Universidade Federal do Ceará
TIP8419 - Tensor Algebra
Prof. André A., Prof. Maryam D.

# Homework 12
## Tensor Train Singular Value Decomposition (TTSVD)

Otacílio B. L. Neto (481736)
(minhotmog@alu.ufc.br)
11th March, 2021

**Problem 1.** We will go to implement the TT-SVD (Tensor Train Singular Value Decomposition) algorithm. The TT Decomposition of a $N$-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, with TT-Ranks $(R_1, R_2, \cdots, R_{N-1})$, is a decomposition of the tensor $\mathcal{X}$ into a train of tensors as

$$\mathcal{X} = \mathbf{G}_1 \times_2^1 \mathcal{G}_2 \times_3^1 \mathcal{G}_3 \times_4^1 \cdots \times_{N-1}^1 \mathcal{G}_{N-1} \times_N^1 \mathbf{G}_N.$$

For a 4-order tensor $\mathcal{X} \in \mathbb{R}^{5\times5\times5\times5}$, implement the TT-SVD algorithm to estimate the TT-cores $\mathbf{G}_1 \in \mathbb{C}^{5\times R_1}$, $\mathcal{G}_2 \in \mathbb{C}^{R_1\times5\times R_2}$, $\mathcal{G}_3 \in \mathbb{C}^{R_2\times5\times R_3}$, and $\mathbf{G}_4 \in \mathbb{C}^{R_3\times5}$ that forms the TT Decomposition of $\mathcal{X}$, for given ranks $(R_1, R_2, R_3)$. For randomly generated $(\mathbf{G}_1, \mathcal{G}_2, \mathcal{G}_3, \mathbf{G}_4)$, compare the original tensor $\mathcal{X}$ with the one reconstructed by the TT-SVD.

**Solution.**
The function is implemented in Julia (v1.4), and is shown in Listing 1. Regarding the present implementation, lines 2–14 define an auxiliary function to compute the tensor train products. The Train Tensor Singular Value Decomposition (TT-SVD) algorithm is then implemented in lines 17–237, following the pseudocode provided in the course material.

```
1  # ==== FUNCTIONS ====
2  # The Tensor-Train product function
3  function ×¹(G...)
4      # Auxiliary variables with dimension/rank information
5      IR  = [size(G[1]); [size(G[i])[2:end] for i=2:length(G)]...]
6      I,R = ([IR[i][1] for i=1:length(G)], [IR[i][2] for i=1:length(G)-1])
7
8      # Tensor-train product loop
9      𝒳 = zeros(I...)
10     for i in CartesianIndices(𝒳), r in CartesianIndices((R[1],R[2],R[3]))
11         𝒳[i] += G[1][i[1],r[1]]*G[2][r[1],i[2],r[2]]*G[3][r[2],i[3],r[3]]*G[4][r[3],i[4]]
12     end
13     return 𝒳     # Returns the resulting tensor
14 end
15
16 # The Tensor-Train Singular Value Decomposition (TTSVD) algorithm
17 function TTSVD(𝒳)
18     I = size(𝒳)                                        # Auxiliary variable
19                                                        #
20     X¹ = reshape(𝒳, (I[1],Π(I[2:4])))                  # Unfolds the tensor [𝒳]₁ ∈ ℝ(I₁,I₂I₃I₄)
21     U,Σ,V = svd(X¹);     Σ = Diagonal(Σ);              # Computes the SVD [𝒳]₁ = UΣVᴴ
22     R₁ = sum(diag(Σ) .> 1e-6)                          # Gets the number of nonzero singular-values
23     U₁ = U[:,1:R₁];  V₁ = (Σ*(V)ᴴ)[1:R₁,:]             # Truncates pair (U,V) given R₁
24     G₁ = U₁                                            # Stores the 1st factor G₁ ∈ ℝ(I₁,R₁)
25                                                        #
26     X² = reshape(V₁, (R₁*I[2], Π(I[3:4])))             # Unfolds the matrix X² = [V₁]₁ ∈ ℝ(R₁I₂,I₃I₄)
27     U,Σ,V = svd(X²);     Σ = Diagonal(Σ);              # Computes the SVD X² = UΣVᴴ
28     R₂ = sum(diag(Σ) .> 1e-6)                          # Gets the number of nonzero singular-values
29     U₂ = U[:,1:R₂];  V₂ = (Σ*(V)ᴴ)[1:R₂,:]             # Truncates pair (U,V) given R₂
30     𝒢₂ = reshape(U₂, (R₁, I[2], R₂))                   # Reshapes the 2nd factor 𝒢₂ ∈ ℝ(R₁,I₂,R₂)
31                                                        #
32     X³ = reshape(V₂, (R₂*I[3], I[4]))                  # Unfolds the matrix X³ = [V₂]₁ ∈ ℝ(R₂I₃,I₄)
33     U,Σ,V = svd(X³);     Σ = Diagonal(Σ);              # Computes the SVD X³ = UΣVᴴ
34     R₃ = sum(diag(Σ) .> 1e-6)                          # Gets the number of nonzero singular-values
35     U₃ = U[:,1:R₃];  V₃ = (Σ*(V)ᴴ)[1:R₃,:]             # Truncates pair (U,V) given R₃
36     𝒢₃ = reshape(U₃, (R₂, I[3], R₃))                   # Reshapes the 3rd factor 𝒢₃ ∈ ℝ(R₂,I₃,R₃)
37     G₄ = V₃[1:R₃,:]                                    # Stores the 4th factor G₄ ∈ ℝ(R₃,I₄)
38
39     return (G₁, 𝒢₂, 𝒢₃, G₄)                            # Returns the decomposition factors
40 end
```

Listing 1: The Tensor-Train Singular Value Decomposition (TTSVD) algorithm for Problem 1.

The function $\times^1(\cdot)$ implements the element-wise TT decomposition through the formula

$$x_{i_1,i_2,\cdots,i_N} = \sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2}\sum_{r_3=1}^{R_3}\cdots\sum_{r_{N-2}=1}^{R_{N-2}}\sum_{r_{N-1}=1}^{R_{N-1}} \mathbf{G}_{1_{i_1,r_1}}\mathcal{G}_{2_{r_1,i_2,r_2}}\mathcal{G}_{3_{r_2,i_3,r_3}}\cdots\mathcal{G}_{N-1_{r_{N-2},i_{N-1},r_{N-1}}}\mathbf{G}_{N_{r_N,i_{N-1}}}.$$

The TT-SVD algorithm follows the procedure described by the tensor network shown in Fig. 1

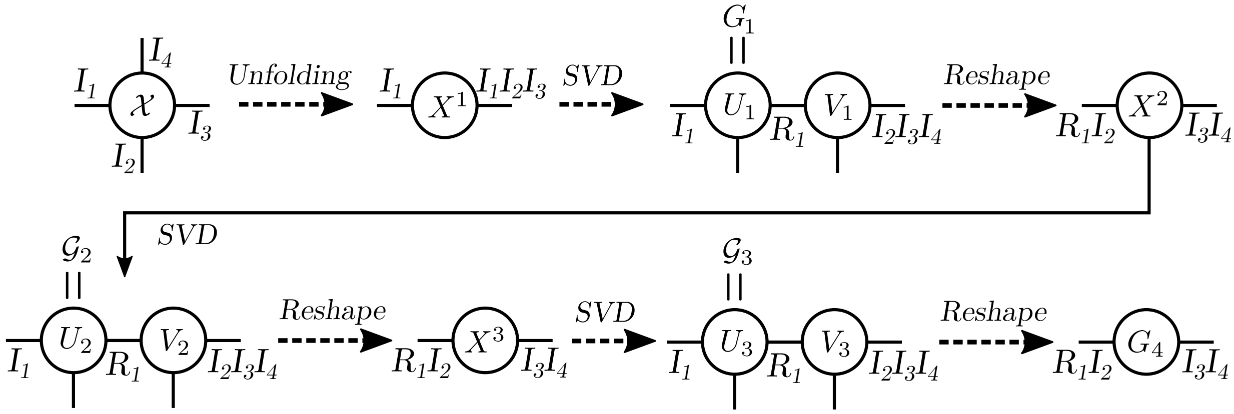


Figure 1: Illustration of the TT-SVD algorithm using a Tensor Network framework.

The implementation is tested through the script presented at Listing 2. The execution of the assertion tests at lines 10 and 18 do not result in any failed assertion, thus indicating that the function outputs the expected results for the given tests. Specifically, executing lines 11 and 19 shows that, on average, $\|\mathcal{X} - \hat{\mathcal{X}}\|_F^2 \approx 1.2 \cdot 10^{-26}$ for $\hat{\mathcal{X}} = \hat{\mathbf{G}}_1 \times_2^1 \hat{\mathcal{G}}_2 \times_3^1 \hat{\mathcal{G}}_3 \times_4^1 \hat{\mathbf{G}}_4$ with all TT-cores obtained from the TT-SVD. Thus, it is possible to conclude that factors $(\hat{\mathbf{G}}_1, \hat{\mathcal{G}}_2, \hat{\mathcal{G}}_3, \hat{\mathbf{G}}_4)$ comprise a valid decomposition of the original tensor $\mathcal{X}$.

```
1  # ==== SCRIPT ====
2  # -- Case 1: Generating 𝒳 from factors (G1, 𝒢2, 𝒢3, G4) --
3  # Randomly generates the TT factors and computes the tensor 𝒳 = G1 ×1 𝒢2 ×1 𝒢3 ×1 G4
4  R = [4 3 4];      # Ranks used for the decomposition
5  G1 = randn(5,R[1]); 𝒢2 = randn(R[1],5,R[2]); 𝒢3 = randn(R[2],5,R[3]); G4 = randn(R[3],5);
6  𝒳 = ×1(G1, 𝒢2, 𝒢3, G4);
7
8  G1_, 𝒢2_, 𝒢3_, G4_ = TTSVD(𝒳);       # Estimates the factors (G1, 𝒢2, 𝒢3, G4)
9
10 @assert 𝒳 ≈ ×1(G1_, 𝒢2_, 𝒢3_, G4_)
11 @show norm(𝒳 - ×1(G1_, 𝒢2_, 𝒢3_, G4_))^2
12
13 # -- Case 2: Directly generating 𝒳 randomly --
14 𝒳 = randn(5, 5, 5, 5);
15
16 G1_, 𝒢2_, 𝒢3_, G4_ = TTSVD(𝒳);       # Estimates the factors (G1, 𝒢2, 𝒢3, G4)
17
18 @assert 𝒳 ≈ ×1(G1_, 𝒢2_, 𝒢3_, G4_)
19 @show norm(𝒳 - ×1(G1_, 𝒢2_, 𝒢3_, G4_))^2
20
21 # ==== ====
```

Listing 2: Script used to test the TTSVD algorithm implementation.

Moreover, we verified that the resulting estimated tensors $(\hat{\mathbf{G}}_1, \hat{\mathcal{G}}_2, \hat{\mathcal{G}}_3, \hat{\mathbf{G}}_4)$ are not actually good approximations of the original factors used to generate $\mathcal{X}$. In fact, we have average distances of

$$\|\mathbf{G}_1 - \hat{\mathbf{G}}_1\|_F^2 = 25.1, \quad \|\mathcal{G}_2 - \hat{\mathcal{G}}_2\|_F^2 = 61.7, \quad \|\mathcal{G}_3 - \hat{\mathcal{G}}_3\|_F^2 = 62.6, \quad \|\mathbf{G}_4 - \hat{\mathbf{G}}_4\|_F^2 = 3 \cdot 10^4.$$

This result is expected, since the decomposition is not unique: is always possible to replace a pair of successive TT-cores $\mathcal{G}_j$ and $\mathcal{G}_{j+1}$ by $\tilde{\mathcal{G}}_j = \mathcal{G}_j \times_3^1 \mathbf{M}_j^{-1}$ and $\tilde{\mathcal{G}}_{j+1} = \mathbf{M}_j \times_2^1 \mathcal{G}_{j+1}$ without changing the resulting tensor $\mathcal{X}$.

□