Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

Neural Networks and Deep Learning

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

Charu C. Aggarwal

Neural Networks and Deep Learning

A Textbook

Springer

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

Charu C. Aggarwal
IBM T. J. Watson Research Center
International Business Machines
Yorktown Heights, NY, USA

ISBN 978-3-319-94462-3 ISBN 978-3-319-94463-0 (eBook)
https://doi.org/10.1007/978-3-319-94463-0

Library of Congress Control Number: 2018947636

© Springer International Publishing AG, part of Springer Nature 2018
This work is subject to copyright. All rights are reserved 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, express 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

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

To my wife Lata, my daughter Sayani,
and my late parents Dr. Prem Sarup and Mrs. Pushplata Aggarwal.

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

Preface

> "Any A.I. smart enough to pass a Turing test is smart enough to know to fail it."—Ian McDonald

Neural networks were developed to simulate the human nervous system for machine learning tasks by treating the computational units in a learning model in a manner similar to human neurons. The grand vision of neural networks is to create artificial intelligence by building machines whose architecture simulates the computations in the human nervous system. This is obviously not a simple task because the computational power of the fastest computer today is a minuscule fraction of the computational power of a human brain. Neural networks were developed soon after the advent of computers in the fifties and sixties. Rosenblatt's perceptron algorithm was seen as a fundamental cornerstone of neural networks, which caused an initial excitement about the prospects of artificial intelligence. However, after the initial euphoria, there was a period of disappointment in which the data hungry and computationally intensive nature of neural networks was seen as an impediment to their usability. Eventually, at the turn of the century, greater data availability and increasing computational power lead to increased successes of neural networks, and this area was reborn under the new label of "deep learning." Although we are still far from the day that artificial intelligence (AI) is close to human performance, there are specific domains like image recognition, self-driving cars, and game playing, where AI has matched or exceeded human performance. It is also hard to predict what AI might be able to do in the future. For example, few computer vision experts would have thought two decades ago that any automated system could ever perform an intuitive task like categorizing an image more accurately than a human.

Neural networks are *theoretically* capable of learning any mathematical function with sufficient training data, and some variants like recurrent neural networks are known to be *Turing complete*. Turing completeness refers to the fact that a neural network can simulate any learning algorithm, *given sufficient training data*. The sticking point is that the amount of data required to learn even simple tasks is often extraordinarily large, which causes a corresponding increase in training time (if we assume that enough training data is available in the first place). For example, the training time for image recognition, which is a simple task for a human, can be on the order of weeks even on high-performance systems. Furthermore, there are practical issues associated with the stability of neural network training, which are being resolved even today. Nevertheless, given that the speed of computers is

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

expected to increase rapidly over time, and fundamentally more powerful paradigms like quantum computing are on the horizon, the computational issue might not eventually turn out to be quite as critical as imagined.

Although the biological analogy of neural networks is an exciting one and evokes comparisons with science fiction, the mathematical understanding of neural networks is a more mundane one. The neural network abstraction can be viewed as a modular approach of enabling learning algorithms that are based on continuous optimization on a computational graph of dependencies between the input and output. To be fair, this is not very different from traditional work in control theory; indeed, some of the methods used for optimization in control theory are strikingly similar to (and historically preceded) the most fundamental algorithms in neural networks. However, the large amounts of data available in recent years together with increased computational power have enabled experimentation with deeper architectures of these computational graphs than was previously possible. The resulting success has changed the broader perception of the potential of deep learning.

The chapters of the book are organized as follows:

1. *The basics of neural networks:* Chapter 1 discusses the basics of neural network design. Many traditional machine learning models can be understood as special cases of neural learning. Understanding the relationship between traditional machine learning and neural networks is the first step to understanding the latter. The simulation of various machine learning models with neural networks is provided in Chapter 2. This will give the analyst a feel of how neural networks push the envelope of traditional machine learning algorithms.

2. *Fundamentals of neural networks:* Although Chapters 1 and 2 provide an overview of the training methods for neural networks, a more detailed understanding of the training challenges is provided in Chapters 3 and 4. Chapters 5 and 6 present radial-basis function (RBF) networks and restricted Boltzmann machines.

3. *Advanced topics in neural networks:* A lot of the recent success of deep learning is a result of the specialized architectures for various domains, such as recurrent neural networks and convolutional neural networks. Chapters 7 and 8 discuss recurrent and convolutional neural networks. Several advanced topics like deep reinforcement learning, neural Turing mechanisms, and generative adversarial networks are discussed in Chapters 9 and 10.

We have taken care to include some of the "forgotten" architectures like RBF networks and Kohonen self-organizing maps because of their potential in many applications. The book is written for graduate students, researchers, and practitioners. Numerous exercises are available along with a solution manual to aid in classroom teaching. Where possible, an application-centric view is highlighted in order to give the reader a feel for the technology.

Throughout this book, a vector or a multidimensional data point is annotated with a bar, such as $\overline{X}$ or $\overline{y}$. A vector or multidimensional point may be denoted by either small letters or capital letters, as long as it has a bar. Vector dot products are denoted by centered dots, such as $\overline{X} \cdot \overline{Y}$. A matrix is denoted in capital letters without a bar, such as R. Throughout the book, the $n \times d$ matrix corresponding to the entire training data set is denoted by D, with n documents and d dimensions. The individual data points in D are therefore d-dimensional row vectors. On the other hand, vectors with one component for each data

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

point are usually n-dimensional column vectors. An example is the n-dimensional column vector $\overline{y}$ of class variables of n data points. An observed value y_i is distinguished from a predicted value $\hat{y}_i$ by a circumflex at the top of the variable.

Yorktown Heights, NY, USA

Charu C. Aggarwal

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

Acknowledgments

I would like to thank my family for their love and support during the busy time spent in writing this book. I would also like to thank my manager Nagui Halim for his support during the writing of this book.

Several figures in this book have been provided by the courtesy of various individuals and institutions. The Smithsonian Institution made the image of the Mark I perceptron (cf. Figure 1.5) available at no cost. Saket Sathe provided the outputs in Chapter 7 for the tiny Shakespeare data set, based on code available/described in [233, 580]. Andrew Zisserman provided Figures 8.12 and 8.16 in the section on convolutional visualizations. Another visualization of the feature maps in the convolution network (cf. Figure 8.15) was provided by Matthew Zeiler. NVIDIA provided Figure 9.10 on the convolutional neural network for self-driving cars in Chapter 9, and Sergey Levine provided the image on self-learning robots (cf. Figure 9.9) in the same chapter. Alec Radford provided Figure 10.8, which appears in Chapter 10. Alex Krizhevsky provided Figure 8.9(b) containing *AlexNet*.

This book has benefitted from significant feedback and several collaborations that I have had with numerous colleagues over the years. I would like to thank Quoc Le, Saket Sathe, Karthik Subbian, Jiliang Tang, and Suhang Wang for their feedback on various portions of this book. Shuai Zheng provided feedbback on the section on regularized autoencoders in Chapter 4. I received feedback on the sections on autoencoders from Lei Cai and Hao Yuan. Feedback on the chapter on convolutional neural networks was provided by Hongyang Gao, Shuiwang Ji, and Zhengyang Wang. Shuiwang Ji, Lei Cai, Zhengyang Wang and Hao Yuan also reviewed the Chapters 3 and 7, and suggested several edits. They also suggested the ideas of using Figures 8.6 and 8.7 for elucidating the convolution/deconvolution operations.

For their collaborations, I would like to thank Tarek F. Abdelzaher, Jinghui Chen, Jing Gao, Quanquan Gu, Manish Gupta, Jiawei Han, Alexander Hinneburg, Thomas Huang, Nan Li, Huan Liu, Ruoming Jin, Daniel Keim, Arijit Khan, Latifur Khan, Mohammad M. Masud, Jian Pei, Magda Procopiuc, Guojun Qi, Chandan Reddy, Saket Sathe, Jaideep Srivastava, Karthik Subbian, Yizhou Sun, Jiliang Tang, Min-Hsuan Tsai, Haixun Wang, Jianyong Wang, Min Wang, Suhang Wang, Joel Wolf, Xifeng Yan, Mohammed Zaki, ChengXiang Zhai, and Peixiang Zhao. I would also like to thank my advisor James B. Orlin for his guidance during my early years as a researcher.

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

I would like to thank Lata Aggarwal for helping me with some of the figures created using PowerPoint graphics in this book. My daughter, Sayani, was helpful in incorporating special effects (e.g., image color, contrast, and blurring) in several JPEG images used at various places in this book.

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

Contents

1 An Introduction to Neural Networks 1
- 1.1 Introduction . . . 1
  - 1.1.1 Humans Versus Computers: Stretching the Limits of Artificial Intelligence . . . 3
- 1.2 The Basic Architecture of Neural Networks . . . 4
  - 1.2.1 Single Computational Layer: The Perceptron . . . 5
    - 1.2.1.1 What Objective Function Is the Perceptron Optimizing? . 8
    - 1.2.1.2 Relationship with Support Vector Machines . . . 10
    - 1.2.1.3 Choice of Activation and Loss Functions . . . 11
    - 1.2.1.4 Choice and Number of Output Nodes . . . 14
    - 1.2.1.5 Choice of Loss Function . . . 14
    - 1.2.1.6 Some Useful Derivatives of Activation Functions . . . 16
  - 1.2.2 Multilayer Neural Networks . . . 17
  - 1.2.3 The Multilayer Network as a Computational Graph . . . 20
- 1.3 Training a Neural Network with Backpropagation . . . 21
- 1.4 Practical Issues in Neural Network Training . . . 24
  - 1.4.1 The Problem of Overfitting . . . 25
    - 1.4.1.1 Regularization . . . 26
    - 1.4.1.2 Neural Architecture and Parameter Sharing . . . 27
    - 1.4.1.3 Early Stopping . . . 27
    - 1.4.1.4 Trading Off Breadth for Depth . . . 27
    - 1.4.1.5 Ensemble Methods . . . 28
  - 1.4.2 The Vanishing and Exploding Gradient Problems . . . 28
  - 1.4.3 Difficulties in Convergence . . . 29
  - 1.4.4 Local and Spurious Optima . . . 29
  - 1.4.5 Computational Challenges . . . 29
- 1.5 The Secrets to the Power of Function Composition . . . 30
  - 1.5.1 The Importance of Nonlinear Activation . . . 32
  - 1.5.2 Reducing Parameter Requirements with Depth . . . 34
  - 1.5.3 Unconventional Neural Architectures . . . 35
    - 1.5.3.1 Blurring the Distinctions Between Input, Hidden, and Output Layers . . . 35
    - 1.5.3.2 Unconventional Operations and Sum-Product Networks . . 36

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

1.6 Common Neural Architectures . . . 37
- 1.6.1 Simulating Basic Machine Learning with Shallow Models . . . 37
- 1.6.2 Radial Basis Function Networks . . . 37
- 1.6.3 Restricted Boltzmann Machines . . . 38
- 1.6.4 Recurrent Neural Networks . . . 38
- 1.6.5 Convolutional Neural Networks . . . 40
- 1.6.6 Hierarchical Feature Engineering and Pretrained Models . . . 42

1.7 Advanced Topics . . . 44
- 1.7.1 Reinforcement Learning . . . 44
- 1.7.2 Separating Data Storage and Computations . . . 45
- 1.7.3 Generative Adversarial Networks . . . 45

1.8 Two Notable Benchmarks . . . 46
- 1.8.1 The MNIST Database of Handwritten Digits . . . 46
- 1.8.2 The ImageNet Database . . . 47

1.9 Summary . . . 48

1.10 Bibliographic Notes . . . 48
- 1.10.1 Video Lectures . . . 50
- 1.10.2 Software Resources . . . 50

1.11 Exercises . . . 51

2 Machine Learning with Shallow Neural Networks 53

2.1 Introduction . . . 53

2.2 Neural Architectures for Binary Classification Models . . . 55
- 2.2.1 Revisiting the Perceptron . . . 56
- 2.2.2 Least-Squares Regression . . . 58
  - 2.2.2.1 Widrow-Hoff Learning . . . 59
  - 2.2.2.2 Closed Form Solutions . . . 61
- 2.2.3 Logistic Regression . . . 61
  - 2.2.3.1 Alternative Choices of Activation and Loss . . . 63
- 2.2.4 Support Vector Machines . . . 63

2.3 Neural Architectures for Multiclass Models . . . 65
- 2.3.1 Multiclass Perceptron . . . 65
- 2.3.2 Weston-Watkins SVM . . . 67
- 2.3.3 Multinomial Logistic Regression (Softmax Classifier) . . . 68
- 2.3.4 Hierarchical Softmax for Many Classes . . . 69

2.4 Backpropagated Saliency for Feature Selection . . . 70

2.5 Matrix Factorization with Autoencoders . . . 70
- 2.5.1 Autoencoder: Basic Principles . . . 71
  - 2.5.1.1 Autoencoder with a Single Hidden Layer . . . 72
  - 2.5.1.2 Connections with Singular Value Decomposition . . . 74
  - 2.5.1.3 Sharing Weights in Encoder and Decoder . . . 74
  - 2.5.1.4 Other Matrix Factorization Methods . . . 76
- 2.5.2 Nonlinear Activations . . . 76
- 2.5.3 Deep Autoencoders . . . 78
- 2.5.4 Application to Outlier Detection . . . 80
- 2.5.5 When the Hidden Layer Is Broader than the Input Layer . . . 81
  - 2.5.5.1 Sparse Feature Learning . . . 81
- 2.5.6 Other Applications . . . 82

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

2.5.7 Recommender Systems: Row Index to Row Value Prediction 83
2.5.8 Discussion 86
2.6 Word2vec: An Application of Simple Neural Architectures 87
2.6.1 Neural Embedding with Continuous Bag of Words 87
2.6.2 Neural Embedding with Skip-Gram Model 90
2.6.3 Word2vec (SGNS) Is Logistic Matrix Factorization 95
2.6.4 Vanilla Skip-Gram Is Multinomial Matrix Factorization 98
2.7 Simple Neural Architectures for Graph Embeddings 98
2.7.1 Handling Arbitrary Edge Counts 100
2.7.2 Multinomial Model 100
2.7.3 Connections with DeepWalk and Node2vec 100
2.8 Summary 101
2.9 Bibliographic Notes 101
2.9.1 Software Resources 102
2.10 Exercises 103

3 Training Deep Neural Networks 105
3.1 Introduction 105
3.2 Backpropagation: The Gory Details 107
3.2.1 Backpropagation with the Computational Graph Abstraction 107
3.2.2 Dynamic Programming to the Rescue 111
3.2.3 Backpropagation with Post-Activation Variables 113
3.2.4 Backpropagation with Pre-activation Variables 115
3.2.5 Examples of Updates for Various Activations 117
3.2.5.1 The Special Case of Softmax 117
3.2.6 A Decoupled View of Vector-Centric Backpropagation 118
3.2.7 Loss Functions on Multiple Output Nodes and Hidden Nodes 121
3.2.8 Mini-Batch Stochastic Gradient Descent 121
3.2.9 Backpropagation Tricks for Handling Shared Weights 123
3.2.10 Checking the Correctness of Gradient Computation 124
3.3 Setup and Initialization Issues 125
3.3.1 Tuning Hyperparameters 125
3.3.2 Feature Preprocessing 126
3.3.3 Initialization 128
3.4 The Vanishing and Exploding Gradient Problems 129
3.4.1 Geometric Understanding of the Effect of Gradient Ratios 130
3.4.2 A Partial Fix with Activation Function Choice 133
3.4.3 Dying Neurons and “Brain Damage” 133
3.4.3.1 Leaky ReLU 133
3.4.3.2 Maxout 134
3.5 Gradient-Descent Strategies 134
3.5.1 Learning Rate Decay 135
3.5.2 Momentum-Based Learning 136
3.5.2.1 Nesterov Momentum 137
3.5.3 Parameter-Specific Learning Rates 137
3.5.3.1 AdaGrad 138
3.5.3.2 RMSProp 138
3.5.3.3 RMSProp with Nesterov Momentum 139

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

3.5.3.4 AdaDelta 139
3.5.3.5 Adam 140
3.5.4 Cliffs and Higher-Order Instability 141
3.5.5 Gradient Clipping 142
3.5.6 Second-Order Derivatives 143
3.5.6.1 Conjugate Gradients and Hessian-Free Optimization 145
3.5.6.2 Quasi-Newton Methods and BFGS 148
3.5.6.3 Problems with Second-Order Methods: Saddle Points . . . 149
3.5.7 Polyak Averaging 151
3.5.8 Local and Spurious Minima 151
3.6 Batch Normalization 152
3.7 Practical Tricks for Acceleration and Compression 156
3.7.1 GPU Acceleration 157
3.7.2 Parallel and Distributed Implementations 158
3.7.3 Algorithmic Tricks for Model Compression 160
3.8 Summary 163
3.9 Bibliographic Notes 163
3.9.1 Software Resources 165
3.10 Exercises 165

4 Teaching Deep Learners to Generalize 169
4.1 Introduction 169
4.2 The Bias-Variance Trade-Off 174
4.2.1 Formal View 175
4.3 Generalization Issues in Model Tuning and Evaluation 178
4.3.1 Evaluating with Hold-Out and Cross-Validation 179
4.3.2 Issues with Training at Scale 180
4.3.3 How to Detect Need to Collect More Data 181
4.4 Penalty-Based Regularization 181
4.4.1 Connections with Noise Injection 182
4.4.2 L_1-Regularization 183
4.4.3 L_1- or L_2-Regularization? 184
4.4.4 Penalizing Hidden Units: Learning Sparse Representations 185
4.5 Ensemble Methods 186
4.5.1 Bagging and Subsampling 186
4.5.2 Parametric Model Selection and Averaging 187
4.5.3 Randomized Connection Dropping 188
4.5.4 Dropout 188
4.5.5 Data Perturbation Ensembles 191
4.6 Early Stopping 192
4.6.1 Understanding Early Stopping from the Variance Perspective 192
4.7 Unsupervised Pretraining 193
4.7.1 Variations of Unsupervised Pretraining 197
4.7.2 What About Supervised Pretraining? 197
4.8 Continuation and Curriculum Learning 199
4.8.1 Continuation Learning 199
4.8.2 Curriculum Learning 200
4.9 Parameter Sharing 200

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

4.10 Regularization in Unsupervised Applications . . . 201
  4.10.1 Value-Based Penalization: Sparse Autoencoders . . . 202
  4.10.2 Noise Injection: De-noising Autoencoders . . . 202
  4.10.3 Gradient-Based Penalization: Contractive Autoencoders . . . 204
  4.10.4 Hidden Probabilistic Structure: Variational Autoencoders . . . 207
    4.10.4.1 Reconstruction and Generative Sampling . . . 210
    4.10.4.2 Conditional Variational Autoencoders . . . 212
    4.10.4.3 Relationship with Generative Adversarial Networks . . . 213
4.11 Summary . . . 213
4.12 Bibliographic Notes . . . 214
  4.12.1 Software Resources . . . 215
4.13 Exercises . . . 215

5 Radial Basis Function Networks 217
5.1 Introduction . . . 217
5.2 Training an RBF Network . . . 220
  5.2.1 Training the Hidden Layer . . . 221
  5.2.2 Training the Output Layer . . . 222
    5.2.2.1 Expression with Pseudo-Inverse . . . 224
  5.2.3 Orthogonal Least-Squares Algorithm . . . 224
  5.2.4 Fully Supervised Learning . . . 225
5.3 Variations and Special Cases of RBF Networks . . . 226
  5.3.1 Classification with Perceptron Criterion . . . 226
  5.3.2 Classification with Hinge Loss . . . 227
  5.3.3 Example of Linear Separability Promoted by RBF . . . 227
  5.3.4 Application to Interpolation . . . 228
5.4 Relationship with Kernel Methods . . . 229
  5.4.1 Kernel Regression as a Special Case of RBF Networks . . . 229
  5.4.2 Kernel SVM as a Special Case of RBF Networks . . . 230
  5.4.3 Observations . . . 231
5.5 Summary . . . 231
5.6 Bibliographic Notes . . . 232
5.7 Exercises . . . 232

6 Restricted Boltzmann Machines 235
6.1 Introduction . . . 235
  6.1.1 Historical Perspective . . . 236
6.2 Hopfield Networks . . . 237
  6.2.1 Optimal State Configurations of a Trained Network . . . 238
  6.2.2 Training a Hopfield Network . . . 240
  6.2.3 Building a Toy Recommender and Its Limitations . . . 241
  6.2.4 Increasing the Expressive Power of the Hopfield Network . . . 242
6.3 The Boltzmann Machine . . . 243
  6.3.1 How a Boltzmann Machine Generates Data . . . 244
  6.3.2 Learning the Weights of a Boltzmann Machine . . . 245
6.4 Restricted Boltzmann Machines . . . 247
  6.4.1 Training the RBM . . . 249
  6.4.2 Contrastive Divergence Algorithm . . . 250
  6.4.3 Practical Issues and Improvisations . . . 251

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

6.5 Applications of Restricted Boltzmann Machines 251
6.5.1 Dimensionality Reduction and Data Reconstruction 252
6.5.2 RBMs for Collaborative Filtering 254
6.5.3 Using RBMs for Classification 257
6.5.4 Topic Models with RBMs 260
6.5.5 RBMs for Machine Learning with Multimodal Data 262
6.6 Using RBMs Beyond Binary Data Types 263
6.7 Stacking Restricted Boltzmann Machines 264
6.7.1 Unsupervised Learning 266
6.7.2 Supervised Learning 267
6.7.3 Deep Boltzmann Machines and Deep Belief Networks 267
6.8 Summary 268
6.9 Bibliographic Notes 268
6.10 Exercises 270

7 Recurrent Neural Networks 271
7.1 Introduction 271
7.1.1 Expressiveness of Recurrent Networks 274
7.2 The Architecture of Recurrent Neural Networks 274
7.2.1 Language Modeling Example of RNN 277
7.2.1.1 Generating a Language Sample 278
7.2.2 Backpropagation Through Time 280
7.2.3 Bidirectional Recurrent Networks 283
7.2.4 Multilayer Recurrent Networks 284
7.3 The Challenges of Training Recurrent Networks 286
7.3.1 Layer Normalization 289
7.4 Echo-State Networks 290
7.5 Long Short-Term Memory (LSTM) 292
7.6 Gated Recurrent Units (GRUs) 295
7.7 Applications of Recurrent Neural Networks 297
7.7.1 Application to Automatic Image Captioning 298
7.7.2 Sequence-to-Sequence Learning and Machine Translation 299
7.7.2.1 Question-Answering Systems 301
7.7.3 Application to Sentence-Level Classification 303
7.7.4 Token-Level Classification with Linguistic Features 304
7.7.5 Time-Series Forecasting and Prediction 305
7.7.6 Temporal Recommender Systems 307
7.7.7 Secondary Protein Structure Prediction 309
7.7.8 End-to-End Speech Recognition 309
7.7.9 Handwriting Recognition 309
7.8 Summary 310
7.9 Bibliographic Notes 310
7.9.1 Software Resources 311
7.10 Exercises 312

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

8 Convolutional Neural Networks 315

8.1 Introduction . . . 315
8.1.1 Historical Perspective and Biological Inspiration . . . 316
8.1.2 Broader Observations About Convolutional Neural Networks . . . 317
8.2 The Basic Structure of a Convolutional Network . . . 318
8.2.1 Padding . . . 322
8.2.2 Strides . . . 324
8.2.3 Typical Settings . . . 324
8.2.4 The ReLU Layer . . . 325
8.2.5 Pooling . . . 326
8.2.6 Fully Connected Layers . . . 327
8.2.7 The Interleaving Between Layers . . . 328
8.2.8 Local Response Normalization . . . 330
8.2.9 Hierarchical Feature Engineering . . . 331
8.3 Training a Convolutional Network . . . 332
8.3.1 Backpropagating Through Convolutions . . . 333
8.3.2 Backpropagation as Convolution with Inverted/Transposed Filter . . . 334
8.3.3 Convolution/Backpropagation as Matrix Multiplications . . . 335
8.3.4 Data Augmentation . . . 337
8.4 Case Studies of Convolutional Architectures . . . 338
8.4.1 AlexNet . . . 339
8.4.2 ZFNet . . . 341
8.4.3 VGG . . . 342
8.4.4 GoogLeNet . . . 345
8.4.5 ResNet . . . 347
8.4.6 The Effects of Depth . . . 350
8.4.7 Pretrained Models . . . 351
8.5 Visualization and Unsupervised Learning . . . 352
8.5.1 Visualizing the Features of a Trained Network . . . 353
8.5.2 Convolutional Autoencoders . . . 357
8.6 Applications of Convolutional Networks . . . 363
8.6.1 Content-Based Image Retrieval . . . 363
8.6.2 Object Localization . . . 364
8.6.3 Object Detection . . . 365
8.6.4 Natural Language and Sequence Learning . . . 366
8.6.5 Video Classification . . . 367
8.7 Summary . . . 368
8.8 Bibliographic Notes . . . 368
8.8.1 Software Resources and Data Sets . . . 370
8.9 Exercises . . . 371

9 Deep Reinforcement Learning 373

9.1 Introduction . . . 373
9.2 Stateless Algorithms: Multi-Armed Bandits . . . 375
9.2.1 Naïve Algorithm . . . 376
9.2.2 ϵ-Greedy Algorithm . . . 376
9.2.3 Upper Bounding Methods . . . 376
9.3 The Basic Framework of Reinforcement Learning . . . 377
9.3.1 Challenges of Reinforcement Learning . . . 379

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

9.3.2 Simple Reinforcement Learning for Tic-Tac-Toe . . . 380
9.3.3 Role of Deep Learning and a Straw-Man Algorithm . . . 380
9.4 Bootstrapping for Value Function Learning . . . 383
9.4.1 Deep Learning Models as Function Approximators . . . 384
9.4.2 Example: Neural Network for Atari Setting . . . 386
9.4.3 On-Policy Versus Off-Policy Methods: SARSA . . . 387
9.4.4 Modeling States Versus State-Action Pairs . . . 389
9.5 Policy Gradient Methods . . . 391
9.5.1 Finite Difference Methods . . . 392
9.5.2 Likelihood Ratio Methods . . . 393
9.5.3 Combining Supervised Learning with Policy Gradients . . . 395
9.5.4 Actor-Critic Methods . . . 395
9.5.5 Continuous Action Spaces . . . 397
9.5.6 Advantages and Disadvantages of Policy Gradients . . . 397
9.6 Monte Carlo Tree Search . . . 398
9.7 Case Studies . . . 399
9.7.1 AlphaGo: Championship Level Play at Go . . . 399
9.7.1.1 Alpha Zero: Enhancements to Zero Human Knowledge . . . 402
9.7.2 Self-Learning Robots . . . 404
9.7.2.1 Deep Learning of Locomotion Skills . . . 404
9.7.2.2 Deep Learning of Visuomotor Skills . . . 406
9.7.3 Building Conversational Systems: Deep Learning for Chatbots . . . 407
9.7.4 Self-Driving Cars . . . 410
9.7.5 Inferring Neural Architectures with Reinforcement Learning . . . 412
9.8 Practical Challenges Associated with Safety . . . 413
9.9 Summary . . . 414
9.10 Bibliographic Notes . . . 414
9.10.1 Software Resources and Testbeds . . . 416
9.11 Exercises . . . 416

10 Advanced Topics in Deep Learning 419
10.1 Introduction . . . 419
10.2 Attention Mechanisms . . . 421
10.2.1 Recurrent Models of Visual Attention . . . 422
10.2.1.1 Application to Image Captioning . . . 424
10.2.2 Attention Mechanisms for Machine Translation . . . 425
10.3 Neural Networks with External Memory . . . 429
10.3.1 A Fantasy Video Game: Sorting by Example . . . 430
10.3.1.1 Implementing Swaps with Memory Operations . . . 431
10.3.2 Neural Turing Machines . . . 432
10.3.3 Differentiable Neural Computer: A Brief Overview . . . 437
10.4 Generative Adversarial Networks (GANs) . . . 438
10.4.1 Training a Generative Adversarial Network . . . 439
10.4.2 Comparison with Variational Autoencoder . . . 442
10.4.3 Using GANs for Generating Image Data . . . 442
10.4.4 Conditional Generative Adversarial Networks . . . 444
10.5 Competitive Learning . . . 449
10.5.1 Vector Quantization . . . 450
10.5.2 Kohonen Self-Organizing Map . . . 450

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

10.6 Limitations of Neural Networks 453
  10.6.1 An Aspirational Goal: One-Shot Learning 453
  10.6.2 An Aspirational Goal: Energy-Efficient Learning 455
10.7 Summary 456
10.8 Bibliographic Notes 457
  10.8.1 Software Resources 458
10.9 Exercises 458

Bibliography **459**

Index **493**

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0

Author Biography

Charu C. Aggarwal is a Distinguished Research Staff Member (DRSM) at the IBM T. J. Watson Research Center in Yorktown Heights, New York. He completed his undergraduate degree in Computer Science from the Indian Institute of Technology at Kanpur in 1993 and his Ph.D. from the Massachusetts Institute of Technology in 1996. He has worked extensively in the field of data mining. He has published more than 350 papers in refereed conferences and journals and authored over 80 patents. He is the author or editor of 18 books, including textbooks on data mining, recommender systems, and outlier analysis. Because of the commercial value of his patents, he has thrice been designated a Master Inventor at IBM. He is a recipient of an IBM Corporate Award (2003) for his work on bio-terrorist threat detection in data streams, a recipient of the IBM Outstanding Innovation Award (2008) for his scientific contributions to privacy technology, and a recipient of two IBM Outstanding Technical Achievement Awards (2009, 2015) for his work on data streams/high-dimensional data. He received the EDBT 2014 Test of Time Award for his work on condensation-based privacy-preserving data mining. He is also a recipient of the IEEE ICDM Research Contributions Award (2015), which is one of the two highest awards for influential research contributions in the field of data mining.

He has served as the general co-chair of the IEEE Big Data Conference (2014) and as the program co-chair of the ACM CIKM Conference (2015), the IEEE ICDM Conference (2015), and the ACM KDD Conference (2016). He served as an associate editor of the IEEE Transactions on Knowledge and Data Engineering from 2004 to 2008. He is an associate editor of the IEEE Transactions on Big Data, an action editor of the Data Mining and Knowledge Discovery Journal, and an associate editor of the Knowledge and Information Systems Journal. He serves as the editor-in-chief of the ACM Transactions on Knowledge Discovery from Data as well as the ACM SIGKDD Explorations. He serves on the advisory board of the Lecture Notes on Social Networks, a publication by Springer. He has served as the vice-president of the SIAM Activity Group on Data Mining and is a member of the SIAM industry committee. He is a fellow of the SIAM, ACM, and the IEEE, for “contributions to knowledge discovery and data mining algorithms.”

Computers connected to subscribing institutions download at:
https://rd.springer.com/book/10.1007/978-3-319-94463-0