much cheaper than that of the kernel method both for training and testing. In order to utilize our algorithm, we construct $T_U = \left[\sqrt{c_0} T_U^{(0)} \ldots \sqrt{c_r} T_U^{(r)}\right]$ where $T_U^{(0)}, \ldots, T_U^{(r)}$ are the TENSORSKETCHs of U in Algorithm 2. Here, the coefficient c_j should be positive and one can compute the



(a) (b) (c)

Figure 2: Speedup per iteration (a) and the ratio to Sinkhorn objective (b) of our algorithm, RFF and Nystrom approximation averaged over 3 tested images in (c). The Sinkhorn algorithm transforms the color information of source images to targets.

optimal coefficient (3.5) by adding non-negativity condition, which is solvable using simple quadratic programming with marginal additional cost.

We use the open-source SVM package (LIBSVM) (Chang & Lin, 2011) and compare our algorithm with the kernel SVM using the exact RBF and the linear SVM using the embedded feature from RFF of the same running time with ours. We run all experiments with 10 cross-validations and report the average of the classification error on the validation dataset. We set $m = 20$ for the dimension of sketches and $r = 3$ for the degree of the polynomial. Table 1 summarizes the results of classification errors, kernel approximation errors and speedups of our algorithm under various real-world datasets. Ours (Coreset-TS) shows better classification errors compared to RFF of comparable running time and runs up to 50 times faster than the exact method. We also observe that large m can improve the performance (see the supplementary material).

4.3. Sinkhorn Algorithm for Optimal Transport

We apply Coreset-TS to the Sinkhorn algorithm for computing the optimal transport distance (Cuturi, 2013). The algorithm is the entropic regularization for approximating the distance of two discrete probability density. It is a fixed-point algorithm where each iteration requires multiplication of an element-wise matrix exponential function with a vector. More formally, given $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n$, and two distinct data points of the same dimension $\{\boldsymbol{x}_i\}_{i=1}^n$, $\{\boldsymbol{y}_j\}_{j=1}^n$, the algorithm iteratively computes $\boldsymbol{u} = \boldsymbol{a} \oslash (\exp^{\odot}(-\gamma D)\boldsymbol{v})$ and $\boldsymbol{v} = \boldsymbol{b} \oslash \left(\exp^{\odot}(-\gamma D)^\top \boldsymbol{u}\right)$ for some initial $\boldsymbol{v}$ where $D_{ij} := \|\boldsymbol{x}_i - \boldsymbol{y}_j\|_2^2$ and $\oslash$ denotes the element-wise division. Hence, the computation can be efficiently approximated using our algorithm or RFF. We also perform the Nyström approximation (Williams & Seeger, 2001) which was recently used for developing a scalable Sinkhorn algorithm (Altschuler et al., 2019). We provide all the details in the supplementary materials.

Given a pair of source and target images as shown in Figure 2 (c),* we randomly sample $\{\boldsymbol{x}_i\}_{i=1}^n$ from RGB pixels in the source image and $\{\boldsymbol{y}_j\}_{j=1}^n$ from those in the target image. We set $m = 20, d = 3, r = 3$ and $\gamma = 1$. Figure 2 (a) reports the speedup per iteration of the tested approximation algorithms over the exact computation and Figure 2 (b) shows the ratios the objective value of the approximated Sinkhorn algorithm to the exact one after 10 iterations. As reported in Figure 2 (a), all algorithms run at orders of magnitude faster than the exact Sinkhorn algorithm. Furthermore, the approximation ratio of ours is much more stable, while both RFF and Nyström have huge variances without any tendency on the dimension n.

Table 2: Test errors on CIFAR100 of linearization of 2 fully-connected layers of AlexNet, i.e., $W_2\texttt{sigmoid}^{\odot}(W_1\boldsymbol{x}) \approx (W_2 T_{W_1}) T_{\boldsymbol{x}}$ where both the input and hidden dimensions are $d = d_h = 1{,}024$ and the degree $r = 3$.

Model	Complexity	m	Error (%)
$W\boldsymbol{x}$	$\mathcal{O}(d)$	–	45.67
$(W_2 T_{W_1}) T_{\boldsymbol{x}}$	$\mathcal{O}(r(d + m \log m))$	100	53.09
		200	47.18
		400	43.90
		800	41.73
$W_2\texttt{sigmoid}^{\odot}(W_1\boldsymbol{x})$	$\mathcal{O}(d_h d)$	–	39.02

4.4. Linearization of Neural Networks

Finally, we demonstrate that our method has a potential to obtain a low-complexity model by linearization of a neural network. To this end, we consider 2 fully-connected layers in AlexNet (Krizhevsky et al., 2012) where each has 1,024 hidden nodes and it is trained on CIFAR100 dataset (Krizhevsky et al., 2009). Formally, given an input $\boldsymbol{x} \in \mathbb{R}^{1,024}$, its predictive class corresponds to $\arg\max_i [W_2\, \texttt{sigmoid}^{\odot}(W_1\boldsymbol{x})]_i$ where $W_1 \in \mathbb{R}^{1,024 \times 1,024}$, $W_2 \in \mathbb{R}^{1,024 \times 100}$ are model parameters. We first train the model for 300 epochs using ADAM opti-

*Images from https://github.com/rflamary/POT.

mizer (Kingma & Ba, 2015) with 0.0005 learning rate. Then, we approximate $\texttt{sigmoid}^{\odot}(W_1\boldsymbol{x}) \approx T_{W_1}T_{\boldsymbol{x}}$ for $T_{W_1} \in \mathbb{R}^{800\times mr}, T_{\boldsymbol{x}} \in \mathbb{R}^{mr}$ using Coreset-TS. After that, we fine-tune the parameters T_{W_1}, W_2 for 200 epochs and evaluate the final test error as reported in Table 2. We choose $r = 3$ and explore various $m \in \{100, 200, 400, 800\}$. Observe that the obtained model $(W_2T_{W_1})T_{\boldsymbol{x}}$ is linear with respect to $T_{\boldsymbol{x}}$ and its complexity (i.e., inference time) is much smaller than that of the original neural network. Moreover, by choosing the sketch dimension m appropriately, its performance is better than that of the vanilla linear model $W\boldsymbol{x}$ as reported in Table 2. The results shed a broad applicability our generic approximation scheme and more exploration on this line would be an interesting direction in the future.

5. Conclusion

In this paper, we design a fast algorithm for sketching element-wise matrix functions. Our method is based on combining (a) a polynomial approximation with (b) the randomized matrix tensor sketch. Our main novelty is on finding the optimal polynomial coefficients for minimizing the overall approximation error bound by balancing the errors of (a) and (b). We expect that the generic scheme would enjoy a broader usage in the future.

Acknowledgements

IH and JS were partially supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00075, Artificial Intelligence Graduate School Program (KAIST)) and the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921). HA was partially supported by BSF grant 2017698 and ISF grant 1272/17.

References

Affandi, R. H., Fox, E., Adams, R., and Taskar, B. Learning the parameters of determinantal point process kernels. In *International Conference on Machine Learning (ICML)*, 2014.

Ahle, T. D., Kapralov, M., Knudsen, J. B., Pagh, R., Velingker, A., Woodruff, D. P., and Zandieh, A. Oblivious sketching of high-degree polynomial kernels. In *Symposium on Discrete Algorithms (SODA)*, 2020.

Altschuler, J., Weed, J., and Rigollet, P. Near-linear time approximation algorithms for optimal transport via Sinkhorn iteration. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Altschuler, J., Bach, F., Rudi, A., and Niles-Weed, J. Massively scalable Sinkhorn distances via the Nyström method. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

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

Avron, H., Clarkson, K. L., and Woodruff, D. P. Sharper Bounds for Regularized Data Fitting. *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2017.

Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines. *Transactions on Intelligent Systems and Technology (TIST)*, 2011.

Charikar, M., Chen, K., and Farach-Colton, M. Finding frequent items in data streams. In *International Colloquium on Automata, Languages and Programming (ICALP)*, 2002.

Cristianini, N., Shawe-Taylor, J., et al. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

Han, I., Malioutov, D., Avron, H., and Shin, J. Approximating the Spectral Sums of Large-scale Matrices using Chebyshev Approximations. *SIAM Journal on Scientific Computing*, 2017.

Har-Peledx, S. Geometric approximation algorithms. *Lecture Notes for CS598, UIUC*, 2008.

Hemmerle, W. J. An explicit solution for generalized ridge regression. *Technometrics*, 1975.

Higham, N. J. *Functions of matrices: theory and computation*. SIAM, 2008.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*, 2015.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Krizhevsky, A. et al. Learning multiple layers of features from tiny images. *Citeseer*, 2009.

Mason, J. C. and Handscomb, D. C. *Chebyshev polynomials*. Chapman and Hall/CRC, 2002.

Meister, M., Sarlos, T., and Woodruff, D. Tight dimensionality reduction for sketching low degree polynomial kernels. *Advances in Neural Information Processing Systems (NIPS)*, 2019.

Pennington, J., Yu, F. X. X., and Kumar, S. Spherical random features for polynomial kernels. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

Pham, N. and Pagh, R. Fast and scalable polynomial kernels via explicit feature maps. In *Conference on Knowledge Discovery and Data Mining (KDD)*, 2013.

Rahimi, A. and Recht, B. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

Rasmussen, C. E. Gaussian processes in machine learning. In *Summer School on Machine Learning*. Springer, 2003.

Scholkopf, B. and Smola, A. J. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

Smola, A. J. and Schökopf, B. Sparse Greedy Matrix Approximation for Machine Learning. In *International Conference on Machine Learning (ICML)*, 2000.

Szego, G. *Orthogonal polynomials*. American Mathematical Soc., 1939.

Ubaru, S., Chen, J., and Saad, Y. Fast Estimation of tr(f(A)) via Stochastic Lanczos Quadrature. *SIAM Journal on Matrix Analysis and Applications*, 2017.

Wang, L., Zhang, J., Zhou, L., Tang, C., and Li, W. Beyond covariance: Feature representation with nonlinear kernel matrices. In *International Conference on Computer Vision (ICCV)*, 2015.

Weinberger, K., Dasgupta, A., Attenberg, J., Langford, J., and Smola, A. Feature hashing for large scale multitask learning. In *International Conference on Machine Learning (ICML)*, 2009.

Williams, C. K. and Seeger, M. Using the Nystrom method to speed up kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.