Tensor-Train Decomposition

Yuke Wang

Rice University

March 11, 2026

What is a Tensor?

A tensor is a multidimensional array.

- Scalar: order 0
- Vector: order 1
- Matrix: order 2
- Tensor: order ≥ 3

General tensor:

$$\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$$

Examples:

- Images: $H \times W \times C$
- Videos: $H \times W \times C \times T$
- Scientific simulations
- Neural network parameters

Curse of Dimensionality

If all dimensions have size n:

$$\mathcal{X} \in \mathbb{R}^{n^d}$$

Storage grows exponentially:

$$\text{Storage} = n^d$$

Example:

$$20^8 = 2.56 \times 10^{10}$$

numbers.
At 8 bytes per number:

$$\approx 200 \text{ GB}$$

Goal: represent high-dimensional tensors efficiently.

Idea: Tensor Decomposition

Tensor decompositions approximate tensors using low-rank structure.
Common formats:

- CP decomposition
- Tucker decomposition
- Tensor-Train (TT) decomposition

Tensor-Train represents tensors as a chain of small cores:

$$\mathcal{X}(i_1, i_2, \ldots, i_d) = G_1(i_1)G_2(i_2)\ldots G_d(i_d)$$

Intuition: 2D Matrix Example

A matrix is a 2D tensor.

$$X = \begin{bmatrix} 2 & 4 & 6 \\ 1 & 2 & 3 \end{bmatrix}$$

Notice:

$$[1, 2, 3] = \frac{1}{2}[2, 4, 6]$$

Thus the matrix has rank 1.

Matrix Low-Rank Factorization

The matrix can be written as:

$$X = ab^T$$

where

$$a = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Then

$$X(i,j) = a_i b_j$$

This is the Tensor-Train representation for a 2D tensor.

Tensor-Train Representation

For a tensor

$$\mathcal{X}(i_1, i_2, \ldots, i_d)$$

TT decomposition:

$$\mathcal{X}(i_1, \ldots, i_d) = G_1(i_1)G_2(i_2)\ldots G_d(i_d)$$

Each core:

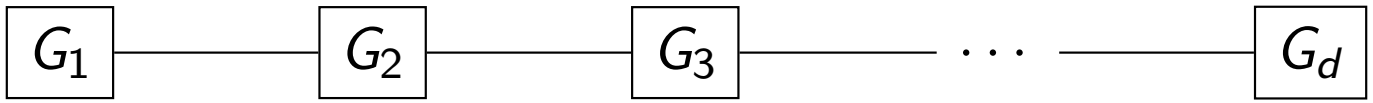
$$G_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$$

with

$$r_0 = r_d = 1$$

r_k are called TT-ranks.

Tensor-Train Structure



Each node is a TT core.

High-Dimensional Tensor Example

Consider a 4D tensor

$$\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times n_4}$$

Goal:

$$\mathcal{X}(i_1, i_2, i_3, i_4) = G_1(i_1) G_2(i_2) G_3(i_3) G_4(i_4)$$

We compute these cores using the TT-SVD algorithm.

Step 1: First Unfolding

Reshape tensor into matrix:

$$X_{(1)} \in \mathbb{R}^{n_1 \times (n_2 n_3 n_4)}$$

Apply SVD:

$$X_{(1)} = U_1 S_1 V_1^T$$

Truncate to rank r_1.

Step 2: First TT Core

First core:

$$G_1 = reshape(U_1)$$

Shape:

$$G_1 \in \mathbb{R}^{1 \times n_1 \times r_1}$$

Remaining tensor:

$$T_2 = S_1 V_1^T$$

Reshape:

$$T_2 \in \mathbb{R}^{r_1 \times n_2 \times n_3 \times n_4}$$

Step 3: Second Decomposition

Reshape:

$$X_{(2)} \in \mathbb{R}^{(r_1 n_2) \times (n_3 n_4)}$$

SVD:

$$X_{(2)} = U_2 S_2 V_2^T$$

Extract core:

$$G_2 \in \mathbb{R}^{r_1 \times n_2 \times r_2}$$

Continue Sequentially

Third unfolding:

$$X_{(3)} \in \mathbb{R}^{(r_2 n_3) \times n_4}$$

SVD:

$$X_{(3)} = U_3 S_3 V_3^T$$

Extract

$$G_3 \in \mathbb{R}^{r_2 \times n_3 \times r_3}$$

Final core:

$$G_4 \in \mathbb{R}^{r_3 \times n_4 \times 1}$$

Final Tensor Train

The tensor becomes

$$\mathcal{X}(i_1, i_2, i_3, i_4) = G_1(i_1)G_2(i_2)G_3(i_3)G_4(i_4)$$

Storage cost:

$$\sum_{k=1}^{d} r_{k-1} n_k r_k$$

instead of

$$\prod_{k=1}^{d} n_k$$

Compression Example

Tensor:

$$20 \times 20 \times 20 \times 20 \times 20$$

Full storage:

$$20^5 = 3,200,000$$

Assume TT-rank $r = 10$.
TT storage:

$$5 \times 20 \times 10^2 = 10,000$$

Compression:

$$320\times$$

Indexing in Tensor-Train

Tensor-Train computes tensor entries by selecting slices from TT cores. For a tensor:

$$\mathcal{X}(i_1, i_2, \ldots, i_d) = G_1(i_1)G_2(i_2)\ldots G_d(i_d)$$

Each index selects a matrix slice:

$$G_k(i_k) \in \mathbb{R}^{r_{k-1} \times r_k}$$

The tensor value is obtained by multiplying these slices sequentially.

Indexing in the 2D Matrix Case

Matrix:

$$X \in \mathbb{R}^{n_1 \times n_2}$$

TT representation:

$$X(i,j) = G_1(i)G_2(j)$$

Core sizes:

$$G_1 \in \mathbb{R}^{1 \times n_1 \times r}, \quad G_2 \in \mathbb{R}^{r \times n_2 \times 1}$$

Indexing i selects a row slice $G_1(i) \in \mathbb{R}^{1 \times r}$.
Indexing j selects a column slice $G_2(j) \in \mathbb{R}^{r \times 1}$.

Computing a Single Matrix Entry

To compute element $X(i,j)$:

$$X(i,j) = G_1(i)G_2(j)$$

Example with rank $r = 2$:

$$G_1(i) = [g_{i1}, g_{i2}], \quad G_2(j) = \begin{bmatrix} h_{1j} \\ h_{2j} \end{bmatrix}$$

Then

$$X(i,j) = g_{i1}h_{1j} + g_{i2}h_{2j}$$

Computing an Entire Row

To compute row i:

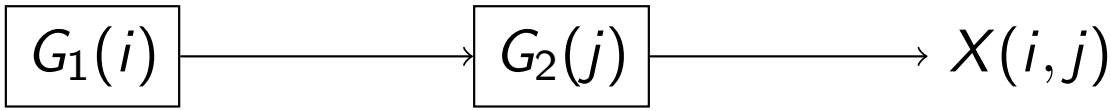
$$X(i,:) = G_1(i)G_2$$

Interpretation:

- $G_1(i)$ is a row vector
- G_2 contains column slices
- Multiplying $G_1(i)$ with G_2 produces the entire row

Cost: $1 \times r$ times $r \times n_2 = 1 \times n_2$

Slice Multiplication Visualization



Applications

Tensor-Train is widely used in:

- Neural network compression
- Large embedding layers
- Scientific computing
- Quantum physics
- High-dimensional PDE solvers

Key Takeaways

- Tensor-Train decomposes tensors into small cores.
- Storage reduces from

$$O(n^d) \to O(dnr^2)$$

- 2D case reduces to matrix low-rank factorization.
- Computed using sequential SVD (TT-SVD).
- Indexing works via selecting slices and multiplying sequentially.