Optimal CUR Matrix Decompositions

Christos Boutsidis[1] David P. Woodruff [2]

Yahoo! Labs
New York

IBM Research
Almaden

Singular Value Decomposition

- $m \times n$ matrix A
- $k < \rho = \mathrm{rank}(A)$
- Low-rank matrix approximation problem:

$$\min_{X \in \mathbb{R}^{m \times n}, \mathrm{rank}(X) \leq k} ||A - X||_{\mathrm{F}}$$

- **Singular Value Decomposition (SVD):**

$$A = U \cdot \Sigma \cdot V^{\mathrm{T}} = \underbrace{\begin{pmatrix} U_k & U_{\rho-k} \end{pmatrix}}_{m \times \rho} \underbrace{\begin{pmatrix} \Sigma_k & \mathbf{0} \\ \mathbf{0} & \Sigma_{\rho-k} \end{pmatrix}}_{\rho \times \rho} \underbrace{\begin{pmatrix} V_k^T \\ V_{\rho-k}^T \end{pmatrix}}_{\rho \times n}$$

- $U_k \in \mathbb{R}^{m \times k}$, $\Sigma_k \in \mathbb{R}^{k \times k}$, and $V_k \in \mathbb{R}^{n \times k}$

Solution via Eckart-Young Theorem

$A_k = U_k \Sigma_k V_k^{\mathrm{T}} = A V_k V_k^{\mathrm{T}}$. $O(mn \min\{m, n\})$ time

CUR Matrix Decomposition

CUR replaces the left and right singular vectors in the SVD with actual columns and rows from the matrix, respectively

$$\left(\begin{array}{c} \\ \\ \mathbf{A} \\ \\ \\ \end{array}\right) = \left(\begin{array}{c} \\ \\ \mathbf{C} \\ \\ \\ \end{array}\right) \left(\ \mathbf{U}\ \right) \left(\quad \mathbf{R} \quad\right) + \left(\begin{array}{c} \\ \\ \mathbf{E} \\ \\ \\ \end{array}\right)$$

$$\left(\begin{array}{c} \\ \\ \mathbf{A} \\ \\ \\ \end{array}\right) = \left(\begin{array}{c} \\ \\ \mathbf{U}_k \\ \\ \\ \end{array}\right) \left(\ \Sigma_k\ \right) \left(\quad \mathbf{V}_k \quad\right) + \left(\begin{array}{c} \\ \\ \mathbf{E} \\ \\ \\ \end{array}\right)$$

Motivation

$$\left(\quad \textbf{user-by-movie} \quad \right) \approx \left(\quad \textbf{users} \quad \right) \left(\; \textbf{U} \; \right) \left(\quad \textbf{movies} \quad \right)$$

- A: users-by-movies ratings matrix.
- C: contains the most “important” users.
- R: contains the most “important” movies.

Optimization problem

Definition (The CUR Problem)

Given

- $\mathsf{A} \in \mathbb{R}^{m \times n}$
- $k < \mathrm{rank}(\mathsf{A})$
- $\varepsilon > 0$

construct

- $\mathsf{C} \in \mathbb{R}^{m \times c}$
- $\mathsf{R} \in \mathbb{R}^{r \times n}$
- $\mathsf{U} \in \mathbb{R}^{c \times r}$

such that:

$$\|\mathsf{A} - \mathsf{CUR}\|_{\mathrm{F}}^2 \leq (1 + \varepsilon) \cdot \|\mathsf{A} - \mathsf{A}_k\|_{\mathrm{F}}^2.$$

with c, r, and $\mathrm{rank}(\mathsf{U})$ **being as small as possible.**

Prior art

Sub-optimal and **randomized** algorithms.

	c	r	$\mathrm{rank}(\mathrm{U})$	$\Vert\mathrm{A}-\mathrm{CUR}\Vert_{\mathrm{F}}^2 \leq$	Time
1	k/ε^2	k/ε	k	$\Vert\mathrm{A}-\mathrm{A}_k\Vert_{\mathrm{F}}^2+\varepsilon\Vert\mathrm{A}\Vert_{\mathrm{F}}^2$	$nnz(\mathrm{A})$
2	k/ε^4	k/ε^2	k	$\Vert\mathrm{A}-\mathrm{A}_k\Vert_{\mathrm{F}}^2+\varepsilon\Vert\mathrm{A}\Vert_{\mathrm{F}}^2$	$nnz(\mathrm{A})$
3	$(k\log k)/\varepsilon^2$	$(k\log k)/\varepsilon^4$	$(k\log k)/\varepsilon^2$	$(1+\varepsilon)\Vert\mathrm{A}-\mathrm{A}_k\Vert_{\mathrm{F}}^2$	n^3
4	$(k\log k)/\varepsilon^2$	$(k\log k)/\varepsilon^2$	$(k\log k)/\varepsilon^2$	$(2+\varepsilon)\Vert\mathrm{A}-\mathrm{A}_k\Vert_{\mathrm{F}}^2$	n^3
5	k/ε	k/ε^2	k/ε	$(1+\varepsilon)\Vert\mathrm{A}-\mathrm{A}_k\Vert_{\mathrm{F}}^2$	n^2k/ε

References:

1. Drineas and Kannan. Symposium on Foundations of Computer Science, 2003.
2. Drineas, Kannan, and Mahoney. SIAM Journal on Computing, 2006.
3. Drineas, Mahoney, and Muthukrishnan. SIAM Journal on Matrix Analysis, 2008.
4. Drineas and Mahoney. Proceedings of the National Academy of Sciences, 2009.
5. Wang and Zhang. Journal of Machine Learning Research, 2013.

Open problems

1. **Optimal CUR**: Can we find relative-error CUR algorithms selecting the optimal number of columns and rows, together with a matrix U with optimal rank?

2. **Input-sparsity-time CUR**: Can we find relative-error CUR algorithms running in input-sparsity-time ($nnz(A)$ time)?

3. **Deterministic CUR**: Can we find relative-error CUR algorithms that are deterministic and run in poly time?

Contributions

1. **Optimal CUR**: *First* optimal CUR algorithms.
2. **Input-sparsity-time CUR**: *First* CUR algorithm with running time proportional to the non-zero entries of A.
3. **Deterministic CUR**: *First* deterministic algorithm for CUR that runs in polynomial time.

Lower bound

Theorem

Fix appropriate matrix $\mathsf{A} \in \mathbb{R}^{n \times n}$*. Consider a factorization* CUR,

$$\|\mathsf{A} - \mathsf{CUR}\|_{\mathrm{F}}^2 \leq (1 + \varepsilon)\|\mathsf{A} - \mathsf{A}_k\|_{\mathrm{F}}^2.$$

Then, for any $k \geq 1$ *and for any* $\varepsilon < 1/3$:

$$c = \Omega(k/\varepsilon),$$

and

$$r = \Omega(k/\varepsilon),$$

and

$$\mathrm{rank}(\mathsf{U}) \geq k/2.$$

Extended lower bound in [Deshpande and Vempala, 2006], [Boutsidis et al, 2011], [Sinop and Guruswami, 2011]

Input-sparsity-time CUR

Theorem

There exists a randomized algorithm to construct a CUR with

$$c = O(k/\varepsilon)$$

and

$$r = O(k/\varepsilon)$$

and

$$\mathrm{rank}\,(\mathrm{U}) = k$$

such that, with constant probability of success,

$$\|\mathrm{A} - \mathrm{CUR}\|_{\mathrm{F}}^2 \le (1+\varepsilon)\|\mathrm{A} - \mathrm{A}_k\|_{\mathrm{F}}^2.$$

Running time: $O\left(nnz\left(\mathrm{A}\right)\log n + (m+n)\cdot poly\left(\log n, k, 1/\varepsilon\right)\right).$

Deterministic CUR

Theorem

There exists a deterministic algorithm to construct a CUR with

$$c = O(k/\varepsilon)$$

and

$$r = O(k/\varepsilon)$$

and

$$\mathrm{rank}\,(\mathsf{U}) = k$$

such that

$$\|\mathsf{A} - \mathsf{CUR}\|_{\mathrm{F}}^2 \leq (1+\varepsilon)\|\mathsf{A} - \mathsf{A}_k\|_{\mathrm{F}}^2.$$

Running time: $O(mn^3 k/\varepsilon)$.

A proto-algorithm (step 1)

1. Construct C with $O(k/\varepsilon)$ columns:
   1. Compute/approx the top k singular vectors of A: $\mathrm{Z}_1 \in \mathbb{R}^{n\times k}$.
      $$\|\mathrm{A}-\mathrm{A}\mathrm{Z}_1\mathrm{Z}_1^{\mathrm{T}}\|_{\mathrm{F}}^2 \le (1+\varepsilon)\,\|\mathrm{A}-\mathrm{A}_k\|_{\mathrm{F}}^2$$
   2. Sample $O(k\log k)$ columns from $\mathrm{Z}_1^{\mathrm{T}}$ (equivalently from A) with leverage scores (Euclidean norms of columns of $\mathrm{Z}_1^{\mathrm{T}}$).

      Down-sample those columns to $c_1 = O(k)$ columns with Batson/Srivastava/Spielman (BSS) sampling ($\mathrm{C}_1 \in \mathbb{R}^{m\times c_1}$).
      $$\underbrace{\left(\quad \mathrm{Z}_1^{\mathrm{T}} \quad\right)}_{n} \longrightarrow \underbrace{\left(\quad \hat{\mathrm{Z}}_1^{\mathrm{T}} \quad\right)}_{O(k\log k)} \longrightarrow \underbrace{\left(\ \tilde{\mathrm{Z}}_1^{\mathrm{T}}\ \right)}_{O(k)}$$
      $$\|\mathrm{A}-\mathrm{C}_1\mathrm{C}_1^{\dagger}\mathrm{A}\|_{\mathrm{F}}^2 \le O(1)\|\mathrm{A}-\mathrm{A}\mathrm{Z}_1\mathrm{Z}_1^{\mathrm{T}}\|_{\mathrm{F}}^2$$
   3. Adaptively sample $c_2 = O(k/\varepsilon)$ columns of A:
      $$\|\mathrm{A}-\mathrm{C}\cdot\mathrm{D}\|_{\mathrm{F}}^2 \le \|\mathrm{A}-\mathrm{A}_k\|_{\mathrm{F}}^2 + (k/c_2)\|\mathrm{A}-\mathrm{C}_1{\mathrm{C}_1}^{\dagger}\mathrm{A}\|_{\mathrm{F}}^2$$

      D is a rank k matrix

Step 2

2. Construct R with $O(k/\varepsilon)$ rows:
   1. Find $Z_2 \in \mathbb{R}^{m\times k}$ in the span of C such that:
      $$\|A - Z_2 Z_2^T A\|_F^2 \leq (1+\varepsilon)\cdot\|A - A_k\|_F^2.$$
   2. How to do this efficiently?
      - Instead of projecting columns of A onto C, we project the columns of AW, where W is a random subspace embedding
      - Find best rank-k approximation of the columns of AW in C
   3. Sample $O(k\log k)$ rows with leverage scores (from Z_2).

      Down-sample those rows to $r_1 = O(k)$ rows with Batson/Spielman/Srivastava (BSS) sampling. ($R_1 \in \mathbb{R}^{r_1\times n}$)
      $$\|A - AR_1^{\dagger}R_1\|_F^2 \leq O(1)\|A - Z_2 Z_2^T A\|_F^2$$
   4. Sample $r_2 = O(k/\varepsilon)$ rows with adaptive sampling++
      $$\|A - Z_2 Z_2^T A R^{\dagger} R\|_F^2 \leq \|A - Z_2 Z_2^T A\|_F^2 + \frac{\mathrm{rank}(Z_2 Z_2^T A)}{r_2}\|A - A R_1{}^{\dagger} R_1\|_F^2$$

Input-sparsity-time CUR

Everything should run in $nnz(\mathrm{A}) \log n + poly(k, 1/\varepsilon)$ time.

1. **Existing tools:**
   - Input-sparsity-time SVD [Clarkson, W, STOC 2013].
   - Leverage-scores sampling [Drineas et al, SIMAX 2008].
   - Input-sparsity-time algorithm to find the “best” rank k approx to a matrix in a given subspace [Kannan et al, COLT 2014].
2. **New tools:**
   - Input-sparsity-time version of the BSS sampling method of [Boutsidis et al, 2011].
   - Input-sparsity-time version of adaptive sampling method [Desphande et al, 2006, Wang and Zhang, 2013].
   - Input-sparsity-time construction of U.

Deterministic CUR

Everything should run in polynomial time and be deterministic.

1. **Existing tools:**
   - Standard SVD algorithm.
   - Standard method to find the "best" rank k approximation to a matrix in a given subspace.
   - Batson/Spielman/Srivastava (BSS) sampling as in [Boutsidis et all, FOCS 2011].

2. **New tools:**
   - Derandomization of the adaptive sampling of [Desphande et al, RANDOM 2006] and [Wang and Zhang, JMLR 2013].

Conclusions

- **Optimal**, $(1+\varepsilon)$-error CUR with $O(k/\varepsilon)$ columns/rows.
- **Input-sparsity-time** algorithm.
- **Deterministic** polynomial-time algorithm.

- Extended abstract appeared in STOC 2014.
- Full version on ArXiv, June 2014.