Introduction to the Book

Being a computer science graduate interested in research and further applying my skills where needed, I came across topics in computational science for my master's and PhD experimentations that required more fundamental mathematical understanding than what I was taught. Any typical computer science degree includes a minimum of five Mathematics courses, and two Physics courses, leaving many concepts (including tensors and their applications) utterly foreign to computing graduates, who will self-study if their work needs them. Since I suffered in self-studying these concepts with very little guidance and math books targetting mathematicians only, I thought about writing this book. I would not consider myself an authority on tensor computing and all its fundamental math requirements; I simply want to communicate my self-learning quest, which hopefully will be helpful for some readers from a similar educational background. I aim to summarise my experience and shed some light on the missing concepts that could have made the material easier if studied in sequence and from a computer scientist to computer scientists. The book idea started in 2013, with an outline and aims only defined. The actual writing started in 2021 because of teaching loads and incomplete research positions. All along, the aim remained to enable vivid understanding and visualisation of these concepts through practicals and tutorials and to discuss conceptual aspects from first principles rather than just listing the definitions, notations, applications, libraries, and algorithms. The python code provided aims to link the high-level code that builds on a massive stack of building blocks to the required theoretical understanding that is often hard to find in one book. Other books or surveys would either fall into the theoretical side (building intuition using text explanation), mathematical side (using equations and deep dive into proofs and properties), or programming higher-level code examples without much explanation of the theory behind it. I aimed to combine the three levels of explanations to draw a big picture that does not lack details of implementation and theory.

The big picture of the current advances in tensor computing is published in papers; surveys are the most educational. The underlying building blocks are assembled from many books, mainly written for mathematicians, that would detail all proofs, properties, and circular definitions. This book summarises introductory/fundamental material from different topics needed to understand the current applications of tensor methods, and each has its many books written at various levels of abstraction and targeting different audiences. A computer scientist would find it hard to pull the threads together coherently to link the statistical machine learning algorithms with the kernel methods in Hilbert space, to the tensor methods in the Riemannian space, and then to learning representation theory with the abstract algebra potentials, and finally to deep learning and its complex generalisation potential.

It will be astonishing after this learning journey to find out that it all comes down to matrix multiplication to achieve coordinate basis change/projections in the different algorithms. The change of basis aims to achieve a better representation of a dataset. This better representation might be a dimensionality reduction, such as the methods in chapter two. The aim might be to disentangle sources or map to a higher dimensional space in which a non-linearly separable dataset, in its current dimension, will be linearly separable in the higher dimension. Mapping to higher

dimensions includes the Kernel method introduced in chapter two and further explained in chapter five. Another abstraction of machine learning models is the type of learning. Inference learning is learning the function that maps input to output from the data; this can be done statically or using data visualisation methods. Deductive learning is learning the coefficients of a given function from the data, such as the regression methods as function approximators which are the fundamental building block for all supervised parametric machine learning algorithms, including Deep Neural Networks. Then transductive learning learns the values of an unknown function for a given small dataset or points of interest, such as the encode-decoder models. This is the first attempt the author is aware of to align the tensor decomposition methods from first principles in linear algebra generalised to multi-linear Algebra. Then, compare their applications and performance evaluations as a generalisation to the matrix decomposition methods and their applications and performance evaluation. This alignment and succession of presentations should enable a vivid understanding of the topic.

The prerequisite knowledge to benefit from this book is mainly some problem-solving skills, exposure to algorithms, and programming experience in any language. Python code can quickly be learned from online tutorials. The use of Python as a popular data mining and machine learning language and providing sample source code, or referring to where a good tutorial is found, for every topic covered and application example should serve as building blocks that are ready for reuse in various applications. Most published papers and their open-source code use high-level calls to functions that build on a stack of mathematical functions that are sometimes parameterised on the high-level call. This book aims to close the gap between the high-level application of these concepts and the mathematical understanding of the operations performed under the hood.

Chapter one introduces linear algebra concepts in relation to machine learning requirements. Chapter two focuses on linear algebra algorithms that address learning the latent structure of a given matrix dataset. Those with a Linear algebra and dimensionality reduction algorithms background can skip the first two chapters, but the remaining chapters are sequential in order. Chapter three focuses on the multi-linear algebra concepts and introduces fundamental tensor decomposition methods. Chapter four expands the tensor decomposition methods to tensor networks' notation and decomposition, starts the applications with tensor completion and tensor regression and finally introduces neural networks and how to tensorise them. Those with previous tensor decomposition backgrounds or who wish to skip the mathematical foundation can skip chapter three. Chapter four understanding enables the sequential reading of chapters six and seven.

Chapter five is an additional topic that is usually omitted when discussing tensor methods. Tensors provide better representation, expressiveness and interpretability that is usually assumed to be understood. The chapter discusses representation learning aligned with methods from abstract algebra and traditional methods to identify how representation is usually performed traditionally in hand-tailored coordinate change and by using deep learning and regularisation. It then considers how tensor methods enrich the representation and how deep learning identifies the most suitable representation for a given dataset and task.

The applications in chapter six should widen the horizon for graduation/MSc thesis project topics rather than the black box approach to machine learning tasks. Many dataset sources discussed with their applications are not as popular as the known Kaggle and UCI datasets. Projects can aim to reproduce existing work or compare the performance of several approaches and identify

shortcomings and possible enhancements. Chapter seven discusses implementation optimisation using parallel programming models and available hardware architectures. Future trends and challenges are also discussed as material for research level that PhD and researchers might need to build on to enhance the methods explained in the book or address the discussed challenges and literature gaps. Although the book covers statistical analysis methods, machine learning and deep learning methods, it can not be considered a reference on these topics. These topics are considered application domains for tensor computing approaches, and this book omits many important details that can be learned from other references. Similarly, the linear algebra and multi-linear algebra concepts discussed are introductory to build for how they are used in the applications presented.

Tensor computing is difficult to be included in a data structures course and can only be an elective in an advanced year of an undergraduate computer science degree or equivalent. The course is preferably taught at a postgraduate level as well or used as a reference for practitioners and researchers working with high-dimensional data structures. It can also be introduced towards the end of a module on machine learning. Since tensor decomposition techniques solve many issues with high dimensional data in machine learning and deep learning, the book can be used to present the preliminaries needed to start applying the existing techniques to many problems. The gained understanding can help in participating in the currently active research on these methods to address many limitations. Topics covered may include introduction & definition, notation, libraries and packages, applications, and parallel programming models.

Machine Learning and Deep Learning algorithms are explained when used in the applications. The equations might be daunting, but they concisely explain a concept or algorithm better than Pseudocode and text. Getting used to reading equations in connection to the intuition explanation in the text and the source code examples enables deeper understanding and closes the gap between theoreticians, computer scientists, and practitioners in the field. If not clear, skip the equation and focus on the intuition and the source code examples. An errata web link is at https://book.manalhelal.com/tensor/front-matter/tensor-book-errata-review/. Please use it to correct any mistake, request a citation that I might have missed by mistake, further clarification, the inclusion of an algorithm or package review, or anything that can make this book more straightforward and complete. I did my best to cite all original contributors to the material I covered, but there is always a possibility of forgetting or using a newer citation where an older one is required. The aim is to make this book as self-contained as possible. All accompanying source code is published at https://github.com/mhelal/TensorsPyBook.

The book aims to unify the notation used in all chapters, with some inevitable violations explained when needed. The general notation rules are as follows:

- Mathematicians start indexing from 1, while computer scientists start indexing from 0.
- Scalars and vectors are represented in small letters such that vectors are indexed, while matrices are represented with capital letters. Tensors are represented with calligraphy letters, but sometimes with capital letters for simplicity or for generalising all tensor shapes. Running indices are in small letters such as i, j, and capital letters I, J denote the upper bound of an index in a mode. Indices are sometimes indicated as subscripts and sometimes between round brackets if more convenient.
- Pseudocode and source code are presented in a boxed outline.