Undergraduate Topics in Computer Science

Liang Wang • Jianxin Zhao • Richard Mortier

OCaml Scientific Computing

Functional Programming in Data Science and Artificial Intelligence

Springer

Liang Wang
Computer Science and Technology
University of Cambridge
Cambridge, UK

Jianxin Zhao
Computer Science and Technology
University of Cambridge
Cambridge, UK

Richard Mortier
Computer Science and Technology
University of Cambridge
Cambridge, UK

ISSN 1863-7310 ISSN 2197-1781 (electronic)
Undergraduate Topics in Computer Science
ISBN 978-3-030-97644-6 ISBN 978-3-030-97645-3 (eBook)
https://doi.org/10.1007/978-3-030-97645-3

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To my wife Maria, our daughters Matilda and Sofia, and my parents Yulin and Fengjin.

– Liang

To my parents and sister; to their unyielding love and support. To Gcores, for being there in my darkest hour.

– Jianxin

To all those who contributed, and those on whose work we built.

– Mort

Foreword

This book is about the harmonious synthesis of functional programming and numerical computation. In recent years, Data Science has taken off in a big way, partly through the rapid increase in availability of data from many sectors of society, whether industry, government or entertainment. There are obvious benefits in operational optimisation of utilities, in a wide variety of applications in healthcare, finance, and subtle analysis of human communication.

The growth in computing power, in handheld devices, on the desk top, in the server, on the cloud combines with this data availability to allow rapid development and deployment of platforms for statistics, Machine Learning and Artificial Intelligence applications. However, many of these systems have been rapid expansions of existing software, based in traditional imperative programming languages such as Python, or C++.

In contrast, functional programming has been the preserve of more formal, or even theoretical branches of computer science until relatively recently. However, steps in the implementation of real world systems in the declarative style have revealed certain long-claimed advantages for languages like OCaml or Haskell. Not only are they more safe (e.g. against security vulnerabilities), but the also now allow for more efficient, agile, and productive development. The languages also offer the hope of system verification, which may be an important property for safety critical application areas (health informatics, as mentioned before, being one obvious area).

Real world OCaml uses include finance, operating systems and cloud applications, and now numerical processing for scientific computation and statistics. Of course, there are now also other languages targeting safer computation in this domain that are still in the procedural or imperative style, including specialised to ML/AI such Julia, or to systems, like Rust.

The starting point for this book is in numerical computation, and the exposition shows with copious examples how OCaml and the Owl library can be used to build up the basic computational units needed to build basic statistical programs. Going beyond that, the authors build up to use cases drawn from many areas of Data Science, Machine Learning, and AI. The book then delves into how to deploy at scale, using parallel, distributed, and accelerated frameworks to gain all the advantages of the model data centre/cloud computing hardware and system environments.

The book does not currently cover Bayesian Inference, nor statistical techniques such as Monte Carlo Markov Chains although it would be fairly easy to build such systems based on material here. One interesting thing to try would be to construct a Domain Specific Language for probabilistic programming based on OCaml & Owl.

This book shows how the expressiveness of OCaml, allows for succinct exposition of applications, fast and safe development of data science applications. AI and ML use in science and engineering is often exploratory, and calls for an agile approach, not just fixed choice of algorithm, with different data, repeatedly applied but rather a rapid choice of algorithm, and organisation, whether neural net architecture, and what level of distributed, parallel, or accelerated deployment. If that is the fast moving world in which you live, then this is the book for you!

Prof. Jon Crowcroft
FRS FREng
Alan Turing Institute
Cambridge, UK
December 2021

Preface

Back in the summer of 2019, we were considering the maintenance of Owl's documentation. We were glad that documentation was serving us well and growing day by day. Then it somehow occurred to us that, now that we have such comprehensive documentation to hand plus many paper drafts and blog posts, why don't we put these different pieces together into something more cohesive – a tutorial book. And that's the root of this book.

We then discussed several possible ways to organise the book. Ultimately we decided to more or less follow the outlines of traditional textbooks in scientific computing rather than how modules are organised in Owl. However, we did not wish to make it yet another data science/machine learning/numerical computing algorithms book, and so you may find this book a little bit different to others.

The book is divided into three parts, each focusing on a different area. Part I begins by introducing how basic numerical techniques are performed in OCaml, including classical mathematical topics (interpolation and quadrature), statistics, and linear algebra. It moves on from using only scalar values to multi-dimensional arrays, introducing the tensor and Ndarray, core data types in any numerical computing system. It concludes with two more classical numerical computing topics, the solution of Ordinary Differential Equations (ODEs) and Signal Processing, as well as introducing the visualisation module we use throughout this book.

Part II is dedicated to advanced optimisation techniques that are core to most current popular data science fields, such as Deep Neural Networks. We do not focus only on applications but also on the basic building blocks, starting with Algorithmic Differentiation, the most crucial building block that in turn enables Deep Neural Networks. We follow this with chapters on Optimisation and Regression, also used in building Deep Neural Networks. We then introduce Deep Neural Networks as well as topic modelling in Natural Language Processing (NLP), two advanced and currently very active fields in both industry and academia.

Part III collects a range of case studies demonstrating how you can build a complete numerical application quickly from scratch using Owl. The cases presented include computer vision and recommender systems.

The book does not assume strict ordering in reading – we hope you can simply jump to the topic that interests you most. We do assume you are familiar with the basics of programming in OCaml. Details of the Owl library are given in its documentation, but this book provides a piece-by-piece introduction. Details of how to set up an Owl programming environment are given in the Appendices.

This book is aimed at anyone with a basic knowledge of functional programming and a desire to explore the world of scientific computing, whether to glimpse the field in the round, to build applications for particular topics, or to deep-dive into how numerical systems are constructed. If you can learn something from what we have been doing in the past several years, we will be happy that this book has succeeded.

However, as it *is* a little different to many others we feel obliged to note a couple of things that this book is explicitly not trying to do.

First, it is neither about the philosophy of functional programming, nor a general introduction either to functional programming or the OCaml language specifically. Rather, it introduces how to use the numerical functionality provided in the Owl library and OCaml language to build various applications and investigate different numerical computing topics. We avoid using very advanced functional programming techniques intentionally in the book, but we do assume that readers have basic understanding of either OCaml or similar languages such as Haskell.

Second, it is not trying to deep-dive into every topic in numerical computing. Almost all the chapters in the first two parts could each be extended to an entire book in its own right – instead, we seek to give a general introduction to the world of numerical computing via functional programming, striking a balance between breadth and depth. To do this we give an overview and introduce Owl's functionality for classical topics (e.g., linear algebra, statistics, optimisation); we present interesting applications and how to use Owl to address them for selected topics (e.g., signal processing, differential equations); and we explain in detail how Owl implements several key machine learning/artificial intelligence topics (e.g., algorithmic differentiation, regression, neural networks), rather than using the library as a "black box" as many alternative presentations do.

We hope you will find this valuable, and we welcome feedback on our approach!

Liang, Jianxin, Mort
Helsinki, Beijing, Cambridge
Jan 2022

Acknowledgements

Developing a complex system then writing a book on a big topic is certainly a very challenging task. It not only requires skills, enthusiasm, persistence, but also needs strong support from families, colleagues, and communities. For years, we have received so much help from so many individuals and organisations that it is almost impossible to make an exhaustive list. Nonetheless, we would particularly like to emphasise that Owl is developed on top of an enormous amount of previous work. Without the continuous efforts of these projects and the intellectual contributions of these people over the years, it would be impossible for us to create this system and deliver this book.

Contents

Acronyms

AD	Algorithmic Differentiation
BERT	Bidirectional Encoder Representations from Transformers
BOW	Bag of Words
CDF	Cumulative Distribution Function
CF	Collaborative Filtering
CNN	Convolutional Neural Network
CSC	Compressed Sparse Column
CSR	Compressed Sparse Row
DFT	Discrete Fourier Transform
DNN	Deep Neural Network
FFT	Fast Fourier Transform
FST	Fast Style Transfer
GAN	Generative Adversarial Network
GEMM	General Matrix Multiply
GRU	Gated Recurrent Unit
ILP	Integer Linear Programming
IR	Information Retrieval
KKT	Karush-Kuhn-Tucker conditions
LDA	Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
LSTM	Long/Short Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MKL	Math Kernel Library
MPE	Mean Percentage Error
MSE	Mean Square Error
NLG	Natural Language Generation
NLP	Natural Language Processing
NST	Neural Style Transfer
ODE	Ordinary Differential Equations
PDE	Partial Differential Equations

PDF	Probability Density Function
PMF	Probability Mass Function
REPL	Read-Eval-Print Loop
RK	Runge-Kutta method
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
RP-Tree	Random Projection Tree
RPN	Region Proposal Network
RSS	Residual Sum of Squares
SF	Survival Function
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TF-IDF	Term Frequency–Inverse Document Frequency
VSM	Vector Space Model