LEARNING APPROACH FOR FAST APPROXIMATE MATRIX FACTORIZATIONS

Haiyan Yu, Zhen Qin, Zhihui Zhu

Electrical and Computer Engineering, University of Denver, Denver, CO 80208 USA

ABSTRACT

Efficiently computing an (approximate) orthonormal basis and low-rank approximation for the input data $\boldsymbol{X}$ plays a crucial role in data analysis. One of the most efficient algorithms for such tasks is the randomized algorithm, which proceeds by computing a projection $\boldsymbol{XA}$ with a random sketching matrix $\boldsymbol{A}$ of much smaller size, and then computing the orthonormal basis as well as low-rank factorizations of the tall matrix $\boldsymbol{XA}$. While a random matrix $\boldsymbol{A}$ is the *de facto* choice, in this work, we improve upon its performance by utilizing a learning approach to find an adaptive sketching matrix $\boldsymbol{A}$ from a set of training data. We derive a closed-form formulation for the gradient of the training problem, enabling us to use efficient gradient-based algorithms. We also extend this approach for learning structured sketching matrix, such as the sparse sketching matrix that performs as selecting a few number of representative columns from the input data. Our experiments on both synthetical and real data show that both learned dense and sparse sketching matrices outperform the random ones in finding the approximate orthonormal basis and low-rank approximations.

***Index Terms*—** Low-rank matrix approximation, learning approach, sketching algorithm

1. INTRODUCTION

Listed as one of the "Top 10 Algorithms" that have greatest influence on the development and practice of science and engineering in the 20th century [1, 2], low-rank matrix factorization or decomposition plays a central role in data analysis and scientific computing, with representative applications including principal component analysis (PCA) [3, 4], seismic denoising [5–7], recommendation systems [8], solving PDEs [9, 10], etc.

While the factorizational approach for matrix computation remains fundamental, in the era of big data, the development of matrix factorization has focused more attention on efficiency than ever before. The reason is that existing computational tools such as SVD solvers already have very high accuracy, while the large-scale data are usually inaccurate, suggesting that the resolution is inherently limited by the imprecision of the data. On the other hand, in some cases where the data is too huge, it is prohibitive to directly apply existing solvers (such as SVD solvers) for the entire data.

Among all the fast methods for computing approximate matrix decompositions, one of the most representative approaches is the *randomized algorithm* [11–17]. Given an input data $\boldsymbol{X}$, randomized algorithms generally involve the following steps: first compute a projection $\boldsymbol{XA}$ with a random sketching matrix $\boldsymbol{A}$ of much smaller size, then compute the orthonormal basis $\boldsymbol{Q}$ of the tall matrix $\boldsymbol{XA}$, and finally compute the matrix factorizations and low-rank approximations if needed by performing on $\boldsymbol{Q}^\top\boldsymbol{X}$. See (1) in Section 2 for the details of efficiently computing an approximate rank-r approximation with the help of $\boldsymbol{Q}$. The main idea is to use a random matrix $\boldsymbol{A}$ (such as a random Gaussian matrix or a random sampling matrix) to preserve most of the action of the matrix $\boldsymbol{A}$, such that the resulting orthonormal basis $\boldsymbol{Q}$ captures the dominant range information of $\boldsymbol{A}$. See [16] for a contemporary survey and detailed introduction.

Recently, the work [18] showed that an adaptive sketching matrix $\boldsymbol{A}$ learned from a set of matrices (i.e., training data) can produce better performance than a random matrix for low-rank approximation. However, the approach is specific to low-rank approximation since it uses power method to compute approximate compact SVD and then applies auto-differential for computing the gradient without giving out explicit forms for the loss function nor the gradients. On the other hand, once $\boldsymbol{A}$ is learned, the testing procedure is different to the training one, as it computes the basis $\boldsymbol{XA}$ exactly with existing SVD solvers instead of the power method. This complicates the analysis of generalization for this approach.

In this work, we aim to learn the sketching matrix $\boldsymbol{A}$ that replaces the random matrix for efficiently computing an approximate basis, which can then be used for many tasks beyond low-rank approximation. Our contributions are summarized as follows:

- We propose a new objective for learning the sketching matrix $\boldsymbol{A}$ such that the orthornomal basis of $\boldsymbol{XA}$ captures most of the targeted subspace. We derive a closed-form formulation for the gradient of the training problem, enabling us to use efficient gradient-based algorithms.

This work was partially supported by NSF grants CCF-2008460 and CCF-2106881. Email: {haiyan.yu, zhen.qin, zhihui.zhu}@du.edu.

- We extend the learning approach for structured sketching matrices and propose projected gradient method for learning such matrices.
- We conduct experiments for finding an approximate orthonormal basis as well as low-rank approximations on both synthetical and real data, and show that both learned dense and sketch sketching matrices outperform the random ones.

The rest of the paper is organized as follows. Section 2 reviews randomized algorithms for approximate matrix factorizations. In Section 3, we introduce our learning approach. Section 4 presents experimetal results on both synthetical and real data.

2. APPROXIMATE MATRIX FACTORIZATIONS

Efficiently computing an orthonormal basis $\boldsymbol{Q} \in \mathbb{R}^{n_1 \times k}$ for the column space of the input matrix $\boldsymbol{X} \in \mathbb{R}^{n_1 \times n_2}$ can facilitate many applications in science and engineering. Some representative applications include performing principle component analysis, constructing rapid orthogonal approximate transform for sampled bandlimited signals [19], and computing various low-rank factorization (such as low-rank approximation, SVD and QR) [16]. Using low-rank approximation as an example, once an orthonormal basis $\boldsymbol{Q}$ is obtained, we can compute an approximate rank-r ($r \leq k$) approximation for $\boldsymbol{X}$ by the following two stages [20]:

$$\begin{aligned} &\text{Stage 1: Compute the best rank-}r\text{ approximation for} \\ &\qquad \text{matrix } \boldsymbol{Q}^\top \boldsymbol{X}\text{, denoted by } \mathrm{rank}_r(\boldsymbol{Q}^\top \boldsymbol{X}); \\ &\text{Stage 2: Form an approximate rank-}r\text{ approximation} \\ &\qquad \text{for } \boldsymbol{X} \text{ by } \boldsymbol{Q}\,\mathrm{rank}_r(\boldsymbol{Q}^\top \boldsymbol{X}). \end{aligned} \tag{1}$$

The main computational burden in the above two steps is computing $\mathrm{rank}_r(\boldsymbol{Q}^\top \boldsymbol{X})$, which can be computed by SVD and is more efficient than directly computing the rank-r approximation for $\boldsymbol{X}$ because $\boldsymbol{Q}^\top \boldsymbol{X}$ has size $k \times n_2$, much smaller than $n_1 \times n_2$.

However, directly computing the orthonormal basis $\boldsymbol{Q}$ for the column space of $\boldsymbol{X}$ is time consuming and has the same order of computational complexity as computing the matrix factorizations directly for $\boldsymbol{X}$. Fortunately, in the era of big data, it is common that the data are inaccurate, and thus an approximate basis and approximate low-rank factorizations will generally suffice since the resolution of the output is inherently limited by the imprecision of the data. In this case, we seek to find an approximate basis $\widetilde{\boldsymbol{Q}}$ that satisfies

$$\boldsymbol{X} \approx \widetilde{\boldsymbol{Q}}\widetilde{\boldsymbol{Q}}^\top \boldsymbol{X},\ \widetilde{\boldsymbol{Q}}^\top \widetilde{\boldsymbol{Q}} = \mathbf{I},$$

and can be constructed efficiently.

One of the most widely studied and used approaches for fast computing an approximate basis $\widetilde{\boldsymbol{Q}} \in \mathbb{R}^{n_2 \times m}$ (where $k \leq m \ll n_2$) is the randomized algorithm [16]. The main idea underlying this type of algorithms is to first project the matrix $\boldsymbol{X}$ onto a much smaller space by a random matrix $\boldsymbol{A} \in \mathbb{R}^{n_2 \times m}$ in the form of $\boldsymbol{X}\boldsymbol{A}$, and then construct a matrix $\widetilde{\boldsymbol{Q}}$ whose columns form an orthonormal basis for the range of $\boldsymbol{X}\boldsymbol{A}$. The second step can be efficiently computed via Gram-Schmidt orthogonalization, or SVD. The first step is essential to make this algorithm effective, requiring the dominant range information of $\boldsymbol{X}$ to be mostly preserved in the projected $\boldsymbol{X}\boldsymbol{A}$. A random matrix $\boldsymbol{A}$ has been proved to satisfy this requirement with high probability as long as m is slightly larger than k, with specific number of m controlling the approximation accuracy [16].

In this work, we aim to learn the sketching matrix $\boldsymbol{A}$ that replaces the random matrix for computing an approximate basis, which can then be used for many tasks beyond low-rank approximation. We will describe our approach in details in the next section.

3. LEARNING SKETCHING MATRIX FOR FAST COMPUTATION OF APPROXIMATE BASIS

Given a set of training matrices $\{\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_N \in \mathbb{R}^{n_1 \times n_2}\}$ sampled from a certain distribution $\mathcal{D}$, our goal is to learn a sketching matrix $\boldsymbol{A} \in \mathbb{R}^{n_2 \times m}$ that is able to preserve the column spaces of $\boldsymbol{X}_i$ once projecting them in the form of $\boldsymbol{X}_i\boldsymbol{A}$. To formally define our objective, let $\mathcal{U}_k(\cdot)$ be the operator that computes the first k left singular vectors. That is, $\mathcal{U}_k(\boldsymbol{X})$ consists of the first k left singular vectors of $\boldsymbol{X}$. To simplify the notation, we will drop the subscript and simply denote by $\mathcal{U}(\boldsymbol{X})$ an orthonormal basis for the column space of $\boldsymbol{X}$, i.e., $\mathcal{U}(\boldsymbol{X})$ can be computed by taking all the left singular vectors corresponding to the nonzero singular values of $\boldsymbol{X}$.

Before describing our objective, we first note that our goal is not to learn $\boldsymbol{A}$ such that $\mathcal{U}_k(\boldsymbol{X}) \approx \mathcal{U}_k(\boldsymbol{X}\boldsymbol{A})$ since it is the subspace instead of the specific basis that we are interested in. Instead, we hope the subspace spanned by $\mathcal{U}_k(\boldsymbol{X}\boldsymbol{A})$ captures most of the one spanned by $\mathcal{U}_k(\boldsymbol{X})$ in the sense that

$$\mathcal{U}_k(\boldsymbol{X}) \approx \mathcal{U}_k(\boldsymbol{X}\boldsymbol{A})\mathcal{U}_k(\boldsymbol{X}\boldsymbol{A})^\top \mathcal{U}_k(\boldsymbol{X}),$$

where $\mathcal{U}_k(\boldsymbol{X}\boldsymbol{A})\mathcal{U}_k(\boldsymbol{X}\boldsymbol{A})^\top$ is the projection onto the column space of $\mathcal{U}_k(\boldsymbol{X}\boldsymbol{A})$. This motivates us to learn $\boldsymbol{A}$ by solving the following learning problem

$$\min_{\boldsymbol{A} \in \mathbb{A}} \frac{1}{N} \sum_{i=1}^{N} \underbrace{\frac{1}{2} \left\| \left(\mathbf{I} - \mathcal{U}_k(\boldsymbol{X}_i\boldsymbol{A})\mathcal{U}_k(\boldsymbol{X}_i\boldsymbol{A})^\top\right) \mathcal{U}_k(\boldsymbol{X}_i) \right\|_F^2}_{f(\boldsymbol{A};\boldsymbol{X}_i)}, \tag{2}$$

where $\mathbb{A} \subset \mathbb{R}^{n_2 \times m}$ is the set of targeted sketching matrices. Note that the term $\left(\mathbf{I} - \mathcal{U}_k(\boldsymbol{X}_i\boldsymbol{A})\mathcal{U}_k(\boldsymbol{X}_i\boldsymbol{A})^\top\right)\mathcal{U}_k(\boldsymbol{X}_i)$ represents the projection of $\mathcal{U}_k(\boldsymbol{X}_i)$ onto the orthogonal complement of the column space of $\mathcal{U}_k(\boldsymbol{X}_i\boldsymbol{A})$, i.e., the residual that is not captured by the column space of $\mathcal{U}_k(\boldsymbol{X}_i\boldsymbol{A})$.

3.1. Learning Dense Sketching Matrices

We will solve (2) by gradient-based methods which are perhaps the most widely used algorithms for large-scale machine learning problems. Towards that end, we establish the gradient of the loss function in (2) in the following theorem.

Theorem 1. *Given $\boldsymbol{X} \in \mathbb{R}^{n_1 \times n_2}$, $\boldsymbol{A} \in \mathbb{R}^{n_2 \times m}$, let $\boldsymbol{XA} = \boldsymbol{USV}^\top$ be the compact SVD of $\boldsymbol{XA}$, where $\boldsymbol{U}, \boldsymbol{V}$ are orthonormal matrices with appropriate dimensions and $\boldsymbol{S} = \mathrm{diag}(s_1, \ldots, s_m)$ is a diagonal matrix. Then the gradient of $f(\boldsymbol{A}; \boldsymbol{X})$ in (2) is given by*

$$\nabla f(\boldsymbol{A}; \boldsymbol{X}) = \boldsymbol{X}^\top (\boldsymbol{\Gamma}_{\boldsymbol{U}} + \boldsymbol{\Gamma}_{\boldsymbol{U}^\perp}) \boldsymbol{V}^\top \tag{3}$$

where

$$\begin{aligned}
\boldsymbol{\Gamma}_{\boldsymbol{U}} &= \boldsymbol{U}(\boldsymbol{F} \odot (\boldsymbol{U}^\top \begin{bmatrix} \boldsymbol{P} & \boldsymbol{0} \end{bmatrix} - \begin{bmatrix} \boldsymbol{P} & \boldsymbol{0} \end{bmatrix}^\top \boldsymbol{U}))\boldsymbol{S} \\
\boldsymbol{\Gamma}_{\boldsymbol{U}^\perp} &= (\mathbf{I} - \boldsymbol{U}\boldsymbol{U}^\top) \begin{bmatrix} \boldsymbol{P} & \boldsymbol{0} \end{bmatrix} \boldsymbol{S}^{-1} \\
\boldsymbol{P} &= -(\boldsymbol{M}\mathcal{U}_k(\boldsymbol{X})^\top + \mathcal{U}_k(\boldsymbol{X})\boldsymbol{M}^\top)[\boldsymbol{U}]_k \\
\boldsymbol{M} &= (\mathbf{I} - [\boldsymbol{U}]_k [\boldsymbol{U}]_k^\top)\mathcal{U}_k(\boldsymbol{X}) \\
\boldsymbol{F}_{jl} &= \begin{cases} \frac{1}{s_l^2 - s_j^2}, & j \neq l, \\ 0, & \text{otherwise,} \end{cases}
\end{aligned}$$

$[\boldsymbol{U}]_k$ is the submatrix of $\boldsymbol{U}$ obtained by taking its first k column, and $\odot$ is the Hadamard product.

Theorem 1 is proved based on the differential of SVD [21]; due to limited space, its formal proof is omitted. One can then apply gradient descent (where the full gradient is $\frac{1}{N}\sum_{i=1}^N \nabla f(\boldsymbol{A}; \boldsymbol{X}_i)$) or stochastic gradient descent [22] for solving the learning problem (2).

3.2. Learning Structured Sketching Matrices

To further improve the computational efficiency, one may use a structured sketching matrix $\boldsymbol{A}$, such as a Toeplitz, circulant, or sparse matrix such that $\boldsymbol{XA}$ can be efficiently computed. Taking a sparse matrix as an example, if each column of $\boldsymbol{A}$ has only one nonzero element, which is known as a sketching matrix, then the computational time for $\boldsymbol{XA}$ is negligible. Denote by $\mathbb{A}_1$ the set of $n_2 \times m$ matrices that has at most one non-zero elements in each column, and $\mathcal{P}_{\mathbb{A}_1}$ the projection onto $\mathbb{A}_1$ that keeps the largest element (in magnitude) in each column. With gradient computed in Theorem 1, we can then use projected gradient descent for learning the structured sketching matrix:

$$\boldsymbol{A}_{t+1} = \mathcal{P}_{\mathbb{A}_1}\left(\boldsymbol{A}_t - \mu_t \frac{1}{N}\sum_{i=1}^N \nabla f(\boldsymbol{A}_t; \boldsymbol{X}_i)\right), \tag{4}$$

where μ_t is the step size. One can also replace the above full gradient by stochastic gradient to reduce computation time for each iteration.

4. EXPERIMENTS

In this section, we test the performance of the proposed method on both synthetical and real data.

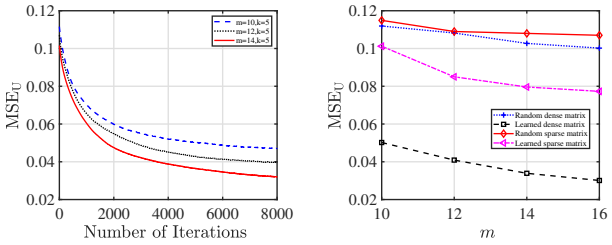
4.1. Synthetical data

In the first set of experiments, we generate each of N training matrices and M testing matrices as follows: first generate $\boldsymbol{L} \in \mathbb{R}^{n_1 \times \gamma}, \boldsymbol{R} \in \mathbb{R}^{n_2 \times \gamma}$ with each element from i.i.d. standard normal random distribution, and then construct a rank-γ matrix $\boldsymbol{X} = \boldsymbol{LR}^\top$. We set $N = 500$, $M = 200$, $n_2 = n_2 = 100$, and $\gamma = 7$. In all the experiments, we initialize the training algorithm with a random dense matrix $\boldsymbol{A}_0 \in \mathbb{R}^{100 \times m}$ with entries generated as i.i.d. standard normal random variables. We run gradient-based method 8000 number of iterations with step size $\mu = 2$.

Set $k = 5$. In Figure 1(a), we plot the convergence of gradient descent for learning a dense sketching matrix in terms of the following relative mean squared error (MSE):

$$\mathrm{MSE}_{\boldsymbol{U}} = \frac{1}{N}\sum_{i=1}^N \frac{\left\| \left(\mathbf{I} - \mathcal{U}_k(\boldsymbol{X}_i\boldsymbol{A})\mathcal{U}_k(\boldsymbol{X}_i\boldsymbol{A})^\top\right)\mathcal{U}_k(\boldsymbol{X}_i)\right\|_F^2}{\|\mathcal{U}_k(\boldsymbol{X}_i)\|_F^2} \tag{5}$$

where $\|\mathcal{U}_k(\boldsymbol{X}_i\boldsymbol{A})\|_F^2 = k$. The $\mathrm{MSE}_{\boldsymbol{U}}$ is the re-scaled version of the training error (2). Recall that the algorithm is initialized with a random Gaussian matrix. Thus, we note from Figure 1(a) that the training error decreases steadily for different choices m, implying that the learned sketching matrices improve upon the random ones, at least on the training data. We can also notice that larger m gives smaller training error, which is consistent with the fact that $\boldsymbol{XA}$ with larger number of columns can preserve more information of $\boldsymbol{X}$.



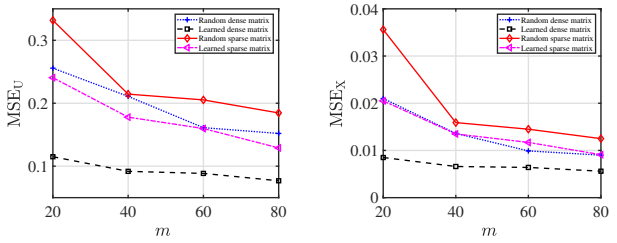
(a) $\mathrm{MSE}_{\boldsymbol{U}}$ in (5) for training (b) $\mathrm{MSE}_{\boldsymbol{U}}$ in (5) for testing

Fig. 1: (a) Convergence of gradient descent for learning dense sketching matrix, (b) comparison of the learned and random sketching matrices on the testing data.

We now use the testing data to verify the performance of two learned sketching matrices, the dense matrix learned by gradient descent and the sparse matrix (i.e, the sketch matrix) learned by the projected gradient descent illustrated in (4). In Figure 1(b), we compare the learned sketching matrices with random sketching matrices for different m in terms of $\mathrm{MSE}_{\boldsymbol{U}}$

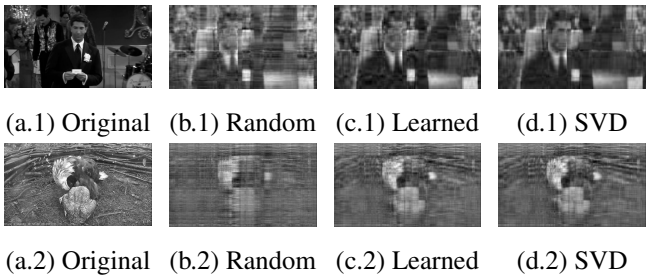
defined in (5) for the testing data. Here, a random dense matrix is generated by each entry being i.i.d. normal standard random variable, and a random sparse matrix is generated by first randomly selecting one support in each column, and then generating its value according to i.i.d. normal standard random distribution, with the rest of the entries being zero. We first observe that the learned dense sketching matrix achieves the best performance for all the choices of m. This is because the dense matrix has much more degrees of freedom to be optimized, thus more ability to adaptive to the data points. On the other hand, it is of interest to note that even the sparse sketching matrices outperforms both the random dense and sparse matrices, further verifying the efficiency of the proposed learning approaches.

4.2. Real data



(a) $\mathrm{MSE}_{\boldsymbol{U}}$ in (5) (b) $\mathrm{MSE}_{\boldsymbol{X}}$ in (6)

Fig. 2: Comparison of different random sketching matrices on the real data "Eagle" in terms of (a) basis approximation error $\mathrm{MSE}_{\boldsymbol{U}}$ defined in (5), (b) low-rank matrix approximation error $\mathrm{MSE}_{\boldsymbol{X}}$ defined in (6).



(a.1) Original (b.1) Random (c.1) Learned (d.1) SVD

(a.2) Original (b.2) Random (c.2) Learned (d.2) SVD

(a.3) Original (b.3) Random (c.3) Learned (d.3) SVD

Fig. 3: From left to right: the original image $\boldsymbol{X}$; an approximate rank-k image $\mathrm{rank}_k(\boldsymbol{X};\boldsymbol{A})$ where $\boldsymbol{A}$ is a dense random matrix; an approximate rank-k image $\mathrm{rank}_k(\boldsymbol{X};\boldsymbol{A})$ where $\boldsymbol{A}$ is the learned dense matrix; the best rank-k image $\mathrm{rank}_k(\boldsymbol{X})$. Here $k = 10$.

In this subsection, as in [18], we conduct experiments on 720×1280 images (or frames) extracted from three different videos, "Logo", "Friends", and "Eagle", downloaded from the Youtube [18]. For each video, we use the first 100 frames as training data, and the next 50 frames as testing data. In this experiment, we will use the learned sketching matrix for computing an approximate rank-k approximation by the following procedures: we first use the learned projection $\boldsymbol{A}$ to find the basis $\boldsymbol{Q} = \mathcal{U}(\boldsymbol{X}\boldsymbol{A})$, which is then plugged into (1) for computing an approximate best rank-k approximation for the testing matrix $\boldsymbol{X}$, denoted by $\mathrm{rank}_k(\boldsymbol{X};\boldsymbol{A}) = \mathcal{U}(\boldsymbol{X}\boldsymbol{A})\,\mathrm{rank}_k(\mathcal{U}(\boldsymbol{X}_i\boldsymbol{A})^\top\boldsymbol{X})$. We use $\mathrm{rank}_k(\boldsymbol{X})$ to denote the best rank-k approximation of $\boldsymbol{X}$. Thus, aside from the MSE_μ in (5), we will also compare the performance using the following relative MSE for the best rank-k approximations:

$$\mathrm{MSE}_{\boldsymbol{X}} = \frac{1}{M}\sum_{i=1}^{M}\frac{\|\mathrm{rank}_k(\boldsymbol{X}_i) - \mathrm{rank}_k(\boldsymbol{X}_i;\boldsymbol{A})\|_F^2}{\|\mathrm{rank}_k(\boldsymbol{X}_i))\|_F^2}, \quad (6)$$

where $\{\boldsymbol{X}_i\}$ are the set of training images. We set $k = 10$.

Figure 2 shows the performance of the learned and random sketching matrix in terms of $\mathrm{MSE}_{\boldsymbol{U}}$ and $\mathrm{MSE}_{\boldsymbol{X}}$ on the data "Logo". Similar to the results in the synthetical data, the learned dense sketching matrix significantly outperforms the random ones in both measures. We also note that the learned sparse matrix (which only has one non-zero entry in each column and has total m number of non-zero elements) not only outperforms the random sparse one, but also achieves on par performance with a random dense one (which has $mn^2 \gg m$ number of entries). Finally, we display the original images, their approximate rank-k approximations, and best rank-k approximations in Figure 3. We can observe from Figure 3 that the approximate rank-k approximations with learned sketching matrices have similar visualization quality as the best rank-k approximations obtained via SVD, and have better visualization quality than the one with random sketching matrices.

5. CONCLUSION

In this paper, we have proposed a learning approach for constructing a sketching matrix that is adaptive to the data set for fast computing a subspace that captures most of the action of a matrix. We derived a closed-form formulation for the corresponding training problem, and then used (projected) gradient descent for learning a (sparse) sketching matrix—the sparse one only has one non-zero elements in each column. The approximate orthonormal basis can then be used for efficiently finding other matrix factorizations, such as SVD, QR, and best low-rank approximations. Experiments on both synthetical and real data demonstrate the effectiveness of the proposed learning approach. Future work includes studying the generalization properties to support the superior experimental results.

6. REFERENCES

[1] J. Dongarra and F. Sullivan, "Guest editors' introduction: The top 10 algorithms," *IEEE Computer Architecture Letters*, vol. 2, no. 01, pp. 22–23, 2000.

[2] G. Stewart, "The decompositional approach to matrix computation," *Computing in Science & Engineering*, vol. 2, no. 1, pp. 50–59, 2000.

[3] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[4] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[5] W. Huang, R. Wang, Y. Chen, H. Li, and S. Gan, "Damped multichannel singular spectrum analysis for 3d random noise attenuation," *Geophysics*, vol. 81, no. 4, pp. V261–V270, 2016.

[6] C. Wang, Z. Zhu, H. Gu, X. Wu, and S. Liu, "Hankel low-rank approximation for seismic noise attenuation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 1, pp. 561–573, 2018.

[7] V. Oropeza and M. Sacchi, "Simultaneous seismic data denoising and reconstruction via multichannel singular spectrum analysis," *Geophysics*, vol. 76, no. 3, pp. V25–V32, 2011.

[8] F. O. Isinkaye, Y. Folajimi, and B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," *Egyptian informatics journal*, vol. 16, no. 3, pp. 261–273, 2015.

[9] L. Greengard and V. Rokhlin, "A new version of the fast multipole method for the laplace equation in three dimensions," *Acta numerica*, vol. 6, pp. 229–269, 1997.

[10] L. Grasedyck and W. Hackbusch, "Construction and arithmetics of h-matrices," *Computing*, vol. 70, no. 4, pp. 295–334, 2003.

[11] C. H. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, "Latent semantic indexing: A probabilistic analysis," *Journal of Computer and System Sciences*, vol. 61, no. 2, pp. 217–235, 2000.

[12] A. Frieze, R. Kannan, and S. Vempala, "Fast monte-carlo algorithms for finding low-rank approximations," *Journal of the ACM (JACM)*, vol. 51, no. 6, pp. 1025–1041, 2004.

[13] P. Drineas, R. Kannan, and M. W. Mahoney, "Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix," *SIAM Journal on computing*, vol. 36, no. 1, pp. 158–183, 2006.

[14] F. Woolfe, E. Liberty, V. Rokhlin, and M. Tygert, "A fast randomized algorithm for the approximation of matrices," *Applied and Computational Harmonic Analysis*, vol. 25, no. 3, pp. 335–366, 2008.

[15] T. Sarlos, "Improved approximation algorithms for large matrices via random projections," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pp. 143–152, IEEE, 2006.

[16] N. Halko, P.-G. Martinsson, and J. A. Tropp, "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions," *SIAM review*, vol. 53, no. 2, pp. 217–288, 2011.

[17] R. Witten and E. Candes, "Randomized algorithms for low-rank matrix factorizations: sharp performance bounds," *Algorithmica*, vol. 72, no. 1, pp. 264–281, 2015.

[18] P. Indyk, A. Vakilian, and Y. Yuan, "Learning-based low-rank approximations," *arXiv preprint arXiv:1910.13984*, 2019.

[19] Z. Zhu, S. Karnik, M. B. Wakin, M. A. Davenport, and J. Romberg, "Roast: Rapid orthogonal approximate slepian transform," *IEEE Transactions on Signal Processing*, vol. 66, no. 22, pp. 5887–5901, 2018.

[20] K. L. Clarkson and D. P. Woodruff, "Numerical linear algebra in the streaming model," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 205–214, 2009.

[21] J. Townsend, "Differentiating the singular value decomposition," tech. rep., Technical Report 2016, https://j-towns. github. io/papers/svd-derivative . . . , 2016.

[22] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *Siam Review*, vol. 60, no. 2, pp. 223–311, 2018.