"Mathematics is the art of giving the same name to different things."
- Henri Poincaré

Extra Project

Randomized SVD

Abror Shopulatov Mohammed Ibrahim Awad Imran Turganov

Mohamed bin Zayed University of Artificial Intelligence

November, 2025

Recall

During the course, we saw:

- Vectors and matrices
- Special matrices: rotations, reflections, projections
- Eigenvalues and eigenvectors — directions that matrices simply scale
- That lead us to matrix decompositions and SVD

Remark: Singular Value Decomposition (SVD)

Any matrix $A \in \mathbb{R}^{m \times n}$ can be factored as:

$$A = U \Sigma V^\top$$

where:

- $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ are orthogonal matrices
- $\Sigma \in \mathbb{R}^{m \times n}$ is diagonal with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r \geq 0$

SVD usage

SVD is everywhere in data science:

- Image and video compression, Principal Component Analysis (PCA)
- Recommender systems, Noise reduction and signal processing

Example (Image Compression with SVD)

Original Image
(full rank)

Rank-100 Approximation
Stores only 9.77% of original!

The Challenge

SVD works beautifully... but what happens we go higher dimentions?

Lemma: Classical SVD Complexity

Computing the full SVD of an $m \times n$ matrix requires:

$$\mathcal{O}(\min\{mn^2, m^2n\})$$

Matrix Size	**Operations**	**Time**
$1{,}000 \times 1{,}000$	$\sim 10^9$	seconds
$10{,}000 \times 10{,}000$	$\sim 10^{12}$	hours
$100{,}000 \times 100{,}000$	$\sim 10^{15}$	**infeasible**

The Problem

Key Observation

Computing SVD of **large matrices** is computationally expensive!

What we could do:

1. Work on faster *computers*
2. Look for smarter *algorithms*
3. Give up!

The real problem:

- **Classical SVD:** Computes *all* singular vectors and values
- **Our goal:** Capture only the **top-**k singular vectors

The Randomized Idea - Intuition

Key Insight: Random sampling preserves geometric structure with high probability

Example: Random Projection

Draw a random matrix $\Omega \in \mathbb{R}^{n\times k}$ with Gaussian entries, then:

$$Y = A\Omega \in \mathbb{R}^{m\times k}$$

What does Y capture?

- Each column of Y "probes" the range of A
- Large singular values of A dominate the response
- Small singular values contribute negligibly
- Result: column space of $Y \approx$ top-k subspace of A

Randomized SVD Algorithm

Definition: Algorithm: Randomized SVD

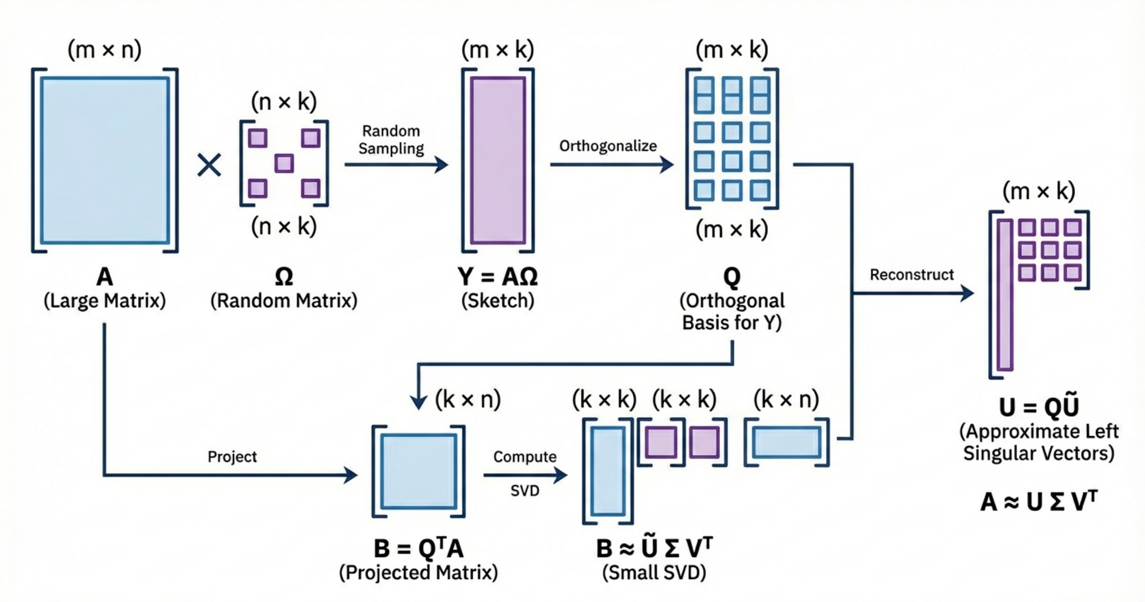
Input: Matrix $A \in \mathbb{R}^{m \times n}$, target rank k where $k \ll \min\{m, n\}$

Steps:

1. **Random sketch:** Draw $\Omega \in \mathbb{R}^{n \times k}$ with i.i.d. Gaussian entries $\mathcal{N}(0, 1)$
   Compute $Y = A\Omega$ [$O(mnk)$]
2. **Orthogonalize:** $Q = \text{orth}(Y)$ via QR factorization [$O(mk^2)$]
3. **Project:** $B = Q^\top A$ [$O(mnk)$]
4. **Small SVD:** Compute $B = \hat{U}_B \Sigma V^\top$ [$O(nk^2)$]
5. **Reconstruct:** $U = Q\hat{U}_B$ [$O(mk^2)$]

Output: $U \in \mathbb{R}^{m \times k}$, $\Sigma \in \mathbb{R}^{k \times k}$, $V \in \mathbb{R}^{n \times k}$
Total complexity: $O(mnk)$ vs. $O(\min\{mn^2, m^2 n\})$ for classical SVD

Visualization



Error Bounds

Lemma: Halko-Martinsson-Tropp (for the brave it can be omitted)

Let $A \in \mathbb{R}^{m \times n}$ with singular values $\sigma_1 \geq \cdots \geq \sigma_n$. Let $\Omega \in \mathbb{R}^{n \times (k+p)}$ have i.i.d. Gaussian entries, and $Q = \mathrm{orth}(A\Omega)$. Then:

Frobenius norm:

$$\mathbb{E}[\|A - QQ^\top A\|_F] \leq \left(1 + \frac{k}{p-1}\right)^{1/2} \left(\sum_{i=k+1}^{n} \sigma_i^2\right)^{1/2}$$

Spectral norm:

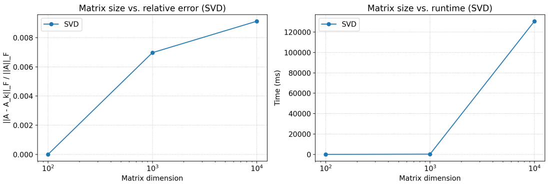
$$\mathbb{E}[\|A - QQ^\top A\|_2] \leq \left(1 + \frac{k}{p-1}\right)^{1/2} \sigma_{k+1}$$

Remark:

With oversampling $p = 10$, error is within $\sim 3\times$ of optimal rank-k approximation!

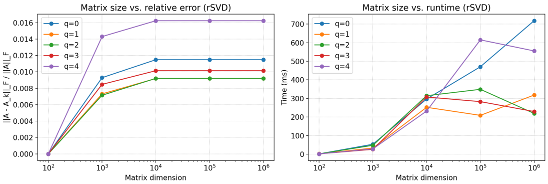
Numerical Results

Experimental setup: Random Gaussian matrices, $n \in \{10^2, 10^3, 10^4, 10^5, 10^6\}$, rank $k = 200$



Classical SVD

Cubic scaling makes it infeasible beyond 10^4



Randomized SVD

Scales to $10^6 \times 10^6$ with $q \in \{0, 1, 2, 3, 4\}$

Key observations:

- Classical SVD: 130+ seconds at $10^4 \times 10^4$
- rSVD: $<$1 second at $10^6 \times 10^6$ (with $q = 1$: 318 ms)
- Power iteration ($q \geq 2$) closes accuracy gap to machine precision

Conclusion

What we've shown:

- Classical SVD is powerful but computationally expensive: $\mathcal{O}(\min\{mn^2, m^2n\})$
- **Randomized SVD** achieves $\mathcal{O}(mn\log(k))$ complexity through random sampling
- **Rigorous error guarantees:** Near-optimal with small oversampling
- **Practical performance:** 10-100$\times$ speedup with negligible accuracy loss

Thank you!

Full paper and code: `https://github.com/IMRUNya/rSVD`