Comparison of SVD Algorithms

Golub–Kahan Bidiagonalization vs Randomized SVD

Sharanya Palit

08/05/2025

1 Introduction

The Singular Value Decomposition (SVD) is a fundamental matrix factorization technique in numerical linear algebra. It expresses any real or complex matrix A as a product:

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

where U and V are orthogonal matrices and Σ is a diagonal matrix containing the singular values of A. SVD has applications in signal processing, data compression, machine learning, and scientific computing.

This project compares two important algorithms for computing the SVD of a matrix:

- **Golub–Kahan Bidiagonalization (Exact SVD)**
- **Randomized SVD (Approximate SVD)**

The goal is to analyze their performance based on speed, accuracy, and numerical stability, especially on large matrices.

2 Why We Care About SVD

SVD is widely used because:

- It provides a stable way to analyze matrices numerically.
- It reveals the rank and range of a matrix.
- It allows efficient low-rank approximations, useful for dimensionality reduction for really big matrices.

However, computing the full SVD is computationally expensive for large datasets. Therefore, approximate methods like Randomized SVD are gaining popularity for efficiency (not accuracy).

3 Algorithm 1: Golub–Kahan Bidiagonalization

This is a classic, accurate algorithm used in standard numerical libraries.

Step 1: Bidiagonalization

Matrix $A \in R^{m \times n}$ is reduced to an upper bidiagonal matrix B using Householder reflections:

$$A = U_1 B V_1^T$$

Here, B is bidiagonal and U_1, V_1 are orthogonal matrices.

Step 2: Diagonalization

Then a QR iteration is used to compute the singular values of B, yielding:

$$A = (U_1 U_2)\Sigma(V_1 V_2)^T$$

Pros:

- Highly accurate and numerically stable.
- Suitable for small to medium matrices.

Cons:

- Slow for large matrices.
- Computationally heavy when only a few singular values are needed.

4 Algorithm 2: Randomized SVD

Randomized SVD is designed for fast approximation, especially for large matrices with low-rank structure.

Step-by-step Process

1. Generate a random Gaussian matrix $\Omega \in R^{n \times (k+p)}$.
2. Compute $Y = A\Omega$, projecting A into a smaller subspace.
3. Compute an orthonormal basis Q using QR decomposition: $Y = QR$.
4. Project A: $B = Q^T A$.
5. Compute SVD of B: $B = \tilde{U}\Sigma V^T$.
6. Approximate SVD of A: $A \approx Q\tilde{U}\Sigma V^T$.

Pros:

- Much faster than traditional SVD.
- Lower memory usage.
- Ideal for large matrices.

Cons:

- Only provides approximate results.
- Sensitive to oversampling and number of iterations.

5 Experimental Comparison

Both methods were tested on a sample matrix using C++. Results:

- **Golub–Kahan (Exact SVD)**
  - Time: 123 ms
  - Reconstruction Error (Frobenius norm): 2.2×10^{-12}
- **Randomized SVD**
  - Time: 7 ms
  - Reconstruction Error: 46.04

Interpretation: While Golub–Kahan SVD is much more accurate, Randomized SVD is significantly faster, especially for large matrices. The trade-off is in approximation error.

6 Conclusion

Both SVD methods have clear strengths:

- Use **Golub–Kahan** when accuracy is critical.
- Use **Randomized SVD** for large-scale data where speed matters more than precision.

Understanding these algorithms and their trade-offs helps choose the right method for applications in data science, machine learning, and scientific

7 References

1. Golub, G. H., & Van Loan, C. F. (2013). *Matrix Computations* (4th ed.). Johns Hopkins University Press.

2. Halko, N., Martinsson, P. G., & Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2), 217–288.

3. Edelman, A., Arias, T. A., & Smith, S. T. (1998). The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2), 303–353.