PERSPECTIVE

National Science Review
10: nwad083, 2023
https://doi.org/10.1093/nsr/nwad083
Advance access publication 25 March 2023

COMPUTER SCIENCE

Special Topic: AI Algorithms and Cases: To Energize Digital Economy

Fast algorithms for singular value decomposition and the inverse of nearly low-rank matrices

Chen Xu[1], Weiwei Xu[2,3,*] and Kaili Jing[4]

PROBLEM

Matrix computation is a fundamental task in science and engineering; it is recognized as one of the seven computational 'giants' of massive data analysis [1], analogous to the seven 'dwarfs' in high-performance computing. In recent years, as the scale of datasets rapidly grows, analysts and engineers frequently need to deal with big matrices with a nearly low-rank structure. The big size of a matrix brings great numerical hurdles to matrix computation. In this work, we focus on two typical tasks in matrix computation: singular value decomposition (SVD) and matrix inverse on nearly low-rank matrices that are potentially of big sizes.

For the convenience of discussion, we denote by $\mathbf{C}^{m\times n}$ the family of complex matrices with m rows and n columns. For any $A\in\mathbf{C}^{m\times n}$, let r be its rank and σ_i, $i=1,2,\ldots,k$, be the set of its singular values with $k=\min\{m,n\}$ and $\sigma_i\geq\sigma_{i+1}$. Matrix A is said to be of low rank if $r<k$ and nearly low rank if

$$\frac{\sum_{i=1}^{r}\sigma_i^2}{\sum_{i=1}^{k}\sigma_i^2}\geq 1-\epsilon$$

for some small threshold $\epsilon>0$.

When A is of nearly low rank, only r-out-of-k singular values are significant; it can be thus approximated by an r-rank matrix $\tilde{A}$ with all insignificant singular values disregarded. It is known that finding $\tilde{A}$ amounts to solving an r-truncated SVD of A with a user-specified value of r. When $m=n$, the techniques used in finding $\tilde{A}$ may also be applied to solve the inverse of $I+A$ with I being an $m\times m$ identity matrix. This was Task 5 in the Arena Contest of the 2022 International Algorithm Case Competition (http://112.74.40.78:8084/).

Solving the SVD or inverse of a nearly low-rank matrix has a wide scope of applications in many scientific areas. For example, in high-dimensional statistics, the number of covariates is larger than the sample size; this naturally leads to a low-rank sample covariance matrix being handled in tasks like regression or principal component analysis. In massive data analysis, the Gram matrix of a kernel method often comes with a nearly low-rank structure due to a large number of correlated entries. In the field of communication, the transmission speed of a multi-antenna wireless system largely relies on the efficiency of solving a series of SVD and inverse problems on the channel information matrices. Since the communication channels are highly similar among the users, the corresponding channel information matrices are typically of nearly low rank.

Solving the SVD or inverse can be numerically costly for big matrices; the task comes with a typical complexity of $O(mn^2)$ or $O(m^3)$, respectively. For an r-rank matrix, techniques such as QR decomposition can be applied to transform the original $m\times n$ problem into $r\times r$ problems [2]. These techniques are helpful in reducing the complexity of the task only when the rank r is known beforehand; this, however, is unrealistic in many applications, where r is typically a 'to-be-determined' value. Moreover, the existing methods to solve the SVD or matrix inverse often suffer from algorithmic instability due to multiple-root issues. All these call for new techniques to handle this functional task, to which we now turn.

ALGORITHMS

We tackle the considered problem by a novel procedure. Our ideas are to apply a new on-line low-rank decomposition tool, called QB decomposition, to transform the original $m\times n$ high-order problems into $r\times r$ lower-order problems, and develop an random renormalization theory to solve the rank r unknown problem. The three strategies adopted are as follows.

Strategy 1: QB decomposition

The low-rank decomposition is a basic tool of complexity reduction of matrix computation, which aims to decompose a matrix $A\in\mathbf{C}^{m\times n}$ into the form $A=QB$, where $Q\in\mathbf{C}^{m\times r}$ is a column orthogonal matrix and $B\in\mathbf{C}^{r\times n}$ a structured matrix (normally, an upper triangular matrix). The classical tools used are QR decomposition and the Gram-Schmit process (GSP). The QR decomposition is efficient for fully rank matrices and has $O(mn^2)$ complexity, while GSP can generate a perfect Q but a partial B because of missing the expression coefficients of

columns relevant to previous columns. Based on the observation that $B = Q^H A$ holds for every low-rank decomposition, we skillfully arrange the computation order and propose a revised GSP procedure; see Algorithm 1. We call such low-rank decomposition the QB decomposition of matrix A.

Algorithm 1: (QB decomposition)

Input: Matrix $A = [a_1, \ldots, a_n] \in \mathbf{C}^{m\times n}$. Let $A_k = [a_1, \ldots, a_k]$ and $A_k = Q_k B_k$ be the QB decomposition of A_k.
Output: QB decomposition of A: $A = QB$.

Compute $q_1 = a_1/\|a_1\|$ and $B_1 = (q_1^H a_1, \ldots, q_1^H a_n)$.
$j := 0, k := 1$;
while $k + j + 1 \le n$ **do**
  $\tilde{q}_{k+1} = (I - \sum_{i=1}^{k} q_i q_i^H) a_{k+1+j}$,
  if $\tilde{q}_{k+1} \neq 0$ **then**
    compute $q_{k+1} = \tilde{q}_{k+1}/\|\tilde{q}_{k+1}\|$;
    compute $B_{k+1} = (\underbrace{0 \ldots 0}_{k}, q_{k+1}^H a_{k+1}, \ldots, q_{k+1}^H a_n)$;
    $k := k + 1$;
  else
    $j := j + 1$.
  end
end
Set $Q = [q_1, \ldots, q_k]$ and $B = \begin{pmatrix} B_1 \\ \vdots \\ B_k \end{pmatrix}$.

The QB decomposition is an $O(mnr)$ complexity procedure, which can be conveniently used to transform the high-order SVD problem of $A \in \mathbf{C}^{m\times n}$ into a lower-order SVD problem of $C \in \mathbf{C}^{r\times r}$, where r is the rank of A. The procedure, say, can be as follows: decompose $A = QB$, $AA^H = QBB^HQ^H$, and then the SVD of A can be obtained from the SVD of $C = BB^H$, where C is an $r \times r$ matrix.

The QB decomposition has a very natural 'online correction' property that benefits reduction of storage and data movements in implementation. In particular, when one more column, say, a_{k+1}, is added into $A_k = Q_k B_k$, the corresponding QB decomposition $A_{k+1} = [A_k \ a_{k+1}]$, $= Q_{k+1} B_{k+1}$ can be computed through Q_k, B_k and a_{k+1} only. Thus, QB decomposition can essentially provide a low-rank representation of a sequence of input vectors. In this point of view, we can apply QB decomposition to sequence $\{q_1, Aq_1, Aq_2, \ldots, Aq_r\}$, justifying the following theorem.

Theorem 1. *Assume that $A \in \mathbf{C}^{n\times n}$ is Hermitian, q_1 is any fixed nonzero unit vector and that the QB decomposition of sequence $\{q_1, Aq_1, Aq_2, \ldots, Aq_r\}$ is given by*

$$[q_1, Aq_1, Aq_2, \ldots, Aq_r] = Q[e_1, B],$$

where $e_1 = (1, 0, \ldots, 0)^T$, $Q = [q_1, q_2, \ldots, q_r]$. Then $A = QBQ^H$ is a Lanczos decomposition, i.e. $\{q_1, q_2, \ldots, q_r\}$ is a Lanczos sequence and B is a triple diagonal matrix, respectively defined by the Lanczos process [3].

Theorem 1 shows that the Lanczos process can be redefined by the QB decomposition.

Strategy 2: Lanczos' process revised

The Lanczos method computes the eigendecomposition of Hermitian matrices, which works by tridiagonalizing the matrices first (the tridiagonalization stage, namely, the Lanczos process), and then computing the deduced, much simplified tridiagonal eigendecomposition problems. It is a state of the art procedure while the instability and multiple-root issues have not be thoroughly resolved.

Theorem 1 bridges the QB decomposition and Lanczos process. Such a connection enables us to rationally revise Lanczos' method to help mitigate those issues. For example, we can substantially improve the algorithmic stability by employing the QB decomposition in place of the Lanczos process, or by adding a QB update after every two steps of the construction of orthogonal rows q_i in the Lanczos process. Furthermore, we proved the following theorem.

Theorem 2. *Assume that $M \in \mathbf{C}^{n\times n}$ is a Hermitian matrix and that $\{q_i\}$ is the sequence defined by the Lanczos process for M. If q_1 satisfies*

$$Mq_1 \neq 0 \text{ and } q_1 \text{ is not an eigenvector of } M \qquad (1)$$

then $\{q_i\}$ is well defined, and $q_i \neq 0$ if and only if $i \le n_0$, where n_0 is the degree of the minimal polynomial of M.

Theorem 2 implies that Lanczos' method (precisely, the Lanczos process) will never be terminated before n_0 steps of update, and the yielded $\{q_i\}$ must locate at different eigensubspaces. Based on this, incorporated with an m_0 times restart scheme, where m_0 is the number of maximal-multiplicity eigenvalues of A, and choosing

$$M_0 = A,$$
$$M_j = \left(I - \sum_{l}^{j} Q_l Q_l^H\right) A, \quad j = 1, \ldots, m_0 - 1$$

(where Q_l is defined as in Algorithm 2), with the initial values satisfying (1) each time, we can then effectively resolve the multiple-root issue. We note that condition (1) is very easily satisfied in use due to the fact that all the vectors satisfying condition (1) form a zero measure set.

Algorithm 2: (Lanczos' process revised)

Input: Hermitian matrix $A \in \mathbf{C}^{n\times n}$. The largest multiplicity m_0 of eigenvalues of A.
Output: The tridiagonal decomposition of A: $A = QBQ^H$.

Take an n-dimensional column vector ζ_1 satisfying (1);
compute $q_1^{(1)} = \zeta_1/\|\zeta_1\|_2$;
for $i = 1, 2, \ldots, m_0$ **do**
  if $i \ge 2$ **then**
    set $\tilde{q}_1^{(i)} = (I - \sum_{l=1}^{i-1} Q_l Q_l^H) A\zeta_i$, where ζ_i is a randomly chosen n-dimensional column vector;
    if $\tilde{q}_1^{(i)} = 0$ **then**
      go to step 5;
    end
    compute $q_1^{(i)} = \tilde{q}_1^{(i)}/\|\tilde{q}_1^{(i)}\|_2$;
  end
  for $k = 1, 2, \ldots$ **do**
    compute $\alpha_k^{(i)} = q_k^{(i)H} A q_k^{(i)}$ and $\tilde{q}_{k+1}^{(i)} = (I - \sum_{l=1}^{i-1} Q_l Q_l^H - \sum_{j=1}^{k} q_j^{(i)} q_j^{(i)H}) A q_k^{(i)}$;
    if $\tilde{q}_{k+1}^{(i)} \neq 0$ **then**
      compute $q_{k+1}^{(i)} = \tilde{q}_{k+1}^{(i)}/\|\tilde{q}_{k+1}^{(i)}\|_2$ and $\beta_k^{(i)} = q_k^{(i)H} A q_{k+1}^{(i)}$;
    else
      break.
    end
  end
  Set $Q_i = [q_1^{(i)}, q_2^{(i)}, \ldots, q_k^{(i)}]$;
end
return $Q = [Q_1, Q_2, \ldots, Q_{m_0}]$ and

$$B = \begin{bmatrix} \alpha_1^{(1)} & \beta_1^{(1)} & & & & \\ \beta_1^{(1)} & \alpha_2^{(1)} & \beta_2^{(1)} & & & \\ & \beta_2^{(1)} & \alpha_3^{(1)} & \ddots & & \\ & & \ddots & \ddots & \beta_{k-1}^{(m_0)} & \\ & & & \beta_{k-1}^{(m_0)} & \alpha_k^{(m_0)} & \end{bmatrix}.$$

With this in wind, we propose a revised Lanczos process; see Algorithm 2. Its complexity is $O(n^2 r)$.

Strategy 3: random renormalization theory

Random renormalization is an attempt to recombine the rows of A, through an action of random transformation Ω (matrix), such that (i) $\text{rank}(A\Omega) = \min\{\text{rank}(A), \text{rank}(\Omega)\}$ and (ii) $A\Omega$ and A have the same low-rank basis, i.e. their QB decompositions $A\Omega = Q_1B_1$ and $A = QB$ have the property that $Q = Q_1$. We proved that there exist random matrices, e.g. Gaussian random and Bernoulli random matrices, that satisfy the random renormalization conditions in high probability.

The random renormalization property (ii) implies that instead of A we can employ $A\Omega$'s QB decomposition to determine the low-rank basis Q, and property (i) then implies that using the QB decomposition of $A\Omega$, not A, may bring an exclusive advantage: $A\Omega$ keeps full column rank and the resulting upper triangular expression B nonsingular, i.e. its diagonal entries all are nonzero whenever $\text{rank}(\Omega) \le \text{rank}(A)$. This advantage can be utilized to resolve the rank r unknown problem in a more economic way. The method is as follows. We recursively define $\Omega_{k+1} = [\Omega_k \quad \xi_{k+1}]$ with a random renormalization vector $\xi_k, k = 1, 2, \ldots$, i.e. $\Omega_k = [\xi_1, \xi_2, \ldots, \xi_k]$ and implement the QB decomposition (using the online version applied to $A\Omega_{k+1} = [A\Omega_k\, A\xi_{k+1}]$), resulting in QB decompositions $A\Omega_k = Q_kB_k$ and $A\Omega_{k+1} = Q_{k+1}B_{k+1}$. Terminate the process when an index r appears such that B_r has no zero entries but B_{r+1} does. Obviously, the rank r of A can be identified in this way and, more exclusively, this is realized in the QB decomposition process of $A\Omega_r$, with no additional computation involved thanks to the online correction property of the QB decomposition. In addition, to avoid the extra computation cost of multiplication of $A\Omega$ in place of A, we suggest applying Bernoulli random renormalization in applications.

To identify the rank of A by counting the number of nonzero diagonal entries of B as above provides an opportunity to treat SVD problems of low-rank and nearly low-rank matrices in a unified way. In effect, assuming that $B(i, i)$ is the ith diagonal element of B and that ε_1 is a tolerance threshold, we can regard $B(i, i)$ as zero if and only if $|B(i, i)| \le \varepsilon_1$, where ε_1 is zero when A is of low rank and a small threshold value when A is of nearly low rank. We proved that ϵ_1 can be specified explicitly by the given tolerance ϵ.

Incorporated with the three strategies above, the finalized algorithms we suggested for fast computation of the SVD and inverse of nearly low-rank matrices are presented in Algorithms 3 and 4, whose complexities are $O(mnr)$ and $O(n^2r)$, respectively.

Algorithm 3: (SVD)

Input: Matrix $A \in \mathbf{C}^{m\times n}$ and tolerance ε_1.
Output: SVD of A: $A = Q\Sigma V^H$.

Take a random renormalization n-dimensional column vector $\Omega_1 = \xi_1$.
Compute $B_1 = \|A\xi_1\|$, $Q_1 = A\xi_1/\|A\xi_1\|$;
for $k = 1, 2, \ldots, n$, **do**
  (1) take any random renormalization vector ξ_{k+1} and set
  $$\Omega_{k+1} = [\Omega_k, \ \xi_{k+1}];$$
  (2) perform the QB decomposition of $A\Omega_{k+1}$:
  $$A\Omega_{k+1} = Q_{k+1}B_{k+1};$$
  (3) **if** $|B_{k+1}(k+1, k+1)| \le \varepsilon_1$ **then**
    output Q_k.
  else
    Return to (1).
  end
end
Let $B = Q_k^H A$. Compute $C = BB^H$.
Compute the eigendecomposition of $C = U\Sigma_1U^H$ using the revised Lanczos process and QR implicit iteration method.
Compute $Q = Q_kU, \Sigma = \Sigma_1^{1/2}, V^H = \Sigma_1^{-1/2}U^HB$.

Algorithm 4: (Inverse of $(I + A)$)

Input: Matrix $A \in \mathbf{C}^{n\times n}$ and tolerance ε_1.
Output: inverse of $(I + A)$: $(I + A)^{-1} = I - ULB$.

Take a random renormalization n-dimensional column vector $\Omega_1 = \xi_1$.
Compute $B_1 = \|A\xi_1\|$, $Q_1 = A\xi_1/\|A\xi_1\|$;
for $k = 1, 2, \ldots, n$, **do**
  (1) take any random renormalization vector ξ_{k+1} and set
  $$\Omega_{k+1} = [\Omega_k, \ \xi_{k+1}];$$
  (2) perform the QB decomposition of $A\Omega_{k+1}$:
  $$A\Omega_{k+1} = Q_{k+1}B_{k+1};$$
  (3) **if** $|B_{k+1}(k+1, k+1)| \le \varepsilon_1$ **then**
    output Q_k.
  else
    Return to (1).
  end
end
Let $B = Q_k^H A$. Compute $BQ_k + I_k = L_1U_1$ by LU decomposition.
Compute $U = Q_kU_1^{-1}, L = L_1^{-1}$.

The proofs of Theorems 1 and 2 are given in the online supplementary material.

EVALUATION

The proposed method is numerically tested with simulations conducted in Matlab®. In particular, we access Algorithms 3 and 4 using a randomly generated $m \times n$ matrix A, where $A = U\Sigma V^H$ with U and V being two unitary matrices and Σ being an $n \times n$ diagonal matrix. We generate U and V with the Matlab code

```
U = orth(randn(m,n)
    +randn(m,n)*1i),
V = orth(randn(n,n)
    +randn(n,n)*1i).
```

For Σ, we first generate a diagonal matrix using the code

```
diag(sort(rand(n,1),'descend'))
```

and then rescale this matrix such that its maximum diagonal element is 1; we keep the top $r = \lfloor n/8 \rfloor$ elements unchanged and further scale down the remaining $n - r$ elements by a factor of 0.01. This ensures that A is of nearly low rank r.

We carry out the simulations on a Windows computer with 3.2 GHz CPUs and 32 GB memory. The algorithm accuracy is evaluated by

$$\text{ER}_{\text{SVD}} = \frac{\|A - \tilde{A}\|_F}{\|A\|_F},$$
$$\text{ER}_{\text{inverse}} = \frac{\|C(I + A) - I_m\|_F}{\sqrt{m}}$$

for the SVD and inverse problems, respectively. Here $\tilde{A}$ denotes the low-rank approximation of A based on the truncated SVD and C denotes the approximated inverse of $(I + A)$. In simulations, we observe that Algorithms 3 and 4 consistently lead to $\text{ER}_{\text{SVD}} \approx 10^{-12}$ and $\text{ER}_{\text{inverse}} \approx 10^{-12}$ over all sizes of A under our consideration. Their accuracy matches that obtained by the standard Matlab functions SVD and `inverse`.

In Fig. 1, we report the run times (in seconds) of Algorithms 3 and 4 over varying sizes of A. It is seen that the new algorithms are about 5–10 times faster

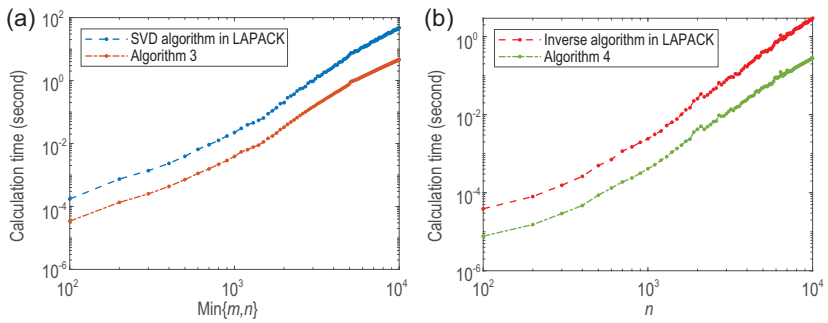

Figure 1. Time (in seconds) to carry out the computational tasks on a nearly low-rank matrix A with m rows and n columns: (a) solving the SVD of A; (b) solving the inverse of $(I + A)$ with $m = n$.

than the algorithms in LAPACK. This justifies the high efficiency of the proposed algorithms.

FUTURE RESEARCH

In this work, we proposed new algorithms for efficiently solving the SVD and inverse of a nearly low-rank matrix. The proposed algorithms effectively reduce the traditional computational complexity from $O(mn^2)$ down to $O(mnr)$; this leads to a significantly $O(n/r)$ times speedup. The proposed algorithms are accurate and stable; they do not rely on strong structure or sparsity requirements. These properties make the new algorithms attractive in practice.

The techniques in the current work can be extended to other matrix computation tasks. For example, the 'online correction' property of the QB decomposition can be used to solve the linear system of equations in a steaming manner. Algorithm 2 can be geared to efficiently solve the product of two low-rank matrices. Algorithm 4 can be modified to solve high-dimensional ridge regression with a huge number of features.

The current study focuses on the conventional setup, where matrix computation is carried out on a single computer [3–6]. To meet the requirement of the big data era [1], it would be promising to extend the proposed algorithms to the distributed or parallel setup.

SUPPLEMENTARY DATA

Supplementary data are available at *NSR* online.

FUNDING

This work was supported in part by the Pengcheng Lab key project (PCL2021A12) and the National R&D Program (2022YFA1003804).

Conflict of interest statement. None declared.

Chen Xu[1], Weiwei Xu[2,3,*] and Kaili Jing[4]
[1]Department of Mathematics and Statistics, University of Ottawa, Canada; [2]School of Mathematics and Statistics, Xi'an JiaoTong University, China; [3]School of Mathematics and Statistics, Nanjing University of Information Science and Technology, China and [4]Department of Mathematics and Statistics, University of Ottawa, Canada
***Corresponding author.**
E-mail: wwxu@nuist.edu.cn

REFERENCES

1. National Resaerch Council. *Frontiers in Massive Data Analysis*. Washington, DC: The National Academies Press, 2013.
2. Golub GH and Van Loan CF. *Matrix Computations*. Baltimore, MD: Johns Hopkins University Press, 2012.
3. Parlett BN and Scott DS. *Math Comput* 1979; **33**: 217–38.
4. Yamazaki I, Bai Z and Simon H *et al*. *ACM Trans Math Softw* 2010; **37**: 1–18.
5. Chan TF. *ACM Trans Math Softw* 1982; **8**: 72–83.
6. Halko N, Martinsson PG and Tropp JA. *SIAM Rev* 2011; **53**: 217–88.