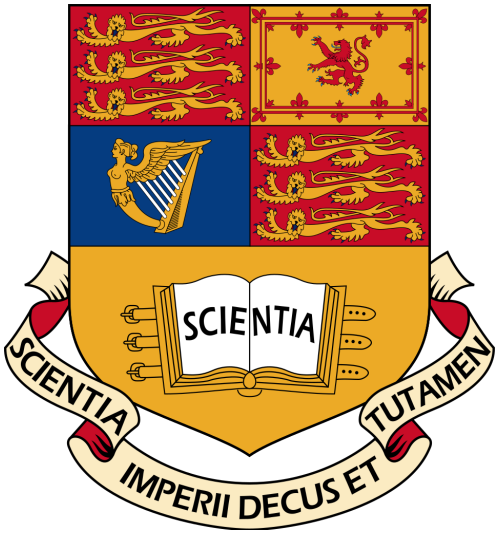
Imperial College London
Department of Electrical and Electronic Engineering



Graph-Regularized Tensor Ensemble Learning for Financial Forecasting - Interim Report

Author:
Ata Yardimci

CID:
01356750

Project Supervisor:
Prof. Danilo Mandic

Second Marker
Prof. Athanassios Manikas

March 29, 2021

Submitted in part fulfilment of the requirements for the degree of
Master of Engineering in Electrical and Electronic Engineering and
the Diploma of Imperial College, June 2021

Abstract

Abstract here

Acknowledgements

I would like to express (the feelings I have) to:

- My supervisor
- My second supervisor
- Other researchers
- My family and friends

Contents

List of Tables

List of Figures

Chapter 1

Project Specification

1.1 Project Definition

Exponentially increasing quantity of data and the development of more powerful and robust processing propels the utilization of machine learning (ML) and signal processing methods in various fields including financial markets. Researchers, in academia and industry, work on different aspects of ML, statistical signal processing and control theory, experimenting with new ideas to improve the state-of-the-art methods. We aim to join this mission with the ideas discussed on this paper while specifically focusing on their applications in financial machine learning and signal processing. The ideas presented, however, are not limited to finance and can be applied to any relevant field.

With financial big data characterized to be multi-dimensional, multi-modal and irregular with very low signal-to-noise ratio, financial signal processing and machine learning comes with many challenges [45]. Figure 1.1 illustrates some challenges of Big Data, which are not just described by high volumes but also by high variety, velocity and veracity. Standard machine learning methods are ill-equipped to tackle the challenges of modern financial big data, in terms of computational complexity and learning generalization, and standard multivariate models are not adequate to account for the structure inherent to real-world markets. This paper aims to introduce a potential solution with the use of graph signal processing techniques, tensors, and

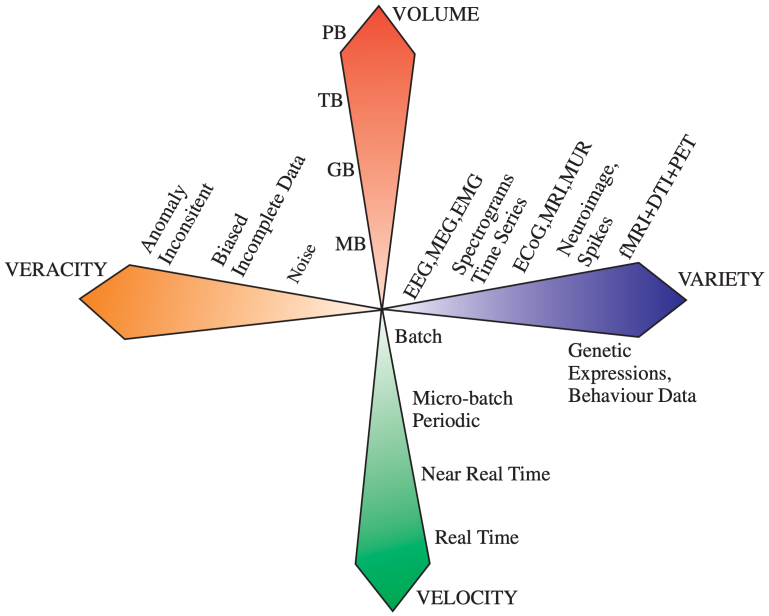


Figure 1.1: Challenges of Big Data [10]

tensor decompositions in a tensor ensemble learning framework [29].

Graphs allow for a rigorous account of irregularly spaced information and incorporate semantic and contextual properties in data. Therefore, graphs are a natural way to represent multi-dimensional and multi-modal financial big data acquired in an irregular domain to model their inherent structural/domain-specific properties. *Graph Signal Processing* (GSP) techniques then generalize traditional signal processing concepts to irregular data domains represented as graphs and are well equipped to exploit the fundamental relations among both the measured data and the underlying graph topology. Graphs are also powerful in vertex clustering and low-dimensional manifold representation with the use of particular techniques such as graph cuts or down-sampling. Finally, by processing data through graph-based filters, it is possible to improve the signal-to-noise ratio of data defined on irregular domains, which can dramatically enhance machine learning applications.

Tensors, which are multi-linear generalizations of vectors and matrices, can be naturally used to model high-dimensional regular lattice-structured graph signals. This approach offers a solution to inadequate modelling of multi-dimensional graphs, through modelling via their corresponding adjacency tensor, allowing to discover intrinsic relations in multi-dimensional data [52].

Tensor Decomposition (TD) techniques, such as Canonical Polyadic Decomposition (CPD), are low-rank factorization methods that can extract latent features from multi-dimensional data, bypassing the bottlenecks imposed by the curse of dimensionality [13]. Through higher-order tensor decomposition, we can develop sophisticated models, capturing multiple interactions and couplings instead of standard pairwise interactions, that is, we can discover hidden components within multi-dimensional data [52].

In this project, we will leverage GSP techniques for robust and regularized modelling of financial big data and utilize the benefits of the tensor framework as an efficient modern signal processing tool to extract economically meaningful latent features from the data. We will also incorporate the Tensor Ensemble Learning (TEL) [29] framework to our work to improve the generalization abilities of our model.

1.2 Related Work

1.2.1 Market Graph and Portfolio Cuts [52]

An application of graph-based financial data analysis is given in [52]: "Investment returns naturally reside on irregular domains, however, standard multivariate portfolio optimization methods are agnostic to data structure. To this end, [52] investigates ways for the domain knowledge to be meaningfully incorporated into the analysis, by means of *portfolio cuts*." [52]

A graph-theoretic portfolio partitioning technique is discussed in [52], which allows to devise robust and tractable asset allocation schemes, by the virtue of a rigorous graph framework for considering smaller, computationally feasible, and economically meaningful clusters of assets, based on graph cuts. This also allows to fully utilize the covariance matrix of asset returns even without the need for its inversion.

The weights of assets in a *minimum-variance* (MV) portfolio [37] is given by

$$\mathbf{w} = \frac{\mathbf{\Sigma}^{-1}\mathbf{1}}{\mathbf{1}^T\mathbf{\Sigma}^{-1}\mathbf{1}}. \tag{1.1}$$

The inversion of matrix $\mathbf{\Sigma}$ in (1.1) may lead to significant errors for ill-conditioned matrices [52]. An approach to promote robustness has been to model assets using *market graphs* [4]. A universe of N assets can naturally be represented as a set of vertices on a market graph, where the edge weight W_{mn} between vertices m and n is defined as the absolute correlation coefficient of their respective assets' returns, i.e.

$$W_{mn} = |\rho_{mn}| = \frac{|\text{cov}\{r_m, r_n\}|}{\sigma_m \sigma_n} \tag{1.2}$$

Therefore, if the assets m and n are statistically independent, $W_{mn} = 0$, and if they are dependent, $W_{mn} > 0$. Note that $\mathbf{W} = \mathbf{W}^T$.

Because the covariance matrix $\mathbf{\Sigma}$ is dense, standard multivariate models implicitly assume full connectivity of the graph and are therefore not adequate to account for the structure inherent to real-world markets [5][6][36]. Therefore, it is desirable to remove unnecessary graph edges to more appropriately model the underlying structure between assets, which can be achieved through vertex clustering. Direct vertex clustering is, however, computationally intractable for large graphs; therefore, we can employ the *minimum cut* vertex clustering method to the graph of portfolio assets, which is a concept referred to as *portfolio cut* in [52]. In this way, smaller graph components can be evaluated quasi-optimally, but in an efficient and rigorous manner.

The proposed graph asset allocation scheme with portfolio cuts in [52] delivered "lower out-sample variance than the standard EW and MV portfolios, thereby attaining a higher Sharpe ratio. This verifies that the removal of possibly spurious statistical dependencies, through portfolio cuts, allows for robust and flexible portfolio constructions." Their approach enable graph-theoretic capital allocation schemes based on measures of connectivity.

1.2.2 Tensor Ensemble Learning [29]

Tensor Ensemble Learning (TEL) is a framework introduced by Kisil in *Tensor Ensemble Learning for Multidimensional Data* [29], which argues that the classical ensemble learning is typically infeasible on multi-dimensional big data applications. [29] introduces a framework

that generalises classic flat-view ensemble learning to multi-dimensional tensor-valued data, achieved by the virtue of tensor decompositions. Kisil shows that the TEL framework naturally compresses multidimensional data in order to take advantage of the inherent multi-way data structure and exploits the benefits of ensemble learning. Kisil's presentation of the framework is available at:

`https://www.youtube.com/watch?v=LjePSyJVly0&ab_channel=CSPImperialDPM`.

1.2.3 Image Representation and Learning With Graph-Laplacian Tucker Tensor Decomposition [25]

Tensor decomposition (TD) models are more powerful in representing the data compared to principal component analysis (PCA) models. However, traditional TD considers attribute information only and discards the pairwise similarity information. Graph-Laplacian tucker tensor decomposition (GLTD), a framework that explores both attributes and pairwise similarity information simultaneously is introduced in [25].

1.2.4 Graph Tensor Networks

Look further into:

- Recurrent Graph Tensor Networks [57]
- Multi-Graph Tensor Networks [56]

Chapter 2

Background

2.1 Financial Signal Processing and Machine Learning

An accurate prediction of asset prices or relevant financial data, i.e. *financial forecasting* has significant benefits in providing a competitive advantage to investors and decision makers. Therefore, *financial signal processing and machine learning* is an important research topic both in academia and industry. Together with significant advantages, financial forecasting comes with many challenges.

Financial big data are characterized to be multi-dimensional and multi-modal, are acquired on irregular domains, and are subject to low signal to noise ratio. These inherent properties bring challenges in terms of computational complexity and create a requirement for robust models in the face of noisy irregular data. Traditional machine learning models, such as neural networks (NN), are non-interpretable black boxes and can easily over-fit to historical financial data, which is yet another big problem in financial forecasting that is explained in detail by Lopez de Prado in [45].

In this paper, we will investigate a solution to these problems by the use of *graph signal processing* (GSP) techniques, *tensors*, and *tensor decompositions* (TD) to create robust models, which can establish links within the underlying structure of the multi-dimensional and multi-modal financial data to extract economically meaningful features.

2.1.1 Financial Terminology

Financial assets can be divided into four major categories: equities, currencies, commodities, and fixed income securities. *Equities* represent ownership in companies and their value is tied to the underlying valuation and future cash-flows of the relevant firm. *Currencies* are traded in pairs and their value is closely linked to the economic outlook of relevant countries. *Commodities* are economic goods that are interchangeable with other goods. Usually, natural resources used as inputs in production of goods and services are referred to as commodities, e.g. copper, crude oil, wheat, etc. Finally, *fixed income securities* are investments that provide returns in the form of fixed periodic interest payments with the eventual return of the initial investment at maturity, e.g. bonds.

There also exists various financial instruments referred to as derivatives, which derive their values – as their names suggest – from their underlying assets. Common derivatives include futures contracts, forwards, options, and swaps. They are often used by firms to speculate price changes and hedge risks. Derivatives are usually leveraged instruments, which increases their potential risks and rewards. Therefore, correct forecasting of price changes could potentially provide high rewards to the investor.

Indices are indicators or statistical measures of changes in a securities market. In the case of financial markets, stock and bond market indices consist of a hypothetical portfolio of securities representing a particular market or a segment of it; you cannot directly invest in an index. They can be used to asses the overall economic activity. For example, the S&P 500 Index is a common benchmark for the U.S. market.

2.1.2 Asset Returns and Portfolios

The asset *total return*, R, and *rate of return*, r, are respectively defined as

$$R = \frac{X_1}{X_0}, \text{ and } r = \frac{X_1 - X_0}{X_0},$$

where X_t is the price of an asset at time t.

Suppose n different assets are available. By apportioning an amount X_0 among the assets, we invest X_{0i} to each asset for $i = 1, 2, \ldots, n$, such that

$$\sum_{i=1}^{n} X_{0i} = X_0.$$

X_{0i} can be expressed as $X_{0i} = w_i X_0$, where w_i is the weight of asset i in the portfolio for $i = 1, 2, \ldots, n$. The asset weight sum to 1, i.e. $\sum_{i=1}^{n} w_i = 1$. If r_i is the rate of return of asset i, the rate of return of the portfolio, r_P, is then given by

$$r_P = \sum_{i=1}^{n} w_i r_i.$$

For any asset, today's value X_0 is deterministic, while the future value X_1 is random. Therefore, the rate of return of an asset and a portfolio are also random. Suppose there are n assets with random rates of return $r_1, r_2, \ldots, r_n$. Then, their expected values are denoted by $\bar{r}_1, \bar{r}_2, \ldots, \bar{r}_n$. The variance of r_i is denoted by σ_i^2, and the covariance of r_i with r_j is denoted by σ_{ij}. Then, the expected return of a portfolio, $\bar{r}_P$, is given by

$$\bar{r}_P = \sum_{i=1}^{n} w_i \bar{r}_i, \tag{2.1}$$

and the variance of the return of a portfolio, σ_P^2 is given by

$$\sigma_P^2 = \sum_{i,j=1}^{n} w_i \sigma_{ij} w_j. \tag{2.2}$$

2.1.3 Mean-Variance Portfolio Theory

The *mean-variance model*, also referred to as the Markowitz Model [37], assumes that investors seek low risk and high reward. The two main components of the mean-variance analysis are the *expected return* and *variance*. The expected return is the estimated return of the investment in the security, while the variance is how spread out the expected return is in terms of probability, indicating the risk of the investment. If two different securities have the same expected return, but one has lower variance, the one with lower variance is the better pick. Similarly, if two different securities have approximately the same variance, the one with the higher return is the better pick. Using (2.1) and (2.2), this idea is formulized as follows:

Given a portfolio expected rate of return target, $\bar{r}_P$, the weights of assets are obtained by solving the following optimisation problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\sum_{i,j=1}^{n} w_i \sigma_{ij} w_j = \frac{1}{2}\sigma_P^2 \\ \text{subject to} \quad & \sum_{i=1}^{n} w_i \bar{r}_i = \bar{r}_P \qquad \text{(exp. return target)} \\ & \sum_{i=1}^{n} w_i = 1 \qquad \text{(weights sum to 1).} \end{aligned}$$

Then, using Lagrange multipliers, we obtain the following objective function L:

$$L = \frac{1}{2}\sum_{i,j=1}^{n} w_i \sigma_{ij} w_j - \lambda\left(\sum_{i=1}^{n} w_i \bar{r}_i - \bar{r}_P\right) - \mu\left(\sum_{i=1}^{n} w_i - 1\right). \tag{2.3}$$

Defining

- $\mathbf{w} = (w_1, w_2, \ldots, w_n) \in \mathbb{R}^n$ as the vector of portfolio weights,

- $\bar{\mathbf{r}} = (\bar{r}_1, \bar{r}_2, \ldots, \bar{r}_n) \in \mathbb{R}^n$ as the vector of exp. asset returns,

- $\mathbf{\Sigma} = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_{nn} \end{bmatrix} \in \mathbb{R}^{n \times n}$ as the covariance matrix of asset returns,

we can write the Markowitz problem in vector notation as

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\mathbf{w}^T \mathbf{\Sigma} \mathbf{w} \\ \text{subject to} \quad & \mathbf{w}^T \bar{\mathbf{r}} - \bar{r}_P = 0 \\ & \mathbf{w}^T \mathbf{1} - 1 = 0. \end{aligned}$$

We can then rewrite the Lagrangian function in (2.3) as

$$L(\mathbf{w}, \lambda, \mu) = \frac{1}{2}\mathbf{w}^T \mathbf{\Sigma} \mathbf{w} - \lambda(\mathbf{w}^T \bar{\mathbf{r}} - \bar{r}_P) - \mu(\mathbf{w}^T \mathbf{1} - 1). \tag{2.4}$$

The optimality conditions for (2.4) are

$$\mathbf{\Sigma}\mathbf{w} - \lambda \bar{\mathbf{r}} - \mu \mathbf{1} = 0, \qquad \bar{\mathbf{r}}^T \mathbf{w} = \bar{r}_P, \qquad \mathbf{1}^T \mathbf{w} = 1.$$

If $\mathbf{\Sigma}$ has full rank and $\bar{r}$ is not a multiple of $\mathbf{1}$, the weights of assets within the portfolio can be obtained by the following calculation:

$$\begin{bmatrix} \mathbf{w} \\ \lambda \\ \mu \end{bmatrix} = \begin{bmatrix} \mathbf{\Sigma} & -\bar{\mathbf{r}} & -\mathbf{1} \\ -\bar{\mathbf{r}}^T & 0 & 0 \\ -\mathbf{1}^T & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{0} \\ -\bar{r}_P \\ -1 \end{bmatrix}. \tag{2.5}$$

For the Minimum-Variance portfolio, the weights can be obtained by

$$\mathbf{w} = \frac{\mathbf{\Sigma}^{-1}\mathbf{1}}{\mathbf{1}^T \mathbf{\Sigma}^{-1} \mathbf{1}}. \tag{2.6}$$

2.2 Graph Signal Processing (GSP)

Graph theory is an established branch of mathematics, roots of which date back to the 18th century and is now a fundamental technique for data analysis across various fields in science and technology. Today, the area of Data Analytics on graphs makes it possible to address completely new paradigms of information and signal processing on graphs, with the abundance of new classes of data sources, which are typically acquired on irregular but structured domains, such as social and web-related networks [51].

In many practical scenarios the signal domain is not designated by equidistant instants in time or a regular grid in a space, e.g., social network modeling or smart grid data domains, which are typically irregular and, in some cases, not even related to the notions of time or space [51]. Therefore, we have to consider properties other than time or space relationships when processing on irregular domains. Furthermore, even if the data is acquired in a well-defined time and space domain, the new contextual and semantic relations between the sensing points, introduced through graphs, equip problem definition with physical relevance, which can provide new insights into analysis and lead to enhanced data processing results [51].

In data domains defined as graphs, the role of classic temporal/spatial sampling points is assumed by the vertices, where the data values are observed, while the edges between the vertices describe the existence and nature of the connections between vertices. This ability to incorporate physically relevant data properties allow graphs to model inherent structural properties of multi-dimensional and multi-modal data and makes graphs well equipped to exploit the fundamental relations among both the measured data and the underlying graph topology.

In this section we first review basic definitions and notations of graphs followed by fundamental graph properties and topologies. Then follows a review of graph spectral analysis, particularly eigendecomposition of graph matrices. Then we do a short review of vertex clustering, graph down-sampling, and determination of the graph topology for practical applications.

2.2.1 Basic Definitions and Notation

Notation used for graphs in this paper is very similar to that proposed in Data Analytics on Graphs Part I [51], which has a very good introduction to basic graph structure. Some of the examples from [51] were used to explain basic graph definitions and properties.

A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{B}\}$ is defined by a set of vertices $\mathcal{V}$ connected by a set of edges $\mathcal{B} \subset \mathcal{V} \times \mathcal{V}$. It can be undirected or directed as depicted respectively in Figure 2.1 (a) and (b) with $N = 8$ vertices, with

$$\mathcal{V} = \{0, 1, 2, 3, 4, 5, 6, 7\}.$$

An edge between vertices 0 and 1 is represented by $(0, 1) \in \mathcal{B}$. The directed graph in 2.1(b) can be described as

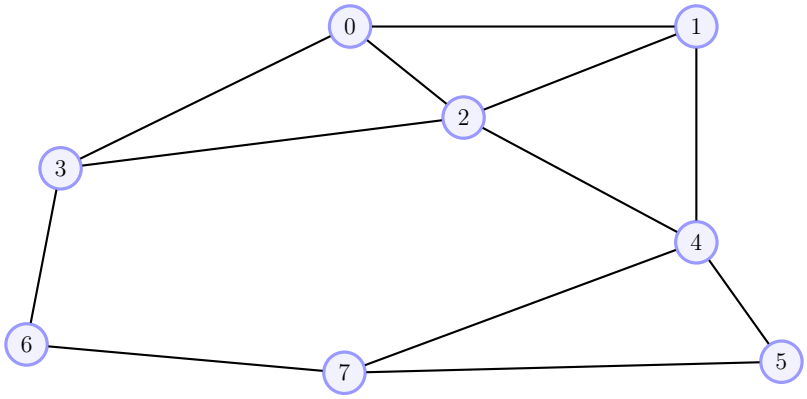
$$\begin{aligned}
&\mathcal{V} = \{0, 1, 2, 3, 4, 5, 6, 7\} \\
&\mathcal{B} \subset \{0, 1, 2, 3, 4, 5, 6, 7\} \times \{0, 1, 2, 3, 4, 5, 6, 7\} \\
&\mathcal{B} = \{(0, 1), (1, 2), (2, 0), (2, 3), (2, 4), (2, 7), (3, 0), \\
&\qquad (4, 1), (4, 2), (4, 5), (5, 7), (6, 3), (6, 7), (7, 2), (7, 6)\}
\end{aligned}$$

If for all vertex n and m $\in \mathcal{V}$, $(n, m) \in \mathcal{B}$ and $(m, n) \in \mathcal{B}$, then the graph is undirected, meaning undirected graphs are special cases of directed graphs.

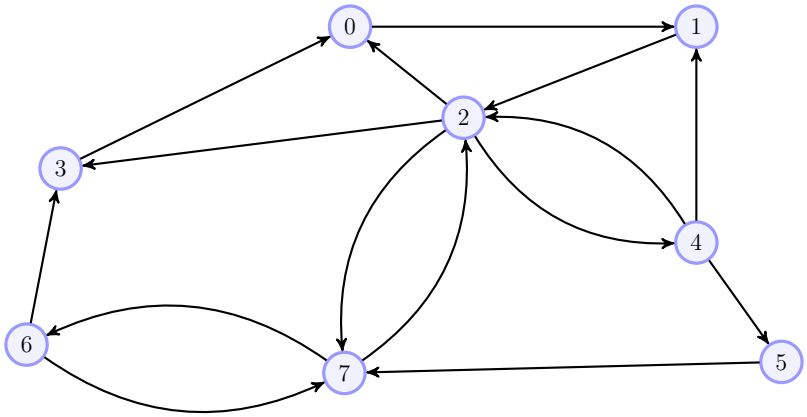
Given $\mathcal{V}$ and $\mathcal{B}$, a graph can be represented by its *adjacency matrix* $\mathbf{A}$, which has dimensions $N \times N$. The entries of an unweighted graph $\mathbf{A}$ are either 0 or 1. The entry A_{mn} is defined as

$$A_{mn} = \begin{cases} 1, & \text{if } (m, n) \in \mathcal{B} \\ 0, & \text{if } (m, n) \notin \mathcal{B}. \end{cases}$$

The adjacency matrices of undirected ($\mathbf{A}_{un}$) and directed ($\mathbf{A}_{dir}$) graphs depicted in Figure 2.1



(a) Undirected



(b) Directed

Figure 2.1: Simple graph structures

are then defined as

$$\mathbf{A}_{un} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}, \quad \mathbf{A}_{dir} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{2.7}$$

Adjacency matrices can be sparse and represent interesting properties; e.g. notice that an adjacency matrix of an undirected graph is always symmetric, i.e.

$$\mathbf{A}_{un} = \mathbf{A}_{un}^T. \tag{2.8}$$

While reflecting the structure arising from the topology of data acquisition, graphs also allow analysis through linear algebra methods. Notice also that different indexing of nodes does not change the actual graph; graphs are *isomorphic* domains. Therefore, $\mathbf{A}_2$, a remunerated version of $\mathbf{A}_1$ can be defined with the appropriate permutation matrix $\mathbf{P}$ as in Equation 2.9.

$$\mathbf{A}_2 = \mathbf{P}\mathbf{A}_1\mathbf{P}^T \tag{2.9}$$

In general, the edges of a graph are weighted, indicating the importance of the connection between two vertices. The weight matrix $\mathbf{W}$, can be viewed as the weighted extension of the adjacency matrix $\mathbf{A}$. It is assumed that the set of weights, $\mathcal{W}$, have non-negative values, so a lack of an edge can again be represented by a 0. View the weighted graph in Figure 2.2 and its corresponding weight matrix $\mathbf{W}$ in Equation 2.10.

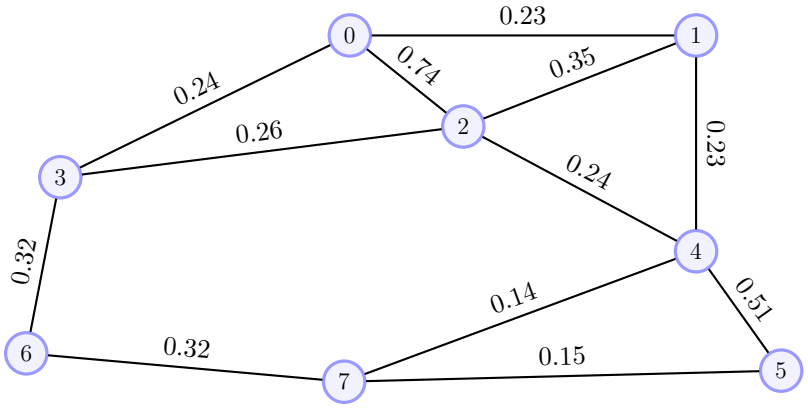


Figure 2.2: Example of an undirected weighted graph.

$$\mathbf{W} = \begin{bmatrix} 0 & 0.23 & 0.74 & 0.24 & 0 & 0 & 0 & 0 \\ 0.23 & 0 & 0.35 & 0 & 0.23 & 0 & 0 & 0 \\ 0.74 & 0.35 & 0 & 0.26 & 0.24 & 0 & 0 & 0 \\ 0.24 & 0 & 0.26 & 0 & 0 & 0 & 0.32 & 0 \\ 0 & 0.23 & 0.24 & 0 & 0 & 0.51 & 0 & 0.14 \\ 0 & 0 & 0 & 0 & 0.51 & 0 & 0 & 0.15 \\ 0 & 0 & 0 & 0.32 & 0 & 0 & 0 & 0.32 \\ 0 & 0 & 0 & 0 & 0.14 & 0.15 & 0.32 & 0 \end{bmatrix} \tag{2.10}$$

The degree matrix, $\mathbf{D}$, of an undirected graph is a diagonal matrix, with elements D_{mm}, which are equal to the sum of the weights of edges connected to vertex m. This is equal to the sum of the elements in row m of the weight matrix, $\mathbf{W}$.

$$D_{mm} = \sum_{n=0}^{N-1} W_{mn} \tag{2.11}$$

For an unweighted undirected graph, D_{mm} is equal to the number of edges connected to vertex m. For degree matrices of directed graphs, see Laplacian of directed graphs. The degree matrix of the graph in Figure 2.2 is defined as

$$\mathbf{D} = \begin{bmatrix} 1.21 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.81 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.59 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.82 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.12 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.66 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.64 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.61 \end{bmatrix} \tag{2.12}$$

Degree centrality: The degree centrality of a vertex can be considered as kind of a popularity measure [44]. It models the importance of a vertex. For an undirected unweighted graph, degree centrality of vertex m is equal to D_{mm}.

The graph Laplacian matrix, $\mathbf{L}$, is another important indicator of graph connectivity. It is defined as

$$\mathbf{L} = \mathbf{D} - \mathbf{W} \tag{2.13}$$

The Laplacian matrix of an undirected graph is symmetric, i.e. $\mathbf{L} = \mathbf{L}^T$. The entries along the diagonal of the Laplacian are non-negative, and the rest of the entries are non-positive real numbers.

The graph Laplacian of the undirected weighted graph in Figure 2.2 is given in Equation 2.14

$$\mathbf{L} = \begin{bmatrix} 1.21 & -0.23 & -0.74 & -0.24 & 0 & 0 & 0 & 0 \\ -0.23 & 0.81 & -0.35 & 0 & -0.23 & 0 & 0 & 0 \\ -0.74 & -0.35 & 1.59 & -0.26 & -0.24 & 0 & 0 & 0 \\ -0.24 & 0 & -0.26 & 0.82 & 0 & 0 & -0.32 & 0 \\ 0 & -0.23 & -0.24 & 0 & 1.12 & -0.51 & 0 & -0.14 \\ 0 & 0 & 0 & 0 & -0.51 & 0.66 & 0 & -0.15 \\ 0 & 0 & 0 & -0.32 & 0 & 0 & 0.64 & -0.32 \\ 0 & 0 & 0 & 0 & -0.14 & -0.15 & -0.32 & 0.61 \end{bmatrix} \tag{2.14}$$

The normalized graph Laplacian, $\mathbf{L}_N$, is often used for practical reasons. For undirected graphs, $\mathbf{L}_N$ is symmetric, and all of its diagonal values are equal to 1. It is defined in Equation 2.15.

$$\mathbf{L}_N = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-1/2} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2} \tag{2.15}$$

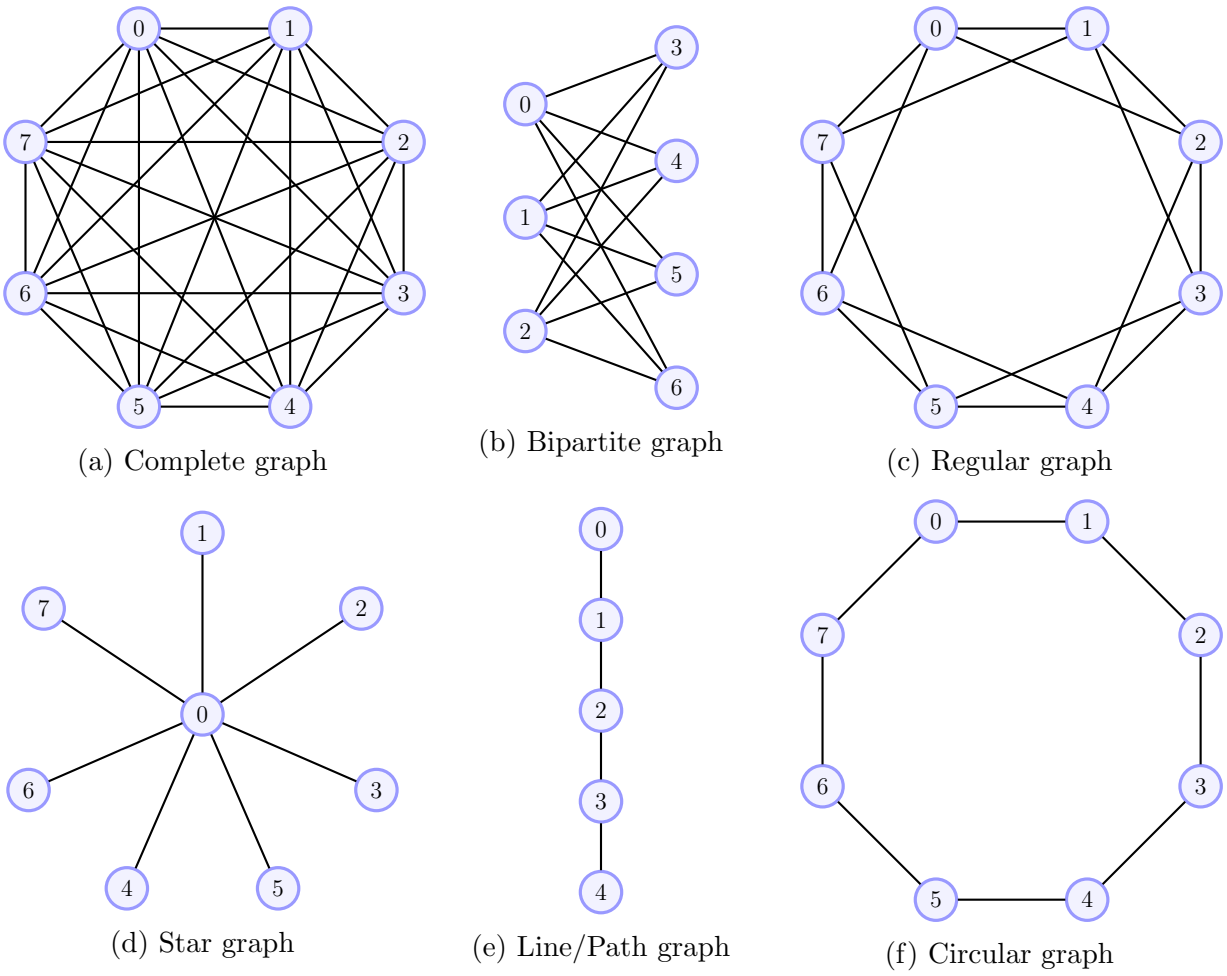
2.2.2 Graph Properties

Some of the most frequently used *graph topologies* are given in Figure 2.3. Properties and notions most relevant to analysis on graphs follow.

1: For an undirected graph, the matrices $\mathbf{A}$, $\mathbf{W}$, and $\mathbf{L}$ are all *symmetric*.

2: A *walk* from vertex m to vertex n is the connected sequence of edges that starts from m and end at n. There can be multiple walks between two vertices.

In an unweighted graph *the length of a walk* is defined by the number of edges included in the sequence. The element in the mth row and the nth column of matrix $\mathbf{A}^K$ gives the number of walks of length K between vertices m and n. This property can be proved by induction [17].



(a) Complete graph (b) Bipartite graph (c) Regular graph

(d) Star graph (e) Line/Path graph (f) Circular graph

Figure 2.3: Commonly used graph topologies

3: The number of walks of length not higher than K is given by

$$\mathbf{B}_K = \mathbf{A} + \mathbf{A}^2 + \cdots + \mathbf{A}^K \tag{2.16}$$

4: *The K-neighborhood* of a vertex is the set of vertices reachable from the given vertex in walks of length K or less. K-neighborhood of vertex m can be determined by the positions of the non-zero elements in the mth row of $\mathbf{B}_K$.

5: *A path* is a walk whereby each vertex can be included once in the sequence. The number of edges included in a path is called *path cardinality* or *path length*, while the sum of the weights along the edges is called *path weight*.

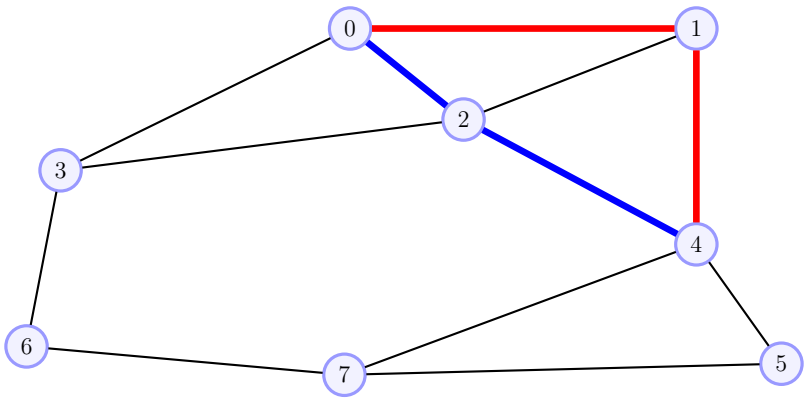


Figure 2.4: Walks of length $K = 2$ from vertex 0 to vertex 4

6: *The distance* r_{mn} between two vertices of an unweighted graph is equal to minimum path length between the given vertices. For example r_{17} in the graph from Figure 2.4 is equal to 2.

7: *The diameter* d of a graph is equal to the largest distance there exists between any pair of vertices, i.e. $d = \max_{m,n \in \mathcal{V}} r_{mn}$. The diameter of the complete graph in Figure 2.3(a) is equal to one, while the diameter of the graph in Figure 2.4 is equal to 3.

8: *Vertex farness (remoteness)* is the sum of all distances of a vertex to other vertices, $f_n = \sum_{n \neq m} r_{mn}$. *The vertex closeness* is then defined as the inverse of the farness, $c_n = 1/f_n$, and can be interpreted as how long it would take to visit the other vertices. For vertex 2 in the graph of Figure 2.4, $f_2 = 10$ and $c_2 = 0.1$.

9: *Vertex or edge betweenness* of a vertex m or edge (m, n) is defined by the number of times it acts as a bridge along the shortest paths between any other two vertices.

10: *Spanning tree and minimum spanning tree.* The spanning tree of a graph is a sub-graph that connects all vertices, therefore it cannot be disconnected, and does not include any cycles. *The cost of a spanning tree* is defined by the sum of the weights of all its edges, and *the minimum spanning tree* is the spanning tree with the minimum cost out of all possible spanning trees in a graph. Spanning trees are typically used in graph clustering analysis.

In data analytics on graphs, the edge weights are typically defined as a function of the vertex distance, e.g. through a Gaussian kernel, $W_{mn} \sim \exp(-r_{mn}^2)$. The cost function for minimum spanning tree (MST) can then be defined as a log-sum of distances, $r_{mn} = -2 \ln W_{mn}$.

11: An undirected graph is called *connected* if there exists a walk between all pairs of its vertices.

12: If the graph is *not connected*, then it is made up of two or more disjoint but locally connected subgraphs (*graph components*). Such disjoint graphs result in a block diagonal adjacency matrix $\mathbf{A}$ and Laplacian $\mathbf{L}$. For M disjoint components, these matrices are of the form

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & 0 & \dots & 0 \\ 0 & \mathbf{A}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{A}_M \end{bmatrix}, \qquad \mathbf{L} = \begin{bmatrix} \mathbf{L}_1 & 0 & \dots & 0 \\ 0 & \mathbf{L}_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathbf{L}_M \end{bmatrix}. \tag{2.17}$$

given that the vertex numbering follows the subgraph structure of the original graph.

13: *The summation operator* for two graphs defined on the same $\mathcal{V}$ with corresponding adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ creates a new graph defined as $\mathbf{A} = \mathbf{A}_1 + \mathbf{A}_2$. A logical summation operation, e.g. $1 + 1 = 1$, can be used to maintain the binary values in $\mathbf{A}$.

14: *The Kronecker (tensor) product* of two disjoint graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{B}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{B}_2)$ yields a new graph $\mathcal{G} = (\mathcal{V}, \mathcal{B})$, where $\mathcal{V} = \mathcal{V}_1 \times \mathcal{V}_2$ and $((n_1, m_1), (n_2, m_2)) \in \mathcal{B}$ only if $(n_1, n_2) \in \mathcal{B}_1$ and $(m_1, m_2) \in \mathcal{B}_2$.

$$\mathbf{A} = \mathbf{A}_1 \otimes \mathbf{A}_2 \tag{2.18}$$

An illustration of the Kronecker product is given in Figure 2.5.

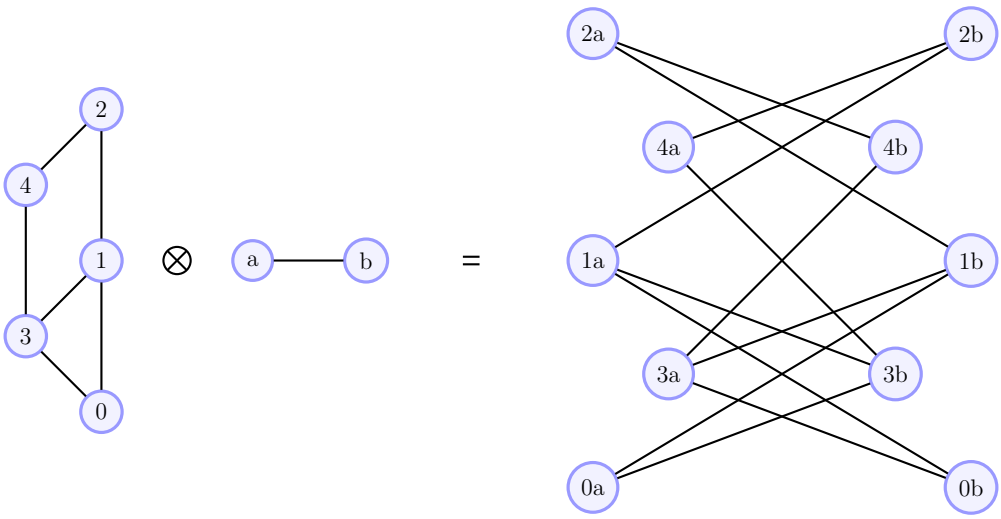


Figure 2.5: Kronecker product of two graphs

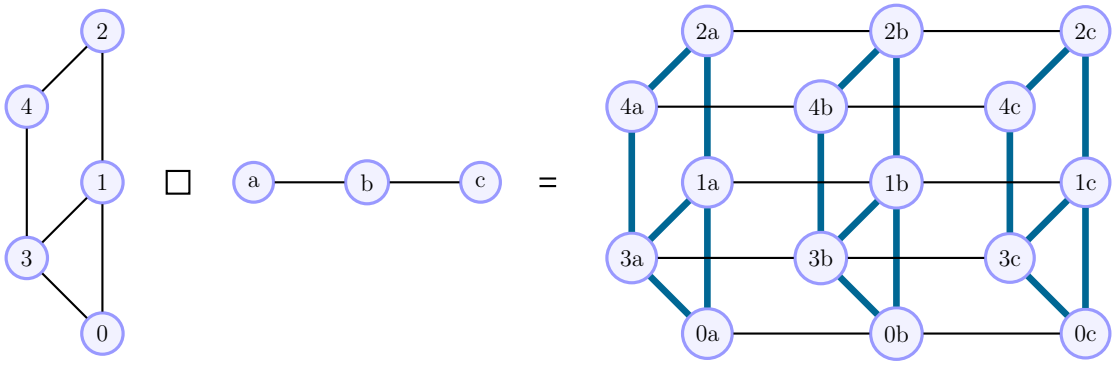


Figure 2.6: Cartesian product of two graphs

15: *The Cartesian product (graph product)* of two disjoint graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{B}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{B}_2)$ yields a new graph $\mathcal{G} = \mathcal{G}_1 \square \mathcal{G}_2 = (\mathcal{V}, \mathcal{B})$, where $\mathcal{V} = \mathcal{V}_1 \times \mathcal{V}_2$ and $((m_1, n_1), (m_2, n_2)) \in \mathcal{B}$ only if

$$m1 = m2 \quad \text{and} \quad (n_1, n_2) \in \mathcal{B}_2 \quad \textit{or} \quad n1 = n2 \quad \text{and} \quad (m_1, m_2) \in \mathcal{B}_1.$$

The adjacency matrix of a Cartesian product of two graphs is then given by the Kronecker sum,

$$\mathbf{A} = \mathbf{A}_1 \otimes \mathbf{I}_{N_2} + \mathbf{I}_{N_1} \otimes \mathbf{A}_2 = \mathbf{A}_1 \oplus \mathbf{A}_2 \tag{2.19}$$

where N_1 and N_2 are the corresponding numbers of vertices in $\mathcal{G}_1$ and $\mathcal{G}_2$, with $\mathbf{I}_{N_1}$ and $\mathbf{I}_{N_2}$ being the identity matrices of orders N_1 and N_2. An illustration of the Cartesian product is given in Figure 2.6. Notice the three-dimensional structure yielded by the Cartesian product of two 2D-graphs. For further details on graph properties see [51].

2.2.3 Eigenvalue Decomposition of Graph Matrices

The adjacency matrix can be decomposed with the eigenvalue decomposition, i.e. $\mathbf{Au} = \lambda\mathbf{u}$, where λ is the eigenvalue corresponding to the eigenvector $\mathbf{u}$. A nontrivial solution for $\mathbf{u}$ exists if $\det||\mathbf{A} - \lambda\mathbf{I}|| = 0$. The characteristic polynomial of matrix $\mathbf{A}$ is then defined as

$$P(\lambda) = \det||\mathbf{A} - \lambda\mathbf{I}|| = \lambda^N + c_1\lambda^{N-1} + \cdots + c_N. \tag{2.20}$$

The roots of the characteristic polynomial are the eigenvalues, $P(\lambda) = 0$, and in general, the number of eigenvalues is equal to the order of $P(\lambda)$. However, if zeros of algebraic multiplicity higher than one exist, the eigenvalues may be repeated. The characteristic polynomial can be rewritten as

$$P(\lambda) = (\lambda - \mu_1)^{p_1}(\lambda - \mu_2)^{p_2}\cdots(\lambda - \mu_{N_m})^{p_{N_m}}, \tag{2.21}$$

where distinct eigenvalues are denoted as $\mu_1, \mu_2, \cdots, \mu_{N_m}$ and their corresponding algebraic multiplicities as $p_1, p_2, \cdots, p_{N_m}$ with $p_1 + p_2 + \cdots + p_{N_m} = N$ and $N_m \leq N$. The minimal polynomial is defined as

$$P_{min}(\lambda) = (\lambda - \mu_1)(\lambda - \mu_2)\cdots(\lambda - \mu_{N_m}). \tag{2.22}$$

Some *properties of the characteristic and minimal polynomial* are:

1. The order of the characteristic polynomial, N, is equal to the number of vertices in the considered graph.
2. For $\lambda = 0$, $P(0) = \det(\mathbf{A}) = (-\mu_1)^{p_1}(-\mu_2)^{p_2}\cdots(-\mu_{N_m})^{p_{N_m}}$.
3. The sum of the eigenvalues is equal to the trace of $\mathbf{A}$, and $c_1 = -\mathrm{tr}\{\mathbf{A}\} = 0$ [19].
4. Coefficient c_2 is equal to the number of edges multiplied by -1 [19].
5. The degree of the minimal polynomial, N_m, is strictly larger than the diameter of the considered graph.

6. The multiplicity of the largest eigenvalue in a connected graph is 1.

7. The set of the adjacency matrix eigenvalues is defined as *the graph adjacency spectrum.*

If all eigenvalues are distinct (of algebraic multiplicity 1), we can write

$$\mathbf{AU} = \mathbf{U\Lambda} \quad \text{or} \quad \mathbf{A} = \mathbf{U\Lambda U}^{-1}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with eigenvalues on the diagonal and $\mathbf{U}$ is a matrix whose columns are composed of eigenvectors $\mathbf{u}_k$. It is common to choose eigenvectors such that $||\mathbf{u}_k||_2^2 = 1$.

For an undirected graph, the adjacency matrix $\mathbf{A}$ is symmetric. Therefore, the associated $\mathbf{A}$ has real eigenvalues, is diagonalizable, and has orthogonal eigenvectors. Hence, for undirected graphs, $\mathbf{U}^{-1} = \mathbf{U}^T$.

Eigenvalue decomposition of the squared adjacency matrix, $\mathbf{A}^2$, is then

$$\mathbf{A}^2 = \mathbf{U\Lambda U}^{-1}\mathbf{U\Lambda U}^{-1} = \mathbf{U\Lambda}^2\mathbf{U}^{-1},$$

and for an arbitrary natural number, m, the above result generalizes to

$$\mathbf{A}^m = \mathbf{U\Lambda}^m\mathbf{U}^{-1},$$

In (2.18) and (2.19), we defined the Kronecker product and the Cartesian product of two graphs with respective adjacency matrices $\mathbf{A}_1$ and $\mathbf{A}_2$. For *the eigendecomposition of the Kronecker product* of matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, the following holds.

$$\begin{aligned}\mathbf{A}_\otimes = \mathbf{A}_1 \otimes \mathbf{A}_2 &= (\mathbf{U}_1\mathbf{\Lambda}_1\mathbf{U}_1^H) \otimes (\mathbf{U}_2\mathbf{\Lambda}_2\mathbf{U}_2^H) \\ &= (\mathbf{U}_1 \otimes \mathbf{U}_2)(\mathbf{\Lambda}_1 \otimes \mathbf{\Lambda}_2)(\mathbf{U}_1 \otimes \mathbf{U}_2)^H\end{aligned}$$

In other words, the eigenvectors of the adjacency matrix of the Kronecker product of graphs are obtained by the Kronecker product of the eigenvectors of the individual adjacency matrices

of the considered graphs, as $\mathbf{u}_{k+lN_1} = \mathbf{u}_k^{(A_1)} \otimes \mathbf{u}_l^{(A_2)}$. Note that the eigenvectors of the individual adjacency matrices are of much lower dimensionality compared to those of the resulting adjacency matrix from the Kronecker product. This can be used to reduce computational complexity when analyzing data on this type of graph domains [51]. The eigenvalues of the resulting adjacency matrix are obtained as

$$\lambda_{k+lN_1} = \lambda_k^{(A_1)} \lambda_l^{(A_2)}$$

For *the eigendecomposition of the Cartesian product* of matrices $\mathbf{A}_1$ and $\mathbf{A}_2$, we have

$$\mathbf{A}_{\oplus} = \mathbf{A}_1 \oplus \mathbf{A}_2 = (\mathbf{U}_1 \otimes \mathbf{U}_2)(\mathbf{\Lambda}_1 \oplus \mathbf{\Lambda}_2)(\mathbf{U}_1 \otimes \mathbf{U}_2)^H$$

with $\mathbf{u}_k^{(A_1)} \otimes \mathbf{u}_l^{(A_2)}$ and $\lambda_{k+lN_1} = \lambda_k^{(A_1)} + \lambda_l^{(A_2)}$ [51][2]. Note that the Kronecker and Cartesian products of graphs have the same eigenvectors, while their spectra (eigenvalues) are different.

Cartesian products of graphs may be used for multidimensional extensions of graph data domains, where the resulting eigenvectors (basis functions) can be efficiently calculated using the eigenvectors of the original graphs, which are of lower dimensionality [51].

Eigenvalue Decomposition can also be performed on the *graph Laplacian* $\mathbf{L}$ for spectral analysis. The Laplacian of an undirected graph can then be decomposed as

$$\mathbf{LU} = \mathbf{U\Lambda} \quad \text{or} \quad \mathbf{L} = \mathbf{U\Lambda U}^T,$$

noting that although the notation used for the eigenvalues and eigenvectors for the graph Laplacian is same as those used for the adjacency matrix, $\mathbf{A}$, their respective values are not directly related. The set of the eigenvalues of the graph Laplacian is referred to as *the graph spectrum* or *graph Laplacian spectrum*.

Some *properties of the Laplacian eigenvalue decomposition* are:

1. Sum of the elements of each row of the graph Laplacian matrix is equal to zero. Therefore, the inner product of any row of $\mathbf{L}$ with a constant vector is also equal to zero, i.e., $\mathbf{Lu} = \mathbf{0} = 0 \cdot \mathbf{u}$, for any constant vector $\mathbf{u}$. Thus, at least one eigenvalue of the Laplacian is zero, $\lambda_0 = 0$ with a corresponding constant eigenvector such as $\mathbf{u}_0 = [1, 1, \cdots, 1]^T/\sqrt{N}$.

2. The multiplicity of the eigenvalue $\lambda_0 = 0$ of the Laplacian is equal to the number of connected subgraphs in the considered graph. No such property holds for the adjacency matrix. Therefore, in this sense, the graph Laplacian matrix carries more physical meaning than the adjacency matrix [51].

   If $\lambda_0 = \lambda_1 = 0$ and $\lambda_2 > 0$, then there are exactly two individually connected but globally disconnected subgraphs in the considered graph. If $\lambda_1 \neq 0$, its value describes the so called *algebraic connectivity of the graph*, where very small values suggest the graph is weakly connected.

3. If there are no isolated vertices, the trace of the normalized Laplacian is equal to the number of vertices.

4. The coefficient c_N in the characteristic polynomial of the graph Laplacian is equal to 0, since $\lambda = 0$ is an eigenvalue. For unweighted graphs, the coefficient c_1 is equal to the number of edges multiplied by -2.

$$P(\lambda) = \det|\mathbf{L} - \lambda\mathbf{I}| = \lambda^N + c_1\lambda^{N-1} + \cdots + c_{N-1}\lambda + c_N$$

5. Graphs with identical spectra are called *isospectral* or *cospectral* graphs, but isospectral graphs are not necessarily isomorphic.

6. The eigenvalues of the normalized graph Laplacian, $\mathbf{L}_N$, are nonnegative and bounded by $0 \leq \lambda \leq 2$.

2.2.4 Vertex Clustering

Vertex clustering is a type of graph learning which aims to group together vertices from the set V into multiple disjoint subsets, V_i, called clusters [51]. Vertices within a cluster are expected to exhibit a larger degree of similarity (in some sense) than with the vertices in other clusters.

With the clustering of graph vertices, that is identifying and arranging the vertices of a graph into non-overlapping vertex subsets, the graph is naturally partitioned into non-overlapping graph segments (components), with vertices exhibiting a relative similarity within each segment.

The notion of vertex similarity metrics and their use to accordingly cluster the vertices into sets of "related" vertices has been a focus of significant research effort in machine learning and pattern recognition [51]. Some of the established vertex similarity measures and corresponding graph clustering methods can be considered within two main categories [47]: (i) clustering based on graph topology, (ii) spectral methods for graph clustering.

Among many such existing methods, the most popular *clustering methods based on graph topology* are [51]:

- Finding the minimum set of edges whose removal would disconnect a graph in some "optimal" or "least disturbance" way (*minimum cut* based clustering).
- Designing clusters within a graph based on the disconnection of vertices or edges which belong to the highest numbers of shortest paths in the graph (*vertex betweenness* and *edge betweenness* based clustering).
- The minimum spanning tree of a graph has been a basis for a number of widely used clustering methods [30][39].
- Analysis of highly connected subgraphs (HCS) [27] has also been used for graph clustering.
- Finally, *graph data analysis* may be used for *machine learned graph clustering*, for example, the k-means based clustering methods [16][24].

2.2.5 Graph Down-Sampling

Due to computational and storage issues, in the case of extremely large graphs, subsampling and down-scaling of graphs is required for their analysis [35]. For a given large graph, $\mathcal{G}$, with N vertices, its down-sampling aims to produce a much simpler graph which retains most of the properties of the original graph, but is both less complex and more physically meaningful and computationally tractable. The similarity between the original large graph $\mathcal{G}$, and the down-scaled graph, $\mathcal{S}$ with M vertices (where $M << N$), is defined with respect to the set of parameters of interest, for example, the connectivity or distribution on a graph. Such criteria may also be related to the spectral behavior of graphs. [51]

Data dimensionality reduction, which is a challenging topic of big data applications, can be achieved through vertex clustering, graph down-sampling and compressive sensing. For detailed information on these topics, see [51].

2.2.6 Determining The Graph Topology

In modern data analytics, while the graph vertices (data sensing points) may naturally be defined by the application, that is not the case for graph edges (the connectivity of vertices) for many applications, such as the application of graphs in sensor networks, in finance, etc. [52]. In those cases, defining the data domain in a graph form becomes part of the problem definition itself, whereby a vertex connectivity scheme has to be determined based on the properties of the vertices. [52] discusses determining or learning of this graph topology in detail, placing a particular emphasis on the use of standard "relationship measures" in this context, including the correlation and precision matrices, together with the ways to combine these with the available prior knowledge and structural conditions, such as the smoothness of the graph signals or sparsity of graph connections.

[52] considers three scenarios for the estimation of graph edges from vertex geometry or data:

- Based on the *geometry of vertex positions*. "In various sensor network setups (such as

temperature, pressure, and transportation), the locations of the sensing positions (vertices) are known beforehand, while the vertex distances convey physical meaning about data dependence and thus may be employed for edge/weight determination."

- Based on *data association and data similarity.* "Various statistical measures are available to serve as data association metrics, with the covariance and precision matrices most commonly used. A strong correlation between data on two vertices would indicate a large weight associated with the corresponding edge. A small degree of correlation would indicate nonexistence of an edge (after weight thresholding)."

- Based on *physically well defined relations among the sensing positions.* "Examples include electric circuits, power networks, linear heat transfer, social and computer networks, spring-mass systems, to mention but a few. In these cases, edge weighting can usually be well defined based on the underlying context of the considered problem."

2.2.7 Other Related Work on Graphs

Data Analytics on Graphs Part I: Graphs and Spectra on Graphs [51]

Data Analytics on Graphs Part II: Signals on Graphs [50]

Data Analytics on Graphs Part III: Machine Learning on Graphs, from Graph Topology to Applications [52]

The Emerging Field of Signal Processing on Graphs, Extending High-Dimensional Data Analysis to Networks and Other Irregular Domains [48]

2.3 Tensors and Tensor Decompositions (TD)

The emergence of multi-dimensional, multi-modal big datasets has highlighted the limitations of standard flat-view matrix models and the necessity to move towards more sophisticated data analysis tools [13]. Multilinear algebra as their mathematical backbone, data analysis techniques using tensor decompositions can extract more general latent components from the data than can the matrix-based methods [13].

A *tensor* is a multidimensional array. A first-order tensor is a vector, a second-order tensor is a matrix and third and higher order tensors are considered *higher-order-tensors*. Tensors and their decompositions are widely used in psychometrics, chemometrics, signal processing, numerical linear algebra, computer vision, numerical analysis, data mining, neuroscience, graph analysis, and elsewhere [31].

Tensor decompositions are methods which are applied to data arrays to extract and explain their properties [31]. They originated in Hitchcock, 1927 [23]. The Canonical Polyadic Decomposition (CDP or CANDECOMP/PARAFAC) and the Tucker Tensor Decomposition can be considered to be higher-order extensions of the matrix principal component analysis (PCA) and singular value decomposition (SVD) [31].

2.3.1 Basic Definitions and Notation

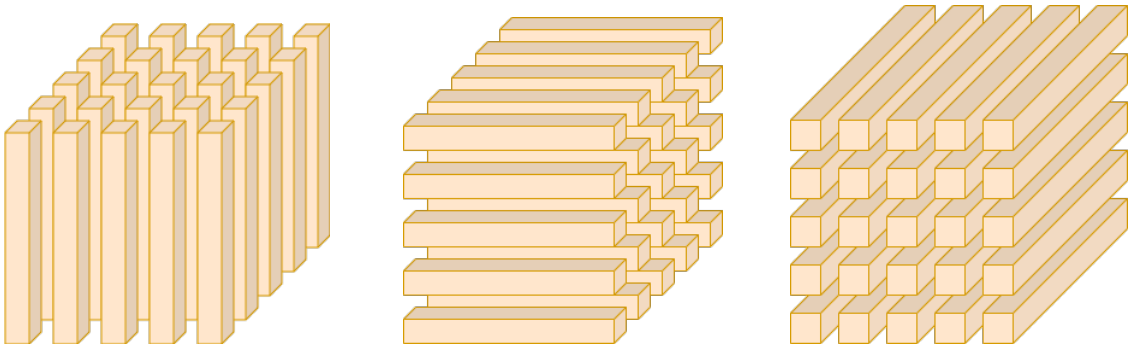
The *order* of a tensor is its number of dimensions, also known as *modes*. Vectors are tensors of order one and are denoted by bold lowercase letters. Matrices are tensors of order two and are denoted by bold capital letters. Higher-order-tensors are denoted by bold Euler script letters. Detailed information on basic notation is given in Table 2.1.

Fibers and slices: *Fibers* are 1D vectors obtained by fixing all but one indices of a tensor. Third-order tensors have column, row, and tube fibers denoted as $\mathbf{x}_{:jk}$, $\mathbf{x}_{i:k}$, and $\mathbf{x}_{ij:}$ respectively. *Slices* are 2D matrices obtained by fixing all but two indices of a tensor. Third-order tensors

$\mathcal{X}, \mathbf{X}, \mathbf{x}, x$	tensor, matrix, vector, scalar
$x_{i_1,i_2,\cdots,i_N}$	$(i_1, i_2, \cdots, i_N)$th entry of $\mathcal{X}$
$\mathbf{x}(:, i_2, i_3, \cdots, i_N)$	1D fiber of tensor $\mathcal{X}$ obtained by fixing all but one index
$\mathbf{X}(:, :, i_3, \cdots, i_N)$	2D matrix slice of tensor $\mathcal{X}$ obtained by fixing all but two indices
$\mathcal{X}(:, :, :, i_4, \cdots, i_N)$	tensor slice of $\mathcal{X}$ obtained by fixing some indices
$\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 I_2 \cdots I_{n-1} I_{n+1} \cdots I_N}$	mode-n matricization of tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$
$\text{vec}(\mathcal{X}) \in \mathbb{R}^{I_1 I_2 \cdots I_N}$	vectorization of tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$
$\mathcal{Y} = \mathcal{X} \times_n \mathbf{U}$	mode-n product of $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ yields $\mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N}$

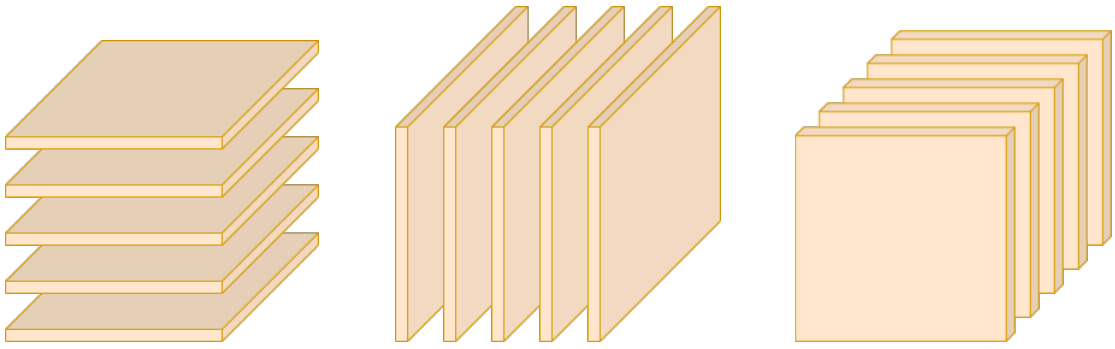
Table 2.1: Tensors basic notation

have horizontal, lateral, and frontal slices denoted as $\mathbf{X}_{i::}$, $\mathbf{X}_{:j:}$, and $\mathbf{X}_{::k}$ respectively. See figures 2.7 and 2.8 for an illustration of fibers and slices of a third-order tensor.



(a) Column fibers: $\mathcal{X}_{:jk}$ (b) Row fibers: $\mathcal{X}_{i:k}$ (c) Tube fibers: $\mathcal{X}_{ij:}$

Figure 2.7: Fibers of a 3rd-order tensor



(a) Horizontal slices: $\mathcal{X}_{i::}$ (b) Lateral slices: $\mathcal{X}_{:j:}$ (c) Frontal slices: $\mathcal{X}_{::k}$

Figure 2.8: Slices of a 3rd-order tensor

The norm and the inner product: The *norm* of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the square root of the sum of the squares of all its elements, i.e.,

$$||\mathcal{X}|| = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \cdots i_N}^2}. \tag{2.23}$$

The *inner product* of two same-sized tensors $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the sum of the products of their corresponding entries, i.e.,

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \cdots i_N} y_{i_1 i_2 \cdots i_N}}. \tag{2.24}$$

It follows that $\langle \mathcal{X}, \mathcal{X} \rangle = ||\mathcal{X}||^2$.

Rank-one tensors and the outer product: The symbol "$\circ$" represents the vector outer product. An Nth-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is *rank-one* if it can be written as the outer product of N vectors, i.e.,

$$\mathcal{X} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \circ \mathbf{a}^{(N)}, \tag{2.25}$$

where each element of $\mathcal{X}$ is defined as

$$x_{i_1 i_2 \cdots i_N} = a_{i_1}^{(1)} a_{i_2}^{(2)} \cdots a_{i_N}^{(N)} \qquad \text{for all } 1 \leq i_n \leq I_n. \tag{2.26}$$

Figure 2.9 illustrates $\mathcal{X} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$.

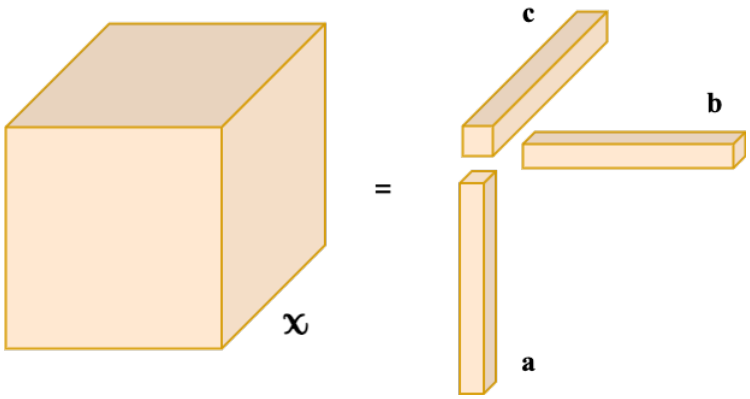


Figure 2.9: Rank-one three-way tensor: $\mathcal{X} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$

Symmetry of tensors: A tensor is referred to be *cubical* if every mode is of same size, i.e., $\mathcal{X} \in \mathbb{R}^{I \times I \times \cdots \times I}$, and a cubical tensor is called *supersymmetric* if its elements stay constant under any permutation of the indices [31] [14]. Tensors can also be (partially) *symmetric* in two or more modes.

Diagonal tensors: A tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is *diagonal* if $x_{i_1 i_2 \cdots i_N} \neq 0$ only when $i_1 = i_2 = \cdots = i_N$.

Matricization of tensors: *Matricization* is also referred to as *unfolding* or *flattening*. It is the process of transforming an N-way tensor into a matrix form by arranging the specified mode-n fibers to be the columns of the resulting matrix. The mode-n matricization of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 I_2 \cdots I_{n-1} I_{n+1} \cdots I_N}$. Consider a three-way tensor $\mathcal{X} \in \mathbb{R}^{3 \times 4 \times 2}$ with frontal slices

$$\mathbf{X}_1 = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}, \qquad \mathbf{X}_1 = \begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix}. \tag{2.27}$$

Then the three mode-n unfoldings are

$$\mathbf{X}_{(1)} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix},$$

$$\mathbf{X}_{(2)} = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 4 & 5 & 6 & 16 & 17 & 18 \\ 7 & 8 & 9 & 19 & 20 & 21 \\ 10 & 11 & 12 & 22 & 23 & 24 \end{bmatrix},$$

$$\mathbf{X}_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & \cdots & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & \cdots & 21 & 22 & 23 & 24 \end{bmatrix}.$$

Particular ordering of the columns is not important as long as they are consistent across relevant calculations. It is also possible to vectorize a tensor, i.e., $\text{vec}(\mathcal{X}) \in \mathbb{R}^{I_1 I_2 \cdots I_N}$. The result of vectorization for the example above is

$$\text{vec}(\mathcal{X}) = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ 24 \end{bmatrix}.$$

The n-mode product: The *n-mode product* of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$ is denoted by $\mathcal{X} \times_n \mathbf{U}$ and is of dimensions $I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N$. Elementwise,

$$(\mathcal{X} \times_n \mathbf{U})_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \cdots i_N} u_{j i_n}. \tag{2.28}$$

Each mode-n fiber is multiplied by matrix $\mathbf{U}$. Therefore, the n-mode product can be expressed in terms of tensor unfolding, i.e.,

$$\mathcal{Y} = \mathcal{X} \times_n \mathbf{U} \quad \Leftrightarrow \quad \mathcal{Y}_{(n)} = \mathbf{U}\mathcal{X}_{(n)}. \tag{2.29}$$

The n-mode product of a tensor with a matrix is related to a change of basis in the case when a tensor defines a multilinear operator [31].

As an example of the n-mode product, let $\mathcal{X}$ be the tensor defined in 2.27 and let $\mathbf{U} = \left[\begin{smallmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{smallmatrix}\right]$. Then the product $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{U} \in \mathbb{R}^{2 \times 4 \times 2}$ is given by

$$\mathbf{Y}_1 = \begin{bmatrix} 22 & 49 & 76 & 103 \\ 28 & 64 & 100 & 136 \end{bmatrix}, \qquad \mathbf{Y}_2 = \begin{bmatrix} 130 & 157 & 184 & 211 \\ 172 & 208 & 244 & 280 \end{bmatrix}.$$

A few properties of n-mode matrix products follow.

1: In a series of multiplications of matrices for distinct modes, the order of the multiplications is not relevant, i.e.,

$$\mathcal{X} \times_n \mathbf{A} \times_m \mathbf{B} = \mathcal{X} \times_m \mathbf{B} \times_n \mathbf{A} \qquad \text{if } m \neq n. \tag{2.30}$$

2: If $m = n$, then

$$\mathcal{X} \times_n \mathbf{A} \times_n \mathbf{B} = \mathcal{X} \times_n (\mathbf{BA}). \tag{2.31}$$

3: The n-mode vector product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with a vector $\mathbf{v} \in \mathbb{R}^{I_N}$ is denoted by $\mathcal{X} \,\bar{\times}_n\, \mathbf{v}$. The result is of dimensions $I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N$, meaning that the result is of order $N - 1$. For $\mathcal{X}$ given in 2.27 and $\mathbf{v} = [1\ 2\ 3\ 4]^T$, then

$$\mathcal{X} \,\bar{\times}_2\, \mathbf{v} = \begin{bmatrix} 70 & 190 \\ 80 & 200 \\ 90 & 210 \end{bmatrix}.$$

In a series of n-mode vector multiplications, precedence matters since the order of the intermediate results changes.

Matrix Kronecker, Khatri–Rao, and Hadamard Products: The *Kronecker product* of matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$ is denoted by $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{(IK) \times (JL)}$ and is defined as

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \cdots & a_{IJ}\mathbf{B} \end{bmatrix} \tag{2.32}$$

$$= \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_1 \otimes \mathbf{b}_2 & \mathbf{a}_1 \otimes \mathbf{b}_3 & \cdots & \mathbf{a}_J \otimes \mathbf{b}_{L-1} & \mathbf{a}_J \otimes \mathbf{b}_L \end{bmatrix}. \tag{2.33}$$

The *Khatri–Rao product* is the "matching columnwise" Kronecker product [31]. The Khatri–Rao product of matrices $\mathbf{A} \in \mathbb{R}^{I \times K}$ and $\mathbf{B} \in \mathbb{R}^{J \times K}$ is denoted by $\mathbf{A} \odot \mathbf{B} \in \mathbb{R}^{(IJ) \times K}$ and is

defined as

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \cdots & \mathbf{a}_K \otimes \mathbf{b}_K \end{bmatrix}. \tag{2.34}$$

If $\mathbf{a}$ and $\mathbf{b}$ are vectors, then their Kronecker and Khatri–Rao products are identical, i.e., $\mathbf{a} \otimes \mathbf{b} = \mathbf{a} \odot \mathbf{b}$.

The *Hadamard product* is the elementwise matrix product. The Hadamard product of matrices $\mathbf{A}$ and $\mathbf{B}$, both of size $I \times J$, is denoted by $\mathbf{A} * \mathbf{B} \in \mathbb{R}^{I \times J}$ and is defined as

$$\mathbf{A} * \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2J}b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \cdots & a_{IJ}b_{IJ} \end{bmatrix}. \tag{2.35}$$

Some properties of the mentioned matrix products follow

$$\begin{aligned} (\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) &= \mathbf{AC} \otimes \mathbf{BD}, \\ (\mathbf{A} \otimes \mathbf{B})^\dagger &= \mathbf{A}^\dagger \otimes \mathbf{B}^\dagger, \\ \mathbf{A} \odot \mathbf{B} \odot \mathbf{C} &= (\mathbf{A} \odot \mathbf{B}) \odot \mathbf{C} = \mathbf{A} \odot (\mathbf{B} \odot \mathbf{C}), \\ (\mathbf{A} \odot \mathbf{B})^T(\mathbf{A} \odot \mathbf{B}) &= \mathbf{A}^T\mathbf{A} * \mathbf{B}^T\mathbf{B}, \\ (\mathbf{A} \odot \mathbf{B})^\dagger &= ((\mathbf{A}^T\mathbf{A}) * (\mathbf{B}^T\mathbf{B}))^\dagger(\mathbf{A} \odot \mathbf{B})^T, \end{aligned}$$

where † denotes the Moore–Penrose pseudoinverse.

2.3.2 Tensorization of Lower Dimensional Data

The process of creating a data tensor from lower-dimensional original data is referred to as *tensorization*, and the following taxonomy is proposed in [13] for tensor generation:

1. *Rearrangement of lower dimensional data structures.* Large-scale vectors or matrices can be tensorized to higher-order tensors and can be compressed through tensor decompo-

sitions if they admit a low-rank tensor approximation; this principle facilitates big data analysis [13].

2. *Mathematical construction.* Among many such examples, the Nth-order moments of a vector-valued random variable form an Nth-order tensor [13] [41].

3. *Experiment design.* Multi-faceted data can naturally be stacked into a tensor.

4. *Naturally tensor data.* Some data sources are readily generated as tensors, e.g., RGB color images, videos, 3D light field displays [13].

The high dimensionality of tensors allow possibilities to obtain compact representations, uniqueness in decompositions, flexibility in the choice of constraints, and generality of components that can be identified [13].

2.3.3 Canonical Polyadic Decomposition (CPD)

We refer to this particular tensor decomposition method as Canonical Polyadic Decomposition (CPD) as in [13]. However, it is referred to as Polyadic form of a tensor in [23], PARAFAC (parallel factors) in [21], CANDECOMP or CAND (canonical decomposition) in [8], Topographic components model in [38], and CP (CANDECOMP/PARAFAC) in [28]. CPD factorizes a

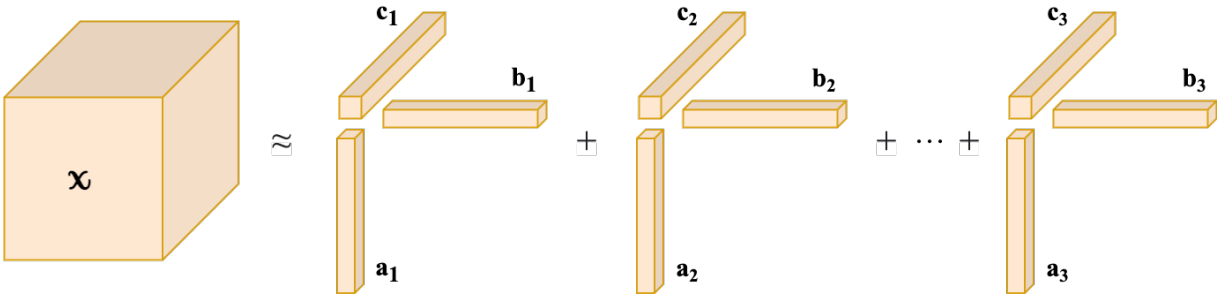


Figure 2.10: Canonical Polyadic Decomposition of a third-order tensor

tensor into a sum of rank-one tensors. It is illustrated for a third-order tensor in Figure 2.10. Given $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$, we can write its CPD as

$$\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \tag{2.36}$$

where R is a positive integer and $\mathbf{a}_r \in \mathbb{R}^I, \mathbf{b}_r \in \mathbb{R}^J, \mathbf{c}_r \in \mathbb{R}^K$. The *factor matrices* are composed of the the vectors from rank-one components, i.e., $\mathbf{A} = [\mathbf{a}_1\ \mathbf{a}_2\ \cdots\ \mathbf{a}_R]$ and likewise for $\mathbf{B}$ and $\mathbf{C}$. Then, we can write $\boldsymbol{\mathcal{X}}$ in matricized form as

$$\mathbf{X}_{(1)} \approx \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T,$$
$$\mathbf{X}_{(2)} \approx \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T,$$
$$\mathbf{X}_{(3)} \approx \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T.$$

The third-order model can be written in terms of the frontal slices, $\mathbf{X}_k$, of $\boldsymbol{\mathcal{X}}$, i.e,

$$\mathbf{X}_k \approx \mathbf{A}\mathbf{D}^{(k)}\mathbf{B}^T, \quad \text{where } \mathbf{D}^{(k)} \equiv \text{diag}(\mathbf{c}_k) \quad \text{for } k = 1, \cdots, K,$$

and similar equations can be written for horizontal and lateral slices. Following [31], we express the CPD model concisely as

$$\boldsymbol{\mathcal{X}} \approx [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \equiv \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \tag{2.37}$$

It is often useful to assume that the columns of $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are normalized with the weights absorbed into $\boldsymbol{\lambda} \in \mathbb{R}^R$, so Equation 2.36 becomes

$$\boldsymbol{\mathcal{X}} \approx [\![\boldsymbol{\lambda}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \equiv \sum_{r=1}^{R} \lambda_r\, \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r. \tag{2.38}$$

For a general Nth-order tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, the CPD is given by

$$\boldsymbol{\mathcal{X}} \approx [\![\boldsymbol{\lambda}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(N)}]\!] \equiv \sum_{r=1}^{R} \lambda_r\, \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(N)}, \tag{2.39}$$

and the mode-n matricized form

$$\mathbf{X}_{(n)} \approx \mathbf{A}^{(n)}\boldsymbol{\Lambda}(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)})^T, \tag{2.40}$$

with $\mathbf{\Lambda} = \text{diag}(\boldsymbol{\lambda})$.

Tensor Rank: The *rank* of a tensor $\boldsymbol{\mathcal{X}}$, rank($\boldsymbol{\mathcal{X}}$), is the minimum number of rank-one tensors that generate $\boldsymbol{\mathcal{X}}$ as their sum [31]. Therefore, it is the minimum R that results in an exact CPD, meaning resulting in an equality in Equation 2.36. An exact CPD with $R = \text{rank}(\boldsymbol{\mathcal{X}})$ is called *rank decomposition*.

Tensor rank is analogous to matrix rank. However, their properties are quite different. For example, the rank of a real-valued tensor may be different over $\mathbb{R}$ and $\mathbb{C}$. Another major difference is that there is no straightforward algorithm to determine the rank of a given tensor; in fact, [22] proves that the problem is NP-hard. In practice, the rank of a tensor is determined numerically by fitting various rank-R CPD models [31].

Another idiosyncrasy of tensors is about maximum and typical ranks. The *maximum rank* is defined as the largest possible rank, while the *typical rank* is any rank that occurs with positive probability [31]. For a three-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$, the following weak upper bound on its maximum rank is known [33]:

$$\text{rank}(\boldsymbol{\mathcal{X}}) \leq \min\{IJ, IK, JK\}. \tag{2.41}$$

Note that in general, the ordering of the modes does not affect the rank of a tensor, i.e., the rank is constant under permutation. For further information on tensor rank, see [31].

Uniqueness: An interesting property of higher-order tensors is that their rank decompositions are often unique, while matrix decompositions are not. Consider matrix $\mathbf{X} \in \mathbb{R}^{I \times J}$ to be of rank R. Its rank decomposition and its SVD are respectively given by

$$\mathbf{X} = \mathbf{A}\mathbf{B}^T = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r,$$

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T.$$

Then we can choose $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}$ and $\mathbf{B} = \mathbf{V}$. Note that given $\mathbf{W}$ is some $R \times R$ orthonormal

matrix, it is also valid to choose $\mathbf{A} = \mathbf{U\Sigma W}$ and $\mathbf{B} = \mathbf{VW}$. In other words, we can construct different sets of factor matrices for the rank decomposition of a rank R matrix.

The rank decomposition for tensors, on the other hand, is unique under much weaker conditions [31]. Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$ be a third order tensor with rank R, i.e.,

$$\boldsymbol{\mathcal{X}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] \equiv \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r,$$

Uniqueness means that this is the only possible combination of rank-one tensors that sums to $\boldsymbol{\mathcal{X}}$, with the exception of the elementary indeterminacies of permutation and scaling [31], i.e. respectively,

$$\boldsymbol{\mathcal{X}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = [\![\mathbf{A\Pi}, \mathbf{B\Pi}, \mathbf{C\Pi}]\!]$$

for any $R \times R$ permutation matrix $\mathbf{\Pi}$, and

$$\boldsymbol{\mathcal{X}} = \sum_{r=1}^{R} (\alpha_r \mathbf{a}_r) \circ (\beta_r \mathbf{b}_r) \circ (\gamma_r \mathbf{c}_r),$$

as long as $\alpha_r \beta_r \gamma_r = 1$ for $r = 1, \ldots, R$.

One well-known result on uniqueness is from Kruskal [34] [33], which depends on the concept of k-rank. The *k-rank* of a matrix $\mathbf{A}$, denoted $k_{\mathbf{A}}$, is defined as the maximum value k such that any k columns are linearly independent [34]. From Kruskal's results [34] [33], we say that a sufficient condition for the uniqueness of the CP decomposition of a three-way tensor is

$$k_{\mathbf{A}} + k_{\mathbf{B}} + k_{\mathbf{C}} \geq 2R + 2,$$

and the sufficient condition for uniqueness of CPD for an N-way tensor with rank R is

$$\sum_{n=1}^{N} k_{\mathbf{A}^{(n)}} \geq 2R + (N - 1). \tag{2.42}$$

For detailed information on uniqueness of the CP decomposition, see [31].

Low-Rank Approximation and the Border Rank: For matrices, the best rank-k approx-

imation is given by the leading k factors of the SVD [18]. That is, given matrix $\mathbf{A}$ of rank R with decomposition

$$\mathbf{A} = \sum_{r=1}^{R} \sigma_r \mathbf{u}_r \mathbf{v}_r \quad \text{with } \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_R > 0,$$

a rank-k approximation that minimizes $||\mathbf{A} - \mathbf{B}||$ is given by

$$\mathbf{B} = \sum_{r=1}^{k} \sigma_r \mathbf{u}_r \mathbf{v}_r.$$

This does not hold for tensors of higher order. A detailed reasoning has been provided by [31] with examples. It reasons that the components of the best rank-k model may not be solved for sequentially and that all factors must be found simultaneously. It therefore suggests that a numerical solution has to be proposed, providing Figure 2.11 as an illustration. A sequence $\{\mathcal{X}_k\}$

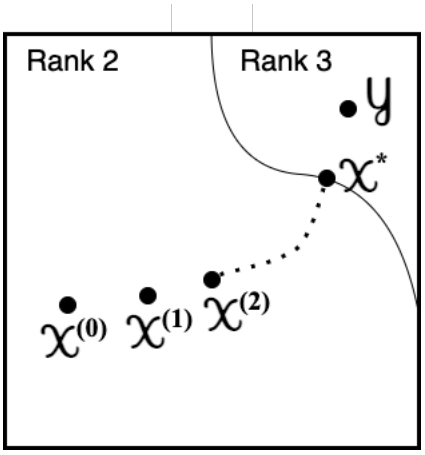


Figure 2.11: Illustration of a sequence of tensors converging to one of higher rank [31]

of rank-two tensors provides increasingly better estimates of rank-three tensor $\mathcal{Y}$. Necessarily, the best approximation is on the boundary of the space of rank-two and rank-three tensors. [31] also argues that there may be a lack of a best rank-k approximation. In that situation, it is useful to consider the concept of *border rank*, which is defined as the minimum number of rank-one tensors that are sufficient to approximate the given tensor with arbitrarily small nonzero error [3]. For further details see [31].

Computing the CP Decomposition One issue in computing a CPD is about choosing the number of rank-one components. Most procedures fit multiple CP decompositions with different numbers of components until one is "good" [31]. Assuming the number of components is fixed, we can use the alternating least squares (ALS) method proposed in [8] [21].

Let $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ be a three-way tensor and R be the number of components in the CPD. Then, the aim is to find

$$\min_{\hat{\mathcal{X}}} ||\mathcal{X} - \hat{\mathcal{X}}||, \text{ with } \hat{\mathcal{X}} = \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r = [\![\boldsymbol{\lambda}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!]. \tag{2.43}$$

The ALS approach fixes $\mathbf{B}$ and $\mathbf{C}$ to solve for $\mathbf{A}$, then fixes $\mathbf{A}$ and $\mathbf{C}$ to solve for $\mathbf{B}$, then fixes $\mathbf{A}$ and $\mathbf{B}$ to solve for $\mathbf{C}$, and continues to repeat the entire procedure until some convergence criterion is satisfied [31]. The full ALS procedure for an Nth-order tensor is shown below.

Algorithm 1 ALS algorithm to compute a CP decomposition with R components for an Nth-order tensor $\mathcal{X}$ of size $I_1 \times I_2 \times \cdots \times I_N$ [31].

procedure CP-ALS($\mathcal{X}, R$)
  initialize $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ for $n = 1, \ldots, N$
  repeat
    for $n = 1, \ldots, N$ **do**
      $\mathbf{V} \leftarrow \mathbf{A}^{(1)T}\mathbf{A}^{(1)} * \cdots * \mathbf{A}^{(n-1)T}\mathbf{A}^{(n-1)} * \mathbf{A}^{(n+1)T}\mathbf{A}^{(n+1)} * \cdots * \mathbf{A}^{(N)T}\mathbf{A}^{(N)}$
      $\mathbf{A}^{(n)} \leftarrow \mathcal{X}_{(n)}(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)})\mathbf{V}^{\dagger}$
      normalize columns of $\mathbf{A}^{(n)}$ (storing norms as $\boldsymbol{\lambda}$)
    end for
  until fit ceases to improve or maximum iterations exhausted
  return $\boldsymbol{\lambda}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}$
end procedure

The factor matrices can be initialized randomly or by setting

$$\mathbf{A}^{(n)} = R \text{ leading left singular vectors of } \mathbf{X}_{(n)} \text{ for } n = 1, \ldots, N.$$

The ALS method can take many iterations to converge and is also not guaranteed to converge to a global minimum, only to a solution where the objective function of (2.43) ceases to decrease. For further details on the Canonical Polyadic Decomposition, see [31].

2.3.4 Tucker Decomposition (TD)

First introduced in [53], the Tucker decomposition, like CPD, goes by different names. Some of which are: Three-mode factor analysis (3MFA/Tucker3) [54], Three-mode PCA (3MPCA) [32], N-mode PCA [26], Higher-order SVD (HOSVD) [15], and N-mode SVD [55]. The Tucker decomposition is a form of higher-order PCA [31]. It decomposes a tensor into a core tensor multiplied by a matrix along each mode. For a third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$, we have

$$\boldsymbol{\mathcal{X}} \approx \boldsymbol{\mathcal{G}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr}\, \mathbf{a}_p \circ \mathbf{b}_q \circ \mathbf{c}_r = [\![\boldsymbol{\mathcal{G}}; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!], \tag{2.44}$$

where $\mathbf{A} \in \mathbb{R}^{I \times P}, \mathbf{B} \in \mathbb{R}^{J \times Q}, \mathbf{C} \in \mathbb{R}^{K \times R}$ are the factor matrices and can be thought of as the principal components in each mode. The tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{P \times Q \times R}$ is called the *core tensor*, with its entries representing the level of interaction between the different components [31]. An illustration of the Tucker decomposition is given in Figure 2.12. Elementwise, the Tucker

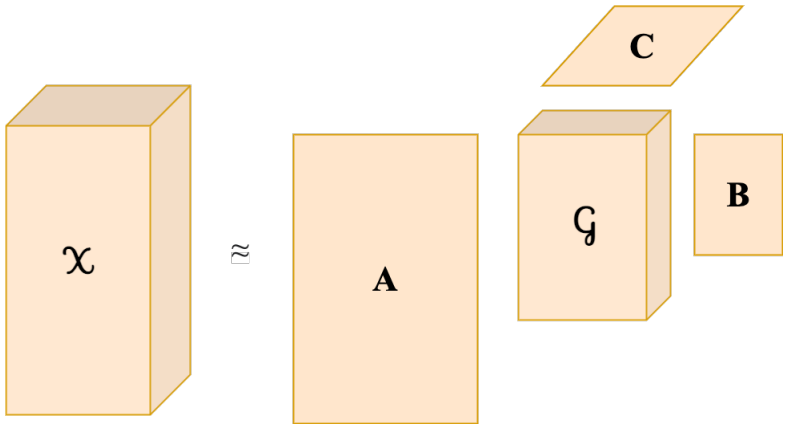


Figure 2.12: Tucker decomposition of a third-order tensor

decomposition in (2.44) is given by

$$x_{ijk} \approx \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} a_{ip} b_{jq} c_{kr} \quad \text{for } i = 1, \ldots, I,\ j = 1, \ldots, J,\ k = 1, \ldots, K.$$

Notice that if P, Q, and R are smaller than I, J, and K, the core tensor $\boldsymbol{\mathcal{G}}$ can be thought as a compressed version of $\boldsymbol{\mathcal{X}}$. Also note that the CPD can be viewed as a special case of the Tucker decomposition where the core tensor is superdiagonal and $P = Q = R$. The matricized

forms are

$$\mathbf{X}_{(1)} \approx \mathbf{A}\mathbf{G}_{(1)}(\mathbf{C} \otimes \mathbf{B})^T,$$
$$\mathbf{X}_{(2)} \approx \mathbf{B}\mathbf{G}_{(2)}(\mathbf{C} \otimes \mathbf{A})^T,$$
$$\mathbf{X}_{(3)} \approx \mathbf{C}\mathbf{G}_{(3)}(\mathbf{B} \otimes \mathbf{A})^T.$$

The Tucker decomposition then can be generalized for Nth-order tensors as

$$\mathcal{X} \approx \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)} = [\![\mathcal{G}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!] \tag{2.45}$$

or, elementwise, as

$$x_{i_1 i_2 \cdots i_N} \approx \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_N=1}^{R_N} g_{r_1 r_2 \cdots r_N} a^{(1)}_{i_1 r_1} a^{(2)}_{i_2 r_2} \cdots a^{(N)}_{i_N r_N} \quad \text{for } i_n = 1, \ldots, I_n, \ n = 1, \ldots, N,$$

with the matricized form as

$$\mathbf{X}_{(n)} \approx \mathbf{A}^{(n)}\mathbf{G}_{(n)}(\mathbf{A}^{(N)} \otimes \cdots \otimes \mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \cdots \otimes \mathbf{A}^{(1)})^T.$$

There are variations of the Tucker decomposition such as the *Tucker2* and *Tucker1*. Due to the existence of various tensor decompositions, there may be confusion about which model to use for a particular application. [9] discusses methods for choosing between CPD and the variants of TD for the three-way case. For further details see [31].

The n-Rank: The *n-rank* of $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_n}$, denoted $\text{rank}_n(\mathcal{X})$, is the column rank of $\mathbf{X}_{(n)}$; in other words, it is the dimension of the vector space spanned by the mode-n fibers. Letting $R_n = \text{rank}_n(\mathcal{X})$ for $n = 1, \ldots, N$, we can define $\mathcal{X}$ as a rank-$(R_1, R_2, \ldots, R_N)$ tensor, though n-mode rank should not be confused with rank (i.e., the minimum number of rank-one components). Also note that $R_n \leq I_n$ for all $n = 1, \ldots, N$.

An exact Tucker decomposition of rank $(R_1, R_2, \ldots, R_N)$, with $R_n = \text{rank}_n(\mathcal{X})$ can easily be

computed. However, computing a TD of rank $(R_1, R_2, \ldots, R_N)$, with $R_n < \text{rank}_n(\mathcal{X})$ for one or more n, is more difficult and will be inexact. Note also that the Tucker decompositions are not unique; for more information on Tucker decompositions and their computation see [31].

2.3.5 Large-Scale Data and Curse of Dimensionality

The size of tensor data easily saturates the processing capability of standard computers. Therefore, we have to consider the cases where tensor dimensions in all or some modes are large or, worse still, tensor order is high [13]. In the context of tensors, the *curse of dimensionality* refers to the fact that the number of elements of an Nth-order tensor scales exponentially with the tensor order N, and computations and data storage quickly becomes unmanageable. Tensor decompositions may be elegantly employed in this context as efficient representation tools [13]. A table from [13] is provided in Table 2.2, which demonstrates the storage complexities of different tensor models.

1. CPD	$\mathcal{O}(NIR)$
2. Tucker	$\mathcal{O}(NIR + R^N)$
3. TT	$\mathcal{O}(NIR^2)$
4. QTT	$\mathcal{O}(NIR^2 \log_2(I))$

Table 2.2: Storage complexities of tensor models for an Nth-order tensor $\mathcal{X} \in \mathbb{R}^{I \times I \times \cdots \times I}$, for which the original storage complexity is $\mathcal{O}(I^N)$ [13]

For detailed information on bypassing the curse of dimensionality with CPD, TD and tensor networks, see [13].

2.3.6 Other Related Work on Tensors and Tensor Decompositions

Tensor Decompositions and Applications [31]

Tensor Decompositions for Signal Processing Applications From Two-way to Multiway Component Analysis [13]

Low-Rank Tensor Networks for Dimensionality Reduction and Large-Scale Optimization Problems: Perspectives and Challenges PART 1 [11]

Tensor Networks for Dimensionality Reduction and Large-Scale Optimizations Part 2 Applications and Future Perspectives [12]

Era of Big Data Processing: A New Approach via Tensor Networks and Tensor Decompositions [10]

Tensor Networks for Latent Variable Analysis: Novel Algorithms for Tensor Train Approximation [43]

Graph Regularized Tensor Train Decomposition [49]

2.4 Using Tensors and Tensor Decompositions on Graphs

2.4.1 Tensor Representation of Lattice-Structured Graphs

Tensors can be considered to be a special class of graph signals, where vertices reside on a high-dimensional regular lattice structure (which can be obtained through the Cartesian product of multiple one-dimensional path graphs), allowing the associated adjacency matrices to exhibit a physically meaningful structured form, referred to as Kronecker summable [52]. This effectively reduces the number of parameters needed to model the entire graph connectivity structure [1] and directly accounts for the higher-dimensional relationships between data sources.

[52] introduces this approach together with a unifying framework called *Multi-Graph Tensor Network* (MGTN) for learning of big data on irregular domains. A quick review of this approach follows.

2.4.2 Tensorization of Graph Signals in High-Dimensional Spaces

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be an N-vertex graph. Each vertex on the graph is associated with a variable (signal), denoted by $x(n) \in \mathbb{R}$, which maps a vertex, $n \in \mathcal{V}$, to a real number, i.e., $x : \mathcal{V} \to \mathbb{R}$. In other words, in a single-dimensional coordinate system, each vertex represents a scalar-valued field. Considering the values of all N vertices in $\mathcal{V}$, we can form the vector $\mathbf{x} \in \mathbb{R}^N$, which defines the mapping $\mathbf{x} : \mathcal{V} \to \mathbb{R}^N$.

If a graph, however, resides in an M-dimensional space, then each vertex, $n \in \mathcal{V}$, has a unique coordinate in this space, denoted by $(i_1, \ldots, i_M) \in \mathbb{N}^M$, where $i_m \in \mathbb{N}$ is the coordinate associated with the mth axis. Therefore, there exists a unique mapping $n \to (i_1, \ldots, i_M)$. So, each graph vertex signal can be viewed as a scalar field in an M-dimensional coordinate system, i.e., $x(n) \equiv x(i_1, \ldots, i_M) \in \mathbb{R} \equiv x : \mathbb{N}^M \to \mathbb{R}$. The total number of discrete sampling points, i.e.

vertices, in this space is then given by

$$\prod_{m=1}^{M} I_m = N. \tag{2.46}$$

Naturally, the collection of samples forms the tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_M}$, with each entry defined as

$$\boldsymbol{\mathcal{X}}_{i_1 \ldots i_M} = x(i_1, \ldots, i_M), \quad \text{where } i_m \in \mathbb{N}, \quad \text{for } m = 1, \ldots, M.$$

2.4.3 Tensor Decomposition

If the underlying field is defined as a *multilinear map* of the form $x : \mathbb{N}^M \to \mathbb{R}$, then it is said to be *linearly separable* and admits the following decomposition:

$$\boldsymbol{\mathcal{X}}_{i_1,\ldots,i_M} = x(i_1, \ldots, i_M) = \prod_{m=1}^{M} x_m(i_m). \tag{2.47}$$

In other words, the value of each entry $\boldsymbol{\mathcal{X}}_{i_1,\ldots,i_M}$ is given by the product of M independent single-dimensional functions, $x_m : \mathbb{N} \to \mathbb{R}$ (where $x_m(i) = [\mathbf{x}_m]_i$), each associated with the mth coordinate axis of the underlying M-dimensional coordinate system. Therefore, a tensor, $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times \cdots \times I_M}$, sampled from a linearly separable field of the kind in (2.47) admits the following rank-one CPD:

$$\boldsymbol{\mathcal{X}} = \mathbf{x}_1 \circ \cdots \circ \mathbf{x}_M, \tag{2.48}$$

with $\mathbf{x}_m \in \mathbb{R}^{I_m}$. The property in (2.48) is also referred to as the *Kronecker separability* condition and is fundamental to most tensor decompositions and learning algorithms [52]. Kronecker separable tensors also admit a vector representation expressed as

$$\text{vec}(\boldsymbol{\mathcal{X}}) = \mathbf{x}_M \otimes \cdots \otimes \mathbf{x}_1 \in \mathbb{R}^N. \tag{2.49}$$

Note that if a scalar field is linearly separable as in (2.47), then the sampled tensor is Kronecker separable as in (2.48) and can therefore be expressed in a vectorized form as in (2.49). Note also the fact that tensor decompositions allow parameter reduction, which is most pronounced

for higher-order tensors, i.e., an order-N tensor model with $\prod_{n=1}^{N} I_n$ parameters reduce to a model with $\sum_{n=1}^{N} I_n$ parameters.

2.4.4 Connectivity of a Tensor

The tensor structure can naturally be modeled as a graph by exploiting the property of lattice-structured graphs which is that they can be decomposed into constituent single-dimensional path graphs. The Cartesian product of M disjoint I_m-vertex path graphs, with $\mathcal{G}_m = (\mathcal{V}_m, \mathcal{E}_m)$ for $m = 1, \ldots, M$, yields an M-dimensional regular lattice structured graph, defined as

$$\mathcal{G} = \mathcal{G}_M \square \cdots \square \mathcal{G}_1 = (\mathcal{V}, \mathcal{E}), \tag{2.50}$$

with the resulting vertex set $\mathcal{V} = \mathcal{V}_M \times \cdots \times \mathcal{V}_1$. The total number of vertices in the resulting graph is then $\prod_{m=1}^{M} I_m = N$.

Denoting the adjacency matrix of the mth path graph $\mathcal{G}_m$ as $\mathbf{A}_m \in \mathbb{R}^{I_m \times I_m}$, the adjacency matrix of the resulting M-dimensional regular lattice graph $\mathcal{G}$ is given by

$$\mathbf{A} = (\mathbf{A}_M \oplus \cdots \oplus \mathbf{A}_1) \in \mathbb{R}^{N \times N}. \tag{2.51}$$

Such an adjacency matrix is said to be *Kronecker summable* [52].

2.4.5 DFT of a Tensor

From (2.51), we can consider tensors as a special class of graphs which exhibit a Kronecker summable adjacency matrix [52]. Therefore, the DFT of a tensor can be naturally obtained from the graph Fourier transform (GFT), which is explained detailly in [50]. The GFT of a graph with a lattice structure can be performed by evaluating the eigenvalue decomposition of the adjacency matrix $\mathbf{A}$, given by

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{-1},$$

where $\mathbf{U} \in \mathbb{R}^{N \times N}$ and $\mathbf{\Lambda} \in \mathbb{R}^{N \times N}$ respectively denote the matrices of eigenvectors and eigenvalues of $\mathbf{A}$. Due to the Kronecker sum structure of $\mathbf{A}$ in (2.51), we can write

$$\mathbf{U} = (\mathbf{U}_M \otimes \cdots \otimes \mathbf{U}_1),$$
$$\mathbf{\Lambda} = (\mathbf{\Lambda}_M \oplus \cdots \oplus \mathbf{\Lambda}_1),$$

where $\mathbf{U}_m \in \mathbb{R}^{I_m \times I_m}$ and $\mathbf{\Lambda}_m \in \mathbb{R}^{I_m \times I_m}$ respectively denote the matrices of eigenvectors and eigenvalues of the mth path graph's adjacency matrix $\mathbf{A}_m$, which is then given by

$$\mathbf{A}_m = \mathbf{U}_m \mathbf{\Lambda}_m \mathbf{U}_m^{-1}.$$

Therefore, the eigenvectors of $\mathbf{A}$ are said to be Kronecker separable, while the eigenvalues are Kronecker summable [52].

A graph signal resulting from an irregular sampling of a field represented by a tensor, $\mathcal{G}$ (a lattice-structured graph), is an unstructured graph, $\tilde{\mathcal{G}}$, with $\tilde{\mathcal{G}} \subset \mathcal{G}$. Although the lattice-structured graph, $\mathcal{G}$, exhibits a Kronecker separable signal vector and a Kronecker summable adjacency matrix, the associated unstructured graph, $\tilde{\mathcal{G}}$, does not have such properties [52]. In other words, since a graph signal resulting from an irregular sampling of a lattice-structured field cannot be represented as a Cartesian product of two path graphs, it is also not Kronecker separable.

2.4.6 Tensor Representation of Multi-Relational Graphs

The rapidly growing prominence of multi-relational network data has brought to light the increasing importance of Data Analytics on domains where the entities are interconnected by multiple relations [52]. Traditional graph models only account for a single relation type, described by the adjacency matrix, $\mathbf{A} \in \mathbb{R}^{N \times N}$. A multi-relational N-vertex graph may, however, account for a larger number, say M, of distinct relation types between vertices, with a separate adjacency matrix $\mathbf{A}_m \in \mathbb{R}^{N \times N}$ for each relation type, i.e. for $m = 1, \ldots, M$.

To model such a multi-relational graph, we construct a third-order tensor, $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{N \times N \times M}$, where the mth frontal slice is given by $\mathbf{A}_m$. In this way, the first two modes represent the entity domain, while the third mode represents the relation domain as illustrated in 2.13. In other words, the tensor entry $\boldsymbol{\mathcal{A}}_{ijk} = 1$ describes the existence of a relation between the ith and jth entries within the kth relation type. The rank-L factorization, referred to as RESCAL

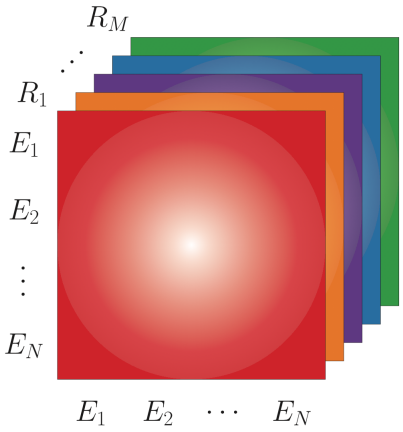


Figure 2.13: A multi-relational adjacency tensor, $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{N \times N \times M}$, where E_n denotes the nth entity and R_m the mth relation type [52]

decomposition in [40], of $\boldsymbol{\mathcal{A}}$ is given by

$$\mathbf{A}_m = \mathbf{U}\mathbf{R}_m\mathbf{U}^T, \quad \text{for } m = 1, \ldots, M, \tag{2.52}$$

where $\mathbf{U} \in \mathbb{R}^{N \times L}$ is a factor matrix, mapping the N-dimensional entity space to an L-dimensional latent space, and matrices $\mathbf{R}_m \in \mathbb{R}^{L \times L}$ model the interactions of latent components within the mth relation type. In terms of the factorization of tensor $\boldsymbol{\mathcal{A}}$, (2.52) can also be expressed as

$$\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{R}} \times_1 \mathbf{U} \times_2 \mathbf{U}, \tag{2.53}$$

where $\boldsymbol{\mathcal{R}} \in \mathbb{R}^{L \times L \times M}$ is the latent core tensor with frontal slices $\mathbf{R}_m$, as illustrated in Figure 2.14. Such a factorization allows for link-based clustering, whereby the entities $E_1, \ldots, E_N$ are clustered according to the information in $\mathbf{U}$ only. In this way, the similarity between entities is computed based on their similarity across multiple relations [52].

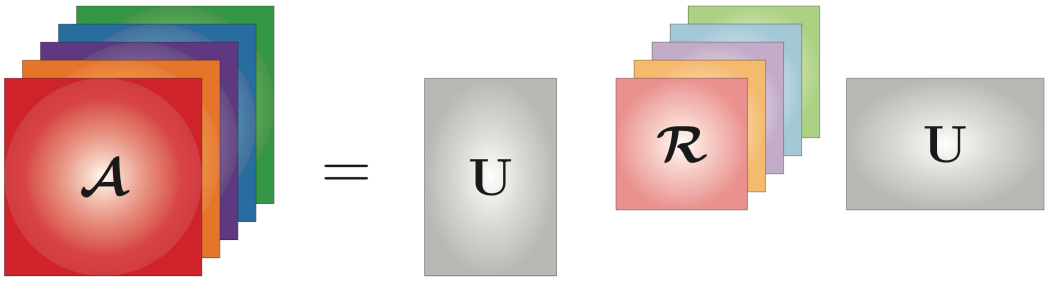


Figure 2.14: Factorization of a multi-relational adjacency tensor, $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{N \times N \times M}$ [52]

2.4.7 Multi-Graph Tensor Networks

Multi-Graph Tensor Network (MGTN) model aims to fully exploit the virtues of both graphs and tensors in a deep learning setting [56]. The MGTN framework is capable of: [52]

- Handling irregular data residing on multiple graph domains,
- Leveraging on the compression and structure-preserving properties of tensor networks, to enhance the expressive power of NNs, at a reduced parameter complexity.

The MGTN generalizes the Recurrent Graph Tensor Network (RGTN) model introduced in [58], which enables deep modelling on irregular domains and was developed to model time-series problems related to sequential data; it was only defined for a single graph domain. Aiming to make the framework introduced with RGTN suitable for applications in a Big Data setting, the MGTN operates in a multi-modal data setting defined on multiple graph domains. An illustration, provided by [52], of the fast-MGTN is given in Figure 2.15.

Tensorization of the dense layers to reduce parameter complexity is briefly discussed in [52]: "After extracting the feature map, it is customary to flatten the extracted features into a vector in order to pass them through dense neural network layers to generate the fMGTN output. To further reduce parameter complexity, the weight matrices of the dense layers can be tensorized and represented in some compressed tensor format, as discussed in Novikov et al. [42], Cichocki et al. [11][12], Calvi et al. [7]. This not only further reduces the number of parameters, but also maintains compatibility with the inherent multi-modal nature of the problem."

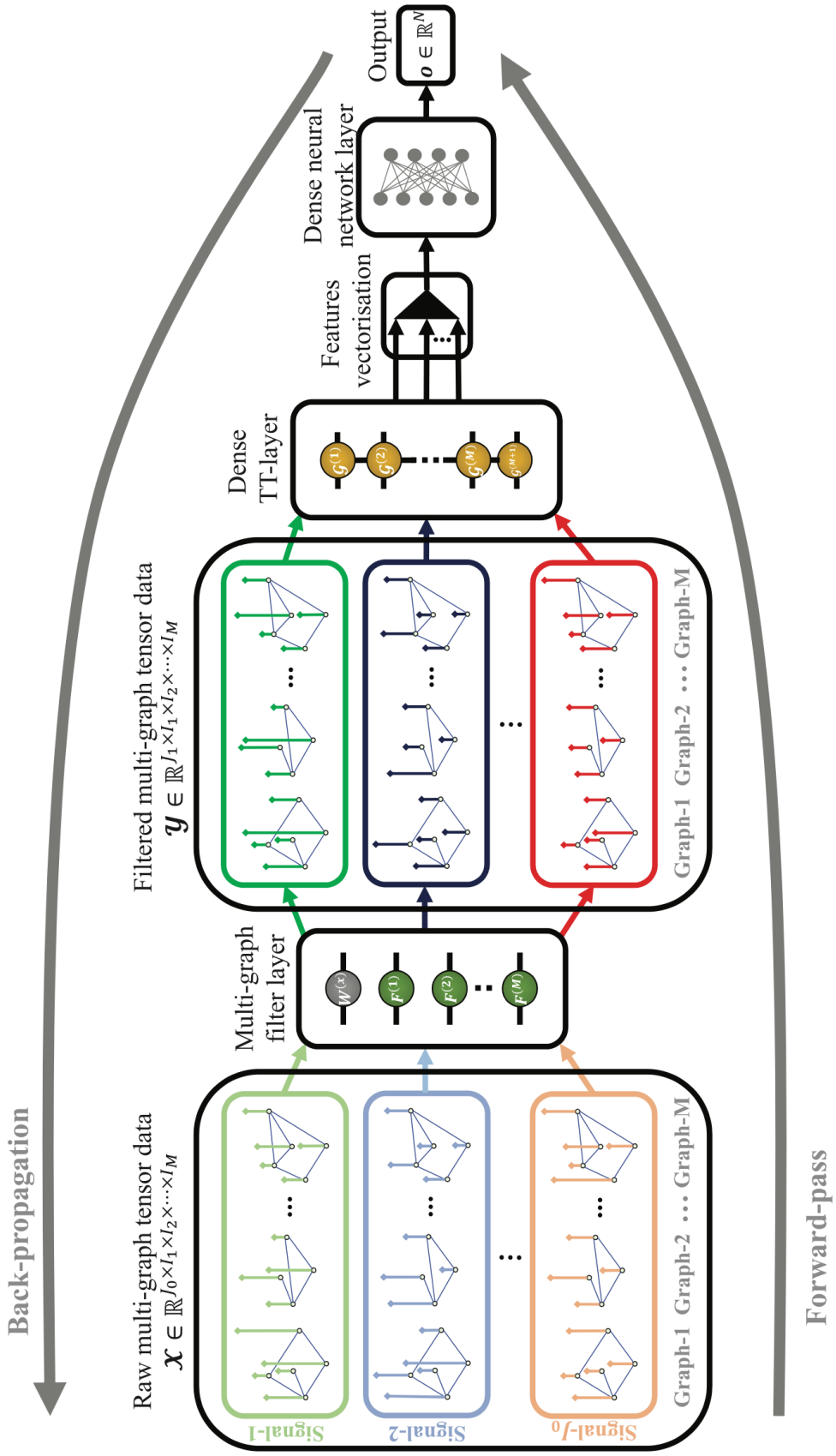


Figure 2.15: The structure of a fast multi-graph tensor network (fMGTN) [52]

2.4.8 Other Related Work on Using Tensors and Tensor Decompositions on Graphs

Image Representation and Learning With Graph-Laplacian Tucker Tensor Decomposition [25]

Data Analytics on Graphs Part III: Machine Learning on Graphs, from Graph Topology to Applications: Chapter 11 [52]

Graph Regularized Tensor Train Decomposition [49]

Interpretable Feature Learning of Graphs using Tensor Decomposition [20]

Signal processing on heterogeneous network based on tensor decomposition [46]

Chapter 3

Implementation Plan

We will be representing multi-dimensional, multi-modal financial big data acquired in an irregular domain as a graph. To determine the underlying graph topology, we will represent assets as vertices and use a relevant "vertex similarity" scheme (perhaps the correlation matrix) to define the edges. Then, we can use relevant GSP techniques such as vertex clustering or down-sampling to regularize the data. We can then tensorize the graph to represent the data in a regular lattice structured domain.

We can then use a chosen tensor decomposition technique (CPD, TD, TT) to extract latent features from the data and use this in a Tensor Ensemble Learning framework to predict results.

We will also look further into Multi-Graph Tensor Network (MGTN) framework and graph based filters introduced in that application. We will look into incorporating that framework with that of Tensor Ensemble Learning.

Chapter 4

Evaluation Plan

We will test the proposed framework on the given historical financial data, and we will evaluate our results indicated by the use of proper loss functions.

We will then compare our results against the benchmark results from other relevant models.

Chapter 5

Ethical, Legal, and Safety Plan

5.1 Safety Issues

Various safety issues for the project are considered and discussed below:

- Data Infrastructure safety

  There are no apparent concerns regarding this issue at the moment. This will become more clear when the data is shared.

- Electrical safety

  There are no electrical equipment involved in the project that constitute an electrical hazard.

- Physical safety

  There are no large or fast-moving objects involved in the project that can cause harm to myself or others.

- Chemical safety

  There are no poisonous, irritant or allergenic materials involved in the project.

- Fire safety

  There are no materials or equipment involved in the project that constitute a fire hazard.

- Biological Safety

  There are no materials involved in the project that constitute a possible biological hazard.

- Animal safety

  There are no animals involved in the project.

- Appliance safety

  There are no lab equipment involved in the project.

- Airspace safety

  There are no drones, rockets or aerial devices involved in the project.

- Study Participant safety

  There are no study participants involved in the project.

5.2 Legal Issues

The only legal issue relevant to the project work depends on the terms and services of the data that is to be provided. Therefore, depending on the terms and services, the use of the data will be restricted to research purposes only and will not be used for any commercial activity, nor will it be distributed to other parties.

5.3 Ethical Issues

There are no relevant ethical or moral issues related to the project.

Bibliography

[1] Davide Bacciu and Danilo P. Mandic. *Tensor Decompositions in Deep Learning*. 2020. arXiv: 2002.11835 [cs.LG].

[2] Sasmita Barik, Ravindra Bapat, and Sukanta Pati. "On the Laplacian spectra of product graphs". In: *Applicable Analysis and Discrete Mathematics* 9 (Apr. 2015), pp. 39–58. DOI: 10.2298/AADM150218006B.

[3] Dario Bini, Milvio Capovani, Francesco Romani, and Grazia Lotti. "O(n2.7799) complexity for n × n approximate matrix multiplication". In: *Information Processing Letters* 8.5 (1979), pp. 234–235. ISSN: 0020-0190. DOI: https://doi.org/10.1016/0020-0190(79)90113-3. URL: https://www.sciencedirect.com/science/article/pii/0020019079901133.

[4] Vladimir Boginski, Sergiy Butenko, and Panos M. Pardalos. "On Structural Properties of the Market Graph". In: *Innovations in Financial and Economic Networks* (2003), pp. 29–45.

[5] Neil Calkin and Marcos Lopez de Prado. "Stochastic flow diagrams". In: *Algorithmic Finance* 3 (2014), pp. 21–42. DOI: 10.3233/AF-140032.

[6] Neil Calkin and Marcos Lopez de Prado. "The Topology of Macro Financial Flows: An Application of Stochastic Flow Diagrams". In: *Algorithmic Finance* 3 (2014), pp. 43–85. DOI: 10.2139/ssrn.2379319.

[7] Giuseppe G. Calvi, Ahmad Moniri, Mahmoud Mahfouz, Qibin Zhao, and Danilo P. Mandic. *Compression and Interpretability of Deep Neural Networks via Tucker Tensor*

Layer: From First Principles to Tensor Valued Back-Propagation. 2020. arXiv: 1903.06133 [cs.LG].

[8] J. Douglas Carroll and Jih-Jie Chang. "Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition". In: *Psychometrika* 35.3 (1970), pp. 283–319. DOI: 10.1007/BF02310791. URL: https://doi.org/10.1007/BF02310791.

[9] Eva. Ceulemans and Henk A. L. Kiers. "Selecting among three-mode principal component models of different types and complexities: A numerical convex hull based method". In: *British Journal of Mathematical and Statistical Psychology* 59.1 (2006), pp. 133–150. DOI: https://doi.org/10.1348/000711005X64817.

[10] Andrzej Cichocki. *Era of Big Data Processing: A New Approach via Tensor Networks and Tensor Decompositions*. 2014. arXiv: 1403.2048 [cs.ET].

[11] Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, and Danilo P. Mandic. "Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part 1 Low-Rank Tensor Decompositions". In: *Foundations and Trends® in Machine Learning* 9.4-5 (2016), pp. 249–429. ISSN: 1935-8245. DOI: 10.1561/2200000059. URL: http://dx.doi.org/10.1561/2200000059.

[12] Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, Masashi Sugiyama, and Danilo P. Mandic. "Tensor Networks for Dimensionality Reduction and Large-scale Optimization: Part 2 Applications and Future Perspectives". In: *Foundations and Trends® in Machine Learning* 9.6 (2017), pp. 249–429. ISSN: 1935-8245. DOI: 10.1561/2200000067. URL: http://dx.doi.org/10.1561/2200000067.

[13] Andrzej Cichocki, Danilo Mandic, Lieven De Lathauwer, Guoxu Zhou, Qibin Zhao, Cesar Caiafa, and HUY ANH PHAN. "Tensor Decompositions for Signal Processing Applications: From two-way to multiway component analysis". In: *IEEE Signal Processing Magazine* 2 (Mar. 2015), pp. 145–163. DOI: 10.1109/msp.2013.2297439.

[14] Pierre Comon, Gene Golub, Lek-Heng Lim, and Bernard Mourrain. *Symmetric tensors and symmetric tensor rank*. 2008. arXiv: 0802.1681 [math.NA].

[15] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. "A Multilinear Singular Value Decomposition". In: *SIAM Journal on Matrix Analysis and Applications* 21.4 (2000), pp. 1253–1278. DOI: 10.1137/S0895479896305696.

[16] Inderjit S. Dhillon, Yuqiang Guan, and Brian Kulis. "Kernel K-Means: Spectral Clustering and Normalized Cuts". In: *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2004, pp. 551–556. DOI: 10.1145/1014052.1014118.

[17] Andrew Duncan. "Powers of the adjacency matrix and the walk matrix". In: (2017).

[18] Carl Eckart and Gale Young. "The approximation of one matrix by another of lower rank". In: *Psychometrika* 1.3 (1936), pp. 211–218. DOI: 10.1007/BF02288367. URL: https://doi.org/10.1007/BF02288367.

[19] *Faddeev–LeVerrier algorithm*. URL: https://en.wikipedia.org/wiki/Faddeev%E2%80%93LeVerrier_algorithm.

[20] Shah Muhammad Hamdi and Rafal Angryk. "Interpretable Feature Learning of Graphs using Tensor Decomposition". In: (Nov. 2019). DOI: 10.1109/ICDM.2019.00037.

[21] R. Harshman. "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis". In: (1970).

[22] Johan Håstad. "Tensor rank is NP-complete". In: *Journal of Algorithms* 11.4 (1990), pp. 644–654. ISSN: 0196-6774. DOI: https://doi.org/10.1016/0196-6774(90)90014-6. URL: https://www.sciencedirect.com/science/article/pii/0196677490900146.

[23] Frank L. Hitchcock. "The Expression of a Tensor or a Polyadic as a Sum of Products". In: *Journal of Mathematics and Physics* 6.1-4 (1927), pp. 164–189. DOI: https://doi.org/10.1002/sapm192761164. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/sapm192761164. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/sapm192761164.

[24] Anil K. Jain. "Data clustering: 50 years beyond K-means". In: *Pattern Recognition Letters* 31.8 (2010), pp. 651–666. DOI: https://doi.org/10.1016/j.patrec.2009.09.011.

[25] B. Jiang, C. Ding, J. Tang, and B. Luo. "Image Representation and Learning With Graph-Laplacian Tucker Tensor Decomposition". In: *IEEE Transactions on Cybernetics* 49.4 (2019), pp. 1417–1426. DOI: 10.1109/TCYB.2018.2802934.

[26] Arie Kapteyn, Heinz Neudecker, and Tom Wansbeek. "An approach to n-mode components analysis". In: *Psychometrika* 51.2 (1986), pp. 269–275.

[27] Samir Khuller. *Approximation algorithms for finding highly connected subgraphs*. Tech. rep. 1998.

[28] Henk A. L. Kiers. "Towards a standardized notation and terminology in multiway analysis". In: *Journal of Chemometrics* 14.3 (2000), pp. 105–122. DOI: https://doi.org/10.1002/1099-128X(200005/06)14:3<105::AID-CEM582>3.0.CO;2-I.

[29] Ilia Kisil, Ahmad Moniri, and Danilo P. Mandic. *Tensor Ensemble Learning for Multidimensional Data*. 2018. arXiv: 1812.06888 [eess.SP].

[30] Jon Kleinberg and Eva Tardos. *Algorithm design*. Pearson Education India, 2006.

[31] Tamara G. Kolda and Brett W. Bader. "Tensor Decompositions and Applications". In: *SIAM Review* 51.3 (2009), pp. 455–500. DOI: 10.1137/07070111X. eprint: https://doi.org/10.1137/07070111X. URL: https://doi.org/10.1137/07070111X.

[32] Pieter M. Kroonenberg and Jan de Leeuw. "Principal component analysis of three-mode data by means of alternating least squares algorithms". In: *Psychometrika* 45.1 (1980), pp. 69–97. DOI: 10.1007/BF02293599.

[33] J. B. Kruskal. "Rank, Decomposition, and Uniqueness for 3-Way and n-Way Arrays". In: *Multiway Data Analysis*. NLD: North-Holland Publishing Co., 1989, pp. 7–18. ISBN: 0444874100.

[34] Joseph B. Kruskal. "Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics". In: *Linear Algebra and its Applications* 18.2 (1977), pp. 95–138. ISSN: 0024-3795. DOI: https://doi.org/10.1016/0024-3795(77)90069-6. URL: https://www.sciencedirect.com/science/article/pii/0024379577900696.

[35] Jure Leskovec and Christos Faloutsos. "Sampling from Large Graphs". In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2006, pp. 631–636. DOI: 10.1145/1150402.1150479.

[36] Marcos Lopez de Prado. "Building Diversified Portfolios that Outperform Out-of-Sample". In: *Journal of Portfolio Management* (2016). DOI: 10.2139/ssrn.2708678.

[37] Harry Markowitz. "Portfolio selection". In: *The Journal of Finance* 7.1 (1952), pp. 77–91. DOI: https://doi.org/10.1111/j.1540-6261.1952.tb01525.x.

[38] J. Mocks. "Topographic components model for event-related potentials and some biophysical considerations". In: *IEEE Transactions on Biomedical Engineering* 35.6 (1988), pp. 482–484. DOI: 10.1109/10.2119.

[39] O.J. Morris, M.de Lee, and A. Constantinides. "Graph theory for image analysis: an approach based on the shortest spanning tree". In: *Communications, Radar and Signal Processing, IEE Proceedings F* 133 (May 1986), pp. 146–152. DOI: 10.1049/ip-f-1:19860025.

[40] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. "A Three-Way Model for Collective Learning on Multi-Relational Data". In: (2011), pp. 809–816.

[41] Chrysostomos L. Nikias and Athina P. Petropulu. *Higher-order spectra analysis : a nonlinear signal processing framework / Chrysostomos L. Nikias, Athina P. Petropulu*. English. PTR Prentice Hall Englewood Cliffs, N.J, 1993, xxii, 537 p. : ISBN: 0136782108.

[42] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. "Tensorizing Neural Networks". In: *Advances in Neural Information Processing Systems* (2015), pp. 442–450.

[43] A. -H. Phan, A. Cichocki, A. Uschmajew, P. Tichavský, G. Luta, and D. P. Mandic. "Tensor Networks for Latent Variable Analysis: Novel Algorithms for Tensor Train Approximation". In: *IEEE Transactions on Neural Networks and Learning Systems* 31.11 (2020), pp. 4622–4636. DOI: 10.1109/TNNLS.2019.2956926.

[44] Federico Pozzi, Bing Liu, Enza Messina, and Elisabetta Fersini. *Sentiment analysis in social networks*. Elsevier, 2017.

[45] Mailoc López de Prado Marcos. *Advances in financial machine learning*. Wiley, 2018.

[46] Y. Qiao, K. Niu, and Z. He. "Signal processing on heterogeneous network based on tensor decomposition". In: (2016), pp. 200–204. DOI: 10.1109/ICNIDC.2016.7974564.

[47] Satu Schaeffer. "Graph Clustering". In: *Computer Science Review* 1 (Aug. 2007), pp. 27–64. DOI: 10.1016/j.cosrev.2007.05.001.

[48] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE Signal Processing Magazine* 30.3 (May 2013), pp. 83–98. ISSN: 1053-5888. DOI: 10.1109/msp.2012.2235192. URL: http://dx.doi.org/10.1109/MSP.2012.2235192.

[49] Seyyid Emre Sofuoglu and Selin Aviyente. *Graph Regularized Tensor Train Decomposition*. 2019. arXiv: 1911.01591 [eess.IV].

[50] Ljubisa Stankovic, Danilo Mandic, Milos Dakovic, Milos Brajovic, Bruno Scalzo, and Anthony G. Constantinides. *Graph Signal Processing – Part II: Processing and Analyzing Signals on Graphs*. 2019. arXiv: 1909.10325 [cs.IT].

[51] Ljubisa Stankovic, Danilo Mandic, Milos Dakovic, Milos Brajovic, Bruno Scalzo, and Tony Constantinides. *Graph Signal Processing – Part I: Graphs, Graph Spectra, and Spectral Clustering*. 2019. arXiv: 1907.03467 [cs.IT].

[52] Ljubisa Stankovic, Danilo Mandic, Milos Dakovic, Milos Brajovic, Bruno Scalzo, Shengxi Li, and Anthony G. Constantinides. *Graph Signal Processing – Part III: Machine Learning on Graphs, from Graph Topology to Applications*. 2020. arXiv: 2001.00426 [cs.IT].

[53] L. R. Tucker. "Implications of factor analysis of three-way matrices for measurement of change". In: *Problems in measuring change.* Ed. by C. W. Harris. University of Wisconsin Press, 1963, pp. 122–137.

[54] Ledyard R. Tucker. "Some mathematical notes on three-mode factor analysis". In: *Psychometrika* 31.3 (1966), pp. 279–311. DOI: 10.1007/BF02289464.

[55] M. Alex O. Vasilescu and Demetri Terzopoulos. "Multilinear Analysis of Image Ensembles: TensorFaces". In: *Computer Vision, Lecture Notes in Comput. Sci. 2350*. Springer Berlin Heidelberg, 2002, pp. 447–460.

[56] Yao Xu, Kriton Konstantinidis, and Danilo Mandic. "Multi-Graph Tensor Networks". In: *Advances in Neural Information Processing Systems* (2020).

[57] Yao Lei Xu and Danilo P. Mandic. "Recurrent Graph Tensor Networks". In: (2020). arXiv: 2009.08727 [cs.LG].

[58] Yao Lei Xu and Danilo P. Mandic. *Recurrent Graph Tensor Networks*. 2020. arXiv: 2009.08727 [cs.LG].