L5a Singular Value Decomposition - Definitions and Facts

April 19, 2018

1 Singular Value Decomposition - Definitions and Facts

1.1 Prerequisites

The reader should be familiar with basic linear algebra concepts and notebooks related to eigenvalue decomposition.

1.2 Competences

The reader should be able to undestand and check the facts about singular value decomposition.

1.3 Selected references

There are many excellent books on the subject. Here we list a few:
Section **??**
Section **??**
Section **??**
Section **??**
Section **??**
Section **??**

1.4 Singular value problems

For more details and the proofs of the Facts below, see Section **??** and Section **??** and the references therein.

1.4.1 Definitions

Let $A \in \mathbb{C}^{m\times n}$ and let $q = \min\{m, n\}$.

The **singular value decomposition** (SVD) of A is

$$A = U\Sigma V^*,$$

where $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary, and $\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots) \in \mathbb{R}^{m\times n}$ with all $\sigma_j \geq 0$.

Here σ_j is the **singular value**, $u_j \equiv U_{:,j}$ is the corresponding **left singular vector**, and $v_j \equiv V_{:,j}$ is the corresponding **right singular vector**.

The **set of singular values** is $sv(A) = \{\sigma_1, \sigma_2, \ldots, \sigma_q\}$.

We assume that singular values are ordered, $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_q \geq 0$.

The **Jordan-Wielandt** matrix is the Hermitian matrix

$$J = \begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix} \in \mathbb{C}^{(m+n)\times(m+n)}.$$

1.4.2 Facts

There are many facts related to the singular value problem for general matrices. We state some basic ones:

1. If $A \in \mathbb{R}^{m\times n}$, then U and V are real.
2. Singular values are unique (uniquely determined by the matrix).
3. $\sigma_j(A^T) = \sigma_j(A^*) = \sigma_j(\bar{A}) = \sigma_j(A)$ for $j = 1, 2, \ldots, q$.
4. $Av_j = \sigma_j u_j$ and $A^* u_j = \sigma_j v_j$ for $j = 1, 2, \ldots, q$.
5. $A = \sigma_1 u_1 v_1^* + \sigma_2 u_2 v_2^* + \cdots + \sigma_q u_q v_q^*$.
6. **Unitary invariance.** For any unitary $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$, $sv(A) = sv(UAV)$.
7. There exist unitary matrices $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ such that $A = UBV$ if and only if $sv(A) = sv(B)$.
8. SVD of A is related to eigenvalue decompositions of Hermitian matrices $A^*A = V\Sigma^T\Sigma V^*$ and $AA^* = U\Sigma\Sigma^T U^*$. Thus, $\sigma_j^2(A) = \lambda_j(A^*A) = \lambda_j(AA^*)$ for $j = 1, 2, \ldots, q$.
9. The eigenvalues of Jordan-Wielandt matrix are $\pm\sigma_1(A), \pm\sigma_2(A), \cdots, \pm\sigma_q(A)$ together with $|m-n|$ zeros. The eigenvectors are obtained from an SVD of A. This relationship is used to deduce singular value results from the results for eigenvalues of Hermitian matrices.
10. $\mathrm{trace}(|A|_{spr}) = \sum_{i=1}^{q} \sigma_i$, where $|A|_{spr} = (A^*A)^{1/2}$.
11. If A is square, then $|\det(A)| = \prod_{i=1}^{n} \sigma_i$.
12. If A is square, then A is singular $\Leftrightarrow \sigma_j(A) = 0$ for some j.
13. **Min-max Theorem.** It holds:

$$\begin{aligned} \sigma_k &= \max_{\dim(W)=k} \min_{x\in W, \|x\|_2=1} \|Ax\|_2 \\ &= \min_{\dim(W)=n-k+1} \max_{x\in W, \|x\|_2=1} \|Ax\|_2. \end{aligned}$$

14. $\|A\|_2 = \sigma_1(A)$.
15. For $B \in \mathbb{C}^{m\times n}$,

$$|\,\mathrm{trace}(AB^*)| \leq \sum_{j=1}^{q} \sigma_j(A)\sigma_j(B).$$

16. **Interlace Theorems.** Let B denote A with the one of its rows *or* columns deleted. Then

$$\sigma_{j+1}(A) \leq \sigma_j(B) \leq \sigma_j(A), \quad j=1,\ldots,q-1.$$

Let B denote A with the one of its rows *and* columns deleted. Then

$$\sigma_{j+2}(A) \leq \sigma_j(B) \leq \sigma_j(A), \quad j=1,\ldots,q-2.$$

17. **Weyl Inequalities.** For $B \in \mathbb{C}^{m\times n}$, it holds:

$$\sigma_{j+k-1}(A+B) \leq \sigma_j(A) + \sigma_k(B), \quad j+k \leq n+1,$$
$$\sum_{j=1}^{k} \sigma_j(A+B) \leq \sum_{j=1}^{k} \sigma_j(A) + \sum_{j=1}^{k} \sigma_j(A), \quad k=1,\ldots,q.$$

1.4.3 Example - Symbolic computation

```
In [1]: using SymPy
```

```
In [2]: A=[ 3   2   1
          -5  -1  -4
           5   0   2]
```

```
Out[2]: 3×3 Array{Int64,2}:
          3   2   1
         -5  -1  -4
          5   0   2
```

```
In [3]: @vars x
```

```
Out[3]: (x,)
```

```
In [4]: B=A'*A
```

```
Out[4]: 3×3 Array{Int64,2}:
         59  11  33
         11   5   6
         33   6  21
```

```
In [5]: # Characteristic polynomial p_B()
        p(x)=simplify(det(B-x*I))
        p(x)
```

Out[5]:

$$-x^3 + 85x^2 - 393x + 441$$

```
In [6]: =map(Rational,solve(p(x),x))
```

```
Out[6]: 3-element Array{Rational{Int64},1}:
                       3//1
         2064549086305011//1125899906842624
         5641202704674385//70368744177664

In [7]: V=Array{Any}(3,3)
        for j=1:3
            V[:,j]=nullspace(B-[j]*I)
        end
        V

Out[7]: 3Œ3 Array{Any,2}:
         -3.2754f-7  -0.519818  -0.854277
          0.948684    0.270146  -0.164381
         -0.316227    0.810438  -0.493142

In [8]: U=Array{Any}(3,3)
        for j=1:3
            U[:,j]=nullspace(A*A'-[j]*I)
        end
        U

Out[8]: 3Œ3 Array{Any,2}:
          0.912871  -0.154138  -0.378032
          0.182574  -0.67409    0.71573
         -0.365148  -0.722388  -0.587215

In [9]: =sqrt.()

Out[9]: 3-element Array{Float64,1}:
         1.73205
         1.35414
         8.95356

In [10]: A-U*diagm()*V'

Out[10]: 3Œ3 Array{Float64,2}:
           2.02284e-7   3.05213e-7   9.51424e-7
           9.53544e-7  -6.66283e-7  -7.23387e-7
          -7.64795e-7   1.85427e-8   3.64772e-7

In [11]: S=svd(A)

Out[11]: ([-0.378032 -0.912871 -0.154137; 0.71573 -0.182574 -0.67409; -0.587215 0.365148 -0.7223

In [12]: typeof(S)

Out[12]: Tuple{Array{Float64,2},Array{Float64,1},Array{Float64,2}}
```

```
In [13]: U=S[1]
         =S[2]
         V=S[3]

Out[13]: 3Œ3 Array{Float64,2}:
          -0.854277   0.0       -0.519818
          -0.164381  -0.948683   0.270146
          -0.493143   0.316228   0.810438

In [14]: V

Out[14]: 3Œ3 Array{Any,2}:
          -3.2754f-7  -0.519818  -0.854277
           0.948684    0.270146  -0.164381
          -0.316227    0.810438  -0.493142
```

1.4.4 Example - Random complex matrix

```
In [15]: m=5
         n=3
         s=srand(421)
         q=min(m,n)
         A=rand(m,n)+im*rand(m,n)

Out[15]: 5Œ3 Array{Complex{Float64},2}:
          0.345443+0.915812im   0.77247+0.198694im   0.958365+0.37833im
           0.68487+0.605095im   0.17008+0.854638im   0.560486+0.834811im
          0.650991+0.83639im   0.525208+0.905889im   0.608612+0.353274im
          0.973053+0.766264im  0.785847+0.0936446im  0.346561+0.831302im
          0.105135+0.810683im  0.135538+0.651562im   0.561248+0.217897im

In [16]: U,,V=svd(A, thin=false)
         U

Out[16]: 5Œ5 Array{Complex{Float64},2}:
          -0.326131-0.323713im  -0.214728+0.0378561im     -0.304393-0.109676im
          -0.204849-0.411755im   0.124224-0.410762im       -0.123119-0.163377im
          -0.277774-0.399041im  -0.235078+0.0866662im      -0.181124+0.460827im
          -0.342932-0.328115im   0.718399+0.105642im        0.147071+0.0774263im
          -0.109764-0.321934im   -0.42215-0.00780174im      0.736473-0.195654im

In [17]:

Out[17]: 3-element Array{Float64,1}:
          3.30202
          0.965571
          0.76542

In [18]: V
```

```
Out[18]: 3Œ3 Array{Complex{Float64},2}:
          -0.657411-0.0im         0.461649-0.0im       -0.595559-0.0im
          -0.525502-0.0714762im  -0.021837+0.45993im    0.563152+0.435416im
          -0.523002-0.114098im   -0.496126-0.573347im   0.192746-0.318484im

In [19]: norm(A-U[:,1:q]*diagm()*V'), norm(U'*U-I), norm(V'*V-I)

Out[19]: (2.2933010552997958e-15, 7.04377263940463e-16, 8.629134923109423e-16)

In [20]: # Fact 4
         @show k=rand(1:q)
         norm(A*V[:,k]-[k]*U[:,k],Inf), norm(A'*U[:,k]-[k]*V[:,k],Inf)

k = rand(1:q) = 1


Out[20]: (1.2947314098277873e-15, 9.305364597889227e-16)

In [21]: ,V=eig(A'*A)

Out[21]: ([0.585867, 0.932328, 10.9034], Complex{Float64}[-0.308357-0.509516im -0.302079+0.34909

In [22]: sqrt.()

Out[22]: 3-element Array{Float64,1}:
          0.76542
          0.965571
          3.30202

In [23]: U,U=eig(A*A')

Out[23]: ([1.66533e-16, 3.61372e-16, 0.585867, 0.932328, 10.9034], Complex{Float64}[-0.257033-0.

In [24]: V

Out[24]: 3Œ3 Array{Complex{Float64},2}:
          -0.657411-0.0im         0.461649-0.0im       -0.595559-0.0im
          -0.525502-0.0714762im  -0.021837+0.45993im    0.563152+0.435416im
          -0.523002-0.114098im   -0.496126-0.573347im   0.192746-0.318484im

In [25]: V

Out[25]: 3Œ3 Array{Complex{Float64},2}:
           -0.308357-0.509516im  -0.302079+0.349097im  -0.642304+0.140124im
          -0.0809315+0.707232im  -0.333508-0.317467im  -0.528661+0.0421748im
            0.372268-0.0im          0.7582+0.0im       -0.535303-0.0im

In [26]: abs.(V'*V)

Out[26]: 3Œ3 Array{Float64,2}:
          2.65135e-16  3.19189e-16  1.0
          1.73089e-15  1.0          3.92523e-16
          1.0          2.02635e-15  4.8473e-16
```

Explain non-uniqueness of U and V!

```
In [27]: # Jordan-Wielandt matrix
         J=[zeros(A*A') A; A' zeros(A'*A)]

Out[27]: 8×8 Array{Complex{Float64},2}:
              0.0+0.0im            0.0+0.0im         0.958365+0.37833im
              0.0+0.0im            0.0+0.0im          0.560486+0.834811im
              0.0+0.0im            0.0+0.0im          0.608612+0.353274im
              0.0+0.0im            0.0+0.0im          0.346561+0.831302im
              0.0+0.0im            0.0+0.0im          0.561248+0.217897im
          0.345443-0.915812im   0.68487-0.605095im        0.0+0.0im
           0.77247-0.198694im   0.17008-0.854638im         0.0+0.0im
          0.958365-0.37833im   0.560486-0.834811im         0.0+0.0im

In [28]: round.(abs.(J),2)

Out[28]: 8×8 Array{Float64,2}:
          0.0    0.0    0.0    0.0    0.0    0.98   0.8    1.03
          0.0    0.0    0.0    0.0    0.0    0.91   0.87   1.01
          0.0    0.0    0.0    0.0    0.0    1.06   1.05   0.7
          0.0    0.0    0.0    0.0    0.0    1.24   0.79   0.9
          0.0    0.0    0.0    0.0    0.0    0.82   0.67   0.6
          0.98   0.91   1.06   1.24   0.82   0.0    0.0    0.0
          0.8    0.87   1.05   0.79   0.67   0.0    0.0    0.0
          1.03   1.01   0.7    0.9    0.6    0.0    0.0    0.0

In [29]: J,UJ=eig(J)

Out[29]: ([-3.30202, -0.965571, -0.76542, -7.50067e-17, 2.84505e-17, 0.76542, 0.965571, 3.30202]

In [30]: J

Out[30]: 8-element Array{Float64,1}:
          -3.30202
          -0.965571
          -0.76542
          -7.50067e-17
           2.84505e-17
           0.76542
           0.965571
           3.30202
```

1.4.5 Example - Random real matrix

```
In [31]: m=8
         n=5
         q=min(m,n)
         A=rand(-9:9,m,n)
```

```
Out[31]: 8Œ5 Array{Int64,2}:
          -8  -5  -7  -6  -7
          -9  -5  -8   6   2
           5  -5   0  -8  -4
           1   7   0   0  -9
          -5   5   4  -9  -5
          -1  -5   6   3  -9
          -8   8   3  -2   3
          -6  -1   7  -5   4

In [32]: U,,V=svd(A)

Out[32]: ([-0.200296 0.785765  -0.0531203 0.214687; 0.441635 0.58177  0.140171 -0.108251;  ; -0.

In [33]: # Fact 10
         trace(sqrtm(A'*A)), sum()

Out[33]: (78.87502223506586, 78.87502223506581)

In [34]: # Fact 11
         B=rand(n,n)
         det(B), prod(svdvals(B))

Out[34]: (-0.2458771005140237, 0.24587710051402362)

In [35]: # Fact 14
         norm(A), [1]

Out[35]: (19.45078341709841, 19.450783417098403)

In [36]: # Fact 15
         B=rand(m,n)
         abs(trace(A*B')), sum(svdvals(A)svdvals(B))

Out[36]: (50.64537766210586, 99.0009477844518)

In [37]: # Interlace Theorems (repeat several times)
         j=rand(1:q)
         Brow=svdvals(A[[1:j-1;j+1:m],:])
         Bcol=svdvals(A[:,[1:j-1;j+1:n]])
         j, , Brow, Bcol

Out[37]: (4, [19.4508, 17.9708, 17.4113, 12.9019, 11.1402], [18.7887, 17.9646, 17.1446, 11.2341,

In [38]: [1:end].>=Brow, [1:end-1].>=Bcol, [2:end].<=Brow[1:end-1], [2:end].<=Bcol

Out[38]: (Bool[true, true, true, true, true], Bool[true, true, true, true], Bool[true, true, tru

In [39]: # Weyl Inequalities
         B=rand(m,n)
         =svdvals(B)
         =svdvals(A+B)
         [  ]
```

```
Out[39]: 5×3 Array{Float64,2}:
          19.1934  19.4508  3.34773
          17.4086  17.9708  0.767945
          15.7687  17.4113  0.675385
          12.858   12.9019  0.370378
          11.0203  11.1402  0.211886
```

```
In [40]: @show k=rand(1:q)
         sum([1:k]),sum([1:k])+sum([1:k])
```

```
k = rand(1:q) = 4
```

```
Out[40]: (65.22873167893384, 72.89621356549667)
```

1.5 Matrix approximation

Let $A = U\Sigma V^*$, let $\tilde{\Sigma}$ be equal to Σ except that $\tilde{\Sigma}_{jj} = 0$ for $j > k$, and let $\tilde{A} = U\tilde{\Sigma}V^*$. Then $\operatorname{rank}(\tilde{A}) \leq k$ and

$$\min\{\|A-B\|_2 : \operatorname{rank}(B) \leq k\} = \|A-\tilde{A}\|_2 = \sigma_{k+1}(A)$$
$$\min\{\|A-B\|_F : \operatorname{rank}(B) \leq k\} = \|A-\tilde{A}\|_F = \left(\sum_{j=k+1}^{q} \sigma_j^2(A)\right)^{1/2}.$$

This is the **Eckart-Young-Mirsky Theorem**.

```
In [41]: A
```

```
Out[41]: 8×5 Array{Int64,2}:
          -8  -5  -7  -6  -7
          -9  -5  -8   6   2
           5  -5   0  -8  -4
           1   7   0   0  -9
          -5   5   4  -9  -5
          -1  -5   6   3  -9
          -8   8   3  -2   3
          -6  -1   7  -5   4
```

```
In [42]:
```

```
Out[42]: 5-element Array{Float64,1}:
          19.4508
          17.9708
          17.4113
          12.9019
          11.1402
```

```
In [43]: @show k=rand(1:q-1)
         k=3
         B=U*diagm([[1:k];zeros(q-k)])*V'

k = rand(1:q - 1) = 2


Out[43]: 8Œ5 Array{Float64,2}:
          -8.76602  -5.51867  -5.3996    -4.51139  -7.74616
          -8.46919  -5.62061  -8.52415    4.41628   3.12919
           3.66858  -2.36294   0.670024  -3.41882  -7.50717
           1.41571   1.12119   2.80764   -4.27887  -4.74724
          -5.24483   4.53814   4.6882    -8.69105  -5.05354
           1.98163  -2.68496  -0.406102  -2.62736  -6.28064
          -7.77341   6.57427   3.46896   -3.3301    4.20708
          -5.37598   4.44032   2.70188   -3.38539   1.47365

In [44]: A

Out[44]: 8Œ5 Array{Int64,2}:
          -8  -5  -7  -6  -7
          -9  -5  -8   6   2
           5  -5   0  -8  -4
           1   7   0   0  -9
          -5   5   4  -9  -5
          -1  -5   6   3  -9
          -8   8   3  -2   3
          -6  -1   7  -5   4

In [45]: norm(A-B), [k+1]

Out[45]: (12.901878235505487, 12.901878235505484)

In [46]: vecnorm(A-B),vecnorm([k+1:q])

Out[46]: (17.045925026559797, 17.045925026559797)

In [ ]:
```