GSVD in Julia 1.0

Ji Wang

1 Definition

1.1 Our definition

The generalized singular value decomposition (GSVD) of an m-by-n matrix A and p-by-n matrix B is the following, where $m + p \geq n$:

$$A = UCRQ^T \quad \text{and} \quad B = VSRQ^T$$

where U, V and Q are orthogonal matrices. Let $k + l$ be the effective numerical rank of the matrix $\begin{pmatrix} A \\ B \end{pmatrix}$, then R is a $(k + l)$-by-n matrix of structure $[0\ R_0]$ where R_0 is $(k + l)$-by-$(k + l)$ and is nonsingular upper triangular matrix, C and S are m-by-$(k + l)$ and p-by-$(k + l)$ non-negative "diagonal" matrices and satisfy $C^TC + S^TS = I$. The nonzero elements of C are in **non-increasing** order while the nonzero elements of S are in **non-decreasing** order.

1.2 Other notable definition of GSVD

1.2.1 Julia 1.0

Compute the generalized SVD of A and B, returning a GeneralizedSVD factorization object F, such that $A = F.U * F.D_1 * F.R_0 * F.Q^T$ and $B = F.V * F.D_2 * F.R_0 * F.Q^T$.

The entries of $F.D1$ and $F.D2$ are related, as explained in the LAPACK documentation for the generalized SVD and the xGGSVD3 routine which is called underneath (in LAPACK 3.6.0 and newer).

1.2.2 LAPACK 3.6.0

The generalized (or quotient) singular value decomposition of an m-by-n matrix A and a p-by-n matrix B is given by the pair of factorizations:

$$A = U\Sigma_1[0, R]Q^T \quad \text{and} \quad B = V\Sigma_2[0, R]Q^T$$

Σ_1 is m-by-r, Σ_2 is p-by-r. (The integer r is the rank of $\begin{pmatrix} A \\ B \end{pmatrix}$, and satisfies $r \leq n$.) Both are real, non-negative and diagonal, and $\Sigma_1^T\Sigma_1 + \Sigma_2^T\Sigma_2 = I$. Write $\Sigma_1^T\Sigma_1 = diag(\alpha_1^2, ..., \alpha_r^2)$ and $\Sigma_2^T\Sigma_2 = diag(\beta_1^2, ..., \beta_r^2)$, where α_i and β_i lie in the interval from 0 to 1. The ratios α_1/β_1,...,α_r/β_r are called the generalized singular values of the pair A, B. If $\beta_i = 0$, then the generalized singular value α_i/β_i is infinite.

1.2.3 MATLAB 2019b

$[U, V, X, C, S] = gsvd(A, B)$ returns unitary matrices U and V, a (usually) square matrix X, and non-negative diagonal matrices C and S so that

$$A = U * C * X'$$
$$B = V * S * X'$$
$$C' * C + S' * S = I$$

A and B must have the same number of columns, but may have different numbers of rows. If A is m-by-p and B is n-by-p, then U is m-by-m, V is n-by-n, X is p-by-q, C is m-by-q and S is n-by-q, where $q = min(m + n, p)$.

The nonzero elements of S are always on its main diagonal. The nonzero elements of C are on the diagonal $diag(C, max(0, q - m))$. If $m \geq q$, this is the main diagonal of C.

$sigma = gsvd(A, B)$ returns the vector of generalized singular values, $sqrt(diag(C'*C)./diag(S'*S))$. The vector sigma has length q and is in **non-decreasing** order, where **q = min(m+n,p)**. In other words, The nonzero elements of S are in **non-increasing** order while the nonzero elements of C are in **non-decreasing** order.

It's interesting to notice that MATLAB has a compact form of products for gsvd.

2 Algorithm

We present our algorithm based on two major decompositions.

2.1 GSVD Pre-processing Routine

Given that $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times n}$, $\text{rank}(B) = l$ and $\text{rank}([A; B]) = k + l = r$, here's a detailed derivation of pre-processing procedure.

Step 1: QR decomposition with column pivoting of B, and determine the effective rank of $B = l$.

$$BP_1 = Q_1 \cdot \begin{array}{c} \\ l \\ p-l \end{array} \begin{array}{c} \begin{array}{cc} l & n-l \end{array} \\ \begin{bmatrix} B_{11}^{(1)} & B_{12}^{(1)} \\ 0 & 0 \end{bmatrix} \end{array}$$

Step 2: RQ decomposition of $\begin{bmatrix} B_{11}^{(1)} & B_{12}^{(1)} \end{bmatrix}$ when $p \geq l$ and $n \neq l$.

$$BP_1 = Q_1 \cdot \begin{array}{c} \\ l \\ p-l \end{array} \begin{array}{c} \begin{array}{cc} n-l & l \end{array} \\ \begin{bmatrix} 0 & B_{12}^{(2)} \\ 0 & 0 \end{bmatrix} \end{array} \cdot Q_2$$

$$BP_1Q_2^T = Q_1 \cdot \begin{array}{c} \\ l \\ p-l \end{array} \begin{array}{c} \begin{array}{cc} n-l & l \end{array} \\ \begin{bmatrix} 0 & B_{12}^{(2)} \\ 0 & 0 \end{bmatrix} \end{array}$$

$$Q_1^TBP_1Q_2^T = \begin{array}{c} \\ l \\ p-l \end{array} \begin{array}{c} \begin{array}{cc} n-l & l \end{array} \\ \begin{bmatrix} 0 & B_{12}^{(2)} \\ 0 & 0 \end{bmatrix} \end{array}$$

Step 3: View $AP_1Q_2^T$ as block matrix and apply QR with column pivoting of $A_{11}^{(1)}$ when $A_{11}^{(1)}$ is not empty, and determine the effective rank of $A_{11}^{(1)} = k$.

$$AP_1Q_2^T = \begin{array}{c} \\ m \end{array} \begin{array}{c} \begin{array}{cc} n-l & l \end{array} \\ \begin{bmatrix} A_{11}^{(1)} & A_{12}^{(1)} \end{bmatrix} \end{array}$$

$$A_{11}^{(1)}P_2 = Q_3 \cdot \begin{array}{c} \\ k \\ m-k \end{array} \begin{array}{c} \begin{array}{cc} k & n-l-k \end{array} \\ \begin{bmatrix} A_{11}^{(2)} & A_{12}^{(2)} \\ 0 & 0 \end{bmatrix} \end{array}$$

Step 4: RQ decomposition of $\begin{bmatrix} A_{11}^{(2)} & A_{12}^{(2)} \end{bmatrix}$ when $n - l \geq k$.

$$A_{11}^{(1)}P_2 = Q_3 \cdot \begin{array}{c} \\ k \\ m-k \end{array} \begin{array}{c} \begin{array}{cc} n-l-k & k \end{array} \\ \begin{bmatrix} 0 & A_{12}^{(3)} \\ 0 & 0 \end{bmatrix} \end{array} \cdot Q_4$$

View $A_{12}^{(1)}$ as block matrix, we have:

$$Q_3^T A P_1 Q_2^T P_2 Q_4^T = \begin{array}{c} \\ k \\ m-k \end{array} \begin{array}{c} \begin{array}{ccc} n-l-k & k & l \end{array} \\ \begin{bmatrix} 0 & A_{12}^{(3)} & A_{13}^{(3)} \\ 0 & 0 & A_{23}^{(3)} \end{bmatrix} \end{array}$$

$$Q_1^T B P_1 Q_2^T P_2 Q_4^T = \begin{array}{c} \\ l \\ p-l \end{array} \begin{array}{c} \begin{array}{cc} n-l & l \end{array} \\ \begin{bmatrix} 0 & B_{12}^{(2)} \\ 0 & 0 \end{bmatrix} \end{array}$$

Step 5: QR decomposition of $A_{23}^{(3)}$ when $m \geq k$.

$$A_{23}^{(3)} = Q_5 \cdot \tilde{A}_{23}^{(3)}$$

$$Q_5^T Q_3^T A P_1 Q_2^T P_2 Q_4^T = \begin{array}{c} \\ k \\ m-k \end{array} \begin{array}{c} \begin{array}{ccc} n-l-k & k & l \end{array} \\ \begin{bmatrix} 0 & A_{12}^{(3)} & A_{13}^{(3)} \\ 0 & 0 & \tilde{A}_{23}^{(3)} \end{bmatrix} \end{array}$$

Let $U = Q_3 Q_5$, $Q = P_1 Q_2^T P_2 Q_4^T$, and $V = Q_1$, we have:

$$\begin{array}{c} \\ m \\ p \end{array} \begin{array}{c} \begin{array}{c} n \end{array} \\ \left[\begin{array}{c} U^T A \\ \hline V^T B \end{array}\right] \end{array} Q = \begin{array}{c} \\ k \\ m-k \\ \\ l \\ p-l \end{array} \begin{array}{c} \begin{array}{ccc} n-l-k & k & l \end{array} \\ \left[\begin{array}{ccc} 0 & A_{12}^{(3)} & A_{13}^{(3)} \\ 0 & 0 & \tilde{A}_{23}^{(3)} \\ \hline 0 & 0 & B_{12}^{(2)} \\ 0 & 0 & 0 \end{array}\right] \end{array}$$

2.2 CS Decomposition Routine

To be completed...

3 Implementation

3.1 2 APIs

We provide two APIs: LAPACK-style one simply returns singular value pairs, namely *alpha* and *beta*; Julia-style follows the definition in current native Julia version and explicitly formulates the diagonal matrices C and S.

1. LAPACK-style: $U, V, Q, alpha, beta, R, k, l = gsvd(A, B, 0)$

2. Julia-style: $U, V, Q, C, S, R, k, l = gsvd(A, B, 1)$

3.2 Building Block and Call Graph

1. Pre-processing:

(1). QR decomposition with column pivoting

(2). RQ decomposition

(3). QR decomposition

2. CSD:

(1). SVD of a single matrix

(2). QR decomposition

(3). QL decomposition

Call graph to be completed...

4 Testing

Currently, testing is done on the tall rectangular case where both A and B have more rows than columns, namely $m, p \geq n$.

In the test routine, we specify the ratios of $m : n : p$.

In the general tests, we create two random matrices as input, and then call $gsvd$ routine. We compute stability benchmarks defined in Performance Section and CPU elapsed time.

Special cases are $\begin{pmatrix} A \\ B \end{pmatrix}$ is full rank. In other words, $k + l = n$. The expected results should be that matrix R no longer has 0 columns in the left most.

Here's an example:

$$A = \begin{bmatrix} -0.124913 & 0.0149558 & 0.114572 \\ 1.7037 & -0.203984 & -1.56267 \\ 1.32727 & -0.158914 & -1.2174 \\ 0.355183 & -0.042526 & -0.32578 \\ -0.804623 & 0.0963374 & 0.738015 \end{bmatrix}$$

$$B = \begin{bmatrix} -0.891686 & -0.347289 & 1.60907 \\ 0.146657 & 0.860828 & -1.08768 \\ -1.20119 & 4.50078 & -2.92048 \\ 0.939956 & -1.65502 & 0.373526 \end{bmatrix}$$

Computed R is:

$$\begin{bmatrix} 0.425876 & -3.13062 & 0.370613 \\ 0.0 & 1.91911 & -0.598304 \\ 0.0 & -2.22045e-16 & 5.94985 \end{bmatrix}$$

5 Performance

5.1 Stability

The following metrics are computed in each stability tests:

$$res_A = \frac{\|U^T AQ - CR\|_1}{max(m,n)\|A\|_1 \epsilon}$$

$$res_b = \frac{\|V^T BQ - SR\|_1}{max(p,n)\|B\|_1 \epsilon}$$

$$orth_U = \frac{\|I - U^T U\|_1}{m\epsilon}$$

$$orth_V = \frac{\|I - V^T V\|_1}{p\epsilon}$$

$$orth_Q = \frac{\|I - Q^T Q\|_1}{n\epsilon}$$

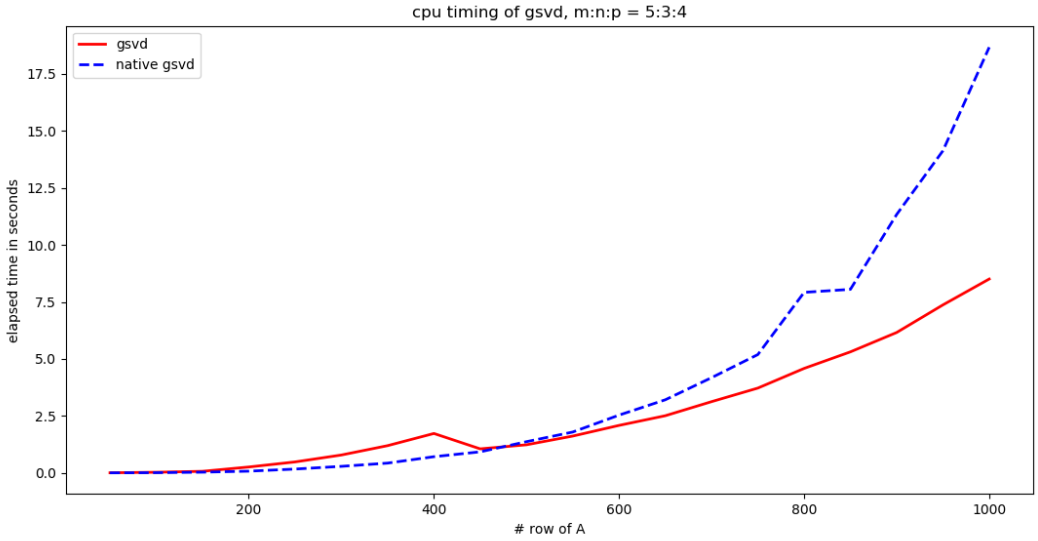
where ϵ is machine precision of input data type.

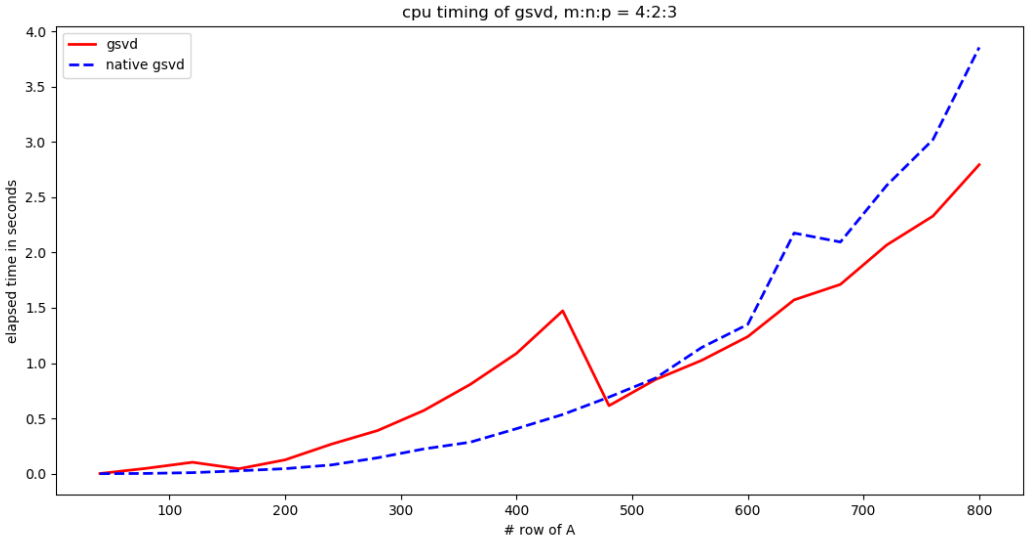
5.2 Timing

1. Test case 1: m:n:p = 5:3:4
2. Test case 2: m:n:p = 4:2:3

6 Novelty and Challenges

To be completed...


cpu timing of gsvd, m:n:p = 5:3:4
gsvd
native gsvd
elapsed time in seconds
row of A
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
200
400
600
800
1000


cpu timing of gsvd, m:n:p = 4:2:3
gsvd
native gsvd
elapsed time in seconds
row of A
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
100
200
300
400
500
600
700
800