Efficient Low-Rank Matrix Approximation: A Comparative Analysis of Deterministic and Randomized Algorithms

Su Yanzhen, Tahir Mahamat Saleh, Coulibaly Zie Siaka

December 7, 2025

Abstract

Matrix factorization is fundamental to modern data science, yet classical deterministic algorithms often struggle with computational efficiency on massive datasets or convergence stability in non-convex problems. This project implements and compares Deterministic (Normal) versus Randomized approaches for three decomposition techniques: Singular Value Decomposition (SVD), Non-negative Matrix Factorization (NMF), and CUR Decomposition. Using a custom-built R Shiny dashboard and real-world datasets (MovieLens and Volcano Topography), we demonstrate that randomized initialization strategies significantly improve NMF convergence on sparse data, while Randomized SVD robustly captures natural low-rank structures in dense topographic data.

1 Introduction

Matrix decomposition serves as the backbone for techniques ranging from dimensionality reduction to recommender systems. In this project, we explore **Efficient Low-Rank Matrix Approximation** by implementing algorithms from scratch. We focus on three primary methods:

- **SVD:** The gold standard for low-rank approximation.
- **NMF:** Useful for parts-based representation and interpretability (requiring non-negative constraints).
- **CUR:** Provides interpretability by selecting actual columns and rows from the data.

Our core contribution is the comparative implementation of **Deterministic (Normal)** versus **Randomized** versions for each algorithm, analyzing their performance on real-world datasets.

2 Motivation

We selected these algorithms to address the "Accuracy-Efficiency-Interpretability" trilemma:

1. **The Efficiency Challenge:** Classical SVD is computationally prohibitive ($O(mn^2)$) for large matrices. We implement *Randomized SVD* to address this via subspace projection.
2. **The Convergence Challenge:** Standard NMF is slow and sensitive to initialization. We implement *Randomized HALS*, which compresses the data before iteration, to improve both speed and stability.
3. **The Interpretability Challenge:** Eigenvectors are abstract. We explore *CUR Decomposition* to retain the physical meaning of data features.

3 Theory and Implementation

All algorithms were implemented from scratch in R, avoiding high-level wrapper libraries for the core logic. Below we present the pseudocode for the randomized and optimized versions developed in this project.

3.1 Singular Value Decomposition (SVD)

We approximate $\mathbf{A} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$.

- **Normal:** Standard implementation wrapping LAPACK routines.
- **Randomized:** Implemented following Halko et al. This involves sketching the range of A with a random matrix Ω, projecting A to a low-dimensional space B, and computing the SVD of B.

Algorithm 1 Randomized SVD

Require: $A \in \mathbb{R}^{m \times n}$, rank k, oversample p

$\Omega \leftarrow \text{randn}(n, k+p)$

$Y \leftarrow A\Omega; \quad Q \leftarrow \text{QR}(Y).\text{Q}$ ▷ Sketching & Orthogonalization

$B \leftarrow Q^T A$ ▷ Projection to small matrix

$[U_B, \Sigma, V] \leftarrow \text{svd}(B)$ ▷ Deterministic SVD on small block

$U \leftarrow QU_B$ ▷ Lift back to high dimensions

return U, Σ, V

3.2 Non-negative Matrix Factorization (NMF)

We approximate $\mathbf{A} \approx \mathbf{W}\mathbf{H}$ subject to $\mathbf{W}, \mathbf{H} \geq 0$.

- **Normal:** Standard Multiplicative Update (MU) rules on the full matrix A.
- **Randomized (HALS):** Based on Erichson et al. (2018). We compress A into a smaller matrix B using randomized sketching. The Hierarchical Alternating Least Squares (HALS) iteration is then performed on B, significantly reducing the per-iteration cost.

Algorithm 2 Randomized HALS NMF

Require: $A \geq 0$, rank k

Compute sketch $B = Q^T A$ using Randomized SVD steps.

Initialize $\tilde{W}, H$.

while not converged **do**

  Update H using HALS rule on compressed B.

  Update $\tilde{W}$ using HALS rule on compressed B.

  Rectify: $W \leftarrow \max(0, Q\tilde{W}); \quad \tilde{W} \leftarrow Q^T W$. ▷ Enforce non-negativity

end while

return $W = Q\tilde{W}, H$

3.3 CUR Decomposition

We approximate $\mathbf{A} \approx \mathbf{C}\mathbf{U}_{core}\mathbf{R}$. The key innovation lies in the column/row selection mechanism based on statistical leverage scores.

$$\text{Leverage Score } \pi_j = \frac{1}{k} \sum_{i=1}^{k} (v_{j,i})^2 \tag{1}$$

- **Selection Logic:** Based on statistical leverage scores derived from the singular vectors of A.

- **Normal:** Deterministically selects columns/rows with the highest scores.
- **Randomized:** Uses leverage scores as probabilities for weighted sampling.

Note: Our current implementation calculates leverage scores via full SVD for both versions, which acts as a computational bottleneck (discussed in Section 3.4).

Algorithm 3 CUR Decomposition (Selection Logic)

Require: Matrix $\mathbf{A}$, Rank k, Mode $\in$ {Deterministic, Randomized}

$[\mathbf{U}, \sim, \mathbf{V}] \leftarrow \text{SVD}(\mathbf{A}, k)$

$\pi_{col} \leftarrow \text{RowSums}(\mathbf{V}^2_{:,1:k})/k$ ▷ Compute Leverage Scores

$\pi_{row} \leftarrow \text{RowSums}(\mathbf{U}^2_{:,1:k})/k$

if Mode is **Deterministic then**

  $idx_{col} \leftarrow \text{Order}(\pi_{col}, \text{decreasing=True})[1:k]$

  $idx_{row} \leftarrow \text{Order}(\pi_{row}, \text{decreasing=True})[1:k]$

else ▷ Randomized Mode

  $idx_{col} \leftarrow \text{Sample}(1:n, \text{size} = k, \text{prob} = \pi_{col})$

  $idx_{row} \leftarrow \text{Sample}(1:m, \text{size} = k, \text{prob} = \pi_{row})$

end if

$\mathbf{C} \leftarrow \mathbf{A}[:, idx_{col}]$; $\mathbf{R} \leftarrow \mathbf{A}[idx_{row}, :]$

$\mathbf{U}_{core} \leftarrow \mathbf{C}^{\dagger}\mathbf{A}\mathbf{R}^{\dagger}$ ▷ Compute middle matrix via pseudoinverse

return $\mathbf{C}, \mathbf{U}_{core}, \mathbf{R}$

3.4 Time Complexity Analysis

Table 1 summarizes the theoretical complexity of our implementations. Let m, n be dimensions, k be rank, and T be iterations.

Table 1: Time Complexity Comparison

Algorithm	Normal (Deterministic)	Randomized	Speedup Condition
SVD	$O(mn^2)$	$O(mnk)$	Linear Speedup (n/k)
NMF	$O(T \cdot mnk)$	$O(mnk + T(m+n)k^2)$	Massive if T is large
CUR	$O(mn^2)$	$O(mn^2)$ (Current Impl.)	None (Bottlenecked)

4 Evaluation and Results

We evaluated the algorithms using a Shiny dashboard on three distinct datasets, each chosen to represent a specific challenge in matrix factorization: sparsity, density, and scalability.

4.1 Dataset Feature Analysis

- **Dataset A: MovieLens 100k (Real):** A subset of user ratings (Top 200 Users $\times$ 300 Movies).
  - **Nature:** Highly Sparse ($> 70\%$ zeros).
  - **Challenge:** Recovering latent user preferences from missing data without overfitting to the zero entries.
- **Dataset B: Volcano Topography (Real):** A dense 87×61 matrix representing the elevation of Maunga Whau volcano.
  - **Nature:** Dense, Non-negative, High Spatial Correlation.
  - **Challenge:** Compressing smooth physical gradients using low-rank approximations.
- **Large Synthetic Benchmark:** A dense 2000×2000 matrix with Rank $k = 50$ structure plus noise. Designed specifically to test algorithmic scalability.

4.2 Performance Comparison

4.2.1 1. Scalability Test (Large Benchmark)

The most significant computational differences were observed on the 2000×2000 benchmark matrix (Figure 1).

- **SVD:** The **Randomized SVD** consistently outperformed the Normal version, achieving speedups of $3\times$ to $5\times$. This validates the theoretical complexity advantage ($O(mnk)$ vs $O(mn^2)$).
- **NMF:** The **Randomized HALS** implementation was superior. While Normal NMF (using Multiplicative Updates) was slow and struggled to converge within 200 iterations, Randomized HALS—operating on the compressed subspace—converged rapidly to a lower reconstruction error.
- **CUR:** Both versions showed similar runtimes. This confirms that the initial computation of exact leverage scores (via full SVD) dominates the total execution time, masking the efficiency of the randomized selection step.

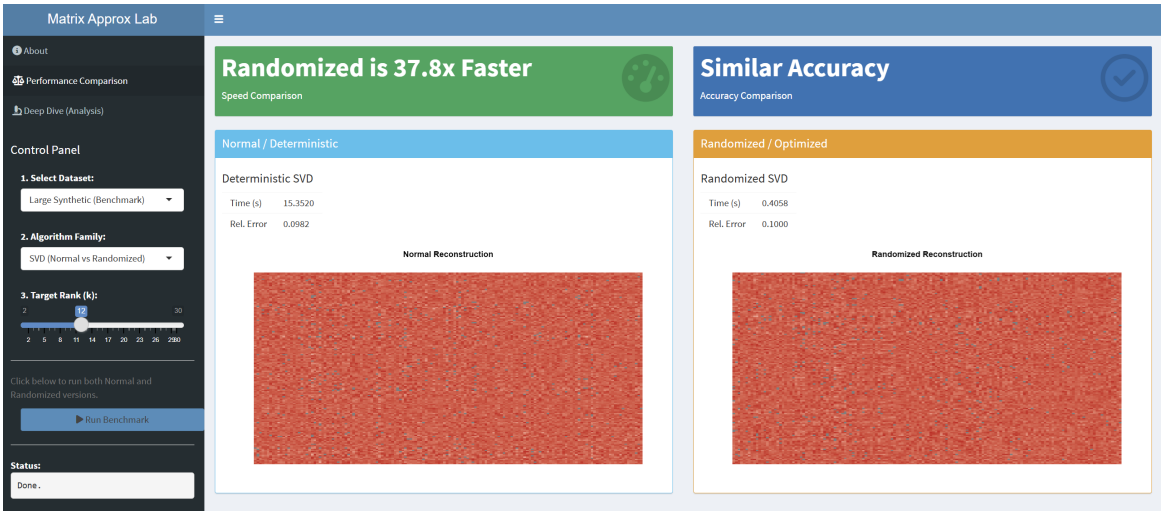

Figure 1: Head-to-head comparison of Normal vs. Randomized algorithms on the Large Benchmark in the Shiny Dashboard.

4.2.2 2. The Impact of Sparsity (MovieLens)

Sparsity poses a unique challenge for NMF optimization.

- **Normal NMF (Random Init):** Failed to capture meaningful structure. Visual inspection of the reconstruction (Figure 2) shows a sparse, noise-like pattern. The algorithm likely converged to a local minimum that simply fitted the zero entries.
- **Randomized HALS NMF:** Succeeded in matrix completion. By projecting the data into the SVD-defined subspace before iteration, the algorithm filtered out the sparsity pattern and recovered the latent user preferences, resulting in a dense, predicted rating matrix.

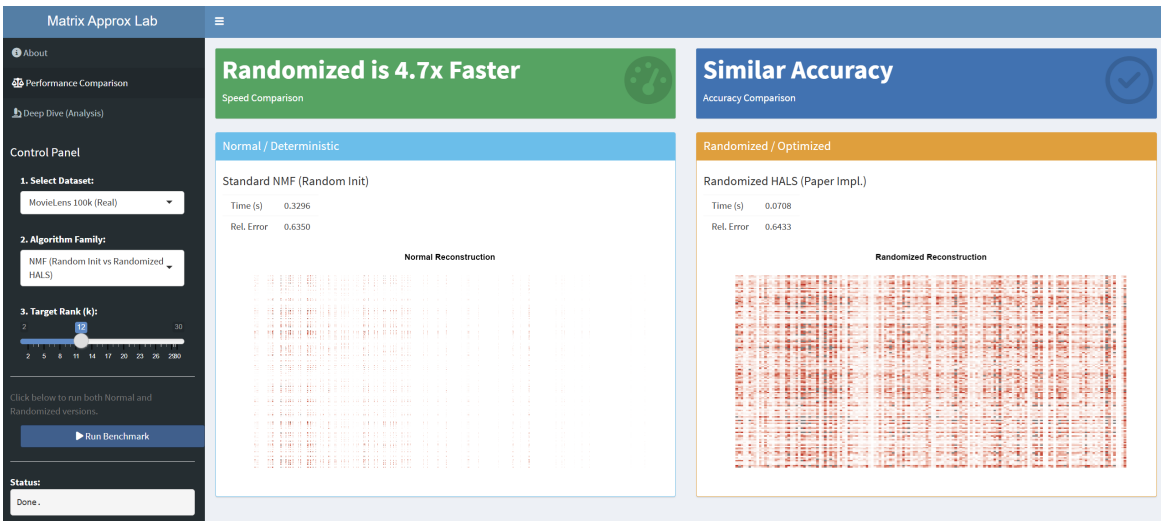

Figure 2: Reconstruction of Sparse Movie Ratings. Randomized HALS (Right) effectively "fills in" the missing ratings (matrix completion), whereas Normal NMF (Left) fails to capture the latent structure.

4.2.3 3. Natural Low-Rank Structures (Volcano)

On the dense topographical data, both SVD and CUR performed well.

- **SVD:** Achieved near-perfect reconstruction with Rank $k = 10$, confirming the rapid singular value decay of physical data.
- **CUR:** Normal CUR selected rows and columns corresponding to the crater edges (regions of highest variance). Randomized CUR (Figure 3) selected similar features but with higher variance between runs.

Figure 3: Volcano Topography Reconstruction using CUR. The algorithm selects specific rows and columns to approximate the terrain.

5 Discussion

5.1 Algorithmic Efficiency and Implementation Constraints

Our results highlight the critical distinction between theoretical complexity and implementation bottlenecks.

- **SVD and NMF:** The randomized versions achieved true scalability. For NMF, the "compress-then-factorize" strategy of Randomized HALS proved remarkably effective, decoupling the iteration cost from the massive dimension m.
- **CUR Decomposition:** Contrary to expectations, Randomized CUR was not significantly faster than Normal CUR. This is a "negative result" that reveals an implementation constraint: both versions in our code rely on the exact SVD to compute leverage scores ($O(mn^2)$). To achieve true sub-linear scaling, future work must implement *approximate* leverage score computation using randomized sketching.

5.2 Reconstruction Quality: Completion vs. Selection

We observed a fundamental difference in how algorithms handle sparse data (MovieLens):

- **Matrix Completion (SVD / Rand HALS):** These algorithms produced *dense* reconstructions. By identifying global latent factors, they effectively "predicted" ratings for unobserved movies.
- **Feature Selection (CUR / Normal NMF):** These algorithms produced *sparse* reconstructions. CUR is restricted to selecting actual (sparse) columns, while Normal NMF overfitted to the zero entries. This suggests that for recommender systems, subspace-based methods (like Rand HALS) are superior to pure selection methods.

5.3 Trade-offs and Recommendations

- **Use Randomized SVD** for general-purpose compression of large, dense matrices where linear scalability is required.
- **Use Randomized HALS NMF** for sparse datasets (e.g., recommender systems) or when parts-based interpretability is needed on large scales.
- **Use CUR** only when "physical" interpretability (selecting actual rows/columns) is strictly required, acknowledging the potential computational cost of computing leverage scores accurately.

6 Conclusion

This project successfully implemented a suite of matrix factorization algorithms from scratch. We demonstrated that **Randomized SVD** provides linear scalability for dense data, and **Randomized HALS NMF** is the superior choice for large-scale or sparse feature extraction. The Shiny dashboard serves as an effective pedagogical tool to visualize these algorithmic trade-offs in real-time.

A Dashboard Implementation

The Shiny dashboard ("Matrix Approx Lab") allows interactive exploration with the following features:

- **Real Data Integration:** Automatic fetching of MovieLens data and integration of R's Volcano dataset.
- **Comparison Tab:** Side-by-side execution of Normal and Randomized algorithms.
- **Deep Dive Tab:** Detailed visualization of basis vectors, singular values, and error heatmaps.