

Studies in Computational Intelligence

Volume 673

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

About this Series

The series “Studies in Computational Intelligence” (SCI) publishes new developments and advances in the various areas of computational intelligence—quickly and with a high quality. The intent is to cover the theory, applications, and design methods of computational intelligence, as embedded in the fields of engineering, computer science, physics and life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in computational intelligence spanning the areas of neural networks, connectionist systems, genetic algorithms, evolutionary computation, artificial intelligence, cellular automata, self-organizing systems, soft computing, fuzzy systems, and hybrid intelligent systems. Of particular value to both the contributors and the readership are the short publication timeframe and the worldwide distribution, which enable both wide and rapid dissemination of research output.

More information about this series at <http://www.springer.com/series/7092>

Wojciech Wieczorek

Grammatical Inference

Algorithms, Routines and Applications

Wojciech Wiecek
Institute of Computer Science
University of Silesia Faculty of Computer
Science and Materials Science
Sosnowiec
Poland

ISSN 1860-949X ISSN 1860-9503 (electronic)
Studies in Computational Intelligence
ISBN 978-3-319-46800-6 ISBN 978-3-319-46801-3 (eBook)
DOI 10.1007/978-3-319-46801-3

Library of Congress Control Number: 2016952872

© Springer International Publishing AG 2017

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.

Printed on acid-free paper

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

Preface

Grammatical inference, the main topic of this book, is a scientific area that lies at the intersection of multiple fields. Researchers from computational linguistics, pattern recognition, machine learning, computational biology, formal learning theory, and many others have their own contribution. Therefore, it is not surprising that the topic has also a few other names such as grammar learning, automata inference, grammar identification, or grammar induction. To simplify the location of present contribution, we can divide all books relevant to grammatical inference into three groups: theoretical, practical, and applicable. In greater part this book is practical, though one can also find the elements of learning theory, combinatorics on words, the theory of automata and formal languages, plus some reference to real-life problems.

The purpose of this book is to present old and modern methods of grammatical inference from the perspective of practitioners. To this end, the Python programming language has been chosen as the way of presenting all the methods. Included listings can be directly used by the paste-and-copy manner to other programs, thus students, academic researchers, and programmers should find this book as the valuable source of ready recipes and as an inspiration for their further development.

A few issues should be mentioned regarding this book: an inspiration to write it, a key for the selection of described methods, arguments for selecting Python as an implementation language, typographical notions, and where the reader can send any critical remarks about the content of the book (subject-matter, listings etc.).

There is a treasured book entitled “Numerical recipes in C”, in which along with the description of selected numerical methods, listings in C language are provided. The reader can copy and paste the fragments of the electronic version of the book in order to produce executable programs. Such an approach is very useful. We can find an idea that lies behind a method and immediately put it into practice. It is a guiding principle that accompanied writing the present book.

For the selection of methods, we try to keep balance between importance and complexity. It means that we introduced concepts and algorithms which are essential to the GI practice and theory, but omitted that are too complicated or too

long to present them as a ready-to-use code. Thanks to that, the longest program included in the book is no more than a few pages long.

As far as the implementation language is concerned, the following requirements had to be taken into account: simplicity, availability, the property of being firmly established, and allowing the use of wide range of libraries. Python and FSharp programming languages were good candidates. We decided to choose IronPython (an implementation of Python) mainly due to its integration with the optimization modeling language. We use a monospaced (fixed-pitch) font for the listings of programs, while the main text is written using a proportional font. In listings, Python keywords are in bold.

The following persons have helped the author in preparing the final version of this book by giving valuable advice. I would like to thank (in alphabetical order): Prof. Z.J. Czech (Silesian University of Technology), Dr. P. Juszczuk, Ph.D. student A. Nowakowski, Dr. R. Skinderowicz, and Ph.D. student L. Strak (University of Silesia).

Sosnowiec, Poland
2016

Wojciech Wieczorek

Contents

1	Introduction	1
1.1	The Problem and Its Various Formulations	1
1.1.1	Mathematical Versus Computer Science Perspectives	1
1.1.2	Different Kinds of Output	2
1.1.3	Representing Languages	3
1.1.4	Complexity Issues	5
1.1.5	Summary	6
1.2	Assessing Algorithms' Performance	7
1.2.1	Measuring Classifier Performance	7
1.2.2	McNemar's Test	8
1.2.3	5×2 Cross-Validated Paired t Test	9
1.3	Exemplary Applications	9
1.3.1	Peg Solitaire	10
1.3.2	Classification of Proteins	12
1.4	Bibliographical Background	15
	References	16
2	State Merging Algorithms	19
2.1	Preliminaries	19
2.2	Evidence Driven State Merging	21
2.3	Gold's Idea	23
2.4	Grammatical Inference with MDL Principle	27
2.4.1	The Motivation and Appropriate Measures	28
2.4.2	The Proposed Algorithm	28
2.5	Bibliographical Background	30
	References	31
3	Partition-Based Algorithms	33
3.1	Preliminaries	33
3.2	The k -tails Method	36
3.3	Grammatical Inference by Genetic Search	37
3.3.1	What Are Genetic Algorithms?	37

3.3.2	Basic Notions of the Genetic Algorithm for GI	37
3.3.3	Our Implementation	39
3.4	CFG Inference Using Tabular Representations	40
3.4.1	Basic Definitions	41
3.4.2	The Algorithm	41
3.4.3	Our Implementation	43
3.5	Bibliographical Background	45
	References	45
4	Substring-Based Algorithms	47
4.1	Error-Correcting Grammatical Inference	47
4.1.1	The GI Algorithm	47
4.1.2	Our Implementation	49
4.2	Alignment-Based Learning	50
4.2.1	Alignment Learning	51
4.2.2	Selection Learning	54
4.2.3	Our Implementation	54
4.3	Bibliographical Background	55
	References	56
5	Identification Using Mathematical Modeling	57
5.1	From DFA Identification to Graph Coloring	57
5.1.1	Encoding	57
5.1.2	Our Implementation	58
5.2	From NFA Identification to a Satisfiability Problem	61
5.2.1	Encoding	61
5.2.2	Our Implementation	62
5.3	From CFG Identification to a CSP	64
5.3.1	Encoding	64
5.3.2	Our Implementation	65
5.4	Bibliographical Background	67
	References	67
6	A Decomposition-Based Algorithm	69
6.1	Prime and Decomposable Languages	69
6.1.1	Preliminaries	69
6.1.2	Cliques and Decompositions	70
6.2	CFG Inference	71
6.2.1	The GI Algorithm	71
6.2.2	Our Implementation	72
6.3	Bibliographical Background	75
	References	75
7	An Algorithm Based on a Directed Acyclic Word Graph	77
7.1	Definitions	77
7.2	Constructing a DAWG From a Sample	78

7.3	Our Implementation	79
7.4	Bibliographical Background	81
	References.	81
8	Applications of GI Methods in Selected Fields	83
8.1	Discovery of Generating Functions.	83
8.1.1	Generating Functions	83
8.1.2	The Schützenberger Methodology.	84
8.1.3	Applications	85
8.2	Minimizing Boolean Functions.	93
8.2.1	Background and Terminology.	94
8.2.2	The Algorithm	96
8.2.3	Our Implementation	97
8.2.4	Examples	98
8.3	Use of Induced Star-Free Regular Expressions.	100
8.3.1	Definitions and an Algorithm	100
8.3.2	An Application in Classification of Amyloidogenic Hexapeptides.	103
8.3.3	An Application in the Construction of Opening Books.	105
8.4	Bibliographical Background	108
	References.	109
	Appendix A: A Quick Introduction to Python	111
	Appendix B: Python’s Tools for Automata, Networks, Genetic Algorithms, and SAT Solving.	129
	Appendix C: OML and its Usage in IronPython	139

Acronyms

CFG	Context-free grammar
CGT	Combinatorial game theory
CNF	Chomsky normal form
CNF	Conjunctive normal form
CSP	Constraint satisfaction problem
DFA	Deterministic finite automaton
DNF	Disjunctive normal form
EDSM	Evidence driven state merging
GA	Genetic algorithm
GI	Grammatical inference
GNF	Greibach normal form
ILP	Integer linear programming
LP	Linear programming
MDL	Minimum description length
MILP	Mixed integer linear programming
NFA	Non-deterministic finite automaton
NLP	Non-linear programming
NP	Non-deterministic polynomial time
OGF	Ordinary generating function
OML	Optimization modeling language
PTA	Prefix tree acceptor
RPNI	Regular positive and negative inference
SAT	Boolean satisfiability problem
TSP	Traveling salesman problem
XML	Extensible markup language