PROBABILISTIC ALGORITHMS FOR FINDING 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 algorithms 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.

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 [7] 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.

The empirical covariance matrices derived from datasets 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 $\boldsymbol{C} \in \mathbb{R}^{k\times n}$.

$$\underset{m\times n}{\boldsymbol{A}} \approx \underset{m\times k}{\boldsymbol{B}} \underset{k\times n.}{\boldsymbol{C}}, \tag{0.1}$$

The matrix $\boldsymbol{B} \times \boldsymbol{C}$ in 0.1 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

$$r(\boldsymbol{A}) \coloneqq \frac{\|\boldsymbol{A}\|_F^2}{\|\boldsymbol{A}\|^2} = \sum_{j=1}^{\min(m,n)} \left(\frac{\sigma_j}{\sigma_1}\right)^2 \tag{0.2}$$

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 [8] 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

$$\boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A} \tag{0.3}$$

  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 the algorithms will belong to this first stage.
- **Stage 2:** Given a matrix $\boldsymbol{Q}$ that satisfies 0.3, 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.

0.1. **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 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

$$\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| \leq \epsilon \tag{0.4}$$

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 problem*, which seeks to find the best rank-k approximation of the matrix.

$$\min_{\operatorname{rank}(\boldsymbol{X}) \leq j} \|\boldsymbol{A} - \boldsymbol{X}\|. \tag{0.5}$$

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

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* = \sum_{j=1}^{\operatorname{rank}(\boldsymbol{A})} \sigma_j \boldsymbol{u}_j \boldsymbol{v}_j^* \tag{0.6}$$

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.

[1] The comparison between both notions of rank can be better understood through the following caracterization. $\operatorname{rank}(\boldsymbol{A}) = \dim(\mathbf{A}B_2^n)$ 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 [8], 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 SVD provides an optimal answer to the *fixed precision problem* [6] through the following important observation [2].

$$\min_{\text{rank}(\boldsymbol{X})\leq k} \|\boldsymbol{A}-\boldsymbol{X}\| = \sigma_{j+1}. \tag{0.7}$$

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 of the matrix $\boldsymbol{A}$. More precisely, we have that $\boldsymbol{B} = \boldsymbol{U}\boldsymbol{\Sigma}_{[j]}^{1/2}$ and $\boldsymbol{C} = \boldsymbol{\Sigma}_{[j]}^{1/2}\boldsymbol{V}^*$ are the best solutions 0.1 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

$$\|\boldsymbol{A}-\boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\| \approx \min_{\text{rank}(\boldsymbol{X})\leq k} \|\boldsymbol{A}-\boldsymbol{X}\| \tag{0.8}$$

where p is called the oversampling parameter.

EXPLICACIO

0.2. **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 $\boldsymbol{w}^{(i)}$ of the input space. Let us repeat this process k times:

$$\boldsymbol{y}^{(i)} = \boldsymbol{A}\boldsymbol{w}^{(i)}, \qquad i = 1,\dots,k \tag{0.9}$$

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? 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 0.10, 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:

$$\boldsymbol{y}^{(i)} = \boldsymbol{A}\boldsymbol{w}^{(i)} = \boldsymbol{B}\boldsymbol{w}^{(i)} + \boldsymbol{E}\boldsymbol{w}^{(i)}, \qquad i = 1,\dots,k+p \tag{0.10}$$

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 holding in high probability. The theory also shows that p can be quite small. In practice, $p = 5$ is more than enough.

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

[2] This is a direct consequence of the Courant-Fisher's *min-max theorem*. This theorem offers a variational characterization of the singular values of a given matrix $\boldsymbol{A}$.

$$\sigma_i(A) = \min_{E,\dim E=i} \max_{\boldsymbol{x}\in S(E)} \|\boldsymbol{A}\boldsymbol{x}\|$$

FER UNA MINI-DEMO

0.3. **Construction of standard matrix factorizations from Q.** This corresponds to **Stage 2** of the algorithm. Once we have $\boldsymbol{Q}$ such that $\boldsymbol{A} \approx \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}$, taking $\boldsymbol{B} = \boldsymbol{Q}$ and $\boldsymbol{C} = \boldsymbol{Q}^*\boldsymbol{A}$ we produce a low rank matrix decomposition $\boldsymbol{A} \approx \boldsymbol{B}\boldsymbol{C}$.

A lot of questions need to be addressed in order to turn these methods into a technology or *off-the-shelf* algorithms, namely, theoretical guarantees on the accuracy of the approximate factorization and experimentally study the gap between theory and practice, i.e, how these methods perform in practice.

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.

1. Stage 1

In the introductory section, we provided some intuition on the randomized procedure and we developed a general Proto-Algorithm 0.2 to find the matrix $\boldsymbol{Q}$. However, Proto-Algorithm 0.2 is very general and can be tunned depending on the problem requirements. The number T_{basic} of flops required by Proto-Algorithm 0.2 satisfies

$$T_{\text{basic}} \sim \ell n\, T_{\text{rand}} + \ell\, T_{\text{mult}} + \ell^2 m \tag{1.1}$$

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.

We will now describe some specific realizations of Proto-Algorithm 0.2 that will be intended for problems with different requirements.

1.0.1. *Randomized Range Finder.* This is the most naive and simplest implementation of Proto-Algorithm 0.2. Given an oversampling parameter p, the *Randomized Range Finder* performs Proto-Algorithm 0.2 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.

Randomized Range Finder

1 Draw an $n \times \ell$ Gaussian random matrix $\boldsymbol{\Omega}$.
2 Form the $m \times \ell$ matrix $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{\Omega}$.
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 $\boldsymbol{Y} = \boldsymbol{Q}\boldsymbol{R}$.

The complexity analysis of the Algorithm 1.0.1 gives

$$T_{\text{Randomized Range Finder}} \sim \mathcal{O}(mn\ell) \tag{1.2}$$

this is because generating a gaussian random number is $\mathcal{O}(1)$ and computing a matrix vector multiplication is $\mathcal{O}(mn)$.

1.0.2. *Adaptive Randomized Range Finder.* One important pitfall of the *Randomized Range Finder* 1.0.1 is that it requires to know in advance the target rank k. However, if we intend to solve the *fixed-precision problem*, we need a scheme to estimate the error $\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}\|$ 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 1.1. *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 $\{\boldsymbol{\omega}^{(i)} : i = 1, 2, \dots, r\}$ of standard Gaussian vectors. Then*

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

except with probability α^{-r}.

Lemma 1.1 says that we can bound the error with high probability using inexpensive computations in an online manner. The Lemma 1.1 applied to our problem reads

$$\|(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{A}\| \leq 10\sqrt{\frac{2}{\pi}} \max_{i=1,\dots,r} \left\|(\mathbf{I} - \boldsymbol{Q}\boldsymbol{Q}^*)\boldsymbol{A}\boldsymbol{\omega}^{(i)}\right\| \tag{1.3}$$

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

The high probability bound 1.3 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

$$\left\|\left(\mathbf{I} - \boldsymbol{Q}^{(\ell)}(\boldsymbol{Q}^{(\ell)})^*\right)\boldsymbol{A}\right\| \leq \varepsilon. \tag{1.4}$$

We call *Adaptive Randomized Range Finder* 1.0.2 to the algorithm derived from Lemma 1.1 that solves this problem.

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
$\boldsymbol{Q} = \boldsymbol{Q}^{(j)}$.

One important question regarding Algorithm 1.0.2 is how good the bound given by Lemma 1.1 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 3.3.4.

1.0.3. *Randomized Power Iteration.* The *Randomized Range Finder* 1.0.1 algorithm assumes that the singular values of the matrix decay fast. This can be seen from equation 0.10, where the small singular values interfere with the calculation of the range. This intuition is made precise in Theorem REF, where the error of the approximation depends on the $k+1$-th singular value.

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 $\boldsymbol{B} = (\boldsymbol{A}\boldsymbol{A}^*)^q\boldsymbol{A}$ 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.

$$\sigma_j(\boldsymbol{B}) = \sigma_j(\boldsymbol{A})^{2q+1}, \qquad j = 1, 2, 3, \dots \tag{1.5}$$

The *Randomized Power Iteration* 1.0.3 algorithm is the same as the *Randomized Range Finder* 1.0.1 but replacing the formula $\boldsymbol{Y} = \boldsymbol{A\Omega}$ by $\boldsymbol{Y} = \boldsymbol{B\Omega}$.

Randomized Power Iteration
1 Draw an $n \times \ell$ Gaussian random matrix $\boldsymbol{\Omega}$.
2 Form the $m \times \ell$ matrix $\boldsymbol{Y} = (\boldsymbol{AA}^*)^q \boldsymbol{A\Omega}$ via alternating application of $\boldsymbol{A}$ and $\boldsymbol{A}^*$.
3 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 $\boldsymbol{Y} = \boldsymbol{QR}$.

The computational complexity of the algorithm is essentially the same because it only requires $2q+1$ as many matrix-multiplications as Algorithm 1.0.1 but the number q is in practice 2,3 or 4. This can be seen from THEOREM, which shows that the power iteration drives the approximation gap to 1 exponentially fast.

1.0.4. *Fast Randomized Range Finder.* A simple inspection to equation 1.1 reveals the computational bottleneck of the sampling procedure. This is the matrix multiplication $\boldsymbol{Y} = \boldsymbol{A\Omega}$ that takes $\mathcal{O}(mn\ell)$ operations for dense matrices, which is the same as the ℓ-SVD (computed after a prior rank-revealing QR factorization [4]).

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

$$\boldsymbol{\Omega} = \sqrt{\frac{n}{\ell}}\, \boldsymbol{DFR}, \tag{1.6}$$

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 $p, q = 1, 2, \dots, n$, and
- $\boldsymbol{R}$ 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 [9], 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

$$T_{\text{struct}} \sim mn\log(\ell) + \ell^2 n \tag{1.7}$$

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 (1.6).
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. Stage 2

The output of the Stage 1 produces an orthonormal matrix $\boldsymbol{Q}$ whose range captures the action of the matrix $\boldsymbol{A}$. The goal of Stage 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 $\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\| \leq \varepsilon$. 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 $\boldsymbol{C} = \boldsymbol{Q}^*\boldsymbol{A}$.

2.0.1. *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 $\|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\| \leq \varepsilon$ maintaining the tolerance ϵ from Stage 1.

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

  (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: $\boldsymbol{D} = \boldsymbol{U}_2\boldsymbol{\Sigma}\boldsymbol{V}^*$.
  (3) Form the product $\boldsymbol{U} = \boldsymbol{Q}_1\boldsymbol{U}_2$.

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

  (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 $\boldsymbol{Q} = \boldsymbol{Q}_1\boldsymbol{Q}_2$.

2.0.2. *Direct SVD.* The procedure described in 2.0.1 to compute the approximate SVD decomposition without sacrificing error, defines what we call the *Direct SVD* 2.0.2.

Direct SVD

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

Although using *Direct SVD* 2.0.2 algorithm for Stage 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 when using Algorithm 1.0.4.

2.0.3. *Faster Procedures.* In order to match the complexity $\mathcal{O}(mn\log(\ell))$ from Stage 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].The proposed algorithm takes $\boldsymbol{Q}$ as input and constructs a rank-k matrix factorization

$$\boldsymbol{A} \approx \boldsymbol{X}\boldsymbol{B} \tag{2.1}$$

where $\boldsymbol{B}$ is a $k \times n$ matrix consisting of k rows extracted from $\boldsymbol{A}$.

The key here is that 2.1 can be produced without any 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 $\boldsymbol{Q}^*\boldsymbol{Q}\boldsymbol{A}$.

3. Full algorithms

Theory

3.1. **Analysis of Stage 1.** This section focuses on assessing the quality of the basis given by Proto-Algorithm 0.2. More precisely, we want to prove rigorous bounds on the approximation error

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

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

We will split the argument into two parts [3]:

(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 $\boldsymbol{A}$. [4]
(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

$$\boldsymbol{A} = \boldsymbol{U} \begin{array}{c} \begin{matrix} k & n-k \end{matrix} \\ \begin{bmatrix} \boldsymbol{\Sigma}_1 & \\ & \boldsymbol{\Sigma}_2 \end{bmatrix} \end{array} \begin{array}{c} \begin{matrix} n \end{matrix} \\ \begin{bmatrix} \boldsymbol{V}_1^* \\ \boldsymbol{V}_2^* \end{bmatrix} \end{array} \begin{array}{c} k \\ n-k \end{array} \tag{3.1}$$

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

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

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 0.2 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 $\boldsymbol{P}_{\boldsymbol{Y}} = \boldsymbol{P}_{\mathrm{Im}(\boldsymbol{Y})} = \boldsymbol{Q}\boldsymbol{Q}^*$ [5]. 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*

$$\|\!|(\mathbf{I} - \boldsymbol{P}_{\boldsymbol{Y}})\boldsymbol{A}|\!\|^2 \leq \|\!|\boldsymbol{\Sigma}_2|\!\|^2 + \left\|\!\left|\boldsymbol{\Sigma}_2\boldsymbol{\Omega}_2\boldsymbol{\Omega}_1^\dagger\right|\!\right\|^2, \tag{3.2}$$

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. EXPLAIN

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 REF, 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 this simple 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.

[3] The authors argue that this bipartite proof is common in the literature of randomized linear algebra
[4] Note that we do not deal with randomness yet.
[5] 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*

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

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 are 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*

$$\left(\mathbb{E}\left\|\boldsymbol{SGT}\right\|_{\mathrm{F}}^2\right)^{1/2} = \left\|\boldsymbol{S}\right\|_{\mathrm{F}} \left\|\boldsymbol{T}\right\|_{\mathrm{F}} \quad \text{and} \quad \mathbb{E}\left\|\boldsymbol{SGT}\right\| \leq \left\|\boldsymbol{S}\right\| \left\|\boldsymbol{T}\right\|_{\mathrm{F}} + \left\|\boldsymbol{S}\right\|_{\mathrm{F}} \left\|\boldsymbol{T}\right\|. \tag{3.3}$$

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*

$$\left(\mathbb{E}\left\|\boldsymbol{G}^{\dagger}\right\|_{\mathrm{F}}^2\right)^{1/2} = \sqrt{\frac{k}{p-1}} \quad \text{and} \quad \mathbb{E}\left\|\boldsymbol{G}^{\dagger}\right\| \leq \frac{\mathrm{e}\sqrt{k+p}}{p}. \tag{3.4}$$

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

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

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 Frobenius error)**.** *The expected approximation error can be bounded as follows*

(1)

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

(2)

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

One interesting quantity is worth examining is $\left\|(\mathbf{I} - \boldsymbol{P_Y})\boldsymbol{A}\right\| / \sigma_{k+1}$ to check the factor that tells how far the approximation is from the optimal rank-k approximation 0.5. We observe that the suboptimality term increases essentially as $\sim \sqrt{k/p}$ and has a term corresponding to the numerical rank 0.2 of the singular values corresponding to the perturbation. [6]

Proof. Hölder's inequality and Theorem 3.1 give

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

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

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

[6] 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 0.2 and its interpretation, I found interesting to highlight its appearance in the theorem.

Putting everything together

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

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

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

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

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

Finally applying Proposition 3.9, we get to the final result

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

□

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 $u, t \geq 1$,*

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

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*

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

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.

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 1.1.

The second set of experiments is ...

3.2. **Details of the implementation.** The experiments have been performed in `Matlab`. The reproducible source code can be found at the following Github repository together with the LaTeXof the report.

3.3. **Gaussian Matrices.** In this set of experiments, we test and analyze the performance of the Algorithms 1.0.1, 1.0.2, 1.0.3 and 1.0.4.

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

$$\boldsymbol{A} = \frac{1}{\sqrt{m}+\sqrt{n}}\boldsymbol{G}, \qquad G_{ij} \sim N(0,1) \tag{3.5}$$

The normalization in 3.5 is to make sure that the norm of $\boldsymbol{A}$ is around 1 with high probability [7].

3.3.1. *Experiments on Randomized Range Finder 1.0.1.*

3.3.2. *Experiments on Randomized Power Iteration 1.0.3.*

3.3.3. *Experiments on Adaptive Randomized Range Finder 1.0.2.* Show the ratio between the tolerance and the actual error.

3.3.4. *Experiments on Fast Randomized Range Finder 1.0.4.*

3.4. **Real Dataset.**

Conclusion

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

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

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

[9] 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.

Department of Mathematics, École Normale Supérieure de Cachan
E-mail address: alexnowakvila@gmail.com

[7] 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 [8].

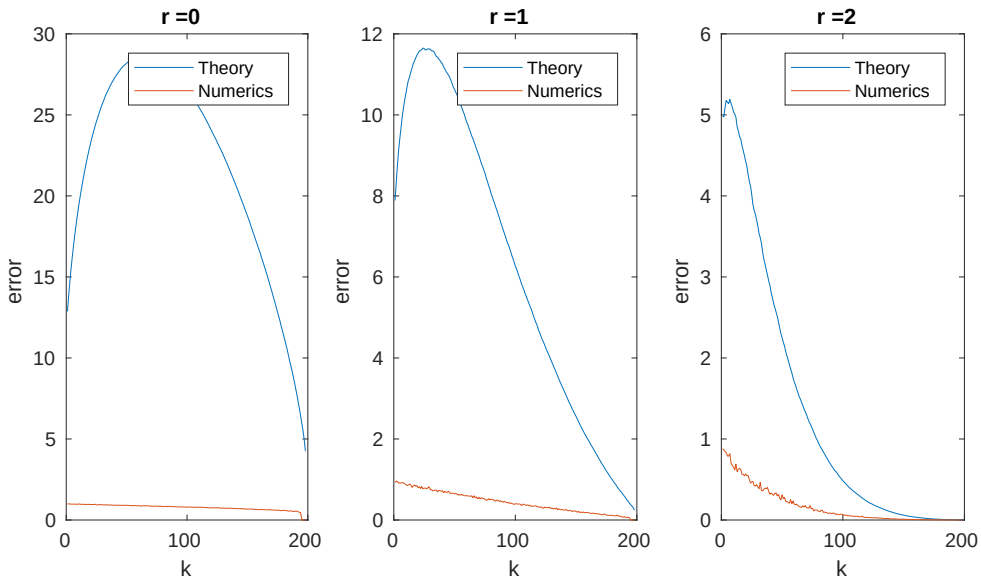
r =0
r =1
r =2
Theory
Numerics
error
k
0
100
200
5
10
15
20
25
30
2
4
6
8
12
1
3

FIGURE 1. bla

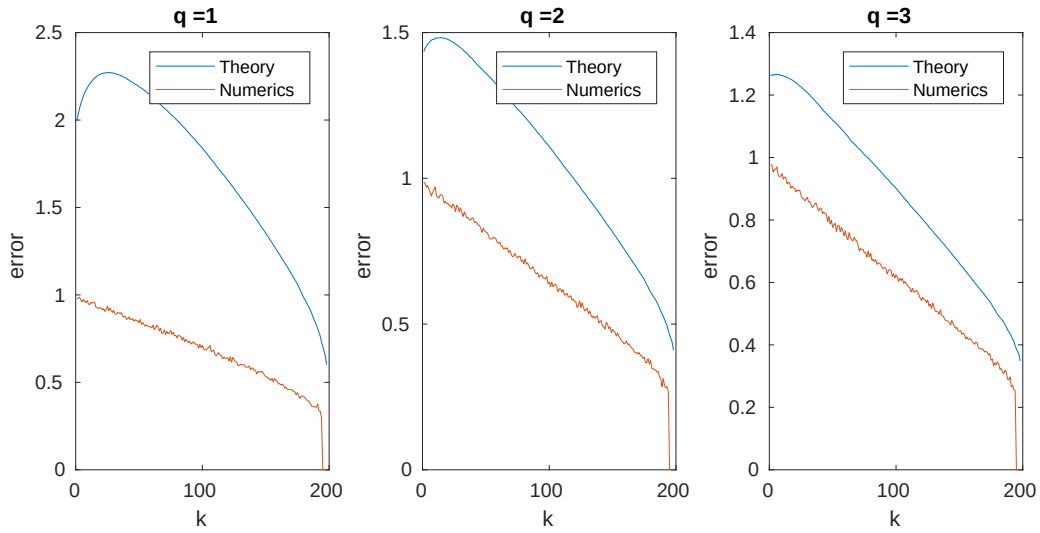
q =1
q =2
q =3
Theory
Numerics
error
k
0
100
200
0.5
1
1.5
2
2.5
0.2
0.4
0.6
0.8
1.2
1.4

FIGURE 2. bla