Simple LU and QR based Non-Orthogonal Matrix Joint Diagonalization

Bijan Afsari

bijan@glue.umd.edu

Institute for Systems Research and Department of Applied Mathematics, University of Maryland, College Park

Outline

- Introduction and Motivation
- Orthogonal Joint Diagonlization (OJD)
- Non-Orthogonal Joint Diagonalization(NOJD), Some Existing methods for
- Challenges in NOJD and Contributions
- QR and LU based JD
- Numerical Experiments

Introduction and Motivation

- (Approximate) Joint Diagonalization Problem:
  Given a set of N symmetric $n \times n$ matrices $\{C_i\}_{i=1}^{N}$ find a non-singular matrix B such that all *BC_iB^T are "as diagonal as possible"*
- Call B a joint diagonalizer
- Diagonalization in the congruence sense and not similarity
- Approximate, because a priori we do not believe JD can happen in an exact sense
- Some motivating problems:
  - In the standard ICA problem: joint diagonalization of cumulant matrix slices
  - In the non-stationary BSS: joint diagonalization of correlation matrices at different time instances
  - relation to tensor diagonalization

Orthogonal JD

- Sometimes the joint diagonalizer can assumed to be orthogonal, so find $\Theta \in$SO(n) such that J_1 is minimized:

$$J_1(\Theta) = \sum_{i=1}^{n} \left\| \Theta C_i \Theta^T - \operatorname{diag}(\Theta C_i \Theta^T) \right\|_F^2 \tag{1}$$

- Since SO(n) is a compact manifold $J_1 : \mathrm{SO}(n) \to \mathbb{R}$ achieves a minimum for any set of matrices, so J_1 is a good cost function for orthogonal JD
- Orthogonal JD is an example of optimization on a compact Lie group (SO(n)).
- An efficient algorithm using on orthogonal Jacobi matrices exists for minimization of J_1:

$$\Theta_{kl}(\theta) = \begin{bmatrix} 1 & \downarrow^{k} & 0 & \downarrow^{l} & 0 \\ {}^{k}\longrightarrow & \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 0 & 1 & 0 & 0 \\ {}^{l}\longrightarrow & \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

OJD using Orthogonal Jacobi Matrices(JADE)

- It is easy to solve the one dimensional problem:

$$\theta_{lk} = \arg\min_{\theta} J_1(\Theta_{lk}(\theta))$$

- **Proposition 2**[Cardoso]: If θ_{kl} is such that $\vec{v} = [\cos 2\theta_{kl} \quad \sin 2\theta_{kl}]^T$ is a unit-norm eigen vector corresponding to the larger eigen value of the matrix $G_{kl}^T G_{kl}$ where G_{kl} is an $N \times 2$ matrix defined as $G_{kl}(i,1) = C_i(k,k) - C_i(l,l)$ and $G_{kl}(i,2) = -2C_i(k,l)$ for $1 \leq i \leq N$, then θ_{kl} minimizes $J_1(\Theta_{kl}(\theta))$.

- **Algorithm:**
  1. `set` $\Theta = I$. `set` ϵ.
  2. `set` $\Psi = I$. `for` $1 \leq k < l \leq n$:
     - Find $\theta_{kl} = \arg\min_{\theta} J_1(\Theta_{kl}(\theta))$ from Proposition 2
     - Update the data $\{C_i\}$: $C_i \leftarrow \Theta_{kl}(\theta_{kl}) C_i \Theta_{lk}(\theta_{kl})^T$
     - Update Ψ: $\Psi \leftarrow \Theta_{kl}(\theta_{kl})\Psi$
  3. Update Θ: $\Theta \leftarrow \Psi\Theta$. `if` $\|\Psi - I\|_F > \epsilon$, `then goto` 2, `else end`

- One run of step 2 is called a sweep.

- Computational cost in each sweep: optimization $\mathcal{O}(Nn^2)$, Θ update $\mathcal{O}(n^3)$, data update $\mathcal{O}(Nn^3)$

- This is an efficient algorithm, it is step-size free, the answer will be orthogonal by construction

Non-Orthogonal JD(NOJD)

- OJD is good but not enough in many cases! Hence do Non-Orthogonal JD.
- NOJD is scale and permutation invariant, i.e. if B is a NOJD then $\Pi\Lambda B$ is also a NOJD for any non-singular diagonal Λ and permutation Π $\Leftrightarrow$ the same inherent indeterminacies as in ICA
- OJD is an $\frac{n(n+1)}{2}$ dimensional problem but NOJD is a $n^2 - n$ one $\Rightarrow$ NOJD has $\frac{n^2-n}{2}$ more degrees of freedom
- Q:Can we simply extend J_1 to GL(n)? A: No. Because, $J_1 : \mathrm{GL}(n) \rightarrow \mathbb{R}$ is not scale invariant and as $\|\Lambda\| \downarrow 0$, $J_1(\Lambda B) \downarrow 0$.

Existing methods for NOJD

$$J_1 : \mathrm{GL}(n) \to \mathbb{R}$$
$$J_1(B) = \sum_{i=1}^{n} \left\| BC_iB^T - \mathrm{diag}(BC_iB^T) \right\|_F^2$$

- To avoid reduction by diagonal matrices and the infimum at $B = 0$ use multiplicative updates as:

$$B_{k+1} = (I + \Delta_k)B_k, \quad \mathrm{diag}(\Delta_k) = 0, B_0 = I_{n\times n} \tag{3}$$

  - $n^2 - n$ degrees of freedom
  - Δ_k from a gradient step [Afsari, Krishnaprasad] or approximate minimization of J_1 [Ziehe et al].
  - Spacial case: if Δ_k is (Lower) Upper-triangular then B_k is (Lower) Upper-triangular and $\det(B_k) = 1$ and $\|B_k\| \geq 1$
- Others: Yeroder, Vanderveen, ...
- How about other cost functions?
  - (Pham, Cardoso) $\{C_i\}_{i=1}^N$ positive definite:
  $$J_3(B) = \sum_{i=1}^{N} \log\left(\frac{\det \mathrm{diag}(BC_iB^T)}{\det BC_iB^T}\right)$$
  - Add a penalty term to J_1

Challenges in NOJD and Contributions

- Algorithm, computational cost:
  - One solution: use QR or LU factorization and replace the high-dimensional minimization problem with a sequence of one-dimensional simple problems
- Cost function: Good cost function for NOJD, scale-invariant, simple, congruence-preserving, .. is not known.
  - One answer $J_2 : \mathrm{GL}(n) \to \mathbb{R}$:

$$J_2(B) = \sum_{i=1}^{N} \|C_i - B^{-1}\mathrm{diag}(BC_iB^T)B^{-T}\|_F^2 \tag{4}$$

  - J_2 is scale invariant and $J_1 = J_2$ on SO(n) and they are scaled-version of each other:

$$\frac{J_1(B)}{n\|B\|_F^2} \leq J_2(B) \leq n\|B^{-1}\|_F^2 J_1(B) \tag{5}$$

  - Catch: J_2 includes B^{-1}, it is not congruence-preserving i.e. it is not in terms of only BC_iB^T's

LU and QR based NOJD

- Denote the Lie group of (upper) lower-triangular matrices of dimension n with diagonal elements of 1 by ($\mathcal{U}(n)$) $\mathcal{L}(n)$. These are $\frac{n(n-1)}{2}$ dimensional manifolds.
- $B \in \mathrm{GL}(\mathsf{n})$ accepts the LU factorization:

$$B = \Pi \Lambda L U, \quad L \in \mathcal{L}(n), U \in \mathcal{U}(n) \tag{6}$$

- $B \in \mathrm{GL}(\mathsf{n})$ accepts the QR factorization:

$$B = \Lambda L \Theta, \quad L \in \mathcal{L}(n), \Theta \in \mathrm{SO}(n) \tag{7}$$

- These structures match the indeterminacies of NOJD problem.
- Idea:
  - Iteratively and alternatively find L and U in LU form or L and Θ in the QR form to minimize J_1 and construct the whole answer by multiplication
  - Replace each of the $\frac{n(n-1)}{2}$ dimensional minimization problems by a sequence of $\frac{n(n-1)}{2}$ one dimensional problems via Jacobi orthogonal and triangular matrices:

LU and QR ..

- Triangular Jacobi matrices corresponding to position (i, j):

$$L_{ij}(a) = \begin{array}{cc} & \begin{array}{ccccc} & \downarrow^{j} & & & \end{array} \\ \begin{array}{c} \\ j \longrightarrow \\ \\ i \longrightarrow \\ \\ \end{array} & \begin{bmatrix} 1 & & 0 & & \\ & 1 & \cdots & & \\ 0 & 0 & 1 & \vdots & 0 \\ & a & 0 & 1 & \\ & \cdots & 0 & 0 & 1 \end{bmatrix} \end{array} \qquad U_{ij}(a) = \begin{array}{cc} & \begin{array}{ccccc} & & & \downarrow^{j} & \end{array} \\ \begin{array}{c} \\ i \longrightarrow \\ \\ j \longrightarrow \\ \\ \end{array} & \begin{bmatrix} 1 & 0 & 0 & & \\ & 1 & 0 & a & \\ 0 & 0 & 1 & \vdots & 0 \\ & 0 & 0 & 1 & \\ & \cdots & 0 & 0 & 1 \end{bmatrix} \end{array} \tag{8}$$

- Interestingly, the one dimensional problems: $\min_a J_1(L_{ij}(a))$ and $\min_a J_1(U_{ij}(a))$ can be solved in closed form:
  Proposition 1: If $\hat{a}$ is such that:

$$\hat{a} = -\frac{\sum_{i=1}^{N} C_i(k, [1 : l-1, l+1 : n]) C_i(l, [1 : l-1, l+1 : n])^T}{\sum_{i=1}^{N} \|C_i(k, [1 : l-1, l+1, : n])\|_F^2} \tag{9}$$

  then: with $k < l$, $\hat{a}$ minimizes $J_1(L_{lk}(a))$ and with $k > l$, $\hat{a}$ minimizes $J_1(U_{lk}(a))$. If $\sum_{i=1}^{N} \|C_i(k, [1 : l-1, l+1, : n])\|_F^2 = 0$ set $\hat{a} = 0$, i.e. J_1 can not be reduced by that particular L_{lk} or U_{lk}.

LU and QR ..

Algorithm LUJ1D:

1. set $B = I$. set ϵ.
2. U-phase: set $U = I$. for $1 \leq l < k \leq n$:
   - Find $a_{lk} = \arg\min_a J_1(U_{lk}(a))$ from Proposition 1
   - $C_i \leftarrow U_{lk}(a_{lk}) C_i U_{lk}(a_{lk})^T$ and $U \leftarrow U_{lk}(a_{lk}) U$
3. L-phase (R-Phase): set $L = I$. for $1 \leq l < k \leq n$:
   - Find $a_{kl} = \arg\min_a J_1(L_{kl}(a))$ from Proposition 1
   - Update $C_i \leftarrow L_{kl}(a_{kl}) C_i L_{kl}(a_{kl})^T$ and $L \leftarrow L_{kl}(a_{kl}) L$
4. if $\|LU - I\|_F > \epsilon$ ($\|L\Theta - I\|_F > \epsilon$), then $B \leftarrow LUB$ ($B \leftarrow L\Theta B$) and goto 2, else end

- Computational cost in each sweep: optimization $\mathcal{O}(Nn^3)$, B update $\mathcal{O}(n^3)$, data update $\mathcal{O}(Nn^3)$
- step-size free, easy to implement, no multiplication of large matrices
- In the U-phase replace $a_{lk} = \min_a J_1(U_{lk}(a))$ with $\theta_{lk} = \arg\min_\theta J_1(\Theta_{lk}(\theta))$ and perform the corresponding updates to have a Q-phase and Algorithm QRJ1D

Practical Issues

- Stopping criterion: $\|LU - I\|$ measure how much B is going to change after a sweep. We also can measure how much J_2 has relatively changed after a sweep and stop if this change is small enough.
- Row Balancing: For large n and N, if the row of the large matrix $C = [C_1, ..., C_N]$ are not balanced in their norms then the value for a found from (9) can be very inaccurate. So after every few sweeps we update the data and B as $C_i \leftarrow DC_iD$ and $B \leftarrow DBD$ where D is an appropriate diagonal matrix. For example: $D(k,k) = \frac{1}{\sqrt{\|C(k,:)\|_F}}$

Use of J_2 for NOJD

$$J_2 : \mathrm{GL}(n) \to \mathbb{R}$$

$$J_2(B) = \sum_{i=1}^{N} \left\| C_i - B^{-1}\mathrm{diag}(BC_iB^T)B^{-T} \right\|_F^2 \tag{10}$$

- Replace J_1 with J_2 in LUJ1D and QRJ1D.
- Simple observation: $L_{ij}^{-1}(a) = L_{ij}(-a)$, $U_{ij}^{-1}(a) = U_{ij}(-a)$
- one dimensional problems: $\min_a J_2(L_{ij}(a))$ and $\min_a J_2(U_{ij}(a))$ are root finding of cubic polynomials and can even be solved in closed form:
  Proposition 3: If $k < l$ then: $J_2(L_{lk}(a)) = a_4a^4 + a_3a^3 + a_2a^2 + a_1a + a_0$ and if $k > l$ then $J_2(U_{lk}(a)) = a_4a^4 + a_3a^3 + a_2a^2 + a_1a + a_0$, where:
  $$a_4 = 4\sum_{i=1}^{N} C_i(k,k)^2, \;\; a_3 = 8\sum_{i=1}^{N} C_i(k,k)C_i(k,l), \;\; a_2 = 2\sum_{i=1}^{N} C_i(k,k)^2 + 2C_i(k,l)^2$$
  and:
  $$a_1 = 4\sum_{i=1}^{N} C_i(k,l)C_i(k,k), \quad a_0 = 2\sum_{i=1}^{N} C_i(k,l)^2$$
- Sweep cost for LUJ2D, QRJ2D: optimization $\mathcal{O}(Nn^2)$, B update $\mathcal{O}(n^3)$, data update $\mathcal{O}(Nn^3)$
- Row balancing again useful

Little bit of Geometry

- $\Theta_{ij}(\theta)$,$U_{ij}(a)$ and $L_{ij}(a)$ correspond to one-dimensional subgroups in SO(n),$\mathcal{U}(n)$ and $\mathcal{L}(n)$, respectively. These one-dimensional subgroups also are geodesics in the mentioned manifolds
- The proposed algorithms are basically (combination of) line searches along all possible directions on curved (SO(n)) and flat spaces ($\mathcal{U}(n)$, $\mathcal{L}(n)$)
- Although the directions are fixed and exhaustive since the computational costs are low the algorithms are efficient