Towards Mastering Tensor Networks: A Comprehensive Guide

Beheshteh T. Rakhshan
Guillaume Rabusseau

Contents

1 Tensor Networks Basics **5**
- 1.1 What are Tensor Networks? . . . 5
- 1.2 Inner Product, Outer Product, Trace and Norm . . . 8
- 1.3 Copy Tensors (Hyperedges) . . . 10
- 1.4 Matrix Factorization in Tensor Networks . . . 12

2 Operations on Tensors **13**
- 2.1 Permute and Reshape Tensors . . . 13
- 2.2 Products . . . 14
- 2.3 Useful TN results . . . 19

3 Tensor Decompositions **20**
- 3.1 CANDECOMP/PARAFAC (CP) Decomposition . . . 20
- 3.2 Tucker Decomposition . . . 22
- 3.3 Tensor Train (TT) Decomposition . . . 23
- 3.4 Efficient Operations in TT Format. . . . 26
- 3.5 Other decompositions: Tensor Ring, PEPS & Hierarchical Tucker . . . 27

4 Computing Gradients with Tensor Networks **28**
- 4.1 Jacobians . . . 28
- 4.2 Applications . . . 31

5 Probability Distributions and Random Vectors **33**
- 5.1 Probability Distributions and Tensor Decompositions . . . 33
  - 5.1.1 Probability Distributions as Tensors . . . 33
  - 5.1.2 Tensor Train Parameterization of Probability Distributions . . . 34
- 5.2 Computing Expectations of Random Tensor Networks . . . 35

$\div$

Notations

$\boldsymbol{\mathcal{A}}$	General tensor containing $n \geq 0$ modes
$\boldsymbol{\mathcal{A}} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$	A 3rd-order tensor $\boldsymbol{\mathcal{A}}$ with shape (d_1, d_2, d_3)
$\boldsymbol{\mathcal{A}}_{i,j,k}$	The (i, j, k)'th element of a 3rd-order tensor $\boldsymbol{\mathcal{A}}$
$\boldsymbol{\mathcal{A}}_{i,:,k}$	Mode-2 fiber of a 3rd-order tensor $\boldsymbol{\mathcal{A}}$, equivalent to a vector $v \in \mathbb{R}^{d_2}$
$\boldsymbol{\mathcal{A}}_{(2)}$	Mode-2 flattening of a (3rd-order) tensor $\boldsymbol{\mathcal{A}}$, equivalent to a matrix $M \in \mathbb{R}^{d_2 \times d_1 d_3}$
$\boldsymbol{\mathcal{A}} \circ \boldsymbol{\mathcal{B}}$	The tensor product (equiv. outer product) of tensors $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$
$\boldsymbol{\mathcal{A}} \simeq \boldsymbol{\mathcal{B}}$	An isomorphism between two tensors $\boldsymbol{\mathcal{A}}$ with $\boldsymbol{\mathcal{B}}$ with compatible shapes
$\mathrm{TN}(G, R)$	Space of all tensors expressable as a tensor network with graph G and rank function R
$\mathrm{TN}(G, R, \mathcal{G})$	A tensor network with graph G, rank function R, and core assignment function $\mathcal{G}$
$\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}} \rangle$	Inner product between tensors $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$
$\left\| \boldsymbol{\mathcal{A}} \right\|_F$	Frobenius norm of the tensor $\boldsymbol{\mathcal{A}}$
$\boldsymbol{\mathcal{A}} \otimes \boldsymbol{\mathcal{B}}$	The Kronecker product of tensors $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$
$\mathbf{A}$	Matrix; A rank-2 tensor better: (tensor of order 2)
$\mathbf{A}_{ij}$	The ij-th element of the matrix $\mathbf{A}$
$\mathbf{A}^{-1}$	The inverse matrix of matrix $\mathbf{A}$
$\mathbf{A}^{\mathsf{T}}$	The transposed matrix of matrix $\mathbf{A}$
$\mathbf{I}$	The identity matrix
$\mathbf{A} \odot \mathbf{B}$	The Khatri-Rao product of matrices $\mathbf{A}$ and $\mathbf{B}$
$\mathbf{A} * \mathbf{B}$	The Hadamard product of matrices $\mathbf{A}$ and $\mathbf{B}$
$\mathbf{a}$	Column vector; A rank-1 tensor
$\mathbf{a}_i$	The i-th element of vector $\mathbf{a}$
$\mathrm{vec}(\mathbf{A})$	Column vector obtained by concatenating the columns of the matrix $\mathbf{A}$
a	Scalar; A rank-0 tensor

Introduction

Tensor methods have gained prominence as an effective approach for managing high-dimensional data. These methods excel at capturing complex structures in multi-modal datasets and have demonstrated success across various domains, including quantum physics, neuroimaging, signal processing, machine learning, financial analysis, biology, and many more. To talk about tensors, we should start with the definition of a tensor. We define an N-th order tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ to be a collection of indexed coefficients $\boldsymbol{\mathcal{T}}_{i_1,\ldots,i_N} \in \mathbb{R}$, referred to as the *elements* of $\boldsymbol{\mathcal{T}}$, where each index i_j is associated with the j-th *dimension* of $\boldsymbol{\mathcal{T}}$ and varies over the set $[d_j] = \{1, 2, \cdots, d_j\}$. N-th order tensors are a natural generalization of vectors and matrices (corresponding to the cases of $N = 1$ and $N = 2$, respectively), where each axis represents a mode. The tuple $d = (d_1, \cdots, d_N)$ is referred to as the *shape* of $\boldsymbol{\mathcal{T}}$, with $d_j \in [d_j]$ referred to as the *dimension* of the j-th mode of $\boldsymbol{\mathcal{T}}$. The collection of all tensors with a given shape d form a vector space of dimension $D = \prod_{j=1}^{N} d_j$, where addition of tensors and multiplication by scalars is defined elementwise as $(\boldsymbol{\mathcal{T}} + \boldsymbol{\mathcal{T}}')_{i_1,\ldots,i_N} = \boldsymbol{\mathcal{T}}_{i_1,\ldots,i_N} + \boldsymbol{\mathcal{T}}'_{i_1,\ldots,i_N}$ and $(c\boldsymbol{\mathcal{T}})_{i_1,\ldots,i_N} = c\boldsymbol{\mathcal{T}}_{i_1,\ldots,i_N}$. This vector space is endowed with the inner product $\left\langle \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{T}}' \right\rangle := \sum_{i_1=1}^{d_1} \cdots \sum_{i_N=1}^{d_N} \boldsymbol{\mathcal{T}}_{i_1,\ldots,i_N} \boldsymbol{\mathcal{T}}'_{i_1,\ldots,i_N}$, as well as the p-norms $\|\boldsymbol{\mathcal{T}}\|_p := \left(\sum_{i_1=1}^{d_1} \cdots \sum_{i_N=1}^{d_N} |\boldsymbol{\mathcal{T}}_{i_1,\ldots,i_N}|^p \right)^{1/p}$ for any $p \geq 1$. We will refer to the case of $p = 2$ as the *Frobenius norm*, denoted by $\|\boldsymbol{\mathcal{T}}\|_F = \|\boldsymbol{\mathcal{T}}\|_2 = \sqrt{\langle \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{T}} \rangle}$

However, as we can see, by increasing the dimensionality of tensors, their representation and manipulation become significantly more complex, and working with tons of indices becomes frustrating. To remedy this problem, tensor networks offer a powerful solution by providing both a compact visual language and an efficient computational framework for handling high-dimensional tensors. Their graphical notations not only simplify complex tensor operations but also enhance interpretability, making tensor algebra more accessible and intuitive. Although widely used in quantum physics, tensor network diagrams remain largely underutilized in mathematically focused fields like deep learning-despite the central role of high-dimensional tensors in these areas. This shows the need for a clear, easy-to-follow guide aimed at people in technical fields, not just those in quantum computing. Our goal is to offer a self-contained, comprehensive guide to tensor networks, providing an accessible set of tools for working with high-dimensional tensors.

Organization

This manuscript is organized into five main chapters.

- Chapter 1 introduces the fundamentals of tensors and their graphical representation. We explore how tensors are depicted and organized using graphical notation, along with key terminology and definitions. The chapter then guides the reader through essential matrix operations and their generalizations to tensors and demonstrates how these operations can be visualized using tensor network diagrams. Finally, more advanced topics—such as copy tensors and matrix factorizations with their graphical notations are introduced toward the end of the chapter.

- Chapter 2 focuses on various operations involving tensors. It begins by introducing tensor fibers and slices, followed by a formal treatment of reshaping tensors into matrices and vectors using tensor network diagrams. The chapter then presents several types of tensor products—including the Kronecker, Khatri-Rao, and Hadamard products—along with their corresponding representations in tensor networks. Readers are also introduced to useful identities associated with these products.At the end of the chapter, we introduce useful tensor network results by defining concepts related to the rank of tensor matricizations.

- Chapter 3 explores tensor decompositions. As mentioned earlier, working with high-order tensors can be challenging, as the number of parameters grows exponentially with the tensor's order. Similar to how singular value decomposition (SVD) simplifies matrix operations, several decomposition methods break tensors into smaller, more manageable components. Among these, the CANDECOMP/PARAFAC (CP) and Tucker decompositions are two of the most widely used techniques, offering compressed representations of high-dimensional tensors. However, the Tucker decomposition still suffers from exponential parameter growth, while the CP decomposition, though more scalable, involves an NP-hard problem when computing the rank. The Tensor Train (TT) decomposition overcomes these challenges by scaling linearly with tensor order and supporting efficient, stable numerical algorithms. Furthermore, the concept of matrix rank can be generalized to tensors in various ways, each corresponding to a different decomposition model. The chapter begins with an introduction to CP decomposition, followed by a detailed walkthrough of various tensor factorizations, including their mathematical formulations

and graphical representations. Finally, we focus on performing efficient operations in the Tensor Train format and introduce more advanced decomposition techniques.

- Chapter 4 presents an intuitive and elegant approach to handling high-order derivatives using tensor network graphical notation. We begin by introducing the classical definitions of gradients and Jacobians, along with their representations in tensor network diagrams. This is followed by a revisiting of classical derivative identities and gradient computations within the tensor network framework. Finally, we explore two of the most common applications of tensor network-based gradients.
- Chapter 5 delves into one of the most powerful applications of tensor networks—their ability to efficiently represent and compute high-dimensional probability distributions. We begin by demonstrating how tensor network diagrams can be used to encode complex probability distributions and compute their marginals in a structured and visually intuitive way. The chapter then shifts focus to the expectations of random tensors, introducing how these expectations can be expressed and manipulated using tensor network graphical notation. This provides a unified and elegant framework for probabilistic reasoning in high-dimensional spaces.

Target Audience

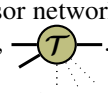
This manuscript is designed as an accessible resource for both academic and industry audiences. It spans a wide range of fields—including mathematics, statistics, physics, machine learning, data science, and beyond—making it a versatile reference for anyone working with high-dimensional data and tensor-based methods. A foundational understanding of first-year undergraduate mathematics is sufficient to grasp the operations discussed. To maintain clarity and ease of interpretation, we primarily illustrate concepts using third-order tensors. The primary goal of this book is to help readers from diverse backgrounds who are looking for intuitive tools to understand and work confidently with high-dimensional tensors.

In this manuscript, we use LaTeX and *Tikz* package to illustrate all tensor diagrams.

1 Tensor Networks Basics

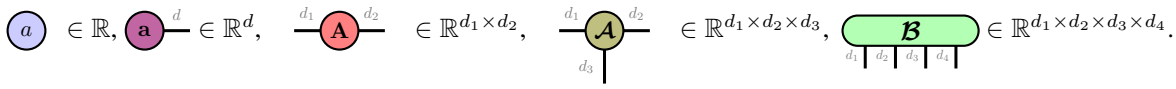
As their order increases, representing and working with tensors becomes more complicated. Tensor networks provide an efficient framework for working with these high-order objects, simplifying both their representation and analysis. The graphical notation of tensor networks offers an intuitive way to visualize and simplify complex tensor operations [Orús, 2014, Biamonte and Bergholm, 2017]. In this section, we introduce basic notions on tensors and common tensor operations using the language of tensor networks.

1.1 What are Tensor Networks?

As their name suggests, Tensor Networks (TNs) are simply tensors connected to each other to form a network. More precisely, a tensor network is a graph where vertices represent tensors and edges represent *contractions*. The graphical notation for tensor networks was first introduced by [Penrose et al., 1971]. Tensors are represented by shapes with legs (edges), i.e.,  . We will use different shapes, such as rectangles, triangles, or circles, and various colors to represent tensors. Shapes and colors serve mainly as visual cues. The *order* of a tensor is its number of dimensions, e.g., an N-th order tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ is a multidimensional array of scalars $\mathcal{T}_{i_1,\ldots,i_N}$ indexed by N indices $i_1 \in [d_1], i_2 \in [d_2], \cdots, i_N \in d_N$. Each axis is called a *mode* of a tensor: an Nth order tensor has N modes.

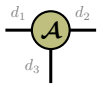
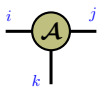
Tensors naturally generalize vectors and matrices to higher-order arrays. As the order increases, representing these arrays becomes more challenging. *Tensor networks* provide a simple and intuitive way to represent and manipulate these higher-order objects. Complex operations on tensors can be represented more easily with graphical notations of tensor networks [Biamonte and Bergholm, 2017, Orús, 2014].

Tensor Network Nodes In a tensor network, each node (or vertex) represents a tensor and the number of legs (incident edges) corresponds to the order of the tensor: scalars are nodes with no edges, vector nodes have single edge, matrix nodes have two edges, and so on, e.g.,,

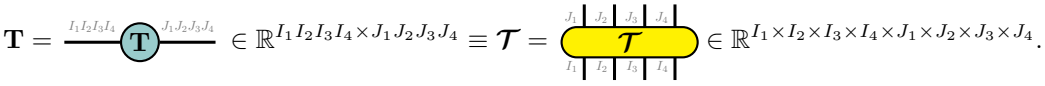


represent a scalar, a vector, a matrix, a 3rd and a 4th order tensors, respectively.

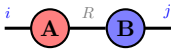
Throughout this manuscript,

- Tensors are represented by colored shapes called nodes, where the colors and shape have no specific meaning (unless stated otherwise).
- The dimension of each mode of a tensor is depicted by a gray letter positioned at the top of the corresponding leg, e.g.,  $\in \mathbb{R}^{d_1 \times d_2 \times d_3}$, is a 3rd order tensor of size $d_1 \times d_2 \times d_3$.
- Indices are presented by blue letters at the very end of edges, i.e., $\boldsymbol{\mathcal{A}}_{i,j,k} =$  is the (i, j, k)th element of the 3rd order tensor $\boldsymbol{\mathcal{A}}$.

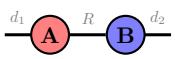
Note that large matrices can sometimes be viewed as high-order tensors through reshaping, e.g.,



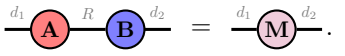
Tensor Network Edges In tensor network diagram, legs are of two types: contracted legs (those connecting two tensors) and un-contracted legs, also called free legs, with one dangling end (i.e., a leg that is not connected to any other tensor). As mentioned above, un-contracted legs correspond to free indices: the number of free legs indicates the order of a tensor: scalars have no free legs, vectors have one, matrices have two, and higher-order tensors have three or more. Contracted legs represent contractions: tensors can be connected along legs of the same sizes, which represents a summation (contraction) over the connected modes. We use the terms summation and contraction interchangeably. The most common contraction operation is *matrix multiplication*. For two matrices $\mathbf{A} \in \mathbb{R}^{d_1 \times R}, \mathbf{B} \in \mathbb{R}^{R \times d_2}$, their matrix product corresponds to a contraction between the 2nd mode of $\mathbf{A}$ and the 1st mode of $\mathbf{B}$:

$$(\mathbf{AB})_{ij} = \quad \text{  } \quad = \sum_{r=1}^{R} \mathbf{A}_{ir}\mathbf{B}_{rj}, \quad \text{for } i \in [d_1], j \in [d_2]\,. \tag{1}$$

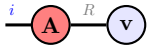
Since this equality is true for all indices i and j, we can simply write

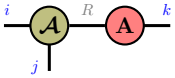
$$\mathbf{AB} = \text{  } \in \mathbb{R}^{d_1 \times d_2}. \tag{2}$$

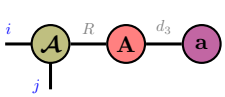
In the diagram above, we can see that there is a connection between the legs with the same size R. This is consistent with matrix multiplication where there is a sum over indices with the same dimension. Finally as the resulting diagram has two free legs, it represents a matrix. Here the final object is a matrix which can be represented as single node, demonstrating how nodes can be merged in tensor networks:

$$\mathbf{AB} = \mathbf{M} \Leftrightarrow \text{  }.$$

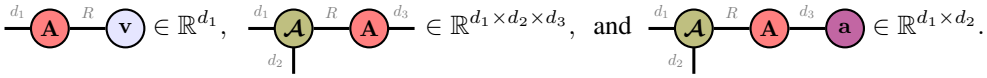
Contraction between two matrices can directly be extended to contractions between a matrix and a vector, a tensor and a matrix, a tensor a matrix and a vector products, etc. Here are some examples of tensor networks with their corresponding mathematical expressions:

$$\mathbf{A} \in \mathbb{R}^{d_1 \times R}, \mathbf{a} \in \mathbb{R}^{R} \;:\; \text{  } = \sum_{r=1}^{R} \mathbf{A}_{ir}\mathbf{v}_r, \tag{3}$$

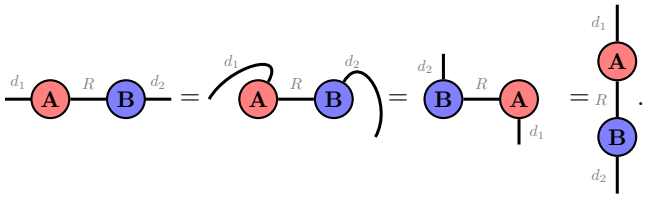
$$\boldsymbol{\mathcal{A}} \in \mathbb{R}^{d_1 \times R \times d_2}, \mathbf{A} \in \mathbb{R}^{R \times d_3} \; : \quad$$  $$= \sum_{r=1}^{R} \boldsymbol{\mathcal{A}}_{ijr} \mathbf{A}_{rk}, \tag{4}$$

$$\boldsymbol{\mathcal{A}} \in \mathbb{R}^{d_1 \times R \times d_2}, \mathbf{A} \in \mathbb{R}^{R \times d_3}, \mathbf{a} \in \mathbb{R}^{d_3} \; : \quad$$  $$= \sum_{r=1}^{R} \sum_{k=1}^{d_3} \boldsymbol{\mathcal{A}}_{ijr} \mathbf{A}_{rk} \mathbf{a}_k. \tag{5}$$

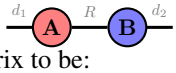
As we can see in all diagrams above, edges between two nodes represent summations. They also indicate that two tensors share dimensions of the same size on the corresponding mode. As explained before, the numbers of free legs in the tensor network corresponds to the order of the tensor it represents. For the previous examples, we have



From mathematical expressions to tensor networks, and back. Tensor networks are simple to work with because there is no strict rules for representing legs and nodes. In tensor network diagrams, nodes and edges can be positioned arbitrarily in the plane, only the graph structure matters. For example, when translating matrix multiplication into a tensor network diagram, the key feature is to ensure the dimension of the connected legs are consistent, e.g., for $\mathbf{A} \in \mathbb{R}^{d_1 \times R}, \mathbf{B} \in \mathbb{R}^{R \times d_2}$ all diagrams below illustrate the *same* matrix multiplication:

$\mathbf{AB} =$ 

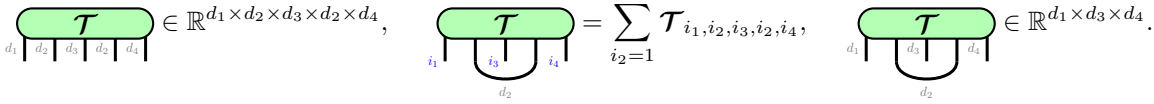
Note that, because of the sizes of $\mathbf{A} \in \mathbb{R}^{d_1 \times R}$ and $\mathbf{B} \in \mathbb{R}^{R \times d_2}$, there is only one way to do the contraction between the two, hence there would be no ambiguity in the diagrams above even if we omitted the dimensions on the edges.

On the other hand, translating tensor network diagrams into mathematical formulations can lead to multiple interpretations. For example, the following matrix multiplication diagram  can be interpreted differently depending on what we choose the shapes of $\mathbf{A}$, $\mathbf{B}$ and the resulting matrix to be:

- if $\mathbf{A} \in \mathbb{R}^{d_1 \times R}, \mathbf{B} \in \mathbb{R}^{R \times d_2}$ and the result is of size $d_1 \times d_2$, then the diagram represents the product $\mathbf{AB}$,
- if $\mathbf{A} \in \mathbb{R}^{R \times d_1}, \mathbf{B} \in \mathbb{R}^{R \times d_2}$ and the result is of size $d_1 \times d_2$, then the diagram represents the product $\mathbf{A}^\mathsf{T}\mathbf{B}$,
- if $\mathbf{A} \in \mathbb{R}^{R \times d_1}, \mathbf{B} \in \mathbb{R}^{R \times d_2}$ and the result is of size $d_2 \times d_1$, then the diagram represents the product $\mathbf{B}^\mathsf{T}\mathbf{A}$,
- ...

Therefore, one should be mindful when translating tensor network diagrams into mathematical formulations. But this is also waht makes tensor networks very useful! When we are working with the diagrams, rather than mathematical expressions, we do not need to care about e.g. transposes...

Contracting two legs of the same node. As explained previously, any two legs of the same dimension can be connected to form an edge in a tensor network, even two legs corresponding to the same node. Consider for example a 5th order tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{d_1 \times d_2 \times d_3 \times d_2 \times d_4}$. Since its 2nd and 4th modes have the same dimension, we can contract them together. Since the resulting tensor network has 3 dangling legs, it represents a 3rd order tensor:



When contracting two legs of an Nth order tensor, the resulting tensor is of order $N - 2$. This is closely related to the notion of *partial trace* that we will present later.

1.2 Inner Product, Outer Product, Trace and Norm

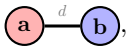
In this section, we use tensor networks to show how the classical notions of inner product, outer product, trace and norm can be generalized to higher order tensors.

Inner product Similarly to the classical inner (euclidean) product between vectors, the inner product of two N-th order tensors $\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{T}} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ is the sum of the products of their entries:

$$\langle \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{T}} \rangle = \sum_{i_1=1}^{d_1} \cdots \sum_{i_N=1}^{d_N} \boldsymbol{\mathcal{S}}_{i_1 \ldots i_N} \boldsymbol{\mathcal{T}}_{i_1 \ldots i_N}.$$

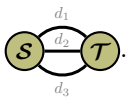
In a tensor network diagram, the summation over all dimensions is obtained by connecting all the legs of the two tensors. This results in a tensor network with no free legs, representing a scalar. For example, for vectors $\mathbf{a} \in \mathbb{R}^{d \times 1}, \mathbf{b} \in \mathbb{R}^{d \times 1}$ we have

$$\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^{d} \mathbf{a}_i \mathbf{b}_i = \tag{6}$$



and for two 3rd order tensors $\boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{T}} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ we have

$$\langle \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{T}} \rangle = \sum_{i_1=1}^{d_1} \sum_{i_2=1}^{d_2} \sum_{i_3=1}^{d_3} \boldsymbol{\mathcal{S}}_{i_1 i_2 i_3} \boldsymbol{\mathcal{T}}_{i_1 i_2 i_3} =$$



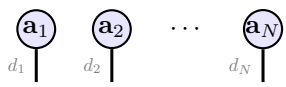
In this book, we will focus only on real valued tensors but the tensor network formalism can easily be adapted to complex valued ones. For example, if the two tensors were complex valued, one would have to take the conjugates of the entries of the second tensor, which could be indicated in the tensor network by having the node $\boldsymbol{\mathcal{T}}^*$ (componentwise conjugate) instead of $\boldsymbol{\mathcal{T}}$.

Outer product Recall that the outer product of two vectors $\mathbf{u} \in \mathbb{R}^m, \mathbf{v} \in \mathbb{R}^n$ is the $m \times n$ rank one matrix $\mathbf{u}\mathbf{v}^\top$. This operation can be generalized to any number of tensors. For example, the outer product of N vectors, $\mathbf{a}_1 \in \mathbb{R}^{d_1}, \cdots, \mathbf{a}_N \in \mathbb{R}^{d_N}$ is the tensor of order N defined by

$$(\mathbf{a}_1 \circ \mathbf{a}_2 \circ \cdots \circ \mathbf{a}_N)_{i_1, i_2, \cdots, i_n} = (\mathbf{a}_1)_{i_1} (\mathbf{a}_2)_{i_2} \cdots (\mathbf{a}_N)_{i_N} \text{ for all } i_1 \in [d_1], \cdots, i_N \in [d_N].$$

Such a tensor is called a **rank one** tensor. The diagram below illustrates the outer product of N vectors:

$$\mathbf{a}_1 \circ \mathbf{a}_2 \circ \cdots \circ \mathbf{a}_N =$$



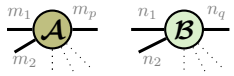
$$\in \mathbb{R}^{d_1 \times \cdots \times d_N}.$$

As a special case, for $N = 2$, we have $\mathbf{a}_1 \circ \mathbf{a}_2 = \mathbf{a}_1 \mathbf{a}_2^\mathsf{T} \in \mathbb{R}^{d_1 \times d_2}$. As we see in the tensor network diagram of the outer product, there are no shared edges, which indeed reflect that there are no contraction (summation) happening in an outer product. The notion of outer product can naturally be extended to higher order tensors. Let $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{m_1 \times \cdots \times m_p}$ and $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{n_1 \times \cdots \times n_q}$. The outer product of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ is the tensor of order $p + q$ defined by

$$(\boldsymbol{\mathcal{A}} \circ \boldsymbol{\mathcal{B}})_{i_1, i_2, \cdots, i_p, j_1, j_2, \cdots, j_q} = \boldsymbol{\mathcal{A}}_{i_1, i_2, \cdots, i_p} \boldsymbol{\mathcal{B}}_{j_1, j_2, \cdots, j_q}.$$

Like for vectors, the tensor networks of the outer product is simply obtained by juxtaposing the corresponding nodes without adding any edge:

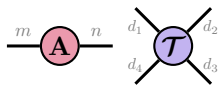
$$\boldsymbol{\mathcal{A}} \circ \boldsymbol{\mathcal{B}} =$$



$$\in \mathbb{R}^{m_1 \times \cdots \times m_p \times n_1 \times \cdots \times n_q}.$$

More generally, for any arbitrary number of tensors with arbitrary orders, the tensor network diagram of their outer product is obtained by simply placing them next to each other, e.g., for $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{d_1 \times d_2 \times d_3 \times d_4}$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{v} \in \mathbb{R}^p$

$$\mathbf{A} \circ \boldsymbol{\mathcal{T}} \circ \mathbf{v} =$$



$$\in \mathbb{R}^{m \times n \times d_1 \times d_2 \times d_3 \times d_4 \times p},$$

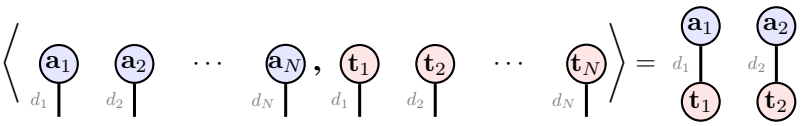
is defined element-wise by $(\mathbf{A} \circ \boldsymbol{\mathcal{T}} \circ \mathbf{v})_{i_1 i_2 j_1 j_2 j_3 j_4 k} = \mathbf{A}_{i_1 i_2} \boldsymbol{\mathcal{T}}_{j_1 \cdots j_4} \mathbf{v}_k$, where $i_1 \in [m], i_2 \in [n], j_l \in [d_l]$ for $l \in [4]$ and $k \in [p]$.

In the proposition below we show a simple visual proof of the fact that inner products of outer products are products of inner products.

Proposition 1. *Let* $\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{T}} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N}$ *with* $\boldsymbol{\mathcal{X}} = \mathbf{a}_1 \circ \mathbf{a}_2 \circ \cdots \circ \mathbf{a}_N$ *and* $\boldsymbol{\mathcal{T}} = \mathbf{t}_1 \circ \mathbf{t}_2 \circ \cdots \circ \mathbf{t}_N$*, then* $\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{T}} \rangle = \prod_{n=1}^{N} \langle \mathbf{a}_n, \mathbf{t}_n \rangle$.

Proof.

$$\langle \boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{T}} \rangle =$$

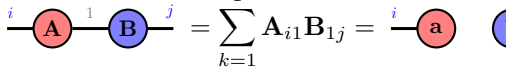


$$= \prod_{n=1}^{N} \langle \mathbf{a}_n, \mathbf{t}_n \rangle$$

□

Edge of dimension one = no edge. Note that in tensor network diagrams, an edge of size one is equivalent to having no edge. For example, in the matrix multiplication, if the contraction edge is of size one, it is equivalent to the outer product of vectors:

$$\mathbf{A} \in \mathbb{R}^{p \times 1}, \quad \mathbf{B} \in \mathbb{R}^{1 \times q}, \quad (\mathbf{AB})_{ij} =$$



$$= \sum_{k=1}^{1} \mathbf{A}_{i1} \mathbf{B}_{1j} = (\mathbf{a} \circ \mathbf{b})_{ij}$$

where $\mathbf{a}$ is the unique column of $\mathbf{A}$ and $\mathbf{b}$ the unique row of $\mathbf{B}$.

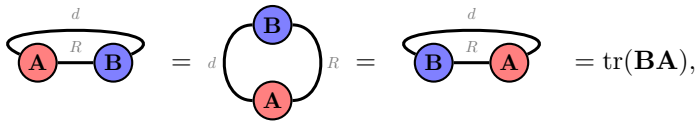
Trace. The trace operation is a special case of tensor contraction applied on square matrices. In tensor networks, the trace is naturally represented by a loop (self-edge) connecting the two legs of the matrix:

$$\mathbf{A} \in \mathbb{R}^{d \times d}, \qquad \operatorname{tr}(\mathbf{A}) = \sum_{i=1}^{d} \mathbf{A}_{ii} =$$

.

Since there are no free legs, it is consistent with the fact that the trace is a scalar. Tensor networks diagrams offers a very simple proof of the invariance of the trace under cyclic permutation. Let's start with the trace of the product of two matrices $\mathbf{A} \in \mathbb{R}^{d \times R}$ and $\mathbf{B} \in \mathbb{R}^{R \times d}$. As we explained before, the way the graph of a tensor network is drawn, only the graph structure is important. This allows us to see that

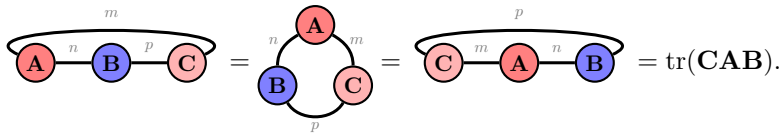
$$\operatorname{tr}(\mathbf{AB}) =$$



$$= \operatorname{tr}(\mathbf{BA}),$$

and similarly for three matrices $\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{n \times p}, \mathbf{C} \in \mathbb{R}^{p \times m}$, we have

$$\operatorname{tr}(\mathbf{ABC}) =$$



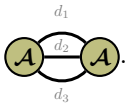
$$= \operatorname{tr}(\mathbf{CAB}).$$

Note that the equalities between the tensor network diagrams are trivial: we just changed the nodes' position without changing the underlying graph's structure. We thus simply interpreted the same diagram in two ways, leading to a less trivial equality between $\operatorname{tr}(\mathbf{ABC})$ and $\operatorname{tr}(\mathbf{CAB})$.

Partial traces The *partial trace* is a common operation used in particular in the context of modeling many-body quantum systems where density matrices of subpart of a system are obtained by performing a partial trace of the density matrix of the whole system. Without delving into the specifics of quantum systems, consider a square matrix of dimension $d_1 d_2 \times d_1 d_2$ which we interpret as a 4th order tensor  . The partial trace of $\mathcal{T}$ over the subspace $\mathbb{R}^{d_1}$ is the $d_2 \times d_2$ matrix obtained by "tracing out" the d_1 dimension by connecting the corresponding legs:  . Similarly, the partial trace of $\mathcal{T}$ over the subspace $\mathbb{R}^{d_2}$ is the $d_1 \times d_1$ matrix  .

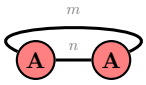
Tensor Frobenius Norm. The Frobenius norm of a tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ is the square root of the sum of its squared elements (i.e., the square root of the inner product with itself). In tensor networks, the norm of a 3rd-order tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ is represented as

$$\|\mathcal{A}\|_F^2 = \langle \mathcal{A}, \mathcal{A} \rangle = \sum_{i_1=1}^{d_1} \sum_{i_2=1}^{d_2} \sum_{i_3=1}^{d_3} \mathcal{A}_{i_1 i_2 i_3}^2 = \quad .$$



It is a direct and natural generalization of the matrix Frobenius norm:

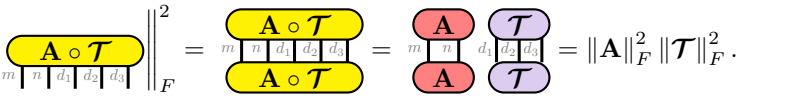
$$\|\mathbf{A}\|_F^2 = \mathrm{tr}(\mathbf{A}\mathbf{A}^\mathsf{T}) = \mathrm{tr}(\mathbf{A}^\mathsf{T}\mathbf{A}) = \quad .$$



Frobenius norm of outer products. Using tensor networks, it is almost trivial to show that the Frobenius norm of an outer product is equal to the product of the Frobenius norms. E.g., for $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, we have

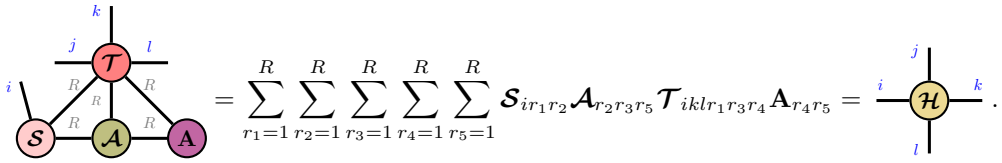
and thus

$$\|\mathbf{A} \circ \mathcal{T}\|_F^2 = \quad = \quad = \quad = \|\mathbf{A}\|_F^2 \, \|\mathcal{T}\|_F^2 \, .$$



Example We conclude this section with a last example showing how the tensor network graphical notations is very useful for representing complicated operations on tensors:

$$= \sum_{r_1=1}^{R} \sum_{r_2=1}^{R} \sum_{r_3=1}^{R} \sum_{r_4=1}^{R} \sum_{r_5=1}^{R} \mathcal{S}_{i r_1 r_2} \mathcal{A}_{r_2 r_3 r_5} \mathcal{T}_{i k l r_1 r_3 r_4} \mathbf{A}_{r_4 r_5} = \quad . \tag{7}$$



1.3 Copy Tensors (Hyperedges)

One sometimes needs to represent contractions between more than two indices. Consider for example the following contraction between there vectors to obtain a scalar: $\sum_{i=1}^{d} \mathbf{a}_i \mathbf{b}_i \mathbf{c}_i$. This operation can be depicted in tensor networks using a special tensor called *copy tensor*. The copy tensor is equivalent to a Kronecker delta, i.e. a hyper-diagonal tensor with ones on the diagonal and zeros elsewhere, and is represented as a black dot in tensor network diagrams.

E.g., the copy tensor  is defined element-wise as

$$\left(\quad \right)_{ijk} = \delta_{ijk} = \begin{cases} 1 & \text{if } i = j = k \\ 0 & \text{otherwise} \end{cases}.$$

Using the copy tensor, the 3-way contraction mentioned above can be represented as

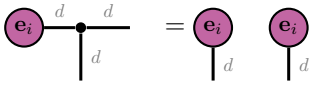
$$= \sum_{i,j,k=1}^{d} \delta_{ijk} \mathbf{a}_i \mathbf{b}_j \mathbf{c}_k = \sum_{i=1}^{d} \mathbf{a}_i \mathbf{b}_i \mathbf{c}_i.$$

We now give a more formal definition of the copy tensor of arbitrary order N.

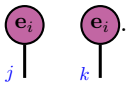
Definition 1. *The N-th order copy tensor*  *is the tensor of shape* $\underbrace{d \times d \cdots \times d}_{N \text{ times}}$ *defined by*

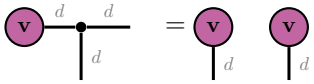
$$= \sum_{i=1}^{d} \underbrace{\mathbf{e}_i \circ \mathbf{e}_i \circ \ldots \circ \mathbf{e}_i}_{N \text{ times}},$$

where $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_d$ *are the vectors of the canonical basis of* $\mathbb{R}^d$.

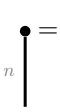
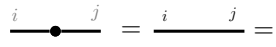
The name copy tensor comes from the fact that one can "copy" the canonical vectors by contraction with the copy tensor. Indeed, let $\mathbf{e}_i \in \mathbb{R}^d$ be a canonical basis vector, then one can check that  . Here is a shot proof of this fact:

$$= \sum_s (\mathbf{e}_i)_s \delta_{sjk} = \sum_s \delta_{is} \delta_{sjk} = \delta_{ijk} = \begin{cases} 1 & \text{if } i = j = k \\ 0 & \text{otherwise.} \end{cases} = (\mathbf{e}_i \mathbf{e}_i^{\mathsf{T}})_{j,k} =$$

 .

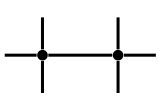
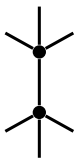
It is important to observe, that the "copy" property of the copy tensor only holds for canonical basis vector: it is not true that  for an arbitrary vector $\mathbf{v}$.

Remark 2. *In the following, we list some useful properties of copy tensors.*

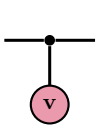
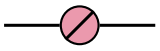
1. *The simplest version of the copy tensor is an all one vector, i.e.,*  $= \sum_{i=1}^{n} \mathbf{e}_i = \overrightarrow{\mathbf{1}}$.  $=$ $=$

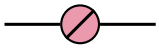
   $$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise,} \end{cases} = \mathbf{I}_{ij}.$$

2. *Interestingly, when contracting any number of legs from one copy tensor to legs of another copy tensor, the resulting tensor network is itself a copy tensor (obtained by merging the contracted legs and dots in the tensor network diagram). For example:*

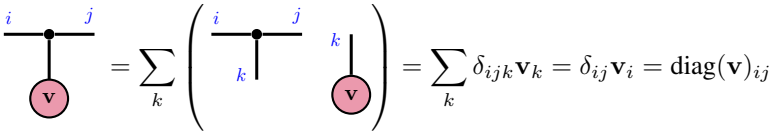
$$\sum_{l_1} \delta_{i_1 j_1 k_1 l_1} \delta_{i_2 j_2 k_2 l_1} =$$  $= \delta_{i_1 j_1 k_1 i_2 j_2 k_2} =$  $=$  $= \sum_{i_1} \delta_{i_1 j_1 k_1 l_1} \delta_{i_1 j_2 k_2 l_2}.$

3. *For any vector* $\mathbf{v} \in \mathbb{R}^n$*, let* $\mathrm{diag}(\mathbf{v}) \in \mathbb{R}^{n\times n}$ *denote the diagonal matrix having the entries of* $\mathbf{v}$ *on the diagonal, and for any matrix* $\mathbf{A} \in \mathbb{R}^{n\times n}$*, let* $\mathrm{diag}(\mathbf{A}) \in \mathbb{R}^n$ *denote the vector containing the diagonal entries of* $\mathbf{A} \in \mathbb{R}^{n\times n}$*.*

Then, $diag(\mathbf{v}) =$  $=$  *and* $diag(\mathbf{A}) =$  ,

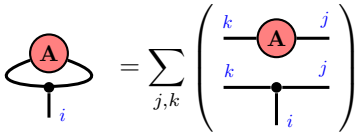
where  *represents a diagonal matrix.*

Proof. For the first claim, for any $i, j \in [n]$, we have



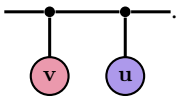
$$= \sum_k \left(\right) = \sum_k \delta_{ijk}\mathbf{v}_k = \delta_{ij}\mathbf{v}_i = \mathrm{diag}(\mathbf{v})_{ij}$$

For the second claim, for any $i \in [n]$, we have



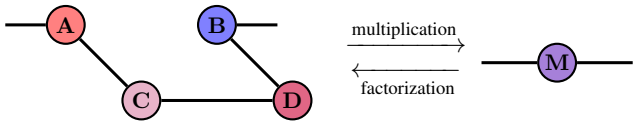
$$= \sum_{j,k} \left(\right) = \sum_{j,k} \delta_{ijk}\mathbf{A}_{kj} = \mathbf{A}_{ii} = \mathrm{diag}(\mathbf{A})_i.$$

□

4. *Consequently we can write from 3 we can write* $diag(\mathbf{v})diag(\mathbf{u}) =$  .

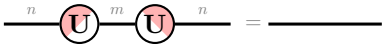
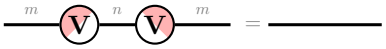
1.4 Matrix Factorization in Tensor Networks

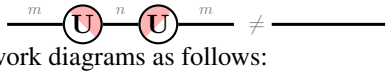
Tensor factorizations can be depicted in tensor network diagrams, just like any other operation. In this section, we will only introduce graphical diagrams of matrix factorizations. More general factorizations on tensors will be covered in Chapter 3. In tensor networks, factorizing means splitting a single node into multiple nodes, while multiplying refers to combining multiple nodes into a single node. This process can be clearly illustrated using tensor network diagram:



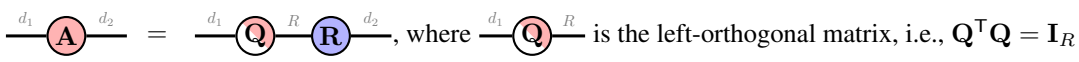
Therefore, we can represent any matrix decomposition in tensor network diagrams, including the celebrated QR and singular value decompositions (SVD). However, before presenting these decompositions in tensor network notation, we need to introduce orthogonal matrices. Higher order tensor orthogonality will be covered in Chapter 3.

Convention.

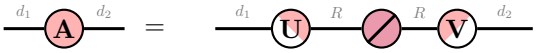
- The matrix $\mathbf{U} \in \mathbb{R}^{m\times n}$ is left orthogonal if the contraction of its transpose with itself from left yields the identity matrix ($\mathbf{U}^\mathsf{T}\mathbf{U} = \mathbf{I}_n$), e.g.,  $= \mathbf{I}_n$.
- The matrix $\mathbf{V} \in \mathbb{R}^{m\times n}$ is right orthogonal if the contraction with its transpose from right yields the identity matrix ($\mathbf{V}\mathbf{V}^\mathsf{T} = \mathbf{I}_m$), e.g.,  $= \mathbf{I}_m$.

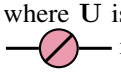
Note that the colored area points towards the identity edge in both left and right orthogonality. That means if the matrix $\mathbf{U}$ is left orthogonal then $\mathbf{U}\mathbf{U}^\mathsf{T}$ is not necessary the identity, i.e.,  $\neq$ $= \mathbf{I}_m$.
The QR decomposition and SVD can be presented in tensor network diagrams as follows:

- **QR Decomposition** For $\mathbf{A} \in \mathbb{R}^{d_1 \times d_2}$, $\mathbf{Q} \in \mathbb{R}^{d_1 \times R}$ and $\mathbf{R} \in \mathbb{R}^{R \times d_2}$,

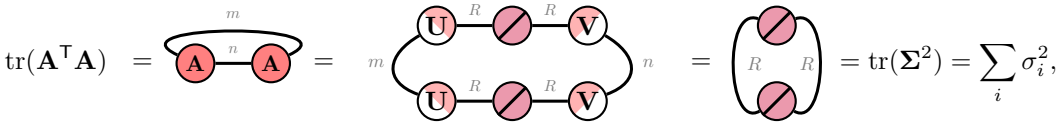


- **Singular Value Decomposition** For $\mathbf{A} \in \mathbb{R}^{d_1 \times d_2}$, $\mathbf{U} \in \mathbb{R}^{d_1 \times R}$, $\Sigma \in \mathbb{R}^{R \times R}$ and $\mathbf{V} \in \mathbb{R}^{R \times d_2}$,



  where $\mathbf{U}$ is the left-orthogonal ($\mathbf{U}^\top \mathbf{U} = \mathbf{I}_R$), $\mathbf{V}$ is the right-orthogonal ($\mathbf{V}\mathbf{V}^\top = \mathbf{I}_R$) and $\mathbf{\Sigma} \in \mathbb{R}^{R \times R} =$  represents the diagonal matrices, respectively.

Assuming $\mathbf{A}$ has the SVD represented in the previous diagram, we can give a short proof in tensor networks of $\mathrm{tr}(\mathbf{A}^\top \mathbf{A}) = \sum_i \sigma_i^2$, where σ_i are the singular values of matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$:



where we use left and right orthogonal property of matrices $\mathbf{U}$ and $\mathbf{V}$.

2 Operations on Tensors

As in the first chapter, a tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ can be seen as a multi-dimensional array with size(order) N. An N-th order or N-way tensor has N modes, where each mode represents one dimension [Kolda and Bader, 2009].

Tensor Fibers. For any mode i (where $i = 1, \ldots, N$), tensor *fibers* are obtained by keeping all indices fixed except the i-th one. For example, a matrix column corresponds to a mode-1 fiber, while a matrix row corresponds to a mode-2 fiber. In a third-order tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, we have $d_2 d_3$ mode-1 fibers, which are vectors of size d_1, i.e., $\mathcal{T}_{:,i_2,i_3} \in \mathbb{R}^{d_1}$, for $i_2 \in [d_2]$ and $i_3 \in [d_3]$. The colon indicates varying the first index while i_2 and i_3 remain fixed. In simple terms, fibers are vectors.

Tensor Slices. Slices of a tensor are two-dimensional arrays (matrices) obtained by fixing all but two indices. For a third-order tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$, there are horizontal, lateral, and frontal slices denoted by $\mathcal{T}_{i_1,:,:} \in \mathbb{R}^{d_2 \times d_3}$, $\mathcal{T}_{:,i_2,:} \in \mathbb{R}^{d_1 \times d_3}$, and $\mathcal{T}_{:,:,i_3} \in \mathbb{R}^{d_1 \times d_2}$, respectively. Therefore, slices are matrices.

2.1 Permute and Reshape Tensors

Permutation and reshaping are two fundamental operations on tensors.

- **Permutation** rearranges the indices of a tensor without changing its overall order. A common example of permutation is the transpose of a matrix.
- **Reshaping** combines multiple indices into larger ones or splits a large index into multiple smaller indices, reducing or increasing the total number of indices while preserving the tensor's overall size

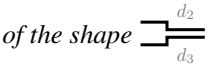
Next, we introduce matricization and vectorization, the two primary reshaping operations on tensors

Definition 2. *(Matricizitation) Let $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N}$. A mode-$n$ matricization denoted as $\mathcal{T}_{(n)} \in \mathbb{R}^{d_n \times d_1 d_2 \cdots d_{n-1} d_{n+1} \cdots d_N}$ for $n \in [N]$ is obtained by unfolding $\mathcal{T}$ into a matrix by taking all mode-n fibers and stacking them together as columns. For example, the matricization of a tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ along mode 1 which is represented by $\mathcal{T}_{(1)}$ is*

$$\begin{bmatrix} | & | & | & | \\ \mathcal{T}_{:,11} & \mathcal{T}_{:,12} & \cdots & \mathcal{T}_{:,d_2 d_3} \\ | & | & | & | \end{bmatrix} \in \mathbb{R}^{d_1 \times d_2 d_3}.$$

Observe that in this matricization, the two indices corresponding to the second and third modes of $\mathcal{T}$ are grouped together to form a new index that ranges from 1 to $d_2 d_3$. In tensor network diagrams, we represent such a grouping of indices by merging the corresponding legs together:

$$\mathcal{T}_{(1)} = \left(\begin{array}{c} d_1 \\ \mathcal{T} \\ d_3 \end{array} \begin{array}{l} \\ \\ d_2 \end{array} \right)_{(1)} = \begin{array}{c} d_1 \\ \mathbf{T} \\ d_3 \end{array} \begin{array}{l} d_2 \\ \\ \end{array} = \begin{array}{c} d_1 \\ \mathbf{T} \\ \end{array} \begin{array}{l} d_2 d_3 \\ \\ \end{array} \in \mathbb{R}^{d_1 \times d_2 d_3}.$$

As we can see, the grouped legs of the shape  *represents an edge of size $d_2 d_3$ in tensor network diagrams. The mode-2 and mode-3 matricization $\mathcal{T}_{(2)} \in \mathbb{R}^{d_2 \times d_1 d_3}$ and $\mathcal{T}_{(3)} \in \mathbb{R}^{d_3 \times d_1 d_2}$ are defined similarly.*

Matricization can also be seen as the flattening or unfolding of a tensor into a matrix. In general, the notion of matricization can be extended to any subset $I \subset [N]$ of the modes of $\mathcal{T}$ which maps I modes of $\mathcal{T}$ to the rows of $\mathbf{T}$ resulting in a matrix $\mathbf{T}_{([I])}$ of size $\prod_{i \in I} d_i \times \prod_{j \in [N] \setminus I} d_j$. For instance, for a 6-th order tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times \cdots \times d_6}$ we can group the indices as follows

$$\mathcal{T}_{i_1,\ldots,i_6} = \begin{array}{ccc} & i_2 \quad i_3 & \\ i_1 & \mathcal{T} & i_4 \\ & i_5 & \\ & i_6 & \end{array} \xrightarrow{I=[3]} \begin{array}{ccc} i_3 & & i_6 \\ i_2 & \mathbf{T} & i_5 \\ i_1 & & i_4 \end{array} = (\mathcal{T}_{([3])})_{i_1 i_2 i_3, i_4 i_5 i_6} \in \mathbb{R}^{d_1 d_2 d_3 \times d_4 d_5 d_6}.$$

Note that in matricization the modes of a tensor are partitioned into two sets. The indices $i_1, i_2, \cdots, i_I$ of a tensor are mapped to the row with index $j = i_{r_1} i_{r_2} \cdots i_{r_I}$ of $\mathbf{T}$ where $j = 1 + \sum_{l=1}^{I} \left[(i_{r_l} - 1) \prod_{m=1}^{l-1} I_{r_m} \right]$ and to the column $k = i_{c_1} i_{c_2} \cdots i_{c_{([N]\setminus I)}} = 1 + \sum_{n=1}^{[N]\setminus I} \left[(i_{c_n} - 1) \prod_{s=1}^{n-1} I_{r_s} \right]$ [Kolda, 2006, Kolda and Bader, 2009]. Likewise, converting a tensor to a matrix, we can also convert a tensor to a vector, which is called vectorization, and it is a special case of matricization.

Definition 3. *(Vectorization) Let $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N}$. The* vectorization *of $\mathcal{T}$ denoted as* $\text{vec}(\mathcal{T}) \in \mathbb{R}^{d_1 d_2 \cdots d_N}$ *is a vector obtained by concatenating its mode-1 fibers. For example the vectorization of a tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ is given by:*

$$\text{vec}(\mathcal{T}) = [\mathcal{T}_{:11} \mathcal{T}_{:12} \cdots \mathcal{T}_{:d_2 d_3}]^{\mathsf{T}} = \begin{array}{c} d_1 \\ \mathcal{T} \\ d_3 \end{array} \begin{array}{l} \\ d_2 \\ \end{array} \in \mathbb{R}^{d_1 d_2 d_3}.$$

We can also see vectorization as a flattening operation that converts a tensor of any order into a vector. Likewise, in matricization, the edge of the shape  presents an edge of size $d_1 d_2 d_3$. In general, throughout this manuscript, convergent edges represent an edge whose size is the product of the sizes of all associated edges. Note that the vectorization of a tensor is equal to the vectorization of mode-1 matricization of the tensor, i.e., for $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \cdots \times d_N}$, $\text{vec}(\mathcal{T}) = \text{vec}(\mathcal{T}_{(1)})$.

2.2 Products

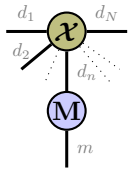
Tensors can be multiplied using different operations, similar to contraction for matrix multiplication, as discussed in Section 1.1. In this section, we provide graphical illustrations of different product operations.

Mode-n Tensor Product. Mode-n products, where a tensor is multiplied by a matrix along a specific mode can be seen as a generalization of matrix products. These include mode-n products between tensors and matrices, as well as tensors and vectors.

1. **Mode-n product (matrix).** The mode-n product of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ with a matrix $\mathbf{M} \in \mathbb{R}^{m \times d_n}$ denoted as $\boldsymbol{\mathcal{X}} \times_n \mathbf{M} \in \mathbb{R}^{d_1 \times \cdots \times d_{n-1} \times m \times d_{n+1} \times \cdots \times d_N}$ is defined as the contraction of a tensor with a matrix

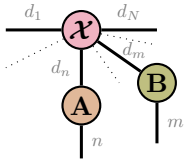
along mode-n of the tensor and mode-2 of the matrix, i.e.,

$$\boldsymbol{\mathcal{X}} \times_n \mathbf{M} = \in \mathbb{R}^{d_1 \times \cdots \times d_{n-1} \times m \times d_{n+1} \times \cdots \times d_N}$$



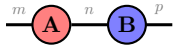
The operation contracts the tensor's n-th mode with the matrix's second mode, replacing the original dimension d_n with a new dimension m. For $\mathbf{A} \in \mathbb{R}^{d_n \times n}, \mathbf{B} \in \mathbb{R}^{d_m \times m}$ and $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ with distinct modes $n \neq m$, the order of multiplication does not matter, i.e.,

$$\boldsymbol{\mathcal{X}} \times_n \mathbf{A} \times_m \mathbf{B} = \quad = \boldsymbol{\mathcal{X}} \times_m \mathbf{B} \times_n \mathbf{A}, \quad \text{For } (n \neq m).$$

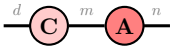


However, if $m = n$ and $\mathbf{A} \in \mathbb{R}^{n \times d_n}$ and $\mathbf{B} \in \mathbb{R}^{p \times n}$ then $(\boldsymbol{\mathcal{X}} \times_n \mathbf{A}) \times_n \mathbf{B} = \boldsymbol{\mathcal{X}} \times_n (\mathbf{BA})$. The mode-$n$ tensor product can be seen as a generalization of matrix multiplication, i.e.,
Let $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{B} \in \mathbb{R}^{p \times n}$ and $\mathbf{C} \in \mathbb{R}^{d \times m}$, then

$$\mathbf{A} \times_2 \mathbf{B} = = \mathbf{A}\mathbf{B}^\mathsf{T} \in \mathbb{R}^{m \times p}.$$



$$\mathbf{A} \times_1 \mathbf{C} = = \mathbf{C}\mathbf{A} \in \mathbb{R}^{d \times n}.$$

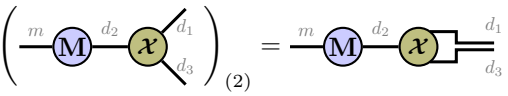


Lastly, we have the following proposition on mode-n tensor multiplication:

Proposition 3. *Let* $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ *and* $\mathbf{M} \in \mathbb{R}^{m \times d_n}$ *then* $(\boldsymbol{\mathcal{X}} \times_n \mathbf{M})_{(n)} = \mathbf{M}\boldsymbol{\mathcal{X}}_{(n)}$.

Proof. We show this identity for the special case $n = 2$ and $N = 3$. The extension to the general case is straightforward. We have

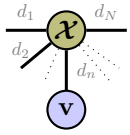
$$(\boldsymbol{\mathcal{X}} \times_2 \mathbf{M})_{(2)} = \left(\right)_{(2)} = = \mathbf{M}\boldsymbol{\mathcal{X}}_{(2)} \in \mathbb{R}^{m \times d_1 d_3}.$$



□

2. **Mode-n product (vector).** The mode-n product of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ with a vector $\mathbf{v} \in \mathbb{R}^{d_n}$, is denoted by $\boldsymbol{\mathcal{X}} \times_n \mathbf{v}$ and is a tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d_1 \times \cdots \times d_{n-1} \times d_{n+1} \times \cdots \times d_N}$. The result is a tensor of order $N - 1$. It can be pictured in a tensor network diagram as

$$\boldsymbol{\mathcal{X}} \times_n \mathbf{v} = \in \mathbb{R}^{d_1 \times \cdots \times d_{n-1} \times d_{n+1} \times \cdots \times d_N}.$$



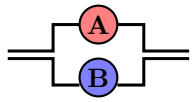
Note that, in mode-n vector multiplication, unlike mode-n matrix multiplication, the order of multiplication matters because it affects intermediate results. Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{d_1 \times \cdots \times d_m \times \cdots \times d_n \times \cdots \times d_N}$ and $\mathbf{a} \in \mathbb{R}^{d_n}, \mathbf{b} \in \mathbb{R}^{d_m}$, then $\boldsymbol{\mathcal{X}} \times_n \mathbf{a} \times_m \mathbf{b} \neq (\boldsymbol{\mathcal{X}} \times_m \mathbf{b}) \times_n \mathbf{a}$, as mode-$n$ vector multiplication drops the m-th dimension [Bader and Kolda, 2006].

Next, we introduce the Kronecker product, as well as the Khatri-Rao and Hadamard products.

Kronecker product Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{p \times q}$ then the Kronecker product, $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{mp \times nq}$ is defined by

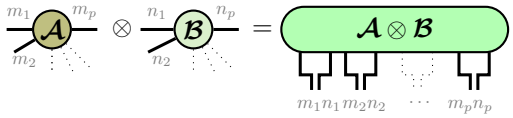
$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \dots & a_{1n}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \dots & a_{2n}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & a_{m2}\mathbf{B} & \dots & a_{mn}\mathbf{B} \end{bmatrix} \tag{8}$$

In tensor networks, the Kronecker product is represented by the following diagram:

$\mathbf{A} \otimes \mathbf{B} =$ 

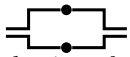
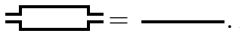
. (9)

As one can see from this diagram, the Kornecker product of two matrices is nothing else than a reshaping of their outer products. More generally, the Kronecker product can be defined for any two tensors with the same order. It is obtained by grouping together and in order each pair of legs of the two tensors, e.g.,

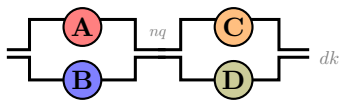
$\mathcal{A} \otimes \mathcal{B} =$ 

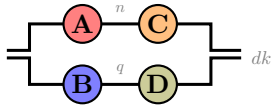
$\in \mathbb{R}^{m_1 n_1 \times \cdots \times m_p n_p}$.

Remark 4. *In the following, we list some useful properties of the Kronecker product.*

1. *Kronecker product is not commutative, i.e.,* $\mathbf{A} \otimes \mathbf{B} \neq \mathbf{B} \otimes \mathbf{A}$
2. $\mathbf{I} \otimes \mathbf{A} = \begin{bmatrix} \mathbf{A} & 0 & \dots & 0 \\ 0 & \mathbf{A} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{A} \end{bmatrix}$ *is a block diagonal matrix.*
3. *The Kronecker product of two tensors of the same order results in a tensor of the same order, while their outer product produces a tensor with double the order.*
4. *By reshaping the Kronecker product* $\mathbf{A} \otimes \mathbf{B}$*, the outer product* $\mathbf{A} \circ \mathbf{B}$ *can be obtained.*
5. ***Kronecker product of copy tensors*** *is*  *while the* ***outer product of copy tensors*** *is*  *, where it reveals that they are not equal, but reshaping of one another.*
6. *The Kronecker product of two* ***identity matrices*** *is another identity matrix, i.e.,*  *. In general, the legs of a Kronecker product can be represented as straight lines, with their size corresponding to the product of the sizes of the respective legs.*
7. *Kronecker product has a mixed product property, i.e.,* $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}$ *where* $\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{p \times q}, \mathbf{C} \in \mathbb{R}^{n \times d}$ *and* $\mathbf{D} \in \mathbb{R}^{q \times k}$

   Proof.

$$(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) =$$ 

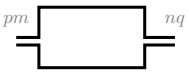
$$=$$ 

$$= \mathbf{AC} \otimes \mathbf{BD}.$$

□

As a special case, we can see $(\mathbf{A} \otimes \mathbf{I})(\mathbf{I} \otimes \mathbf{B}) = \mathbf{A} \otimes \mathbf{B}$.

8. *As we can see in the derivation above, it is often useful to reshape* 

as 

and vice versa when dealing with identities involving Kronecker products.

9. *For* $\mathbf{A} \in \mathbb{R}^{m \times m}$ *and* $\mathbf{B} \in \mathbb{R}^{n \times n}$*, we have the equality* $\operatorname{tr}(\mathbf{A} \otimes \mathbf{B}) = \operatorname{tr}(\mathbf{A}) \operatorname{tr}(\mathbf{B})$*, which can easily be proved by the following diagram:*

$$\operatorname{tr}(\mathbf{A} \otimes \mathbf{B}) = \quad = \operatorname{tr}(\mathbf{A}) \operatorname{tr}(\mathbf{B}).$$

10. ***Sylvester Identity***

$$\operatorname{vec}(\mathbf{AXB}) = (\mathbf{B}^{\mathsf{T}} \otimes \mathbf{A})\operatorname{vec}(\mathbf{X}),$$

where $\mathbf{A} \in^{m \times n}, \mathbf{X} \in \mathbb{R}^{n \times p}$ *and* $\mathbf{B} \in \mathbb{R}^{p \times q}$.

Proof.

$$\operatorname{vec}(\mathbf{AXB}) = \operatorname{vec}\Big(\quad \Big) = \quad = \quad$$

$$= (\mathbf{A} \otimes \mathbf{B}^{\mathsf{T}})\operatorname{vec}(\mathbf{X}).$$

□

The Sylvester identity *is a useful identity to solve the system of equations of the form* $\mathbf{AX} + \mathbf{XB} = \mathbf{C}$.

Useful relationships exist between the n-mode matrix product, matricization, and Kronecker products, i.e.,

Proposition 5. *Let* $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_p}$, $\mathbf{A}_1 \in \mathbb{R}^{d_1 \times n_1}, \mathbf{A}_2 \in \mathbb{R}^{d_2 \times n_2}, \ldots, \mathbf{A}_p \in \mathbb{R}^{d_p \times n_p}$, *then*

$$(\boldsymbol{\mathcal{A}} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \times_3 \mathbf{A}_3 \times_4 \ldots \times_p \mathbf{A}_p)_{(n)} = \mathbf{A}_n \boldsymbol{\mathcal{A}}_{(n)} (\mathbf{A}_1 \otimes \ldots \otimes \mathbf{A}_{n-1} \otimes \mathbf{A}_{n+1} \otimes \ldots \otimes \mathbf{A}_p)^{\mathsf{T}}$$

Proof. For $p = 3$ and $n = 1$,

$$(\boldsymbol{\mathcal{A}} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \times_3 \mathbf{A}_3)_{(1)} = \Big(\quad \Big)_{(1)} = \quad$$

$$= \quad = \mathbf{A}_1 \boldsymbol{\mathcal{A}}_{(1)} (\mathbf{A}_2 \otimes \mathbf{A}_3)^{\mathsf{T}}.$$

□

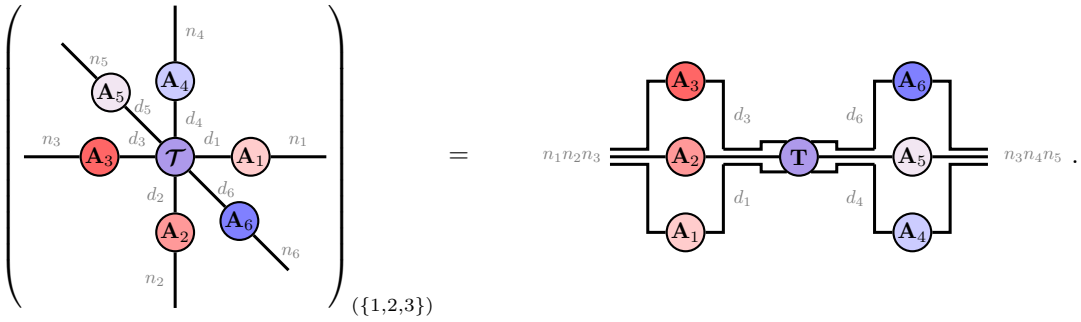
Proposition 6. *Let $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_p}$ and $\mathbf{A}_i \in \mathbb{R}^{d_i \times n_i}$ for $i \in [p]$. Let $m \in [p]$ and recall that $\mathbf{T}_{(m)}$ denotes reshaping the tensor $\mathcal{T}$ in a matrix of size $d_1 d_2 \cdots d_m \times d_{m+1} \ldots d_p$ by mapping the first m indices to rows and the remaining ones to columns. Then,*

$$(\mathcal{T} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \cdots \times_p \mathbf{A}_p)_{([m])} = (\mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \cdots \otimes \mathbf{A}_m)\mathbf{T}_{([m])}(\mathbf{A}_{m+1} \otimes \mathbf{A}_{m+2} \otimes \cdots \otimes \mathbf{A}_p)^{\mathsf{T}}.$$

This identity can be extended to arbitrary subsets $I \subset [p]$:

$$(\mathcal{T} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \cdots \times_p \mathbf{A}_p)_{(I)} = \left(\bigotimes_{i \in I} \mathbf{A}_i\right) \mathbf{T}_{([m])} \left(\bigotimes_{i \in [p] \setminus I} \mathbf{A}_i\right)^{\mathsf{T}}.$$

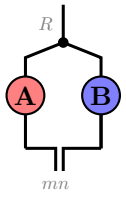
Proof. For $p = 6$ and $I = \{1, 2, 3\}$,



□

Khatri-Rao product The Khatri-Rao product is the column-wise Kronecker product [Smilde et al., 2005, Bro, 1998]. Let $\mathbf{A} \in \mathbb{R}^{m \times R}$ and $\mathbf{B} \in \mathbb{R}^{n \times R}$ then the Khatri-Rao product of $\mathbf{A}$ and $\mathbf{B}$ denoted as $\mathbf{A} \odot \mathbf{B} \in \mathbb{R}^{mn \times R}$ is defined by

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} | & | & & | \\ \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \ldots & \mathbf{a}_R \otimes \mathbf{b}_R \\ | & | & & | \end{bmatrix} =$$



,

where $\mathbf{a}_1, \ldots, \mathbf{a}_R \in \mathbb{R}^m$ are the columns of $\mathbf{A}$ and $\mathbf{b}_1, \ldots, \mathbf{b}_R \in \mathbb{R}^n$ are the columns of $\mathbf{B}$. In the corresponding tensor network diagram, the copy tensor represents the fact that the resulting matrix only contains the Kronecker product of *matching* columns of $\mathbf{A}$ and $\mathbf{B}$.

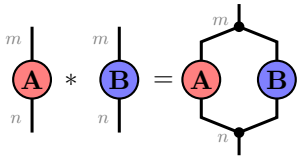
Remark 7. *Some of the Khatri-Rao product properties are listed below:*

1. *Like the Kronecker product, the Khatri-Rao product is not commutative, i.e., $\mathbf{A} \odot \mathbf{B} \neq \mathbf{B} \odot \mathbf{A}$.*
2. *The Khatri-Rao product is associative, i.e., $\mathbf{A} \odot (\mathbf{B} \odot \mathbf{C}) = (\mathbf{A} \odot \mathbf{B}) \odot \mathbf{C}$, where $\mathbf{A} \in \mathbb{R}^{m \times R}, \mathbf{B} \in \mathbb{R}^{n \times R}$ and $\mathbf{C} \in \mathbb{R}^{s \times R}$.*
3. *The columns of $\mathbf{A} \odot \mathbf{B}$ form a subset of the columns of the Kronecker product $\mathbf{A} \otimes \mathbf{B}$.*

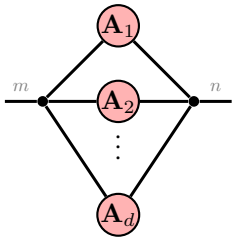
Hadamard product Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{m \times n}$ be matrices of the same dimension, then the Hadamard product is the matrix $\mathbf{A} * \mathbf{B} \in \mathbb{R}^{m \times n}$ defined element-wise by

$$(\mathbf{A} * \mathbf{B})_{ij} = \mathbf{A}_{ij}\mathbf{B}_{ij}.$$

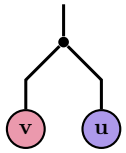
In tensor network diagrams, the Hadamard product can easily be represented using copy tensors:

$\mathbf{A} * \mathbf{B} =$ 

More generally, for any $\mathbf{A}_1 \in \mathbb{R}^{m\times n} \dots \mathbf{A}_d \in \mathbb{R}^{m\times n}$ we have

$\mathbf{A}_1 * \mathbf{A}_2 \dots * \mathbf{A}_d =$ 

Remark 8. *We list here some useful properties of the Hadamard product.*

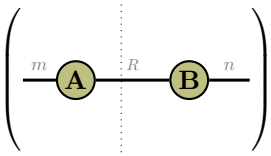
- *The Hadamard product of two copy tensors is a copy tensor.*
- *From 2, items 4 and five, we can write* $\mathbf{v} * \mathbf{u} =$ 

$= diag(\mathbf{v})\mathbf{u}$

In the following section, we introduce several theorems that are useful for tensor decompositions.

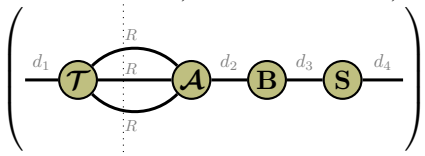
2.3 Useful TN results

Theorem 9. *(Rank of matricization of TN) For an arbitrary tensor network, the rank of any matricization is bounded by the weight of the cut of the graph.*

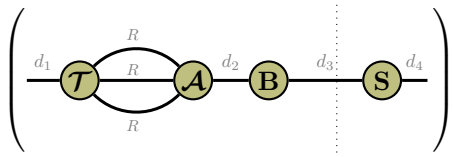
1. *For* $\mathbf{A} \in \mathbb{R}^{m\times R}$ *and* $\mathbf{B} \in \mathbb{R}^{R\times n}$, $rank(\mathbf{AB}) =$ *rank* 

$\leq R$

2. *Let* $\mathcal{T} \in \mathbb{R}^{d_1\times R\times R\times R}$, $\mathcal{A} \in \mathbb{R}^{R\times R\times R\times d_2}$, $\mathbf{B} \in \mathbb{R}^{d_2\times d_3}$ *and* $\mathbf{S} \in \mathbb{R}^{d_3\times d_4}$, *then*

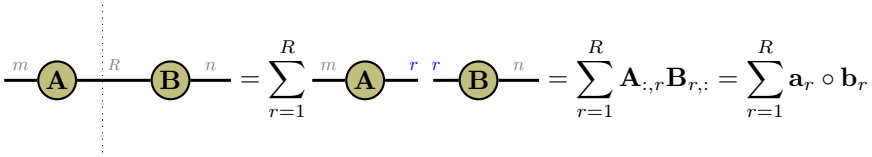
rank 

$\leq R^3$ *and*

rank 

$\leq d_3$

cutting edge Any cut of the graph of a tensor network corresponds to one way to express the tensor as a sum of outer products of tensors. The weight of the cut corresponds to the number of summands. For example:



where $\mathbf{a}_r$ are the columns of $\mathbf{A} \in \mathbb{R}^{m \times R}$ and $\mathbf{b}_r$ are the rows of $\mathbf{B} \in \mathbb{R}^{R \times n}$.

3 Tensor Decompositions

Working with high-order tensors is computationally expensive because the number of elements grows exponentially with the tensor's order. Tensor decompositions have emerged as powerful and efficient tools to address this issue. Similar to matrix factorizations, tensor decompositions break down a high-order tensor into smaller components with lower order and fewer entries, making them easier to work with. However, the notion of rank of a matrix can be extended in various ways for tensors, each of them associated with a different factorization / decomposition model. In this chapter, we introduce the most well-known tensor decomposition models.

3.1 CANDECOMP/PARAFAC (CP) Decomposition

The CP decomposition [Hitchcock, 1927] of an N-th order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ decomposes $\boldsymbol{\mathcal{A}}$ as the sum of a finite number of rank-one tensors. Equivalently, it is a linear combination of R rank-one tensors where R is called the rank of the decomposition:

$$\boldsymbol{\mathcal{A}} = \sum_{r=1}^{R} \mathbf{a}_1^{(r)} \circ \ldots \circ \mathbf{a}_N^{(r)},$$

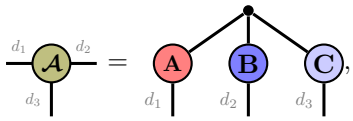
where $\mathbf{a}_n^{(1)}, \ldots, \mathbf{a}_n^{(R)} \in \mathbb{R}^{d_n}$ for each $n \in [N]$.

By grouping all the vectors $\mathbf{a}_n^{(r)}$ in factor matrices,

$$\mathbf{A}_1 = \begin{pmatrix} \mathbf{a}_1^{(1)} & \ldots & \mathbf{a}_1^{(R)} \end{pmatrix} \in \mathbb{R}^{d_1 \times R}, \ldots, \mathbf{A}_N = \begin{pmatrix} \mathbf{a}_N^{(1)} & \ldots & \mathbf{a}_N^{(R)} \end{pmatrix} \in \mathbb{R}^{d_N \times R},$$

the CP decomposition is concisely noted as $\boldsymbol{\mathcal{A}} = [\![\mathbf{A}_1, \cdots, \mathbf{A}_N]\!]$.

In tensor networks, a CP decomposition $\boldsymbol{\mathcal{A}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ is represented by



where the black dot is the third order copy tensor introduced in Section 1.3.

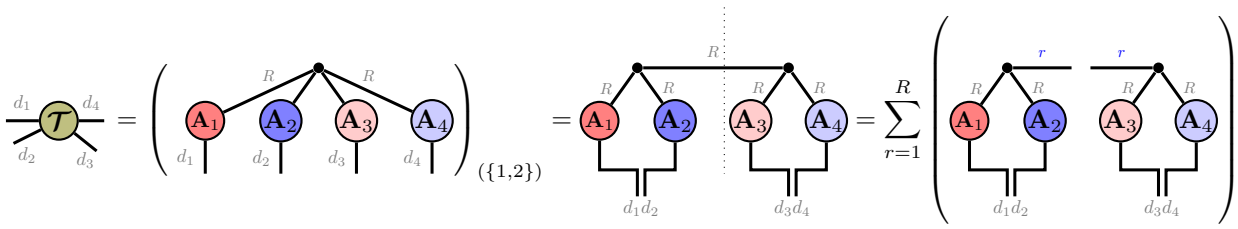
CP Rank of a Tensor The most fundamental and oldest concept of rank for tensors is the CP rank, first introduced by [Hitchcock, 1927]. The CP rank of a tensor is defined as the minimum number of rank-one tensors required to express the tensor as their sum. While this definition is similar to that of matrix rank, the properties of tensor rank differ significantly from those of matrix rank. From a computational standpoint, one key difference is that, unlike matrix rank, there is no straightforward polynomial-time algorithm to determine a tensor's CP rank. In fact, computing the rank of a tensor is an NP-hard problem [Hillar and Lim, 2013]. There are several variations of tensor rank, each associated with a specific tensor decomposition. As we progress through this chapter, we will introduce some of these different types of ranks

Remark 10. *We list here some interesting properties of the CP rank and the CP decomposition.*

1. *From Theorem 9 and Remark 2 (item 3), one can show that if a tensor $\boldsymbol{\mathcal{A}}$ admits a rank R CP decomposition, decomposition, then all its matricizations have a rank upper bounded by R:*

Proposition 11. *If $\mathcal{A} = [\![\mathbf{A}_1, \cdots, \mathbf{A}_N]\!]$ is rank R CP decomposition of $\mathcal{A}$, then $(\operatorname{rank}(\mathcal{A}))_I \leq R$ for any $I \subset [N]$.*

The following tensor network illustrates this result.



2. *The CP rank of a tensor $\mathcal{A} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ can easily be upper bounded as*

$$\operatorname{rank}_{\mathrm{CP}}(\mathcal{A}) \leq \min_{n \in [N]} \prod_{i \neq n} d_i.$$

3. *For order 2 tensors $\mathbf{A} \in \mathbb{R}^{d_1 \times d_2}$, we can recover the classical notion of rank R factorization.*

$$\mathbf{A} = \sum_{r=1}^{R} \mathbf{a}_1^{(r)} \circ \mathbf{a}_2^{(r)} = \begin{pmatrix} \mathbf{a}_1^{(1)} & \dots & \mathbf{a}_1^{(R)} \end{pmatrix} \begin{pmatrix} \mathbf{a}_2^{(1)} & \dots & \mathbf{a}_2^{(R)} \end{pmatrix}^{\mathsf{T}}.$$

4. *The CP decomposition $\mathcal{A} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ can be expressed using the Kronecker delta*

$$\mathcal{A}_{i,j,k} = \sum_{r_1, r_2, r_3} \delta_{r_1, r_2, r_3} \mathbf{A}_{i, r_1} \mathbf{B}_{j, r_2} \mathbf{C}_{k, r_3},$$

as well as with mode-n products (see 2.2)

$$\mathcal{A} = \mathcal{I} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$$

where $\mathcal{I}$ is the 3rd order copy tensor.

5. *The rank of the second-order tensors (matrices), over the fields $\mathbb{R}$ and $\mathbb{C}$ is the same. However, for higher order tensors (N-th order tensors with $N \geq 3$) the rank can vary depending on the decomposition field [Kruskal, 1989].*

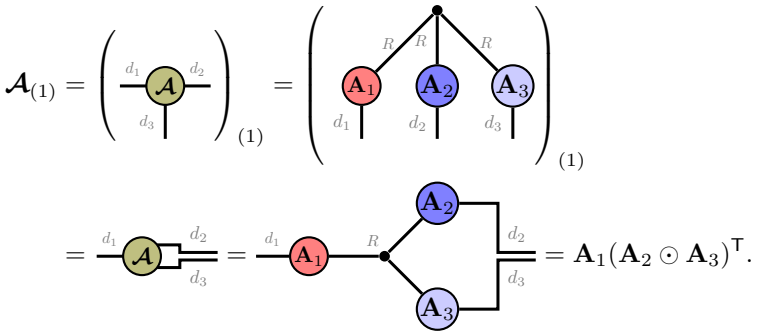
6. *The CP of N-th order d-dimensional tensors ($d_1 = \cdots = d_N = d$) using only $\mathcal{O}(NdR)$ parameters instead of d^N. If R is small then the number of parameters can be significantly reduced. Therefore, a smaller CP rank R results in a more efficient CP decomposition.*

The following proposition shows how the mode n matricization of the CP decomposition of a tensor can be expressed as a Khatri-Rao product

Proposition 12. *Let $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N}$. If $\mathcal{A} = [\![\mathbf{A}_1, \mathbf{A}_2, \dots, \mathbf{A}_N]\!]$, then*

$$\mathcal{A}_{(n)} = \mathbf{A}_n \left(\mathbf{A}_1 \odot \dots \odot \mathbf{A}_{(n-1)} \odot \mathbf{A}_{(n+1)} \odot \dots \odot \mathbf{A}_N \right)^{\mathsf{T}}.$$

Proof. For $N = 3$ and $n = 1$,



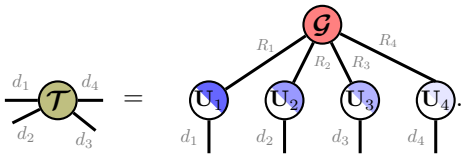
□

3.2 Tucker Decomposition

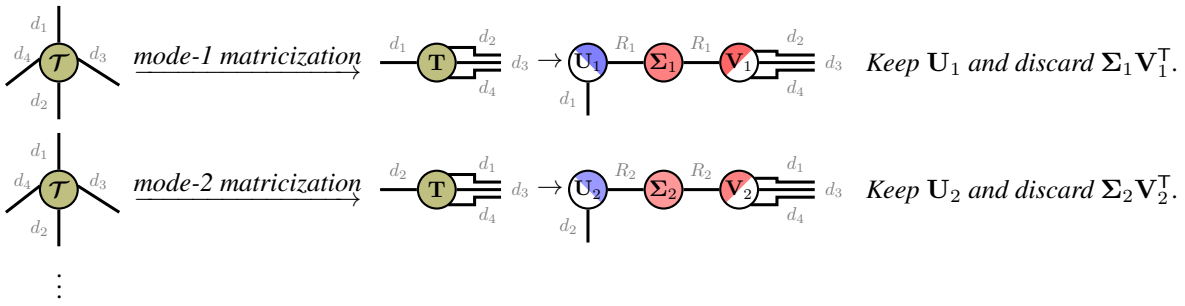
Tucker introduced the Tucker decomposition which factorizes an N-th order tensor into a smaller tensor and N factor matrices. In this decomposition, the smaller tensor is called the core tensor [Tucker, 1966]. The Tucker decomposition consists in N mode-n products (see, e.g., 1) between the core tensor and the factor matrices. Let $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$. The Tucker decomposition of the tensor $\boldsymbol{\mathcal{T}}$ is given by $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{U}_1 \times_2 \ldots \times_{N-1} \mathbf{U}_{N-1} \times_N \mathbf{U}_N$, where $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$ is the core tensor and $\mathbf{U}_i \in \mathbb{R}^{d_i \times R_i}, i \in [N]$ are the factor matrices. The tuple $(R_1, R_2, \cdots, R_N)$, which contains the dimensions of the core tensor along all its modes, is called the Tucker (or multilinear) rank of $\boldsymbol{\mathcal{T}}$. It is not difficult to show that the factor matrices $\mathbf{U}_i \in \mathbb{R}^{d_i \times R_i}$ can always be chosen to be unitary (see item 6 in Remark 13 below).

Remark 13. *We list here some interesting properties of the Tucker decomposition and the HOSVD algorithm.*

1. *The Tucker decomposition of a fourth-order tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{d_1 \times \cdots \times d_4}$ can be illustrated using tensor networks notation as follows.*

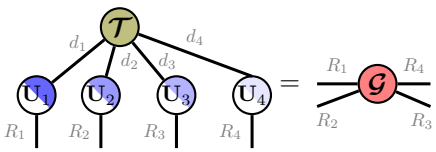


2. ***Computing The Tucker Decomposition.*** *The basic idea of the Tucker decomposition is to find the R_n leading left singular vectors in mode n, independently of the other modes. This algorithm, known as Higher-Order SVD (HOSVD), is depicted below. We begin by using the fact that an SVD exists for any mode-n matricization of the tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$. For simplicity, we illustrate this process for $N = 4$,*



   Construct the tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_4}$ by performing a mode-n product with the transpose of the retained factor matrices for each corresponding mode (for $n \in [4]$), i.e.,

$$\boldsymbol{\mathcal{T}} \times_1 \mathbf{U}_1^\mathsf{T} \times_2 \ldots \times_4 \mathbf{U}_4^\mathsf{T} = \quad = \quad$$

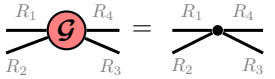


   The core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1 \times \cdots \times R_4}$ along with the factor matrices $\mathbf{U}_1, \ldots, \mathbf{U}_4$ forms a Tucker decomposition of the tensor $\boldsymbol{\mathcal{T}}$. If all the SVDs of the matricization of $\boldsymbol{\mathcal{T}}$ are exact, the resulting Tucker decomposition will be exact as well. When using truncated SVDs, the overall approximation error of the Tucker decomposition can be expressed as a function of the errors made in each truncated SVD (see [De Lathauwer et al., 2000]).

3. *By above derivation we can conclude that $vec(\boldsymbol{\mathcal{G}}) = (\mathbf{U}_1 \otimes \cdots \otimes \mathbf{U}_N) vec(\boldsymbol{\mathcal{T}})$*

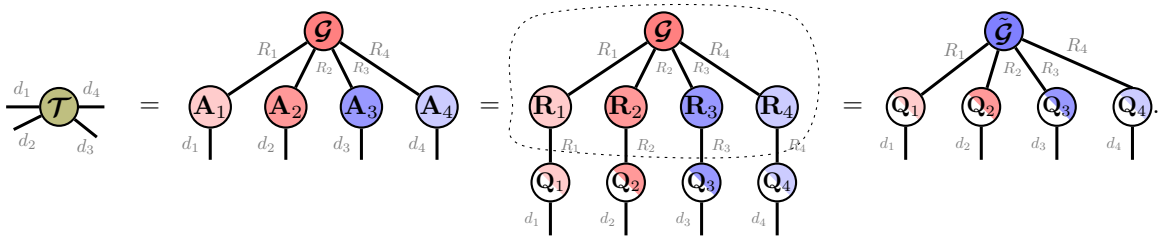
   Proof. □

4. *If $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{U}_1 \times_2 \ldots \times_{N-1} \mathbf{U}_{N-1} \times_N \mathbf{U}_N$ with all the $\mathbf{U}_i$ being orthogonal, then $\|\boldsymbol{\mathcal{T}}\|_F = \|\boldsymbol{\mathcal{G}}\|_F$.*

5. *If we don't enforce any orthogonality constraints on the factor matrices and fix the core tensor to be a copy tensor,*  *, then the CP decomposition is recovered.*

6. *If* $\mathcal{T} = \mathcal{G} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 \times_3 \mathbf{A}_3 \times_4 \mathbf{A}_4$*, with* $\mathbf{A}_1, \mathbf{A}_2, \mathbf{A}_3$ *and* $\mathbf{A}_4$ *not necessarily orthogonal, then there exists* $\tilde{\mathcal{G}}$ *(of the same size as* $\mathcal{G}$*) and* $\mathbf{Q}_1, \mathbf{Q}_2, \mathbf{Q}_3$ *and* $\mathbf{Q}_4$ *orthogonal such that* $\mathcal{T} = \tilde{\mathcal{G}} \times_1 \mathbf{Q}_1 \times_2 \mathbf{Q}_2 \times_3 \mathbf{Q}_3 \times_4 \mathbf{Q}_4$*.*

   Proof. By performing QR decomposition on each factor matrix, we obtain



□

7. *The Tucker rank of a tensor* $\mathcal{T}$ *is determined by the rank of its matricizations, i.e.,* $\operatorname{rank}(\mathcal{T}_{(i)})$ *for* $i \in [N]$ *[De Lathauwer et al., 2000].*

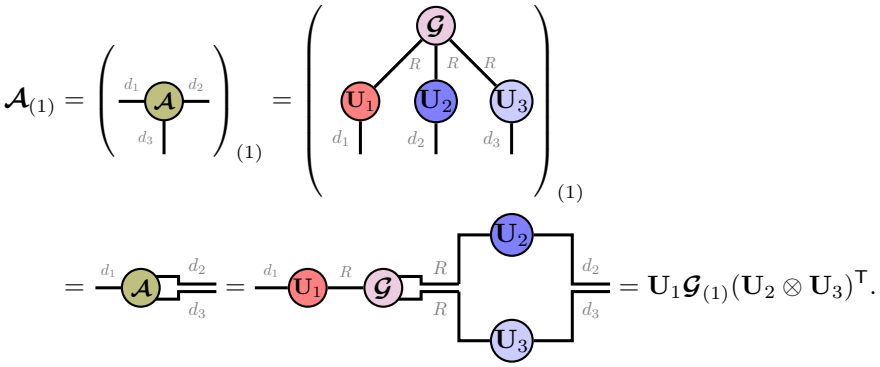
8. *For an* N*-th order* d*-dimensional tensor, the number of parameters in its Tucker decomposition is* $\mathcal{O}(R^N + NdR)$ *assuming* $R_1 = \cdots = R_N = R$ *and* $d_1 = \cdots = d_N = d$*.*

9. *We conclude this section by showing how the mode* n *matricization of the Tucker decomposition of a tensor can be expressed as a Kronecker product.*

   Proposition 14. *Let* $\mathcal{A} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N}$*. If* $\mathcal{A} = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \ldots \times_N \mathbf{U}_N$ *then*

$$\mathcal{A}_{(i)} = \mathbf{U}_i \mathcal{G}_{(i)} \left(\mathbf{U}_1 \otimes \ldots \otimes \mathbf{U}_{(i-1)} \otimes \mathbf{U}_{(i+1)} \otimes \ldots \otimes \mathbf{U}_N\right)^{\mathsf{T}}.$$

   Proof. For $N = 3$ and $n = 1$ assuming $R_1 = R_2 = R_3 = R$,



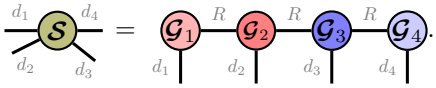
□

3.3 Tensor Train (TT) Decomposition

The Tensor Train (TT) decomposition [Oseledets, 2010] is another popular tensor factorization model. It decomposes an N-th order tensor into N smaller third-order tensors. A rank-$(R_1, \ldots, R_{N-1})$ *tensor train decomposition* of a tensor $\mathcal{S} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ factorizes it into a product of N third-order tensors, $\mathcal{G}_n \in \mathbb{R}^{R_{n-1} \times d_n \times R_n}$ for $n \in [N]$ (with $R_0 = R_N = 1$):

$$\mathcal{S}_{i_1, \cdots, i_N} = \sum_{r_0, \cdots, r_N} \prod_{n=1}^{N} \mathcal{G}_n(r_{n-1}, i_n, r_n) = \mathrm{TT}((\mathcal{G}_n)_{n=1}^{N}),$$

for all $i_1 \in [d_1], \cdots, i_N \in [d_N]$, where each r_n ranges from 1 to R_n, for $n \in [N]$. The *TT-rank* of $\boldsymbol{\mathcal{S}}$ is the smallest $(R_1, R_2, \ldots, R_{N-1})$ such that $\boldsymbol{\mathcal{S}} = \mathrm{TT}((\boldsymbol{\mathcal{G}}_n)_{n=1}^N)$, where $\mathrm{TT}((\boldsymbol{\mathcal{G}}_n)_{n=1}^N)$ represents a TT decomposition with core tensors $\boldsymbol{\mathcal{G}}_1, \ldots, \boldsymbol{\mathcal{G}}_N$. The rank can be viewed as a parameter that controls the expressivity of the TT decomposition. That is with a sufficiently large rank, a TT decomposition can represent any arbitrary tensor. The TT decomposition can be represented in tensor networks, e.g., for a 4-th order tensor:

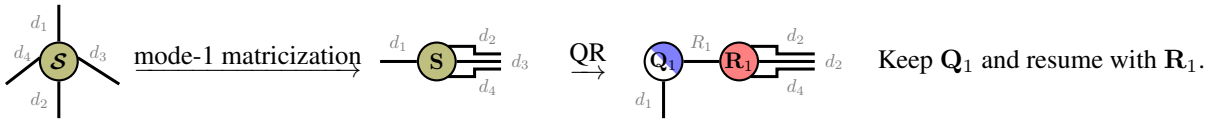


In the physics community, the TT decomposition is also known as a *Matrix Product State (MPS)*, the intermediate edges are referred to as bond dimensions, and the free legs are called physical dimensions. Next, we explain how to construct the TT tensor from a tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ using singular value decomposition (SVD), known as the TT-SVD algorithm [Oseledets, 2010]. The following theorem establishes the existence of a minimal tensor train decomposition for any arbitrary tensor, and its proof is constructive and outlines the TT-SVD algorithm.

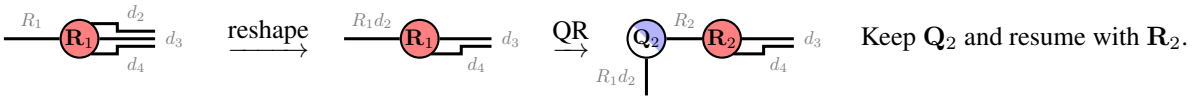
Theorem 15. ***(Computing (Orthogonal) Tensor Train Decomposition.)*** *For any $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$, let $\mathbf{S}_{([n])} \in \mathbb{R}^{d_1 \ldots d_n \times d_{n+1} \ldots d_N}$ be the matricization obtained by mapping the first n modes of $\boldsymbol{\mathcal{S}}$ to the rows of $\mathbf{S}$. Then the TT rank of $\boldsymbol{\mathcal{S}}$, $(R_1, \cdots, R_{N-1})$ is given by $R_n = \mathrm{rank}\,(\mathbf{S}_{([n])})$ for all $n \in [N-1]$.*

Proof. Let $R_n = \mathrm{rank}\,(\mathbf{S}_{([n])})$. We construct the TT decomposition of 4th order tensor $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{d_1 \times \cdots \times d_4}$ by successive QR decomposition as follows.

1.

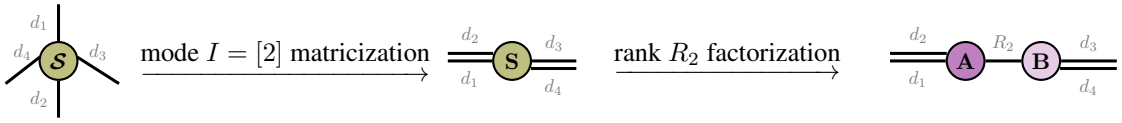


2.

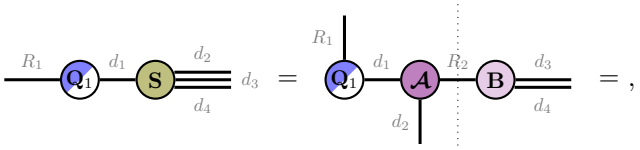


Note that we need to show that $\mathrm{rank}(\mathbf{R}_1) \leq R_2$ for the rank R_2 exact QR decomposition of $\mathbf{R}_1$ to exist. First observe that since $\mathrm{rank}(\mathbf{S}_{([2])}) = R_2$, there exist $\mathbf{A} \in \mathbb{R}^{d_1 d_2 \times R_2}, \mathbf{B} \in in\mathbb{R}^{R \times d_3 d_4}$ such that:

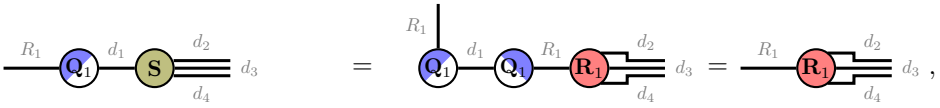
3.



By contracting $\mathbf{Q}_1$ with the first-mode matricization $\mathbf{S}_{(1)}$ and using the above factorization with reshaping $\mathbf{A}$ to $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{d_1 \times d_2 \times R_2}$, we obtain

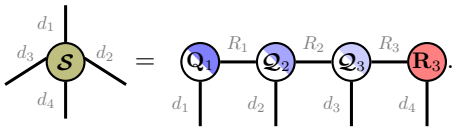


showing that the reshaping of $\mathbf{Q}_1 \mathbf{S}_{(1)}$ into a $R_1 d_2 \times d_3 d_4$ matrix has rank at most R_2. At the same time, since $\mathbf{Q}_1$ is orthogonal and comes from the QR factorization of $\mathbf{S}_{(1)}$, have that $\mathbf{Q}_1 \mathbf{S}_{(1)}$ is equal to $\mathbf{R}_1$:



showing indeed that the rank of $\mathbf{R}_1$ after being reshaped into a $R_1 d_2 \times d_3 d_4$ matrix is at most R_2.

By repeating steps 2 and 3 in this fashion for $\mathbf{R}_2$ and $\mathbf{R}_3$, we ultimately obtain a TT decomposition with the rank (R_1, R_2, R_3) where $R_n = \text{rank}(\mathbf{S}_{([n])})$ for $n \in [3]$, i.e.,

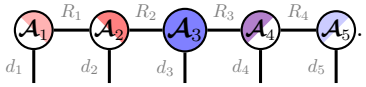


resulting in a TT decomposition where all cores are orthogonal except for the last core, which is $\mathbf{R}_3$. □

We can also compute an orthogonal TT decomposition through the truncated SVD which is called the TT-SVD algorithm (see e.g., [Oseledets, 2010]).
Now, we define the notions of left orthogonality, right orthogonality and the canonical form of the TT decomposition.

Definition 4. *A core tensor $\mathcal{A}_n \in \mathbb{R}^{R_{n-1} \times d_n \times R_n}$ for $n \in [N]$ is left-orthogonal if $(\mathcal{A}_n)^{\mathsf{T}}_{(3)}(\mathcal{A}_n)_{(3)} = \mathbf{I}_{R_n}$ and right-orthogonal if $(\mathcal{A}_n)_{(1)}(\mathcal{A}_n)^{\mathsf{T}}_{(1)} = \mathbf{I}_{R_{n-1}}$.*

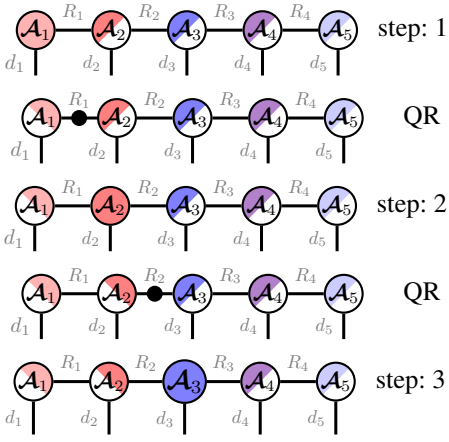
Definition 5. *The TT decomposition $\text{TT}((\mathcal{A}_n)_{n=1}^N) \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ is said to be in canonical format with respect to a fixed index $j \in [N]$ if all cores with $n < j$ are left-orthogonal and all cores with $n > j$ are right-orthogonal, e.g., the following TT decomposition is in canonical form w.r.t. its 3rd core tensor:*



The cores at the left side of $\mathcal{A}_3$ are left-orthogonal and the cores at the right are right-orthogonal. The core $\mathcal{A}_3$ is called the center of orthogonality. Note that any TT decomposition can efficiently be converted to a canonical form w.r.t. any index $j \in [N]$ by performing a series of QR decompositions on the core tensors [Holtz et al., 2012, Evenbly, 2018].

Another popular way to obtain a TT decomposition is using Alternating Least Square (ALS) method:

Computing TT with Alternating Least Square (ALS) The single-site Tensor Train Alternating Least Squares (TT-ALS) method [Holtz et al., 2012] starts with a TT decomposition in canonical form, initialized with a crude approximation (e.g., with random cores, or using TT-SVD). In the first step, the core $\mathcal{A}_1$ of the decomposition is non-orthogonal. During sweeps from left to right (respectively right to left), the algorithm keeps all cores fixed except for the nth one, $\mathcal{A}_n$ and optimizes it accordingly. After updating $\mathcal{A}_n$, a QR decomposition is applied, and the non-orthonormal part is merged to the left (or right, depending on the direction of the sweep), a step which is called *core orthogonalization*. A Half-sweep of TT-ALS is presented above. In each non-QR step, the fully colored core is



optimized and in each QR step the non-orthogonal component (depicted by a black circle) is absorbed into the next core. This procedure repeats until reaching the right side of the decomposition, and then the same procedure is repeated from the right until reaching the left side (not demonstrated in this figure).

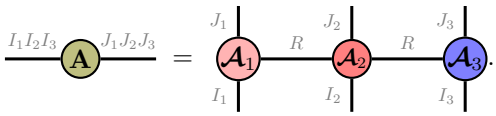
In the next section, we first introduce the TT-matrix, also known as the Matrix Product Operator (MPO), and then demonstrate how to perform operations efficiently in the TT format.

3.4 Efficient Operations in TT Format.

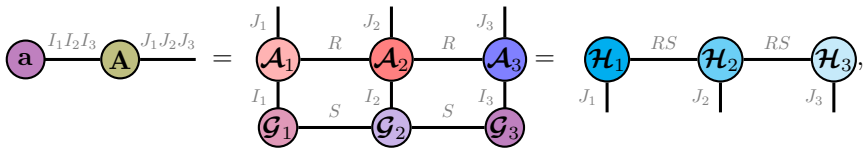
Matrix Product Operator (MPO) As a generalization of the TT decomposition, we introduce the Matrix Product Operator (MPO) [Oseledets, 2010]. An MPO, also known as a TT-matrix, is a chain of four-way tensors used to represent a matrix. It was originally developed to describe operators acting on multi-body quantum systems. Simply put, an MPO is a method of representing a matrix using tensors. Suppose that we have a matrix of size $\mathbf{A} \in \mathbb{R}^{I_1 I_2 \dots I_N \times J_1 J_2 \dots J_N}$. For $n \in [N]$, let $\boldsymbol{\mathcal{A}}_n \in \mathbb{R}^{R_{n-1} \times I_n \times J_n \times R_n}$ with $R_0 = R_N = 1$ and $R_1 = \cdots = R_{N-1} = R$. A rank R MPO decomposition of $\mathbf{A}$ is given by

$$\mathbf{A}_{i_1 i_2 \cdots N, j_1 j_2 \dots j_N} = (\boldsymbol{\mathcal{A}}_1)_{i_1, j_1, :} (\boldsymbol{\mathcal{A}}_2)_{:, i_2, j_2, :} \cdots (\boldsymbol{\mathcal{A}}_{N-1})_{:, i_{N-1}, j_{N-1}, :} (\boldsymbol{\mathcal{A}}_N)_{:, i_N, j_N},$$

for all indices $i_1 \in [I_1], \cdots, i_N \in [I_N]$ and $j_1 \in [J_1], \dots, j_N \in [J_N]$; we will use the notation $\mathbf{A} = \mathrm{MPO}((\boldsymbol{\mathcal{A}}_n)_{n=1}^N)$ to denote the MPO format. The MPO decomposition for a matrix $\mathbf{A} \in \mathbb{R}^{I_1 I_2 I_3 \times J_1 J_2 J_3}$ in a tensor network notation can be represented by:



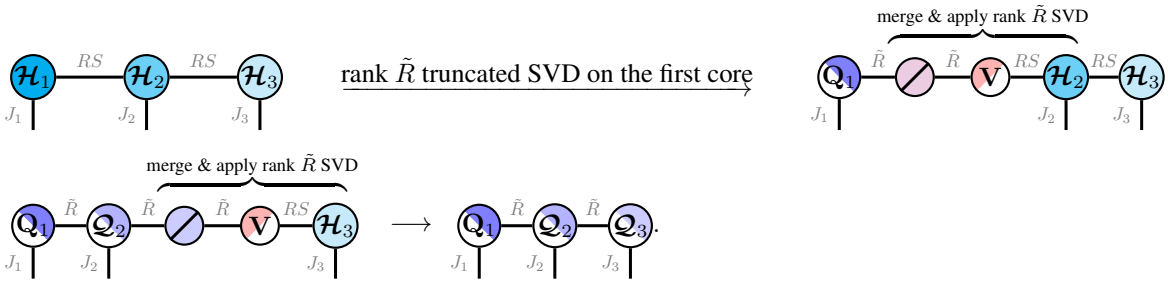
Mat-vec Product in MPO. The product between a matrix $\mathbf{A} \in \mathbb{R}^{I_1 I_2 I_3 \times J_1 J_2 J_3}$ and a vector $\mathbf{a} \in \mathbb{R}^{I_1 I_2 I_3}$ given in TT format can be computed efficiently in the TT format directly, e.g.,



(10)

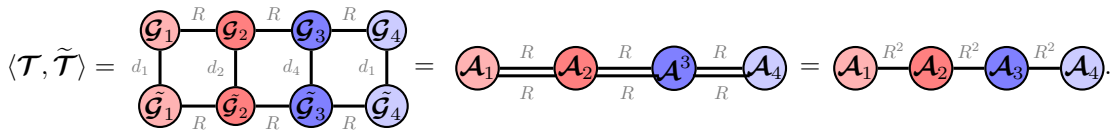
where the final tensor is a TT vector of rank RS since multiplication of two TT tensors increases the rank to the multiplication of ranks. As one can see, when performing operations in the TT format, the TT ranks may increase. TheTT-rounding algorithm [Oseledets, 2010] can be used to reduce the ranks when needed (at the cost of some approximation error).

TT-rounding During operations in TT format (e.g., summation, multiplication, etc.), the rank of the resulting tensor tends to increase. To control this growth while maintaining accuracy, we can reduce the rank. This is done by taking the TT tensor obtained from Eqn. (10) and applying SVD to each core. To obtain the rank $\tilde{R} \leq RS$ TT decomposition, we can apply truncated SVD with rank $\tilde{R}$ on each core of the tensor obtained from Eqn. (10):



One can show that this procedure is *exactly* equivalent to performing TT-SVD on the tensor $\mathrm{TT}(\boldsymbol{\mathcal{G}}_1, \boldsymbol{\mathcal{G}}_2, \boldsymbol{\mathcal{G}}_3)$.

Inner Product As mentioned in Section 1.1, for both tensors and vectors, the inner product is obtained by connecting all corresponding indices. Suppose that we have two fourth order tensors $\boldsymbol{\mathcal{T}} = \mathrm{TT}((\boldsymbol{\mathcal{G}}_n)_{n=1}^N)$, $\widetilde{\boldsymbol{\mathcal{T}}} = \mathrm{TT}((\tilde{\boldsymbol{\mathcal{G}}}_n)_{n=1}^N) \in \mathbb{R}^{d_1 \times \dots \times d_4}$ in the TT format. Then, the inner product can be computed efficiently, e.g.,

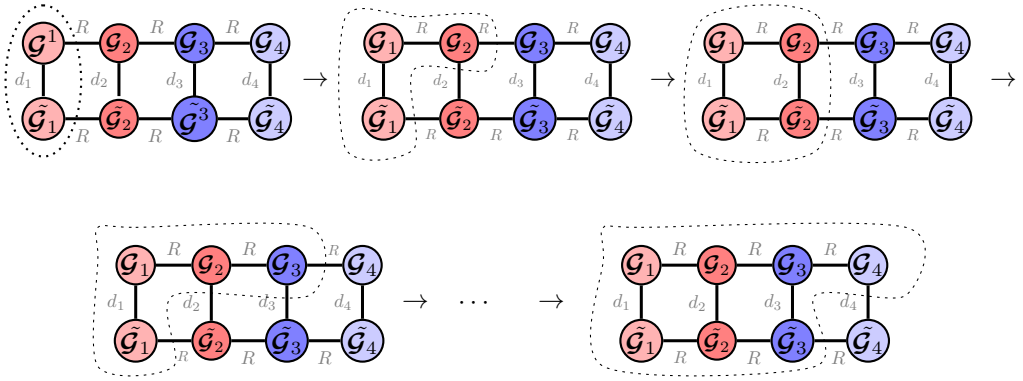


(11)

We can see that this representation is correct by simply using the definition,

$$\langle \boldsymbol{\mathcal{T}}, \widetilde{\boldsymbol{\mathcal{T}}} \rangle = \sum_{i_1=1}^{d_1} \cdots \sum_{i_4=1}^{d_4} \boldsymbol{\mathcal{T}}_{i_1,\dots i_4} \tilde{\boldsymbol{\mathcal{T}}}_{i_1,\dots i_4}.$$

Observe that the complexity of computing the inner product of two N-th order tensors in TT format with the process described in Eqn. (11) is $\mathcal{O}(NdR^4)$ provided that $d_1 = \cdots = d_N = d$ and $R_2 = \cdots = R_{N-1} = R$. This represents a significant improvement compared to the standard method, which has a complexity of $\mathcal{O}(d^N)$. Therefore, the TT format provides an efficient and practical approach for performing operations on high-dimensional tensors. Moreover, cores can be contracted in an even more efficient manner, e.g.,



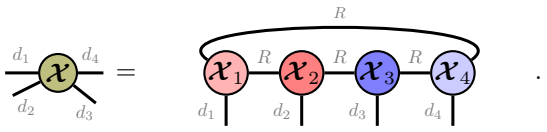
As we can see above, the complexity of computing the inner product between two vectors of size $\mathcal{O}(d^N)$ can be reduced to $\mathcal{O}(NdR^3)$. In general, determining the optimal order of contraction for an arbitrary tensor network is an NP-hard problem [Chi-Chung et al., 1997].
As expected, several other tensor networks are not covered in this chapter. In the next section, we introduce some of them, which are notable generalizations of the Tucker and TT decompositions.

3.5 Other decompositions: Tensor Ring, PEPS & Hierarchical Tucker

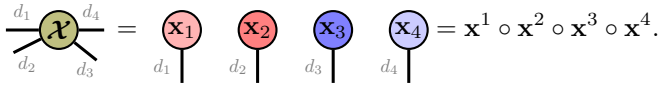
Tensor Ring Decomposition The Tensor Ring (TR) decomposition is a generalization of the TT decomposition [Zhao et al., 2016]. Originally introduced in quantum physics, it has recently gained popularity in the machine learning community [Wang et al., 2017, 2018, Yuan et al., 2018]. While the TR decomposition is known to have some numerical stability issues, it generally requires less storage and achieves better compression ratios than the TT decomposition in practice. Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ be an N-th order tensor. Let $\boldsymbol{\mathcal{X}}_n \in \mathbb{R}^{R_{n-1} \times d_n \times R_n}$ for $n \in [N]$ be the core tensors. For simplicity let $R_0 = R_1 = \cdots = R_{N-1} = R_N = R$. A rank R *tensor ring decomposition* of the tensor $\boldsymbol{\mathcal{X}}$ is given by

$$\boldsymbol{\mathcal{X}}_{i_1,\dots,i_N} = \sum_{r_0=1}^{R} \cdots \sum_{r_n=1}^{R} (\boldsymbol{\mathcal{X}}_1)_{r_0,i_1,r_1} (\boldsymbol{\mathcal{X}}_2)_{r_1,i_2,r_2} \cdots (\boldsymbol{\mathcal{X}}_{N-1})_{r_{N-2},i_{N-1},r_{N-1}} (\boldsymbol{\mathcal{X}}_N)_{r_{N-1},i_N,r_N},$$

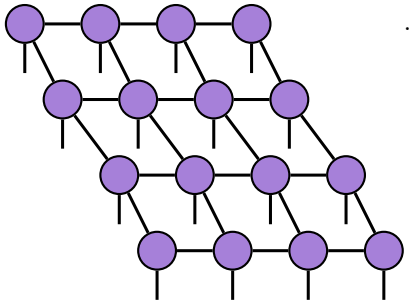
for all indices $i_1 \in [d_1], \cdots, i_N \in [d_N]$. The TR decomposition can be represented in a tensor network notation for a fourth order tensor as follows:



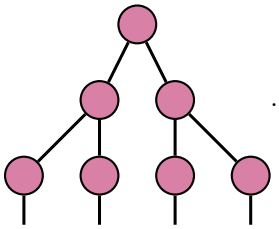
As a special case of tensor ring decomposition, we can also obtain a rank-one decomposition by setting the ranks equal to one:



Projected Entangled Pair States (PEPS) The family of Projected Entangled Pair States (PEPS) have emerged as one of the generalization of the TT decomposition to higher dimension in quantum physics community [Verstraete and Cirac, 2004, Orús, 2014]. PEPS are tensor networks structured as two-dimensional arrays of fifth-order core tensors. Below is the PEPS tensor diagram corresponding to a 4×4 square lattice:



Tree Tensor Networks/ Hierarchical Tucker Decomposition Another tensor network of particular interest is the Tree Tensor Networks (TTN) or Hierarchical Tucker (HT) decomposition, originally introduced in [Hackbusch and Kühn, 2009, Grasedyck, 2010]. A key advantage of this tensor network is its higher expressive power compared to simpler models like the CP decomposition. The tensor network diagram for the HT decomposition of fourth order tensor is shown below:



4 Computing Gradients with Tensor Networks

Optimizing tensor networks in a general setting is a significant challenge across many research areas. While optimization techniques for two-dimensional matrices have achieved considerable success, extending these methods to tensor networks in three or more dimensions remains difficult [Liao et al., 2019]. This complexity stems from the high computational cost of tensor contractions and the lack of efficient optimization algorithms for higher-dimensional cases. Additionally, manually computing gradients using the chain rule is feasible only for specifically designed and simple tensor network structures [Wang et al., 2011]. In this chapter, we introduce an elegant and intuitive approach for efficiently computing higher-order derivatives using tensor network graphical notations efficiently.
In the next section, we first define the classical concepts of gradients and the Jacobian for functions that take vectors as inputs. We then extend these concepts to functions that take tensors as inputs and produce tensors as outputs.

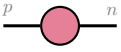
4.1 Jacobians

We first recall the classical notions of gradient and Jacobian of functions defined over $\mathbb{R}^n$.

Definition 6. *For $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^p$ the gradient of f and the Jacobian of g at $\theta \in \mathbb{R}^n$ are defined by*

$$\nabla_\theta f = \left[\frac{\partial f(\theta)}{\partial \theta_1}, \frac{\partial f(\theta)}{\partial \theta_2}, \ldots, \frac{\partial f(\theta)}{\partial \theta_n}\right]^\mathsf{T} =$$  $$\in \mathbb{R}^n$$

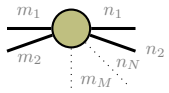
and

$$\frac{\partial g(\theta)}{\partial \theta} = \left(\frac{\partial g(\theta)_i}{\partial \theta_j}\right)_{\substack{i=1,\cdots,p \\ j=1,\cdots,n}} =$$  $$\in \mathbb{R}^{p \times n}.$$

We can naturally extend the notion of Jacobian to functions that take tensors as input, and output tensors (of arbitrary orders).

Definition 7. *If $f : \mathbb{R}^{n_1 \times \cdots \times n_N} \to \mathbb{R}^{m_1 \times \cdots \times m_M}$, then the Jacobian tensor of f at $\theta \in \mathbb{R}^{n_1 \times \cdots \times n_N}$, is the tensor $\frac{\partial f}{\partial \theta} \in \mathbb{R}^{m_1 \times \cdots \times m_M \times n_1 \times \cdots \times n_N}$ defined by*

$$\frac{\partial f(\theta)}{\partial \theta} = \left(\frac{\partial f(\theta)_{i_1,\ldots,i_M}}{\partial \theta_{j_1,\ldots,j_N}} \right)_{\substack{i_1 \in [m_1], \cdots, i_M \in [m_M] \\ j_1 \in [n_1], \cdots, j_N \in [n_N]}} =$$

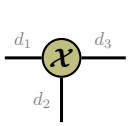


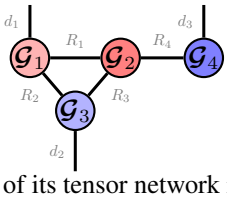
$$\in \mathbb{R}^{m_1 \times \cdots \times m_M \times n_1 \times \cdots \times n_N}.$$

The following theorem demonstrates that first-order derivatives of tensor networks w.r.t. a core tensor can be easily computed using tensor networks.

Theorem 16. *Let $\mathcal{T}$ be a tensor given as a tensor network, where $\mathcal{G}$ is a core tensor appearing only once in the tensor network. Then $\frac{\partial \mathcal{T}}{\partial \mathcal{G}}$ is obtained by removing $\mathcal{G}$ from the tensor network of $\mathcal{T}$.*

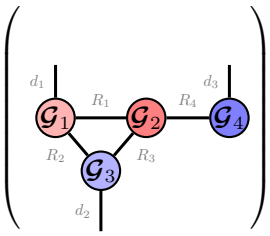
The following examples illustrate how Theorem 16 can be applied to tensor networks.

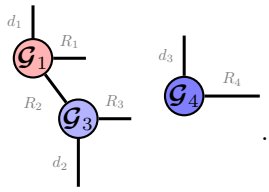
Examples. Let 

$=$


be a tensor network. Observe that we can think of $\mathcal{X}$ as a function of any of the core tensors of its tensor network representation. We can thus naturally ask what the Jacobian of $\mathcal{X}$ is with respect to, e.g., $\mathcal{G}_2$ or $\mathcal{G}_4$. The previous theorem gives us an easy way to compute these derivatives:

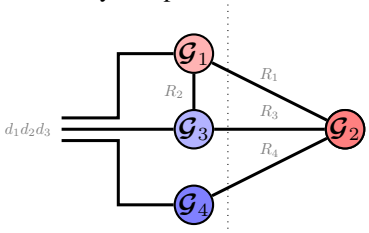
- According to Theorem 16, to find the gradient of $\mathcal{X}$ with respect to $\mathcal{G}_2$, we need to remove the core $\mathcal{G}_2$ from the tensor network, i.e.,

$$\frac{\partial}{\partial \mathcal{G}_2}$$


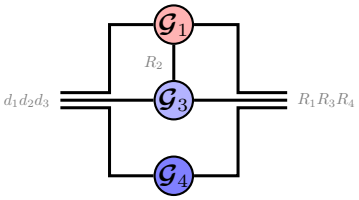
$=$


.

  To understand why this procedure works, we can view the tensor decomposition of $\mathcal{X}$ as a matrix-vector product:

$\mathbf{M}\mathbf{v} =$


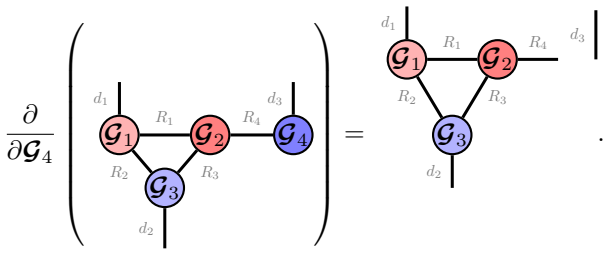
where $\mathbf{v} = \mathrm{vec}(\mathcal{G}_2)$ and $\mathbf{M}$ is the matrix obtained by matricization of the remaining part of the tensor network. Using the classical identity for the Jacobian of a linear map, the expected result is obtained:

$$\frac{\partial \mathbf{M}\mathbf{v}}{\partial \mathbf{v}} = \mathbf{M} =$$


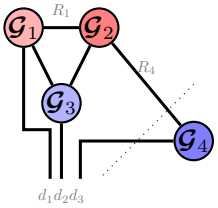
$\in \mathbb{R}^{d_1 d_2 d_3 \times R_1 R_3 R_4}$.

- According to Theorem 16 to find the gradient of $\mathcal{X}$ with respect to $\mathcal{G}_4$, we need to remove the core $\mathcal{G}_4$ from the

corresponding tensor network. i.e.,

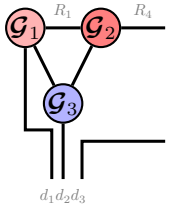


Observe that the dangling leg of $\mathcal{G}_4$ remains in the Jacobian tensor! To understand why this is the case, we can again view the tensor decomposition of $\boldsymbol{\mathcal{X}}$ as a matrix-vector product $\mathbf{Mv}$, such that $\mathbf{Mv} =$



where now $\mathbf{v} = \text{vec}(\mathcal{G}_4)$. Taking derivative of this tensor network with respect to $\mathbf{v} = \text{vec}(\mathbf{G}_4)$ gives us the expected result:

$$\frac{\partial \mathbf{Mv}}{\partial \mathbf{v}} = \mathbf{M} =$$

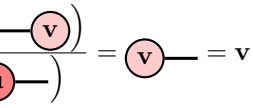
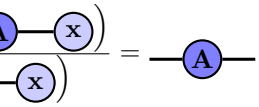
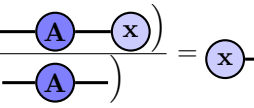
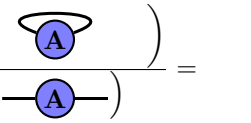
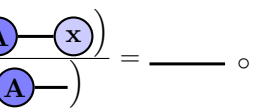


$$\in \mathbb{R}^{d_1 d_2 d_3 \times R_4}.$$

Note that taking the derivative with respect to a tensor removes only the corresponding node from the network, but leaves its connecting edges (legs) intact.

We now illustrate how Theorem 16 can be used to effortlessly recover classical derivatives and gradient computations.

Re-deriving Classical Identities with Tensor Networks

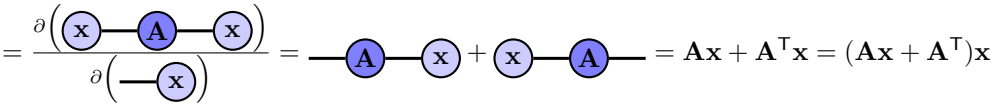
1. $\frac{\partial \langle \mathbf{u}, \mathbf{v} \rangle}{\partial \mathbf{u}} =$  $= \mathbf{v}$
2. $\frac{\partial \mathbf{Ax}}{\partial \mathbf{x}} =$  $= \mathbf{A}$
3. $\frac{\partial \mathbf{x}^\top \mathbf{Ax}}{\partial \mathbf{A}} =$  $= \mathbf{x} \circ \mathbf{x}$
4. $\frac{\partial \, \text{tr}(\mathbf{A})}{\partial \mathbf{A}} =$  $= \mathbf{I}$
5. $\frac{\partial \mathbf{Ax}}{\partial \mathbf{A}} =$  $= \mathbf{I} \circ \mathbf{x}.$

Theorem 16 applies when the tensor with respect to which the derivative is taken appears only once in the network. The following theorem shows how to handle the case where the tensor appears multiple times.

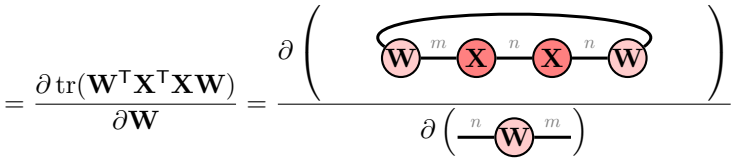
Theorem 17. *Let $\mathcal{T}$ be a tensor network where $\mathcal{G}$ is a core tensor. If $\mathcal{G}$ appears k times in the tensor network of $\mathcal{T}$, then $\frac{\partial \mathcal{T}}{\partial \mathcal{G}}$ is obtained by summing k copies of the tensor network of $\mathcal{T}$, where a different occurrence of $\mathcal{G}$ is removed in each copy.*

The following examples illustrate how Theorem 17 can be applied.

Examples.

- $\frac{\partial \mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x}}{\partial \mathbf{x}} =$  $= \mathbf{A}\mathbf{x} + \mathbf{A}^\mathsf{T}\mathbf{x} = (\mathbf{A}\mathbf{x} + \mathbf{A}^\mathsf{T})\mathbf{x}$
- $\mathbf{X} \in \mathbb{R}^{n\times m}$, $\mathbf{W} \in \mathbb{R}^{m\times n}$

$$\frac{\partial \left\|\mathbf{X}\mathbf{W} - \mathbf{Y}\right\|_F^2}{\partial \mathbf{W}} = \frac{\partial \operatorname{tr}(\mathbf{W}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{W})}{\partial \mathbf{W}} =$$



$$= 2\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{W}.$$

We conclude this chapter by presenting several applications of tensor network gradients.

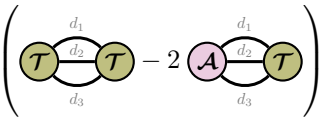
4.2 Applications

There is a wide range of applications for tensor network gradients, we highlight two notable examples here.

We first briefly introduce the notion of loss functions and explain how the chain rule can be easily applied with expressions involving tensors and Jacobian tensors.

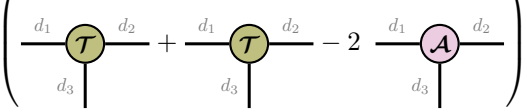
A loss function $L : S \times S \to \mathbb{R}^{\geq 0}$ serves to measure the error or cost incurred by a computational model, algorithm, or decision process when producing the output $\hat{y} \in S$ as the prediction of the ground truth $y \in S$. The goal of many learning and optimization algorithms is to minimize this loss. One approach is to use the Frobenius norm of the difference between the predicted output and the ground truth. To minimize the Frobenius norm when both the input and output are tensors, Jacobians can be used to compute the necessary gradients. Let $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{d_1\times d_2\times d_3}$ be the ground turth and $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{d_1\times d_2\times d_3}$ be the predicted output. To minimize the loss $L = \|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{T}}\|_F^2$ and by using Theorem 16 we can write:

$$\frac{\partial \left\|\boldsymbol{\mathcal{A}} - \boldsymbol{\mathcal{T}}\right\|_F^2}{\partial \boldsymbol{\mathcal{T}}} = \frac{\partial}{\partial \boldsymbol{\mathcal{T}}}\left(\|\boldsymbol{\mathcal{A}}\|_F^2 + \|\boldsymbol{\mathcal{T}}\|_F^2 - 2\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{T}}\rangle\right) = \frac{\partial}{\partial \boldsymbol{\mathcal{T}}}\left(\right.$$

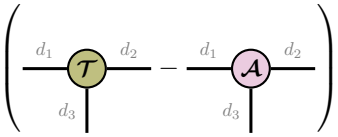


$$\left.\right)$$

$$=$$



$$= 2$$



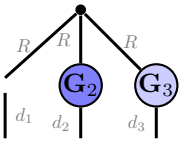
.

We are now ready to present the two notable applications of tensor network gradient computations.

- **CP Decomposition with Gradient Descent** To compute the CP decomposition (see Section 3.1) of an arbitrary tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{d_1\times\cdots\times d_N}$ one can use gradient descent by defining the *Loss* function as $L = \|\boldsymbol{\mathcal{T}} - [\![\mathbf{G}_1, \cdots, \mathbf{G}_N]\!]\|_F^2$. Here $[\![\mathbf{G}_1, \cdots, \mathbf{G}_N]\!]$ denotes the CP decomposition where $\mathbf{G}_1, \cdots, \mathbf{G}_N$ are the factor matrices. To optimize w.r.t. the factor matrices, we can apply gradient descent to the loss function with respect to them. For instance,

in the case of $N = 3$, to compute $\mathbf{G}_1$ we can apply the chain rule and write $\frac{\partial L}{\partial \mathbf{G}_1} = \frac{\partial L}{\partial [\![\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3]\!]} \frac{\partial [\![\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3]\!]}{\partial \mathbf{G}_1}$, where $\frac{\partial L}{\partial [\![\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3]\!]} = 2(\boldsymbol{\mathcal{T}} - [\![\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3]\!]) \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and

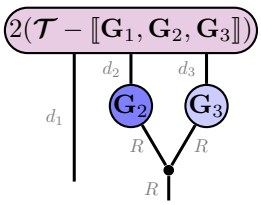
$$\frac{\partial [\![\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3]\!]}{\partial \mathbf{G}_1} =$$



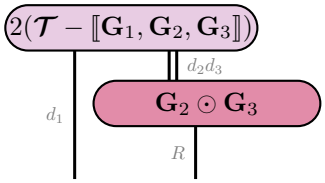
$$\in \mathbb{R}^{d_1 \times d_2 \times d_3 \times R}.$$

Therefore, by applying first-mode matricization and using the Khatri-Rao product, we obtain:

$$\frac{\partial L}{\partial \mathbf{G}_1} = \frac{\partial L}{\partial [\![\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3]\!]} \frac{\partial [\![\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3]\!]}{\partial \mathbf{G}_1} =$$

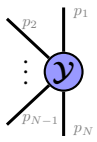


$$= 2(\boldsymbol{\mathcal{T}} - [\![\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3]\!])_{(1)}(\mathbf{G}_2 \odot \mathbf{G}_3) =$$

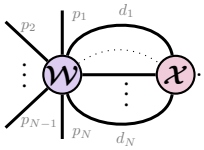


$$\in \mathbb{R}^{d_1 \times R}.$$

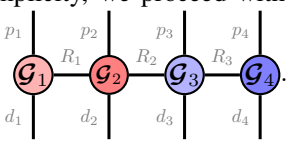
- **Machine Learning** Without nonlinear activation functions, dense neural networks reduce to a series of matrix multiplications. However, due to memory constraints, the dense weight matrices of fully connected layers can be parameterized using tensor factorizations, making them more efficient and easier to work with [Novikov et al., 2015]. Let $\mathbf{W} \in \mathbb{R}^{P \times D}$ be the weight matrix and $\mathbf{x} \in \mathbb{R}^D$ the input vector of a neural network. The output $\mathbf{y} \in \mathbb{R}^P$ is obtained by contracting the weight matrix $\mathbf{W}$ with the input vector $\mathbf{x}$. Suppose $P = \Pi_{i=1}^N p_i$ and $D = \Pi_{i=1}^N d_i$. Then by reshaping $\mathbf{W}$ and $\mathbf{x}$ in tensors we have



$$= \sum_{d_1, \cdots, d_N} \boldsymbol{\mathcal{W}}_{p_1 \cdots p_N, d_1 \cdots d_N} \boldsymbol{\mathcal{X}}_{d_1 \cdots d_N} =$$

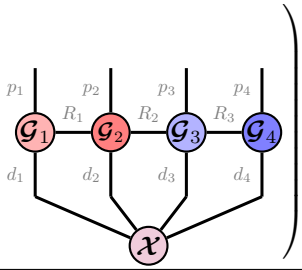


For simplicity, we proceed with the case $N = 4$. Let $\boldsymbol{\mathcal{W}}$ be in the MPO format (see Section 3.4). i.e., $\boldsymbol{\mathcal{W}} =$

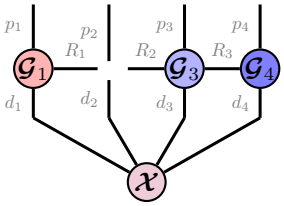


. During the backpropagation step, since we aim to update the weight matrix, we need to optimize it with respect to all the core tensors. According to the chain rule, this gives: $\frac{\partial L}{\partial \boldsymbol{\mathcal{G}}_i} = \frac{\partial L}{\partial \boldsymbol{\mathcal{Y}}} \frac{\partial \boldsymbol{\mathcal{Y}}}{\partial \boldsymbol{\mathcal{G}}_i}$ for $i \in [4]$, where L denotes the *Loss* function. For instance, suppose $i = 2$ and we need to find the derivative of $\boldsymbol{\mathcal{Y}}$ with respect to $\boldsymbol{\mathcal{G}}_2$. By applying Theorem 16 we just need to remove $\boldsymbol{\mathcal{G}}_2$ from the tensor network to obtain the result, i.e.,

$$\frac{\partial \boldsymbol{\mathcal{Y}}}{\partial \boldsymbol{\mathcal{G}}_2} = \frac{\partial \left(\right.}{\partial \boldsymbol{\mathcal{G}}_2}$$



$\left. \right) =$



$$\in \mathbb{R}^{p_1 \times R_1 \times p_2 \times R_2 \times p_3 \times p_4}.$$

5 Probability Distributions and Random Vectors

In this chapter, we explore one of the most useful applications of tensor networks: their ability to intuitively and efficiently represent high-dimensional probability distributions. For example, some tensor network decompositions, such as TT, allow for the efficient encoding of both probability distributions and their marginals [Gardiner and Lopez-Piqueres, 2024, Glasser et al., 2019]. We start this chapter by showing how tensor networks can be used to efficiently model probability distributions. We then shift of our focus to studying statistical properties of tensors obtained by contracting random tensors together. Both parts of this chapter will illustrate how tensor networks can help us easily derive non-trivial results involving probability distributions and high-order tensors.

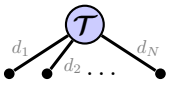
5.1 Probability Distributions and Tensor Decompositions

5.1.1 Probability Distributions as Tensors

One of the popular approaches to modeling a probability distribution with a tensor network is to represent the probabilities directly. Consider a multivariate probability mass function, $\mathbb{P}(X_1, \cdots, X_N)$ of N discrete random variables. This joint probability distribution can be seen as an N-th order tensor with non-negative entries summing to one:

Definition 8. *Let $\mathbb{P}$ be a joint distribution over N discrete random variables $X_1, \cdots, X_N$ taking their values in $[d_1], [d_2], \cdots, [d_N]$, respectively. We say that the tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$, defined by $\mathcal{T}_{i_1 \cdots i_N} = \mathbb{P}(X_1 = i_1, \cdots, X_N = i_N)$,* represents *the probability $\mathbb{P}$, which we denote by $\mathcal{T} \simeq \mathbb{P}(X_1, \cdots, X_N)$. By construction, the entries of $\mathcal{T}$ are non-negative and sums to one:*

$$\sum_{i_1, \cdots, i_N} \mathcal{T}_{i_1, \cdots, i_N} = \quad = 1,$$



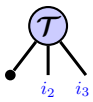
where ●——— *is a vector with all ones (see item 1 in Remark 2).*

Now that this foundation has been established, marginal and conditional probabilities can be expressed using tensor network notation.

Marginal Probabilities Let $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ be the tensor representing the probability distribution $\mathbb{P}(X_1, X_2, X_3)$ then

1. Using the law of total probabilities, the marginal distribution of X_2 and X_3 can be expressed as

$$\mathbb{P}(X_2 = i_2, X_3 = i_3) = \sum_{i_1=1}^{d_1} \mathbb{P}(X_1 = i_1, X_2 = i_2, X_3 = i_3) = \quad = \sum_{i_1=1}^{d_2} \mathcal{T}_{i_1, i_2, i_3},$$



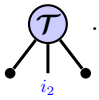
   for all i_2, i_3. This shows that the tensor representing the marginal distribution can be obtained from a simple operation on the tensor representing the joint:

   (TODO add TNs:)

   If [TN of T] $\simeq P(X_1, X_2, X_3)$, then [TN of T contracted with vector of ones] $\simeq P(X_2, X_3)$,

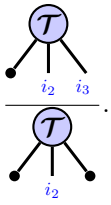
2. We can similarly express the marginal distribution of X_1:

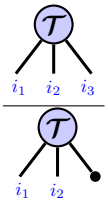
$$\sum_{i_1, i_3} \mathcal{T}_{i_1, i_2, i_3} = \sum_{i_1, i_3} \mathbb{P}(i_1, i_2, i_3) = \mathbb{P}(i_2) = \quad .$$



In the following, we use $\mathbb{P}(i_1, \cdots, i_N)$ for simplicity to denote the probability $\mathbb{P}(X_1 = i_1, \cdots, X_N = i_N)$. As we can see, to represent marginals in tensor network notation, we simply need to contract a vector of all ones with the corresponding legs.
In a similar way we can also represent the conditional probability distributions:

Conditional Probabilities Let $\mathbb{P}(i_n|i_1, \cdots, i_n)$ denotes the conditional probability of $X_n = i_n$ given that the preceding $n-1$ random variables are $i_1, \cdots, i_{n-1}$. For the third order probability tensor $\mathcal{T} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ the conditional probabilities can be presented in tensor network notations:

1. $\mathbb{P}(i_1|i_2) = \frac{\mathbb{P}(i_1,i_2)}{\mathbb{P}(i_2)} = \frac{\sum_{i_3} \mathbb{P}(i_1,i_2,i_3)}{\sum_{i_1,i_3} \mathbb{P}(i_1,i_2,i_3)} =$ 

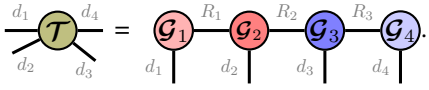
2. $\mathbb{P}(i_3|i_1, i_2) = \frac{\mathbb{P}(i_1,i_2,i_3)}{\mathbb{P}(i_1,i_2)} = \frac{\mathbb{P}(i_1,i_2,i_3)}{\sum_{i_3} \mathbb{P}(i_1,i_2,i_3)} =$ 

5.1.2 Tensor Train Parameterization of Probability Distributions

As we saw in Section 3, working with high-order tensors becomes computationally expensive because the number of elements grows exponentially with the tensor's order. This of course also applies for tensors representing joint probability distributions: the tensor representation grows exponentially with the number of random variables. One way to address this issue is of course to parameterize the probability as a low-rank tensor network!

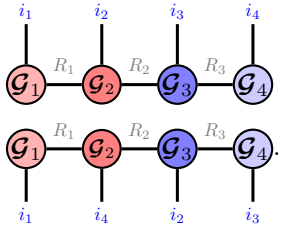
We will show here how this can be done with the tensor train decomposition. This can also be done with other decomposition models, but the tensor train format is particularly interesting and has been very successfully used in quantum physics to model many-body systems **??**.

Let $\mathcal{T} \in \mathbb{R}^{d_1 \times \cdots \times d_N}$ be a probability tensor parameterized in the tensor train format:



Since $\mathcal{T}$ represents a probability distribution, its entries must be non-negative and sum to one. However, it is not clear which constraints one should impose on the core tensors $\mathcal{G}_1, \cdots, \mathcal{G}_N$ to ensure these two properties.

One easy way to ensure non-negativity of the entries of $\mathcal{T}$ is to constrain the core tensors to have non-negative entries. Another way, with its root in quantum physics, is to assume that the entries of $\mathcal{T}$ are the square roots of the probabilities, rather than the probabilities themselves: $\mathcal{T}_{i_1,\cdots,i_N} = \sqrt{\mathbb{P}(X_1 = i_1, \cdots, X_N = i_N)}$. When $\mathcal{T}$ is in the TT format, this corresponds to the following tensor network:

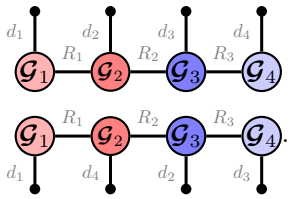
$\mathbb{P}(X_1 = i_1, \cdots, X_4 = i_4) = \mathcal{T}^2_{i_1,\cdots,i_4} =$ 

To ensure that the probabilities sum to one, we can choose to model un-normalized probabilities, since, as we will see, computing the normalization constant is very easy when the probability tensor is in TT format. Now we assume that

$$\mathbb{P}(X_1 = i_1, \cdots, X_N = i_N) = \frac{\mathcal{T}^2_{i_1,\cdots,i_N}}{\zeta}$$

where $\zeta = \sum_{i_1,\cdots,i_n} = \mathcal{T}^2_{i_1,\cdots,i_N}$ is the normalization constant. Similarly to efficient operations in the TT format presented in Section **??**, this normalization factor can be computed efficiently in the TT format:

$$\zeta = \sum_{i_1,\cdots,i_N} \mathcal{T}^2_{i_1,\cdots,i_4} =$$



The idea above is also called *Born Machine* [Glasser et al., 2019, Han et al., 2018].

5.2 Computing Expectations of Random Tensor Networks

We begin this section by presenting two essential propositions about the expected value of the inner product of independent random tensor networks and the outer product of random matrices, where the elements are drawn identically and independently from the standard normal distribution.

Proposition 18. *Let $\mathcal{A}$ and $\mathcal{B}$ be two random and independent tensor networks. Then their inner product in expectation is*

$$\mathbb{E}\left(\mathcal{A} \quad \mathcal{B} \right) = \mathbb{E}(\mathcal{A}) \quad \mathbb{E}(\mathcal{B}).$$

Proof. The result follows from the linearity of the expected value and the independence of $\mathcal{A}$ and $\mathcal{B}$. □

Proposition 19. *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be a random matrix whose elements are drawn identically and independently distributed (i.i.d) from the standard normal distribution. Then the expected value of the outer product of $\mathbf{A}$ with itself is*

$$\mathbb{E}\left(\mathbf{A} \quad \mathbf{A} \right) = \quad ,$$

where the right-hand side is the outer product of two Kronecker delta (i.e., $= \delta_{i_1 i_3} \delta_{i_2 i_4}$ *for $i_1, i_3 \in [m]$ and $i_2, i_4 \in [n]$).*

Proof. Since the entries of $\mathbf{A}$ are drawn independently from $\mathcal{N}(0,1)$, its vecorization follows a mutlivariate normal distribution, $\text{vec}(\mathbf{A}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{mn})$, and thus $\mathbb{E}(\text{vec}(\mathbf{A})\text{vec}(\mathbf{A})^\top) = \mathbf{I}_{mn}$. This means that

$$\mathbb{E}(\mathbf{A}_{i_1 i_2} \mathbf{A}_{i_3 i_4}) = \begin{cases} 1 & \text{if } i_1 = i_3 \text{ and } i_2 = i_4 \\ 0 & \text{otherwise,} \end{cases}$$

hence, $\mathbb{E}(\mathbf{A} \circ \mathbf{A})_{i_1 i_2 i_3 i_4} = \mathbb{E}(\mathbf{A}_{i_1 i_2} \mathbf{A}_{i_3 i_4}) = \delta_{i_1 i_3} \delta_{i_2 i_4} =$, for all $i_1, i_3 \in [m]$ and $i_2, i_4 \in [n]$. □

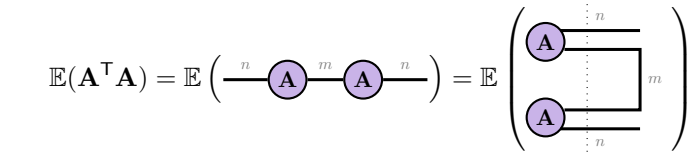
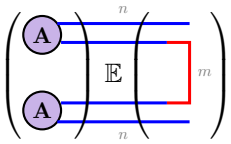
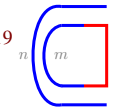
Remark 20. *Note that different orderings of indices of $\mathbb{E}(\mathbf{A} \circ \mathbf{A})$ lead to the following tensor networks:*

- $\mathbb{E}(\mathbf{A} \circ \mathbf{A})_{i_1 i_2 i_4 i_3} = \mathbb{E}(\mathbf{A}_{i_1 i_2} \mathbf{A}_{i_4 i_3}) = \delta_{i_1 i_4} \delta_{i_2 i_3} = \begin{cases} 1 & \textit{if } i_1 = i_4 \textit{ and } i_2 = i_3 \\ 0 & \textit{otherwise,} \end{cases} =$,
- $\mathbb{E}(\mathbf{A} \circ \mathbf{A})_{i_1 i_4 i_2 i_3} = \mathbb{E}(\mathbf{A}_{i_1 i_4} \mathbf{A}_{i_2 i_3}) = \delta_{i_1 i_2} \delta_{i_4 i_3} = \begin{cases} 1 & \textit{if } i_1 = i_2 \textit{ and } i_3 = i_4 \\ 0 & \textit{otherwise,} \end{cases} =$.

Next, we show how Propositions 18 and 19 can be used to derive a very simple tensor network derivation of the expected value of the product of a random matrix with gaussian entries with itself.

Proposition 21. *If $\mathbf{A} \in \mathbb{R}^{m\times n}$ be a random matrix whose elements are drawn i.i.d from the standard normal distribution then $\mathbb{E}(\mathbf{A}^\top\mathbf{A}) = m\mathbf{I}_n$.*

Proof. Using Propositions 18 and 19 we can write

$\mathbb{E}(\mathbf{A}^\top\mathbf{A}) =$  $=$ Prop. 18 $=$  $=$ Prop. 19 $=$  $= m\mathbf{I}_n.$

The final diagram is obtained by contracting the legs of size m. As observed, once the contraction is performed, the resulting expression is simply the trace of the identity matrix, which corresponds to the circle of size m in the equality above. □

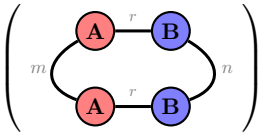
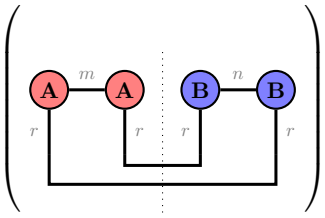
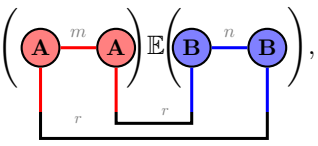
The previous proposition can be interpreted as a statement on the mean of a Wishart distribution. Recall that sampling a matrix $\mathbf{X}$ from the Wishart distribution $W_p(\mathbf{\Sigma}, m)$, where $\mathbf{\Sigma} \in \mathbb{R}^{n\times n}$ is a covariance matrix, is done by sampling each column of a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ from the multivariate normal $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ independently and setting $\mathbf{X} = \mathbf{A}^\top\mathbf{A}$. Hence, the matrix $\mathbf{A}^\top\mathbf{A}$ in Proposition 18 follows the Wishart distribution Wishart distribution $W_p(\mathbf{I}, m)$.

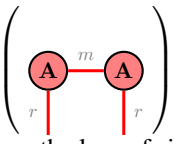
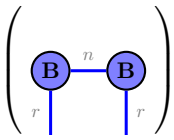
Remark 22. *As a consequence of Proposition 18, if $\mathbf{A} \in \mathbb{R}^{m\times n}$ is a random matrix with entries drawn from the standard normal distribution, then $\mathbb{E}\,\|\mathbf{A}\|_F^2 = mn$. This result can be extended to higher order tensors whose entries are drawn i.i.d. from the standard normal distribution, e.g., if $\mathcal{T} \in \mathbb{R}^{d_1\times d_2\times\cdots\times d_N}$ then $\mathbb{E}\,\|\mathcal{T}\|_F^2 = d_1 d_2 \cdots d_N$.*

By leveraging the properties of random normal matrices from the previous propositions, we can compute the expected norm of their product. Again, the proof of this proposition is very simple using tensor networks and would be more intricate if we were to explicitly use indices and summations instead.

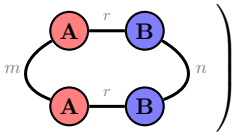
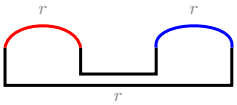
Proposition 23. *Let $\mathbf{A} \in \mathbb{R}^{m\times r}$ and $\mathbf{B} \in \mathbb{R}^{r\times n}$ be random matrices whose entries are independently and identically distributed (i.i.d.) according to the standard normal distribution with mean zero and variance one. Then, the expected value of the squared Frobenius norm of their product is given by $\mathbb{E}\|\mathbf{AB}\|_F^2 = mnr$.*

Proof. Using Proposition 18 we can write

$\mathbb{E}(\|\mathbf{AB}\|_F^2) = \mathbb{E}$  $= \mathbb{E}$  $= \mathbb{E}$  ,

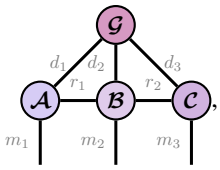
and by Proposition 21 we have, $\mathbb{E}$  $= m$  and $\mathbb{E}$  $= n$  .

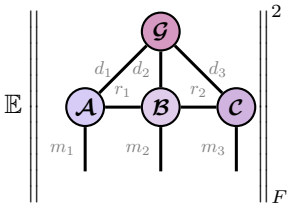
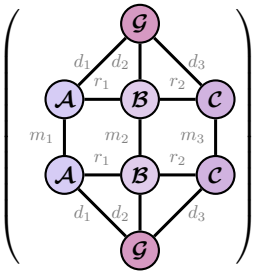
Since there are contractions between the legs of size r we finally obtain

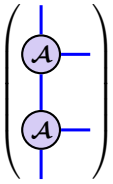
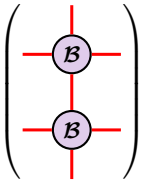
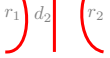
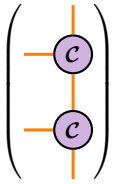
$\mathbb{E}(\|\mathbf{AB}\|_F^2) = \mathbb{E}$  $= mn$  $= mnr.$

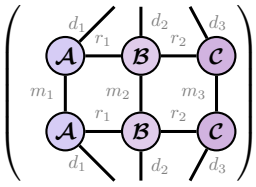
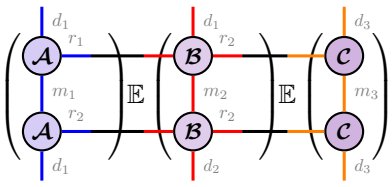
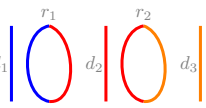
□

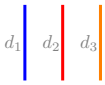
Now that we understand how to compute the expected norm of products of Gaussian random matrices, we can extend this method to evaluate expectations of more complex tensor networks.

Example. Suppose we have an arbitrary tensor network, e.g., $\boldsymbol{\mathcal{A}} =$  , where $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{m_1 \times d_1 \times r_1}$, $\boldsymbol{\mathcal{B}} \in \mathbb{R}^{m_2 \times r_1 \times d_2 \times r_2}$, $\boldsymbol{\mathcal{C}} \in \mathbb{R}^{m_3 \times r_2 \times d_3}$ and $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ are tensors whose elements are drawn i.i.d. from the standard normal distribution, and we want to compute the expectation of $\|\boldsymbol{\mathcal{A}}\|_F^2$. We can write it in tensor network graphical notations:

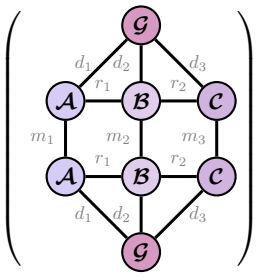
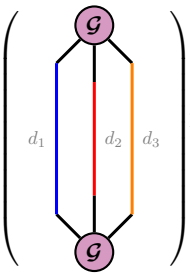
$\mathbb{E}$  $=\mathbb{E}$  ,

by Proposition 21 we know $\mathbb{E}$  $= m_1$  , $\mathbb{E}$  $= m_2$  , and $\mathbb{E}$  $=$ m_3  , and thus by Proposition 18 we can write:

$\mathbb{E}$  $=\mathbb{E}$  $= m_1 m_2 m_3$ 

$= m_1 m_2 m_3 r_1 r_2$  .

Therefore again by Propisition 18 since $\boldsymbol{\mathcal{G}}$ is independent of $\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}$ and $\boldsymbol{\mathcal{C}}$ we can write

$\mathbb{E}$  $= m_1 m_2 m_3 r_1 r_2 \mathbb{E}$  $\overset{\text{Prop. 21}}{=} m_1 m_2 m_3 r_1 r_2 d_1 d_2 d_3$.

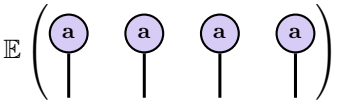
We can see how efficiently complex computations can be performed using tensor networks. We can now introduce more identities involving random Gaussian matrices.

1. The first identity is the result of *Isserlis'* theorem [Isserlis, 1918].

   Theorem 24. *Let* $\mathbf{a} \in \mathbb{R}^n$ *be a random vector whose elements are drawn i.i.d. from the normal distribution with*

mean zero and variance one. Then

$$\mathbb{E}(\mathbf{a}^{\otimes 4}) = \mathbb{E}\left(\right.$$



$$\left.\right)$$

$$=$$

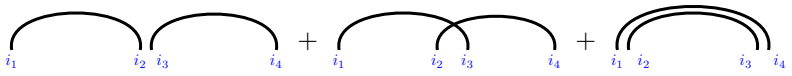
where each term in the sum above corresponds to a different reshaping of the identity matrix. Note that in the theorem above, the zero variance can be replaced with an arbitrary variance σ^2. In that case, each term in the final sum should be multiplied by a factor of σ^2 accordingly.

Proof. Element-wise, by using Isserlis' theorem and using the fact that $\mathbf{a} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, for $i_1, \cdots, i_4 \in [n]$, we have

$$\begin{aligned}(\mathbb{E}[\mathbf{a}^{\otimes 4}])_{i_1,i_2,i_3,i_4} &= \mathbb{E}[\mathbf{a}_{i_1}\mathbf{a}_{i_2}\mathbf{a}_{i_3}\mathbf{a}_{i_4}] = \mathbb{E}[\mathbf{a}_{i_1}\mathbf{a}_{i_2}]\mathbb{E}[\mathbf{a}_{i_3}\mathbf{a}_{i_4}] + \mathbb{E}[\mathbf{a}_{i_1}\mathbf{a}_{i_3}]\mathbb{E}[\mathbf{a}_{i_2}\mathbf{a}_{i_4}] + \mathbb{E}[\mathbf{a}_{i_1}\mathbf{a}_{i_4}]\mathbb{E}[\mathbf{a}_{i_2}\mathbf{a}_{i_3}] \\ &= \delta_{i_1 i_2}\delta_{i_3 i_4} + \delta_{i_1 i_3}\delta_{i_2 i_4} + \delta_{i_1 i_4}\delta_{i_2 i_3},\end{aligned}$$

which in tensor network notation gives:

$$\mathbb{E}(\mathbf{a}^{\otimes 4})_{i_1 i_2 i_3 i_4} =$$

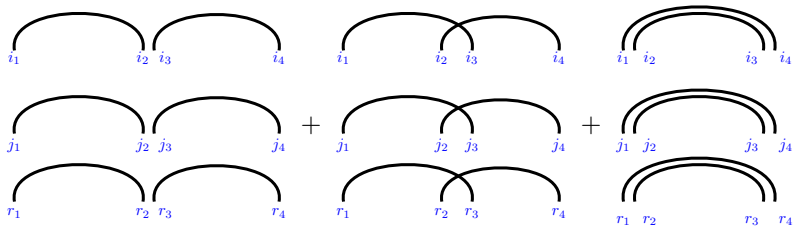


$$.$$

□

We can also extend the result of *Isserlis' theorem* to the case where the random variable takes the form of a higher-order tensor. For example for the third order tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{d_1 \times d_2 \times d_3}$ and $n = 4$ we can write

$$\mathbb{E}(\boldsymbol{\mathcal{A}}^{\otimes 4})_{i_1 \cdots i_4 j_1 \cdots j_4 r_1 \cdots r_4} =$$



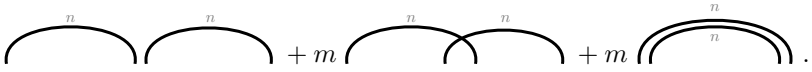
$$,$$

for $j_1, \cdots, i_4 \in [d_1], j_1, \cdots, j_4 \in [d_2]$ and $r_1, \cdots, r_4 \in [d_3]$. As we can see above, we can follow this pattern and find the 4th moments for any order of a tensor.

2. Using the above results, we can derive $\mathbb{E}((\mathbf{A}^\top \mathbf{A})^{\otimes 2})$:

Proposition 25. *Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ be a random matrix whose elements are drawn i.i.d from the standard normal distribution then*

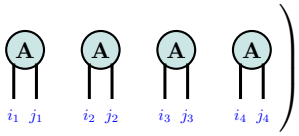
$$\mathbb{E}((\mathbf{A}^\top \mathbf{A})^{\otimes 2})) = m^2$$



$$.$$

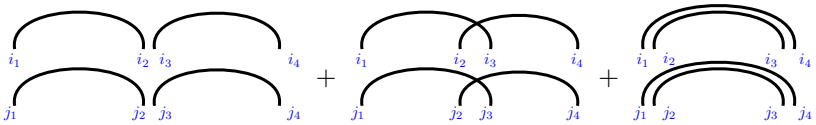
Proof. By applying Theorem 24 to matrices and using Proposition 21, we can derive the desired expression. To present $\mathbb{E}((\mathbf{A}^\top \mathbf{A})^{\otimes 2})$ element-wise using tensor networks, we observe that

$$\mathbb{E}(\mathbf{A}^{\otimes 4})_{i_1,\cdots,i_4,j_1,\cdots,j_4} = \mathbb{E}\left(\right.$$



$$\left.\right)$$

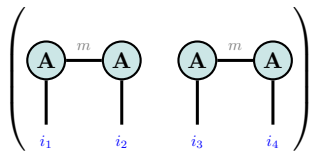
$$=$$



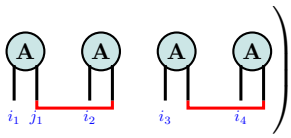
$$,$$

for $i_1, \cdots, i_4 \in [n]$ and $j_1, \cdots, j_4 \in [m]$. It follows that:

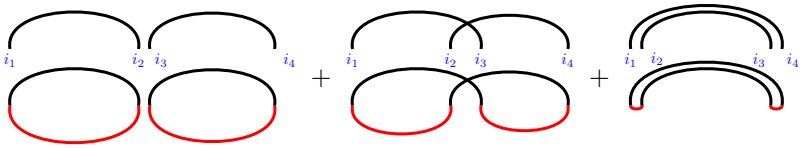
$$\mathbb{E}((\mathbf{A}^\top \mathbf{A})^{\otimes 2})_{i_1 i_2 i_3 i_4} = \mathbb{E}$$



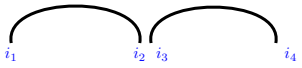
$$= \mathbb{E}$$



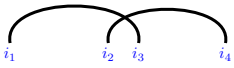
$$=$$



$$= m^2$$



$$+ m$$



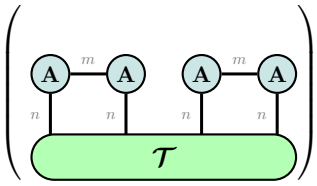
$$+ m$$

.

As we can see since $j_1 = j_2$ and $j_3 = j_4$, the corresponding legs should be contracted in the third equality. Therefore, the circles of size m appear due to the contractions present in the network. □

3. More generally, if $\mathbf{A} \in \mathbb{R}^{m \times n}$ is a random matrix with i.i.d. standard normal entries and $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n \times n \times n \times n}$ is an arbitrary random tensor independent of $\mathbf{A}$, then contracting $\boldsymbol{\mathcal{T}}$ with $\mathbf{A}$ yields:

$$\mathbb{E}$$

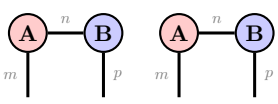


$$= \sum_{i_1 i_2 i_3 i_4} \mathbb{E}(\boldsymbol{\mathcal{T}}_{i_1 i_2 i_3 i_4})(m^2 \delta_{i_1 i_2} \delta_{i_3 i_4} + m \delta_{i_1 i_3} \delta_{i_2 i_4} + m \delta_{i_1 i_4} \delta_{i_2 i_3})$$

$$=$$

4. Let $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times p}$ be two random matrices whose elements are drawn i.i.d from the standard normal distribution, then

$$\mathbb{E}((\mathbf{AB})^{\otimes 2}) = \mathbb{E}$$

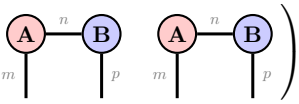


$$= n$$

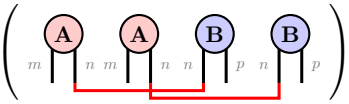
.

Proof. Using Proposition 18 and 19 we can write:

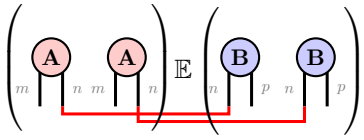
$$\mathbb{E}$$



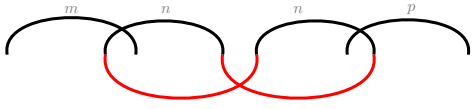
$$= \mathbb{E}$$



$$\overset{\text{Prop. 18}}{=} \mathbb{E}$$



$$\overset{\text{Prop. 19}}{=}$$



$$= n$$

.

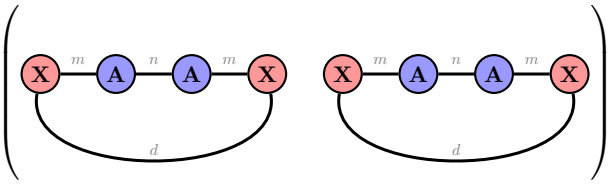
□

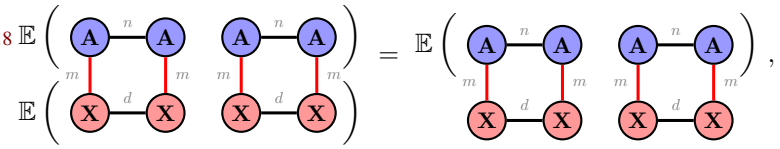
5. The next proposition is a useful formula a special case of Theorem 3.3.15 item (iv) of [Gupta and Nagar, 2018].

Proposition 26. *Let* $\mathbf{A} \in \mathbb{R}^{m\times n}$ *be a random matrix with i.i.d. standard normal entries, and let* $\mathbf{X} \in \mathbb{R}^{d\times m}$ *be a constant matrix then*

$$\mathbb{E}(\mathrm{tr}(\mathbf{X}\mathbf{A}\mathbf{A}^\mathsf{T}\mathbf{X}^\mathsf{T})^2) = n^2\,\|\mathbf{X}\|_F^4 + 2n\,\mathrm{tr}((\mathbf{X}^\mathsf{T}\mathbf{X})^2).$$

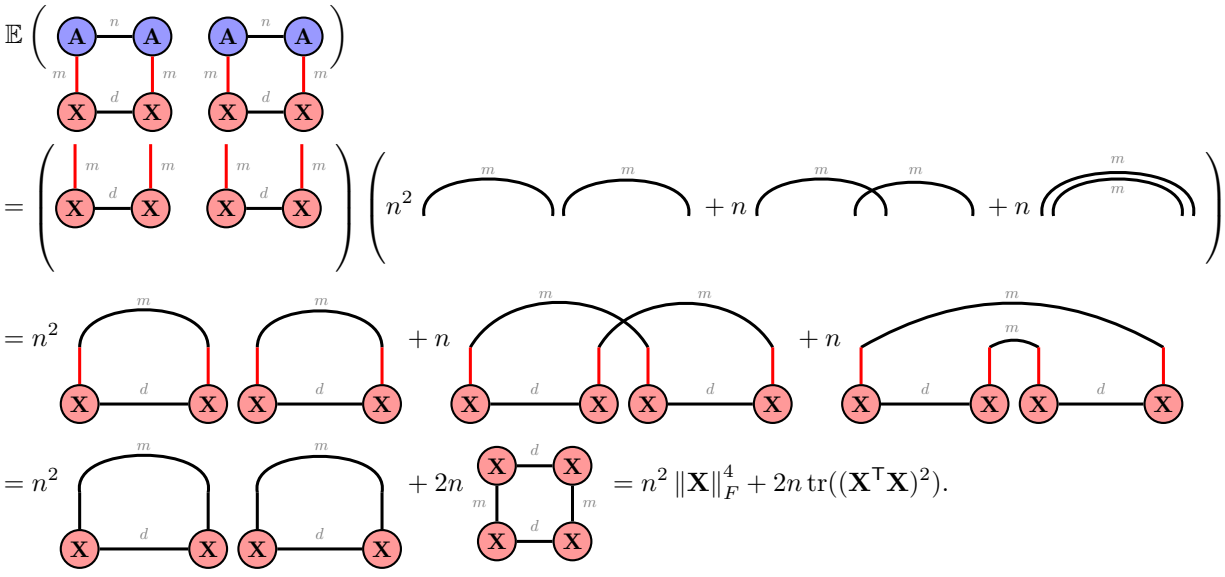
Proof. We prove this identity by using tensor network notations:

$\mathbb{E}(\mathrm{tr}(\mathbf{X}\mathbf{A}\mathbf{A}^\mathsf{T}\mathbf{X}^\mathsf{T})^2) = \mathbb{E}$ 

$\overset{\text{Prop. 18}}{=}$ 

The last equality is obtained by using the fact that the matrix $\mathbf{X}$ is a constant. By Proposition 25 we know:

$\mathbb{E}((\mathbf{A}^\mathsf{T}\mathbf{A})^{\otimes 2})) = n^2$  $+ n$  $+ n$  . Therefore,



□

References

Brett W Bader and Tamara G Kolda. Algorithm 862: Matlab tensor classes for fast algorithm prototyping. *ACM Transactions on Mathematical Software (TOMS)*, 32(4):635–653, 2006.

Jacob Biamonte and Ville Bergholm. Tensor networks in a nutshell. *arXiv preprint arXiv:1708.00006*, 2017.

Rasmus Bro. Multi-way analysis in the food industry, model, algorithms and applications. *Doctoral Thesis, University of Amsterdam*, 1998.

Lam Chi-Chung, P Sadayappan, and Rephael Wenger. On optimizing a class of multi-dimensional loops with reduction for parallel execution. *Parallel Processing Letters*, 7(02):157–168, 1997.

Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

Glen Evenbly. Gauge fixing, canonical forms, and optimal truncations in tensor networks with closed loops. *Physical Review B*, 98(8):085155, 2018.

John Gardiner and Javier Lopez-Piqueres. Tensor network estimation of distribution algorithms. *arXiv preprint arXiv:2412.19780*, 2024.

Ivan Glasser, Ryan Sweke, Nicola Pancotti, Jens Eisert, and Ignacio Cirac. Expressive power of tensor-network factorizations for probabilistic modeling. *Advances in neural information processing systems*, 32, 2019.

Lars Grasedyck. Hierarchical singular value decomposition of tensors. *SIAM journal on matrix analysis and applications*, 31(4):2029–2054, 2010.

Arjun K Gupta and Daya K Nagar. *Matrix variate distributions*. Chapman and Hall/CRC, 2018.

Wolfgang Hackbusch and Stefan Kühn. A new scheme for the tensor representation. *Journal of Fourier analysis and applications*, 15(5):706–722, 2009.

Zhao-Yu Han, Jun Wang, Heng Fan, Lei Wang, and Pan Zhang. Unsupervised generative modeling using matrix product states. *Physical Review X*, 8(3):031012, 2018.

Christopher J Hillar and Lek-Heng Lim. Most tensor problems are np-hard. *Journal of the ACM (JACM)*, 60(6):1–39, 2013.

Frank L Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

Sebastian Holtz, Thorsten Rohwedder, and Reinhold Schneider. The alternating linear scheme for tensor optimization in the tensor train format. *SIAM Journal on Scientific Computing*, 34(2):A683–A713, 2012.

Leon Isserlis. On a formula for the product-moment coefficient of any order of a normal frequency distribution in any number of variables. *Biometrika*, 12(1/2):134–139, 1918.

Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

Tamara Gibson Kolda. Multilinear operators for higher-order decompositions. Technical report, Sandia National Laboratories (SNL), Albuquerque, NM, and Livermore, CA ..., 2006.

Joseph B Kruskal. Rank, decomposition, and uniqueness for 3-way and n-way arrays. In *Multiway data analysis*, pages 7–18. 1989.

Hai-Jun Liao, Jin-Guo Liu, Lei Wang, and Tao Xiang. Differentiable programming tensor networks. *Physical Review X*, 9(3):031041, 2019.

Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. *Advances in neural information processing systems*, 28, 2015.

Román Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of physics*, 349:117–158, 2014.

Ivan V Oseledets. Approximation of 2ˆd\times2ˆd matrices using tensor decomposition. *SIAM Journal on Matrix Analysis and Applications*, 31(4):2130–2145, 2010.

Roger Penrose et al. Applications of negative dimensional tensors. *Combinatorial mathematics and its applications*, 1: 221–244, 1971.

Age K Smilde, Rasmus Bro, and Paul Geladi. *Multi-way analysis: applications in the chemical sciences*. John Wiley & Sons, 2005.

Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

Frank Verstraete and J Ignacio Cirac. Renormalization algorithms for quantum-many body systems in two and higher dimensions. *arXiv preprint cond-mat/0407066*, 2004.

Ling Wang, Ying-Jer Kao, and Anders W Sandvik. Plaquette renormalization scheme for tensor network states. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 83(5):056703, 2011.

Wenqi Wang, Vaneet Aggarwal, and Shuchin Aeron. Efficient low rank tensor ring completion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5697–5705, 2017.

Wenqi Wang, Yifan Sun, Brian Eriksson, Wenlin Wang, and Vaneet Aggarwal. Wide compression: Tensor ring nets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9329–9338, 2018.

Longhao Yuan, Jianting Cao, Xuyang Zhao, Qiang Wu, and Qibin Zhao. Higher-dimension tensor completion via low-rank tensor ring decomposition. In *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1071–1076. IEEE, 2018.

Qibin Zhao, Guoxu Zhou, Shengli Xie, Liqing Zhang, and Andrzej Cichocki. Tensor ring decomposition. *arXiv preprint arXiv:1606.05535*, 2016.