KMI/ASL1 – Algoritmy a složitost 1

Singulární rozklad matice

Jan Konečný

5. prosince 2013

Věta o singulárním rozkladu matic

Věta

Jakákoli $m \times n$ matice A s $m \geq n$ může být rozložena na

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

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

$$\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n),$$
$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0.$$

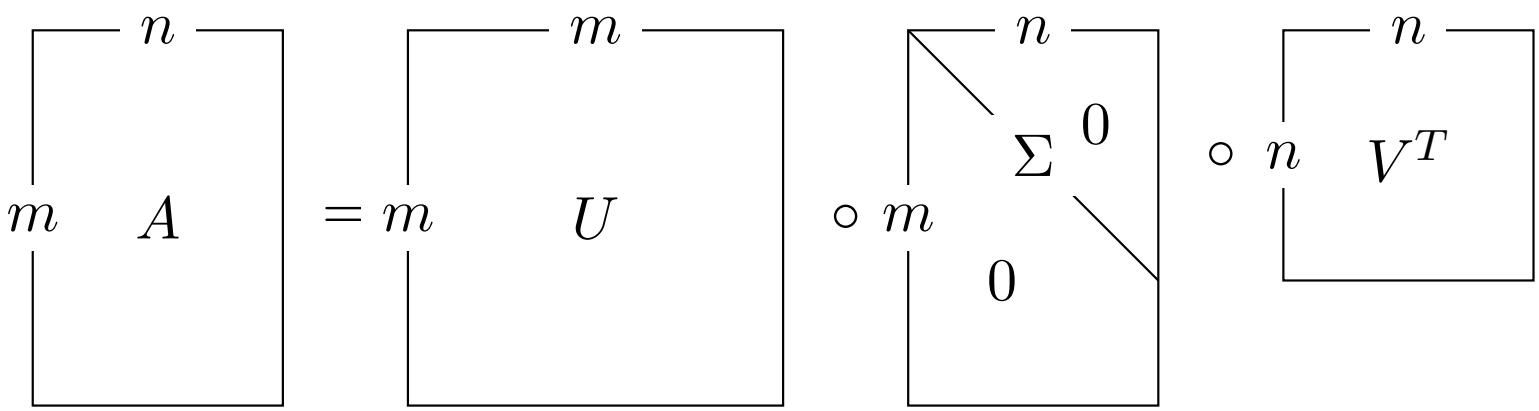
Poznámka

(1) se nazývá *singulární rozklad* (singular value decomposition, SVD); $\sigma_1, \sigma_2, \ldots, \sigma_n$ se nazývají *singulární hodnoty*.

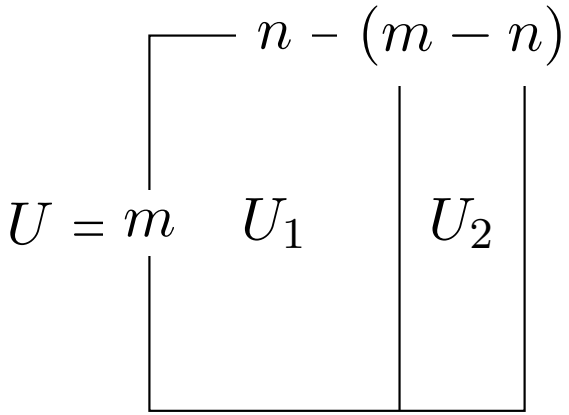
Poznámka

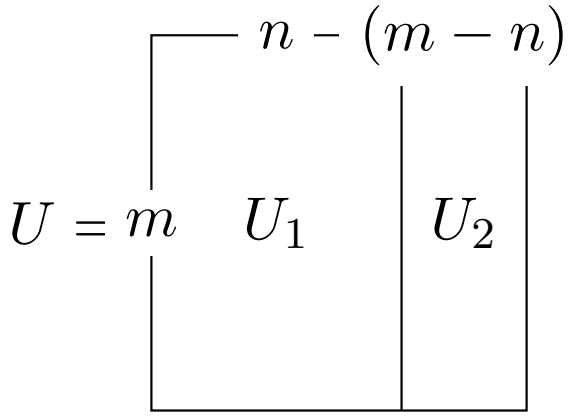
Předpoklad $m \geq n$ není nijak omezující: v opačném případě lze větu aplikovat na A^T.

SVD symbolicky

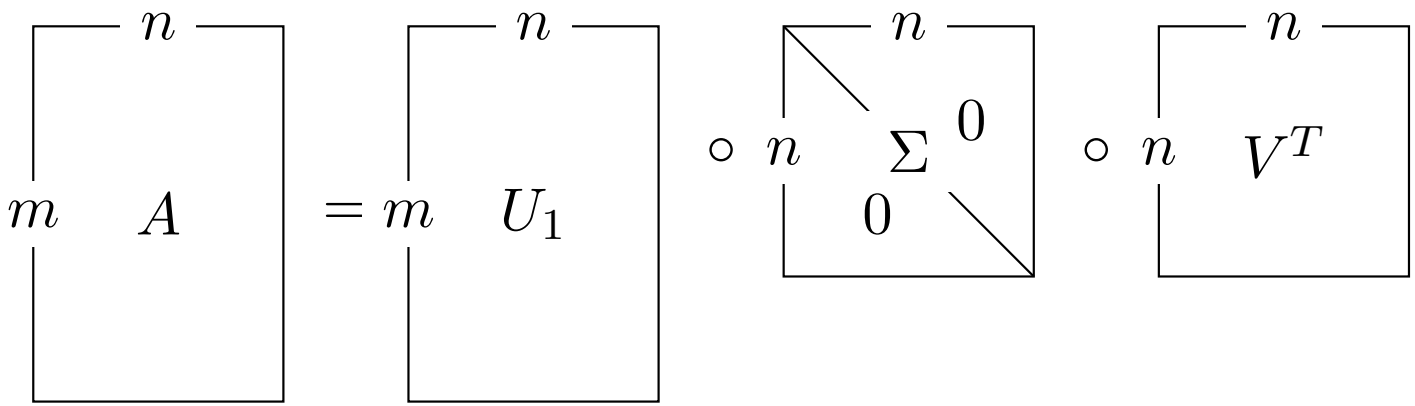


Matici U můžeme vertikálně rozseknout na $m \times n$ matici U_1 a $m \times (m-n)$ matici U_2.





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



Důkaz věty o singulárním rozkladu

Důkaz (část I)

Uvažujme maximalizační problém

$$\sup_{\|\vec{x}\|_2=1} \|A\vec{x}\|_2 .$$

Tedy chceme najít normovaný (sloupcový) vektor $\vec{x}$, tak aby 2-norma (tj. Euklidovská velikost) (sloupcového) vektoru $A\vec{x}$ byla co největší.

Připomínka

2-norma matice:

$$\|A\|_2 = \max_{\vec{x}} \frac{\|A\vec{x}\|_2}{\|\vec{x}\|_2}$$

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

Důkaz (část II.)

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

Podle násl. lemmatu můžeme zkonstruovat ortogonální matice

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

Lemma

Pro matici $Q_1 \in \mathbb{R}^{m\times k}$ s ortonormálními sloupci existuje matice $Q_2 \in R^{m\times(m-k)}$ tak že $Q = (Q_1Q_2)$ je ortogonální matice.

To lemma 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.

Důkaz (část III.)

Máme tedy ortogonální matice

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

Pak

$$Z_1^T A W_1 = (\vec{y}\overline{Z}_2)^T A(\vec{x}\overline{W}_2) = \begin{pmatrix} \vec{y}^T A\vec{x} & \vec{y}^T A\overline{W}_2 \\ \overline{Z}_2^T A\vec{x} & \overline{Z}_2^T A\overline{W}_2 \end{pmatrix} = \begin{pmatrix} \sigma_1 & \vec{y}^T A\overline{W}_2 \\ 0 & \overline{Z}_2^T A\overline{W}_2 \end{pmatrix}$$

kde $\sigma_1 = \vec{y}^T A\vec{x}$.

Použila se rovnost $\overline{Z}_2^T A\vec{x} = \sigma_1 \overline{Z}_2^T \vec{y} = 0$.

Důkaz (část IV.)

Označme

$$B = \overline{Z}_2^T A \overline{W}_2$$

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

Pak

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

Z toho dostáváme:

$$\left\| A_1 \begin{pmatrix} \sigma_1 \\ \vec{w} \end{pmatrix} \right\|_2 \geq \sigma_1^2 + \vec{w}^T\vec{w}$$

A z toho:

$$\|A_1\|_2 \geq \sqrt{\sigma_1^2 + \vec{w}^T\vec{w}}$$

Důkaz (část V.)

Máme tedy:

$$\|A_1\|_2 \geq \sqrt{\sigma_1^2 + \vec{w}^T \vec{w}}.$$

Ale přitom

$$\|A_1\|_2 = \left\|Z_1^T A W_1\right\|_2 = \|A\|_2 = \sigma_1,$$

protože Z_1 a W_1 jsou ortogonální matice.

Z toho dostáváme, že $\vec{w} = 0$. Celkově tedy

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

Tedy, provedli jsme jeden krok v diagonalizaci matice A.

[

Důkaz (část VI.)]
Předpokládejme, že

$$B = Z_2 \begin{pmatrix} \Sigma_2 \\ 0 \end{pmatrix} W_2^T \quad \Sigma_2 = \mathrm{diag}(\sigma_2, \ldots, \sigma_n).$$

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

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

Tímto je dokázána existence.

Důkaz (část VII.; jednoznačnost)

Všimněme si, že σ_1 je jednoznačně daná tím, že je rovno $\|A\|_2$.

Předpokládáme, že existuje $\vec{x'}$, t.ž. $\left\|\vec{x'}\right\|_2 = 1$ a $\left\|A\vec{x'}\right\|_2 = \sigma_1$.
Definujme jednotkový vektor $\vec{z}$, ortogonální s $\vec{x}$:

$$\vec{z} = \frac{\vec{x'} - (\vec{x}^T\vec{x'})\vec{x}}{\left\|\vec{x'} - (\vec{x}^T\vec{x'})\vec{x}\right\|_2}$$

Protože $\|A\|_2 = \sigma_1$, máme $\|A\vec{z}\|_2 \leq \sigma_1$.

Musí to být rovnost, protože platí $\vec{x'} = c\vec{x} + s\vec{z}$ pro nějaké konstanty $|c|^2 + |s|^2 = 1$. A my bychom dostali $\left\|A\vec{x'}\right\|_2 < \sigma_1$.

To je tedy druhý singulární vektor A odpovídající hodnotě σ_1. To vede k existenci $\vec{y}$, t.ž. $\|\vec{y}\|_2 = 1$ a $A_1\vec{y} = \sigma_1$. Tedy pokud singulární vektor $\vec{x}$ není unikátní, pak σ_1 není jednoduchá.

Vlastnosti matic přes SVD

Věta

Hodnost $r(A)$ matice A je počet singulárních hodnot.

Označme počet nenulových singulárních hodnot r.

Důkaz.

Hodnost diagonální matice je rovna počtu jejích nenulových hodnot,
a v rozkladu $A = U\Sigma V^T$, U a V jsou regulární (mají plnou hodnost).
Takže $r(A) = r(\Sigma) = r$. □

Věta

$\|A\|_2 = \sigma_1$

Věta

Obor hodnot $\mathrm{R}(A) = \langle u_1, \ldots, u_r \rangle$,
Nulový prostor $\mathrm{N}(A) = \langle v_{r+1}, \ldots, v_n \rangle$.

Připomínka

Pro $A \in \mathbb{R}^{m \times n}$:
Obor hodnot $\mathrm{R}(A) = \{\vec{b} \in \mathbb{R}^m \mid b = \vec{A\vec{x}}$ pro nějaké $x \in \mathbb{R}^n\}$,
Nulový prostor $\mathrm{N}(A) = \{\vec{x} \in \mathbb{R}^n \mid A\vec{x} = \vec{0}\}$.

Zápis $\langle \vec{x_1}, \ldots, \rangle$ označuje protor generovaný vektory $\vec{x}_1, \ldots$

Důkaz.

Jednoduchý následek toho, že $\mathrm{R}(\Sigma) = \{e_1, \ldots, e_r\}$ a
$\mathrm{N}(\Sigma) = \{e_{r+1}, \ldots, e_n\}$. □

Věta

(Nenulové) singulární hodnoty A jsou druhé odmocniny (nenulových) vlastních hodnot $A^T A$ nebo AA^T (tyto matice mají stejné nenulové hodnoty).

Důkaz.

Z výpočtu

$$A^T A = (U\Sigma V^T)^T (U\Sigma V^T) = V\Sigma^T U^T U\Sigma V^T = V\Sigma^T \Sigma V^T$$

vidíme, že $A^T A$ je podobná $\Sigma^T \Sigma$, a proto má tytéž vlastní hodnoty.
Vlastnosti diagonální matice $\Sigma^T \Sigma$ jsou $\sigma_1^2, \sigma_2^2, \ldots, \sigma_n^2$.
Podobně pro AA^T. □

Aproximace maticí nižší hodností

- SVD může být zapsano jako suma matic hodnosti jedna:

$$A = \sum_{j=1}^{n} \sigma_j \vec{u}_j \vec{v}_j^T$$

- Nejlepší aproximace A hoností h vzhledem k 2-normě je

$$A_h = \sum_{j=1}^{h} \sigma_j \vec{u}_j \vec{v}_j^T$$

  s $\|A - A_h\|_2 = \sigma_{h+1}$

- Stejně tak u Frobeniovy normy: $||A - A_h||_F = \sqrt{\sigma_{h+1}^2 + \cdots + \sigma_n^2}$.

Připomínka

Frobeniova norma:

$$||A||_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{\frac{1}{2}}$$

K čemu to je?

- Výpočet vlastností matic přes SVD
  - hodnost,
  - báze oboru hodnot a nulového prostoru,
  - norma matice $\|\cdot\|_2$,
- Aproximace maticí nižší hodností
- Zpracování signálu/obrazu
  - komprese
  - odstranění šumu

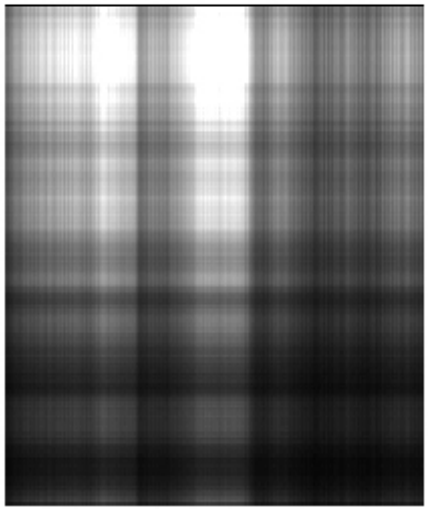
Poznámka

Mrkněte na: `Lec 29 | MIT 18.06 Linear Algebra, Spring 2005.`

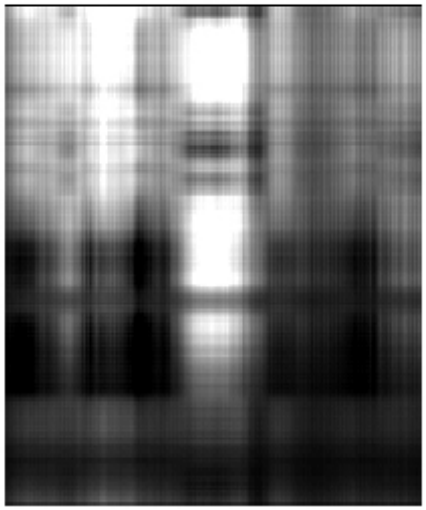
Zpracování signálu/obrazu (ukradeno)

Original, rank=200

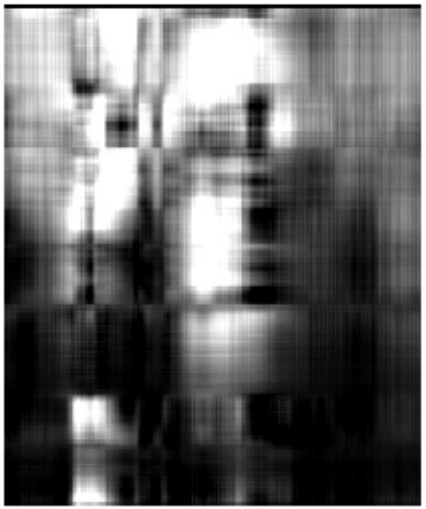
Rank=1



Rank=2



Rank=5



Rank=15

Rank=50

Použití v information retrieval

M. W. Berry, S.T. Dumais, G.W. O'Brien.
Using linear algebra for intelligent information retrieval.
Siam Review, 37(4): 583–595, 1995.

Hledáme v repozitáři dokument týkající se určitého tématu.

dokument = vektor (řádek matice) čísel – každá dimenze je jedno možné slovo.
čísla: slovo není přítomno: 0
slovo je přítomno: 1 nebo počet nebo frekvence.

repozitář = matice (dokumenty $\times$ slova)
problém: příliš mnoho sloupců (100 000+)

Vyhledávání informací (latent semantic indexing)

Základní problém: je zadán dotaz: slova k vyhledávání

- najdi všechny dokumenty, který je obsahuje
- najdi dokumenty, které se jich obsahově týkají.
- dokumenty setřiď podle duležitosti

dotaz = krátký pseudodokument obsahující pouze vyhledávaná slova; tedy může být popsán tímtéž vektorem

cíl = najdi řádky matice, které jsou podobné

podobnost: kosinová podobnost

- vektory jsou blízké, pokud ukazují na stejné místo
- euklidovská vzdálenost není dobrá (dokument je vždy velmi malý)

Vyhledávání informací (latent semantic indexing)

Několik problémů:

- příliš mnoho dimenzí (možných slov 100000+)
- homonyma (bear – medvěd, nosit)
- synonyma (car, automobile)

Lepší je namísto matice A použít matici U z SVD rozkladu $A = U\Sigma V$ ořezanou na nějaké vhodné k, např k kolem 10000.
Výhody:

- eliminace velké části redundance
- synonyma budou blízko sebe v transformovaném prostoru
- homonyma budou umístěna v oblasti mezi svými významy (blíže k tomu častějšímu)

Dokonce může být použito k vyhledávání napříč jazyky (vyhledávané slovo je v jiném jazyce než nalezené dokumenty).

S.T.Dumais, T.A. Letsche, M.L. Littman, T.K. Landauer.
Automatic cross-language retrieval using Latent Semantic Indexing.
In AAAI-97 Spring Symposium Series: Cross-Language Text and Speech Retrieval, pages 18–24, 1997.

Hodnocení objektů a atributů podle zajímavosti

Po oříznutí je jen k směrů, kterými můžou vektory ukazovat a být ortogonální.
vektory které jsou u středu:

- jsou podobné s mnoha jinými objekty
- jsou odlišné od mnoha jiných objektů

Objekty které skončí daleko od středu jsou zajímavé.
Například: matice dokumenty $\times$ slova

- dokumenty používající běžná slova běžným způsobem budou nezajímavé,
- dokumenty používající pouze vzácná slova budou nezajímavé.

Použití:

D.B. Skillicorn.
Beyond keyword filtering for message and conversation detection.
In IEEE Int. Conf. on Intelligence and Security Informatics (ISI2005), pages 231–243. Spriger-Verlag LNCS 3495, May 2005

kolaborativní filtrování (collaborative filtering)

Doporučovací systémy: používají informace o objektech a jejich vlastnostech, aby předvídaly nové vlastnosti.

Například: Amazon, Youtube

několik způsobů, jak to vytvořit: obsahové doporučovaní (content recommender) $\times$ kolaborativní filtrování

obsahové doporučovaní (content recommender) × kolaborativní filtrování

content recommender

- každý produkt má seznam vlastností
- zákazník má seznam vlastnosti, který vzniká slitím vlastností produktů, které koupil.
- doporučovaný produkt pak bude ten, jehož seznam vlastností se bude shodovat se seznamem vlastností zákazníka.

nevýhody:

- vlastnosti produktu musí určovat člověk
- nijak není využíváno informace, jak nakupovali ostatní zákazníci

kolaborativní filtrování

v nejjednodušší formě:
matice zákazníci $\times$ produkty, 0 nekoupil, 1 koupil. (nebo třeba hodnocení 1–10, jak se mu to líbilo)

měli bychom najít podobné řádky a doporučit produkty které nejsou v řádku aktuáního zákazníka, ale jsou v těch podobných.

několik problému s touto jednoduchou formou:

- nalezený nejpodobnější řádek má malý překryv, (velmi velká a řídká matice)
- příliš mnoho knih k doporučení, není zjevné jak vybrat nejlepší
- „homonyma“ a „synonyma“

kolaborativní filtrování

Zlepšení pomocí SVD (hledání v U):

- lidé nenakupují náhodně, mají preference – dají se očekávat faktory.
- po oříznutí menší dimenzionalita.

(malý) problém: pracujeme teď s maticí zákazníci $\times$ faktory ...
Můžeme například:

- posunout nalezený nejbližší rádek směrem k zákazníkově
- takto upravěný řádek zobrazit zpět do prostoru produktů (vynásobením maticemi Σ a V).
- doporučit ty produkty, kde došlo k největšímu zvýšení

Můžeme uvažovat (podporovat) pouze „zajímavé“ zákazníky.

kolaborativní filtrování

Rozšíření SVD: PDDP
Principal Direction Divisive Partitioning používá SVD vytvoření rozhodovacího stromu:

- Vypočti SVD, uvažuj směr prvního singulárního vektoru
- Rozděl objekty podle jejich pozice vzhledem k tomuto vektoru. Výsledné 2 části řádky původní matice jsou odděleny.
- Rekurzivně zpracuj ty 2 části.

Výsledkem je binární strom, který dělí data způsobem, který odráží jejich důležitost.

`www.cs.umn.edu/~boley/Distribution/PDDP.html`

Výpočet SVD

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:

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

- protože singulární hodnoty matice A odpovídají odmocninám vlastních hodnot matice $A^T A$.
- nejdřív pomocí podobnostních transformací matice A sestrojíme B, která je bidiagonální.
- vlastní hodnoty $A^T A$ jsou počítány jako jako vlastní hodnoty $B^T B$ (je tridiagonální).
- výpočet vlastních hodnot tridiagonální matice je snažší.

Householderova transformace (též Householderův reflektor)

Věta

Nechť $\vec{x}, \vec{y}$ jsou stejně velké vektory. Existuje matice H, t.ž. $H\vec{x} = \vec{y}$ a

$$H = I - 2\vec{w}\vec{w}^T,$$

kde

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

Matice ve tvaru $H = I - 2\vec{w}\vec{w}^T$, kde $\vec{w}$ je vektor takový, že $\|w\|_2 = 1$.

Poznámka

Geometrický vektor $H\vec{x}$ reprezentuje zrcadlový odraz vektoru $\vec{x}$ vzhledem k nadrovině kolmé k vektoru $\vec{w}$.

Householderova transformace je symetrická a ortogonální. Takže máme

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

Zpátky k bidiagonalizaci

Sestavíme dvě konečné sekvence Householderových transformací:

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

(kde $||\vec{x}^{(k)}|| = ||\vec{y}^{(k)}|| = 1$) tak, že

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

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

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

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

Ilustrace Householderovy bidiagonilační procedury na malé 6×5 matici.

Násobením zleva dostáváme

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

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

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

- Dostali jsme bidiagonální matici B, která má stejné singulární hodoty jako A.
- Víme, že singulární hodnoty B jsou vlastní hodnoty $B^T B$.
- $B^T B$ je tridiagonální.

Následuje algoritmus pro výpočet vlastních hodnot tridiagonální matice.

QR-algoritmus pro symetrickou tridiagonální matici

Redukujeme tridiagonální matici T na diagonální matici

$$T \mapsto \Lambda = Q^T T Q, \quad Q = Q_1 Q_2 \cdots , \tag{2}$$

kde $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_n)$.

Matice Q_i budou ortogonální ale budou konstruovány pomocí rovinných rotací.

Ale: neexistuje konečný algoritmus pro výpočet Λ.
Počítáme sekvenci matic.

$$T_0 := T, \quad T_i = Q_i^T T_{i-1} Q_i, \quad i = 1, 2, \ldots, \tag{3}$$

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 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**.

QR Algoritmus pro symetrickou tridiagnální matici

Demonstrace

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

Po aplikaci jedoho kroku dostáváme:

```
T =  1.0000    0.7071         0         0         0         0
     0.7071    2.0000    1.2247         0         0         0
          0    1.2247    2.3333   -0.9428         0         0
          0         0   -0.9428    1.6667    0.8660         0
          0         0         0    0.8660    2.0000   -0.5000
          0         0         0         0   -0.5000    3.0000
```

QR Algoritmus pro symetrickou tridiagnální matici

Po dalších 3 krocích dostáváme (zobrazen je jen pravý dolní roh 2×2)

```
 2.36530292572181   -0.02609619264716
-0.02609619264716    3.24632297453998

 2.59270689576885    0.00000366571479
 0.00000366571479    3.24697960370634

 2.77097818052654    0.00000000000000
-0.00000000000000    3.24697960371747
```

mimodiagonální hodnoty jsou snulovány, můžeme poslední sloupec odseknout a pokražovat stejným způsobem zbytkem matice.

mimodiagonální hodnoty jsou snulovány, můžeme poslední sloupec odseknout a pokražovat stejným způsobem zbytkem matice.

$$\begin{pmatrix} 0.4374 & 0.3176 & 0 & 0 & 0 \\ 0.3176 & 0.7961 & -0.4395 & 0 & 0 \\ 0 & -0.4395 & 1.3198 & 0.2922 & 0 \\ 0 & 0 & 0.2922 & 3.4288 & 0.5902 \\ 0 & 0 & 0 & 0.5902 & 2.7710 \end{pmatrix}$$

Opětně ...

$$\begin{pmatrix} 3.74629910763238 & -0.01184028941948 \\ -0.01184028941948 & 2.44513898239641 \end{pmatrix}$$

$$\begin{pmatrix} 3.68352336882524 & 0.00000009405188 \\ 0.00000009405188 & 2.44504186791263 \end{pmatrix}$$

$$\begin{pmatrix} 3.54823766699472 & 0.00000000000000 \\ -0.00000000000000 & 2.44504186791263 \end{pmatrix}$$

Algoritmus pokračuje vypuštěním této vlastní hodnoty a sníží dimenzi zpracovávané matice o 1.

Předběžná implementace:

```
function [D,it] = qrtrid(T);
  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(1:i,1: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 $T(1:i, 1:i)$ je krok QR algoritmu prováděn dokud není splněna limitní podmínka

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

kde C je malá konstanta a ϵ je jednotkové zaokrouhlení.

Poznámka

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 neefektivní počítat QR dekompozici klasickým algorimtem, který je založený na Householderovém algoritmu.
- Namísto toho by dekompozice měla být počítána pomocí $n - 1$ rovinných rotací.

Demonstrace: 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 elemety (označené $+$).

Po $n-1$ krocích dostaneme horní triangulární matici s třemi nenulovými diagonálami.

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

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

$$T^{(1)} = RQ + \tau I = Q^T(T^{(0)} - \tau I)Q + \tau I = Q^T T^{(0)} Q, \qquad (4)$$

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

Ukázali jsme následující tvrzení.

Tvrzení

QR krok pro tridiagonální matici

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

je podobnostní tranformace

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

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

- 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).

QR-algoritmus pro tridiagonální matici s implicitními posuny

- Posuny mohou být prováděny implicině.
- Tato varianta je založená na tzv. *implicit Q theorem*, který je zde popsán v lehce zjednodušené podobě:

Věta

Nechť A je symetrická matice a Q, V jsou ortogonální matice, tak že $Q^T AQ$ a $V^T AV$ jsou tridiagonální. Pak, pokud prvni sloupce Q a V jsou stejné, $q_1 = v_1$, pak platí $q_i = \pm v_i$, $i = 2, 3, \ldots, n$.

Tedy, 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.

Tato procedura je implementována následovně:
Určeme první rotaci tak, aby

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} \alpha_1 - \tau \\ \beta_1 \end{pmatrix} = \begin{pmatrix} \times \\ 0 \end{pmatrix} \tag{5}$$

Definujeme

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

a aplikujeme tuto rotaci na T.

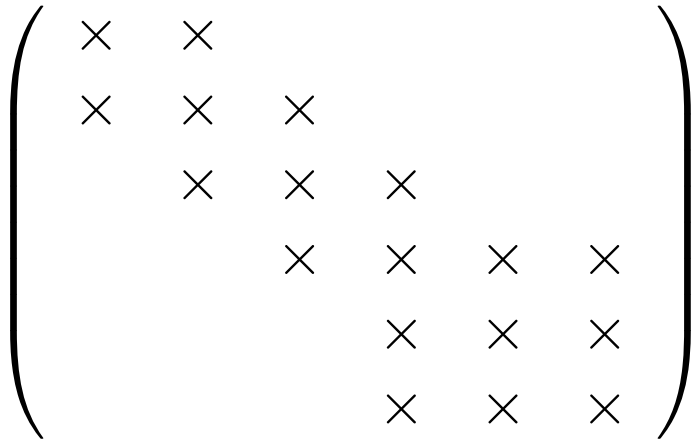
Násobení zleva $G_1^T T$ uvede nový nenulový prvek na prvním řádku a nový nenulový prvek se také (symetricky) objeví v prvním sloupci pri násobení zprava:

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

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:



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 vyplývá, že tento postup je ekvivalentní QR kroku tak, jak je definován v Tvrzení.

Výpočet vlastních hodnot $B^T B$ bez explicitního výpočtu $B^T B$

Máme

$$B = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ & \alpha_2 & \beta_2 & & \\ & & \ddots & \ddots & \\ & & & \alpha_{n-1} & \beta_{n-1} \\ & & & & \alpha_n \end{pmatrix}.$$

Wilkinsonův posun τ (ekvivalentní posunu u $B^T B$)
Tedy jakov vlastní hodnota matice

$$\begin{pmatrix} \alpha_{n-1}^2 + \beta_{n-1}^2 & \alpha_{n-1}\beta_n \\ \alpha_{n-1}\beta_n & \alpha_n^2 + \beta_n^2 \end{pmatrix}$$

která je blíže k $\alpha_n^2 + \beta_n^2$.

Vypočítáme matici rotace Q_1, tak aby

$$= \begin{pmatrix} c & s \\ -s & c \end{pmatrix}^T \cdot \begin{pmatrix} \alpha_1^2 - \tau \\ \alpha_1 \beta_2 \end{pmatrix} = \begin{pmatrix} * \\ 0 \end{pmatrix}$$

a vypočítáme BQ_1.

Dostaneme něco takového:

$$BQ_1 = \begin{pmatrix} \times & \times & & & & & \\ * & \times & \times & & & & \\ & & \times & \times & & & \\ & & & \times & \times & & \\ & & & & \times & \times & \\ & & & & & \times & \end{pmatrix}$$

- potřebujeme snulovat $*$,
- chceme najít P_2 a G_2 tak, aby $P_2(BQ_1)Q_2$ je bidiagonální a $Q_2\vec{e}_1 = \vec{e}_1$.

Zbytek jasný...