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

Homework 8

Multidimensional Least-Squares Khatri-Rao Factorization

1 Problem 1

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

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

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-KRF algorithm in Python.

```
def MLS_KRF(X, I):
    R = X.shape[1]
    Ir = I
    Ir.reverse()
    N = len(I)
    As, A_list = [], []
    for r in range(R):
        xr = X[:, r]
        Xr = np.reshape(xr, Ir).T
        Sr, Ur = HOSVD(Xr)

        Ar = []
        for n in range(N):
            ur = Ur[N-1-n]
            ar = np.array((Sr.flat[0]**(1/N))*ur[:, 0])
            Ar.append(ar)

        A_list.append(Ar)

    for n in range(N):
        An = []
        for r in range(R):
            An.append(np.array(A_list)[r, n])
        As.append(np.array(An).T)

    return As
```

Listing 1: Function definitions for this work

It provides all estimations $\hat{\mathbf{A}}^{(n)} = [\hat{\mathbf{a}}_1^{(n)}, \hat{\mathbf{a}}_2^{(n)}, \dots, \hat{\mathbf{a}}_R^{(n)}]$ by constructing a rank-1 tensor and computing it's HOSVD $\mathcal{X}_r = \mathcal{S}_r \times_1 \mathbf{U}_r^{(1)} \times_2 \mathbf{U}_r^{(2)} \cdots \times_N \mathbf{U}_r^{(N)}$. Then for $n = 1, \dots, N$, we compute with equation 1 the elements.

$$\hat{\mathbf{a}}_r^{(n)} = \sqrt[N]{(\mathcal{S}_r)_{1,1,\dots,1}} \cdot \mathbf{u}_{r,1}^{(N-n+1)} \tag{1}$$

Declaring a random tensor $\mathcal{A} \in \mathbb{C}^{2\times2\times2}$ and matrix $\mathbf{X} = \mathbf{A}^{(1)} \diamond \mathbf{A}^{(2)} \diamond \mathbf{A}^{(3)}$ and using it as input on the method in Listing 1, we have estimated $\hat{\mathbf{A}}^{(1)}$, $\hat{\mathbf{A}}^{(2)}$ and $\hat{\mathbf{A}}^{(3)}$. Comparing $\mathbf{A}^{(3)}$ and $\hat{\mathbf{A}}^{(3)}$, they are clearly different.

$$\hat{\mathbf{A}}^{(3)} = \begin{bmatrix} 1.23 + 0.48j & -1.1 + 1.71j & -0.97 - 1.82j & 0.36 - 1.15j \\ 1.17 - 0.34j & 0.17 - 0.86j & 1.25 - 0.71j & 0.09 - 1.5j \end{bmatrix}$$

$$\mathbf{A}^{(3)} = \begin{bmatrix} -1.27 + 1.01j & -2.03 + 0.64j & -1.55 - 0.27j & -1.19 + 0.01j \\ -0.36 + 1.45j & 0.71 - 0.58j & 0.16 - 1.09j & -1.44 + 0.37j \end{bmatrix}$$

But we can verify that $\hat{\mathbf{A}}^{(1)} \diamond \hat{\mathbf{A}}^{(2)} \diamond \hat{\mathbf{A}}^{(3)} = \mathbf{A}^{(1)} \diamond \mathbf{A}^{(2)} \diamond \mathbf{A}^{(3)}$, which means that this factorization doesn't have a unique solution.

2 Problem 2

Assuming 1000 Monte Carlo experiments, generate $\mathbf{X}_0 = \mathbf{A} \diamond \mathbf{B} \diamond \mathbf{C} \in \mathbb{C}^{I_1 I_2 I_2 \times R}$, for randomly chosen $\mathbf{A} \in \mathbb{C}^{I_1 \times R}$, $\mathbf{B} \in \mathbb{C}^{I_2 \times R}$ and $\mathbf{C} \in \mathbb{C}^{I_3 \times R}$, with $R = 4$, 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

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

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 $I_1 = 2$, $I_2 = 3$ and $I_3 = 4$. Let us define the normalized mean square error (NMSE) measure as follows

$$NMSE(\mathbf{X}_0) = \frac{1}{1000} \sum_{i=1}^{1000} \frac{\|\hat{\mathbf{X}}_0(i) - \mathbf{X}_0(i)\|_F^2}{\|\mathbf{X}_0(i)\|_F^2}, \tag{3}$$

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).

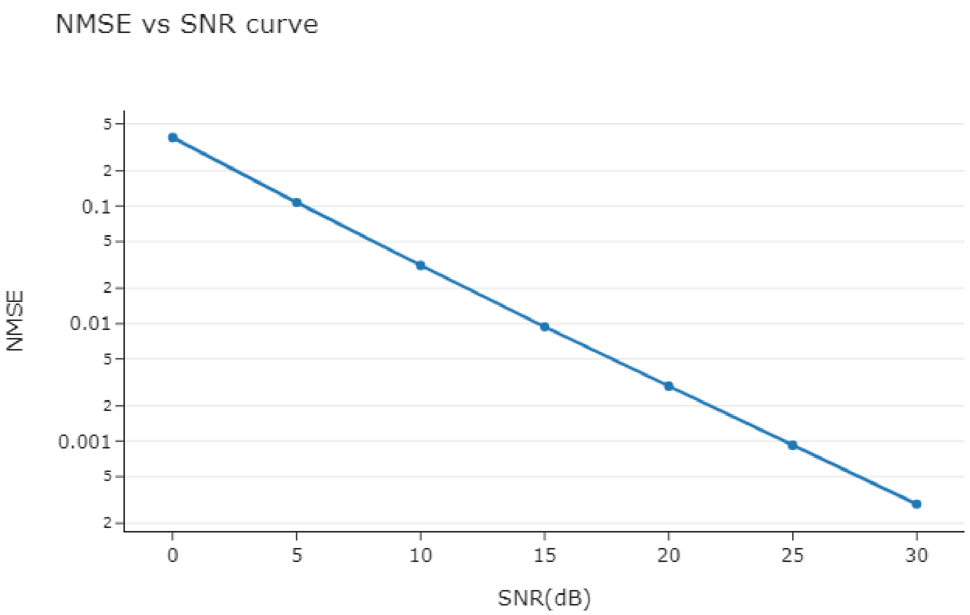
The implementation of this problem can be seen in Listing 2.

```
I1, I2, I3 = 2, 3, 4
R = 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(I1, R) + randn(I1, R)*1j
        B = randn(I2, R) + randn(I2, R)*1j
        C = randn(I3, R) + randn(I3, R)*1j

        X_0 = khatri_rao(khatri_rao(A, B), C)

        V = randn(I1*I2*I3, R) + randn(I1*I2*I3, R)*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
        An = MLS_KRF(X, [I1, I2, I3])

        X_0_hat = khatri_rao(khatri_rao(An[0], An[1]), An[2])

        exp_errors.append(error(X_0, X_0_hat))

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

Listing 2: Function definitions for 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 1: 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

def khatri_rao(A, B):
    assert A.shape[1] == B.shape[1]

    A = A.T
    B = B.T

    result = []

    for Ai in range(0, len(A)):
        for Bi in range(0, len(B)):
            if Ai == Bi:
                result.append(np.kron(A[Ai], B[Bi]))

    return np.array(result).T
```

Listing 3: Function definitions for this work