


Undergraduate Topics in Computer Science


Series Editor

Ian Mackie, University of Sussex, Brighton, UK


Advisory Editors

Samson Abramsky , Department of Computer Science, University of Oxford, Oxford, UK

Chris Hankin , Department of Computing, Imperial College London, London, UK


Mike Hinchey , Lero – The Irish Software Research Centre, University of Limerick, Limerick, Ireland

Dexter C. Kozen, Department of Computer Science, Cornell University, Ithaca, NY, USA

Andrew Pitts , Department of Computer Science and Technology, University of Cambridge, Cambridge, UK

Hanne Riis Nielson , Department of Applied Mathematics and Computer Science, Technical University of Denmark, Kongens Lyngby, Denmark

Steven S. Skiena, Department of Computer Science, Stony Brook University, Stony Brook, NY, USA

Iain Stewart , Department of Computer Science, Durham University, Durham, UK

‘Undergraduate Topics in Computer Science’ (UTiCS) delivers high-quality instructional content for undergraduates studying in all areas of computing and information science. From core foundational and theoretical material to final-year topics and applications, UTiCS books take a fresh, concise, and modern approach and are ideal for self-study or for a one- or two-semester course. The texts are all authored by established experts in their fields, reviewed by an international advisory board, and contain numerous examples and problems, many of which include fully worked solutions.

The UTiCS concept relies on high-quality, concise books in softback format, and generally a maximum of 275–300 pages. For undergraduate textbooks that are likely to be longer, more expository, Springer continues to offer the highly regarded Texts in Computer Science series, to which we refer potential authors.

More information about this series at <https://link.springer.com/bookseries/7592>

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>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2022

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors, and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

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 they 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 as 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 Narray, 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

Part I Numerical Techniques

| | | |
|----------|-----------------------------------|----|
| 1 | Introduction | 3 |
| 1.1 | Scientific Computing | 3 |
| 1.2 | Functional Programming | 4 |
| 1.3 | OCaml Numerical Library | 5 |
| 1.4 | System Architecture | 7 |
| 1.5 | How to Run Code | 8 |
| 1.6 | Summary | 9 |
| | References | 9 |
| 2 | Numerical Algorithms | 11 |
| 2.1 | Interpolation | 11 |
| 2.1.1 | Polynomial Interpolation | 11 |
| 2.1.2 | Rational Interpolation | 15 |
| 2.2 | Integration | 18 |
| 2.2.1 | Basic Algorithms | 19 |
| 2.2.2 | Gaussian Quadrature | 23 |
| 2.3 | Special Functions | 26 |
| 2.4 | Summary | 29 |
| | References | 29 |
| 3 | Statistics | 31 |
| 3.1 | Random Variables | 31 |
| 3.1.1 | Descriptive Statistics | 32 |
| 3.2 | Distribution | 33 |
| 3.2.1 | Discrete Distribution | 34 |
| 3.2.2 | Continuous Distribution | 36 |
| 3.2.3 | Special Distributions | 38 |
| 3.3 | Multiple Variables | 39 |
| 3.3.1 | Joint and Conditional Probability | 39 |

| | | |
|----------|---|-----------|
| 3.3.2 | Covariance and Correlation | 41 |
| 3.4 | Sampling | 42 |
| 3.4.1 | Unbiased Estimator | 43 |
| 3.4.2 | Inferring Population Parameters | 43 |
| 3.5 | Hypothesis Tests | 45 |
| 3.5.1 | Theory | 45 |
| 3.5.2 | Gaussian Distribution in Hypothesis Testing | 46 |
| 3.5.3 | Two-Sample Inferences | 47 |
| 3.5.4 | More Tests | 48 |
| 3.6 | Summary | 48 |
| | References | 49 |
| 4 | Linear Algebra | 51 |
| 4.1 | Vectors and Matrices | 51 |
| 4.2 | Gaussian Elimination | 53 |
| 4.2.1 | LU Factorisation | 54 |
| 4.2.2 | Inverse and Transpose | 57 |
| 4.3 | Vector Space | 57 |
| 4.3.1 | Rank and Basis | 58 |
| 4.3.2 | Orthogonality | 59 |
| 4.3.3 | Solving $Ax = b$ | 60 |
| 4.3.4 | Matrix Sensitivity | 62 |
| 4.4 | Determinants | 63 |
| 4.5 | Eigenvalues and Eigenvectors | 64 |
| 4.5.1 | Complex Matrices | 66 |
| 4.5.2 | Similarity Transformation and Diagonalisation | 67 |
| 4.6 | Positive Definite Matrices | 68 |
| 4.6.1 | Positive Definiteness | 68 |
| 4.6.2 | Singular Value Decomposition | 69 |
| 4.7 | Sparse Matrices | 72 |
| 4.8 | Summary | 74 |
| | References | 74 |
| 5 | N-Dimensional Arrays | 75 |
| 5.1 | Ndarray | 75 |
| 5.1.1 | Types | 75 |
| 5.1.2 | Ndarray Creation and Properties | 76 |
| 5.1.3 | Map, Fold, and Scan | 77 |
| 5.1.4 | Comparison Functions | 78 |
| 5.1.5 | Iteration Functions | 79 |
| 5.1.6 | Manipulation Functions | 79 |
| 5.1.7 | Serialisation | 81 |
| 5.2 | Slicing | 81 |
| 5.2.1 | Slice Definition | 81 |
| 5.2.2 | Conventions and Examples | 82 |

| | | |
|----------|--|------------|
| 5.2.3 | Advanced Usage | 85 |
| 5.3 | Broadcasting | 87 |
| 5.4 | Tensors | 89 |
| 5.5 | Summary | 92 |
| | References | 93 |
| 6 | Ordinary Differential Equations | 95 |
| 6.1 | Defining an ODE | 95 |
| 6.1.1 | Exact Solutions | 96 |
| 6.1.2 | Linear Systems | 96 |
| 6.2 | Solving ODEs Numerically | 97 |
| 6.3 | ODE Solvers | 100 |
| 6.3.1 | Solving Linear Oscillator System | 101 |
| 6.3.2 | Solver Structure | 101 |
| 6.3.3 | Symplectic Solver | 103 |
| 6.3.4 | Features and Limits | 104 |
| 6.4 | Examples of ODE Solvers | 105 |
| 6.4.1 | Explicit ODE | 105 |
| 6.4.2 | Two-body Problem | 106 |
| 6.4.3 | Lorenz Attractor | 108 |
| 6.4.4 | Damped Oscillation | 111 |
| 6.5 | Stiffness | 113 |
| 6.5.1 | Solving Non-Stiff ODEs | 114 |
| 6.5.2 | Solve Stiff ODEs | 115 |
| 6.6 | Summary | 116 |
| | References | 116 |
| 7 | Signal Processing | 119 |
| 7.1 | Discrete Fourier Transform | 119 |
| 7.2 | Fast Fourier Transform | 121 |
| 7.2.1 | Example: 1-D Discrete Fourier Transforms | 122 |
| 7.3 | Applications of FFTs | 124 |
| 7.3.1 | Finding the Period of Sunspots | 124 |
| 7.3.2 | Determine the Tone | 126 |
| 7.3.3 | Image Processing | 127 |
| 7.4 | Filtering | 130 |
| 7.4.1 | Example: Smoothing | 130 |
| 7.4.2 | Gaussian Filter | 131 |
| 7.4.3 | Signal Convolution | 133 |
| 7.4.4 | FFT and Image Convolution | 134 |
| 7.5 | Summary | 136 |
| | References | 136 |

Part II Advanced Data Analysis Techniques

| | | |
|----------|--|-----|
| 8 | Algorithmic Differentiation | 141 |
| 8.1 | Chain Rule | 141 |
| 8.2 | Differentiation Methods | 143 |
| 8.2.1 | Numerical Differentiation | 143 |
| 8.2.2 | Symbolic Differentiation | 144 |
| 8.2.3 | Algorithmic Differentiation | 145 |
| 8.3 | How Algorithmic Differentiation Works | 145 |
| 8.3.1 | Forward Mode | 146 |
| 8.3.2 | Reverse Mode | 148 |
| 8.3.3 | Forward Mode or Reverse Mode? | 150 |
| 8.4 | A Strawman AD Engine | 151 |
| 8.4.1 | Implementation of Forward Mode | 151 |
| 8.4.2 | Implementation of Reverse Mode | 153 |
| 8.4.3 | A Unified Implementation | 157 |
| 8.5 | Forward and Reverse Propagation API | 161 |
| 8.5.1 | Expressing Computation | 162 |
| 8.5.2 | Example: Forward Mode | 162 |
| 8.5.3 | Example: Reverse Mode | 163 |
| 8.6 | High-Level Functions | 163 |
| 8.6.1 | Derivative and Gradient | 164 |
| 8.6.2 | Jacobian | 165 |
| 8.6.3 | Hessian and Laplacian | 166 |
| 8.6.4 | Other APIs | 168 |
| 8.7 | Internals of Algorithmic Differentiation | 168 |
| 8.7.1 | Architecture and Components | 169 |
| 8.7.2 | Extending AD | 171 |
| 8.7.3 | Lazy Evaluation | 172 |
| 8.8 | Summary | 173 |
| | References | 173 |
| 9 | Optimisation | 175 |
| 9.1 | Objective Functions | 175 |
| 9.2 | Root Finding | 176 |
| 9.3 | Univariate Function Optimisation | 178 |
| 9.3.1 | Use Derivatives | 178 |
| 9.3.2 | Golden Section Search | 180 |
| 9.4 | Multivariate Function Optimisation | 180 |
| 9.4.1 | Nelder-Mead Simplex Method | 181 |
| 9.4.2 | Gradient Descent Methods | 182 |
| 9.4.3 | Conjugate Gradient Method | 185 |
| 9.4.4 | Newton and Quasi-Newton Methods | 186 |
| 9.5 | Global Optimisation and Constrained Optimisation | 188 |
| 9.6 | Summary | 189 |
| | References | 189 |

| | |
|--|-----|
| 10 Regression | 191 |
| 10.1 Linear Regression | 191 |
| 10.1.1 Problem: Where to open a new McDonald's restaurant? | 191 |
| 10.1.2 Cost Function | 193 |
| 10.1.3 Solving Problem with Gradient Descent | 194 |
| 10.2 Multiple Regression | 197 |
| 10.2.1 Feature Normalisation | 198 |
| 10.2.2 Analytical Solution | 200 |
| 10.3 Non-Linear Regressions | 201 |
| 10.4 Regularisation | 203 |
| 10.4.1 Ols, Ridge, Lasso, and Elastic_net | 204 |
| 10.5 Logistic Regression | 205 |
| 10.5.1 Sigmoid Function | 205 |
| 10.5.2 Cost Function | 206 |
| 10.5.3 Example | 207 |
| 10.5.4 Multi-class Classification | 210 |
| 10.6 Support Vector Machines | 210 |
| 10.6.1 Kernel and Non-Linear Boundary | 212 |
| 10.6.2 Example | 213 |
| 10.7 Model Error and Selection | 214 |
| 10.7.1 Error Metrics | 215 |
| 10.7.2 Model Selection | 216 |
| 10.8 Summary | 217 |
| References | 217 |
| 11 Neural Network | 219 |
| 11.1 The Perceptron | 219 |
| 11.2 Yet Another Regression | 220 |
| 11.2.1 Model Representation | 220 |
| 11.2.2 Forward Propagation | 221 |
| 11.2.3 Back Propagation | 222 |
| 11.2.4 Feedforward Network | 223 |
| 11.2.5 Layers | 223 |
| 11.2.6 Activation Functions | 224 |
| 11.2.7 Initialisation | 225 |
| 11.2.8 Training | 226 |
| 11.2.9 Test | 227 |
| 11.3 Neural Network Module | 228 |
| 11.3.1 Neurons | 229 |
| 11.3.2 Training Parameters | 232 |
| 11.4 Convolutional Neural Network | 235 |
| 11.5 Recurrent Neural Network | 236 |
| 11.6 Generative Adversarial Network | 239 |
| 11.7 Summary | 241 |
| References | 241 |

| | | |
|-----------|--|-----|
| 12 | Vector Space Modelling | 243 |
| 12.1 | Introduction | 243 |
| 12.2 | Text Corpus Analysis | 244 |
| 12.2.1 | Building a Text Corpus | 245 |
| 12.2.2 | Using Corpus Module | 246 |
| 12.3 | Vector Space Models | 248 |
| 12.3.1 | Word Embedding and BERT | 248 |
| 12.3.2 | Bag of Words (BOW) | 249 |
| 12.4 | Term Frequency–Inverse Document Frequency (TF-IDF) | 251 |
| 12.5 | Latent Dirichlet Allocation (LDA) | 253 |
| 12.5.1 | Models | 254 |
| 12.5.2 | Dirichlet Distribution | 256 |
| 12.5.3 | Gibbs Sampling | 256 |
| 12.5.4 | Topic Modelling Example | 257 |
| 12.6 | Latent Semantic Analysis | 259 |
| 12.7 | Search Relevant Documents | 260 |
| 12.7.1 | Euclidean and Cosine Similarity | 261 |
| 12.7.2 | Linear Searching | 262 |
| 12.8 | Summary | 263 |
| | References | 263 |

Part III Use Cases

| | | |
|-----------|--------------------------------------|-----|
| 13 | Case Study: Image Recognition | 267 |
| 13.1 | Types of Networks | 267 |
| 13.1.1 | LeNet | 267 |
| 13.1.2 | AlexNet | 268 |
| 13.1.3 | VGG | 269 |
| 13.1.4 | ResNet | 270 |
| 13.1.5 | SqueezeNet | 270 |
| 13.1.6 | Capsule Network | 271 |
| 13.2 | Building the InceptionV3 Network | 271 |
| 13.2.1 | InceptionV1 and InceptionV2 | 272 |
| 13.2.2 | Factorisation | 273 |
| 13.2.3 | Grid Size Reduction | 276 |
| 13.2.4 | InceptionV3 Architecture | 277 |
| 13.3 | Preparing Weights | 278 |
| 13.4 | Processing Image | 279 |
| 13.5 | Running Inference | 282 |
| 13.6 | Applications | 283 |
| 13.7 | Summary | 284 |
| | References | 284 |

| | |
|---|-----|
| 14 Case Study: Instance Segmentation | 287 |
| 14.1 Introduction | 287 |
| 14.2 Object Detection Architectures | 288 |
| 14.2.1 R-CNN | 289 |
| 14.2.2 Fast R-CNN | 289 |
| 14.2.3 Faster R-CNN | 290 |
| 14.2.4 Mask R-CNN | 290 |
| 14.3 Mask R-CNN Network | 291 |
| 14.3.1 Building Mask R-CNN | 291 |
| 14.3.2 Feature Extractor | 292 |
| 14.3.3 Proposal Generation | 293 |
| 14.3.4 Classification | 293 |
| 14.4 Run the Code | 295 |
| 14.5 Summary | 295 |
| References | 296 |
| 15 Case Study: Neural Style Transfer | 297 |
| 15.1 Content and Style | 297 |
| 15.1.1 Content Reconstruction | 298 |
| 15.1.2 Style Recreation | 301 |
| 15.1.3 Combining Content and Style | 303 |
| 15.1.4 Running NST | 304 |
| 15.2 Extending NST | 304 |
| 15.3 Fast Style Transfer | 305 |
| 15.3.1 Building FST Network | 305 |
| 15.3.2 Running FST | 308 |
| 15.4 Summary | 309 |
| References | 310 |
| 16 Case Study: Recommender System | 311 |
| 16.1 Introduction | 311 |
| 16.2 Key Components And Pipeline | 312 |
| 16.3 Reducing Dimensionality | 315 |
| 16.4 Random Projection | 316 |
| 16.5 Vector Storage Optimisation | 318 |
| 16.6 Combining Multiple Trees | 319 |
| 16.7 Critical Boundary Elimination | 320 |
| 16.8 Search Operations Parallelisation | 322 |
| 16.9 Code Implementation | 323 |
| 16.10 Summary | 327 |
| References | 327 |

| | | |
|-------------------|---------------------------|-----|
| Appendix A | Conventions of Owl | 329 |
| A.1 | Pure vs Impure | 329 |
| A.2 | Ndarray vs Scalar | 330 |
| A.3 | Infix Operators | 331 |
| A.4 | Module Structures | 331 |
| A.5 | Operator Extension | 333 |
| Appendix B | Visualisation | 337 |
| B.1 | Plotting in Owl | 337 |
| B.1.1 | Create Plots | 337 |
| B.1.2 | Specification | 338 |
| B.1.3 | Subplots | 340 |
| B.1.4 | Multiple Lines | 341 |
| B.1.5 | Legend | 343 |
| B.1.6 | Drawing Patterns | 344 |
| B.2 | Plot Types | 345 |
| Index | | 357 |

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 |