jh160025-DAHI

片桐孝洋（東京大学）

High-performance Randomized Matrix Computations for Big Data Analytics and Applications

Background

- Developing random sketching algorithms with high-performance implementations on supercomputers to compute **singular value decomposition (SVD) and linear system (LS)** solutions of very large-scale matrices.
- Few numerical solvers, especially **randomized algorithms**, are designed to tackle very large-scale matrix computations on the latest supercomputers. .
- We intend to develop efficient sketching schemes to compute approximate SVD and LS solutions of large-scale matrices. The main idea is to sketch the matrices by randomized algorithms **to reduce the computational dimensions and then suitably integrate the sketches** to improve the accuracy and to lower the computational costs.
- We intend to implement the proposed algorithms on supercomputers. One essential component of this project is to develop effective **automatic software auto-tuning (AT)** technologies, so that the package can fully take advantage of the computational capabilities of the target supercomputers that include CPU homogeneous and CPU-GPU heterogeneous parallel computers.

Members

- **Takahiro Katagiri** (Nagoya U., Japan) : AT (ppOpen-AT), parallel eigenvalue algorithms, and supercomputer implementations.
- **Weichung Wang** (National Taiwan U., Taiwan): Numerical linear algebra, parallel computing, and AT (surrogate-assisted turning). and big data applications.
- **Su-Yun Huang** (Institute of Statistical Science, Academia Sinica, Taiwan) : Mathematical statistics and machine learning (random sketching algorithm).
- **Kengo Nakajima** (U. Tokyo, Japan) : Parallel algorithms in numerical iterative method (hybrid MPI/OpenMP execution).
- **Osni Marques** (LBNL, USA) : Eigenproblem and its implementation (LAPACK, SVD algorithms).
- **Feng-Nan Hwang** (National Central U., Taiwan) : Eigenproblem and its parallelization (SLEPc, SVD algorithms)
- **Toshio Endo** (TITECH, Japan) : System software (optimizations for hierarchical memory and adaptation of its AT)

iSVD Algorithm

Rank-k SVD

$$\boldsymbol{A} \approx \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^\top$$

$\boldsymbol{U}_k$ is an $m \times k$ orthonormal matrix that $k < m$, $\boldsymbol{\Sigma}_k$ is a $k \times k$ diagonal matrix, and $\boldsymbol{V}_k$ is an $n \times k$ orthonormal matrix. The columns of $\boldsymbol{U}_k$ and $\boldsymbol{V}_k$ are the leading left singular vectors and right singular vectors of $\boldsymbol{A}$, respectively. The diagonal entries of $\boldsymbol{\Sigma}_k$ are the k largest singular values of $\boldsymbol{A}$.

Randomized SVD (rSVD)

1. Sample a random matrix for low-dimensional projection.
2. Project *A* to the corresponding low-dimensional subspace.
3. Find a small-sized SVD in the low-dimensional subspace.

Algorithm 1 Randomized SVD with single sketch (rSVD).

Require: Input $\boldsymbol{A}$ (real $m \times n$ matrix), k (desired rank of approximate SVD), p (oversampling parameter), $\ell = k + p$ (dimension of the sketched column space), q (power of projection)

Ensure: Approximate rank-k SVD of $\boldsymbol{A} \approx \widetilde{\boldsymbol{U}}_k \widetilde{\boldsymbol{\Sigma}}_k \widetilde{\boldsymbol{V}}_k^\top$

1: Generate an $n \times \ell$ random matrix $\boldsymbol{\Omega}$ ($= \boldsymbol{\Omega}_{gp}$ or $\boldsymbol{\Omega}_{cs}$)
2: Assign $\boldsymbol{Y} \leftarrow (\boldsymbol{A}\boldsymbol{A}^\top)^q \boldsymbol{A}\boldsymbol{\Omega}$
3: Compute $\boldsymbol{Q}$ whose columns are orthonormal basis of $\boldsymbol{Y}$
4: Compute SVD of $\boldsymbol{Q}^\top \boldsymbol{A} = \widetilde{\boldsymbol{W}}_\ell \widetilde{\boldsymbol{\Sigma}}_\ell \widetilde{\boldsymbol{V}}_\ell^\top$
5: Assign $\widetilde{\boldsymbol{U}}_\ell \leftarrow \boldsymbol{Q}\widetilde{\boldsymbol{W}}_\ell$
6: Extract the largest k singular-pairs from $\widetilde{\boldsymbol{U}}_\ell$, $\widetilde{\boldsymbol{\Sigma}}_\ell$, $\widetilde{\boldsymbol{V}}_\ell$ to obtain $\widetilde{\boldsymbol{U}}_k$, $\widetilde{\boldsymbol{\Sigma}}_k$, $\widetilde{\boldsymbol{V}}_k$.

[By Ting-Li Chen, Su-Yun Huang, Hung Chen, David Chang, Chen-Yao Lin, and Weichung Wang]

Algorithm 2 Integrated SVD with multiple sketches (iSVD).

Require: Input $\boldsymbol{A}$ (real $m \times n$ matrix), k (desired rank of approximate SVD), p (oversampling parameter), $\ell = k + p$ (dimension of the sketched column space), q (power of projection), N (number of random sketches)

Ensure: Approximate rank-k SVD of $\boldsymbol{A} \approx \widehat{\boldsymbol{U}}_k \widehat{\boldsymbol{\Sigma}}_k \widehat{\boldsymbol{V}}_k^\top$

1: Generate $n \times \ell$ random matrices $\boldsymbol{\Omega}_{[i]}$ for $i = 1, \ldots, N$
2: Assign $\boldsymbol{Y}_{[i]} \leftarrow (\boldsymbol{A}\boldsymbol{A}^\top)^q \boldsymbol{A}\boldsymbol{\Omega}_{[i]}$ for $i = 1, ..., N$ with $\boldsymbol{\Omega}_{[i]} = \boldsymbol{\Omega}_{gp}$ or $\boldsymbol{\Omega}_{cs}$ (in parallel)
3: Compute $\boldsymbol{Q}_{[i]}$ whose columns are orthonormal basis of $\boldsymbol{Y}_{[i]}$ (in parallel)
4: Integrate $\overline{\boldsymbol{Q}} \leftarrow \{\boldsymbol{Q}_{[i]}\}_{i=1}^N$ (by Algorithm 3 or Algorithm 4)
5: Compute SVD of $\overline{\boldsymbol{Q}}^\top \boldsymbol{A} = \widehat{\boldsymbol{W}}_\ell \widehat{\boldsymbol{\Sigma}}_\ell \widehat{\boldsymbol{V}}_\ell^\top$
6: Assign $\widehat{\boldsymbol{U}}_\ell \leftarrow \overline{\boldsymbol{Q}}\widehat{\boldsymbol{W}}_\ell$
7: Extract the largest k singular-pairs from $\widehat{\boldsymbol{U}}_\ell$, $\widehat{\boldsymbol{\Sigma}}_\ell$, $\widehat{\boldsymbol{V}}_\ell$ to obtain $\widehat{\boldsymbol{U}}_k$, $\widehat{\boldsymbol{\Sigma}}_k$, $\widehat{\boldsymbol{V}}_k$

Algorithm 3 Integration of $\{\boldsymbol{Q}_{[i]}\}_{i=1}^N$ based on Kolmogorov-Nagumo type averages.

Require: $\boldsymbol{Q}_{[1]}, \boldsymbol{Q}_{[2]}, \ldots, \boldsymbol{Q}_{[N]}$

Ensure: Integrated $\overline{\boldsymbol{Q}}$ defined in (3.1)

1: Assign the current iterate $\boldsymbol{Q}_c \leftarrow \boldsymbol{Q}_{[1]}$ (or initialized with another $\boldsymbol{Q}_{[i]}$)
2: **while** (not convergent) **do**
3: Compute $\varphi_{\boldsymbol{Q}_c}(\overline{\boldsymbol{P}}) = (\boldsymbol{I} - \boldsymbol{Q}_c\boldsymbol{Q}_c^\top)\overline{\boldsymbol{P}}\boldsymbol{Q}_c$.
4: Select a step size τ_+
5: Perform the inverse mapping $\varphi_{\boldsymbol{Q}_c}^{-1}(\tau_+ \varphi_{\boldsymbol{Q}_c}(\overline{\boldsymbol{P}}))$ to get $\boldsymbol{Q}_+$ defined in (3.7)
6: Assign $\boldsymbol{Q}_c \leftarrow \boldsymbol{Q}_+$
7: **end while**
8: Output $\overline{\boldsymbol{Q}} = \boldsymbol{Q}_c$

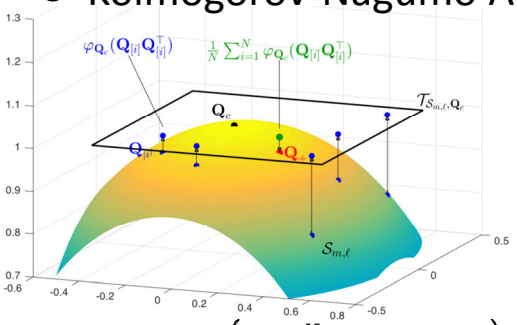
Algorithm 4 Integration of $\{\boldsymbol{Q}_{[i]}\}_{i=1}^N$ based on gradient ascent method.

Require: $\boldsymbol{Q}_{[1]}, \boldsymbol{Q}_{[2]}, \ldots, \boldsymbol{Q}_{[N]}$

Ensure: Integrated $\overline{\boldsymbol{Q}}$ that is the solution of (3.1) and (3.8)

1: Assign the current iterate $\boldsymbol{Q}_c \leftarrow \boldsymbol{Q}_{[1]}$ (or initialized with another $\boldsymbol{Q}_{[i]}$)
2: **while** (not convergent) **do**
3: Compute the gradient $\boldsymbol{G}_F(\boldsymbol{Q}_c) = \overline{\boldsymbol{P}}\boldsymbol{Q}_c$ and the projected gradient $\boldsymbol{D}_F(\boldsymbol{Q}_c) = (\boldsymbol{I} - \boldsymbol{Q}_c\boldsymbol{Q}_c^\top)\overline{\boldsymbol{P}}\boldsymbol{Q}_c$
4: Set the current search curve $\boldsymbol{Q}(\tau) = (\boldsymbol{I} - \frac{\tau}{2}\boldsymbol{M}_c)^{-1}(\boldsymbol{I} + \frac{\tau}{2}\boldsymbol{M}_c)\boldsymbol{Q}_c \in \mathcal{S}_{m,\ell}$
5: Compute the next iterate $\boldsymbol{Q}_+ = \boldsymbol{Q}(\tau_+)$ by a selected step-size τ_+
6: Assign $\boldsymbol{Q}_c \leftarrow \boldsymbol{Q}_+$
7: **end while.**
8: Output $\overline{\boldsymbol{Q}} = \boldsymbol{Q}_c$

Kolmogorov-Nagumo Average



$$\boldsymbol{Q}_+\boldsymbol{Q}_+^\top \leftarrow \varphi_{\boldsymbol{Q}_c}^{-1}\left(\frac{\tau_+}{N}\sum_{i=1}^N \varphi_{\boldsymbol{Q}_c}(\boldsymbol{Q}_{[i]}\boldsymbol{Q}_{[i]}^\top)\right)$$

One Step Moving of iSVD