Tensor Algebra
University Federal do Ceará
PPGETI - 2020.2
Saulo Mendes de Melo

Homework 9

Multidimensional Least-Squares Kronecker Factorization

1 Problem 1

On practice 4 we implement the LS-KronF(Least Square Kronecker Factorization) algorithm, now we will go to implement the MLS-KronF(Multidimensional Least Square Kronecker Factorization) algorithm. Then, Let $\mathbf{X} \approx \mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)} \cdots \otimes \mathbf{A}^{(N)} \in \mathbb{C}^{I_1 I_2 \ldots I_N \times J_1 J_2 \ldots J_N}$ be a matrix composed by Kronecker product of N matrices $\mathbf{A}^{(n)} \in \mathbb{C}^{I_n \times J_n}$, with $n = 1, 2, \ldots, N$. For $N = 3$ and I_n and J_n arbitrary implement the MLS-KronF for that estimate $\mathbf{A}^{(1)}$, $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$ by solving the following problem.

$$(\hat{\mathbf{A}}^{(1)}, \hat{\mathbf{A}}^{(2)}, \hat{\mathbf{A}}^{(3)}) = \min_{\mathbf{A},\mathbf{B}} \|\mathbf{X} \otimes \hat{\mathbf{A}}^{(1)} \otimes \hat{\mathbf{A}}^{(2)} \otimes \hat{\mathbf{A}}^{(3)}\|_F^2$$

Compere the estimate matrices $\mathbf{A}^{(1)}$, $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$ with the original ones. What can you conclude? Explain the results.

Solution

The Listing 1 has the source code of the Multilinear LS-KronF algorithm in Python.

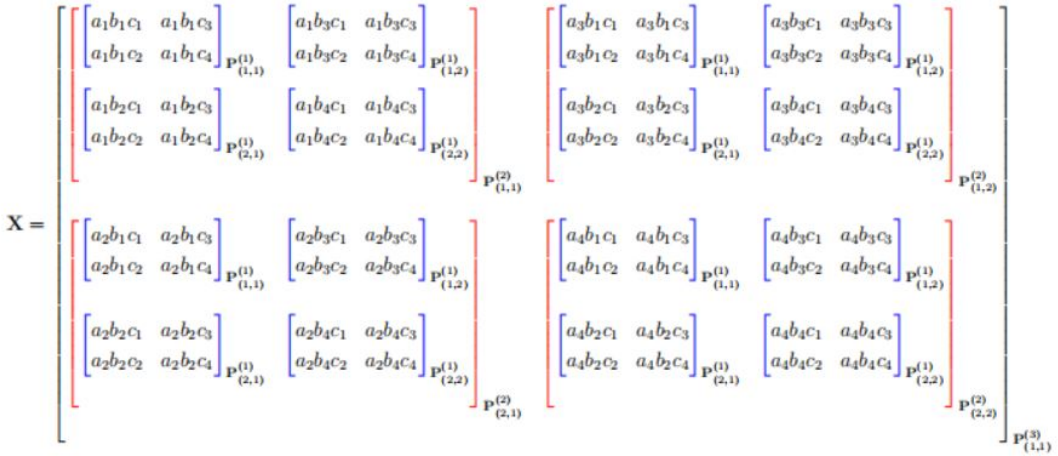
```python
def MLS_KronF(M, P, Q):
    X_reshaped = []
    X_dash = []
    for M_v in np.hsplit(M, P[0]):
        for M_h in np.vsplit(M_v, Q[0]):

            X_dash_flat = []
            for M1_v in np.hsplit(M_h, P[1]):
                for M1_h in np.vsplit(M1_v, Q[1]):
                    X_dash_flat.append(M1_h.T.flatten())

            X_dash.append(np.array(X_dash_flat))

    for X_ in X_dash:
        X_reshaped.append(X_.flatten())

    X_reshaped = np.array(X_reshaped).T

    S, U = HOSVD(tl.fold(X_reshaped, 0, (P[1]*Q[1], P[2]*Q[2], P[0]*Q[0])))

    As = ((S.flat[0]**(1/3))*U[2][:, 0]).reshape(P[0], Q[0]).T

    Bs = ((S.flat[0]**(1/3))*U[0][:, 0]).reshape(P[1], Q[1]).T

    Cs = ((S.flat[0]**(1/3))*U[1][:, 0]).reshape(P[2], Q[2]).T

    return As, Bs, Cs
```

This function returns $\mathbf{A}^{(1)}$, $\mathbf{A}^{(2)}$ and $\mathbf{A}^{(3)}$, and the process can be divided in two main steps: The first part of the method takes care of reshaping the input matrix according to figure 1, creating every $\overline{\mathbf{X}}^{(n,m)}$ to build a reshaped matrix:

$$\overline{\mathbf{X}} = [\text{vec}(\overline{\mathbf{X}}^{(1,1)}); \ldots; \text{vec}(\overline{\mathbf{X}}^{(N,1)}); \text{vec}(\overline{\mathbf{X}}^{(1,1)}); \ldots \text{vec}(\overline{\mathbf{X}}^{(1,M)});]$$

This matrix is then folded along the 3-mode into a tensor $\overline{\mathcal{X}}$.

Figure 1: Matrix reshaping for Multilinear LS-KronF



The second part then consists of applying an HOSVD estimation. Comparing the estimated matrix with the original one, we can see that they are different, but it is easily verifyable that $\hat{\mathbf{A}}^{(1)} \otimes \hat{\mathbf{A}}^{(2)} \otimes \hat{\mathbf{A}}^{(3)} = \mathbf{A}^{(1)} \otimes \mathbf{A}^{(2)} \otimes \mathbf{A}^{(3)}$, which means that this factorization doesn't have a unique solution.

$$\hat{\mathbf{A}}^{(3)} = \begin{bmatrix} 1.70 + 1.47j & -0.62 + -0.54j \\ 0.34 + 1.10j & -0.58 - 0.58j \end{bmatrix}$$

$$\mathbf{A}^{(3)} = \begin{bmatrix} -2.47 + 2.01j & 0.86 + 0.79j \\ -0.44 - 1.57j & 0.80 + 0.85j \end{bmatrix}$$

2 Problem 2

Assuming 1000 Monte Carlo experiments, generate $\mathbf{X}_0 = \mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C} \in \mathbb{C}^{I_1 I_2 I_2 \times J_1 J_2 J_2}$, for randomly chosen $\mathbf{A} \in \mathbb{C}^{I_1 \times J_1}$, $\mathbf{B} \in \mathbb{C}^{I_2 \times J_2}$ and $\mathbf{C} \in \mathbb{C}^{I_3 \times J_3}$, whose elements are drawn from a normal distribution. Let $\mathbf{X} = \mathbf{X}_0 + \alpha \mathbf{V}$ be a noisy version of $\mathbf{X}_0$, where $\mathbf{V}$ is the additive noise term, whose elements are drawn from a normal distribution. The parameter α controls the power (variance) of the noise term, and is defined as a function of the signal to noise ratio (SNR), in dB, as follows

$$\text{SNR}_{\text{dB}} = 10 \log_{10} \left(\frac{\|\mathbf{X}_0\|_F^2}{\|\alpha \mathbf{V}\|_F^2} \right) \tag{1}$$

Assuming the SNR range [0, 5, 10, 15, 20, 25, 30] dB, find the estimates $\hat{\mathbf{A}}$, $\hat{\mathbf{B}}$ and $\hat{\mathbf{C}}$ obtained with the MLS-KRF algorithm for the configurations $I_1 = J_1 = 2$, $I_2 = J_2 = 3$

and $I_3 = J_3 = 4$. Let us define the normalized mean square error (NMSE) measure as follows

$$NMSE(\mathbf{X}_0) = \frac{1}{1000}\sum_{i=1}^{1000}\frac{\|\hat{\mathbf{X}}_0(i) - \mathbf{X}_0(i)\|_F^2}{\|\mathbf{X}_0(i)\|_F^2}, \tag{2}$$

where $\mathbf{X}_0(i)$ e $\hat{\mathbf{X}}_0(i)$ represent the original data matrix and the reconstructed one at the ith experiment, respectively. For each SNR value and configuration, plot the NMSE vs. SNR curve. Discuss the obtained results.

Note: For a given SNR (dB), the parameter to be used in your experiment is determined from equation (1).

Solution

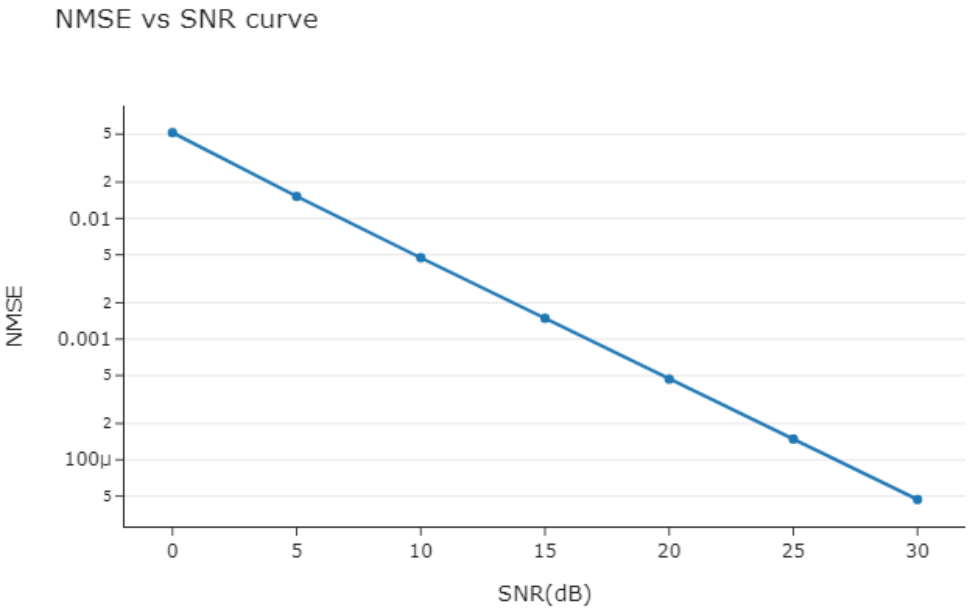
Listing 2 contains the implementation of the experiments.

```
P = [2, 3, 4]
Q = [2, 3, 4]

SNRdb_range = [0, 5, 10, 15, 20, 25, 30]
No_experiments = 1000

NMSE = []

for SNRdb in SNRdb_range:
    exp_errors = [] # list of errors for the experiments

    for i in range(No_experiments):
        A = randn(P[0], Q[0]) + randn(P[0], Q[0])*1j
        B = randn(P[1], Q[1]) + randn(P[1], Q[1])*1j
        C = randn(P[2], Q[2]) + randn(P[2], Q[2])*1j

        X_0 = np.kron(np.kron(A, B), C)

        V = randn(P[0]*P[1]*P[2], Q[0]*Q[1]*Q[2]) + randn(P[0]*P[1]*P[2], Q[0]*Q[1]*Q[2])*1j

        # calculate alpha and X
        alpha = np.sqrt((1/10**(SNRdb/10))*(norm(X_0, 'fro')**2/norm(V, 'fro')**2))
        X = X_0 + alpha*V

        # Estimante via MLS-KRF
        As, Bs, Cs = MLS_KronF(X, P, Q)

        X_0_hat = np.kron(np.kron(As, Bs), Cs)

        exp_errors.append(error(X_0, X_0_hat))

    NMSE.append(np.sum(exp_errors)/No_experiments)
```

Listing 2: Definition of other functions used in this work

As a result of this experiment, the graph 1 show the NMSE for each SNR value. There is a clear logarithmic decreasing trend as the noise becomes smaller.

Figure 2: NMSE vs. SNR Curve



3 Appendix

```python
def error(X_0, X_0_hat):
    return norm(X_0_hat - X_0, 'fro')**2/norm(X_0, 'fro')**2

def Hermitian(X):
    return np.array(np.matrix(X).H)

def HOSVD(X, ml_rank=None):
    U_list = []
    S = X

    for n in range(X.ndim):
        X_n = tl.unfold(X, n)
        Un, _, _ = svd(X_n)

        if ml_rank:
            Un = Un[:, :ml_rank[n]]

        U_list.append(Un)

        S = mode_dot(S, Hermitian(Un), n)

    return S, U_list
```

Listing 3: Experiments