Low-Rank Incremental Methods for Computing Dominant Singular Subspaces

Christopher G. Baker[1,2] Kyle A. Gallivan[1]
Paul Van Dooren[3]

[1]School of Computational Science
Florida State University

[2]Computer Science Research Institute
Sandia National Laboratories

[3]Department of Mathematical Engineering
Université catholique de Louvain

April 2008 / Copper Mountain

Acknowledgments

Funding

- NSF Grants ACI0324944 and CCR9912415
- School of Computational Science, FSU
- Baker was funded as a student at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy; contract/grant number: DE-AC04-94AL85000.

Outline

Outline

The Singular Value Decomposition

Definition

The singular value decomposition of an $m \times n$ matrix A is

$$A = U\Sigma V^T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma \\ 0 \end{bmatrix} V^T = U_1 \Sigma V^T$$

with orthogonal U, V; Σ diagonal with non-decreasing, non-negative entries.

Terminology

The columns of U_1 and V are left and right singular vectors. Diagonal entries of Σ are singular values. Largest singular values are dominant, as are their corresponding singular vectors.

Dominant SVD

Applications

Many applications require only the dominant singular triplets, e.g., PCA, KLT, POD.

Computation

Numerous approaches for computing the dominant SVD:

- compute the full SVD and truncate the unneeded part;
- transform to an eigenvalue problem, compute relevant eigenvectors via iterative eigensolver, back-transform;
- use iterative solver to compute dominant SVD:
  - Riemannian optimization gives many approaches [ABG2007]
  - Non-linear equation $\longrightarrow$ JD-SVD [Hochstenbach2000]
  - Low-rank incremental methods

Outline

Incremental SVD

Motivation

In many applications, the production of the matrix A happens incrementally. This has motivated numerous methods for SVD updating. [e.g., Businger; Bunch,Nielson]

Benefits

- Latency in producing new columns of A can be amortized in the SVD update
- "Online" SVD is useful/necessary in some applications

Drawbacks

- Computation, storage is expensive
- Still computes the full SVD

Low-Rank Incremental SVD

More efficient approach

The low-rank incremental SVD methods follow the example of the incremental SVD methods, but track only a low-dimensional subspace.

History

Repeatedly and independently described in the literature:

- 1995: Manjunath, Chandrasekaran, Yang: “Eigenspace Update Algorithm”
- 2000: Levy, Lindenbaum: “Sequential Karhunen-Loeve”
- 2001: Chahlaoui, Gallivan, Van Dooren: “Recursive SVD”
- 2002: Brand: “Incremental SVD”
- 2004: Baker, Gallivan, Van Dooren

Full-Rank Incremental SVD Operation

Kernel Step

Given a matrix A with factorization $A = U\Sigma V^T$, compute updated factorization of augmented matrix $\begin{bmatrix} A & A_+ \end{bmatrix}$:

$$U_+\Sigma_+V_+^T = \begin{bmatrix} A & A_+ \end{bmatrix} = \begin{bmatrix} U\Sigma V^T & A_+ \end{bmatrix}$$

Incremental Algorithm

- Partition $A = \begin{bmatrix} A_1 & A_2 & \dots & A_b \end{bmatrix}$
- Initialize $A_1 = U_1\Sigma_1V_1^T$
- for $i = 2, \dots, b$
  - Update factorization:

$$U_i\Sigma_iV_i^T = \begin{bmatrix} U_{i-1}\Sigma_{i-1}V_{i-1}^T & A_i \end{bmatrix}$$

Low-rank Incremental SVD Operation

- Perform a low-rank version of the incremental SVD

Kernel Step

Given a factorization $U\Sigma V^T$ and columns A_+, compute dominant SVD $U_+\Sigma_+V_+^T \approx \begin{bmatrix} U\Sigma V^T & A_+ \end{bmatrix}$.

Heuristic motivation

Approximation of an approximation is an approximation, right?

$$
\begin{aligned}
U_1\Sigma_1V_1^T &\approx A_1 \\
U_2\Sigma_2V_2^T &\approx \begin{bmatrix} U_1\Sigma_1V_1^T & A_2 \end{bmatrix} \approx\approx \begin{bmatrix} A_1 & A_2 \end{bmatrix} \\
&\dots \\
U_b\Sigma_bV_b^T &\approx\approx A
\end{aligned}
$$

Low-rank Incremental SVD Operation

The algorithm

Given the factorization $U\Sigma V^T$ and new columns A:

1. Expand the factorization via Gram-Schmidt:

$$\begin{bmatrix} U\Sigma V^T & A \end{bmatrix} = \hat{Q}\hat{R}\hat{W}^T \doteq \begin{bmatrix} U & Q \end{bmatrix} \begin{bmatrix} \Sigma & R_2 \\ 0 & R_3 \end{bmatrix} \begin{bmatrix} V & 0 \\ 0 & I \end{bmatrix}^T$$

2. Compute transformations G_u, G_v that decouple the singular subspaces in $\hat{R}$:

$$G_u^T \hat{R} G_v = \begin{bmatrix} \bar{R}_1 & 0 \\ 0 & \bar{R}_2 \end{bmatrix}, \qquad \sigma(\bar{R}_1) > \sigma(\bar{R}_2)$$

3. Insert G_u, G_v into expanded factorization:

$$\bar{Q}\bar{R}\bar{W}^T \doteq (\hat{Q}G_u)(G_u^T \hat{R} G_v)(G_v^T \hat{W}^T) = \hat{Q}\hat{R}\hat{W}^T$$

4. Truncate the dominated part of the factorization.

Algorithm features

Cost/benefit

Benefits:

- Requires only a single pass through A
- Exploits latency in producing/retrieving columns of A
- Flop count is linear: $O(mnk)$
  - Leading coefficient varies according to requirements on structure of intermediate factorizations
  - Method from [Baker2004] requires $10mnk$ flops
- Storage of $O(mk + nk)$ is minimal

Drawback:

- Factorization is inexact due to truncation
- Previous literature makes no suggestion for improving factorization

Outline

What is really happening?

New interpretation

Take an orthogonal matrix $D = \begin{bmatrix} D_1 & \dots & D_b \end{bmatrix}$. Consider the low-rank incremental SVD of $AD = \begin{bmatrix} AD_1 & \dots & AD_b \end{bmatrix}$.

A locally optimal solver

At iterate U_i, Σ_i, V_i, the algorithm inputs AD_{i+1} and chooses V_{i+1} which maximizes $\mathit{trace}\left(V^T A^T A V\right)$ over all orthonormal V in $\mathrm{span}(\begin{bmatrix} V_i & D_{i+1} \end{bmatrix})$.

Implications

- IncSVD of A (i.e., $D = I$) implicitly performs coordinate ascent, optimization-based eigensolve of $A^T A$
- Choice of D gives a hook to affect the performance.

Multi-pass Method

Targeted initialization

- Given approximate right singular vectors $\hat{V}$, choose D:
$$D = \begin{bmatrix} \hat{V} & D_2 & \dots & D_b \end{bmatrix}$$
- The iteration immediately captures the information in $\hat{V}$ and improves thereafter.
- Use to restart the algorithm if A is still available.
- Can be done in a pass-efficient manner.

Better choices for D?

- If D_i is exact dominant right singular vectors, then incremental algorithm is exact.
- Speedup convergence by inserting gradient information into D.

Summary

- The decoupling technique makes explicit the effort necessary to implement a member of this family of methods.
- The novel analysis shows the link to an iterative, optimization-based eigensolver approach.
- This analysis allows the description of methods which can exploit multiple passes through A.
- Convergence proof with rate of convergence is forthcoming.
- “Killer apps” wanted.