PROBABILISTIC ALGORITHMS FOR CONSTRUCTING APPROXIMATE MATRIX DECOMPOSITIONS

ALEX NOWAK

Abstract. Low-rank matrix approximations are obliquous in many areas ranging from data analysis to scientific computing. From a data science point of view, probably the most important application is due to Principal Component Analysis (PCA), which aims to reveal hidden linear structure in massive datasets through a low-rank matrix decomposition. Consequently, the complexity of the algorithm plays a central role in the applicability of the algorithms to big data. The most common approximative factorization is the so-called truncated singular value decomposition (k-SVD) which can be computed in $\mathcal{O}(mnk)$ floating-point operations, where k is the target rank of the decomposition and m and n are the corresponding dimensions of the matrix. In this review, we introduce to the reader randomized algorithms that can achieve the aforementioned task with numerous advantadges compared to the classical algorithms. These randomized methods are based on the fact that the image of a low-rank matrix can be approximated by the action of the matrix to a reasonable amount of random vectors from the input space. Starting from this point, it is possible to develop algorithms that achieve a complexity of $\mathcal{O}(mn\log(k))$ for dense-matrices, matches the flop count of classical Krylov subspace methods for sparse matrices with a gain in robustness, and for large matrices that can not be stored in memory (RAM), they achieve a constant number of passes compared to the $\mathcal{O}(k)$ for classical algorithms.

1. Introduction

Matrix factorization is listed as one of the most influential set of techniques during the 20th century [3], among the Fast Fourier Transform, MCMC sampling methods and others. As Stewart [8] argues, the principle of the decompositional approach aims to construct computational platforms from which a variety of problems can be solved.

Although the decompositional approach to matrix computation remains fundamental, nowadays in the era of big data, most of the classical algorithms are inadequate to tackle most of the problems.

Data matrices are now incredibly big, making most of the classical approaches too expensive in terms of computation. Moreover, it is common in information sciences to have data which is missing or innaccurate. This gives the opportunity to sacrify some accuracy on the algorithm to gain on computation, which classical algorithms are not able to do. Another important aspect is the role of data transfer in the computational cost of a given algorithm, i.e, techniques that may perform fewer passes over the data may be substantially faster in practice.

In this review, we present, analyse and test *randomized* algorithms for matrix factorizations. This set of novel techniques addresses the issues stated above, i.e, can trade-off computation and accuracy to an arbitrary precision, gain on robustness, and reduce the number of passes on big datasets when data transfer is expensive.

The purpose of approximated low-rank matrix factorization is to factorize a given matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ into a product of two smaller matrices $\boldsymbol{B} \in \mathbb{R}^{m\times k}$ and $\boldsymbol{C} \in \mathbb{R}^{k\times n}$.

$$\begin{array}{ccccc} \boldsymbol{A} & \approx & \boldsymbol{B} & \boldsymbol{C}, \\ m\times n & & m\times k & k\times n. \end{array} \tag{1.1}$$

The matrix $\boldsymbol{B}\times\boldsymbol{C}$ in 1.1 is called a rank-k approximation of the matrix $\boldsymbol{A}$.

The inner dimension k is called the *numerical rank* of the matrix. This quantity differs from the *algebraic rank*, which is defined as the dimension of the image. The numerical rank is commonly defined as follows

As a project of the course: *Sparsity and Compressed Sensing* at ENS Cachan given by Pr. Gabriel Peyré.

$$r(\boldsymbol{A}) \coloneqq \frac{\|\boldsymbol{A}\|_F^2}{\|\boldsymbol{A}\|^2} = \sum_{j=1}^{\min(m,n)} \left(\frac{\sigma_j}{\sigma_1}\right)^2 \tag{1.2}$$

and it gives a better understanding of how accurate a rank-k approximation can be. Note that we always have $r(\boldsymbol{A}) \leq \operatorname{rank}(\boldsymbol{A})$. The notion of numerical rank appears in [10] and has been studied at the Theory Reading Group course of the master in depth [1].

The task of computing a low-rank approximation to a given matrix can be split into two computational stages. The first is to construct a low dimensional subspace that can capture the action of the matrix. The second is to restrict the matrix to the low dimensional subspace and then compute a standard factorization (QR, SVD, etc) of the reduced matrix.

- **Stage 1:** Compute an approximate basis for the range of the input matrix $\boldsymbol{A}$. We want to find a matrix $\boldsymbol{Q}$ with a small number of orthonormal columns such that

$$\boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A} \tag{1.3}$$

  The main idea is to approximate the range of the matrix via a randomized method. This can be accomplised by iteratively computing the image of random vectors from the input space and then orthogonalizing. All the randomness of these methods will belong to this first stage.
- **Stage 2:** Given a matrix $\boldsymbol{Q}$ that satisfies 1.3, compute a standard factorization (QR, SVD, etc.) of $\boldsymbol{A}$. Note that taking $\boldsymbol{B} = \boldsymbol{Q}$ and $\boldsymbol{C} = \boldsymbol{Q}^*\boldsymbol{A}$ we already have a low rank approximation of the matrix. There is no randomness at this stage and only classical linear algebra computations are incolved.

During the rest of the introduction, we will review some basics about matrix approximation and we will provide to the reader the basic aspects and insights of both stages, which will be further studied in depth in the main body.

1.1. Approximating the range of a matrix via randomness.

The problem of finding the best ϵ-approximation of a given matrix $\boldsymbol{A}$ is called the *fixed-precision approximation problem*. More concretely, we are given a tolerance ϵ and the goal is to find a matrix $\boldsymbol{Q}$ with $k = k(\epsilon)$ columns such that

$$\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| \leq \epsilon \tag{1.4}$$

The goal here is to find a $\boldsymbol{Q}$ with the smaller number of columns possible.

Another closely related problem is the so-called *fixed-rank approximation problem*, which seeks to find the best rank-k approximation of the matrix.

$$\min_{\operatorname{rank}(\boldsymbol{X}) \leq j} \|\boldsymbol{A} - \boldsymbol{X}\|. \tag{1.5}$$

[1] The comparison between both notions of rank can be better understood through the following caracterization. $\operatorname{rank}(\boldsymbol{A}) = \dim(\boldsymbol{A}B_2^n)$ and $r(\boldsymbol{A}) = d(\boldsymbol{A}B_2^n)$ where B_2^n is the euclidean ball, hence, $\boldsymbol{A}B_2^n$ is the ellipsoid with the axis of magnitude the singular values σ_j's. Here, $\dim(\cdot)$ denotes the algebraic dimension and $d(\cdot)$ denotes the *statistical dimension*. The statistical dimension is defined as $d(T) \coloneqq \frac{h(T-T)^2}{\operatorname{diam}(T)^2} \sim \frac{w(T)^2}{\operatorname{diam}(T)^2}$ where $h(T)^2 = \mathbb{E}\sup_{t\in T}\langle g, t\rangle^2$ and $w(T) = \mathbb{E}\sup_{t\in T}\langle g, t\rangle$ is the gaussian width. As discussed in [10], the statistical dimension is a more stable notion of dimension, in the same way that the numerical rank is more stable than the algebraic rank.

The Singular Value Decomposition (SVD) [2] is key to analyze this problem. Recall that the SVD of a matrix $\boldsymbol{A}$ is the following decomposition

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* = \sum_{j=1}^{\operatorname{rank}(\boldsymbol{A})} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^* \tag{1.6}$$

where $\{\boldsymbol{u}_k\}_k, \{\boldsymbol{v}_k\}_k$ are orthonormal basis on the output and input space respectively, $\sigma_1 \geq \sigma_2, \ldots, \sigma_{\operatorname{rank}(\boldsymbol{A})} \geq 0$ are the ordered singular values.

The SVD provides an optimal answer to the *fixed-rank approximation problem* [7] through the following important observation

$$\min_{\operatorname{rank}(\boldsymbol{X})\leq k} \|\boldsymbol{A} - \boldsymbol{X}\| = \sigma_{k+1}. \tag{1.7}$$

It is straightforward to check that the optimum is attained at $\boldsymbol{X}_* = \sum_{j=1}^{k} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^*$, namely, the k-truncated SVD (k-SVD) of the matrix $\boldsymbol{A}$. More precisely, we have that $\boldsymbol{B} = \boldsymbol{U}\boldsymbol{\Sigma}_{[k]}^{1/2}$ and $\boldsymbol{C} = \boldsymbol{\Sigma}_{[k]}^{1/2}\boldsymbol{V}^*$ are the best solutions 1.1 when the rank is fixed.

Let's suppose now that we know the desired rank k in advance. The goal is to find a matrix $\boldsymbol{Q}$ with $k+p$ orthonormal columns such that

$$\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| \approx \min_{\operatorname{rank}(\boldsymbol{X})\leq k} \|\boldsymbol{A} - \boldsymbol{X}\| \tag{1.8}$$

where p is called the *oversampling parameter*.

1.2. **Intuition of the randomized method to find Q.** The key observation that leverage these methods is the fact that the matrix $\boldsymbol{Q}$ can be found by sampling, and now the reader will obtain intuition on how this can be done.

Suppose we seek a basis for the range of a matrix $\boldsymbol{A}$ with algebraic rank k. Random elements $\boldsymbol{y}^{(i)}$ from the range can be computed by computing the image of random vectors $\boldsymbol{w}^{(i)}$ from the input space. Let us repeat this process k times:

$$\boldsymbol{y}^{(i)} = \boldsymbol{A}\boldsymbol{w}^{(i)}, \quad i = 1, \ldots, k \tag{1.9}$$

Thanks to the randomness, the set $\{\boldsymbol{w}^{(i)}\}_{i=1}^k$ is likely to be in general linear position and no vector will fall in $\ker \boldsymbol{A}$ if this is a set of measure zero under the probability measure we sampled from. Therefore, an orthogonalization procedure gives the desired orthonormal basis.

What happens if the matrix $\boldsymbol{A}$ has not exact algebraic rank equal to k [3]? Write $\boldsymbol{A} = \boldsymbol{B} + \boldsymbol{E}$ where $\boldsymbol{B}$ is a rank-k matrix containing the information we seek and $\boldsymbol{E}$ a small perturbation. We want a basis that covers the range of $\boldsymbol{B}$, however, if we repeat procedure 1.10, the vectors will be affected by the perturbation and the $\{\boldsymbol{y}^{(i)}\}_{i=1}^k$ will have small components that will make them fall outside the desired space.

To overcome this issue, the idea is to take p more samples:

$$\boldsymbol{y}^{(i)} = \boldsymbol{A}\boldsymbol{w}^{(i)} = \boldsymbol{B}\boldsymbol{w}^{(i)} + \boldsymbol{E}\boldsymbol{w}^{(i)}, \quad i = 1, \ldots, k+p \tag{1.10}$$

[2]The Singular Value Decomposition is a central tool for a data scientist. It is the algorithmic tool used to perform *Principal Component Analysis* (PCA) on a dataset $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ of size N and dimensionality d. PCA finds linear hidden linear structure in the data by looking at the eigenvectors with largest eigenvalues of the empirical covariance matrix of the data $\hat{\boldsymbol{\Sigma}}_N = \boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} = \frac{1}{N}\sum_{j=1}^N \boldsymbol{x}_j\boldsymbol{x}_j^{\mathrm{T}}$. These eigenvectors and eigenvalues can also be found by computing the SVD of the data matrix $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\mathrm{T}}$, because $\hat{\boldsymbol{\Sigma}}_N = \boldsymbol{V}\boldsymbol{D}^2\boldsymbol{V}^{\mathrm{T}}$. One important remark is that PCA does not suffer from the curse of dimensionality. For example, if we assume sub-gaussianity on the random variable X, then we have $\mathbb{E}\,\|\hat{\boldsymbol{\Sigma}}_N - \boldsymbol{\Sigma}\| \leq \varepsilon\|\boldsymbol{\Sigma}\|$ if we take a sample size of $N \sim \varepsilon^{-2}d$, i.e, the number of samples scales linearly with the dimensionality of the dataset. If instead of sub-gaussianity, we relax it to $\|X\|_2 \leq K(\mathbb{E}\,\|X\|_2^2)^{1/2}$ we only get an extra log-term. This is one of the main reasons why PCA works!

[3]In practice this is always the case.

The enriched set $\{\boldsymbol{y}^{(i)}\}_{i=1}^{k+p}$ has much more chance of spanning the desired subspace, and this is grounded with some theoretical results; see for example Theorem 3.11. The theory also shows that p can be quite small. In practice, $p = 5$ is enough.

Here in the box below, we illustrate the basic prodecure to find $\boldsymbol{Q}$ that approximates the range of $\boldsymbol{A}$ through randomization. We call it Proto-Algorithm.

PROTO-ALGORITHM: SOLVING THE FIXED-RANK PROBLEM

1 Draw a random $n \times (k + p)$ test matrix $\boldsymbol{\Omega}$.
2 Form the matrix product $\boldsymbol{Y} = \boldsymbol{A\Omega}$.
3 Construct a matrix $\boldsymbol{Q}$ whose columns form an orthonormal basis for the range of $\boldsymbol{Y}$.

1.3. **Construction of standard matrix factorizations from Q.** This corresponds to **Stage 2** of the algorithm. Once we have $\boldsymbol{Q}$ such that $\boldsymbol{A} \approx \boldsymbol{QQ}^*\boldsymbol{A}$, taking $\boldsymbol{B} = \boldsymbol{Q}$ and $\boldsymbol{C} = \boldsymbol{Q}^*\boldsymbol{A}$ we directly produce a low rank matrix decomposition $\boldsymbol{A} \approx \boldsymbol{BC}$. However, there are more involved methods to produce a decomposition from $\boldsymbol{Q}$ with computational advantadges compared to the direct method.

One we have $\boldsymbol{A} \approx \boldsymbol{BC}$, standard matrix decompositions (e.g. SVD, QR) can be easily derived as we show in 2.2.1.

A lot of questions need to be addressed in order to turn these methods into a technology or *off-the-shelf* algorithms. The following sections will we devoted to this.

2. Algorithms

In this section we will describe the randomized algorithms in detail, provide the corresponding computational complexity analysis, and state the main theoretical results that guarantee the accuracy of the approximation.

We split this section in three main parts: Stage 1 2.1 and Stage 2 2.2 that will study in depth the algorithmic details of both stages, and Full Algorithms 2.3 in which full algorithms will be presented for a diversity of matrices with certain properties.

2.1. **Stage 1.** In the introductory section, we provided some intuition on the randomized procedure and we developed a general Proto-Algorithm 1.2 to find the range-approximating matrix $\boldsymbol{Q}$. However, Proto-Algorithm 1.2 is very general and can be tunned depending on the problem requirements. The number T_{basic} of flops required by Proto-Algorithm 1.2 satisfies

$$T_{\text{basic}} \sim \ell n\, T_{\text{rand}} + \ell\, T_{\text{mult}} + \ell^2 m \tag{2.1}$$

where T_{rand} is the cost of generating a Gaussian random number and T_{mult} is the cost of multiplying $\boldsymbol{A}$ by a vector. The last term comes from the orthonormalization procedure of $\boldsymbol{Y}$.

We will now describe some specific realizations of Proto-Algorithm 1.2 that will be intended for problems with different requirements.

2.1.1. *Randomized Range Finder.* This is the most naive and simplest implementation of Proto-Algorithm 1.2. Given an oversampling parameter p, the *Randomized Range Finder* performs Proto-Algorithm 1.2 with a gaussian test matrix $\boldsymbol{\Omega} \in \mathbb{R}^{n\times\ell}$ with $\ell = k + p$ and k being a pre-specified target rank. Then, it orthogonalizes the rows of the resulting matrix $\boldsymbol{Y}$ by computing a QR decomposition. A numerical issue arises when computing the orthogonalization procedure due to the fact that the columns of $\boldsymbol{Y}$ are almost linearly dependent. The authors in [5] found that using the *double orthogonalization* [1] was enough to guarantee stability of the procedure.

The complexity analysis of the Algorithm 2.1.1 gives

$$T_{\text{Randomized Range Finder}} \sim \mathcal{O}(mn\ell) \tag{2.2}$$

Randomized Range Finder
1 Draw an $n \times \ell$ Gaussian random matrix $\boldsymbol{\Omega}$.
2 Form the $m \times \ell$ matrix $\boldsymbol{Y} = \boldsymbol{A\Omega}$.
3 Construct an $m \times \ell$ matrix $\boldsymbol{Q}$ whose columns form an orthonormal basis for the range of $\boldsymbol{Y}$, e.g., using the QR factorization $\boldsymbol{Y} = \boldsymbol{QR}$.

this is because generating a gaussian random number is $\mathcal{O}(1)$ and computing a matrix vector multiplication is $\mathcal{O}(mn)$ [4]. One important observation is that the complexity $\mathcal{O}(mn\ell)$ can be prohibitive for massive datasets [5]. A variant of this procedure will be studied in 2.1.4 to get around this problem,

2.1.2. *Adaptive Randomized Range Finder.* One important pitfall of the *Randomized Range Finder* 2.1.1 is that it requires to know in advance the target rank k. However, if we intend to solve the *fixed-precision approximation problem*, we need a scheme to estimate the error $\|\boldsymbol{A} - \boldsymbol{QQ}^*\boldsymbol{A}\|$ during the algorithm in order to match the requited tolerance ϵ.

This scheme is possible and it is direct consequence of the following lemma.

Lemma 2.1. *Let $\boldsymbol{B}$ be a real $m \times n$ matrix. Fix a positive integer r and a real number $\alpha > 1$. Draw an independent family $\{\boldsymbol{\omega}^{(i)} : i = 1, 2, \ldots, r\}$ of standard Gaussian vectors. Then*

$$\|\boldsymbol{B}\| \leq \alpha\sqrt{\frac{2}{\pi}} \max_{i=1,\ldots,r} \left\|\boldsymbol{B\omega}^{(i)}\right\|$$

except with probability α^{-r}.

Lemma 2.1 says that we can bound the error with high probability using inexpensive computations in an online manner. The Lemma 2.1 applied to our problem reads

$$\|(\mathbf{I} - \boldsymbol{QQ}^*)\boldsymbol{A}\| \leq 10\sqrt{\frac{2}{\pi}} \max_{i=1,\ldots,r} \left\|(\mathbf{I} - \boldsymbol{QQ}^*)\boldsymbol{A\omega}^{(i)}\right\| \tag{2.3}$$

with probability at least $1 - 10^{-r}$.

The high probability bound 2.3 gives a simple online scheme to decide when we have a good enough $\boldsymbol{Q}$ that matches the pre-specified tolerance. The goal here is to find an integer l and a $m \times l$ orthonormal matrix $\boldsymbol{Q}^{(l)}$ such that

$$\left\|\left(\mathbf{I} - \boldsymbol{Q}^{(\ell)}(\boldsymbol{Q}^{(\ell)})^*\right)\boldsymbol{A}\right\| \leq \varepsilon. \tag{2.4}$$

We call *Adaptive Randomized Range Finder* 2.1.2 to the algorithm derived from Lemma 2.1 that solves this problem.

One important question regarding Algorithm 2.1.2 is how good the bound given by Lemma 2.1 is in practice. If there is a significant gap between theory and practice the optimal l will be overestimated. This question will be addressed in the experimental section 4.2.

2.1.3. *Randomized Power Iteration.* The *Randomized Range Finder* 2.1.1 algorithm assumes that the singular values of the matrix decay fast. This can be intuitively seen from equation 1.10, where the small singular values interfere with the calculation of the range. This intuition is made precise in Theorem 3.11, where the error of the approximation depends on the σ_{k+1} and $\sum_{j>k} \sigma_j^2$.

The goal here is to reduce the weight of the small singular values by taking powers of the matrix whose range we want to approximate. Instead of applying the sampling scheme to $\boldsymbol{A}$, we will apply it to $\boldsymbol{B} = (\boldsymbol{AA}^*)^q\boldsymbol{A}$ where $q > 0$ is a small integer.

The matrix $\boldsymbol{B}$ has the same singular vectors than $\boldsymbol{A}$ (hence, the same range), but its singular values decay much more quickly.

[4]Note that parallel computation can be extremely helpful here to drastically reduce the effective complexity of the product $\boldsymbol{A\Omega}$.

[5]Obviously, we can't hope to do better than $\mathcal{O}(mn)$ which is the size of the matrix we want to approximate. The term that we want to address is the linear term in ℓ. Classical algorithms to compute the ℓ-SVD cost $\mathcal{O}(mn\ell)$, by first computing s rank-revealing QR factorization [4] and then manipulating the terms to get to the desired factorization.

Adaptive Randomized Range Finder

Draw standard Gaussian vectors $\boldsymbol{\omega}^{(1)}, \dots, \boldsymbol{\omega}^{(r)}$ of length n.
For $i = 1, 2, \dots, r$, compute $\boldsymbol{y}^{(i)} = \boldsymbol{A}\boldsymbol{\omega}^{(i)}$.
$j = 0$.
$\boldsymbol{Q}^{(0)} = [\,]$, the $m \times 0$ empty matrix.
while $\max\left\{ \left\|\boldsymbol{y}^{(j+1)}\right\|, \left\|\boldsymbol{y}^{(j+2)}\right\|, \dots, \left\|\boldsymbol{y}^{(j+r)}\right\| \right\} > \varepsilon/(10\sqrt{2/\pi})$,
    $j = j + 1$.
    Overwrite $\boldsymbol{y}^{(j)}$ by $\left(\mathbf{I} - \boldsymbol{Q}^{(j-1)}(\boldsymbol{Q}^{(j-1)})^*\right)\boldsymbol{y}^{(j)}$.
    $\boldsymbol{q}^{(j)} = \boldsymbol{y}^{(j)} / \left\|\boldsymbol{y}^{(j)}\right\|$.
    $\boldsymbol{Q}^{(j)} = [\boldsymbol{Q}^{(j-1)}\ \boldsymbol{q}^{(j)}]$.
    Draw a standard Gaussian vector $\boldsymbol{\omega}^{(j+r)}$ of length n.
    $\boldsymbol{y}^{(j+r)} = \left(\mathbf{I} - \boldsymbol{Q}^{(j)}(\boldsymbol{Q}^{(j)})^*\right)\boldsymbol{A}\boldsymbol{\omega}^{(j+r)}$.
    for $i = (j+1), (j+2), \dots, (j+r-1)$,
        Overwrite $\boldsymbol{y}^{(i)}$ by $\boldsymbol{y}^{(i)} - \boldsymbol{q}^{(j)}\left\langle \boldsymbol{q}^{(j)},\ \boldsymbol{y}^{(i)} \right\rangle$.
    end for
end while
$\boldsymbol{Q} = \boldsymbol{Q}^{(j)}$.

$$\sigma_j(\boldsymbol{B}) = \sigma_j(\boldsymbol{A})^{2q+1}, \qquad j = 1, 2, 3, \dots \tag{2.5}$$

The *Randomized Power Iteration* 2.1.3 algorithm is the same as the *Randomized Range Finder* 2.1.1 but replacing the formula $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$ by $\boldsymbol{Y} = \boldsymbol{B}\boldsymbol{\Omega}$.

Randomized Power Iteration

Draw an $n \times \ell$ Gaussian random matrix $\boldsymbol{\Omega}$.
Form the $m \times \ell$ matrix $\boldsymbol{Y} = (\boldsymbol{A}\boldsymbol{A}^*)^q\boldsymbol{A}\boldsymbol{\Omega}$ via alternating application of $\boldsymbol{A}$ and $\boldsymbol{A}^*$.
Construct an $m \times \ell$ matrix $\boldsymbol{Q}$ whose columns form an orthonormal basis for the range of $\boldsymbol{Y}$, e.g., via the QR factorization $\boldsymbol{Y} = \boldsymbol{Q}\boldsymbol{R}$.

The computational complexity of the Algorithm 2.1.3 is essentially the same because it only requires $2q + 1$ as many matrix-multiplications as Algorithm 2.1.1 but the number q is in practice 2,3 or 4. This can be seen from Corollary 3.14, which shows that the power iteration drives the approximation gap to 1 exponentially fast as q increases.

2.1.4. *Fast Randomized Range Finder.* A simple inspection to equation 2.1 reveals the computational bottleneck of the sampling procedure. This is the matrix multiplication $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$ that takes $\mathcal{O}(mn\ell)$ operations for dense matrices, which is the same as the ℓ-SVD.

The key idea is to use a *structured* random matrix that allows us to compute the product in $\mathcal{O}(mn\log(\ell))$ operations.

The simplest structured random matrix that meets our goals is the so-called *subsampled random Fourier transform* (SRFT).

An SRFT is an $n \times \ell$ matrix of the form

$$\boldsymbol{\Omega} = \sqrt{\frac{n}{\ell}}\,\boldsymbol{D}\boldsymbol{F}\boldsymbol{R}, \tag{2.6}$$

where

- $\boldsymbol{D}$ is an $n \times n$ diagonal matrix whose entries are independent random variables uniformly distributed on the complex unit circle.

- $\boldsymbol{F}$ is the $n \times n$ unitary discrete Fourier transform (DFT), whose entries take the values $f_{pq} = n^{-1/2}\, \mathrm{e}^{-2\pi \mathrm{i}(p-1)(q-1)/n}$ for $p, q = 1, 2, \ldots, n$
- $\boldsymbol{R}$ is an $n \times \ell$ matrix that samples ℓ coordinates from n uniformly at random, i.e., its ℓ columns are drawn randomly without replacement from the columns of the $n \times n$ identity matrix.

Now, via a subsampled FFT [11], we can compute the sample matrix $\boldsymbol{Y} = \boldsymbol{A\Omega}$ with $\mathcal{O}(mn \log(\ell))$ operations.

The total number of operations required by this procedure is reduced to

$$T_{\text{struct}} \sim mn \log(\ell) + \ell^2 n \tag{2.7}$$

Hence, the computational complexity of the approach is essentially $\mathcal{O}(mn \log(\ell))$.

Fast Randomized Range Finder	
1	Draw an $n \times \ell$ SRFT test matrix $\boldsymbol{\Omega}$, as defined by (2.6).
2	Form the $m \times \ell$ matrix $\boldsymbol{Y} = \boldsymbol{A\Omega}$ using a (subsampled) FFT.
3	Construct an $m \times \ell$ matrix $\boldsymbol{Q}$ whose columns form an orthonormal basis for the range of $\boldsymbol{Y}$, e.g., using the QR factorization $\boldsymbol{Y} = \boldsymbol{QR}$.

2.2. **Stage 2.** The output of the Stage 1 2.1 produces an orthonormal matrix $\boldsymbol{Q}$ whose range captures the action of the matrix $\boldsymbol{A}$. The goal of Stage 2 2.2 is to produce standard approximate matrix factorizations of $\boldsymbol{A}$ using this $\boldsymbol{Q}$.

This subsection is divided into three parts; first, we will show how to compute standard approximate matrix factorizations (SVD and QR) from a general approximate low-rank factorization. Recall that taking $\boldsymbol{B} = \boldsymbol{Q}$ and $\boldsymbol{C} = \boldsymbol{Q}^*\boldsymbol{A}$ we readily have a factorization that satisfies $\|\boldsymbol{A} - \boldsymbol{BC}\| \leq \varepsilon$. Then, we will describe in detail the *Direct SVD* algorithm, which will consist in constructing an SVD from $\boldsymbol{B}$ and $\boldsymbol{C}$. Finally, we will comment on other more involved methods that avoid computing the expensive product $\boldsymbol{C} = \boldsymbol{Q}^*\boldsymbol{A}$.

2.2.1. *Compute standard QR and SVD from a general factorization.* Now we will specify how we can compute the standards SVD and QR decompositions from a general low rank decomposition $\|\boldsymbol{A} - \boldsymbol{BC}\| \leq \varepsilon$ maintaining the tolerance ϵ from Stage 1 2.1.

- *SVD decomposition:* $\|\boldsymbol{A} - \boldsymbol{U\Sigma V}^*\| \leq \varepsilon$

  (1) Compute a QR factorization of $\boldsymbol{B}$ so that $\boldsymbol{B} = \boldsymbol{Q}_1\boldsymbol{R}_1$.
  (2) Form the product $\boldsymbol{D} = \boldsymbol{R}_1\boldsymbol{C}$, and compute an SVD: $\boldsymbol{D} = \boldsymbol{U}_2\boldsymbol{\Sigma V}^*$.
  (3) Form the product $\boldsymbol{U} = \boldsymbol{Q}_1\boldsymbol{U}_2$.

- *QR decomposition:* $\|\boldsymbol{A} - \boldsymbol{QR}\| \leq \varepsilon$

  (1) Compute a QR factorization of $\boldsymbol{B}$ so that $\boldsymbol{B} = \boldsymbol{Q}_1\boldsymbol{R}_1$.
  (2) Form the product $\boldsymbol{D} = \boldsymbol{R}_1\boldsymbol{C}$, and compute a QR factorization: $\boldsymbol{D} = \boldsymbol{Q}_2\boldsymbol{R}$.
  (3) Form the product $\boldsymbol{Q} = \boldsymbol{Q}_1\boldsymbol{Q}_2$.

2.2.2. *Direct SVD.* The procedure described in 2.2.1 to compute the approximate SVD decomposition without sacrificing error, defines what we call the *Direct SVD* 2.2.2.

Direct SVD	
1	Form the matrix $\boldsymbol{B} = \boldsymbol{Q}^*\boldsymbol{A}$.
2	Compute an SVD of the small matrix: $\boldsymbol{B} = \widetilde{\boldsymbol{U}}\boldsymbol{\Sigma V}^*$.
3	Form the orthonormal matrix $\boldsymbol{U} = \boldsymbol{Q}\widetilde{\boldsymbol{U}}$.

Although using *Direct SVD* 2.2.2 algorithm for Stage 2 2.2 does not incur additional errors, the computation of $\boldsymbol{C} = \boldsymbol{Q}^*\boldsymbol{A}$ is in general too expensive for dense matrices.

More concretely, the product costs $\mathcal{O}(mn\ell)$, even more expensive that the cost of Stage 1 2.1 when using the accelerated Algorithm 2.1.4.

2.2.3. *Different Procedures.*

- *Match the cost $\mathcal{O}(mn\log(\ell))$ of accelerated Stage 1* 2.1.4.
  In order to match the complexity $\mathcal{O}(mn\log(\ell))$ from Stage 1 2.1 we must avoid the product $\boldsymbol{Q}^*\boldsymbol{A}$.
  In [5], the authors propose algorithms based on row extraction of $\boldsymbol{Q}$ via its *Interpolative Decomposition* $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{Q}_{(J,:)}$ [2]. Now, $\boldsymbol{Q}_{(J,:)}$ is a $k \times k$ matrix. The proposed algorithm takes $\boldsymbol{Q}$ as input and constructs a rank-k matrix factorization

$$\boldsymbol{A} \approx \boldsymbol{X}\boldsymbol{B} \tag{2.8}$$

  where $\boldsymbol{B}$ is a $k \times n$ matrix consisting of k rows extracted from $\boldsymbol{A}$.
  The key here is that 2.8 can be produced without any large matrix-matrix multiplication resulting in a total of $\mathcal{O}(k^2(m+n))$ operations. The drawback is that the initial error is larger than the one incurred by $\boldsymbol{Q}^*\boldsymbol{Q}\boldsymbol{A}$ [6].
- *Single-Pass algorithms*
  The previously described algorithms require revisit the matrix $\boldsymbol{A}$ multiple times. In [5] they propose single-pass algorithms relying on the observation that all the information you need to compute the decomposition is in the matrices $\boldsymbol{\Omega}, \boldsymbol{Y}$ and $\boldsymbol{Q}$. They call the Algorithm *Eigenvalue Decomposition in One-pass*. However, this algorithm too adds additional error to the Stage 1 [7].

2.3. **Full Algorithms.** Let's now propose a full algorithm depending on the input matrix properties.

- *General Matrices That Fit in Core Memory*
  In this case, the appropiate method for Stage 1 is the accelerated Algorithm 2.1.4 that uses structured random matrices. For Stage 2, use the *row-extraction technique* described in 2.2.3. The overall cost T_{random} reduces to

$$\boxed{T_{\text{random}} \sim mn\log(k) + k^2(m+n)} \tag{2.9}$$

  The overall approximation error satisfies [8]

$$\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*\| \lesssim n \cdot \sigma_{k+1}, \tag{2.10}$$

- *Matrices for which Matrix-Vector Products are Cheap.*
  In this set of matrices we include for example sparse and structured matrices for which the matrix-vector multiplication is substantially cheaper than the general case.
  The appropiate method for Stage 1 is Algorithm 2.1.1 with p constant [9], or more generally Algorithm 2.1.3 with $q > 0$. For Stage 2, we apply the *Direct SVD* 2.2.2. The total cost T_{sparse} satisfies

$$\boxed{T_{\text{sparse}} = (2q+2)\,(k+p)\,T_{\text{mult}} + \mathrm{O}(k^2(m+n))} \tag{2.11}$$

  The overall approximation error satisfies (see Corollary 3.14).

$$\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^*\| \lesssim (kn)^{1/2(2q+1)} \cdot \sigma_{k+1}. \tag{2.12}$$

---

[6] In [5], they prove that the error is increased at most by $\left(\sqrt{1+4k(n-k)}\right)\varepsilon$, which can be large when ε is not that small or kn is large.

[7] Authors in [5] argue that this issue can be addressed with extra oversampling.

[8] This can be derived by combining the error analysis of range approximation with SRFT (which can be found in [5]) and the additional error incurred by the *row-extraction technique*.

[9] If we are dealing with the *fixed-precision problem*, then we use Algorithm 2.1.2.

3. Theory

3.1. **Analysis of Stage 1 2.1.** This section focuses on assessing the quality of the basis given by Proto-Algorithm 1.2. More precisely, we want to prove rigorous bounds on the approximation error

$$\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| \tag{3.1}$$

where $\|\cdot\|$ denotes either the operator norm or Frobenius norm.

We will split the argument into two parts [10]:

(1) Provide a generic error bound that depends on the interaction between the test matrix $\boldsymbol{\Omega}$ and the right and left singular values of $\boldsymbol{A}$. [11]
(2) Estimate the error using the distribution of the random matrix. We provide both expectation and probability tail bounds for the error.

3.1.1. *(1) Error bounds via Linear Algebra.* As we aim to compute a rank-k approximation of $\boldsymbol{A}$, we appropiately partition the exact SVD as

$$\boldsymbol{A} = \boldsymbol{U} \begin{array}{c} \begin{matrix} k & n-k \end{matrix} \\ \begin{bmatrix} \boldsymbol{\Sigma}_1 & \\ & \boldsymbol{\Sigma}_2 \end{bmatrix} \end{array} \begin{array}{c} \begin{matrix} n \end{matrix} \\ \begin{bmatrix} \boldsymbol{V}_1^* \\ \boldsymbol{V}_2^* \end{bmatrix} \end{array} \begin{matrix} k \\ n-k \end{matrix} \tag{3.2}$$

Now, let $\boldsymbol{\Omega}_i = \boldsymbol{V}_i\boldsymbol{\Omega}$ for $i = 1, 2$. Express $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$ as

$$\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega} = \boldsymbol{U} \begin{array}{c} \begin{matrix} \ell \end{matrix} \\ \begin{bmatrix} \boldsymbol{\Sigma}_1\boldsymbol{\Omega}_1 \\ \boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2 \end{bmatrix} \end{array} \begin{matrix} k \\ n-k \end{matrix}$$

where $\boldsymbol{\Sigma}_1\boldsymbol{\Omega}_1$ controls most of the action of $\boldsymbol{Y}$, and $\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2$ is a small perturbation.

The Proto-Algorithm 1.2 computes an orthogonal basis $\boldsymbol{Q}$ of $\mathrm{Im}(\boldsymbol{Y})$. In other words, we can express the orthogonal projection to $\mathrm{Im}(\boldsymbol{Y})$ as $\boldsymbol{P}_{\boldsymbol{Y}} = \boldsymbol{P}_{\mathrm{Im}(\boldsymbol{Y})} = \boldsymbol{Q}\boldsymbol{Q}^*$ [12]. The following Theorem 3.1 bounds the squared error provides a deterministic error bound to the squared error.

Theorem 3.1 (Deterministic error bound)**.** *We have that*

$$\|\!|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}|\!\|^2 \leq \|\!|\boldsymbol{\Sigma}_2|\!\|^2 + \left\|\!\left|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^{\dagger}\right|\!\right\|^2, \tag{3.3}$$

where $\|\!|\cdot|\!\|$ denotes either the spectral norm or the Frobenius norm.

Remark 3.2. Note that $\boldsymbol{\Sigma}_1$ does not appear in the error bound.

Remark 3.3. The first term is a deterministic clean error term; we want to compute a rank-k approximation so the error can not be smaller than this term. The second term is a random term that depends on the interaction of the right singular values of $\boldsymbol{A}$ amplified by $\boldsymbol{\Sigma}_2$.

We would also like to be able to analyze the power scheme described in 2.1.3, i.e, $\boldsymbol{B} = (\boldsymbol{A}\boldsymbol{A}^*)\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}^{2q+1}\boldsymbol{V}^*$. The rationale behind the power scheme was that the random approximation of the k-dimensional gross action of $\boldsymbol{A}$ can be improved if we amplify $\boldsymbol{\Sigma}_1 - \boldsymbol{\Sigma}_2$ by power iteration. This can be easily verified by the following Theorem 3.4.

Theorem 3.4 (Power scheme)**.** *Let $\boldsymbol{A}$ be an $m \times n$ matrix, and let $\boldsymbol{\Omega}$ be an $n \times \ell$ matrix. Fix a nonnegative integer q, form $\boldsymbol{B} = (\boldsymbol{A}\boldsymbol{A}^*)^q\boldsymbol{A}$, and compute the sample matrix $\boldsymbol{Z} = \boldsymbol{B}\boldsymbol{\Omega}$. Then*

$$\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{A}\| \leq \|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{B}\|^{1/(2q+1)}.$$

Remark 3.5. Let's consider the operator norm, i.e, $\|\boldsymbol{\Sigma}_1\| = \sigma_{k+1}$. Then

$$\|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{A}\| \leq \|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{B}\|^{1/(2q+1)} \leq \left(1 + \left\|\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^{\dagger}\right\|^2\right)^{1/(4q+2)} \sigma_{k+1}$$

so the power scheme shrinks the suboptimality exponentially fast.

[10] The authors argue that this bipartite proof is common in the literature of randomized linear algebra
[11] Note that we do not deal with randomness yet.
[12] We simplify the notation of the orthogonal projectoin to $\boldsymbol{P}_{\boldsymbol{Y}}$

Finally, we can ask what are the consequences of truncating the SVD of $\boldsymbol{P_Z A}$, i.e, compute its best rank-k approximation.

Theorem 3.6 (Analysis of Truncated SVD)**.** *Let $\boldsymbol{A}$ be an $m \times n$ matrix with singular values $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq \dots$, and let $\boldsymbol{Z}$ be an $m \times \ell$ matrix, where $\ell \geq k$. Suppose that $\widehat{\boldsymbol{A}}_{(k)}$ is a best rank-k approximation of $\boldsymbol{P_Z A}$ with respect to the spectral norm. Then*

$$\left\| \boldsymbol{A} - \widehat{\boldsymbol{A}}_{(k)} \right\| \leq \sigma_{k+1} + \|(\mathbf{I} - \boldsymbol{P_Z})\boldsymbol{A}\|.$$

Remark 3.7. The result of Theorem 3.6 is quite pessimistic, and in practice we observe that truncating the SVD is not that damaging in the randomized setting.

3.1.2. *(2) Bounds on the gaussian setting.* First we start by providing a bunch of results on gaussian matrices that will be key to prove the bounds on expectation and probability tails.

Proposition 3.8 (Expected norm of a scaled Gaussian matrix)**.** *Fix matrices $\boldsymbol{S}, \boldsymbol{T}$, and draw a standard Gaussian matrix $\boldsymbol{G}$. Then*

$$\left(\mathbb{E} \|\boldsymbol{SGT}\|_{\mathrm{F}}^2 \right)^{1/2} = \|\boldsymbol{S}\|_{\mathrm{F}} \|\boldsymbol{T}\|_{\mathrm{F}} \quad \text{and} \quad \mathbb{E} \|\boldsymbol{SGT}\| \leq \|\boldsymbol{S}\| \|\boldsymbol{T}\|_{\mathrm{F}} + \|\boldsymbol{S}\|_{\mathrm{F}} \|\boldsymbol{T}\|. \tag{3.4}$$

Proposition 3.9 (Expected norm of a pseudo-inverted Gaussian matrix)**.** *Draw a $k \times (k+p)$ standard Gaussian matrix $\boldsymbol{G}$ with $k \geq 2$ and $p \geq 2$. Then*

$$\left(\mathbb{E} \left\|\boldsymbol{G}^\dagger\right\|_{\mathrm{F}}^2 \right)^{1/2} = \sqrt{\frac{k}{p-1}} \quad \text{and} \quad \mathbb{E} \left\|\boldsymbol{G}^\dagger\right\| \leq \frac{\mathrm{e}\sqrt{k+p}}{p}. \tag{3.5}$$

Proposition 3.10 (Concentration for functions of a Gaussian matrix)**.** *Suppose that h is a Lipschitz function on matrices:*

$$|h(\boldsymbol{X}) - h(\boldsymbol{Y})| \leq L \|\boldsymbol{X} - \boldsymbol{Y}\|_{\mathrm{F}} \quad \text{for all } \boldsymbol{X}, \boldsymbol{Y}.$$

Draw a standard Gaussian matrix $\boldsymbol{G}$. Then

$$\mathbb{P}\left\{ h(\boldsymbol{G}) \geq \mathbb{E}\, h(\boldsymbol{G}) + Lt \right\} \leq \mathrm{e}^{-t^2/2}.$$

Now, we are ready to state and proof the main theorems in expectations, and afterwards we will confirm that the error does not oscillate too much around the mean by proving the corresponing bounds on the tails of the distribution.

Theorem 3.11 (Average error)**.** *The expected approximation error can be bounded as follows*

(1)
$$\mathbb{E} \|(\mathbf{I} - \boldsymbol{P_Y})\boldsymbol{A}\|_{\mathrm{F}} \leq \sigma_{k+1} \sqrt{\left(1 + \frac{k}{p-1}\right) r(\boldsymbol{\Sigma_2})}$$

(2)
$$\mathbb{E} \|(\mathbf{I} - \boldsymbol{P_Y})\boldsymbol{A}\| \leq \sigma_{k+1} \left(1 + \sqrt{\frac{k}{p-1}} + \frac{\mathrm{e}\sqrt{k+p}}{p} \sqrt{r(\boldsymbol{\Sigma_2})} \right)$$

One interesting quantity is worth examining is $\|(\mathbf{I} - \boldsymbol{P_Y})\boldsymbol{A}\| / \sigma_{k+1}$ to check the factor that tells how far the approximation is from the optimal rank-k approximation 1.5. We observe that the suboptimality term increases essentially as $\sim \sqrt{k/p}$ and has a term corresponding to the numerical rank 1.2 of the singular values corresponding to the perturbation. [13] We will go through the proof for the sake of illustration.

Proof. Hölder's inequality and Theorem 3.1 give

$$\mathbb{E} \|(\mathbf{I} - \boldsymbol{P_Y})\boldsymbol{A}\|_{\mathrm{F}} \leq \left(\mathbb{E} \|(\mathbf{I} - \boldsymbol{P_Y})\boldsymbol{A}\|_{\mathrm{F}}^2 \right)^{1/2} \leq \left(\left\|\boldsymbol{\Sigma}_2\right\|_{\mathrm{F}}^2 + \mathbb{E} \left\|\boldsymbol{\Sigma}_2 \boldsymbol{\Omega}_2 \boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}}^2 \right)^{1/2}.$$

[13] The original statement of Theorem 3.11 from [5] does not explicitly write $r(\boldsymbol{\Sigma_2})$. However, given that we introduced the concept of numerical rank 1.2 and its interpretation, I found interesting to highlight its appearance in the theorem.

Then, we condition on $\boldsymbol{\Omega}_1$ and use Proposition 3.8 and first part of Proposition 3.9

$$\mathbb{E}\left\|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}}^2 = \mathbb{E}\left(\mathbb{E}\left[\left\|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}}^2 \mid \boldsymbol{\Omega}_1\right]\right) = \mathbb{E}\left(\|\boldsymbol{\Sigma}_2\|_{\mathrm{F}}^2\left\|\boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}}^2\right) = \|\boldsymbol{\Sigma}_2\|_{\mathrm{F}}^2 \cdot \mathbb{E}\left\|\boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}}^2 = \frac{k}{p-1}\cdot\|\boldsymbol{\Sigma}_2\|_{\mathrm{F}}^2,$$

Putting everything together

$$\mathbb{E}\,\|(\mathbf{I}-\boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\|_{\mathrm{F}} \leq \left(1+\frac{k}{p-1}\right)^{1/2}\|\boldsymbol{\Sigma}_2\|_{\mathrm{F}}.$$

and the first part if proved.
The bound on the operator norm is very similar, Theorem 3.1 implies that

$$\mathbb{E}\,\|(\mathbf{I}-\boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\| \leq \mathbb{E}\left(\|\boldsymbol{\Sigma}_2\|^2 + \left\|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\right\|^2\right)^{1/2} \leq \|\boldsymbol{\Sigma}_2\| + \mathbb{E}\left\|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\right\|.$$

Conditioning again on $\boldsymbol{\Omega}_1$, we can bound the expectation w.r.t $\boldsymbol{\Omega}_2$

$$\mathbb{E}\left\|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\right\| \leq \mathbb{E}\left(\|\boldsymbol{\Sigma}_2\|\left\|\boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}} + \|\boldsymbol{\Sigma}_2\|_{\mathrm{F}}\left\|\boldsymbol{\Omega}_1^\dagger\right\|\right) \leq \|\boldsymbol{\Sigma}_2\|\left(\mathbb{E}\left\|\boldsymbol{\Omega}_1^\dagger\right\|_{\mathrm{F}}^2\right)^{1/2} + \|\boldsymbol{\Sigma}_2\|_{\mathrm{F}}\cdot\mathbb{E}\left\|\boldsymbol{\Omega}_1^\dagger\right\|.$$

Finally applying Proposition 3.9, we get to the final result

$$\mathbb{E}\left\|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\right\| \leq \sqrt{\frac{k}{p-1}}\,\|\boldsymbol{\Sigma}_2\| + \frac{\mathrm{e}\sqrt{k+p}}{p}\,\|\boldsymbol{\Sigma}_2\|_{\mathrm{F}}.$$

□

Finally, we will state the bounds on the tails that prove that the previously expectation bounds are representative of the random behavior.

Theorem 3.12 (Deviation bounds for the Frobenius error)**.** *Frame the hypotheses of Theorem 3.11. Assume further that $p \geq 4$. For all $u, t \geq 1$,*

$$\|(\mathbf{I}-\boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\|_{\mathrm{F}} \leq \left(1+t\cdot\sqrt{12k/p}\right)\left(\sum\nolimits_{j>k}\sigma_j^2\right)^{1/2} + ut\cdot\frac{\mathrm{e}\sqrt{k+p}}{p+1}\cdot\sigma_{k+1},$$

with failure probability at most $5t^{-p} + 2\mathrm{e}^{-u^2/2}$.

Theorem 3.13 (Deviation bounds for the spectral error)**.** *Frame the hypotheses of Theorem 3.11, and assume further that $p \geq 4$. Then*

$$\|(\mathbf{I}-\boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}\| \leq \left(1+8\sqrt{(k+p)\cdot p\log p}\right)\sigma_{k+1} + 3\sqrt{k+p}\left(\sum\nolimits_{j>k}\sigma_j^2\right)^{1/2},$$

with failure probability at most $6p^{-p}$.

Similar bounds can also be proven for the power scheme [5] that give a high probability guarantee for the bound 3.4.

Finally, let's analyze the error bounds on the the Power Scheme 2.1.3 that are directly derived from Theorem 3.4 using Hölder's inequality and the bound $\left\|\Sigma_2^{2q+1}\right\|_{\mathrm{F}} \leq \left(\sqrt{\min\{m,n\}-k}\right)\sigma_{k+1}^{1/(2q+1)}$.

Corollary 3.14 (Average spectral error for the power scheme)**.** *Frame the hypotheses of Theorem 3.11. Define $\boldsymbol{B} = (\boldsymbol{A}\boldsymbol{A}^*)^q\boldsymbol{A}$ for a nonnegative integer q, and construct the sample matrix $\boldsymbol{Z} = \boldsymbol{B}\boldsymbol{\Omega}$. Then*

$$\mathbb{E}\,\|(\mathbf{I}-\boldsymbol{P}_{\boldsymbol{Z}})\boldsymbol{A}\| \leq \left[1+\sqrt{\frac{k}{p-1}}+\frac{\mathrm{e}\sqrt{k+p}}{p}\cdot\sqrt{\min\{m,n\}-k}\right]^{1/(2q+1)}\sigma_{k+1}.$$

From Corollary 3.14 we observe how as we increase q, the power scheme drives the extra factor in the error to one exponentially fast.

4. Experiments

In this section, we briefly present our experiments on some of the algorithms presented at the previous sections. Two sets of experiments are presented.

The first tests are on powers of random gaussian matrices of different sizes. The goal of this set of numerics is to check the performance of the algorithms and make sure they are working as expected. We also analyze the sharpness of the bounds dictated by the theory. In particular, the bound on the expectation of the error given by 3.11 and the error estimation procedure which was motivated by Lemma 2.1. We have used an *oversampling parameter* $p = 5$ for all of the experiments.

The second set of experiments analyses the performance of the *Randomized Power Method* 2.1.3 on MNIST and on a matrix appearing in image processing.

4.1. **Details of the implementation.** The experiments have been implemented in `Matlab`. The reproducible source code can be found at the following Github repository together with the LaTeX of the report. Please read `README` documentation to run the code on the `Matlab` interactive command line.

4.2. **Gaussian Matrices.** In this set of experiments, we test and analyze the performance of the Algorithms 2.1.1, 2.1.2, 2.1.3 and 2.1.4.

The experiments are performed on powers of gaussian random matrices of the form:

$$\boldsymbol{A} = \frac{1}{\sqrt{m}+\sqrt{n}}\boldsymbol{G}, \qquad G_{ij} \sim N(0,1) \tag{4.1}$$

The normalization in 4.1 is to make sure that the norm of $\boldsymbol{A}$ is around 1 with high probability [14]. As seen from Equation 2.5, the singular values decay faster for higher powers of the matrix.

4.2.1. *Experiments on Randomized Range Finder 2.1.1.* See Figure 4.2.1.

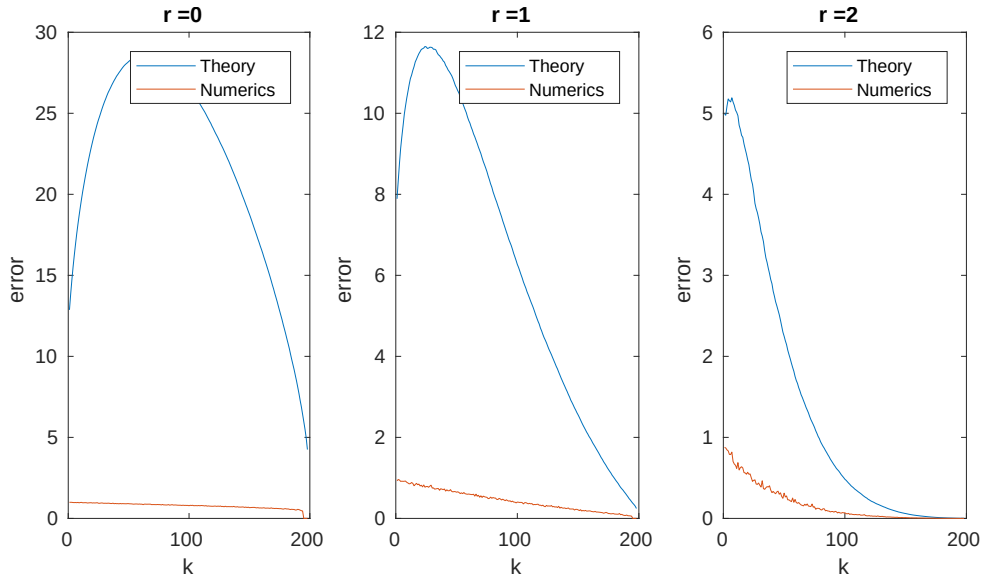

Figure 1. Comparison between theoretical mean bound 3.11 and numerical error 3.1 produced by Algorithm 2.1.1 for $(\boldsymbol{A}\boldsymbol{A}^*)^r\boldsymbol{A}$ with $r = 0, 1, 2$.

[14] This is a direct consequence of Sudakov-Fernique's inequality that compares the supremum of two gaussian processes when one is dominated by the other. More precisely, you have the bound on the expectation of the norm $\mathbb{E}\,\|\boldsymbol{G}\| \leq \sqrt{m}+\sqrt{n}$ and also an accompanying tail bound $\mathbb{P}\left\{\|\boldsymbol{G}\| \geq \sqrt{m}+\sqrt{n}+t\right\} \leq 2\exp(-ct^2)$. Using Gordon's inequality (generalization of Sudakov Fernique's), you can also prove lower bounds on the smallest singular value $\mathbb{E}\,\|\boldsymbol{G}^\dagger\| \geq \sqrt{m}-\sqrt{n}$ and $\mathbb{P}\left\{\|\boldsymbol{G}\| \leq \sqrt{m}-\sqrt{n}-t\right\} \leq 2\exp(-ct^2)$. These results on concentration of measure have been studied at the Theory Reading Group following the book on High Dimensional Probability from R.Vershynin which I highly recommend [10].

4.2.2. *Experiments on Randomized Power Iteration 2.1.3.* See Figure 4.2.2.

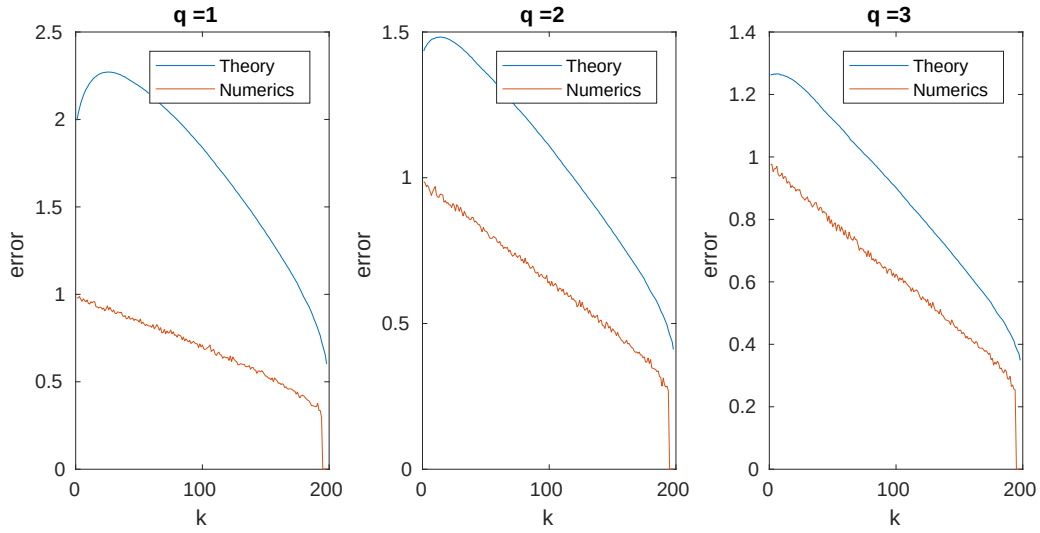

FIGURE 2. Comparison between theoretical mean bound from Corollary 3.14 and numerical error 3.1 produced by Algorithm 2.1.2 for $\boldsymbol{A}$ and $q = 1, 2, 3$

4.2.3. *Experiments on Adaptive Randomized Range Finder 2.1.2.* In this experiment, Algorithm 2.1.2 is applied to $(\boldsymbol{A}\boldsymbol{A}^*)^2\boldsymbol{A}$ with pre-specified tolerances ranging from 0 to 1. We have fitted a line to the numerical error produced vs the tolerance (the dependence was close to linear). The slope of the resulting line is **0.045**, i.e, for the matrices used in the experiment, the bound 2.3 has a numerical suboptimality of 0.045.

4.2.4. *Experiments on Fast Randomized Range Finder 2.1.4.* Algorithm 2.1.4 has been applied to several powers of $\boldsymbol{A}$ (using SRFT test matrices) and the error curve for different target ranks is essentially the same as the one found using Algorithm 2.1.1.

4.3. **Real Datasets.**

4.3.1. *MNIST.* In this experiment, we used 10,000 examples from the MNIST dataset [6] of 28x28 black and white images of handwritten digits.

The goal is to compare Algorithm 2.1.1 with the power version 2.1.3 on the matrix $\boldsymbol{M} \in \mathbb{R}^{10000\times 784}$ where each row is a flattened image. The singular values of $\boldsymbol{M}$ decay fast, hence, the power scheme does not significantly improve the randomized scheme for $q > 1$. See Figure 4.3.2.

4.3.2. *A Large, Sparse, Noisy Matrix Arising in Image Processing.* One way to tackle standard tasks in image processing is to use the information about the local geometry of the image through the *graph Laplacian*. The dominant eigenvectors of this operator provide coordinates that help to smooth out noisy image patches [9].

First, define $\widetilde{\boldsymbol{W}}$ as

$$\widetilde{w}_{ij} = \exp\{-\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|^2/\sigma^2\} \tag{4.2}$$

where $\boldsymbol{x}^{(i)}$ is a vector gathering the intensities of a neighborhood of a certain size of the pixel i. Here, σ controls the level of sensitivity. We then define the sparse matrix $\boldsymbol{W}$ constructed by zeroing out all the entries in $\widetilde{\boldsymbol{W}}$ except the 7 largest ones in each row. Finally, the laplacian operator is defined as

$$\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2} \tag{4.3}$$

where $\boldsymbol{D}$ is the diagonal matrix with entries $d_{ii} = \sum_j w_{ij}$.

In order to find the low-frequency eigenvectors of $\boldsymbol{L}$, we must find the eigenvectors with dominant eigenvalues of

$$\boldsymbol{A} = \boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2} \tag{4.4}$$

In our experiment, we have taken a patch of size 57x57 from the classical `Lenna.png` photo. The matrix $\widetilde{\boldsymbol{W}}$ is constructed with patches of size 5x5 centered at the corresponding pixel with zero-padding at the boundary. The resulting matrix $\boldsymbol{A}$ has size 3249x3249 and has low decaying eigenvalues. Hence, as can be seen in Figure 4.3.2, the power scheme from Algorithm 2.1.3 is very advantageous.

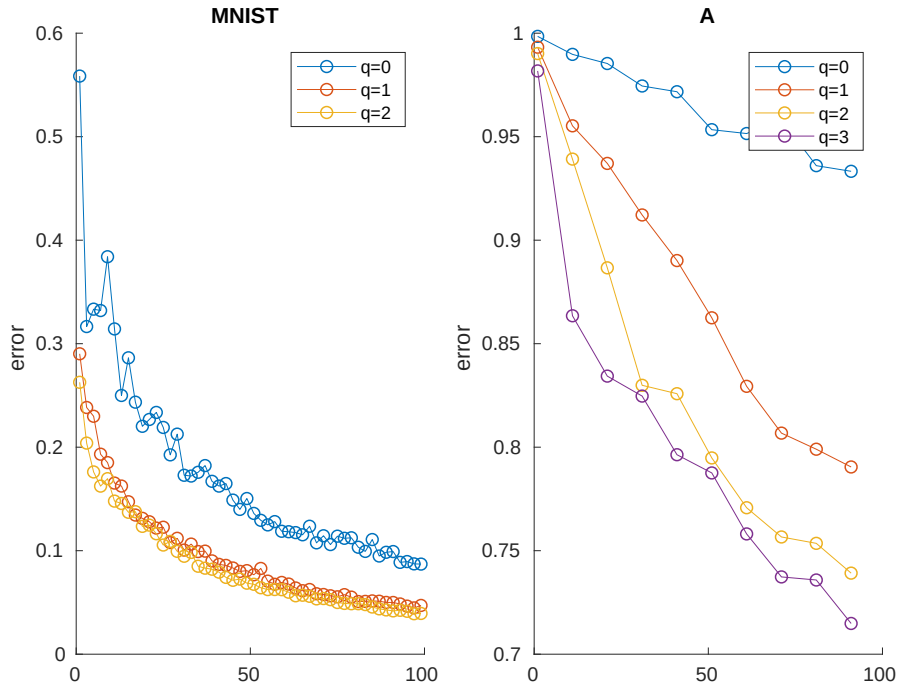

FIGURE 3. Approximation error 3.1 for the power scheme 2.1.3 for MNIST (Left) and $\boldsymbol{A}$ 4.4 (Right) for k ranging from 1 to 100.

5. Conclusion

Randomization schemes for approximate matrix factorization are a powerful and versatile tool that **every data scientist should know about**. These methods are more robust and scalable than classical algorithms, and despite not achieving the same level of accuracy, the error gap can be perfectly controlled with the *oversampling parameter* p and the number of power iterations q.

Moreover, these methods rely on theoretical bounds resulting from theorems on concentration of measure. We have seen in Experiment 4.2.1 that the bounds on the expectation can be very loose for certain matrices. One interesting direction of research can be the sharpening of bounds like 3.11 under additional hypothesis on the matrix $\boldsymbol{A}$.

There are several variants in this set of methods that are designed for matrices with different properties, see Section 2.3. One drawback is that sometimes they can be not that easy to verify before applying the algorithm, for e.g, the decaying rate of the spectrum.

From a more personal point of view, I have personally appreciated both the theoretical and experimental section of this project and how they complement each other. I think it is extremely important to have nice experimental results always grounded by theory.

References

[1] Åke Björck. "Numerics of gram-schmidt orthogonalization". In: *Linear Algebra and Its Applications* 197 (1994), pp. 297–316.

[2] Hongwei Cheng et al. "On the compression of low rank matrices". In: *SIAM Journal on Scientific Computing* 26.4 (2005), pp. 1389–1404.

[3] Jack Dongarra and Francis Sullivan. "Guest editors introduction: The top 10 algorithms". In: *Computing in Science & Engineering* 2.1 (2000), pp. 22–23.

[4] Ming Gu and Stanley C Eisenstat. "Efficient algorithms for computing a strong rank-revealing QR factorization". In: *SIAM Journal on Scientific Computing* 17.4 (1996), pp. 848–869.

[5] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. "Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions". In: *SIAM review* 53.2 (2011), pp. 217–288.

[6] Yann LeCun. "The MNIST database of handwritten digits". In: *http://yann. lecun. com/exdb/mnist/* (1998).

[7] Leon Mirsky. "Symmetric gauge functions and unitarily invariant norms". In: *The quarterly journal of mathematics* 11.1 (1960), pp. 50–59.

[8] GW Stewart. "The decompositional approach to matrix computation". In: *Computing in Science & Engineering* 2.1 (2000), pp. 50–59.

[9] Arthur D Szlam, Mauro Maggioni, and Ronald R Coifman. "Regularization on graphs with function-adapted diffusion processes". In: *Journal of Machine Learning Research* 9.Aug (2008), pp. 1711–1739.

[10] Roman Vershynin. *High Dimensional Probability*. 2016.

[11] Franco Woolfe et al. "A fast randomized algorithm for the approximation of matrices". In: *Applied and Computational Harmonic Analysis* 25.3 (2008), pp. 335–366.

Master "Mathématiques, Vision et Apprentissage", École Normale Supérieure de Cachan
E-mail address: alexnowakvila@gmail.com