Joint diagonalization

July 2019

1 Joint Diagonalization Implementation

The goal is to have an implementation of the model that requires as few computational resources as possible. Therefore, fast alternative algorithms to perform certain calculations have been researched and implemented.

1.1 Joint Diagonalization Algorithms

Simultaneous diagonalization, either approximate or exact, of a set of matrices has been proposed as a solution to statistical identification problems [1]. The approximate joint diagonalization of a set of real m-square symmetrical matrices

$$R = \{R_0, R_1, ..., R_J\}$$

where R_0 is positive definite, is an essential tool in BSS algorithms. The goal is to find the unmixing matrix W, also called the diagonalizer or separation matrix, such that the matrices WR_jW^T are as diagonal as possible for $j = 0, ..., J$.

SOBI includes a whitening step that ensures that $R_0 = I$. The problem consists of computing the W that minimizes some criterion. Let $\mathit{Off}(R) = \sum_{k \neq l} r_{l,k}^2$, where $r_{l,k}$ is the (l,k)th entry of matrix R. Then the solution for W is obtained by minimizing w.r.t. W the criterion

$$C_{jd}(W, R) = \sum_{j=0}^{J} \mathit{Off}(WR_jW^T) \tag{1}$$

Let $||R||_F^2 = tr(RR^T) = \sum_{l,k} r_{l,k}^2$ denote the squared Frobenius norm of R. Then minimizing (7) is equivalent to minimizing

$$C_{lsw}(W, D, R) = \sum_{j=0}^{J} ||R_j - W^T D_j W||_F^2 \tag{2}$$

w.r.t. W and the set of diagonal matrices

$$D = \{D_0, D_1, ...D_J\}$$

$||W||_F$ denotes the Frobenius norm of matrix W and is defined as

$$||W||_F = \sqrt{\sum_i \sum_j |w_{ij}|^2} \tag{3}$$

The diagonalization algorithms iterate over the solution space until convergence has been reached of the criterion drops below some given error threshold.

1.1.1 Jacobi Angles for Joint Diagonalization

Simultaneous diagonalization using Jacobi Angles is an extension of the Jacobi technique: a joint diagonality criterion is iteratively optimized under plane rotations [1].

Note again that simultaneous diagonalization may be obtained by minimizing equation (7) w.r.t. matrix W. The extended Jacobi technique constructs W as a product of plane rotations globally applied to all matrices in R.

Denote the operation $rotate(k,l,c,s)$ the complex rotation of a matrix R along axis (k,l) by complex angles c and s

$$\begin{aligned} R_k &= c * R_k + s * R_l \\ R_l &= c * R_l - s * R_k \end{aligned} \tag{4}$$

with $c, s \in C$ and $|c|^2 + |s|^2 = 1$. For each (k,l) the complex angles c and s should minimize the objective function

$$O(c,s) = \sum_j \mathit{Off}(rotate(R_j,k,l,c,s) * R * rotate(R_j,k,l,c,s)) \tag{5}$$

For a given pair (k,l) of indices, a $3x3$ real symmetric matrix G is defined as

$$G = Real\left(\sum_j h^H(R_j)h(R_j)\right) \tag{6}$$

$$h(R) = [r_{kk} - r_{ll}, r_{kl} + r_{lk}, i(r_{lk} - r_{kl})] \tag{7}$$

Then, under the constraint $|c|^2 + |s|^2 = 1$, the objective function $O(c,s)$ is minimized at

$$c = \sqrt{\frac{x+r}{2r}} \qquad s = \frac{y - iz}{\sqrt{2r(x+r)}} \qquad r = \sqrt{x^2 + y^2 + z^2} \tag{8}$$

where $(x,y,z)^T$ is the eigenvector associated to the largest eigenvalue of G. The algorithm considers for each pair of indices all matrices, and has complexity $O(Jm^2)$ for each iteration.

1.1.2 Fast Frobenius Algorithm

The Fast Frobenius algorithm is an iterative scheme to approximate the solution that minimizes (8). The basic idea is to use the invertibility of the matrix W as a constraint preventing convergence to the zero solution [4]. This invertibility can be enforced by carrying out the update of W in each iteration in multiplicative form as

$$W_{(n+1)} = (I + V_{(n)})W_{(n)} \tag{9}$$

where I denotes the identity matrix, the update matrix V is constrained to have zeros on the main diagonal, and n is the iteration number. The off-diagonal components of V are chosen to minimize the C_{lsw} criterion. To maintain invertibility of W is suffices to ensure invertibility of $(I + V_{(n)})$. This is done using the Levi-Desplanques theorem:

Theorem 1 *If an NxN matrix A is strictly diagonally dominant, then it is invertible.*

An NxN matrix A is strictly diagonally dominant if

$$|a_{ii}| > \sum_{j \neq i} |a_{ij} \qquad \forall i = 1,..,N \tag{10}$$

Note that the diagonal terms of $(I + V_{(n)})$ are equal to one. Therefore, is suffices to ensure that

$$\max_i \sum_{j \neq i} |v_{ij}| = ||V_{(n)}||_\infty < 1 \quad \text{or} \quad V_{(n)} = \frac{\theta}{||V_{(n)}||_F} V_{(n)} \tag{11}$$

For some θ ¡ 1. The entries for matrix $V_{(n)}$ are computed as

$$v_{ij} = \frac{z_{ij} y_{ji} - z_{ii} y_{ij}}{z_{jj} z_{ii} - z_{ij}^2} \quad \text{and} \quad v_{ji} = \frac{z_{ij} y_{ij} - z_{jj} y_{ji}}{z_{jj} z_{ii} - z_{ij}^2} \tag{12}$$

$$z_{ij} = \sum_k^J D_i^k D_j^k \tag{13}$$

$$y_{ij} = \sum_k^J D_j^k \frac{E_{ij}^k + E_{ji}^k}{2} \tag{14}$$

where D^k and E^k contain the diagonal and non-diagonal entries of matrix R_k respectively. The algorithm has complexity $O(Jm^2)$ per iteration.

1.1.3 ACDC Algorithm

Minimizing the criterion C_{lsw} in equation (8) w.r.t W is equivalent to minimizing the following criterion w.r.t mixing matrix $A = W^{-1}$ [3]:

$$C_{lsa}(A, D, R) = \sum_{j=0}^J ||R_j - AD_jA^T||_F^2 \tag{15}$$

The ACDC algorithm alternates between the following minimization schemes:

1. Minimization w.r.t D: the minimization of (21) w.r.t D for fixed A is given by:

$$\hat{D}_j = Diag\{\left[(A^TA) \odot (A^TA)\right]^{-1} diag(A^TR_jA)\} \tag{16}$$

   where $\odot$ denotes the Hadamard's product.

2. Minimization w.r.t A: the minimization of (21) w.r.t A for fixed D is realized by successive minimizations on the columns $a_1, .., a_m$ of A. For each column a_k of A while other columns are fixed, the minimization is comes from:

$$P_k = \sum_j D_j(k) \left[R_j - \sum_{i \neq k} D_j(i) a_i a_i^T \right] \tag{17}$$

   If v is the unit norm eigenvector associated to the largest eigenvalue λ of P_k, then

$$\hat{a}_k = \frac{\sqrt{\lambda} v}{\sqrt{\sum_j D_j^2(k)}} \tag{18}$$

The ACDC algorithm has complexity $O(Jm^3)$ per iteration.

1.1.4 LSB Algorithm

The least squares solution of (8) w.r.t diagonalizer W (renamed to B) can be rewritten as [2]:

$$C_{lsb}(B, D, R) = \sum_{j=0}^{J} ||BR_jB^T - D_j||_F^2 \tag{19}$$

Without constrains on B and D, the criterion is equal to zero for $B = 0$. Therefore, we impose constrictions such that $D_0 = I$ and $B_0 = I$. Minimization w.r.t. D is obtained by:

$$\hat{D}_j = Diag\{diag(BR_jB^T)\} \tag{20}$$

Then D in the criterion C_{lsb} can be eliminated:

$$C_{lsb} = \sum_{j=0}^{J} Off(BR^jB^T) \tag{21}$$

under constraint $Diag\{diag(BR_0B\} = I$. The minimization w.r.t. B is realized by successive minimizations on the rows of B, $b_1, .., b_m$:

$$C_{lsb}(B, R) = \sum_{j=0}^{J} \sum_{L=1}^{m} \sum_{i \neq L}^{m} (b_L^T R_j b_i)^2 \tag{22}$$

The minimum w.r.t. row b_l minimizes

$$Q_l = \sum_j R_j B(l)^T B(l) R_j \tag{23}$$

if R and B are whitened. $B(l)$ is obtained by removing the l-th row from B. The new solution $\hat{b}_l$ is the unit norm eigenvector associated to the smallest eigenvalue of Q_l. The LSB provides a sequence of matrices B_n by changing one row of the current matrix. For each full iteration the complexity is $O(Jm^3)$.

1.2 Joint Diagonalization Computations

Some of the computations in the joint diagonalization can be further optimized by either estimating the solution or solving the closed-form expression directly.

1.2.1 Power Iteration

The ACDC algorithm requires the computation of the largest eigenvalue and its associated eigenvector of some matrix A. Usually, computing the eigensystem is done by eigenvalue decomposition, which includes solving the characteristic polynomial and provides every eigenvalue and eigenvector of the matrix. This allows the largest eigenvalue and its eigenvector to be selected.

In stead of computing the complete eigenvalue decomposition, it can be faster (for large matrices) to perform the power iteration algorithm to approximate only the eigenvector corresponding to the largest eigenvalue. Starting from a random vector v_0, repeat until convergence:

$$v_{i+1} = \frac{Av_k}{||Av_k||} \tag{24}$$

If A has a unique largest eigenvalue and the starting vector has a non-zero component in the direction of the largest eigenvector, then a subsequence v_i converges to the eigenvector associated with the largest eigenvalue.

1.2.2 Rayleigh Quotient

For a given real symmetric matrix A and non-zero vector v, the Rayleigh quotient is defined as

$$RQ(A,v) = \frac{v^T A v}{v^T v} \tag{25}$$

The Rayleigh quotient reaches it maximum value $RQ(A,v) = \lambda_{max}$, the largest eigenvalue of A, when v is the corresponding eigenvector.

1.2.3 Einstein Summation Convention

The Einstein summation convention is a notational convention that implies the summation over indexed terms in a formula. Not only does this achieve notational brevity, but it can be used to combine multiple steps of computation into one.

Since the introduction of the function einsum in the Python package numpy up untill at least numpy version 1.14.0, einsum provides some advantages in memory accessing that allow some computations, like computing the outer product, to be faster than using numpy's built in function. In the joint diagonalization computations, the outer product of two vectors is computed using einsum.

1.3 Cross-correlation

The matrix of cross-correlations of the measured (whitened) signal $X(t)$ at time-lag τ is defined as

$$R(\tau) = \mathbb{E}[X(t)X(t-\tau)] = \int_{-\infty}^{\infty} X(t)\overline{X}(t-\tau)\,\mathrm{d}t \tag{26}$$

1.3.1 Data Whitening

Whitening of data X is performed by finding the transformation matrix U such that $X_{white} = UX$ results in X_{white} having unit diagonal co-variance. A commonly used choice for U is the eigensystem of the co-variance matrix associated with X. The eigenvectors in U rotate X along an axis that ensures the unit co-variance. Alternatively, by directly calculating the singular value decomposition, a factorization of X, we obtain

$$U\Sigma V = X \tag{27}$$

The columns of U and V are the left-singular values and right singular vectors of X respectively. The left singular vectors of X are a set of orthonormal eigenvectors of XX^T, and the right singular vectors are a set of orthonormal eigenvectors of X^TX. Since U and V are both orthonormal, the whitened matrix can be obtained by

$$UV = X_{white} \tag{28}$$

1.3.2 Fourier Transform Convolution

The cross-correlation over all lags (from -nsamples to +nsamples) can be computed using the Fourier transform convolution. For two signals x_1 and x_2 the cross-correlation is defined as

$$xcorr(\tau) = \sum_{t=0}^{|x1|-1} x_1(t)x_2^*(t-\tau+|x_1|-1) \tag{29}$$

where $x_2(t) = 0$ when t is outside the range of x_2. This is equal to computing the convolution of the two signals as $(x_1 * x_2)(\tau - |x_1| + 1)$.

Fourier transform convolve is generally very efficient for computing cross-correlations. However, when the signal is very long and only a few lags are needed, a lot of unnecessary computations are done.

1.3.3 Calculating Cross-correlation Directly

The matrix of cross-correlations $R(\tau)$ of the whitened signal $X(t)$ can be computed directly for each τ. The signal is first shifted by the current τ, and then the expected value of the matrix product is computed. Instead of computing the last two steps separately, these steps are combined into one computation using Einstein notation. When only a few lags are needed, this proves to be the faster method for computing cross-correlation.

References

[1] Jean-François Cardoso and Antoine Souloumiac. Jacobi angles for simultaneous diagonalization. *SIAM J. Mat. Anal. Appl.*, 17(1):161–164, January 1995.

[2] S. Degerine and E. Kane. A Comparative Study of Approximate Joint Diagonalization Algorithms for Blind Source Separation in Presence of Additive Noise. *IEEE Transactions on Signal Processing*, 55(6):3022–3031, 2007.

[3] A. Yeredor. Non-orthogonal joint diagonalization in the least-squares sense with application in blind source separation. *IEEE Transactions on Signal Processing*, 50(7):1545–1553, 2002.

[4] A. Ziehe, P. Laskov, G. Nolte, and K.R. Muller. A Fast Algorithm for Joint Diagonalization with Non-orthogonal Transformations and its Applications to Blind Source Seperation. *Journal of Machine Learning Research*, (5):777–800, 2004.