ALS1 – Přednáška 7
SVD – výpočet

1 Výpočet SVD

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

1.1 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:

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

Toto se dělá, protože singulární hodnoty matice A odpovídají odmocninám vlastních hodnot matice A^TA. My nejdřív podobnostních transformací matice A sestrojíme B, která je bidiagonální. Vlastní hodnoty A^TA jsou počítány jako vlastní hodnoty B^TB, která je tridiagonální. Výpočet vlastních hodnot tridiagonální matice je jednodušší.

Doplnění 1 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*

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

Nechť x a y jsou stejně velké vektory, můžeme vytvořit Householderovu transformaci tak, že $y = Hx$ tak, že položíme

$$w = \frac{x - y}{\|x - y\|_2}.$$

Sestavíme dvě konečné sekvence Householderových transformací:

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

$$W^{(k)} = I - 2y^{(k)}{y^{(k)}}^T \quad k = 1, 2, \ldots, n-2 \tag{3}$$

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

$$\underbrace{H^{(n)} \cdots H^{(1)}}_{H} A \underbrace{W^{(1)} \cdots W^{(n-2)}}_{W^T} = \left(\begin{array}{cccc} \alpha_1 & \beta_1 & & \\ & \alpha_2 & \beta_2 & \\ & & \ddots & \ddots \\ & & & \alpha_{n-1} \; \beta_{n-1} \\ & & & \alpha_n \\ \hline 0 & \cdots & & 0 \\ \vdots & & & \vdots \\ 0 & \cdots & & 0 \end{array}\right).$$

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:

$$A^{(k+\frac{1}{2})} = H^{(k)} A^{(k)} \qquad (k = 1, 2, \ldots, n)$$

$$A^{(k+1)} = A^{(k+\frac{1}{2})} W^{(k)} \qquad (k = 1, 2, \ldots, n)$$

pak $H^{(k)}$ je určeno tak, že

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

a $W^{(k)}$ tak, že

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

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

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

podle (2) a (3).

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

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:

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

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

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

Pokračujeme analogickým způsobem.

1.2 QR-algoritmus pro symetrickou tridiagonální matici

Redukujeme matici T na diagonální matici

$$T \mapsto \Lambda Q^T T Q, \quad Q = Q_1 Q_2 \cdots, \tag{4}$$

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

$$T_0 := T, \quad T_i = Q_i^T T_{i-1} Q_i, \quad i = 1, 2, \ldots, \tag{5}$$

tak, že konvertuje na diagonální matici

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

Ukážeme na příkladech, že konvergence je velmi rychlá a v aritmetice plovoucí řádové čárky algoritmus dokonce může být považován za konečný. Protože všechny transformace (5) 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=1(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**.

Algoritmus pokračuje vypuštěním této vlastní hodnoty 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.

Remark 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,

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

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

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

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

$$T^{(1)} = RG_1G_2 \cdots G_{n-1} + \tau I = \begin{pmatrix} \times & \times & \times & & & & \\ + & \times & \times & \times & & & \\ & + & \times & \times & \times & & \\ & & + & \times & \times & \times & \\ & & & + & \times & \times & \\ & & & & + & \times & \end{pmatrix}.$$

Udělali jsme podobnostní transformaci s $Q = G_1G_2 \cdots G_{n-1}$ a s použitím $R = Q^T(T^{(0)} - \tau I)$ můžeme psát

$$T^{(1)} = RQ + \tau I = Q^T(T^{(0)} - \tau I)Q + \tau I = Q^T T^{(0)} Q, \tag{6}$$

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

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

Ukázali jsme následující tvrzení.

Proposition 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 (6) by se mohlo zdát, že ty posuny nehrají 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).

Theorem 1. *Symetrický QR algoritmus s Wilkinsonovými posuny konverguje kubicky k rozkladu na vlastní hodnoty.*

1.3 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.

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

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 Tvrzení 1. Tato procedura je implementována následovně:

Určeme první rotaci tak, aby

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} \alpha_1 - \tau \\ \beta_1 \end{pmatrix} = \begin{pmatrix} \times \\ 0 \end{pmatrix} \tag{7}$$

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:

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

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

Všimněte si, že posun jsme použili pouze při první rotaci. Rotace byly aplikovány na *neposunutou* tridiagonální matici. Z Věty 2 vyplývá, že tento postup je ekvivalentní QR kroku tak, jak je definován v Tvrzení 1.