

Rozklad na singulární hodnoty (SVD; Singular values decomposition)

Text pojednává o rozkladu na singulární hodnoty (SVD), popisuje matematické vlastnosti rozkladu, popis algoritmu, který se používá pro výpočet SVD, a aplikace SVD. Text čerpá z [?]. Text je určen pro studenty kurzu KMI/ALS1.

1 Rozklad

Tato část rozebírá vlastnosti SVD; především to, že SVD rozklad existuje pro každou matici (věta 1.2).

Budeme potřebovat následující tvrzení:

Tvrzení 1.1. *Pro matici $Q_1 \in \mathbb{R}^{m \times k}$ s ortonormálními sloupci existuje matice $Q_2 \in \mathbb{R}^{m \times (m-k)}$ tak že $Q = Q_1 \cdot Q_2$ je ortogonální matice.*

Tvrzení 1.1 říká, že ortonormální báze podprostoru prostoru $\mathbb{R}^m$ může být rozšířena na ortonormální bázi celého prostoru. Toto tvrzení je známý výsledek z lineární algebry.

Doplnění 1. *2-norma matice:*

$$\|A\|_2 = \max_x \frac{\|Ax\|_2}{\|x\|_2}$$
$$\|A\|_2 = \sqrt{\max_{1 \leq i \leq n} \lambda_i(A^T, A)} =: \sigma_{\max}(A)$$

Věta 1.2. *Jakákoli $m \times n$ matice A s $m \geq n$ může být rozložena na*

$$A = U \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^T, \quad (1)$$

kde $U \in \mathbb{R}^{m \times m}$ a $V \in \mathbb{R}^{n \times n}$ jsou ortogonální a $\Sigma \in \mathbb{R}^{n \times n}$ je diagonální

$$\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n),$$

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n \geq 0.$$

Důkaz. Předpoklad $m \geq n$ není nijak omezující: v opačném případě lze větu aplikovat na A^T .

Uvažujme maximalizační problém

$$\sup_{\|x\|_2=1} \|Ax\|_2$$

Protože hledáme supremum spojitě funkce nad uzavřenou množinou, je tohoto suprema dosaženo pro nějaký vektor x . Položme $Ax = \sigma_1 y$, kde $\|y\|_2 = 1$ a $\sigma_1 = \|A\|_2$ (podle definice).

Podle Tvzení 1.1 můžeme zkonstruovat ortogonální matice

$$Z_1 = (y \bar{Z}_2) \in \mathbb{R}^{m \times m}, W_1 = (x \bar{W}_2) \in \mathbb{R}^{m \times m}.$$

Pak

$$Z_1^T A W_1 = \begin{pmatrix} \sigma_1 & y^T A \bar{W}_2 \\ 0 & \bar{Z}_2^T A \bar{W}_2 \end{pmatrix},$$

protože $y^T A x = \sigma_1$ a $Z_2^T A x = \sigma_1 \bar{Z}_2^T y = 0$.

Položme

$$A_1 = Z_1^T A W_1 = \begin{pmatrix} \sigma_1 & w^T \\ 0 & B \end{pmatrix}.$$

Pak

$$\frac{1}{\sigma_1^2 + w^T w} \|A_1(\sigma_1 w)\|_2^2 = \frac{1}{\sigma_1^2 + w^T w} \left\| A_1 \begin{pmatrix} \sigma_1^2 + w^T w \\ B w \end{pmatrix} \right\|_2^2 \geq \sigma_1^2 + w^T w.$$

Ale $\|A_1\|_2^2 = \|Z_1^T A W_1\|_2^2 = \sigma_1^2$. Takže musí platit $w = 0$. A tedy jsme provedli jeden krok v diagonalizaci A . Důkaz je pak dokončen indukcí:

Předpokládejme, že

$$B = Z_2 \begin{pmatrix} \Sigma_2 & W_2 \\ 0 & \end{pmatrix} \quad \Sigma_2 = \text{diag}(\sigma_2, \dots, \sigma_n).$$

Pak máme

$$A = Z_1 \begin{pmatrix} \sigma_1 & 0 \\ 0 & B \end{pmatrix} W_1^T = Z_1 \begin{pmatrix} 1 & 0 \\ 0 & Z_2 \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & W_2^T \end{pmatrix} W_1^T.$$

Definujeme U, Σ, V jako

$$U = Z_1 \begin{pmatrix} 1 & 0 \\ 0 & Z_2 \end{pmatrix}, \quad \Sigma = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix}, \quad V = W_1 \begin{pmatrix} 1 & 0 \\ 0 & W_2^T \end{pmatrix}$$

Věta je tímto dokázána. $\square$

Sloupce U a V se nazývají **singulární vektory**, diagonální prvky σ_i matice Σ se nazývají **singulární hodnoty**.

1.1 SVD symbolicky

$$\begin{array}{c} n \\ \hline m \quad A \end{array} = \begin{array}{c} m \\ \hline m \quad U \end{array} \circ \begin{array}{c} n \\ \hline m \quad \begin{array}{c} \diagup \quad \Sigma \quad \diagdown \\ 0 \end{array} \end{array} \circ \begin{array}{c} n \\ \hline n \quad V^T \end{array}$$

Matici U můžeme vertikálně rozseknout na $m \times n$ matici U_1 a $m \times (m - n)$ matici U_2 .

$$U = \begin{array}{c} n \quad m - n \\ \hline m \quad \begin{array}{c} U_1 \quad U_2 \end{array} \end{array}$$

Při násobení $U\Sigma$ se hodnoty z U_2 násobí pouze nulami. Můžeme tedy psát:

$$\begin{array}{c} n \\ \hline m \quad A \end{array} = \begin{array}{c} n \\ \hline m \quad U_1 \end{array} \circ \begin{array}{c} n \\ \hline n \quad \begin{array}{c} \diagup \quad \Sigma \quad \diagdown \\ 0 \end{array} \end{array} \circ \begin{array}{c} n \\ \hline n \quad V^T \end{array}$$

2 Výpočet SVD

Výpočet SVD se obvykle děje v několika krocích, které jsou popsány v této sekci.

2.1 Householderova bidiagonalizace

Předpokládejme, že matice A je $m \times n$ a $m \geq n$. První krok ve výpočtu SVD matice A je zredukovat ji na horní bidiagonální matici pomocí Householderových transformací zleva a zprava:

$$A = H \begin{pmatrix} B \\ 0 \end{pmatrix} W^T, \quad B = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & \\ & & \ddots & \ddots & \\ & & & \alpha_{n-1} & \beta_{n-1} \\ & & & & \alpha_n \end{pmatrix}. \quad (2)$$

Toto se dělá, protože singulární hodnoty matice A odpovídají odmocninám vlastních hodnot matice $A^T A$. My nejřív podobnostních transformací matice A sestrojíme B , která je bidiagonální. Vlastní hodnoty $A^T A$ jsou počítány jako jako vlastní hodnoty $B^T B$, která je tridiagonální. Výpočet vlastních hodnot tridiagonální matice je snazší.

Doplnění 2. Householderova transformace (též Householderův reflektor): Matice ve tvaru $H = I - 2ww^T$, kde w je vektor takový, že $\|w\|_2 = 1$. Geometrický vektor Hx reprezentuje zrcadlový odraz vektoru x vzhledem k nadrovině kolmé k vektoru w . Householderova transformace je symetrická a ortogonální. Takže máme

$$H = H^T = H^{-1}.$$

Sestavíme dvě konečné sekvence Householderových transformací:

$$H^{(k)} = I - 2x^{(k)}x^{(k)T} \quad k = 1, 2, \dots, n \quad (3)$$

$$W^{(k)} = I - 2y^{(k)}y^{(k)T} \quad k = 1, 2, \dots, n-2 \quad (4)$$

(kde $x^{(k)T}x^{(k)} = y^{(k)T}y^{(k)} = 1$) tak, že

$$\underbrace{H^{(n)} \dots H^{(1)}}_H A \underbrace{W^{(1)} \dots W^{(n-2)}}_{W^T} = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & \\ & & \ddots & \ddots & \\ & & & \alpha_{n-1} & \beta_{n-1} \\ & & & & \alpha_n \\ \hline 0 & \dots & & & 0 \\ \vdots & & & & \vdots \\ 0 & \dots & & & 0 \end{pmatrix}.$$

Poznámka: Kvůli vlastnostem Householderových transformací máme

$$A = H \begin{pmatrix} B \\ 0 \end{pmatrix} W^T \quad \text{a} \quad \begin{pmatrix} B \\ 0 \end{pmatrix} = HAW^T.$$

Pokud položíme $A^{(1)} = A$ a definujeme:

$$\begin{aligned} A^{(k+\frac{1}{2})} &= H^{(k)} A^{(k)} & (k = 1, 2, \dots, n) \\ A^{(k+1)} &= A^{(k+\frac{1}{2})} W^{(k)} & (k = 1, 2, \dots, n) \end{aligned}$$

pak $H^{(k)}$ je určeno tak, že

$$a_{ik}^{k+\frac{1}{2}} = 0 \quad (i = k+1, \dots, m)$$

a $W^{(k)}$ tak, že

$$a_{kj}^{k+1} = 0 \quad (j = k+2, \dots, n)$$

Protože během této redukce používáme pouze ortogonální transformace, matice B (resp. $\begin{pmatrix} B \\ 0 \end{pmatrix}$) má tytéž singulární hodnoty jako A . Necht' σ je singulární hodnota A se singulárními vektory u a v . Pak $Av = \sigma u$ je ekvivalentní

$$\begin{pmatrix} B \\ 0 \end{pmatrix} \bar{v} = \sigma \bar{u}, \quad \bar{v} = W^T v, \quad \bar{u} = H^T u$$

podle (3) a (4).

Takže, pokud máme SVD

$$\begin{pmatrix} B \\ 0 \end{pmatrix} = \hat{U} \Sigma \hat{V}^T,$$

pak

$$A = H \hat{U} \Sigma \hat{V}^T W^T,$$

takže $U = H \hat{U}$, $V = \hat{V} W$.

Je zjevné, že matice $B^T B$ je tridiagonální a symetrická. Vhodnou metodou pro výpočet singulárních hodnot B je tridiagonální QR algoritmus s implicitními posuny aplikovanou na matici $B^T B$ bez toho, aby byla explicitně vypočítána.

Ilustrace Householderovy bidiagonalizační procedury na malé 6×5 matici. Násobením zleva dostáváme

$$A^{(1+\frac{1}{2})} = H^{T(1)} \begin{pmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{pmatrix} = \begin{pmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{pmatrix}.$$

Následně další rotací zprava chceme dostat nulové prvky v prvním (od sloupce 3 po n). Abychom toho dosáhli, zvolíme:

$$\mathbb{R}^{5 \times 5} \ni W^{(1)} = \begin{pmatrix} 1 & 0 & & & \\ 0 & Z_1 & & & \\ & & cc & & \\ & & & & \\ & & & & \end{pmatrix},$$

kde Z_1 je Householderova transformace. Protože tato transformace nemění elementy v prvním sloupci, nuly, které jsme předtím dostali do prvního sloupce zůstanou. Výsledek prvního kroku je

$$H^{T(1)}AW^{(1)} = \begin{pmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{pmatrix} W^{(1)} = \begin{pmatrix} \times & * & 0 & 0 & 0 \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \\ 0 & * & * & * & * \end{pmatrix} =: A^{(2)}.$$

Pokračujeme analogickým způsobem.

2.2 QR-algoritmus pro symetrickou tridiagonální matici

Redukujeme matici T na diagonální matici

$$T \mapsto \Lambda Q^T T Q, \quad Q = Q_1 Q_2 \cdots, \quad (5)$$

kde $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$. Matice Q_i budou ortogonální ale budou konstruovány pomocí rovinných rotací. Ovšem neexistuje konečný algoritmus pro výpočet Λ . Počítáme sekvenci matic.

$$T_0 := T, \quad T_i = Q_i^T T_{i-1} Q_i, \quad i = 1, 2, \dots, \quad (6)$$

tak, že konvertuje na diagonální matici

$$\lim_{i \rightarrow \infty} T_i = \Lambda.$$

Ukážeme na příkladech, že konvergence je velmi rychlá a v aritmetice plovcoucí řádové čárky algoritmus dokonce může být považován za konečný. Protože všechny transformace (6) jsou podobnostní transformace, diagonální prvky matice Λ jsou vlastní hodnoty matice T .

První verze QR-algoritmu pro symetrickou tridiagonální matici $T \in \mathbb{R}^{n \times n}$ (zjednodušený MATLAB kód).

```
for i=1:maxit          % dočasné zjednodušení
    mu=wilkshift(T(n-1:n,n-1:n));
    [Q,R]=qr(T-mu*I)
    T=R*Q+mu*I
end
```

```
function mu=wilkshift(T);
% vypočti Wilkinsonův posun
l = eig(T);
if abs(l(1)-T(2,2))<abs(l(2)-T(2,2))
    mu=l(1);
else
    mu=l(2);
end
```

Vidíme, že je nejdříve vypočítána QR dekompozice posunuté matice $QR = T - \mu I$ a pak je posun přidán zpět $T := RQ + \mu I$. Posun je ta vlastní hodnota 2×2 submatice v pravém dolním rohu, která je blíže t_{nn} . Tomuto se říká **Wilkinsonův posun**.

Demonstrace

$$T = \begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & -1 & 2 & -1 & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix}$$

TODO 1. *demonstrace*

Algoritmus pokračuje vypuštěním této vlastní hodnoty a sníží dimenzi zpracovávané matice o 1. Předběžná implementace algoritmu je popsána níže.

```
function [D,it] = qrtrid(T);
%Vypočítej vlastní hodnoty pro symetrickou tridiagonální
%Matici použitím QR algoritmu s explicitními Wilkinsonovými posuny
n=size(T,1); it=0;
for i=n:-1:3
    while abs(T(i-1,i)) > (abs(T(i,i))+abs(T(i-1,i-1)))*C*eps
        it=it+1;
        mu=wilkshift(T(i-1:i,i-1:i));
        [Q,R]=qr(T(i:i,i:i))-mu*eye(i);
        T=R*Q+mu*eye(i);
    end
    D(i)=T(i,i);
end
D(1:2)=eig(T(1:2,1:2))';
```

Pro submatici $T(i:i, i:i)$ je krok QR algoritmu prováděn dokud není splněna limitní podmínka

$$\frac{|t_{i-1,i}|}{|t_{i-1,i-1}| \cdot |t_{i,i}|} < C\mu,$$

kde C je malá konstanta a μ je jednotkové zaokrouhlení.

Poznámka 1. U skutečných algoritmů je limitní podmínka o něco komplikovanější. Navíc, kontrolují se všechny mimodiagonální hodnoty. Pokud je některá z nich blízká 0, matice se rozdělí na submatice a pokračuje se ve výpočtu nad každou z nich zvlášť.

Je samozřejmě neefektivní počítat QR dekompozici klasickým algoritmem, který je založený na Householderovém algoritmu. Na místo toho by dekompozice měla být počítána pomocí $n - 1$ rotací. Proceduru demonstrujeme na příkladu s tridiagonální 6×6 maticí $T = T^{(0)}$. První element pod diagonálou (shora) je snulována rotací zleva, $G_1^T(T^{(0)} - \tau I)$, pak je snulován druhý subdiagonální prvek, $G_2^T G_1^T(T^{(0)} - \tau I)$. Symbolicky,

$$\begin{pmatrix} \times & \times & & & & \\ \times & \times & \times & & & \\ & \times & \times & \times & & \\ & & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix} \rightarrow \begin{pmatrix} \times & \times & + & & & \\ 0 & \times & \times & + & & \\ & 0 & \times & \times & & \\ & & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}.$$

Vznikají nové nenulové elementy (označené +). Po $n - 1$ krocích dostaneme horní triangulární matici s třemi nenulovými diagonálami.

$$R = G_{n-1}^T \cdots G_1^T(T^{(0)} - \tau I) = \begin{pmatrix} \times & \times & + & & & \\ 0 & \times & \times & + & & \\ & 0 & \times & \times & & \\ & & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}.$$

Pak aplikujeme tyto rotace zprava, $RG_1 \cdots G_{n-1}$, tj. začínáme rotací zahrnující první dva sloupce. Pak pokračujeme rotací druhého a třetího sloupce. Výsledek po prvních dvou krocích je

$$\begin{pmatrix} \times & \times & \times & & & \\ + & \times & \times & \times & & \\ & + & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \\ & & & & & \times \end{pmatrix}.$$

Vidíme, že vzniklé nuly se zase systematicky zaplňují. Po $n - 1$ krocích dostáváme

$$T^{(1)} = RG_1 G_2 \cdots G_{n-1} + \tau I = \begin{pmatrix} \times & \times & \times & & & \\ + & \times & \times & \times & & \\ & + & \times & \times & \times & \\ & & + & \times & \times & \times \\ & & & + & \times & \times \\ & & & & + & \times \end{pmatrix}.$$

Udělali jsme podobnostní transformaci s $Q = G_1 G_2 \cdots G_{n-1}$ a s použitím $R = Q^T(T^{(0)} - \tau I)$ můžeme psát

$$T^{(1)} = RQ + \tau I = Q^T(T^{(0)} - \tau I)Q + \tau I = Q^T T^{(0)} Q, \quad (7)$$

takže víme, že $T^{(1)}$ je symetrická.

$$T^{(1)} = \begin{pmatrix} \times & \times & & & & \\ \times & \times & \times & & & \\ & \times & \times & \times & & \\ & & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times & \times \\ & & & & & \times & \times \end{pmatrix}.$$

Ukázali jsme následující tvrzení.

Tvrzení 2.1. *QR krok pro tridiagonální matici*

$$QR = T^{(k)} - \mu_k I, \quad T^{(k+1)} = RQ + \mu_k I,$$

je transformace podobnosti

$$T^{(k+1)} = Q^T T^{(k)} Q,$$

a tridiagonální struktura je zachována. Transformace může být vypočtena pomocí rotací během $\mathcal{O}(n)$ operací v plovoucí řádové čárce.

Z (7) by se mohlo zdát, že ty posuny nehrajou žádnou významnou roli. Naopak, používání posunů je nutné, aby byl tento algoritmus efektivní. Pokud by posuny nebyly dělány, algoritmus by konvergoval velmi pomalu (zhruba tak, jako mocninná metoda). Naopak se dá ukázat, že QR algoritmus má velice rychlou konvergenci (nebudeme dokazovat).

Věta 2.2. *Symetrický QR algoritmus s Wilkinsonovými posuny konverguje kubicky k rozkladu na vlastní hodnoty.*

2.3 QR-algoritmus pro tridiagonální matici s implicitními posuny

Důležité hledisko na QR algoritmu je, že posuny mohou být prováděny implicitně. Tato varianta je založená na tzv. *implicit Q theorem*, který je zde popsán v lehce zjednodušené podobě.

Věta 2.3. *Nechť A je symetrická matice a Q, V jsou ortogonální matice, tak že $Q^T A Q$ a $V^T A V$ jsou tridiagonální. Pak, pokud první sloupce Q a V jsou stejné, $q_1 = v_1$, pak platí $q_i = \pm v_i$, $i = 2, 3, \dots, n$.*

Důsledek této věty je, že pokud určíme a aplikujeme první transformaci v QR rozkladu $T - \mu I$, a pokud zkonstruujeme zbytek transformací tak, že nakonec

dostaneme tridiagonální matici, tak jsme provedli QR krok jako v Tvzení 2.1. Tato procedura je implementována následovně:

Určeme první rotaci tak, aby

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} \alpha_1 - \tau \\ \beta_1 \end{pmatrix} = \begin{pmatrix} \times \\ 0 \end{pmatrix} \quad (8)$$

Definujeme

$$G_1^T = \begin{pmatrix} c & s & & & \\ -s & c & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}$$

a aplikujeme tuto rotaci na T . Násobení zleva $G_1^T T$ uvede nový nenulový prvek na prvním řádku a nový nenulový prvek se také (symetricky) objeví v prvním sloupci při násobení zprava:

$$G_1^T T G_1 = \begin{pmatrix} \times & \times & + & & & \\ \times & \times & \times & & & \\ + & \times & \times & \times & & \\ & & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix},$$

kde $+$ představuje nový nenulový pohyb. Dále určíme rotaci v (2,3)-rovině, která zruší ty nové nenulové prvky a současně uvede nové nenulové prvky níže:

$$G_2^T G_1^T T G_1 G_2 = \begin{pmatrix} \times & \times & 0 & & & \\ \times & \times & \times & + & & \\ 0 & \times & \times & \times & & \\ & + & \times & \times & \times & \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}.$$

Tímto způsobem „ženeme“ nenulové prvky dolů, dokud nedostáváme:

$$\begin{pmatrix} \times & \times & & & & \\ \times & \times & \times & & & \\ 0 & \times & \times & \times & & \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & \times & \times & \times \end{pmatrix}.$$

Pak se nenulových prvku, které jsou mimo diagonály, finální rotací zbavíme úplně a dostaneme opět tridiagonální formu.

Všimněte si, že posun jsme použili pouze při první rotaci. Rotace byly aplikovány na *neposunutou* tridiagonální matici. Z Věty 2.3 vyplývá, že tento postup je ekvivalentní QR kroku tak, jak je definován v Tvzení 2.1.