Low rank matrix approximation

L. Grigori

Inria Paris, Sorbonne Université

January 2022

Plan

Low rank matrix approximation

Rank revealing QR factorization

Randomized algorithms for low rank approximation

Plan

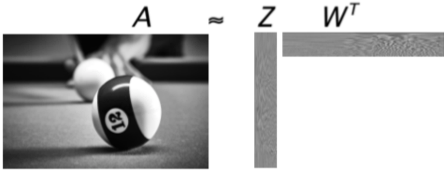
Low rank matrix approximation

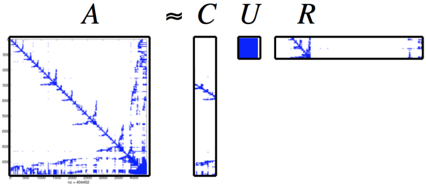
Rank revealing QR factorization

Randomized algorithms for low rank approximation

Low rank matrix approximation

- Problem: given $A \in \mathbb{R}^{m \times n}$, compute rank-k approximation ZW^T, where Z is $m \times k$ and W^T is $k \times n$.





- Problem with diverse applications
  - from scientific computing: fast solvers for integral equations, H-matrices
  - to data analytics: principal component analysis, image processing, ...

$$Ax \rightarrow ZW^T x$$

$$\textit{Flops} \quad 2mn \rightarrow 2(m+n)k$$

Singular value decomposition

Given $A \in \mathbb{R}^{m\times n}$, $m \geq n$ its singular value decomposition is

$$A = U\Sigma V^T = \begin{pmatrix} U_1 & U_2 & U_3 \end{pmatrix} \cdot \begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} V_1 & V_2 \end{pmatrix}^T$$

where

- U is $m \times m$ orthogonal matrix, the left singular vectors of A , U_1 is $m \times k$, U_2 is $m \times n - k$, U_3 is $m \times m - n$
- Σ is $m \times n$, its diagonal is formed by $\sigma_1(A) \geq \ldots \geq \sigma_n(A) \geq 0$ Σ_1 is $k \times k$, Σ_2 is $n - k \times n - k$
- V is $n \times n$ orthogonal matrix, the right singular vectors of A, V_1 is $n \times k$, V_2 is $n \times n - k$

Norms

$$
\begin{aligned}
||A||_p &= \max_{||x||_p=1} ||Ax||_p \\
||A||_F &= \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2} = \sqrt{\sigma_1^2(A) + \ldots \sigma_n^2(A)} \\
||A||_2 &= \sigma_{max}(A) = \sigma_1(A)
\end{aligned}
$$

Some properties:

$$||A||_2 \leq ||A||_F \leq \sqrt{min(m,n)}||A||_2$$

Orthogonal Invariance: If $Q \in \mathbb{R}^{m\times m}$ and $Z \in \mathbb{R}^{n\times n}$ are orthogonal, then

$$||QAZ||_F = ||A||_F$$
$$||QAZ||_2 = ||A||_2$$

Low rank matrix approximation

- Best rank-k approximation $A_k = U_k \Sigma_k V_k$ is rank-k truncated SVD of A [Eckart and Young, 1936]

$$\min_{rank(\tilde{A}_k) \leq k} ||A - \tilde{A}_k||_2 = ||A - A_k||_2 = \sigma_{k+1}(A) \quad (1)$$

$$\min_{rank(\tilde{A}_k) \leq k} ||A - \tilde{A}_k||_F = ||A - A_k||_F = \sqrt{\sum_{j=k+1}^{n} \sigma_j^2(A)} \quad (2)$$

Image, size 1190×1920

Rank-10 approximation, SVD

Rank-50 approximation, SVD

- Image source: `https://pixabay.com/photos/billiards-ball-play-number-half-4345870/`

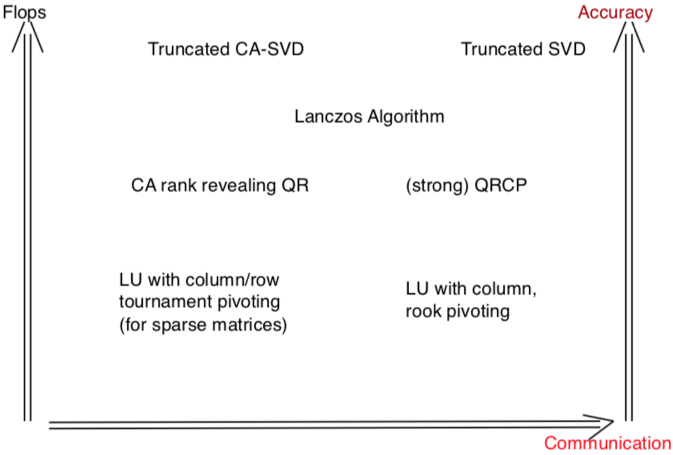
Large data sets

Matrix A might not exist entirely at a given time, rows or columns are added progressively.

- Streaming algorithm: can solve an arbitrarily large problem with one pass over the data (a row or a column at a time).
- Weakly streaming algorithm: can solve a problem with $O(1)$ passes over the data.

Matrix A might exist only implicitly, and it is never formed explicitly.

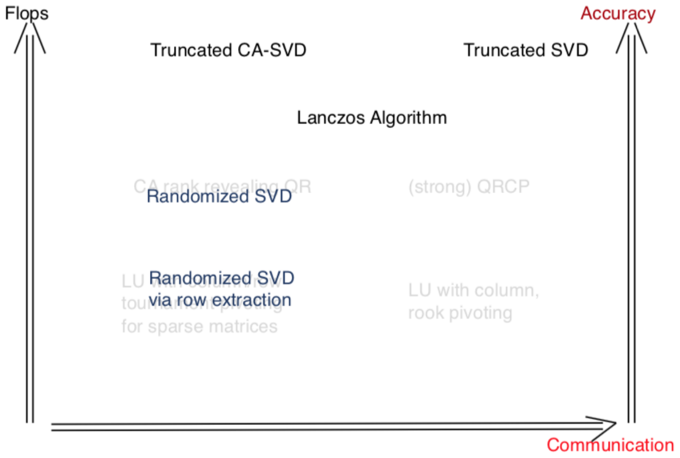
Low rank matrix approximation: trade-offs



Communication optimal if computing a rank-k approximation on P processors requires

$$\# \text{ messages} = \Omega\left(\log_2 P\right).$$

Low rank matrix approximation: trade-offs



Communication optimal if computing a rank-k approximation on P processors requires

$$\# \text{ messages} = \Omega\left(\log_2 P\right).$$

Idea underlying many algorithms

Compute $\tilde{A}_k = \mathcal{P}A$, where $\mathcal{P} = \mathcal{P}^o$ or $\mathcal{P} = \mathcal{P}^{so}$ is obtained as:

1. Construct a low dimensional subspace $X = range(A\Omega_1)$, $\Omega_1 \in \mathbb{R}^{n \times l}$ that approximates well the range of A, e.g.

$$\|A - \mathcal{P}^o A\|_2 \leq \gamma \sigma_{k+1}(A), \text{ for some } \gamma \geq 1,$$

   where Q_1 is orth. basis of $(A\Omega_1)$

$$\mathcal{P}^o = A\Omega_1(A\Omega_1)^+ = Q_1 Q_1^T, \text{ or equiv } \mathcal{P}^o a_j := arg \min_{x \in X} \|x - a_j\|_2$$

2. Select a semi-inner product $\langle \Theta_1 \cdot, \Theta_1 \cdot \rangle_2$, $\Theta_1 \in \mathbb{R}^{l' \times m}$ $l' \geq l$, define

$$\mathcal{P}^{so} = A\Omega_1(\Theta_1 A\Omega_1)^+ \Theta_1, \text{ or equiv } \mathcal{P}^{so} a_j := arg \min_{x \in X} \|\Theta_1(x - a_j)\|_2$$

Idea underlying many algorithms

Compute $\tilde{A}_k = \mathcal{P}A$, where $\mathcal{P} = \mathcal{P}^o$ or $\mathcal{P} = \mathcal{P}^{so}$ is obtained as:

1. Construct a low dimensional subspace $X = range(A\Omega_1)$, $\Omega_1 \in \mathbb{R}^{n \times l}$ that approximates well the range of A, e.g.

$$\|A - \mathcal{P}^o A\|_2 \leq \gamma \sigma_{k+1}(A), \text{ for some } \gamma \geq 1,$$

   where Q_1 is orth. basis of $(A\Omega_1)$

$$\mathcal{P}^o = A\Omega_1(A\Omega_1)^+ = Q_1 Q_1^T, \text{ or equiv } \mathcal{P}^o a_j := arg \min_{x \in X} \|x - a_j\|_2$$

2. Select a semi-inner product $\langle \Theta_1 \cdot, \Theta_1 \cdot \rangle_2$, $\Theta_1 \in \mathbb{R}^{l' \times m}$ $l' \geq l$, define

$$\mathcal{P}^{so} = A\Omega_1(\Theta_1 A\Omega_1)^+ \Theta_1, \text{ or equiv } \mathcal{P}^{so} a_j := arg \min_{x \in X} \|\Theta_1(x - a_j)\|_2$$

Properties of the approximations

Definitions and some of the results taken from [Demmel et al., 2019].

Definition

[low-rank approximation] A matrix A_k satisfying $\|A - A_k\|_2 \leq \gamma\sigma_{k+1}(A)$ for some $\gamma \geq 1$ will be said to be a (k, γ) *low-rank approximation of* A.

Definition

[spectrum preserving] If A_k satisfies

$$\sigma_j(A) \geq \sigma_j(A_k) \geq \gamma^{-1}\sigma_j(A)$$

for $j \leq k$ and some $\gamma \geq 1$, it is a (k, γ) *spectrum preserving*.

Definition

[kernel approximation] If A_k satisfies

$$\sigma_{k+j}(A) \leq \sigma_j(A - A_k) \leq \gamma\sigma_{k+j}(A)$$

for $1 \leq j \leq n - k$ and some $\gamma \geq 1$, it is a (k, γ) *kernel approximation of* A.

Plan

Low rank matrix approximation

Rank revealing QR factorization

Randomized algorithms for low rank approximation

Rank revealing QR factorization

Given A of size $m \times n$, consider the decomposition

$$AP_c = QR = Q \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix}, \tag{3}$$

where R_{11} is $k \times k$, P_c and k are chosen such that $||R_{22}||_2$ is small and R_{11} is well-conditioned.

- By the interlacing property of singular values [Golub, Van Loan, 4th edition, page 487],

  $$\sigma_i(R_{11}) \leq \sigma_i(A) \text{ and } \sigma_j(R_{22}) \geq \sigma_{k+j}(A)$$

  for $1 \leq i \leq k$ and $1 \leq j \leq n-k$.
- $\sigma_{k+1}(A) \leq \sigma_{max}(R_{22}) = ||R_{22}||$

Rank revealing QR factorization

Given A of size $m \times n$, consider the decomposition

$$AP_c = QR = Q \begin{bmatrix} R_{11} & R_{12} \\ & R_{22} \end{bmatrix}. \tag{4}$$

If $||R_{22}||_2$ is small,

- $Q(:,1:k)$ forms an approximate orthogonal basis for the range of A,

$$\begin{aligned} A(:,j) &= \sum_{i=1}^{min(j,k)} R(i,j)Q(:,i) \quad \in span\{Q(:,1), \ldots Q(:,k)\} \\ Range(A) &\in span\{Q(:,1), \ldots Q(:,k)\} \end{aligned}$$

- $P_c \begin{bmatrix} -R_{11}^{-1}R_{12} \\ I \end{bmatrix}$ is an approximate right null space of A.

Rank revealing QR factorization

The factorization from equation (5) is rank revealing if

$$1 \leq \frac{\sigma_i(A)}{\sigma_i(R_{11})}, \frac{\sigma_j(R_{22})}{\sigma_{k+j}(A)} \leq \gamma_1(n, k),$$

for $1 \leq i \leq k$ and $1 \leq j \leq \min(m, n) - k$, where

$$\sigma_{max}(A) = \sigma_1(A) \geq \ldots \geq \sigma_{min}(A) = \sigma_n(A)$$

It is **strong** rank revealing [Gu and Eisenstat, 1996] if in addition

$$||R_{11}^{-1} R_{12}||_{max} \leq \gamma_2(n, k)$$

Low rank approximation with strong RRQR

Given $A \in \mathbb{R}^{m \times n}$ and $R_{11} \in \mathbb{R}^{k \times k}$,

$$
\begin{aligned}
AP_c &= QR = \begin{pmatrix} Q_1 & Q_2 \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} \\ & R_{22} \end{pmatrix}, \\
\tilde{A}_{qr} &= Q_1 \begin{pmatrix} R_{11} & R_{12} \end{pmatrix} P_c^T = Q_1 Q_1^T A = \mathcal{P}^o A
\end{aligned}
$$

- It can be shown that

$$\sigma_j(R_{22}) = \sigma_j(A - \tilde{A}_{qr})$$

- [Gu and Eisenstat, 1996] show that given k and f, there exists permutation $V \in \mathbb{R}^{n \times n}$ such that the factorization satisfies,

$$
\begin{aligned}
1 \leq \frac{\sigma_i(A)}{\sigma_i(R_{11})}, \frac{\sigma_j(R_{22})}{\sigma_{k+j}(A)} &\leq \gamma(n,k), \quad \gamma(n,k) = \sqrt{1 + f^2 k(n-k)} \\
||R_{11}^{-1} R_{12}||_{max} &\leq f
\end{aligned}
$$

  for $1 \leq i \leq k$ and $1 \leq j \leq \min(m,n) - k$.
- Cost: $4mnk$ (QRCP) plus $O(mnk)$ flops and $O(k \log_2 P)$ messages.

$\rightarrow$ $\tilde{A}_{qr}$ with strong RRQR is $(k, \gamma(n,k))$ spectrum preserving and kernel approximation of A

QR with column pivoting [Businger and Golub, 1965]

Idea:

- At first iteration, trailing columns decomposed into parallel part to first column (or e_1) and orthogonal part (in rows $2:m$).
- The column of maximum norm is the column with largest component orthogonal to the first column.

Implementation:

- Find at each step of the QR factorization the column of maximum norm.
- Permute it into leading position.
- If rank(A) = k, at step $k+1$ the maximum norm is 0.
- No need to compute the column norms at each step, but just update them since

$$Q^T v = w = \begin{bmatrix} w_1 \\ w(2:n) \end{bmatrix}, \;\; ||w(2:n)||_2^2 = ||v||_2^2 - w_1^2$$

QR with column pivoting [Businger and Golub, 1965]

Sketch of the algorithm

column norm vector: $colnrm(j) = ||A(:,j)||_2, j = 1:n$.

for j = 1 : n **do**

Find column p of largest norm

if $colnrm[p] > \epsilon$ **then**

1. Pivot: swap columns j and p in A and modify $colnrm$.
2. Compute Householder matrix H_j s.t. $H_j A(j:m,j) = \pm ||A(j:m,j)||_2 e_1$.
3. Update $A(j:m, j+1:n) = H_j A(j:m, j+1:n)$.
4. Norm downdate $colnrm(j+1:n)^2 -= A(j, j+1:n)^2$.

else Break

end if

end for

If algorithm stops after k steps

$$\sigma_{max}(R_{22}) \leq \sqrt{n-k} \max_{1 \leq j \leq n-k} ||R_{22}(:,j)||_2 \leq \sqrt{n-k}\,\epsilon$$

Strong RRQR [Gu and Eisenstat, 1996]

Since

$$det(R_{11}) = \prod_{i=1}^{k} \sigma_i(R_{11}) = \sqrt{det(A^T A)} / \prod_{i=1}^{n-k} \sigma_i(R_{22})$$

a stron RRQR is related to a large $det(R_{11})$. The following algorithm interchanges columns that increase $det(R_{11})$, given f and k.

Compute a strong RRQR factorization, given k:

Compute $A\Pi = QR$ by using QRCP
while there exist i and j such that $det(\tilde{R}_{11})/det(R_{11}) > f$, where
$R_{11} = R(1:k, 1:k)$, $\Pi_{i,j+k}$ permutes columns i and $j+k$,
$R\Pi_{i,j+k} = \tilde{Q}\tilde{R}, \tilde{R}_{11} = \tilde{R}(1:k, 1:k)$ **do**
Find i and j
Compute $R\Pi_{i,j+k} = \tilde{Q}\tilde{R}$ and $\Pi = \Pi\Pi_{i,j+k}$
end while

Strong RRQR (contd)

It can be shown that

$$\frac{det(\tilde{R}_{11})}{det(R_{11})} = \sqrt{\left(R_{11}^{-1}R_{12}\right)_{i,j}^{2} + \omega_i^2\left(R_{11}\right)\chi_j^2\left(R_{22}\right)} \tag{5}$$

for any $1 \leq i \leq k$ and $1 \leq j \leq n-k$ (the 2-norm of the j-th column of A is $\chi_j(A)$, and the 2-norm of the j-th row of A^{-1} is $\omega_j(A)$).

Compute a strong RRQR factorization, given k:

Compute $A\Pi = QR$ by using QRCP

while $\max_{1\leq i\leq k, 1\leq j\leq n-k} \sqrt{\left(R_{11}^{-1}R_{12}\right)_{i,j}^{2} + \omega_i^2\left(R_{11}\right)\chi_j^2\left(R_{22}\right)} > f$ **do**

Find i and j such that $\sqrt{\left(R_{11}^{-1}R_{12}\right)_{i,j}^{2} + \omega_i^2\left(R_{11}\right)\chi_j^2\left(R_{22}\right)} > f$

Compute $R\Pi_{i,j+k} = \tilde{Q}\tilde{R}$ and $\Pi = \Pi\Pi_{i,j+k}$

end while

Strong RRQR (contd)

- $det(R_{11})$ strictly increases with every permutation, no permutation repeats, hence there is a finite number of permutations to be performed.

Strong RRQR (contd)

Theorem

[Gu and Eisenstat, 1996] If the QR factorization with column pivoting as in equation (5) *satisfies inequality*

$$\sqrt{\left(R_{11}^{-1}R_{12}\right)_{i,j}^{2} + \omega_i^2\left(R_{11}\right)\chi_j^2\left(R_{22}\right)} < f$$

for any $1 \le i \le k$ *and* $1 \le j \le n-k$, *then*

$$1 \le \frac{\sigma_i(A)}{\sigma_i(R_{11})}, \frac{\sigma_j(R_{22})}{\sigma_{k+j}(A)} \le \sqrt{1 + f^2 k(n-k)},$$

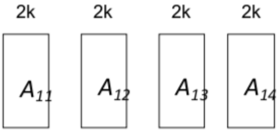
for any $1 \le i \le k$ *and* $1 \le j \le \min(m,n) - k$.

Deterministic column selection: tournament pivoting

1D tournament pivoting (1Dc-TP)

- 1D column block partition of A, select k cols from each block with strong RRQR

$$\begin{array}{cccc} (A_{11} & A_{12} & A_{13} & A_{14}) \\ \| & \| & \| & \| \\ (Q_{00}R_{00}P_{c00}^T & Q_{10}R_{10}P_{c10}^T & Q_{20}R_{20}P_{c20}^T & Q_{30}R_{30}P_{c30}^T \\ \downarrow & \downarrow & \downarrow & \downarrow \\ I_{00} & I_{10} & I_{20} & I_{30} \end{array}$$



- Reduction tree to select k cols from sets of $2k$ cols,

$$\begin{array}{cc} (A(:, I_{00} \cup I_{10}) & A(:, I_{20} \cup I_{30});) \\ \| & \| \\ (Q_{01}R_{01}P_{c01}^T & Q_{11}R_{11}P_{c11}^T) \\ \downarrow & \downarrow \\ I_{01} & I_{11} \end{array}$$

$$A(:, I_{01} \cup I_{11}) = Q_{02}R_{02}P_{c02}^T \rightarrow I_{02}$$

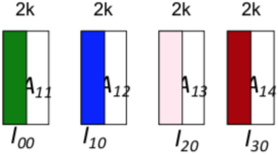
- Return selected columns $A(:, I_{02})$

Deterministic column selection: tournament pivoting

1D tournament pivoting (1Dc-TP)

- 1D column block partition of A, select k cols from each block with strong RRQR

$$
\begin{array}{cccc}
(A_{11} & A_{12} & A_{13} & A_{14}) \\
\| & \| & \| & \| \\
(Q_{00} R_{00} P_{c00}^T & Q_{10} R_{10} P_{c10}^T & Q_{20} R_{20} P_{c20}^T & Q_{30} R_{30} P_{c30}^T \\
\downarrow & \downarrow & \downarrow & \downarrow \\
I_{00} & I_{10} & I_{20} & I_{30}
\end{array}
$$



- Reduction tree to select k cols from sets of $2k$ cols,

$$
\begin{array}{cc}
(A(:, I_{00} \cup I_{10}) & A(:, I_{20} \cup I_{30});) \\
\| & \| \\
(Q_{01} R_{01} P_{c01}^T & Q_{11} R_{11} P_{c11}^T) \\
\downarrow & \downarrow \\
I_{01} & I_{11}
\end{array}
$$

$$A(:, I_{01} \cup I_{11}) = Q_{02} R_{02} P_{c02}^T \rightarrow I_{02}$$

- Return selected columns $A(:, I_{02})$

Deterministic column selection: tournament pivoting

1D tournament pivoting (1Dc-TP)

- 1D column block partition of A, select k cols from each block with strong RRQR

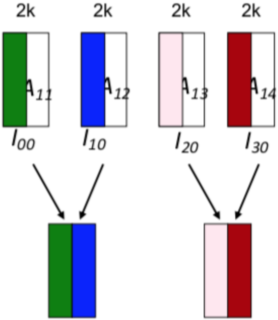
$$\begin{array}{cccc} (A_{11} & A_{12} & A_{13} & A_{14}) \\ \| & \| & \| & \| \\ (Q_{00}R_{00}P_{c00}^T & Q_{10}R_{10}P_{c10}^T & Q_{20}R_{20}P_{c20}^T & Q_{30}R_{30}P_{c30}^T) \\ \downarrow & \downarrow & \downarrow & \downarrow \\ I_{00} & I_{10} & I_{20} & I_{30} \end{array}$$

- Reduction tree to select k cols from sets of $2k$ cols,

$$\begin{array}{cc} (A(:, I_{00} \cup I_{10}) & A(:, I_{20} \cup I_{30});) \\ \| & \| \\ (Q_{01}R_{01}P_{c01}^T & Q_{11}R_{11}P_{c11}^T) \\ \downarrow & \downarrow \\ I_{01} & I_{11} \end{array}$$

$$A(:, I_{01} \cup I_{11}) = Q_{02}R_{02}P_{c02}^T \rightarrow I_{02}$$

- Return selected columns $A(:, I_{02})$

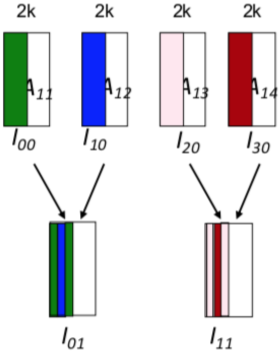


Deterministic column selection: tournament pivoting

1D tournament pivoting (1Dc-TP)

- 1D column block partition of A, select k cols from each block with strong RRQR

$$
\begin{array}{cccc}
(A_{11} & A_{12} & A_{13} & A_{14}) \\
\| & \| & \| & \| \\
(Q_{00}R_{00}P_{c00}^T & Q_{10}R_{10}P_{c10}^T & Q_{20}R_{20}P_{c20}^T & Q_{30}R_{30}P_{c30}^T) \\
\downarrow & \downarrow & \downarrow & \downarrow \\
I_{00} & I_{10} & I_{20} & I_{30}
\end{array}
$$



- Reduction tree to select k cols from sets of $2k$ cols,

$$
\begin{array}{cc}
(A(:, I_{00} \cup I_{10}) & A(:, I_{20} \cup I_{30});) \\
\| & \| \\
(Q_{01}R_{01}P_{c01}^T & Q_{11}R_{11}P_{c11}^T) \\
\downarrow & \downarrow \\
I_{01} & I_{11}
\end{array}
$$

$$A(:, I_{01} \cup I_{11}) = Q_{02}R_{02}P_{c02}^T \rightarrow I_{02}$$

- Return selected columns $A(:, I_{02})$

Deterministic column selection: tournament pivoting

1D tournament pivoting (1Dc-TP)

- 1D column block partition of A, select k cols from each block with strong RRQR

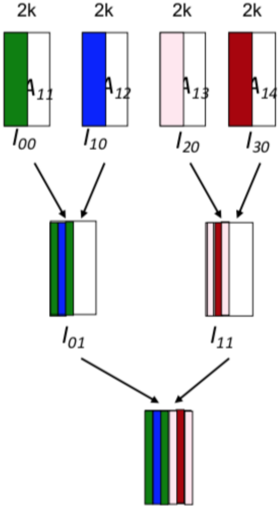
$$\begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ \| & \| & \| & \| \\ Q_{00}R_{00}P_{c00}^T & Q_{10}R_{10}P_{c10}^T & Q_{20}R_{20}P_{c20}^T & Q_{30}R_{30}P_{c30}^T \\ \downarrow & \downarrow & \downarrow & \downarrow \\ I_{00} & I_{10} & I_{20} & I_{30} \end{pmatrix}$$

- Reduction tree to select k cols from sets of $2k$ cols,

$$\begin{pmatrix} A(:, I_{00} \cup I_{10}) & A(:, I_{20} \cup I_{30}); \\ \| & \| \\ Q_{01}R_{01}P_{c01}^T & Q_{11}R_{11}P_{c11}^T \\ \downarrow & \downarrow \\ I_{01} & I_{11} \end{pmatrix}$$

$$A(:, I_{01} \cup I_{11}) = Q_{02}R_{02}P_{c02}^T \rightarrow I_{02}$$

- Return selected columns $A(:, I_{02})$



Deterministic column selection: tournament pivoting

1D tournament pivoting (1Dc-TP)

- 1D column block partition of A, select k cols from each block with strong RRQR

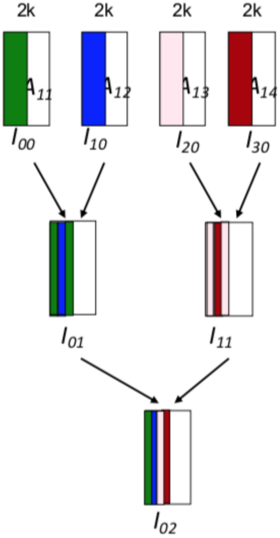
$$
\begin{array}{cccc}
(A_{11} & A_{12} & A_{13} & A_{14}) \\
\| & \| & \| & \| \\
(Q_{00}R_{00}P_{c00}^T & Q_{10}R_{10}P_{c10}^T & Q_{20}R_{20}P_{c20}^T & Q_{30}R_{30}P_{c30}^T) \\
\downarrow & \downarrow & \downarrow & \downarrow \\
I_{00} & I_{10} & I_{20} & I_{30}
\end{array}
$$

- Reduction tree to select k cols from sets of $2k$ cols,

$$
\begin{array}{cc}
(A(:, I_{00} \cup I_{10}) & A(:, I_{20} \cup I_{30});) \\
\| & \| \\
(Q_{01}R_{01}P_{c01}^T & Q_{11}R_{11}P_{c11}^T) \\
\downarrow & \downarrow \\
I_{01} & I_{11}
\end{array}
$$

$$A(:, I_{01} \cup I_{11}) = Q_{02}R_{02}P_{c02}^T \rightarrow I_{02}$$

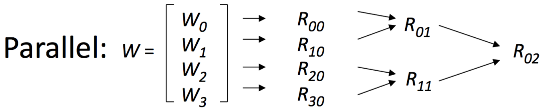
- Return selected columns $A(:, I_{02})$



Select k columns from a tall and skinny matrix

Given W of size $m \times 2k$, $m >> k$, k columns are selected as:

$W = QR_{02}$ using TSQR
$R_{02} P_c = Q_2 R_2$ using QRCP
Return $WP_c(:, 1:k)$



Rank revealing properties of CA-RRQR

It is shown in [Demmel et al., 2015] that the column permutation computed by CA-RRQR satisfies

$$\chi_j^2\left(R_{11}^{-1}R_{12}\right)+\left(\chi_j\left(R_{22}\right)/\sigma_{\min}(R_{11})\right)^2\leq F_{TP}^2,\ \text{for } j=1,\ldots,n-k. \qquad (6)$$

where F_{TP} depends on k, f, n, the shape of reduction tree used during tournament pivoting, and the number of iterations of CARRQR.

CA-RRQR - bounds for one tournament

Selecting k columns by using tournament pivoting reveals the rank of A with the following bounds:

$$1 \leq \frac{\sigma_i(A)}{\sigma_i(R_{11})}, \frac{\sigma_j(R_{22})}{\sigma_{k+j}(A)} \leq \sqrt{1 + F_{TP}^2(n-k)},$$

$$||R_{11}^{-1}R_{12}||_{max} \leq F_{TP}$$

- Binary tree of depth $\log_2(n/k)$,

$$F_{TP} \leq \frac{1}{\sqrt{2k}} (n/k)^{\log_2\left(\sqrt{2}fk\right)}. \qquad (7)$$

  The upper bound is a decreasing function of k when $k > \sqrt{n/(\sqrt{2}f)}$.
- Flat tree of depth n/k,

$$F_{TP} \leq \frac{1}{\sqrt{2k}} \left(\sqrt{2}fk\right)^{n/k}. \qquad (8)$$

CA-RRQR : 2D tournament pivoting

- A distributed on $P_r \times P_c$ procs as e.g.

$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \end{pmatrix}$$

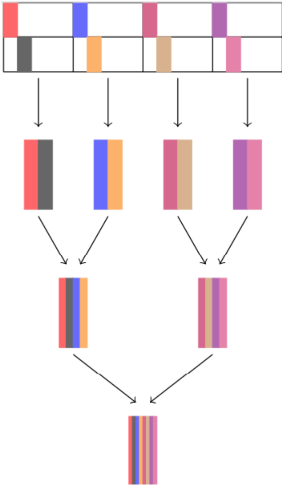
- Select k cols from each column block by 1Dr-TP,

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} \quad \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} \quad \begin{pmatrix} A_{13} \\ A_{23} \end{pmatrix} \quad \begin{pmatrix} A_{14} \\ A_{24} \end{pmatrix}$$
$$\downarrow \qquad\quad \downarrow \qquad\quad \downarrow \qquad\quad \downarrow$$
$$I_{00} \qquad I_{10} \qquad I_{20} \qquad I_{30}$$

- Apply 1Dc-TP on sets of k selected cols,

$$A(:, I_{00}) \quad A(:, I_{10}) \quad A(:, I_{20}) \quad A(:, I_{30})$$

- Return columns selected by 1Dc-TP $A(:, I_{02})$



CA-RRQR : 2D tournament pivoting

- A distributed on $P_r \times P_c$ procs as e.g.

$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \end{pmatrix}$$

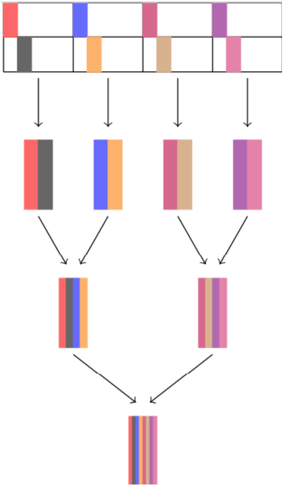
- Select k cols from each column block by 1Dr-TP,

$$\begin{matrix} \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} & \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} & \begin{pmatrix} A_{13} \\ A_{23} \end{pmatrix} & \begin{pmatrix} A_{14} \\ A_{24} \end{pmatrix} \\ \downarrow & \downarrow & \downarrow & \downarrow \\ I_{00} & I_{10} & I_{20} & I_{30} \end{matrix}$$

- Apply 1Dc-TP on sets of k selected cols,

$$A(:, I_{00}) \quad A(:, I_{10}) \quad A(:, I_{20}) \quad A(:, I_{30})$$

- Return columns selected by 1Dc-TP $A(:, I_{02})$



CA-RRQR : 2D tournament pivoting

- A distributed on $P_r \times P_c$ procs as e.g.

$$A = \begin{pmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \end{pmatrix}$$

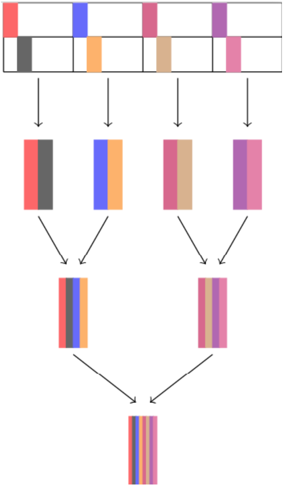
- Select k cols from each column block by 1Dr-TP,

$$\begin{matrix} \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} & \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} & \begin{pmatrix} A_{13} \\ A_{23} \end{pmatrix} & \begin{pmatrix} A_{14} \\ A_{24} \end{pmatrix} \\ \downarrow & \downarrow & \downarrow & \downarrow \\ I_{00} & I_{10} & I_{20} & I_{30} \end{matrix}$$

- Apply 1Dc-TP on sets of k selected cols,

$$A(:, I_{00}) \quad A(:, I_{10}) \quad A(:, I_{20}) \quad A(:, I_{30})$$

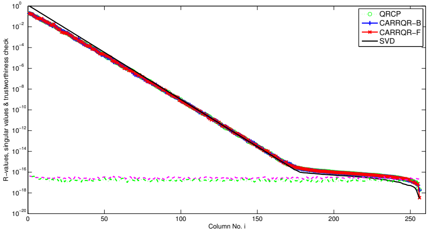
- Return columns selected by 1Dc-TP $A(:, I_{02})$

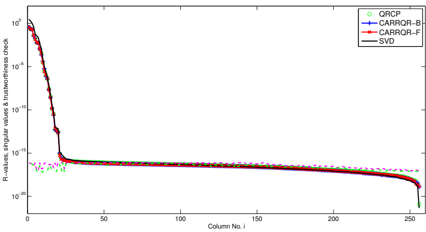


Numerical results

- Stability close to QRCP for many tested matrices.
- Absolute value of diagonals of R, L referred to as R-values, L-values.
- Methods compared
  - RRQR: QR with column pivoting
  - CA-RRQR-B with tournament pivoting based on binary tree
  - CA-RRQR-F with tournament pivoting based on flat tree
  - SVD

Numerical results (contd)





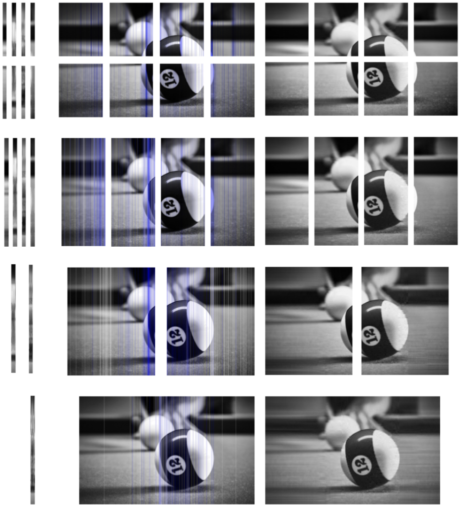
- Left: exponent - exponential Distribution, $\sigma_1 = 1$, $\sigma_i = \alpha^{i-1}$ $(i = 2, \dots, n)$, $\alpha = 10^{-1/11}$ [Bischof, 1991]
- Right: shaw - 1D image restoration model [Hansen, 2007]

$$\epsilon \min\{||(A\Pi_0)(:,i)||_2\,, ||(A\Pi_1)(:,i)||_2\,, ||(A\Pi_2)(:,i)||_2\} \tag{9}$$

$$\epsilon \max\{||(A\Pi_0)(:,i)||_2\,, ||(A\Pi_1)(:,i)||_2\,, ||(A\Pi_2)(:,i)||_2\} \tag{10}$$

where $\Pi_j (j = 0, 1, 2)$ are the permutation matrices obtained by QRCP, CARRQR-B, and CARRQR-F, and ϵ is the machine precision.

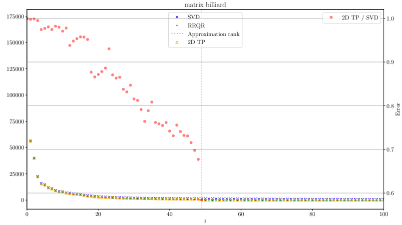
CA-RRQR : 2D tournament pivoting



Numerical experiments

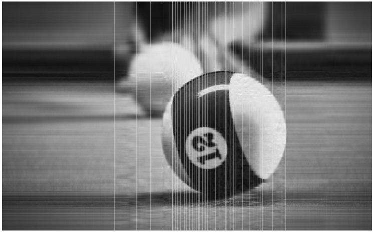
Original image, size 1190×1920

Singular values and ratios



Rank-10 approx, 2D TP 8×8 procs

Rank-50 approx, 2D TP 8×8 procs



- Image source: `https://pixabay.com/photos/billiards-ball-play-number-half-4345870/`

Plan

Low rank matrix approximation

Rank revealing QR factorization

Randomized algorithms for low rank approximation

Randomized algorithms - main idea

- Construct a low dimensional subspace that captures the action of A.
- Restrict A to the subspace and compute a standard QR or SVD factorization.

Obtained as follows:

1. Compute an approximate basis for the range of A ($m \times n$)
   find Q ($m \times k$) with orthonormal columns and approximate A by the projection of its columns onto the space spanned by Q:

$$A \approx QQ^T A$$

2. Use Q to compute a standard factorization of A

Source: Halko et al, *Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decomposition*, SIREV 2011.

Johnson-Lindenstrauss transform

Definition 3 from [Woodruff, 2014].
A random matrix $\Omega_1 \in \mathbb{R}^{k \times m}$ is a Johnson-Lindenstrauss transform with parameters ϵ, δ, n, or JLT(n, ϵ, δ), if with probability at least $1 - \delta$ for any n-element subset $V \subset \mathbb{R}^m$, for all $x_i, x_j \in V$, we have

$$|\langle \Omega_1 x_i, \Omega_1 x_j \rangle - \langle x_i, x_j \rangle| \leq \epsilon \|x_i\|_2 \|x_j\|_2 \tag{11}$$

- If $x_i = x_j$ we obtain $\|\Omega_1 x_i\|_2^2 = (1 \pm \epsilon)\|x_i\|_2^2$.
- It can also be expressed as: given all vectors $x_i, x_j \in V$ are rescaled to be unit vectors, then for all $x_i, x_j \in V$ we require to hold:

$$\begin{aligned} \|\Omega_1 x_i\|_2^2 &= (1 \pm \epsilon)\|x_i\|_2^2 && (12) \\ \|\Omega_1 (x_i + x_j)\|_2^2 &= (1 \pm \epsilon)\|x_i + x_j\|_2^2 && (13) \end{aligned}$$

Proof that we obtain relation (14):

$$\begin{aligned} \langle \Omega_1 x_i, \Omega_1 x_j \rangle &= \left(\|\Omega_1 (x_i + x_j)\|_2^2 - \|\Omega_1 x_i\|_2^2 - \|\Omega_1 x_j\|_2^2\right)/2 \\ &= \left((1 \pm \epsilon)\|x_i + x_j\|_2^2 - (1 \pm \epsilon)\|x_i\|_2^2 - (1 \pm \epsilon)\|x_j\|_2^2\right)/2 \\ &= \langle x_i, x_j \rangle \pm O(\epsilon) \end{aligned}$$

Johnson-Lindenstrauss transform (contd)

Let $\Omega_1 \in \mathbb{R}^{k \times m}$ be a matrix whose entries are independent standard normal random variables, multiplied by $1/\sqrt{k}$. If $k = O(\epsilon^{-2} \log(n/\delta))$, then Ω_1 is a JLT(n, ϵ, δ).

Source: Theorem 4 in [Woodruff, 2014], see also Theorem 2.1 and proof in S. Dasgupta, A. Gupta, 2003, *An Elementary Proof of a Theorem of Johnson and Lindenstrauss*

Oblivious subspace embedding

Let $\Omega_1 \in \mathbb{R}^{k\times m}$ be a matrix whose entries are independent standard normal random variables, multiplied by $1/\sqrt{k}$. If $k = O(\epsilon^{-2}(n + \log(1/\delta)))$, then Ω_1 is an oblivious subspace embedding (OSE) with parameters (n, ϵ, δ). That is, with probability at least $1 - \delta$ for any n-dimensional subspace $\mathbf{V} \subset \mathbb{R}^m$, for all $x_i, x_j \in \mathbf{V}$, we have

$$|\langle \Omega_1 x_i, \Omega_1 x_j\rangle - \langle x_i, x_j\rangle| \leq \epsilon \|x_i\|_2 \|x_j\|_2 \tag{14}$$

Source: Theorem 6 in [Woodruff, 2014]

Typical randomized truncated SVD

Algorithm

Input: $m \times n$ matrix A, desired rank k, $l = p + k$ exponent q.

1. Sample an $n \times l$ test matrix Ω_1 with independent mean-zero, unit-variance Gaussian entries.
2. Compute $Y = (AA^T)^q A\Omega_1$ /* Y is expected to span the column space of A */
3. Construct $Q \in \mathbb{R}^{m \times l}$ with columns forming an orthonormal basis for the range of Y.
4. Compute $B = Q^T A$
5. Compute the SVD of $B = \hat{U}\Sigma V^T$

Return the approximation $\tilde{A}_k = Q\hat{U} \cdot \Sigma \cdot V^T$

Randomized truncated SVD ($q = 0$)

The best approximation is when Q equals the first $k + p$ left singular vectors of A. Given $A = U\Sigma V^T$,

$$\begin{aligned} QQ^T A &= U(1:m, 1:k+p)\Sigma(1:k+p, 1:k+p)(V(1:n, 1:k+p) \\ ||A - QQ^T A||_2 &= \sigma_{k+p+1} \end{aligned}$$

Theorem 1.1 from Halko et al. If Ω_1 is chosen to be i.i.d. N(0,1), $k, p \geq 2$, $q = 1$, then the expectation with respect to the random matrix Ω_1 is

$$\mathbb{E}(||A - QQ^T A||_2) \leq \left(1 + \frac{4\sqrt{k+p}}{p-1}\sqrt{min(m,n)}\right)\sigma_{k+1}(A)$$

and the probability that the error satisfies

$$||A - QQ^T A||_2 \leq \left(1 + 11\sqrt{k+p} \cdot \sqrt{min(m,n)}\right)\sigma_{k+1}(A)$$

is at least $1 - 6/p^p$.

For $p = 6$, the probability becomes .99.

Randomized truncated SVD

Theorem 10.6, Halko et al. Average spectral norm. Under the same hypotheses as Theorem 1.1 from Halko et al.,

$$\mathbb{E}(||A - QQ^T A||_2) \leq \left(1 + \sqrt{\frac{k}{p-1}}\right)\sigma_{k+1}(A) + \frac{e\sqrt{k+p}}{p}\left(\sum_{j=k+1}^{n} \sigma_j^2(A)\right)^{1/2}$$

- Fast decay of singular values:
  If $\left(\sum_{j>k} \sigma_j^2(A)\right)^{1/2} \approx \sigma_{k+1}$ then the approximation should be accurate.
- Slow decay of singular values:
  If $\left(\sum_{j>k} \sigma_j^2(A)\right)^{1/2} \approx \sqrt{n-k}\sigma_{k+1}$ and n large, then the approximation might not be accurate.

Source: G. Martinsson's talk

Power iteration $q \geq 1$

The matrix $(AA^T)^q A$ has a faster decay in its singular values:

- has the same left singular vectors as A
- its singular values are:

$$\sigma_j((AA^T)^q A) = (\sigma_j(A))^{2q+1}$$

Cost of randomized truncated SVD

- Randomized SVD requires $2q + 1$ passes over the matrix.
- The last 3 steps of the algorithms cost:
  (2) Compute $Y = (AA^T)^q A\Omega_1$: $2(2q+1) \cdot nnz(A) \cdot (k+p)$
  (3) Compute QR of Y: $2m(k+p)^2$
  (4) Compute $B = Q^T A$: $2nnz(A) \cdot (k+p)$
  (5) Compute SVD of B: $O(n(k+p)^2)$
- If $nnz(A)/m \geq k + p$ and $q = 1$, then (2) and (4) dominate (3).
- To be faster than deterministic approaches, the cost of (2) and (4) need to be reduced.

Fast Johnson-Lindenstrauss transform

Find sparse or structured Ω_1 such that computing $A\Omega_1$ is cheap, e.g. a subsampled random Hadamard transform (SRHT).
Given $n = 2^p, l < n$, the SRHT ensemble embedding $\mathbb{R}^n$ into $\mathbb{R}^l$ is defined as

$$\Omega_1 = \sqrt{\frac{n}{l}} \cdot P \cdot H \cdot D, \text{ where} \tag{15}$$

- $D \in \mathbb{R}^{n \times n}$ is diagonal matrix of uniformly random signs, random variables uniformly distributed on ± 1
- $H \in \mathbb{R}^{n \times n}$ is the normalized Walsh-Hadamard transform
- $P \in \mathbb{R}^{l \times n}$ formed by subset of l rows of the identity, chosen uniformly at random (draws l rows at random from HD).

References: Sarlos'06, Ailon and Chazelle'06, Liberty, Rokhlin, Tygert and Woolfe'06.

Fast Johnson-Lindenstrauss transform (contd)

Definition of Normalized Walsh–Hadamard Matrix
For given $n = 2^p$, $H_n \in \mathbb{R}^{n \times n}$ is the non-normalized Walsh-Hadamard transform defined recursively as,

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad H_n = \begin{pmatrix} H_{n/2} & H_{n/2} \\ H_{n/2} & -H_{n/2} \end{pmatrix}. \tag{16}$$

The normalized Walsh-Hadamard transform is $H = n^{-1/2} H_n$.

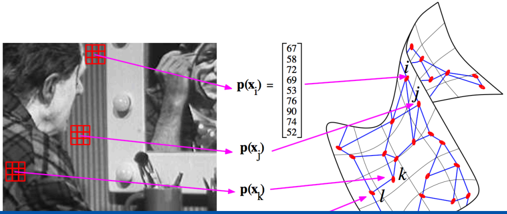
Cost of matrix vector multiplication (Theorem 2.1 in [Ailon and Liberty, 2008]):
For $x \in \mathbb{R}^n$ and $\Omega_1 \in \mathbb{R}^{l \times n}$, computing $\Omega_1 x$ costs $2n \log_2(l+1)$ flops.

Results from image processing (from Halko et al)

- A matrix A of size 9025×9025 arising from a diffusion geometry approach.
- A is a graph Lapacian on the manifold of 3×3 patches.
- 95×95 pixel grayscale image, intensity of each pixel is an integer ≤ 4095.
- Vector $x^{(i)} \in \mathbb{R}^9$ gives the intensities of the pixels in a 3×3 neighborhood of pixel i.
- W reflects similarities between patches, $\sigma = 50$ reflects the level of sensitivity,

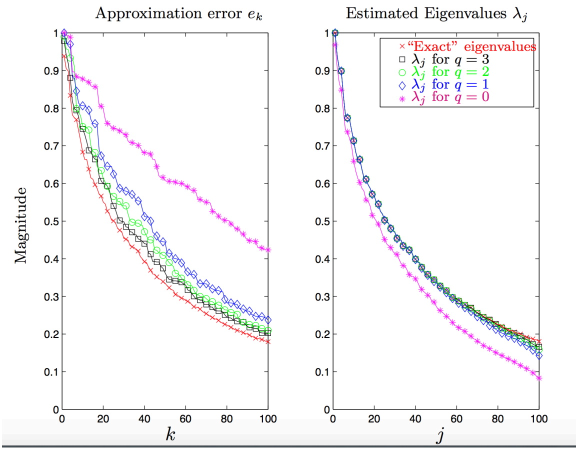
$$w_{ij} = exp\{-||x^{(i)} - x^{(j)}||^2/\sigma^2\},$$

- Sparsify W, compute dominant eigenvectors of $A = D^{-1/2} W D^{-1/2}$.



Experimental results (from Halko et al)

- Approximation error : $||A - QQ^T A||_2$
- Estimated eigenvalues for $k = 100$



More details on CA deterministic algorithms

- [Demmel et al., 2015] Communication avoiding rank revealing QR factorization with column pivoting Demmel, Grigori, Gu, Xiang, SIAM J. Matrix Analysis and Applications, 2015.
- Low rank approximation of a sparse matrix based on LU factorization with column and row tournament pivoting, with S. Cayrols and J. Demmel, Inria TR 8910.

References (1)

Ailon, N. and Liberty, E. (2008).
Fast dimension reduction using rademacher series on dual bch codes.
In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 1–9, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Bischof, C. H. (1991).
A parallel QR factorization algorithm with controlled local pivoting.
SIAM J. Sci. Stat. Comput., 12:36–57.

Businger, P. A. and Golub, G. H. (1965).
Linear least squares solutions by Householder transformations.
Numer. Math., 7:269–276.

Demmel, J., Grigori, L., Gu, M., and Xiang, H. (2015).
Communication-avoiding rank-revealing qr decomposition.
SIAM Journal on Matrix Analysis and its Applications, 36(1):55–89.

Demmel, J., Grigori, L., and Rusciano, A. (2019).
An improved analysis and unified perspective on deterministic and randomized low rank matrix approximations.
Technical report, Inria.
available at https://arxiv.org/abs/1910.00223.

Eckart, C. and Young, G. (1936).
The approximation of one matrix by another of lower rank.
Psychometrika, 1:211–218.

Eisenstat, S. C. and Ipsen, I. C. F. (1995).
Relative perturbation techniques for singular value problems.
SIAM J. Numer. Anal., 32(6):1972–1988.

References (2)

Gu, M. and Eisenstat, S. C. (1996).
Efficient algorithms for computing a strong rank-revealing QR factorization.
SIAM J. Sci. Comput., 17(4):848–869.

Hansen, P. C. (2007).
Regularization tools: A matlab package for analysis and solution of discrete ill-posed problems.
Numerical Algorithms, (46):189–194.

Woodruff, D. P. (2014).
Sketching as a tool for numerical linear algebra.
Found. Trends Theor. Comput. Sci., 10(1–2):1–157.

Results used in the proofs

- Interlacing property of singular values [Golub, Van Loan, 4th edition, page 487]
  Let $A = [a_1| \dots |a_n]$ be a column partitioning of an $m \times n$ matrix with $m \geq n$. If $A_r = [a_1| \dots |a_r]$, then for $r = 1 : n-1$

  $$\sigma_1(A_{r+1}) \geq \sigma_1(A_r) \geq \sigma_2(A_{r+1}) \geq \dots \geq \sigma_r(A_{r+1}) \geq \sigma_r(A_r) \geq \sigma_{r+1}(A_{r+1}).$$

- Given $n \times n$ matrix B and $n \times k$ matrix C, then ([Eisenstat and Ipsen, 1995], p. 1977)

  $$\sigma_{min}(B)\sigma_j(C) \leq \sigma_j(BC) \leq \sigma_{max}(B)\sigma_j(C), j = 1, \dots, k.$$