

Si $A \in \mathbb{C}^{m \times n}$, $A = (a_{ij})$, la matriz A^H es la transpuesta de A con entradas \bar{a}_{ji} : $\forall i=1 \dots m, j=1 \dots n$

Si $A \in \mathbb{C}^{m \times n}$ entonces A tiene columnas ortonormales si $A^H A = I_n$ y $A A^H = I_m$ (en el caso A real $A^T A = I_n$, $A A^T = I_m$)

Si $A \in \mathbb{C}^{k \times k}$ entonces A es unitaria si $A^H A = A A^H = I_k$
($A^{-1}, A^T A = A A^T = I_k$ y A se dice ortogonal)

SVD

Si $A \in \mathbb{C}^{m \times n}$ entonces $A = U \Sigma V^H$ donde $U \in \mathbb{C}^{m \times m}$, $V \in \mathbb{C}^{n \times n}$ unitarias
 $\Sigma \in \mathbb{R}^{m \times n}$ diagonal

Como $A^H = V \Sigma^T U^H$ asumimos $m \geq n$ s.p.g. y $r = \text{rank}(A) \therefore r \leq n$

Si denotamos $U = [u_1 \ u_2 \ \dots \ u_m]$, $V = [v_1 \ v_2 \ \dots \ v_n]$, $\Sigma = \text{diag}(\sigma_i)_{i=1}^n$, entonces u_i, v_i son los vectores singulares de A izquierdos y derechos respectivamente, σ_i son los valores " " " y cumplen $\sigma_i > 0$ y son únicos. Tipicamente las entradas de Σ y las correspondientes columnas de U y V se ordenan de acuerdo a:

$$\sigma_1 > \sigma_2 > \dots > \sigma_r > 0, \sigma_{r+1} = \sigma_{r+2} = \dots = \sigma_n = 0$$

Los algoritmos (Jacobi) que vemos se aplican para A real y se extiende al caso complejo por ello asumimos $a_{ij} \in \mathbb{R} \ \forall i, j$ en lo siguiente, con esta suposición $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$ ortogonales

Si A es simétrica se pueden ajustar los signos de los entradas de Σ de modo que $U = V$ y $\therefore A = V^T D V$ (descomposición espectral) con U ortogonal ($U^T U = U U^T = I$) y D diagonal

Entre las aplicaciones de la SVD:
+) Procesamiento digital de imágenes y señales
+) Sistemas de recomendación
+) Componentes principales

Los algoritmos numéricos que calculan la SVD pueden ser basados en QR o basados en Jacobi. En computación searial se prefieren los " " por la velocidad. Los basados en Jacobi son más precisos y poseen la propiedad de que pueden ejecutarse en paralelo.

Hay 2 tipos de métodos basados en Jacobi: two-sided y one-sided. Los primeros son computacionalmente más costosos y típicamente se utilizan para encontrar la descomposiciónpectral de una matriz simétrica, los one-sided además de usarse para calcular la SVD se adaptan para calcular la descomposiciónespectral.

Método Jacobi one-sided (MJSOS)

Idea: Construimos una matriz V orogonal tal que $AV = W + \text{carga residual}$ ortogonales, normalizando las columnas de W (bajo la norma euclídea) distintas de cero obtenemos: $W = [U_r \ 0] \begin{pmatrix} \Sigma_r & 0 \\ 0 & 0 \end{pmatrix}$ con $U_r \in \mathbb{R}^{m \times r}$ con columnas orthonormales $\neq 0$, y $\Sigma_r = \text{diag}(\sigma_i)_{i=1}^r$

∴ Una SVD para A se da por $A = U_r \Sigma_r V_r^T$ donde $V_r \in \mathbb{R}^{n \times r}$ tiene r columnas de $V \in \mathbb{R}^{n \times n}$

Algoritmo: $A^{(0)} = A$, $V^{(0)} = I_n$

$$A^{(k+1)} = A^{(k)} U^{(k)}, \quad V^{(k+1)} = V^{(k)} U^{(k)}, \quad k > 0$$

$U^{(k)}$ son matrices de rotación de plano (i, j) es decir es una identidad pero con elementos:

$$U_{ii}^{(k)} = \cos(\theta) \quad U_{ij}^{(k)} = \sin(\theta) \quad \text{suponiendo } i < j \text{ p. ej.}$$

$$U_{ji}^{(k)} = -\sin(\theta) \quad U_{jj}^{(k)} = \cos(\theta) \quad \text{donde: } U^{(k)} = (U_{ij}^{(k)})_{i,j=1}^n$$

La multiplicación $A^{(k)} U^{(k)}$ sólo afecta a 2 columnas de $A^{(k)}$:

$$(a_i^{(k+1)}, a_j^{(k+1)}) = (a_i^{(k)}, a_j^{(k)}) \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix}$$

$$\text{donde } A^{(k)} = \begin{bmatrix} a_1^{(k)} & a_2^{(k)} & \cdots & a_n^{(k)} \end{bmatrix}$$

El ángulo θ se elige:

- 1) Si $a_i^{(k)\top} a_j^{(k)} = 0$ \Rightarrow las columnas son ortogonales
- 2) $\theta \in (-\pi/4, \pi/4)$ tal que $a_i^{(k+1)\top} a_j^{(k+1)} = 0$

para resolver $a_i^{(k+1)\top} a_j^{(k+1)} = 0$ de 2) se calcula

$$\cos \theta = \frac{1}{\sqrt{1+t^2}}, \quad \sin \theta = \tan \theta t \quad \text{donde:}$$

$$t = \frac{\operatorname{signo}(z)}{|z| + \sqrt{1+z^2}}$$

$$z = \frac{\|a_j^{(k)}\|_2^2 - \|a_i^{(k)}\|_2^2}{2a_i^{(k)\top} a_j^{(k)}}$$

Obs. Para la implementación de lo anterior:

+) Usar $\frac{a_i^{(k)\top} a_j^{(k)}}{\|a_i^{(k)}\| \|a_j^{(k)}\|} < \text{Tol}$ para la condición $a_i^{(k)\top} a_j^{(k)} = 0$ con $\text{Tol} \leq 10^{-8}$

+) Si $\text{num} = \|a_j^{(k)}\|_1^2 - \|a_i^{(k)}\|_1^2$, $\text{den} = 2a_i^{(k)\top} a_j^{(k)}$ para evitar problemas de underflow usamos: si $|\text{den}| \leq \epsilon_{m,q} |\text{num}|$ ent. $\theta = 0$ en este caso calculamos $t, z, U^{(k)}$

+) $\begin{pmatrix} a_i^{(k+1)} & a_j^{(k+1)} \\ v_i^{(k+1)} & v_j^{(k+1)} \end{pmatrix} = \begin{pmatrix} a_i^{(k)} & a_j^{(k)} \\ v_i^{(k)} & v_j^{(k)} \end{pmatrix} U^{(k)}$ actualizamos las columnas de A, V

+) Usar un contador "rot" que se incrementa si las columnas son ortogonales
El algoritmo termina si $\text{rot} = \frac{n(n-1)}{2}$ en un sweep:

En los algoritmos tradicionales de Jacobi one-sided las rotaciones se realizan en una secuencia fija llamada "sweep". Cada sweep consiste de $\frac{n(n-1)}{2} = \binom{n}{2}$ rotaciones y en cada sweep se ortogonalizan 2 columnas. El algoritmo iterativo termina si en un sweep todas las columnas son ortogonales

Ejecución de (i,j)

- +) Ordenamiento cíclico por renglones: Un sweep involucra los pares de columnas $(1,2), (1,3), \dots, (1,n), (2,3), (2,4), \dots, (2,n), \dots, (n-1,n)$
 Este ordenamiento se ejecuta de forma secuencial no es posible realizarlo en paralelo y siempre converge si $\theta(\leq \frac{\pi}{4})$ (convergencia uniforme)

Ordenamiento round robin

-) Para ejecución en paralelo de MTOS
-) Genera $\frac{n(n-1)}{2}$ pares de columnas en $n-1$ pasos con $\frac{n}{2}$ procesadores
 $(n$ par, si n es impar se agrega una columna de ceros a la matriz original)

Ej. $n=8, p=4$ (num. de procesadores)

$$\begin{array}{c}
 \text{Paso 1: } \begin{matrix} P_1 & P_2 & P_3 & P_4 \\ 1 & 3 & 5 & 7 \\ 2 & 4 & 6 & 8 \end{matrix} \\
 \text{flechas} \quad \leftarrow \quad \rightarrow \quad \rightarrow \quad \rightarrow \\
 \text{rotar en sentido} \\
 \text{de las manecillas} \quad \dots \quad \text{Paso 7: } \begin{matrix} 1 & 5 & 7 & 8 \\ 3 & 2 & 4 & 6 \end{matrix}
 \end{array}$$

Ventajas/Desventajas

-) No hay garantía teórica de convergencia pero en la práctica funciona
 -) $p \leq n/2$ por lo que no es posible utilizar demasiados procesadores
 -) Se extiende al método de Jacobi por bloques one-sided
- +) Otros: ordenamiento de anillo y odd-even - artículo de Brent-Toumoutchiyan
 En este artículo además se añade en la parte de rotaciones:

Intercambiar columnas a_i con a_j si

$$\|a_i\| < \|a_j\| \text{ para } i < j \text{ y ortogonalizamos posteriormente}$$

Método de Jacobi por bloques one-sided (MTBOS)

Dividimos $A = [A_1 \ A_2 \ A_3 \ \dots \ A_r]$ con n_i : #dor. de A_i $\forall i=1,\dots,r$
tal que $n_1+n_2+\dots+n_r=n$, r rango de A

Típicamente $n_1=n_2=\dots=n_{r-1}=n_0$ de modo que $n=(r-1)n_0+n_r$, $n_r \leq n_0$
 n_0 se elige de acuerdo al cache de memoria disponible

Recordamos el algoritmo MTOS: $A^{(0)}=A$, $V^{(0)}=I_n$
 $A^{(k+1)}=A^{(k)}U^{(k)}$, $V^{(k+1)}=V^{(k)}U^{(k)}$, $k > 0$

Para hacerlo por bloques definimos

$$U^{(k)}$$
 de $n \times n$ matriz de rotación: $U^{(k)} = \begin{pmatrix} I_1 & & & \\ U_{ii}^{(k)} & U_{ij}^{(k)} & & \\ & & I_2 & \\ U_{ji}^{(k)} & & U_{jj}^{(k)} & \\ & & & I_3 \end{pmatrix}$
 Los bloques q no se han identificado en \rightarrow
 son iguales a cero

Con (i,j) elegidos en el paso del algoritmo anterior de acuerdo a una estrategia de pivoteo p.ej. los ordenamientos visto anteriormente q no han
en su secuencia paralelo el algoritmo

La multiplicación $A^{(k)}U^{(k)}$ tiene por objetivo hacer ortogonales las
columnas del bloque i con las columnas del bloque j de $A^{(k)}$

$U_{ii}^{(k)}, U_{jj}^{(k)}$ son matrices cuadradas de orden n_i, n_j respectivamente

I_1 es de tamaño $\sum_{k=1}^{i-1} n_k$, I_2 de $\sum_{k=i+1}^{j-1} n_k$, I_3 de $\sum_{k=j+1}^r n_k$

La matriz ortogonal $\hat{U}^{(k)} = \begin{pmatrix} U_{ii}^{(k)} & U_{ij}^{(k)} \\ U_{ji}^{(k)} & U_{jj}^{(k)} \end{pmatrix}$ de tamaño $n_i + n_j$

se llama submatriz pivote de $U^{(k)}$ para el paso k y se calcula

Diagonalizando la siguiente matriz simétrica positiva definida para el par (i, j)

$$\hat{A}_{ij}^{(k)} = \left[A_i^{(k)} \ A_j^{(k)} \right]^T \left[A_i^{(k)} \ A_j^{(k)} \right] = \begin{pmatrix} A_i^{(k)T} A_i^{(k)} & A_i^{(k)T} A_j^{(k)} \\ A_j^{(k)T} A_i^{(k)} & A_j^{(k)T} A_j^{(k)} \end{pmatrix}$$

e) derivar calcular $\hat{U}^{(k)}$, $\hat{A}_{ij}^{(k)}$ tiene que: $\hat{U}^{(k)T} \hat{A}_{ij}^{(k)} \hat{U}^{(k)} = \hat{A}_{ij}^{(k)}$

$\hat{U}^{(k)}$ divididos por bloques como antes y con $\hat{U}^{(k)}$ se construye $U^{(k)}$ que se aplica a $A^{(k)}, V^{(k)}$ posteriormente

Obs o) Diagonalizar $\hat{A}_{ij}^{(k)}$ equivale a hacer ortogonales el bloque i de $A^{(k)}$ con el bloque j de $A^{(k)}$

- Excepto por una parte del 1º sweep los 2 bloques diagonales de $\hat{A}_{ij}^{(k)}$ serán diagonales, lo que reduce el número de operaciones para calcular $\hat{A}_{ij}^{(k)}$
- El algoritmo MTBOS es: Dado (i, j) en un sólo procesador
 - Calcular $\hat{A}_{ij}^{(k)}$
 - Diagonalizar $\hat{A}_{ij}^{(k)}$ $\rightarrow (A_i^{(k+1)}, A_j^{(k+1)}) = (A_i^{(k)}, A_j^{(k)}) \hat{U}^{(k)}$
 - $(V_i^{(k+1)}, V_j^{(k+1)}) = (V_i^{(k)}, V_j^{(k)}) \hat{U}^{(k)}$

4) Comunicación y transmisión entre procesadores de bloque(s)

El paso computacional más costoso es 3) por la obs. anterior que reduce el costo de 1) y porque 2) se realiza en el cache de la memoria

- recordar que $\hat{A}_{ij}^{(k)}$ es de tamaño $n_i \times n_j$ y se pliegan estos tamaños -

- No hay una prueba teórica de convergencia para el MTBOS pero en la práctica se ha visto que $A^{(k)} \rightarrow U \sum_r V^{(k)} \rightarrow V_r$

- Si se tiene un ordenamiento en paralelo para elegir (i, j) y se tienen P procesadores entonces $r = 2P$, es decir elegir 2 bloques de columnas por c/procesador; al inicio se puede asignar 2 bloques \downarrow 1 consecutivos por procesador, así A (matriz inicial) está distribuida y la

diagonalización de $\hat{A}_{ij}^{(k)}$ si se realiza en un solo procesador. Considerar mover la menor cantidad de bloques de un procesador a otro para no alejtar el método

- o) Una ventaja del MTBOS es la paralelización del cálculo de la SVD de A y realizar en paralelo; pero una desventaja es el tiempo de este método, pues el método de bidiagonalizar en aunque no es igual de preciso es más rápido, para reducir el tiempo del MTBOS ver artículos OTBSA de Oksa, Vajtersic en las referencias