fullsvd( ) — Full singular value decomposition

Description   Syntax   Remarks and examples   Conformability   Diagnostics   Also see

Description

`fullsvd(`A`, `U`, `s`, `Vt`)` calculates the singular value decomposition of $m \times n$ matrix A, returning the result in U, s, and Vt. Singular values in s are sorted from largest to smallest.

`fullsdiag(`s`, `k`)` converts column vector s returned by `fullsvd()` into matrix S. In all cases, the appropriate call for this function is

`S = fullsdiag(s, rows(A)-cols(A))`

`_fullsvd(`A`, `U`, `s`, `Vt`)` does the same as `fullsvd()`, except that, in the process, it destroys A. Use of `_fullsvd()` in place of `fullsvd()` conserves memory.

`_svd_la()` is the interface to the LAPACK SVD routines and is used in the implementation of the previous functions. There is no reason you should want to use it. `_svd_la()` is similar to `_fullsvd()`. It differs in that it returns a real scalar equal to 1 if the numerical routines fail to converge, and it returns 0 otherwise. The previous SVD routines set s to contain missing values in this unlikely case.

Syntax

void	`fullsvd(`*numeric matrix A*`, `*U*`, `*s*`, `*Vt*`)`
numeric matrix	`fullsdiag(`*numeric colvector s*`, `*real scalar k*`)`
void	`_fullsvd(`*numeric matrix A*`, `*U*`, `*s*`, `*Vt*`)`
real scalar	`_svd_la(`*numeric matrix A*`, `*U*`, `*s*`, `*Vt*`)`

Remarks and examples

Remarks are presented under the following headings:

Documented here is the full SVD, appropriate in all cases, but of interest mainly when A: $m \times n$, $m < n$. There is a thin SVD that conserves memory when $m \geq n$; see [M-5] **svd()**. The relationship between the two is discussed in *Relationship between the full and thin SVDs* below.

Introduction

The SVD is used to compute accurate solutions to linear systems and least-squares problems, to compute the 2-norm, and to determine the numerical rank of a matrix.

The singular value decomposition (SVD) of A: $m \times n$ is given by

$$A = USV'$$

where

U:	$m \times m$ and orthogonal (unitary)
S:	$m \times n$ and diagonal
V:	$n \times n$ and orthogonal (unitary)

When A is complex, the transpose operator $'$ is understood to mean the conjugate transpose operator.

Diagonal matrix S contains the singular values and those singular values are real even when A is complex. It is usual (but not required) that S is arranged so that the largest singular value appears first, then the next largest, and so on. The SVD routines documented here do this.

The full SVD routines return U and $Vt = V'$. S is returned as a column vector s, and S can be obtained by

```
S = fullsdiag(s, rows(A)-cols(A))
```

so we will write the SVD as

```
A = U * fullsdiag(s, rows(A)-cols(A)) * Vt
```

Function `fullsvd(A, U, s, Vt)` returns the U, s, and Vt corresponding to A.

Relationship between the full and thin SVDs

A popular variant of the SVD is known as the thin SVD and is suitable for use when $m \geq n$. Both SVDs have the same formula,

$$A = USV'$$

but U and S have reduced dimensions in the thin version:

Matrix	Full SVD	Thin SVD
U:	$m \times m$	$m \times n$
S:	$m \times n$	$n \times n$
V:	$n \times n$	$n \times n$

When $m = n$, the two variants are identical.

The thin SVD is of use when $m > n$, because then only the first n diagonal elements of S are nonzero, and therefore only the first n columns of U are relevant in $A = USV'$. There are considerable memory savings to be had in calculating the thin SVD when $m \gg n$.

As a result, many people call the thin SVD the SVD and ignore the full SVD altogether. If the matrices you deal with have $m \geq n$, you will want to do the same. To obtain the thin SVD, see [M-5] **svd( )**.

Regardless of the dimension of your matrix, you may wish to obtain only the singular values. In this case, see `svdsv()` documented in [M-5] **svd()**. That function is appropriate in all cases.

The contents of s

Given A: $m \times n$, the singular values are returned in s: $\min(m, n) \times 1$.

Let's consider the $m = n$ case first. A is $m \times m$ and the m singular values are returned in s, an $m \times 1$ column vector. If A were 3×3, perhaps we would get back

```
: s
             1
     +---------+
  1  |  13.47  |
  2  |    5.8  |
  3  |   2.63  |
     +---------+
```

If we needed it, we could obtain S from s simply by creating a diagonal matrix from s

```
: S = diag(s)

: S
[symmetric]
             1        2        3
     +--------------------------+
  1  |  13.47                   |
  2  |      0      5.8          |
  3  |      0        0     2.63 |
     +--------------------------+
```

although the official way we are supposed to do this is

```
: S = fullsdiag(s, rows(A)-cols(A))
```

and that will return the same result.

Now let's consider $m < n$. Let's pretend that A is 3×4. The singular values will be returned in 3×1 vector s. For instance, s might still contain

```
: s
             1
     +---------+
  1  |  13.47  |
  2  |    5.8  |
  3  |   2.63  |
     +---------+
```

The S matrix here needs to be 3×4, and `fullsdiag()` will form it:

```
: fullsdiag(s, rows(A)-cols(A))
             1        2        3        4
     +-----------------------------------+
  1  |  13.47        0        0        0 |
  2  |      0      5.8        0        0 |
  3  |      0        0     2.63        0 |
     +-----------------------------------+
```

The final case is $m > n$. We will pretend that A is 4×3. The s vector we get back will look the same

```
: s
           1
    ┌─────────┐
  1 │  13.47  │
  2 │    5.8  │
  3 │   2.63  │
    └─────────┘
```

but this time, we need a 4×3 rather than a 3×4 matrix formed from it.

```
: fullsdiag(s, rows(A)-cols(A))
           1       2       3
    ┌─────────────────────────┐
  1 │  13.47       0       0  │
  2 │      0     5.8       0  │
  3 │      0       0    2.63  │
  4 │      0       0       0  │
    └─────────────────────────┘
```

Possibility of convergence problems

See *Possibility of convergence problems* in [M-5] **svd( )**; what is said there applies equally here.

Conformability

fullsvd(A, U, s, Vt):

input:

A: $m \times n$

output:

U: $m \times m$

s: $\min(m, n) \times 1$

Vt: $n \times n$

result: *void*

fullsdiag(s, k):

input:

s: $r \times 1$

k: 1×1

output:

result: $r + k \times r$, if k ≥ 0

$r \times r - k$, otherwise

_fullsvd(A, U, s, Vt):

input:

A: $m \times n$

output:

A: 0×0

U: $m \times m$

s: $\min(m, n) \times 1$

Vt: $n \times n$

result: *void*

`_svd_la(`A`,` U`,` s`,` Vt`)`:

input:

A:	$m \times n$	

output:

A:	$m \times n$,	but contents changed
U:	$m \times m$	
s:	$\min(m, n) \times 1$	
Vt:	$n \times n$	
result:	1×1	

Diagnostics

`fullsvd(`A`,` U`,` s`,` Vt`)` and `_fullsvd(`A`,` s`,` Vt`)` return missing results if A contains missing. In all other cases, the routines should work, but there is the unlikely possibility of convergence problems, in which case missing results will also be returned; see *Possibility of convergence problems* in [M-5] **svd( )**.

`_fullsvd()` aborts with error if A is a view.

Direct use of `_svd_la()` is not recommended.

Also see

[M-5] **norm( )** — Matrix and vector norms

[M-5] **pinv( )** — Moore–Penrose pseudoinverse

[M-5] **rank( )** — Rank of matrix

[M-5] **svd( )** — Singular value decomposition

[M-5] **svsolve( )** — Solve AX=B for X using singular value decomposition

[M-4] **Matrix** — Matrix functions

For suggested citations, see the FAQ on citing Stata documentation.