

Urszula Stańczyk  
Lakhmi C. Jain *Editors*

# Feature Selection for Data and Pattern Recognition

# **Studies in Computational Intelligence**

Volume 584

## **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 world-wide distribution, which enable both wide and rapid dissemination of research output.

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

Urszula Stańczyk · Lakhmi C. Jain  
Editors

# Feature Selection for Data and Pattern Recognition

*Editors*

Urszula Stańczyk  
Institute of Informatics  
Silesian University of Technology  
Gliwice  
Poland

Lakhmi C. Jain  
Faculty of Education, Science, Technology  
and Mathematics  
University of Canberra  
Canberra  
Australia

and

University of South Australia  
Mawson Lakes Campus  
Adelaide  
Australia

ISSN 1860-949X

ISSN 1860-9503 (electronic)

Studies in Computational Intelligence

ISBN 978-3-662-45619-4

ISBN 978-3-662-45620-0 (eBook)

DOI 10.1007/978-3-662-45620-0

Library of Congress Control Number: 2014958565

Springer Heidelberg New York Dordrecht London  
© Springer-Verlag Berlin Heidelberg 2015

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

Springer-Verlag GmbH Berlin Heidelberg is part of Springer Science+Business Media  
([www.springer.com](http://www.springer.com))

# Preface

This research book provides the reader with a selection of high-quality texts dedicated to current progress, new developments and research trends in feature selection for data and pattern recognition. In particular, this volume points to a number of advances topically subdivided into four parts:

- estimation of importance of characteristic features, their relevance, dependencies, weighting and ranking;
- rough set approach to attribute reduction with focus on relative reducts;
- construction of rules and their evaluation;
- and data- and domain-oriented methodologies.

The volume presents one introductory and 13 reviewed research papers, reflecting the work of 29 researchers from 11 countries, namely Australia, Canada, Germany, Greece, Hungary, Italy, Japan, Malaysia, Poland, Slovenia and USA.

Compilation of this book has been made possible by many people. Our sincere thanks go to the laudable efforts of many individual persons, groups and institutions that supported them in their valuable work. We wish to express our gratitude to the contributing authors and all who helped us in review procedures of the submitted manuscripts. In addition, the editors and authors of this volume extend an expression of gratitude to the members of staff at Springer, for their support in making this volume possible.

Poland, September 2014  
Australia

Urszula Stańczyk  
Lakhmi C. Jain

# Contents

<b>1 Feature Selection for Data and Pattern Recognition:</b>	
<b>An Introduction</b> . . . . .	1
Urszula Stańczyk and Lakhmi C. Jain	
1.1 Introduction . . . . .	1
1.2 Chapters of the Book . . . . .	3
1.3 Concluding Remarks . . . . .	5
References . . . . .	5

## Part I Estimation of Feature Importance

<b>2 All Relevant Feature Selection Methods and Applications</b> . . . . .	11
Witold R. Rudnicki, Mariusz Wrzesień and Wiesław Paja	
2.1 Introduction . . . . .	12
2.1.1 Definitions . . . . .	14
2.1.2 Algorithms for All-Relevant Feature Selection . . . . .	16
2.1.3 Random Forest . . . . .	18
2.2 Testing Procedure . . . . .	18
2.2.1 Data Sets . . . . .	18
2.2.2 Classification . . . . .	19
2.2.3 Feature Selection . . . . .	20
2.3 Results and Discussion . . . . .	21
2.3.1 Classification . . . . .	22
2.3.2 Feature Selection . . . . .	23
2.4 Conclusions . . . . .	27
References . . . . .	27

<b>3 Feature Evaluation by Filter, Wrapper, and Embedded Approaches</b>	29
Urszula Stańczyk	
3.1 Introduction . . . . .	29
3.2 Characteristic Features for Stylometric Analysis of Texts . . . . .	30
3.3 Approaches to Feature Selection . . . . .	31
3.3.1 Filters . . . . .	32
3.3.2 Wrappers . . . . .	32
3.3.3 Embedded Solutions . . . . .	33
3.3.4 Ranking of Features . . . . .	33
3.4 Details of Research Framework . . . . .	34
3.4.1 Input Data Sets. . . . .	34
3.4.2 Machine Learning Techniques Used in Research. . . . .	35
3.4.3 Search Parameters. . . . .	36
3.5 Feature Evaluation by Ranking . . . . .	36
3.6 Feature Evaluation by Backward Reduction . . . . .	38
3.6.1 Relief Ranking . . . . .	38
3.6.2 Embedded DRSA Ranking. . . . .	40
3.6.3 Comparison of Feature Reduction Results . . . . .	41
3.7 Conclusions . . . . .	42
References. . . . .	43
<b>4 A Geometric Approach to Feature Ranking Based Upon Results of Effective Decision Boundary Feature Matrix</b>	45
Claudia Diamantini, Alberto Gemelli and Domenico Potena	
4.1 Introduction . . . . .	45
4.2 Feature Ranking for Classification: The Background Picture. . . . .	46
4.2.1 Intrinsic Discriminant Dimension of a Classification Task . . . . .	46
4.2.2 Classical Feature Selection Strategies . . . . .	47
4.2.3 A Multiple-Challenge Case Study for Feature Ranking. . . . .	50
4.3 Focus on Feature Extraction Based Ranking. . . . .	50
4.3.1 Linear Models . . . . .	50
4.3.2 Feature Extraction Based on Decision Boundary . . . . .	51
4.4 Feature Ranking Based on Effective Decision Boundary Feature Matrix . . . . .	52
4.4.1 Geometric Considerations . . . . .	52
4.4.2 The Algorithm . . . . .	54
4.5 Experiments . . . . .	56
4.5.1 Experimental Setting . . . . .	56
4.5.2 Benchmarking the EDBFM Ranking Method . . . . .	59
4.6 Conclusions . . . . .	67
References. . . . .	67

Contents	ix
----------	----

<b>5 Weighting of Features by Sequential Selection . . . . .</b>	71
Urszula Stańczyk	
5.1 Introduction . . . . .	71
5.2 Background . . . . .	72
5.2.1 Algorithms for Feature Selection. . . . .	73
5.2.2 Connectionist Classifier . . . . .	75
5.2.3 Rule-Based Classification. . . . .	75
5.2.4 Textual Analysis. . . . .	77
5.3 Experimental Setting . . . . .	77
5.4 Sequential Forward Selection . . . . .	78
5.5 Sequential Backward Selection . . . . .	83
5.6 Concluding Remarks . . . . .	87
References. . . . .	89

## Part II Rough Set Approach to Attribute Reduction

<b>6 Dependency Analysis and Attribute Reduction in the Probabilistic Approach to Rough Sets . . . . .</b>	93
Wojciech Ziarko	
6.1 Introduction . . . . .	93
6.2 Variable Precision Rough Sets . . . . .	95
6.2.1 Set Approximations in the VPRS Approach . . . . .	96
6.2.2 Absolute Set Approximation Regions . . . . .	98
6.3 Dependencies in Approximation Spaces. . . . .	99
6.3.1 Absolute Certainty Gain . . . . .	99
6.3.2 Absolute Dependency Gain . . . . .	100
6.3.3 Average Dependency Gain. . . . .	100
6.4 Probabilistic Decision Tables . . . . .	101
6.4.1 Attributes. . . . .	101
6.4.2 Decision Tables . . . . .	102
6.4.3 Classification Tables . . . . .	103
6.5 Dependencies in Decision Tables . . . . .	104
6.5.1 Functional and Partial Functional Dependencies . . . . .	104
6.5.2 $\lambda$ -Dependency Measure . . . . .	105
6.6 $\lambda$ -Dependency-Based Reduct . . . . .	106
6.7 Probabilistic Decision Rules. . . . .	108
6.8 Significance of $\lambda$ -Reduct Attributes. . . . .	108
6.9 $\lambda$ -Core Collection of Attributes . . . . .	109
6.10 Final Remarks . . . . .	109
References. . . . .	110

<b>7 Structure-Based Attribute Reduction: A Rough Set Approach . . . . .</b>	113
Yoshifumi Kusunoki and Masahiro Inuiguchi	
7.1 Introduction . . . . .	113
7.2 Structure-Based Attribute Reduction in Rough Set Models . . . . .	115
7.2.1 Decision Tables . . . . .	115
7.2.2 Rough Set Models . . . . .	117
7.2.3 Reducts in Rough Set Models . . . . .	121
7.2.4 Boolean Functions Representing Reducts . . . . .	124
7.3 Structure-Based Attribute Reduction in Variable Precision Rough Set Models . . . . .	127
7.3.1 Rough Membership Function . . . . .	127
7.3.2 Variable Precision Rough Set Models . . . . .	128
7.3.3 Structure-Based Reducts in Variable Precision Rough Set Models . . . . .	133
7.3.4 Boolean Functions Representing Reducts . . . . .	136
7.4 Structure-Based Attribute Reduction in Dominance-Based Rough Set Models . . . . .	144
7.4.1 Decision Tables Under Dominance Principle and Dominance-Based Rough Set Models . . . . .	144
7.4.2 Structure-Based Reducts in Dominance-Based Rough Set Models . . . . .	150
7.4.3 Boolean Functions Representing Reducts . . . . .	154
7.5 Concluding Remarks . . . . .	157
References . . . . .	158

### Part III Rule Discovery and Evaluation

<b>8 A Comparison of Rule Induction Using Feature Selection and the LEM2 Algorithm . . . . .</b>	163
Jerzy W. Grzymała-Busse	
8.1 Introduction . . . . .	163
8.2 Rule Induction Based on Feature Selection . . . . .	165
8.3 LEM2 . . . . .	166
8.4 Inconsistent Data . . . . .	168
8.5 LERS Classification System . . . . .	171
8.6 Experiments . . . . .	171
8.7 Conclusions . . . . .	175
References . . . . .	175

<b>9</b>	<b>Meta-actions as a Tool for Action Rules Evaluation . . . . .</b>	177
	Hakim Touati, Zbigniew W. Raś and James Studnicki	
9.1	Introduction . . . . .	178
9.2	Decision Systems . . . . .	179
9.3	Action-Rules . . . . .	179
9.3.1	Action Rules Evaluation . . . . .	180
9.4	Meta-actions. . . . .	181
9.4.1	Meta-actions Influence Matrix . . . . .	182
9.4.2	Mining Meta-actions Effects. . . . .	183
9.4.3	Meta-action Evaluation . . . . .	185
9.5	Meta-actions Versus Action Rules. . . . .	186
9.5.1	Action Rules Selection by Meta-actions. . . . .	188
9.6	Side Effects . . . . .	190
9.6.1	Meta-actions Side Effects. . . . .	190
9.6.2	Action Rules Side Effects . . . . .	190
9.7	Experiments . . . . .	191
9.7.1	Dataset Description . . . . .	191
9.7.2	Meta-action Extraction. . . . .	192
9.7.3	Action Rules Extraction and Evaluation. . . . .	195
9.8	Conclusion. . . . .	196
	References. . . . .	196
<b>10</b>	<b>Irrelevant Feature and Rule Removal for Structural Associative Classification Using Structure-Preserving Flat Representation. . . . .</b>	199
	Izwan Nizal Mohd Shaharanee and Fedja Hadzic	
10.1	Introduction . . . . .	200
10.2	Related Work . . . . .	201
10.2.1	Relationship Between Feature Subset Selection and Frequent Subtree Interestingness . . . . .	203
10.3	Problem Background. . . . .	204
10.3.1	Feature Subset Selection . . . . .	205
10.3.2	Modeling Tree-Structured Data. . . . .	207
10.3.3	Database Structure Model (DSM) . . . . .	209
10.3.4	Tree to Flat Conversion Example Using Academic Institution WebLogs Data . . . . .	211
10.3.5	Representing Disconnected Trees w.r.t. DSM . . . . .	212
10.4	Method and Experimental Setup . . . . .	214
10.5	Experimental Evaluation . . . . .	218
10.5.1	Experiment Set 1—CRM Data . . . . .	219
10.5.2	Experiment Set 2—CSLOGS Data . . . . .	221

10.5.3	Experiment Set 3—Academic Institution Web Log Data . . . . .	223
10.6	Conclusion and Future Work . . . . .	225
	References . . . . .	226
<b>Part IV Data- and Domain-Oriented Methodologies</b>		
<b>11</b>	<b>Hubness-Aware Classification, Instance Selection and Feature Construction: Survey and Extensions to Time-Series . . . . .</b>	231
	Nenad Tomašev, Krisztian Buza, Kristóf Marussy and Piroska B. Kis	
11.1	Introduction . . . . .	232
11.2	Problem Formulation and Basic Notations . . . . .	233
11.3	Dynamic Time Warping . . . . .	234
11.4	Hubs in Time-Series Data . . . . .	238
11.5	Hubness-Aware Classification of Time-Series . . . . .	240
11.5.1	hw- $k$ NN: Hubness-Aware Weighting . . . . .	241
11.5.2	h-FNN: Hubness-Based Fuzzy Nearest Neighbor . . . . .	243
11.5.3	NHBNN: Naive Hubness Bayesian $k$ -Nearest Neighbor . . . . .	244
11.5.4	HIKNN: Hubness Information $k$ -Nearest Neighbor . . . . .	246
11.5.5	Experimental Evaluation of Hubness-Aware Classifiers . . . . .	247
11.6	Instance Selection and Feature Construction for Time-Series Classification . . . . .	254
11.6.1	Instance Selection for Speeding-Up Time-Series Classification . . . . .	254
11.6.2	Feature Construction . . . . .	256
11.7	Conclusions and Outlook . . . . .	258
	References . . . . .	259
<b>12</b>	<b>Selection of Visual Descriptors for the Purpose of Multi-camera Object Re-identification . . . . .</b>	263
	Piotr Dalka, Damian Ellwart, Grzegorz Szwoch, Karol Lisowski, Piotr Szczuko and Andrzej Czyżewski	
12.1	Introduction . . . . .	263
12.2	Video Preprocessing . . . . .	266
12.3	Multi-camera Object Tracking . . . . .	267
12.4	Visual Object Descriptors . . . . .	268
12.4.1	Colour Histogram . . . . .	269
12.4.2	Vertical Trace . . . . .	271

12.4.3	Moment Invariants . . . . .	272
12.4.4	Colour Layout Descriptors . . . . .	272
12.4.5	Co-occurrence Matrices Statistical Parameters. . . . .	273
12.4.6	Edge Histogram Descriptor . . . . .	275
12.4.7	Local Binary Pattern . . . . .	276
12.4.8	Local Image Features . . . . .	277
12.5	Descriptor Evaluation Methods. . . . .	278
12.5.1	RS Index . . . . .	278
12.5.2	SD Index . . . . .	279
12.5.3	Dissimilarity Measure . . . . .	280
12.5.4	Result Aggregation . . . . .	281
12.6	Object Identification . . . . .	283
12.7	Experiments and Results . . . . .	288
12.7.1	Datasets. . . . .	288
12.7.2	Descriptor Evaluation . . . . .	289
12.7.3	Object Identification . . . . .	296
12.8	Conclusions . . . . .	300
	References. . . . .	301
<b>13</b>	<b>Improving the Recognition Performance of Moment Features by Selection</b> . . . . .	305
	George A. Papakostas	
13.1	Introduction . . . . .	305
13.2	Image Moment Features . . . . .	306
13.2.1	Continuous Orthogonal Moments . . . . .	307
13.2.2	Discrete Orthogonal Moments . . . . .	311
13.2.3	Image Reconstruction by the Method of Moments . . . . .	314
13.2.4	Moments Interpretation . . . . .	315
13.3	Is There a Need for Moments' Selection? . . . . .	318
13.4	Moment Features Selection . . . . .	319
13.4.1	GA-Based Selection . . . . .	320
13.4.2	Relief Algorithm. . . . .	320
13.5	Experimental Study. . . . .	321
13.5.1	Benchmark Datasets . . . . .	321
13.5.2	Datasets Pre-processing . . . . .	321
13.5.3	Simulations . . . . .	323
13.6	Conclusions . . . . .	325
	References. . . . .	326

<b>14 Signature Selection for Grouped Features with a Case Study on Exon Microarrays . . . . .</b>	329
Sangkyun Lee	
14.1 Introduction . . . . .	329
14.1.1 Regularized Regression . . . . .	330
14.2 Regularized Regression Methods for Grouped Features . . . . .	332
14.2.1 Group Lasso . . . . .	333
14.2.2 Overlapping Group Lasso . . . . .	336
14.2.3 Sparse Group Lasso . . . . .	338
14.3 A Case Study on Exon Microarray Data . . . . .	340
14.3.1 Data . . . . .	341
14.3.2 Algorithms for Comparison . . . . .	342
14.3.3 Comparison of Performance . . . . .	343
14.4 Discussion . . . . .	347
14.5 Conclusion . . . . .	347
References . . . . .	348
<b>Index . . . . .</b>	351

# Editors and Contributors

## About the Editors



**Urszula Stańczyk** received the M.Sc. degree in Computer Science, and Ph.D. degree (with honours) in Technical Sciences with specialisation in Informatics from the Silesian University of Technology (SUT), Gliwice, Poland. She is with the Institute of Informatics, SUT. From 2003 to 2010 Editor-in-Chief of the “Activity Report” for the Institute, Dr. Stańczyk is a member of KES International ([www.kesinternational.org](http://www.kesinternational.org)), ADAA Group (<http://adaa.polsl.pl/>), and MIR Labs (<http://www.mirlabs.org/>), a member of Program Committees for several scientific conferences, and one of the key persons responsible for establishing a series

of *International Conferences on Man-Machine Interactions* (ICMMI). She is a member of the Editorial Board of Intelligent Decision Technologies: An International Journal (<http://www.iospress.nl/journal/intelligent-decision-technologies>).

Her research interests include artificial intelligence, pattern recognition and classification, neural and rough processing, feature extraction and selection, induction of decision rules, rule quality measures, stylometric processing of texts, data mining. She co-edited conference proceedings, authored and co-authored a two-volume monograph on synthesis and analysis of logic circuits, academic textbooks on arithmetic of digital systems, book chapters, conference papers and journal articles focused on applications of computational intelligence techniques to stylometry.



**Dr. Lakhmi C. Jain** is with the Faculty of Education, Science, Technology and Mathematics at the University of Canberra, Australia and University of South Australia, Australia. He is a Fellow of the Institution of Engineers Australia.

Dr. Jain founded the KES International for providing a professional community the opportunities for publications, knowledge exchange, cooperation and teaming. Involving around 5,000 researchers drawn from universities and companies worldwide, KES facilitates international cooperation and generate synergy in teaching and research. KES regularly provides networking opportunities for professional community through one of the largest conferences of its kind in the area of KES ([www.kesinternational.org](http://www.kesinternational.org)).

His interests focus on the artificial intelligence paradigms and their applications in complex systems, security, e-education, e-healthcare, unmanned air vehicles and intelligent agents.

## Contributors

**Krisztian Buza** Faculty of Mathematics, Informatics and Mechanics, University of Warsaw (MIMUW), Warsaw, Poland

**Andrzej Czyżewski** Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Gdańsk, Poland

**Piotr Dalka** Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Gdańsk, Poland

**Claudia Diamantini** Dipartimento di Ingegneria Dell'Informazione, Università Politecnica Delle Marche, Ancona, Italy

**Damian Ellwart** Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Gdańsk, Poland

**Alberto Gemelli** Dipartimento di Ingegneria Dell'Informazione, Università Politecnica Delle Marche, Ancona, Italy

**Jerzy W. Grzymała-Busse** Department of Electrical Engineering and Computer Science, University of Kansas, Lawrence, USA; Department of Expert Systems and Artificial Intelligence, University of Information Technology and Management, Rzeszów, Poland

**Fedja Hadzic** Department of Computing, Curtin University, Perth, Australia

**Masahiro Inuiguchi** Graduate School of Engineering Sciences, Osaka University, Toyonaka, Osaka, Japan

**Lakhmi C. Jain** Faculty of Education, Science, Technology and Mathematics, University of Canberra, Canberra, ACT, Australia; School of Electrical and Information Engineering, University of South Australia, Adelaide, Australia

**Piroska B. Kis** Department of Mathematics and Computer Science, College of Dunaújváros, Dunaújváros, Hungary

**Yoshifumi Kusunoki** Graduate School of Engineering, Osaka University, Suita, Osaka, Japan

**Sangkyun Lee** Technische Universität Dortmund, Dortmund, Germany

**Karol Lisowski** Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Gdańsk, Poland

**Kristóf Marussy** Department of Computer Science and Information Theory, Budapest University of Technology and Economics, Budapest, Hungary

**Wiesław Paja** Faculty of Applied IT, University of Information Technology and Management, Rzeszów, Poland

**George A. Papakostas** Human Machines Interaction (HMI) Laboratory, Department of Computer and Informatics Engineering, Eastern Macedonia and Thrace (EMT) Institute of Technology, Kavala, Greece

**Domenico Potena** Dipartimento di Ingegneria Dell'Informazione, Università Politecnica Delle Marche, Ancona, Italy

**Zbigniew W. Raś** The University of North Carolina at Charlotte, Charlotte, NC, USA; Warsaw University of Technology, Warsaw, Poland

**Witold R. Rudnicki** Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw, Warsaw, Poland

**Izwan Nizal Mohd Shaharanee** School of Quantitative Sciences, Universiti Utara Malaysia, Sintok, Malaysia

**Urszula Stańczyk** Institute of Informatics, Silesian University of Technology, Gliwice, Poland

**James Studnicki** College of Public Health, The University of North Carolina at Charlotte, Charlotte, NC, USA

**Piotr Szczuko** Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Gdańsk, Poland

**Grzegorz Szwoch** Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Gdańsk, Poland

**Nenad Tomašev** Institute Jožef Stefan, Artificial Intelligence Laboratory, Ljubljana, Slovenia

**Hakim Touati** College of Computing and Informatics, The University of North Carolina at Charlotte, Charlotte, NC, USA

**Mariusz Wrzesień** Faculty of Applied IT, University of Information Technology and Management, Rzeszów, Poland

**Wojciech Ziarko** Department of Computer Science, University of Regina, Regina, Canada

# Chapter 1

## Feature Selection for Data and Pattern Recognition: An Introduction

Urszula Stańczyk and Lakhmi C. Jain

**Abstract** Surrounded by data and information in various forms we need to characterise and describe objects of our universe using some attributes of nominal or numerical type. Selection of features can be performed basing on domain knowledge, executed through dedicated approaches, driven by some particular inherent properties of methodologies and techniques employed, or governed by other factors or rules. This chapter presents a general and brief introduction to topics of feature selection for data and pattern recognition. Its main aim is to provide short descriptions of the chapters included in this volume.

**Keywords** Feature · Feature selection · Pattern recognition · Data mining

### 1.1 Introduction

Some say that our earliest memories form when, as children, we learn to describe the world we live in, and express verbally what we feel and think, how we perceive other people, objects, events, abstract concepts. While we grow older, we learn to detect and recognise patterns [20], and our discriminating skills grow as well. We develop associations, preferences and dislikes, which are employed, consciously and subconsciously, when choices are made, actions taken.

Imagine opening an unknown thick book and finding in it a whole page dedicated to a line of thought of some character, jumping from one topic to another, along with connecting ideas, feelings and memories, digressions. Without looking at the cover or the title page, by similarity to a stream of consciousness, one instantly thinks

---

U. Stańczyk (✉)

Institute of Informatics, Silesian University of Technology,  
Akademicka 16, 44-100 Gliwice, Poland  
e-mail: urszula.stanczyk@polsl.pl

L.C. Jain

Faculty of Education, Science, Technology and Mathematics,  
University of Canberra, Canberra, ACT 2601, Australia  
e-mail: lakhmi.jain@unisa.edu.au

about James Joyce as the author. A painting with a group of posed ballet dancers upon a stage we would associate with Degas, and water lilies in a pond with Monet. Hearing rich classical organ music we could try to guess Bach as the composer. In each of these exemplary cases we have a chance of correct recognition basing on some characteristic features the authors are famous for. Our brains recognise lily flowers or organ tunes, yet to make other people or machines capable of the same we need to explain these specific elements, which means describing, expressing them in understandable and precise terms.

Characterisation of things is a natural element of life, some excel at it while others are not so good. Yet anybody can make basic distinctions, especially with some support system. Some of how these characteristics play into problems we need to tackle, tasks waiting to be solved, comes intuitively, some we get from observations or experiments, drawn conclusions. Some pointers are rather straightforward while others indirect or convoluted.

According to a dictionary definition a *feature* is a distinctive attribute or aspect of something and it is used as a synonym for characteristic, quality, or property [29, 38]. With such meaning it is employed in general language descriptions but also in more confined areas of technical sciences, computer technologies, in particular in the domain of data mining and pattern recognition [24, 30, 39].

For automatic recognition and classification [11, 27] all objects of the universe of discourse need to be perceived through information carried by their characteristics and in cases when this information is incomplete or uncertain the resulting predictive accuracies of constructed systems, whether they induce knowledge from available data in supervised or unsupervised manner [28], relying on statistics-oriented calculations [8, 19] or heuristic algorithms, could be unsatisfactory or falsified, making observations and conclusions unreliable.

The performance of any inducer depends on the raw input data on which inferred knowledge is based [21], exploited attributes, the approach or methodology of data mining applied, but also on the general dimensionality of the problem [40]. Contemporary computer technologies with their high computational capabilities aid in processing, but still for huge data sets, and very high numbers of variables the process, even if feasible, can take a lot of time and effort, require unnecessary or impractically large storage.

Typically the primary goal is to achieve the maximal classification accuracy but we need to take into account practical aspects of obtained solutions, and consider compromises with trade-offs such as some loss in performance for much shortened time, less processing, lower complexity, or smaller structure of the system.

Feature selection is an explicit part of most knowledge mining approaches—some attributes are chosen over others while forming a set of characteristic features in the first place [10, 18]. Here the choice can be supported by expert knowledge. Once some subset of variables is available, using it to construct a rule classifier, a rule induction algorithm leads to particular choices of conditions for all constituent rules, either usual or inhibitory. In a similar manner in a decision tree construction specific attributes are to be checked at its nodes, and artificial neural networks through their

learning rule establish the degrees of importance or relevance of features. Such examples can be multiplied.

Even for working solutions it is worthwhile to study attributes as it is not out of realm of possibility that some of them are excessive or repetitive, even irrelevant, or there exist other alternatives of the same merit, and once such variables are discovered, different selection can improve the performance, if not with respect to the classification accuracy, then by better understanding of analysed concepts, possibly more explicit presentation of information [23].

With all these factors and avenues to explore it is not surprising that the problem of feature selection, with various meanings of this expression, is actively pursued in research, which has given us the motivation for dedicating this book to this area.

## 1.2 Chapters of the Book

The 13 chapters included in this volume are grouped into four parts. What follows is a short description of the content for each chapter.

### Part I Estimation of Feature Importance

Chapter 2 is devoted to a review of the field of all-relevant feature selection, and presentation of the representative algorithm [5, 25]. The problem of all-relevant feature selection is first defined, then key algorithms are described. Finally the Boruta algorithm is explained in a greater detail and applied both to a collection of synthetic and real-world data sets, with comments on performance, properties and parameters.

Chapter 3 illustrates the three approaches to feature selection and reduction [17]: filters, wrappers, and embedded solutions [25], combined for the purpose of feature evaluation. These approaches are used when domain knowledge is unavailable or insufficient for an informed choice, or in order to support this expert knowledge to achieve higher efficiency, enhanced classification, or reduced sizes of classifiers. The classification task under study is that of authorship attribution with balanced data.

Chapter 4 presents a method of feature ranking that calculates the relative weight of features in their original domain with an algorithmic procedure [3]. The method supports information selection of real world features and is useful when the number of features has costs implications. It has at its core a feature extraction technique based on effective decision boundary feature matrix, which is extended to calculate the total weight of the real features through a procedure geometrically justified [28]. Chapter 5 focuses on weighting of characteristic features by the processes of their sequential selection. A set of all accessible attributes can be reduced backwards, or variables examined one by one can be selected forward. The choice can be conditioned by the performance of a classification system, in a wrapper model, and the observations with respect to selected variables can result in assignment

of weights. The procedures are employed for rule [37] and connectionist [26] classifiers, applied in the task of authorship attribution.

## Part II Rough Set Approach to Attribute Reduction

Chapter 6 discusses two probabilistic approaches [44] to rough sets: the variable precision rough set model [43] and the Bayesian rough set model, as they apply to data dependencies detection, analysis and their representation. The focus is on the analysis of data co-occurrence-based dependencies appearing in classification tables and probabilistic decision tables acquired from data. In particular, the notion of attribute reduct, in the framework of probabilistic approach, is of interest in the chapter and it includes two efficient reduct computation algorithms.

Chapter 7 provides an introduction to a rough set approach to attribute reduction [1], treated as removing condition attributes with preserving some part of the lower/upper approximations of the decision classes, because the approximations summarize the classification ability of the condition attributes [42]. Several types of reducts according to structures of the approximations are presented, called “structure-based” reducts. Definitions and theoretical results for structures-based attribute reduction are given [33, 36].

## Part III Rule Discovery and Evaluation

Chapter 8 compares a strategy of rule induction based on feature selection [32], exemplified by the LEM1 algorithm, with another strategy, not using feature selection, exemplified by the LEM2 algorithm [15, 16]. The LEM2 algorithm uses all possible attribute-value pairs as the search space. It is shown that LEM2 significantly outperforms LEM1, a strategy based on feature selection in terms of an error rate. The LEM2 algorithm induces smaller rule sets with the smaller total number of conditions as well. The time complexity for both algorithms is the same [31]. Chapter 9 addresses action rules extraction. Action rules present users with a set of actionable tasks to follow to achieve a desired result. The rules are evaluated using their supporting patterns occurrence and their confidence [41]. These measures fail to measure the feature values transition correlation and applicability, hence meta-actions are used in evaluating action rules, which is presented in terms of likelihood and execution confidence [14]. Also an evaluation model of the application of meta-actions based on cost and satisfaction is given.

Chapter 10 explores the use of a feature subset selection measure, along with a number of common statistical interestingness measures, via structure-preserving flat representation for tree-structured data [34, 35]. A feature subset selection is used prior to association rule generation. Once the initial set of rules is obtained, irrelevant rules are determined as those that are comprised of attributes not determined to be statistically significant for the classification task [22].

## Part IV Data- and Domain-Oriented Methodologies

Chapter 11 gives a survey of hubness-aware classification methods and instance selection. The presence of hubs, the instances similar to exceptionally large number

of other instances, has been shown to be one of the crucial properties of time-series data sets [4, 7]. There are proposed some selected instances for feature construction, detailed description of the algorithms provided, and experimental results on large number of publicly available real-world time-series data sets shown.

Chapter 12 presents an analysis of descriptors that utilize various aspects of image data: colour, texture, gradient, and statistical moments, and this list is extended with local features [2]. The goal of the analysis is to find descriptors that are best suited for particular task, i.e. re-identification of objects in a multi-camera environment. For descriptor evaluation, scatter and clustering measures [12] are supplemented with a new measure derived from calculating direct dissimilarities between pairs of images [5, 6].

Chapter 13 deals with the selection of the most appropriate moment features used to recognise known patterns [13]. For this purpose, some popular moment families are presented and their properties are discussed. Two algorithms, a simple Genetic Algorithm (GA) and the Relief algorithm are applied to select the moment features that better discriminate human faces and facial expressions, under several pose and illumination conditions [9].

Chapter 14 contains considerations on grouped features. When features are grouped, it is desirable to perform feature selection groupwise in addition to selecting individual features. It is typically the case in data obtained by modern high-throughput genomic profiling technologies such as exon microarrays. To handle grouped features, feature selection methods are discussed with the focus on a popular shrinkage method, lasso, and its variants, that are based on regularized regression with generalized linear models [6].

### 1.3 Concluding Remarks

In this book some advances and research dedicated to feature selection for data and pattern recognition are presented. Even though it has been the subject of interest for some time, feature selection remains one of actively pursued avenues of investigations due to its importance and bearing upon other problems and tasks. It can be studied within a domain from which features are extracted, independently of it, taking into account specific properties of involved algorithms and techniques, with feedback from applications, or without it. Observations from executed experiments can bring local and global conclusions, with theoretical and practical significance.

## References

1. Abraham, A., Falcón, R., Bello, R. (eds.): Rough Set Theory: A True Landmark in Data Analysis. Studies in Computational Intelligence, vol. 174. Springer, Berlin (2009)
2. Baxes, G.A.: Digital Image Processing: Principles and Applications. Wiley, New York (1994)

3. Blum, A., Langley, P.: Selection of relevant features and examples in machine learning. *Artif. Intell.* **97**, 245–271 (1997)
4. Botsch, M.: Machine Learning Techniques for Time Series Classification. Cuvillier, San Francisco (2009)
5. Breiman, L.: Random forests. *Mach. Learn.* **45**, 5–32 (2001)
6. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth & Brooks, Monterey (1984)
7. Carbonell, J.G. (ed.): Machine Learning. Paradigms and Methods. MIT Press, Boston (1990)
8. Chao, L.L.: Introduction to Statistics. Brooks Cole Publishing Co., Monterey (1980)
9. Cipolla, R., Pentland, A.: Computer Vision for Human-Machine Interaction. Cambridge University Press, Cambridge (1998)
10. Dash, M., Liu, H.: Feature selection for classification. *Intell. Data Anal.* **1**, 131–156 (1997)
11. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis. Wiley, New York (1973)
12. Everitt, B.: Cluster Analysis. Heinemann Educational Books, London (1980)
13. Flusser, J., Zitova, B., Suk, T.: Moments and Moment Invariants in Pattern Recognition. Wiley, New York (2009)
14. Fuernkranz, J., Gamberger, D., Lavrac, N.: Foundations of Rule Learning. Springer, Berlin (2012)
15. Grzymala-Busse, J.W.: Knowledge acquisition under uncertainty—a rough set approach. *J. Intell. Robot. Syst.* **1**, 3–16 (1988)
16. Grzymala-Busse, J.W.: Data with missing attribute values: generalization of indiscernibility relation and rule induction. *Trans. Rough Sets* **1**, 78–95 (2004)
17. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. *J. Mach. Learn. Res.* **3**, 1157–1182 (2003)
18. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A.: Feature Extraction. Foundations and Applications. Springer, Berlin (2006)
19. Hamburg, M.: Statistical Analysis for Decision Making. Harcourt Brace Jovanovich Inc., New York (1983)
20. Hofstadter, D.: Metamagical Themas: Questing for the Essence of Mind and Pattern. Basic Books Inc., New York (1985)
21. Holland, J.H., Holyoak, K.J., Nisbett, R.E.: Induction: Processes of Inference, Learning, and Discovery. MIT Press, Boston (1986)
22. Hollander, M., Wolfe, D.A.: Nonparametric Statistical Methods, 2nd edn. Wiley, New York (1999)
23. Jensen, R., Shen, Q.: Computational Intelligence and Feature Selection. Wiley, Hoboken (2008)
24. Kloesgen, W., Zytkow, J. (eds.): Handbook of Data Mining and Knowledge Discovery. Oxford University Press, New York (2002)
25. Kohavi, R., John, G.: Wrappers for feature selection. *Artif. Intell.* **97**, 273–324 (1997)
26. Krawiec, K., Stefanowski, J.: Machine Learning and Neural Nets. Poznan University of Technology Press, Poznan (2003)
27. Kuncheva, L.I.: Combining Pattern Classifiers. Methods and Algorithms. Wiley, Hoboken (2004)
28. Liu, H., Motoda, H.: Computational Methods of Feature Selection. Chapman and Hall/ CRC, Boca Raton (2007)
29. Longman Dictionary of Contemporary English, 6th revised edn. Pearson Longman, London (2014)
30. Maimon, O., Rokach, L. (eds.): Data Mining and Knowledge Discovery Handbook, 2nd edn. Springer, Berlin (2010)
31. Meyers, R.A. (ed.): Encyclopedia of Complexity and Systems Science. Springer, Berlin (2009)
32. Pawlak, Z.: Rough Sets. Theoretical Aspects of Reasoning About Data. Kluwer Academic Publishers, Boston (1991)
33. Polkowski, L., Skowron, A. (eds.): Rough Sets in Knowledge Discovery 1: Methodology and Applications. Physica-Verlag, Heidelberg (1998)

34. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo (1993)
35. Roiger, R.J., Geatz, M.W.: Data Mining. A Tutorial-Based Primer. Addison-Wesley, Boston (2003)
36. Slowinski, R.: Intelligent Decision Support. Handbook of Applications and Advances of the Rough Sets Theory. Kluwer Academic Publishers, Boston (1992)
37. Stefanowski, J.: Algorithms of Decision Rule Induction in Data Mining. Poznan University of Technology Press, Poznan (2001)
38. The Merriam-Webster Dictionary. <http://www.merriam-webster.com/>
39. Wang, J. (ed.): Data Mining: Opportunities and Challenges. Idea Group Publishing, Hershey (2003)
40. Weiss, S.M., Indurkhy, N.: Predictive Data Mining. A Practical Guide. Morgan Kaufmann Publication, San Francisco (1998)
41. Witten, I., Frank, E.: Data Mining. Practical Machine Learning Tools and Techniques. Elsevier, Amsterdam (2005)
42. Zanakis, H., Doukidis, G., Zopounidis, Z. (eds.): Decision Making: Recent Developments and Worldwide Applications. Kluwer Academic Publishers, Boston (2000)
43. Ziarko, W.: Variable precision rough set model. *J. Comput. Syst. Sci.* **46**(1), 39–59 (1993)
44. Ziarko, W.: Probabilistic approach to rough sets. *Int. J. Approx. Reason.* **49**, 272–284 (2008)

# **Part I**

## **Estimation of Feature Importance**

# **Chapter 2**

# **All Relevant Feature Selection Methods and Applications**

**Witold R. Rudnicki, Mariusz Wrzesień and Wiesław Paja**

**Abstract** All-relevant feature selection is a relatively new sub-field in the domain of feature selection. The chapter is devoted to a short review of the field and presentation of the representative algorithm. The problem of all-relevant feature selection is first defined, then key algorithms are described. Finally the Boruta algorithm, under development at ICM, University of Warsaw, is explained in a greater detail and applied both to a collection of synthetic and real-world data sets. It is shown that algorithm is both sensitive and selective. The level of falsely discovered relevant variables is low—on average less than one falsely relevant variable is discovered for each set. The sensitivity of the algorithm is nearly 100 % for data sets for which classification is easy, but may be smaller for data sets for which classification is difficult, nevertheless, it is possible to increase the sensitivity of the algorithm at the cost of increased computational effort without adversely affecting the false discovery level. It is achieved by increasing the number of trees in the random forest algorithm that delivers the importance estimate in Boruta.

**Keywords** All-relevant feature selection · Strong and weak relevance · Feature importance · Boruta · Random forest

---

W.R. Rudnicki (✉)

Interdisciplinary Centre for Mathematical and Computational Modelling,  
University of Warsaw, Pawiańskiego 5A, 02-106 Warsaw, Poland  
e-mail: W.Rudnicki@icm.edu.pl

M. Wrzesień · W. Paja

Faculty of Applied IT, University of Information Technology and Management,  
Sucharskiego 2, 35-225 Rzeszów, Poland  
e-mail: mwrzesien@wsiz.rzeszow.pl

W. Paja

e-mail: wpaja@wsiz.rzeszow.pl

## 2.1 Introduction

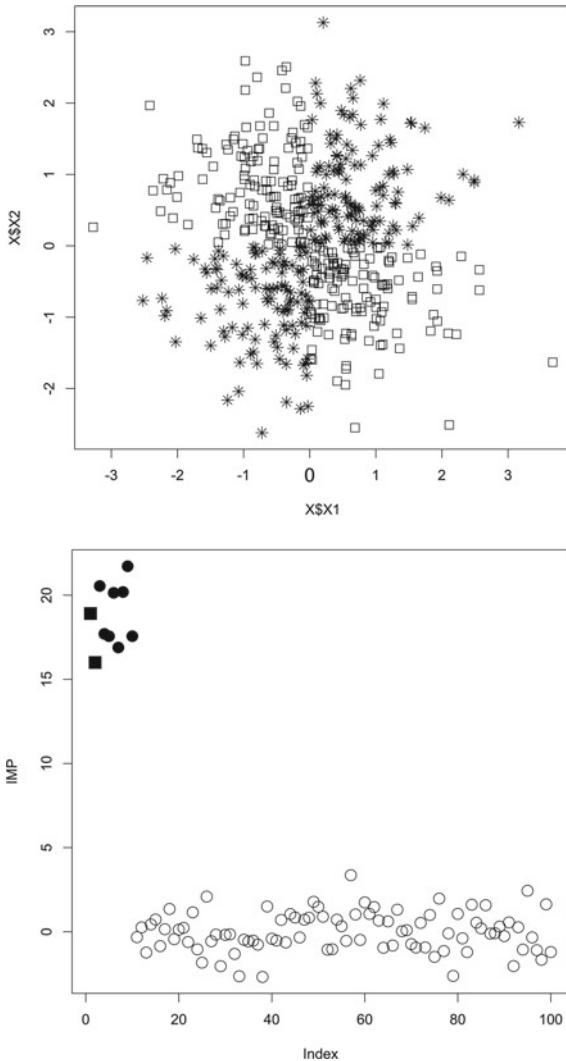
The usual goal of feature selection in machine learning is to find the best set of features that allows one to build useful models of studied phenomena. The chapter is devoted to a different application of feature selection process, where building a machine learning model is merely a tool for extracting all features that are relevant for a problem. The relevance is considered in a broad sense—it is sufficient for a feature to be declared relevant, when it is useful for building a machine learning model of the problem under scrutiny at some context. One may ask why this goal is *relevant* at all? Why should anyone be interested in this type of relevance?

Let us firstly describe a toy problem that illustrates a need for the all-relevant feature selection in an artificially transparent setting. Let us construct a system containing 100 objects described with one hundred real-valued variables  $X_1, \dots, X_{100}$ , and one binary decision variable  $D$ . The descriptive variables  $X_1$  and  $X_2$  are drawn from a normal distribution  $N(0, 1)$ . The value of the decision variable is determined from values of these variables in the following manner. It is one (**TRUE**) if both variables have the same sign and is zero (**FALSE**) if their signs differ. The descriptive variables  $X_3, \dots, X_{10}$  are obtained as a linear combination of  $X_1$  and  $X_2$ , and normalised to  $N(0, 1)$ . The variables  $X_{11}, \dots, X_{100}$ , are drawn from a normal distribution  $N(0, 1)$ . Finally the indexes of the variables are randomly permuted. The goal of the researcher is to determine which variables are responsible for the value of a decision variable.

There is a very easy path to the solution of this problem. One could take a classifier that is able to rank feature importance and select two most important features. Unfortunately this path may lead us astray, as displayed in Fig. 2.1, that shows the ranking of feature importance for our toy problem returned by a random forest (RF) [2] classifier. Here, for clarity, variable indexes are not permuted.

The toy problem is simple enough that it can be solved directly by a brute force approach. It is sufficient to build 4,950 models including two variables to find one that gives perfect classification and hence is the most likely to be built on two variables used to generate the model. However, for real life problems a number of descriptive variables may be much larger, connections between these variables and decision may be more complicated, measurements are subject to noise. Moreover, one does not know beforehand how many variables influence decision. Finally, while for our toy problem the model based on two variables used to generate the model usually gives best results, this is not guaranteed to work in a general case. Hence the brute force approach will not work in most cases.

As an example of a real-life application we may consider deciphering connection between gene expression levels in humans with some medical condition. In this case a number of variables is roughly twenty thousands, it is not known how many genes are involved and how, and last but not least—measurements are subject both to normal variability and experimental error. Analysis of such problem can be split into two separate tasks: determination which variables are connected in some way with the decision variable, and then identification of those variables that are responsible for



**Fig. 2.1** The illustration for the toy problem. The *upper panel* shows projection of the system on the plane ( $X_1X_2$ ). The *lower panel* shows the importance of variables in random forest classifier. *Solid squares* correspond to variables used to generate the decision variable, *solid circles* correspond to combinations of  $X_1$  and  $X_2$  and *open circles* correspond to random variables. It is clear that importance of variables obtained from random forest can be used to discern informative and non-informative features. Nevertheless, the importance ranking does not allow to detect the variables used for generation of the decision variable

a value of the decision variable. The first task can be tackled using the all-relevant feature selection approach. The solution of the second task, which generally is much

harder, should be easier when all relevant variables are identified and hence the number of variables is reduced.

For example in our toy problem a perfect algorithm for all-relevant feature selection should find that 10 variables out of 1,000 are somehow connected with decision variable, therefore the number of models tested in the brute force approach can be reduced to 45. In a medical problem of a researcher studying a connection between gene expression and a medical condition, a number of genes to consider may be reduced from multiple thousands to hundreds or tens, or maybe even a handful of variables. A domain specific knowledge can be then applied to build a model of a problem under scrutiny.

### 2.1.1 Definitions

Up to this point the notion of relevance was used without definition, instead we relied on its intuitive understanding. However, it has been already observed by Kohavi and John [7] that there are several definitions of relevance that may be contradictory and misleading. They proposed that two degrees of relevance (strong and weak) are required to encompass all notions that are usually associated with this term. In their approach the relevance is defined in the absolute terms, with the help of an ideal Bayes classifier.

**Definition 1** A feature  $X$  is *strongly relevant* when removal of  $X$  alone from the data always results in deterioration of the prediction accuracy of the ideal Bayes classifier.

**Definition 2** A feature  $X$  is *weakly relevant* if it is not strongly relevant and there exists a subset of features  $S$ , such that the performance of ideal Bayes classifier on  $S$  is worse than the performance on  $S \cup \{X\}$ .

**Definition 3** A feature  $X$  is *irrelevant* if it is neither strongly nor weakly relevant.

One should note, that an information system might be constructed in such a way, that there are no strongly relevant attributes. Indeed, it is easy to notice that the toy system described above does not contain strongly relevant attributes.

Another useful notions were introduced by Nilson et al. [13], who used concepts of weakly and strongly relevant features to define formally two problems of feature selection. A *minimal optimal problem* in feature selection has the goal to find the minimal set of attributes giving the best possible classifier. The other is an *all relevant problem*, where one is interested in finding all strongly and weakly relevant attributes.

**Definition 4** (*Minimal optimal problem*) Find a set of attributes consisting of all strongly relevant attributes and such subset of weakly relevant attributes, that all remaining weakly relevant attributes contain only redundant information.

**Definition 5** (*All-relevant problem*) Find all strongly relevant and all weakly relevant attributes.

It has been shown by Nilsson and co-workers, that exact solution of the all relevant problem requires an exhaustive search, which is intractable for all but smallest systems.

The relevance defined earlier is a qualitative notion—a feature can either be relevant or irrelevant. It is also an objective property of the system under scrutiny, independent from the classifier used for building a model. This notion is distinct from *importance of variable*, that is a quantitative and classifier-dependent measure of the contribution of a variable to a model of the system. One can use various measures of importance of variable, provided that they satisfy the simple condition—the importance of relevant variables should be higher than importance of irrelevant ones. A useful and intuitive measure of importance was introduced by Breiman in random forest (RF) classification algorithm [2].

**Definition 6** (*Importance of a variable*) is the loss of the classification accuracy of the model that was built using this variable, when the information on the variable's value is withdrawn.

A final concept that will be used often enough in the current chapter to deserve a mention in this section is a *contrast variable*.

**Definition 7** (*Contrast variable*) is such descriptive variable that does not carry information on the decision variable by design.

It is added to the system in order to discern relevant and irrelevant variables. It may be obtained by drawing from theoretically justified probability distribution e.g. normal or uniform; it may be also obtained from real variables by random permutation of their values between objects. Application of contrast variables for feature selection was first proposed by Stoppiglia et al. [15] and then independently by Tuv et al. [17], and Rudnicki et al. [14].

One may notice, that any all-relevant feature selection algorithm is a special type of classification algorithm. It assigns variables to two classes: *relevant* or *non relevant*. Hence the performance of the algorithms can be measured using the same quantities that are used for estimation of ordinary classifiers. Two measures are particularly useful for estimation of performance: sensitivity  $S$  and positive predictive value  $PPV$ . Sensitivity  $S$  is measured as

$$S = TP / (TP + FN), \quad (2.1)$$

where  $TP$  is a number of truly relevant features recognised by an algorithm,  $FN$  is a number of truly relevant features that are not recognised by an algorithm and  $FP$  is a number of non relevant features that are incorrectly recognised as relevant. Positive predictive value  $PPV$  is measured as

$$PPV = TP / (TP + FP). \quad (2.2)$$

### 2.1.2 Algorithms for All-Relevant Feature Selection

There are two issues that are non-existent for the *minimal optimal* problem, but are very important for the *all relevant* one. The first one is detection of weakly relevant attributes that can be completely obscured by other attributes, the second one is discerning between weakly but truly relevant variables from those that are only seemingly relevant due to random fluctuations.

The concepts of strong and weak relevance, and consequently also the problem of *all relevant* feature selection, are defined in a context of a perfect classifier that is able to use all available information. Yet, in real-world applications one is restricted to use imperfect classification algorithms, that are not capable of using all information present in the information system, and this may influence the outcome of the feature selection algorithm. In particular, an algorithm may not be able to find and use some of the relevant features. In many cases this will not disturb solution of the *minimal optimal* problem, provided that final predictions of a classifier are sufficiently accurate; yet it will significantly decrease a sensitivity of an *all relevant* feature selection. Hence a classification algorithm used in *all relevant* feature selection should be able to detect weak and redundant attributes.

Algorithms that may be used for finding all the relevant features [3, 6, 9, 14, 17] are designed around ensembles of decision trees, either using the random forest algorithm [2] or an algorithm specially tailored for the task. The choice of decision trees as base learners is due to their flexibility and relative robustness, when multiple redundant features are present in the data set. Moreover, the estimate of the variable importance is easily obtained for tree-based ensembles.

The second issue, namely discerning between the truly and randomly relevant attributes arises because the analysis is performed for finite size samples. This gives a chance for random correlations to emerge and significantly influence the results. The probability of such an event increases with the decreasing number of objects; the effect is also boosted by overall large number of attributes, which in addition increases chances for random interactions between features. This issue is handled by introducing ‘contrast variables’ which are used as a reference. A statistical test is performed that compares the importance of original variables with that of contrast variables.

Contrast variables have been used to find all relevant variables by four independent groups. Tuv et al. [17] in ACE algorithm used ensembles of shallow classification trees and iterative procedure in which the influence of the most important variables on decision was removed in order to reveal variables of secondary importance. In each step only these variables that were more important in the statistical test than the 75th percentile of contrast variables were deemed important.

Rudnicki et al. [14] introduced Boruta algorithm that used the importance estimate from the random forest. The algorithm started by establishing initial ranking of variables in random forest. Then the algorithm performed an iterative procedure in which the least important variables were consecutively turned into contrast variables by permuting their values between objects. Then the threshold level was increased by a predefined step and procedure was repeated until the self-consistence was achieved.

The procedure was carried out until the importance of all contrast variables was lower than that of the unperturbed variables.

Huynh-Thu et al. [6] independently proposed a procedure that aimed at the same goal from another end. In this approach the procedure starts similarly from establishing the ranking of importance from RF. Then the algorithm estimates the importance of noninformative variables by turning all variables into contrast variables. In the following steps the algorithm iteratively introduces back the informative variables into the information system and computes the importance of both  $i$  informative variables (the original most important variables) and  $N - i$  noninformative variables.

Dramiński et al. [4] introduced the MCFS algorithm to improve a feature ranking obtained from an ensemble of decision trees. It was constructed in such a way that eliminated known bias of random forest towards variables with fewer number of values. The algorithm was later extended for use as an all-relevant feature selection algorithm [3] by introducing a comparison of the importance of variables with the maximal importance obtained from a set where all variables were uninformative.

The second version of Boruta [9] was introduced to improve computational efficiency and used a different heuristic procedure. In this version the original dataset is extended with random contrast variables. For each original attribute a 'shadow' attribute is generated by randomly permuting its values. Then, for each attribute, it is tested whether its importance is higher than the maximal importance achieved by a contrast attribute. In order to obtain statistically significant results this procedure is repeated several times, with contrast variables generated independently for each iteration. After each iteration, the algorithm checks how many times the importance of tested attributes is higher (or lower) than that of the highest ranked contrast variable. Once this number is significantly higher than allowed under hypothesis of equality with importance of highest random contrast, the attribute is deemed relevant and not tested further. On the other hand, if this number is significantly lower than allowed under the same hypothesis, then the attribute is deemed irrelevant and permanently removed from the data set. The corresponding contrast variables can be either retained or removed from the dataset; the former choice increases precision of the result, whereas the other greatly improves computational efficiency. The algorithm is terminated when either the relevance of all attributes is established or until predefined number of steps is executed. The result of the algorithm is the assignment of each variable to one of three classes—relevant, irrelevant, unresolved (or tentative). The final decision about the unresolved (tentative) attributes is left to the user.

All these algorithms are quite similar to each other: they are based on the ensemble of trees, they use similar measures of importance and use contrast variables to discern relevant and non relevant attributes. They differ mostly in implementation of the statistical test as well as in performance. The current study is devoted to detailed analysis of performance of Boruta algorithm for a family of synthetic data sets with varying number of truly relevant variables and total number of variables, and hence varying difficulty. The difficulty is measured as the error level of the random forest classifier built on the truly relevant variables. The small scale tests performed by us have shown that results of these algorithms are also similar, hence we believe that the analysis performed for single algorithm will be relevant also for other algorithms.

### 2.1.3 Random Forest

The random forest algorithm is used in the current work both as a classifier and as an engine for the feature selection algorithm, hence we give below a short summary of its most important qualities. It is designed as an ensemble of weak classifiers that combine their results during the final classification of each object. Individual classifiers are built as classification trees. Each tree is constructed using different bootstrap sample of the training set, roughly 1/3 of objects is not used for building a tree. At each step of the tree construction a different subset of attributes is randomly selected and a split is performed using an attribute which leads to a best distribution of data between nodes of the tree.

Each object has not been used by roughly 1/3 of trees. This object is called ‘out of bag’ (OOB) for these trees, and they are the OOB trees for this object. One may perform (OOB) error estimate by comparing the classification of the ensemble of the OOB trees for each object with the true decision. The OOB object can be used also for estimation of variables’ importance using following procedure. For each tree all its’ OOB objects are classified and the number of votes for a correct class is recorded. Then values of the variable under scrutiny are randomly permuted across objects, the classification is repeated and the number of votes for a correct class is again recorded. The importance of the variable for the single tree can be then defined as a difference between a number of correct votes cast in original and permuted system, divided by number of objects. The importance of the variable under scrutiny is then obtained by averaging importance measures for individual trees. The implementation of random forest in R library [11] is used in Boruta and also was used for classification tasks.

## 2.2 Testing Procedure

Boruta algorithm is a wrapper on the random forest, hence it is likely that quality of feature selection depends on the quality of random forest model. Therefore in the first step of the testing procedure we performed a series of tests of the random forest algorithm itself on synthetic data sets. Then the performance of the all-relevant feature selection algorithm was examined on the selected synthetic data sets as well as on few real-world data sets.

### 2.2.1 Data Sets

Synthetic data sets were constructed as variants of the well known hypercube problem. In this problem a set of points are generated in corners of  $D$ -dimensional hypercube, each coordinate of the corner is either +1 or -1. The corners of the hypercube were assigned to one of two classes using two methods. The first one relies on random process. Corners of a hypercube are numbered  $1, \dots, 2^D$ , then a random sample of length  $2^{(D-1)}$  is drawn from the range  $(1, \dots, 2^D)$  and corners with these numbers are

assigned to class 1; the remaining corners assigned to class 2. The second method is deterministic. Corners with odd number of  $-1$  coordinates are assigned to class 1 and the remaining corners are assigned to class 2. The points were generated using three methods. In the first one, the points are generated from multidimensional Gaussian distribution with mean zero and standard deviation one and assigned to the nearest corner. In the second method, the multidimensional uniform distribution spanned on  $(-1, 1)$  interval was used instead of Gaussian. In the third one, points were drawn from  $2^D$  multidimensional Gaussian distributions with standard deviation 0.1, each centred on the respective corner of the hypercube. Then two classes of additional features were added to each data set. Features from the first class were obtained as a linear combination of original variables. Features from the second class were drawn randomly from the normal distribution. As a result we obtain the data set described with three types of features. The *generative features* are the original variables used to define the value of decision variable. The *combination features* are obtained as linear combinations of generative features and hence they are also connected with decision variable. These two sets of features are by definition relevant. One should note, however, that features of both types are weakly relevant—it is possible to replace any of the features with combination of other features. The remaining variables are *random features*—they are not connected with decision variable.

Multiple data sets with varying numbers of generative, combination and random features, as well as varying number of objects were generated using four combinations of the class assignment and point distributions methods, resulting in four series of data sets. The first series, denoted as *NORM* used the deterministic class assignment and single Gaussian in a centre for generation of data points. The second one, denoted as *UNI* used deterministic class assignment and uniform distribution of points, the third one used random class assignment and uniform distribution of points. The last series was obtained using random class assignment and Gaussians centered on corners of the hypercube for points generation. Two last series were generated with functions *mlbench.xor* and *mlbench.hypercube* from the *mlbench* package [10] in R [16], with default parameters for data dispersion and are denoted as *XOR* and *HYPER*.

In addition to analysis of synthetic data sets the relevance of the variables was examined for four recently published data sets deposited in the UCI repository [1]: MicroMass, QSAR biodegradation (Q-b) [12], Turkiye Student Evaluation Data Set (TSE) [5] and Amazon Commerce Reviews Set (ACRS).

### 2.2.2 Classification

The tests of classification accuracy for random forest were performed for four series of data sets described earlier. However, the data sets used for survey of classification results were simpler than those used for feature selection. The number of generative variables varied between 2 and 8, the number of combination features was either zero or two times the number of original features and the number of objects varied between 100 and 2,000. The systems were not extended with random features—the goal of this survey was to find the region of parameter space that is feasible for

classification in the best settings, without additional noise from random features. The random forest implementation in R [11] was used to perform classification, using the default parameters: 500 trees in ensemble and the number of variables used for split generation set at the square root of the total.

### 2.2.3 Feature Selection

Two series of data sets were selected for further analysis with Boruta feature selection algorithm implemented in a package in R [8]. In higher dimensions both functions using deterministic class assignment generated data sets that were too difficult for the random forest algorithm, hence only sets generated with the help of two functions from *mlbench* package that were based on random were used in feature selection test.

The result of the classification testing has shown that the quality of models depends monotonically on the number of objects—the OOB classification error decreases with increasing number of objects. This relationship was universal, but in some cases the number of objects required to obtain model of good quality was very high. This is especially true for the high dimensional problems. Therefore to reduce the number of variable parameters we fixed the number of objects at single value 500, that allowed us to scan a wide range of difficulties for numbers of variables varying between 50 and 10,000.

The tests were performed for the following grid of parameters describing data sets:  $N_{gen} = (2, 3, 4, 5)$  generative variables  $\times N_{comb} = (5, 10, 20, 50, 100, 200, 500)$  combination variables  $\times N_{all} = (50, 100, 200, 500, 1,000, 2,000, 5,000, 10,000)$  all variables, where  $N_{all} = N_{gen} + N_{comb} + N_{rand}$  (and  $N_{rand}$  is a number of random variables). Obviously, the grid points corresponding to negative number of random variables were not explored. The number of variable parameters in the test is four, therefore generation of every possible combination is neither feasible nor interesting. Data sets that are either very easy or very difficult are not interesting for further analysis. For the purpose of this work the data set was considered easy when the OOB estimate of the classification error of random forest model is below 2 % and it is considered hard when the OOB error is above 30 %. Therefore only a subset of possible datasets within the range of parameters was generated and tested. For each number of generative variables the initial test system was generated that comprised of 500 objects with 5 combination features and 50 random features. Then the number of objects, combination features and random features was varied until either easy or hard region of the parameter space was found.

Additionally, the influence of number of trees in the forest on the feature selection procedure was examined. To this end, the entire procedure was repeated using random forest classifiers obtained with three different numbers of trees, namely 500, 1,000 and 2,000.

Despite fixing the number of objects and examining only two set series of data sets, the number of possible combinations was still too high to be practical, hence not all of the possible grid points were examined. The set of combinations examined is given in Table 2.1.

**Table 2.1** Data sets generated for the all relevant feature selection analysis

# objects	# generative variables	# combination variables	Total variables	# Trees
500	(2, 3, 4, 5)	(5, 10, 20, 50)	100	(500, 1,000, 2,000)
500	(2, 3, 4, 5)	(5, 10, 20, 50, 100)	200	(500, 1,000, 2,000)
500	(2, 3, 4, 5)	(5, 10, 20, 50, 100, 200)	500	(500, 1,000, 2,000)
500	(2, 3, 4, 5)	(5, 10, 20, 50, 100, 200)	1,000	(500, 1,000, 2,000)
500	(2, 3, 4, 5)	(5, 10, 20, 50, 100, 200)	2,000	(500, 1,000, 2,000)
500	(2, 3, 4, 5)	(5, 10, 20, 50, 100, 200)	5,000	(500, 1,000, 2,000)

In contrast with the synthetic data sets, information on true relevance of variables is unknown for real-world data sets. Therefore, we can measure directly neither sensitivity nor PPV of the algorithm. However, we can estimate the PPV using contrast variables, by measuring how many of them algorithm deems relevant. To this end, we generate contrast variables as ‘shadows’ of original variables, which are obtained by copying values of original variables and randomly permuting them between objects. Each variable is accompanied by a shadow variable. The system extended in this way is then analysed with the Boruta algorithm. Then the PPV estimate is obtained as

$$PPV^* = \frac{N_{relevant}(X_{original})}{N_{relevant}(X_{original}) + N_{relevant}(X_{contrast})}, \quad (2.3)$$

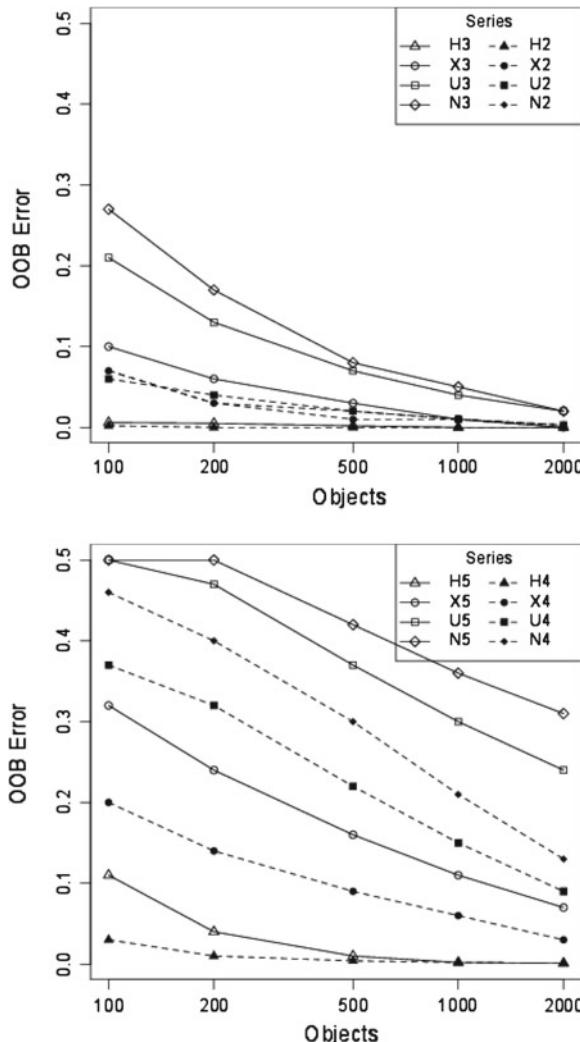
where  $PPV^*$  denotes approximate  $PPV$ ,  $N_{relevant}(X_{original})$  and  $N_{relevant}(X_{contrast})$  are respectively a number of original and contrast variables that algorithm has deemed relevant. Entire analysis was repeated five times to check robustness of the results. Boruta algorithm assigns variables to three classes: (*Confirmed*, *Tentative*, *Rejected*). One can treat the *Tentative* class either as relevant or irrelevant, hence two measures of  $PPV^*$  were used,  $PPV_c^*$  and  $PPV_t^*$  that differed in the assignment of the *Tentative* variables. The former assigns them to irrelevant, whereas latter to relevant class.

### 2.3 Results and Discussion

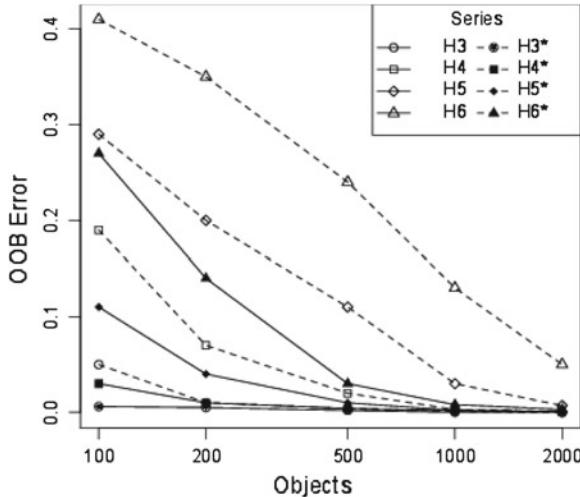
Four series of datasets were generated using small variations of the same approach, nevertheless, the results differ significantly for these sets. Two series of synthetic data sets generated with deterministic class assignment were generally difficult to classify with random forest algorithm. The classification results for these sets were satisfactory (OOB error less than 30 %) only for low dimensional problems (2 and 3). The problems of higher dimensionality were solvable only when large number of objects was available. Therefore further analysis for synthetic sets was performed for two remaining series.

### 2.3.1 Classification

The results of the classification survey are in general agreement with intuitive expectations, see Fig. 2.2. Increasing the dimensionality of the problem makes it more difficult, adding noise to the problem makes it more difficult, and increasing the number of objects helps in building better models.



**Fig. 2.2** Classification results for four variants of hypercube problem. *Top* 2D and 3D models. *Bottom* 4D and 5D models. Labels for series are constructed from the first letter of the series name and dimension of the problem, for example X3 denotes three dimensional data sets from XOR series, H5 denotes five dimensional data sets from HYPER series etc



**Fig. 2.3** The OOB error for two series of HYPER data sets, one with points described with generative variables only, and the other with additional combination features. The number of combination features is two times the number of generative features. The datasets with combination features are marked with a ‘\*’

One result that may be less intuitive is that introduction of features that are linear combinations of original variables may improve the classification. These features in some cases may form lower dimensional subspace that allows to separate clusters located originally in corners of the hypercube. This is not universal, but observed for the last series. Hence presence of the combination features in the data set may facilitate transition of a problem that is formally  $N$ -dimensional to easier  $(N-k)$ -dimensional one. The effect is displayed in Fig. 2.3. The classification error is significantly lower for series with original generative features augmented with linear combinations. This result shows that relationship between importance and true relevance may not be straightforward.

### 2.3.2 Feature Selection

In line with expectations the results of the feature selection are correlated with the results of the classifications. It is difficult to identify important features for data sets that are difficult to classify and relatively easy for those that are easy to classify. This is clearly visible in Table 2.2 that collects the overall results of the survey of the synthetic data sets. For the XOR series the sensitivity is very high for easy 2-dimensional data sets and drops to 25 % for hard 5-dimensional data sets. On the other hand, the level of false discovery is uniformly low—the expected value of false positive discovery is 0.3. It means that on average only 3 falsely relevant variables

**Table 2.2** The cumulative results for the XOR and HYPER series of data sets

DIM	XOR					HYPER				
	TP	FP	FN	Sensitivity (%)	PPV (%)	TP	FP	FN	Sensitivity (%)	PPV (%)
2	51.2	0.3	0.2	98	98	—	—	—	—	—
3	42.1	0.2	8.2	81	97	50.5	0.1	1.3	97	99.7
4	37.0	0.3	15.0	65	99	52.2	0.1	0.7	96	99.6
5	14.2	0.3	36.2	23	97	47.7	0.1	5.3	91	99.0

The average number of false positive, false negative, sensitivity and PPV were computed for the entire range of parameters

**Table 2.3** Cumulative results for four dimensional data set

Ntotal	Mean TP	Mean FP	Mean FN	Mean sensitivity (%)	Mean PPV (%)
100	24.5	0.7	0.8	91.7	97.4
200	40.1	0.5	0.9	90.2	98.7
500	61.6	0.1	6.6	81.5	99.8
1,000	52.4	0.3	15.7	53.6	99.4
2,000	48.2	0.1	19.9	59.7	99.9
5,000	29.7	0.1	42.6	33.8	99.6
10,000	25.6	0.1	57.2	34.2	99.6

The average number of true and false positive, false negative, sensitivity and PPV are displayed for varying number of total variables. The averaging was performed over variable number of combination variables

should be expected in 10 runs of Boruta algorithm. Both sensitivity and PPV are very high for the sets in the HYPER series, hence deeper analysis is devoted to the more difficult XOR series.

The four-dimensional data sets are examined in closer detail in the Table 2.3, where the results for a range of total number of variables is presented. It is clear that the sensitivity of the algorithm drops with increasing number of variables, in line with the number of false positive discoveries.

The drop in sensitivity with increasing number of variables is expected behaviour. When the number of variables is large, the chance for a variable to be included in a tree in few first splits is diminished, hence the impact of individual variable is a subject to larger variability when compared with systems with a small number of variables. Therefore it is more difficult to discern relevant variables with lesser impact from random ones. This effect can be circumvented by increasing a number of trees in the system; see Table 2.4, where cumulative data for all four-dimensional sets is presented as well as a more detailed analysis of a five-dimensional set.

Another interesting effect is presented in Table 2.5. Systems with different number of relevant variables have variable behaviour of sensitivity when the number of variables is increasing. For example, when the total number of variables is 500, the sensitivity is 100 % for a system with 54 relevant variables, whereas it is 87 % for a system with 204 relevant variables. When the number of random variables is

**Table 2.4** Change of sensitivity as a function of a number of trees in Boruta

Ntree	Average for 4D systems				Average for 5D systems			
	TP	FN	Sensitivity (%)	PPV (%)	TP	FN	Sensitivity (%)	PPV (%)
100	—	—	—	—	40.2	164.8	19.6	100
200	—	—	—	—	70.2	134.8	34.2	100
500	34.9	21.1	57.8	99.7	123.6	81.4	60.3	100
1,000	39.9	16.0	63.5	99.5	157.6	47.4	76.9	100
2,000	43.4	12.5	68.5	98.9	180.4	24.6	88.0	99.8
5,000	—	—	—	—	189.6	15.4	92.5	99.5
10,000	—	—	—	—	193.2	11.8	94.2	99.4
20,000	—	—	—	—	195.6	9.4	95.4	99.4

The average results for all 4-dimensional systems examined with Boruta using 500, 1,000, and 2,000 trees are shown in the left panel. The more detailed inspection of results for sets described with 5 generative, 200 combination and 1,000 total variables is presented in the right panel. Average results for five instances are presented for Boruta using 100 to 20,000 trees

**Table 2.5** Results of feature selection presented for two series of 4-dimensional data sets for varying total number of variables in the system

Ncomb	Ntotal	Mean TP	Mean FP	Mean FN	Mean sensitivity (%)	Mean PPV (%)
50	100	54	0.0	0.0	100.0	100.0
	200	54	0.7	0.0	100.0	98.8
	500	54	0.0	0.0	100.0	100.0
	1,000	46.7	1.3	7.3	86.4	97.2
	2,000	52.0	0.3	2.0	96.3	99.4
	5,000	21.7	0.0	32.3	40.1	100.0
	10,000	10.7	0.0	43.3	19.8	100.0
200	500	176.7	0.0	27.3	86.6	100.0
	1,000	173.3	0.0	30.7	85.0	100.0
	2,000	145.7	0.0	58.3	71.4	100.0
	5,000	112.7	0.0	91.3	55.2	100.0
	10,000	84.7	0.0	119.3	41.5	100.0

The average number of true and false positive, false negative, sensitivity and PPV are displayed. The averaging was performed over Random Forest models built from 500, 1,000, and 2,000 trees

increased, the sensitivity for the system with 54 relevant variables drops faster than for the system with 204 ones, reaching 20 % when total number of variables arrives at 10,000, whereas the sensitivity for the system with 204 relevant features is still 40 % at this point.

This effect is most likely due to the method for generation of splits in random forest algorithm. The subset of variables is randomly selected from all variables and split is performed for the variable that produces the best split. When the number of relevant variables is large in comparison with the sample size, the variables with low

importance are rarely selected and hence their apparent importance is similar to that of random variables. When the number of relevant variables is small, but not very small, then there is a good chance that one or two relevant variables will be selected at each step. In this case the truly relevant variables have the highest chance to be selected and hence their apparent importance is high. Finally when the number of variables is very small in comparison with the number of total variables the chance of truly relevant variable being included in the sample is small and this again decreases the apparent importance of relevant variables in comparison with random ones, and hence decreases sensitivity.

The systematic survey of range of synthetic data sets generated with varying parameters shows that the results of Boruta algorithm are robust. While the sensitivity may be low for systems described with very large number of variables, nevertheless, the variables that are reported as relevant are relevant with very high probability.

### 2.3.2.1 Real-World Data Sets

Boruta algorithm has been also applied to four real-world data sets recently deposited in the UCI repository (see Table 2.6). In this case only the false discovery ratio could be estimated since the true relevance of the attributes is unknown. In two cases of the sets described with small number of attributes nearly all attributes were deemed relevant.

The level of false discovery was very low. In all cases the  $PPV_c^*$  was 100%—not a single false discovery was made with the strict definition of relevance. With the more relaxed definition, accommodating also Boruta’s tentative class as relevant, some false discoveries were reported for QSAR biodegradation data set. Nevertheless, even in this case the expected value of false discovery was 0.4 and  $PPV_t^*$  was 98.9%. Therefore we may assume that nearly all features identified by Boruta as relevant are truly so. The case of QSAR biodegradation data set could suggest that variables assigned by Boruta to tentative class, bear higher risk of being false positive.

**Table 2.6** Results for the real-world data sets from the UCI repository

Dataset	Data		Original			Contrast			$PPV_c^*$	$PPV_t^*$
	Instances	Variables	Conf	Tent	Rej	Conf	Tent	Rej	(%)	(%)
Q-b	1,055	41	36.2	0.8	4.0	0.0	0.4	40.6	100.0	98.9
TES	5,820	33	30.0	1.0	1.0	0.0	0.0	32.0	100.0	100.0
MM-500	931	1,300	293	66	941	0	0	1,300	100	100
MM-1000	931	1,300	363	58	879	0	0	1,300	100	100
ACRS	1,500	10,000	220	84	9,696	0.0	0.0	10,000.0	100.0	100.0

The MicroMass data set was analysed with Random Forest runs with 500 and 1,000 trees that are described as MM-500 and MM-1000, respectively. The number of variables marked as confirmed (Conf), tentative (Tent) and rejected (Rej) is reported for original and contrast variables. The  $PPV_c^*$  was computed according to Eq. 2.3 counting as relevant only these variables with *confirmed* status, for  $PPV_t^*$  also variables with *tentative* status were taken into account

Nevertheless, in the case of MicroMass data set all attributes deemed tentative by Boruta using 500 trees, were later deemed confirmed by Boruta using 1,000 trees, without any false positive hits. This suggests that when the number of variables deemed tentative is large, it is quite likely that most of them are truly relevant and Boruta run with larger number of trees is required.

## 2.4 Conclusions

As it was demonstrated in the chapter, the all-relevant feature selection algorithms are capable of discerning between relevant and non relevant variables. The Boruta algorithm, which was used as a representative algorithm of the class, was examined on a wide range of synthetic problems and several recently published real-world data sets. Algorithm works particularly well for systems for which good quality models may be obtained by means of random forest classification algorithm. The sensitivity of the algorithm is close to 100 % for such systems. The sensitivity of Boruta can be improved by utilising random forest with larger number of decision trees. The level of false discoveries is very low for all data sets examined, therefore *all relevant* feature selection is suitable for generation of robust knowledge.

The main factor limiting analysis with Boruta algorithm is time of computations. The single iteration of the random forest algorithm can take several hours for larger systems. The algorithm in the best case requires at least time equivalent to 30 random forest iterations to complete, hence entire analysis may take more than one CPU-week. The random forest is computationally demanding and its implementation in R, while very useful, is not very efficient for large problems. In particular, while the random forest is trivially parallel its implementation is strictly sequential. This limits application of the algorithm for analysis of truly large datasets described with tens or even hundreds thousands variables and thousands of objects.

**Acknowledgments** Computations were partially performed at the Interdisciplinary Centre for Mathematical and Computational Modelling, University of Warsaw, Poland, grant G34-5. Authors would like to thank Mr. Rafał Niemiec for technical help.

## References

1. Bache, K., Lichman, M.: UCI machine learning repository. <http://archive.ics.uci.edu/ml> (2013)
2. Breiman, L.: Random forests. *Mach. Learn.* **45**, 5–32 (2001)
3. Draminski, M., Kierczak, M., Koronacki, J., Komorowski, J.: Monte Carlo feature selection and interdependency discovery in supervised classification. In: Koronacki, J. (ed.) *Advances in Machine Learning II*. SCI, vol. 263, pp. 371–385. Springer (2010)
4. Draminski, M., Rada-Iglesias, A., Enroth, S., Wadelius, C., Koronacki, J., Komorowski, J.: Monte Carlo feature selection for supervised classification. *Bioinformatics* **24**(1), 110–117 (2008)

5. Gunduz, N., Fokoue, E.: UCI machine learning repository. <http://archive.ics.uci.edu/ml/datasets/Turkiye+Student+Evaluation> (2013)
6. Huynh-Thu, V.A., Wehenkel, L., Geurts, P.: Exploiting tree-based variable importances to selectively identify relevant variables. In: JMLR: Workshop and Conference Proceedings, vol. 4, pp. 60–73 (2008)
7. Kohavi, R., John, G.: Wrappers for feature subset selection. *Artif. Intell.* **97**(1), 273–324 (1997)
8. Kursa, M.B., Rudnicki, W.R.: Feature selection with the Boruta package. *J. Stat. Softw.* **36**(11), 1–13 (2010)
9. Kursa, M.B., Jankowski, A., Rudnicki, W.R.: Boruta—a system for feature selection. *Fundam. Inform.* **101**(4), 271–285 (2010)
10. Leisch, F., Dimitriadou, E.: mlbench: machine learning benchmark problems. R package version 2.1–1 (2010)
11. Liaw, A., Wiener, M.: Classification and regression by random forest. *R News* **2**(3), 18–22 (2002). <http://CRAN.R-project.org/doc/Rnews/>
12. Mansouri, K., Ringsted, T., Ballabio, D., Todeschini, R., Consonni, V.: Quantitative structure-activity relationship models for ready biodegradability of chemicals. *J. Chem. Inf. Model.* **53**(4), 867–878 (2013)
13. Nilsson, R., Peña, J.M., Björkegren, J., Tegnér, J.: Detecting multivariate differentially expressed genes. *BMC Bioinform.* **8**, 150 (2007)
14. Rudnicki, W.R., Kierczak, M., Koronacki, J., Komorowski, J.: A statistical method for determining importance of variables in an information system. In: Greco, S., Hata, Y., Hirano, S., Inuiguchi, M., Miyamoto, S., Nguyen, H., Slowinski, R. (eds.) *Rough Sets and Current Trends in Computing*, vol. 4259/2006, pp. 557–566. Springer, Berlin/Heidelberg (2006)
15. Stoppiglia, H., Dreyfus, G., Dubois, R., Oussar, Y.: Ranking a random feature for variable and feature selection. *J. Mach. Learn. Res.* **3**(7–8), 1399–1414 (2003)
16. Team, R.C.: R: A language and environment for statistical computing. R foundation for statistical computing, Vienna, Austria (2012). <http://www.R-project.org/>
17. Tuv, E., Borisov, A., Torkkola, K.: Feature selection using ensemble based ranking against artificial contrasts. In: The 2006 IEEE International Joint Conference on Neural Network Proceedings, pp. 2181–2186. IEEE (2006)

# Chapter 3

## Feature Evaluation by Filter, Wrapper, and Embedded Approaches

Urszula Stańczyk

**Abstract** The choice of particular variables for construction of a set of characteristic features relevant to classification can be executed in a kind of external process with respect to a classification system employed in pattern recognition, it can depend on the performance of such system, or it can involve some inherent mechanism, build-in in the system. The three types of approaches correspond to three categories of methodologies typically exploited in feature selection and reduction: filters, wrappers, and embedded solutions, respectively. They are used when domain knowledge is unavailable or insufficient for an informed choice, or in order to support this expert knowledge to achieve higher efficiency, enhanced classification, or reduced sizes of classifiers. The chapter illustrates the combinations of the three approaches with the aim of feature evaluation, for binary classification with balanced, for the task of authorship attribution that belongs with stylometric analysis of texts.

**Keywords** Feature evaluation · Filter · Wrapper · Embedded solution · DRSA · ANN · Stylometry · Authorship attribution

### 3.1 Introduction

Since inductive learning systems can suffer from both insufficient and excessive numbers of characteristic features they depend on, the problem of feature selection and reduction has become quite popular and widely studied, with methodologies applied typically grouped into three main categories: filters, wrappers, and embedded solutions [17].

Filters work independently on a classifier involved in pattern recognition, regardless of its specifics and parameters [14]. The choice of attributes is performed basing on some algorithms, quality measures, for example by referring to information

---

U. Stańczyk (✉)

Institute of Informatics, Silesian University of Technology,

Akademicka 16, 44-100 Gliwice, Poland

e-mail: urszula.stanczyk@polsl.pl

theory. Filters are general in nature and this generality should be understood here as applicability to any domain, any inducer. This universality is, however, most often achieved at a cost of some lower classification accuracy than for other approaches.

In wrappers selection of features is conditioned by the performance of the inducer itself and its characteristics [16]. Typically, the predictive accuracy is considered as the most important and deciding factor. Dependence on some particular classifier means loss of generality and bias, but at the same time close tailoring of the set of inputs to local requirements usually results in improved performance.

A solution is called embedded when an algorithm for feature selection and elimination is a part of the learning system, some inherent dedicated mechanism that is actively used [8]. As examples from this category there can be given construction of decision trees, artificial neural networks with pruning of input neurons, activation of relative reducts in rough set processing.

The chapter presents examples of combined filter, wrapper, and embedded approaches for rule and connectionist classifiers employed for evaluation of features in stylometric (or computational stylistics) domain, for a case of binary authorship attribution. The considered features reflect lexical and syntactic characteristics expressing writing styles [2]. The stylistic features are studied and evaluated within two contexts: firstly by their established rankings, secondly in the observed performance of classifiers employing sequential backward selection while following these rankings.

The text of the chapter is organised as follows. Section 3.2 presents fundamental notions of stylometric processing of texts and features used in such analysis. Section 3.3 is dedicated to the differences in approaches to variable selection process, while Sect. 3.4 provides some details of experimental setup, and Sects. 3.5 and 3.6 contain illustration of test results. Section 3.7 concludes the chapter.

## 3.2 Characteristic Features for Stylometric Analysis of Texts

Stylometry is a branch of science dedicated to understanding of writing styles, their characteristics and descriptive elements, shared and unique traits, aiming at knowledge discovery from linguistic point of view, but also at author characterisation, comparison, and recognition [7, 31]. Stylometric processing typically involves either statistic-oriented computer-aided computations [20], or methodologies from machine learning domain [34]. Once we obtain a definition of a writing style by some characteristic features, the task of recognising it can be perceived as pattern recognition, with text samples categorised and classified by their authors.

A style is a phenomenon which we grasp and recognise rather intuitively, but usually have trouble with more formal definitions and descriptions [3]. While we can typically tell that we prefer someone's style over others, expressing the reasons for our preferences, especially not in some general qualificatory terms such as "good", "bad", "enjoyable", "boring", etc., but in more detail, comes much harder.

To employ contemporary data mining techniques for stylometric analysis [1], quantitative instead of qualitative descriptors are required and they are based on statistics of linguistic features. As their selection reflects the richness of language, the list of existing possibilities is practically endless. The markers often exploit term frequencies of occurrence [24] and are divided into four categories: lexical, syntactic, structural, and content specific [30].

Lexical descriptors provide information about total numbers of characters or words, averages of numbers of characters per word or sentence, words per sentence, distributions of these numbers. Syntactic markers express the structure of sentences as created by punctuation marks [4]. Structural attributes reflect the overall organisation of a text into paragraphs, sections, headings, signatures, embedded formatting elements. Content-specific features refer to words and phrases of key meaning in some context [6]. Out of these four groups, typically in authorship attribution tasks there are chosen lexical and syntactic descriptors [41].

Even though it is universally acknowledged that it is possible to execute reliable authorship attribution while using stylometric descriptors as characteristic features, there is no consensus with regard to the way in which these sets of variables should be constructed. Of course, as always there is needed a sufficiently high number of representative text samples, but basing on them various candidate subsets of attributes can be prepared and the knowledge about their efficiency and relevance for the purposes of classification is unavailable a priori.

With the absence of domain knowledge about the importance of attributes a different attitude can be tried, by applying some methodology that by itself can discover relevance of variables, or some approaches dedicated to feature selection and reduction, either single, or in combinations. Even when expert knowledge is available, feature selection algorithms can help with dimensionality reduction, improvement of obtained results. When the task of feature set construction is considered in the context of data processing and mining, it can be biased by a particular technique used, with the result of the possible existence of alternative feature sets, found by other approaches or computations. Thus the widely accepted procedure is to propose some candidate set of attributes (chosen by arbitrary assumptions, using statistics, or heuristics), test its quality, and optimise it for the set criteria.

### 3.3 Approaches to Feature Selection

In many classification tasks the total number of possible features that can be employed is relatively high. Using all of them would result in respectively high dimensionality which encumbers processing (even may make it impractical), also the presence of too many variables is a drawback to most inducers even when these attributes by themselves are relevant for the task, not to mention irrelevant or redundant variables which can obscure other patterns [18]. In such cases several candidate subsets can be tried and their efficiency tested, or we can employ some of algorithms explicitly dedicated to feature selection and reduction.

Depending on the organisation of a search process, feature selection algorithms are typically categorised as belonging with filters, wrappers, or embedded approaches. There are also constructed combinations of approaches, where for example firstly a filter is employed, then wrapper, or when a wrapper is used as a filter. It is also possible to apply some algorithm to obtain ranking of attributes, basing on which feature selection or reduction is next executed.

### ***3.3.1 Filters***

Filters are completely separate processes to systems used for classification, working independently on their performance and other parameters. They can be treated as kind of pre-processing procedures. They exploit information contained in input data sets looking for example for information gain, entropy, consistency [9].

One of popular algorithms from this group is Relief, in its original form invented for binary classification (later modified to allow for multiple classes) [43]. Relief assigns scores to variables depending on how well they discern decision classes. It randomly samples the training set, looking for the two examples that are nearest to the one selected, one from the same class (near-hit) while the other from the opposite class (near-miss), and basing on this iteratively accumulates weights for attributes. One of the drawbacks of Relief algorithm is the fact that it looks for all relevant features and cannot discern redundant features, even when they are relevant in a very low degree, thus each variable has some weight assigned.

The general nature of filters makes them applicable in all cases, yet the fact that they totally disregard the performance of a classification system employing the set of selected variables causes typically worse results than other approaches and it is considered as a disadvantage.

### ***3.3.2 Wrappers***

In a wrapper approach to feature selection it is argued that the best evaluation of some candidate variable subset is obtained by checking its usefulness in classification, as the estimated predictive accuracy is typically considered to be the most important indicator of relevance for attributes [22]. The induction algorithm can be run over the entire training set and then measured against the testing set, or a cross-validation method can be employed.

Since the search and selection process is adjusted to specific characteristic of the inducer, they can show a bias, resulting in an increased performance for the chosen classifier but worse results for another, especially when they significantly vary in properties. In other words wrappers tend to construct sets of attributes which are customised, tailored to some particular task and some particular system.

Another disadvantage of this approach is in computational costs required. Execution of the learning algorithm for many subsets of features can become unfeasible, not only when there are very high numbers of attributes to consider, but also in cases when the training step is complex and time-consuming even for smaller numbers of variables. For example artificial neural networks deal much better with more than necessary inputs than for situations when their number is low.

When an inducer is able by itself to select some features while disregarding others due to some additional procedures dedicated to dimensionality reduction, they cannot be used for the wrapper mode to play its role. If such processing is employed, it results in an embedded approach.

Wrapper model can be used not only for feature selection or reduction, but for other purposes, to better adjust some parameters of a classification system. An example of such procedure constitutes establishing preference orders for values of conditional attributes in Dominance-based Rough Set Approach, when there is insufficient domain knowledge for such definitions [39].

### ***3.3.3 Embedded Solutions***

Several of predictors have their own, inherent mechanisms, built-in in the learning algorithm, dedicated to feature selection. When such mechanism is actively used we have an embedded solution [23].

As an example of this category there can be given input pruning for artificial neural networks [19, 21] that leads to establishing by repetitive computations which of network inputs have very low influence on the network outputs. In decision trees in fact at each node some feature is selected, and this decision is a constituent element of the algorithm, cannot be simply separated from it. For rough sets such function is played by relative reducts, subsets of attributes which guarantee the same quality of approximation as the entire set of variables [26].

When a set of all relative reducts is treated as yet another entity, another form of expression for available knowledge on features extracted from instances, we can assign weights [25] and define some quality measures for them [40, 42], to be used in feature selection. These measures and weights can take into account how often each attribute is included in reducts of specific cardinalities, and the same statistics for other variables included in the same reducts. This kind of processing leads to ordering of features, which can be interpreted as their ranking.

### ***3.3.4 Ranking of Features***

When we proceed through the entire set of available features with an application of any of the aforementioned approaches to feature selection, either single or in combinations, as a result these features become ordered by a value of some score or

measure, considered then a weight or rank, assigned through processing. Therefore, we obtain in such case a ranking of variables [27].

When a ranking of attributes is exploited in some processing, for feature selection or reduction, and it is executed independently on the procedures that led to ranking in the first place, by a formal definition the ranking can be then perceived as a filter. Even a wrapper can be employed as a ranking filter in the subsequent stage of calculations, as long as it follows a search path that gives the ordering of all variables, and the inducer from the second stage is different from the first one.

### 3.4 Details of Research Framework

Before conducting any experiments several decisions needed to be made, with respect to:

- input data sets—defined by the numbers of analysed learning and testing samples, and a set of available stylometric characteristic features,
- machine learning techniques used in classification,
- the point in the feature space where search procedures start and directions of the search,
- the stopping criterion for the search process,
- evaluation method for a candidate variable subset,
- organisation of the search,

as described below in more detail.

#### 3.4.1 Input Data Sets

To ensure reliability of detected patterns in linguistic habits and preferences, statistics must be obtained basing on several samples of writing, with each sample of sufficient length. In the considered case there were taken novels by two famous writers, Thomas Hardy and Henry James. Since within documents that are so long it is natural to perceive some small variations of styles depending on the character of text parts (narrative or dialog), they were divided into smaller samples, corresponding to chapters or sections, to keep comparable length and size.

For all these prepared parts next the characteristic features were extracted for 25 arbitrarily selected lexical and syntactic descriptors (which proved to be useful in some past research on authorship attribution [35, 36]), by calculation of frequencies of usage for some function words and punctuation marks as follows:

- lexical markers (17)—but, and, not, in, with, that, what, for, by, if, from, at, to, as, on, of, this,
- syntactic markers (8)—a comma, a fullstop, a colon, a semicolon, a bracket, a question mark, an exclamation mark, a hyphen.

### **3.4.2 Machine Learning Techniques Used in Research**

In the research performed two distinctively different approaches to classification were used, namely artificial neural networks simulated by software in multilayer perceptron (MLP) topology, and decision algorithms induced within Dominance-based Rough Set Approach (DRSA).

#### **3.4.2.1 ANN Classifier**

MLP is a feed-forward, unidirectional network, which at the training phase often employs backpropagation algorithm [11]. Firstly for all connections some random weights are assigned, then they are modified in such way that results in minimising the difference between the value generated on the network output and the one that is expected, for all outputs and all training samples. Its popularity MLP owes to good generalisation properties—once a network learns characteristics of the training set, it can correctly classify also unknown instances.

Within the first steps of ANN classifier construction that encompasses establishing network parameters the number of input nodes was set to the number of considered variables, the number of outputs as corresponding to two recognised classes, and in the internal structure the number of hidden layers was set to two, with the total number of neurons in them equal to the number of inputs. To minimise the influence of random interconnection weights on the training phase the multi-starting procedure was employed, with repetitive learning and calculations of median, minimal, and maximal performance.

#### **3.4.2.2 DRSA Classifier**

In rough set approach the objects of the universe are perceived through granules of knowledge [28]. In classical version, invented by Pawlak [29], these granules are equivalence classes of instances that cannot be discerned basing on values of considered conditional attributes. Classical Rough Set Approach (CRSA) enables only nominal classification, which can be insufficient for multicriteria decision making [10]. Replacing indiscernibility relation with dominance and observing weak preference orderings in value sets of attributes allows for ordinal classification and gives Dominance-based Rough Set Approach [12].

DRSA approximates dominance cones—upward and downward unions of decision classes and induces decision rules that form rule-based classifiers. The advantage of this approach is enhanced understanding of approximated information, as rules explicitly specify the conditions that need to be met for some object to be classified to the specific class (to be precise, in this case to the union of classes). There are many algorithms for rule induction and depending on them the sets of generated rules can greatly vary [5, 32], not only with respect to their cardinalities, but, more

importantly, in quality. In the performed tests all rules on learning samples were found and then in each studied case limited by introducing hard constraints to the type of rules and their support [37, 38].

The inferred rules can be certain (exact), possible, or approximate and only the first type without additional processing indicates recognition results. In the research conducted only certain rules were taken into account.

To reduce processing time required also a very strict rule was applied to ambiguous decisions (cases of no rules matching, or multiple rules but with contradicting decisions)—instead of solving the matter by voting, weighting, or both, all ambiguous decisions were always treated as incorrect.

### **3.4.3 Search Parameters**

Yet another of the search parameters, which we need to decide upon before we start, is the point in the input feature space, from which the algorithm begins its execution. Taking into account the dimensionality of this search space, and all candidate subsets of features that can be found, this initial set is either arbitrarily selected as empty and then variables are added to it in forward direction, or as an entire set of attributes from which elements are reduced backward, or there is chosen some other subset and then we can both add and remove features, checking in both directions.

The exhaustive search, with evaluation of all possible candidate subsets, is rarely executed as it is typically too time-consuming, and only a part of these subsets are tested. We can stop the search when the maximal performance is obtained (but we cannot be then sure that the maximum is global and not local), but we can also make the search complete with respect to the search path—that is it can end upon reaching the point in space that is opposite to the starting one. This last attitude was exploited in executed tests.

The search procedures started with the entire set of considered stylometric features and then their sequential backward selection was executed, by removing one variable at a time, until there was no attribute left. To evaluate a subset of features two separate groups of samples were involved—one for induction phase and the other for testing.

## **3.5 Feature Evaluation by Ranking**

Within the research presented in this chapter the two-step work framework was implemented. The first step encompassed evaluation of relevance for characteristic features by obtaining their ranking.

The rankings of features were calculated by Relief algorithm [43], and by employing embedded DRSA processing. Relief algorithm establishes through iterative calculations how well individual attributes discern defined decision classes, whereas in rough processing (described in detail in [42]) ranks of variables depend on their occurrences in relative reducts, to which weights are assigned.

Both rankings are listed in Table 3.1. The attributes are given in decreasing order with respect to their importance as perceived by the two algorithms, that is starting with these variables which are considered more important, then with gradually lower and lower significance expressed by scores they were assigned. For comparison purposes also both reversed orderings of variables were tested in the second stage of experiments, presented in the next section.

When the two rankings of features are compared against each other we can observe some similarities, after all both algorithms operate on the same input set. However, there are also significant differences. They both consider “and” attribute as the most important and “as” takes the same place, but the rest of variables is put in a different order. “But” and “that”, placed among lowest ranking for DRSA order, are among the highest ranking features for Relief. “Of”, the last on Relief list, is in the upper half for DRSA ranking, and so on. Different focus of both ranking algorithms resulted in different weights being assigned to majority of attributes.

**Table 3.1** Rankings of features obtained in the first stage of experiments

Nr	Ranking by	
	Relief algorithm	Embedded DRSA processing
1	and	and
2	that	not
3	but	by
4	from	.
5	:	on
6	not	from
7	?	in
8	for	;
9	by	if
10	-	of
11	what	,
12	at	!
13	!	:
14	with	at
15	if	-
16	on	to
17	as	as
18	to	this
19	this	with
20	in	?
21	(	what
22	.	(
23	,	that
24	;	for
25	of	but

### 3.6 Feature Evaluation by Backward Reduction

Within the second phase of experiments the considered stylometric features were evaluated by observation of performance of classification systems executing sequential backward elimination, while following the previously established ranking. It resulted in combining filter, wrapper, and embedded approaches.

Firstly there were constructed ANN and DRSA classifiers for the entire set of studied features. The trained network correctly recognised median 83.33 % of samples, while generated all rules on examples algorithm, with hard constraints on minimal support required of rules being 41, classified without any ambiguity 76.67 % of instances. These two results were next treated as reference points, when sequential backward elimination was executed.

For neural networks reduction of variables corresponded to decreasing the number of inputs (and also the number of neurons in the two hidden layers) and repeating the complete training procedure for such modified topologies.

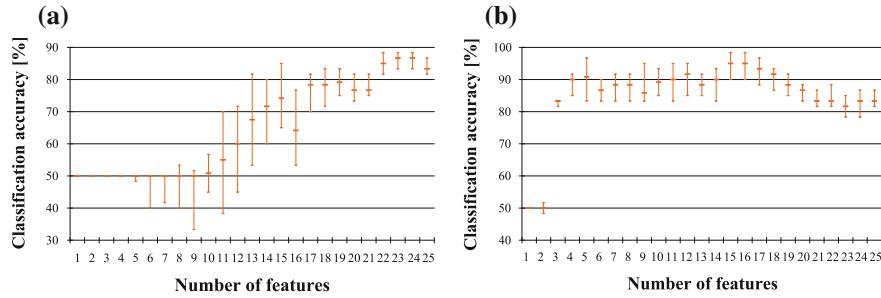
For DRSA processing it is also possible to conduct it in the same way, that is eliminate attributes and construct a new classifier by induction of new decision rules. However, with generation of all rules in each case the task would be very time-consuming—for the entire set of variables the algorithm comprised 46,191 constituent rules. With lower number of features we can expect this number to decrease yet we can also expect that at least some part of these rules would be the same. Therefore, instead of generating the same rules over and over again, another attitude is employed and the process of attribute reduction is applied to the already induced all rules on examples algorithm for the entire set of features. When a variable is reduced, all rules having conditions on this variable are discarded (regardless on other conditions they may include in the premise part) from the set and the new, reduced algorithm is constructed. Such process is executed much faster than repeated induction of rules.

The experiments were organised in two series, depending on the ranking of characteristic features controlling backward reduction for both types of classifiers employed in the stylometric task of authorship attribution.

#### 3.6.1 Relief Ranking

The first group of executed tests was focused on elimination of characteristic features for ANN and DRSA classifiers, while following the ranking of attributes returned by Relief algorithm (see Table 3.1).

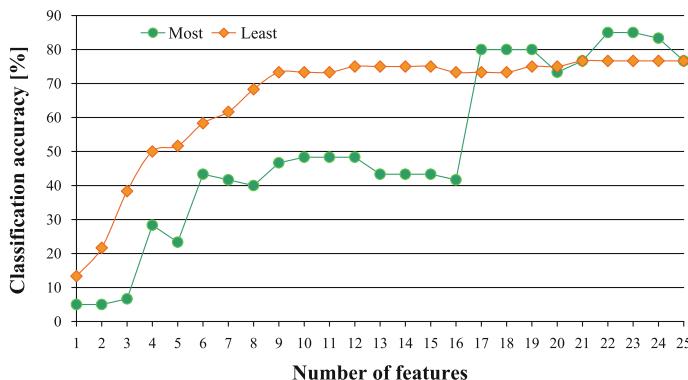
The performance of the connectionist classifier for decreasing number of input nodes is plotted in Fig. 3.1. The graph in Fig. 3.1a displays reduction of variables in the decreasing order, starting with those which are ranked the highest, and then along lower and lower rank. Elimination of features in the reversed order, that is when the first to go are the variables with the lowest ranking, is given in Fig. 3.1b.



**Fig. 3.1** ANN classification accuracy in relation to the number of considered features, observed in sequential backward elimination process while employing feature ranking obtained by Relief filter: **a** decreasing order, **b** increasing order. For each median there is indicated maximal and minimal performance

We can observe that reduction of the highest ranking features in the initial phase, for just few variables discarded, gives a slight increase in performance, but it soon falls down to an unacceptably low level. On the other hand, rejection of low ranking features enables to keep the classification accuracy at sufficiently high level even when there are only few inputs left for the network to learn from.

Similar results were obtained for rule classifiers as shown in Fig. 3.2. The trends reflect those previously detected, but overall comparison of both types of classifiers indicates that ANN outperforms DRSA decision algorithm. This conclusion is not entirely accurate due to the fact that the network with the used topology classifies without any ambiguity while for DRSA classifier ambiguities did occur and were treated as incorrect decisions.

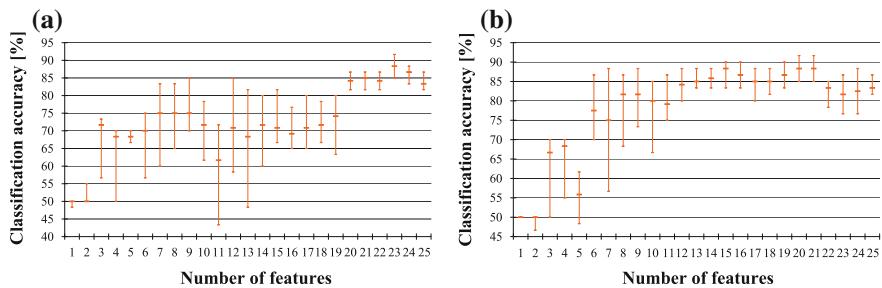


**Fig. 3.2** DRSA classification accuracy in relation to the number of considered features, observed in sequential backward elimination process of all rules on examples algorithm, while employing feature ranking obtained by Relief filter for a decreasing order (Most series), and increasing order (Least series)

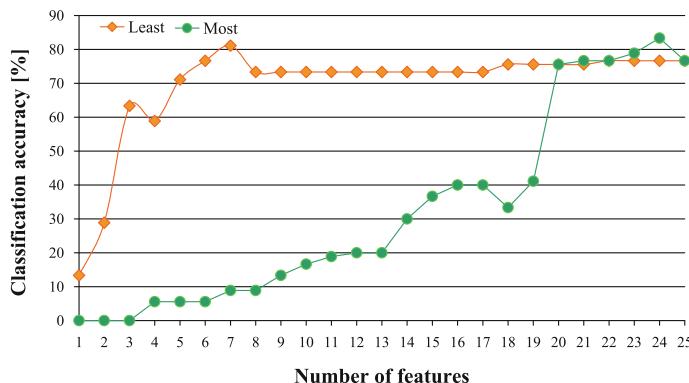
### 3.6.2 Embedded DRSA Ranking

The second batch of performed tests employed an embedded DRSA ranking (see Table 3.1) in backward reduction of features, again for both connectionist and rule-based classifiers. Classification results for artificial neural networks are presented in Fig. 3.3, with reduction of the highest ranking features in Fig. 3.3a and elimination of lowest ranking variables in Fig. 3.3b. The pattern visible in the two graphs shows close resemblance to the one observed for the first tested ranking. We can also note that in most cases there are bigger differences between the maximal and minimal performances of the tested networks.

Figure 3.4 illustrates reduction of characteristic features with highest and lowest rank, based on the embedded DRSA approach, for decision algorithms. For the decreasing order in the initial phase of reduction the results are acceptable, but once



**Fig. 3.3** ANN classification accuracy in relation to the number of considered features, observed in sequential backward elimination process while employing feature ranking obtained by an embedded approach based on relative reducts: **a** decreasing order, **b** increasing order. For each median there is indicated maximal and minimal performance



**Fig. 3.4** DRSA classification accuracy in relation to the number of considered features, observed in sequential backward elimination process of all rules on examples algorithm, while employing feature ranking obtained by embedded approach based on relative reducts, for a decreasing order (Most series), and increasing order (Least series)

more than five important variables are eliminated the performance gets worse quickly and irrecoverably. We can reduce significantly more lower ranking variables before the performance degrades below a satisfactory level.

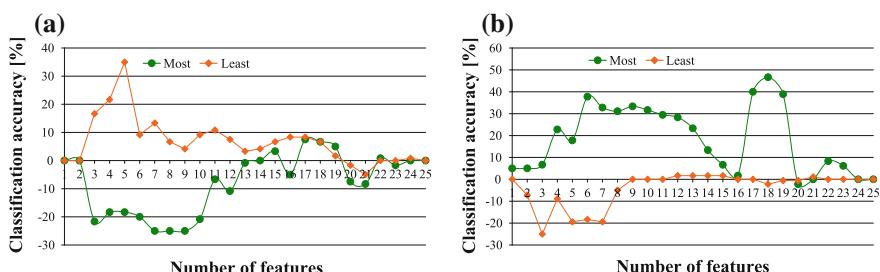
### 3.6.3 Comparison of Feature Reduction Results

For both types of classifiers we can also observe the differences and similarities in the performance in the feature reduction process while following the two defined rankings, as shown in Fig. 3.5. The displayed values correspond to the calculated difference in classification accuracy for the currently considered number of features, Relief-based reduction minus DRSA-based reduction. For ANN classifiers the differences are calculated only with respect to median classification accuracies.

For artificial neural networks for both increasing and decreasing orders for both rankings the results are close as the differences are equal to or close to zero. The discrepancies are visible when there are fewer than a half of features left. Then reduction of those with lower ranks returns better results for Relief (at maximum for five variables, when the difference in classification accuracy is 35 %), and for elimination of higher ranking attributes DRSA-based ordering is more advantageous (at maximum outperforms by 25 % for 8, 7 and 9 remaining features).

For rule classifiers for the decreasing order for both rankings the classification accuracies are almost the same for the first 16–17 reduced variables. Then, for fewer than 8 inputs, Relief ranking based selection of attributes is outperformed by DRSA-based reduction. For elimination of higher ranking variables in most cases the results are better while employing Relief than DRSA-based ranking.

As can be seen in presented graphs, executing sequential backward reduction of characteristic features driven by ranking of these features obtained previously results in several cases with significant gains in terms of lower dimensionality, increased predictive accuracies of the constructed classification systems, and decreased storage requirements. Observations based on performance allow for evaluation of used attributes and estimation of their relevance for particular tasks.



**Fig. 3.5** Differences in classification for reduction of characteristic features along Relief and embedded DRSA ranking, for decreasing orders (Most series), and increasing orders (Least series) for: **a** ANN classifier, **b** DRSA classifier

### 3.7 Conclusions

Both connectionist and rule-based classifiers are often used in cases when observation of some subtle patterns in input data sets is required for recognition, when available information is incomplete and uncertain, and generalisation property is exploited to its fullest extent. Stylometry, which is dedicated to analysis of writing styles, constitutes an example of an application domain with such characteristics, and it employs machine learning techniques and methodologies to solve its tasks of writer characterisation, comparison, and, the one considered as the most important, writer recognition.

Once a writing style is defined by stylometric characteristic features that correspond to linguistic descriptors, authorship attribution can be treated as classification, binary or multi-class, depending on the total number of compared writers.

All types of classification systems can suffer from excessive numbers of variables, by extended processing time, complex calculations required, or simply by irrelevant features obscuring those that are of the highest importance for a task and causing worse performance. Those and many other reasons give motivation to pursue the avenue of various feature selection and evaluation approaches and their efficiency.

The chapter illustrates the two-step processing framework for research on feature selection and evaluation, combining filter, wrapper, and embedded approaches to the problem. Within the first phase two rankings of variables are obtained, the one with application of Relief algorithm (as implemented in WEKA Software), and the second embedded in rough set model, referring to the concept of relative reducts, and weights defined for them. The two rankings are then used in the second stage of experiments to filter out the input features for artificial neural networks (simulated by software) and rule classifiers induced within Dominance-based Rough Set Approach (DRSA). The performance is observed from the starting point where the entire set of available features is involved in classification, then in sequential backward elimination of one variable at a time, till there are no features left, which is the stopping point for processing.

For both types of classifiers, and both rankings, decreasing as well as increasing orderings of attributes were tested and overall results compared, revealing significant gains in classification accuracies and the reduced sizes of systems, and by that betraying the importance of attributes, which validates the presented methodology.

**Acknowledgments** All texts used in the performed experiments are available for on-line reading and download thanks to Project Gutenberg (<http://www.gutenberg.org>). 4eMka Software used in DRSA processing [13, 33] was developed at the Laboratory of Intelligent Decision Support Systems, (<http://www-idss.cs.put.poznan.pl/>), Poznan University of Technology, Poland. For simulation of ANN there was used California Scientific Brainmaker software package. Ranking of features with Relief algorithm was executed with WEKA software [15].

## References

1. Ahonen, H., Heinonen, O., Klemettinen, M., Verkamo, A.: Applying data mining techniques in text analysis. Technical Report C-1997-23. Department of Computer Science, University of Helsinki, Finland (1997)
2. Argamon, S., Karlgren, J., Shanahan, J.: Stylistic analysis of text for information access. In: Proceedings of the 28th International ACM Conference on Research and Development in Information Retrieval, Brazil (2005)
3. Argamon, S., Burns, K., Dubnov, S. (eds.): *The Structure of Style: Algorithmic Approaches to Understanding Manner and Meaning*. Springer, Berlin (2010)
4. Baayen, H., van Haltern, H., Tweedie, F.: Outside the cave of shadows: using syntactic annotation to enhance authorship attribution. *Lit. Linguist. Comput.* **11**(3), 121–132 (1996)
5. Bayardo Jr, R., Agrawal, R.: Mining the most interesting rules. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 145–154 (1999)
6. Berber Sardinha, T.: Using key words in text analysis: practical aspects. Available on-line from <ftp://ftp.liv.ac.uk/pub/linguistics> (1999)
7. Craig, H.: Stylistic analysis and authorship studies. In: Schreibman, S., Siemens, R., Unsworth, J. (eds.) *A Companion to Digital Humanities*. Blackwell, Oxford (2004)
8. Dash, M., Liu, H.: Feature selection for classification. *Intell. Data Anal.* **1**, 131–156 (1997)
9. Dash, M., Liu, H.: Consistency-based search in feature selection. *Artif. Intell.* **151**, 155–176 (2003)
10. Deutsch, I., Gediga, G.: *Rough Set Data Analysis: A Road to Noninvasive Knowledge Discovery*. Methodos Publishers, Bangor (2000)
11. Fiesler, E., Beale, R.: *Handbook of Neural Computation*. Oxford University Press, Oxford (1997)
12. Greco, S., Matarazzo, B., Słowiński, R.: Rough set theory for multicriteria decision analysis. *Eur. J. Oper. Res.* **129**(1), 1–47 (2001)
13. Greco, S., Matarazzo, B., Słowiński, R.: Dominance-based rough set approach as a proper way of handling graduality in rough set theory. *Trans. Rough Sets* **7**, 36–52 (2007)
14. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. *J. Mach. Learn. Res.* **3**, 1157–1182 (2003)
15. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.: The WEKA data mining software: an update. *SIGKDD Explor.* **11**(1), 10–18 (2009)
16. Jelonek, J., Krawiec, K., Stefanowski, J.: Comparative study of feature subset selection techniques for machine learning tasks. In: Proceedings of the 7th Workshop on Intelligent Information Systems (1998)
17. Jensen, R., Shen, Q.: *Computational Intelligence and Feature Selection*. Wiley, Hoboken (2008)
18. John, G., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In: Cohen, W., Hirsh, H. (eds.) *Machine Learning: Proceedings of the 11th International Conference*, pp. 121–129. Morgan Kaufmann Publishers (1994)
19. Kavzoglu, T., Mather, P.: Assessing artificial neural network pruning algorithms. In: Proceedings of the 24th Annual Conference and Exhibition of the Remote Sensing Society, pp. 603–609. Greenwich (2011)
20. Khmelev, D., Tweedie, F.: Using Markov chains for identification of writers. *Lit. Linguist. Comput.* **16**(4), 299–307 (2001)
21. Kingston, G., Maier, H., Lambert, M.: A statistical input pruning method for artificial neural networks used in environmental modelling. In: *Transactions of the 2nd Biennial Meeting of the International Environmental Modelling and Software Society*, pp. 87–92. Osnabruēck, Germany (2004)
22. Kohavi, R., John, G.: Wrappers for feature subset selection. *Artif. Intell.* **97**(1–2), 273–324 (1997)

23. Lal, T., Chapelle, O., Weston, J., Elisseeff, E.: Embedded methods. In: Guyon, I., Nikravesh, M., Gunn, S., Zadeh, L. (eds.) *Feature Extraction: Foundations and Applications*. Studies in Fuzziness and Soft Computing, vol. 207, pp. 137–165. Springer, Berlin (2006)
24. Lynam, T., Clarke, C., Cormack, G.: Information extraction with term frequencies. In: Proceedings of the Human Language Technology Conference, pp. 1–4. San Diego (2001)
25. Moshkov, M., Piliszczuk, M., Zielosko, B.: On partial covers, reducts and decision rules with weights. *Trans. Rough Sets* **6**, 211–246 (2006)
26. Moshkov, M., Skowron, A., Suraj, Z.: On covering attribute sets by reducts. In: Kryszkiewicz, M., Peters, J., Rybinski, H., Skowron, A. (eds.) *Rough Sets and Emerging Intelligent Systems Paradigms. LNCS (LNAI)*, vol. 4585, pp. 175–180. Springer, Berlin (2007)
27. Novaković, J., Strbac, P., Bulatović, D.: Toward optimal feature selection using ranking methods and classification algorithms. *Yugosl. J. Oper. Res.* **21**(1), 119–135 (2011)
28. Pawlak, Z.: Computing, artificial intelligence and information technology: rough sets, decision algorithms and Bayes' theorem. *Eur. J. Oper. Res.* **136**, 181–189 (2002)
29. Pawlak, Z.: Rough sets and intelligent data analysis. *Inf. Sci.* **147**, 1–12 (2002)
30. Peng, R.: Statistical aspects of literary style. Bachelor's Thesis, Yale University (1999)
31. Peng, R., Hengartner, H.: Quantitative analysis of literary styles. *Am. Stat.* **56**(3), 15–38 (2002)
32. Sikora, M.: Rule quality measures in creation and reduction of data rule models. In: Greco, S., Hata, Y., Hirano, S., Inuiuguchi, M., Miyamoto, S., Nguyen, H., Słowiński, R. (eds.) *Rough Sets and Current Trends in Computing. Lecture Notes in Computer Science*, vol. 4259, pp. 716–725. Springer (2006)
33. Słowiński, R., Greco, S., Matarazzo, B.: Dominance-based rough set approach to reasoning about ordinal data. *LNCS (LNAI)* **4585**, 5–11 (2007)
34. Stańczyk, U.: Dominance-based rough set approach employed in search of authorial invariants. In: Kurzyński, M., Woźniak, M. (eds.) *Computer Recognition Systems 3. AISC*, vol. 57, pp. 315–323. Springer, Berlin (2009)
35. Stańczyk, U.: DRSA decision algorithm analysis in stylometric processing of literary texts. In: Szczuka, M., Kryszkiewicz, M., Ramanna, S., Jensen, R., Hu, Q. (eds.) *Rough Sets and Current Trends in Computing. LNCS (LNAI)*, vol. 6086, pp. 600–609. Springer, Berlin (2010)
36. Stańczyk, U.: Rough set-based analysis of characteristic features for ANN classifier. In: Grana Romay, M., Corchado, E., Garcia-Sebastian, M. (eds.) *Hybrid Artificial Intelligence Systems Part 1. LNCS (LNAI)*, vol. 6076, pp. 565–572. Springer, Berlin (2010)
37. Stańczyk, U.: On performance of DRSA-ANN classifier. In: Corchado, M., Kurzyński, E., Woźniak, M. (eds.) *Hybrid Artificial Intelligence Systems Part 2. LNCS (LNAI)*, vol. 6679, pp. 172–179. Springer, Berlin (2011)
38. Stańczyk, U.: Rule-based approach to computational stylistics. In: Bouvry, P., Kłopotek, M., Marciniak, M., Mykowiecka, A., Rybiński, H. (eds.) *Security and Intelligent Information Systems. LNCS (LNAI)*, vol. 7053, pp. 168–179. Springer, Berlin (2012)
39. Stańczyk, U.: On preference order of DRSA conditional attributes for computational stylistics. In: Decker, H., Lhotska, L., Link, S., Basl, J., Tjoa, A. (eds.) *Database and Expert Systems Applications. LNCS*, vol. 8056, pp. 26–33. Springer, Berlin (2013)
40. Stańczyk, U.: Relative reduct-based estimation of relevance for stylometric features. In: Catañia, B., Guerrini, G., Pokorny, J. (eds.) *Advances in Databases and Information Systems. LNCS*, vol. 8133, pp. 135–147. Springer, Berlin (2013)
41. Stańczyk, U.: Rough set and artificial neural network approach to computational stylistics. In: Ramanna, S., Howlett, R., Jain, L. (eds.) *Emerging Paradigms in Machine Learning, Smart Innovation, Systems and Technologies*, vol. 13, pp. 441–470. Springer, Berlin (2013)
42. Stańczyk, U.: Weighting of attributes in an embedded rough approach. In: Gruca, A., Czachórski, T., Kozielski, S. (eds.) *Man-Machine Interactions. AISC*, vol. 242, pp. 475–483. Springer, Berlin (2013)
43. Sun, Y., Wu, D.: A RELIEF based feature extraction algorithm. In: *Proceedings of the SIAM International Conference on Data Mining*, pp. 188–195 (2008)

## Chapter 4

# A Geometric Approach to Feature Ranking Based Upon Results of Effective Decision Boundary Feature Matrix

Claudia Diamantini, Alberto Gemelli and Domenico Potena

**Abstract** This chapter presents a new method of *Feature Ranking* (FR) that calculates the *relative weight of features* in their original domain with an algorithmic procedure. The method supports information selection of real world features and is useful when the number of features has costs implications. The *Feature Extraction* (FE) techniques, although accurate, provide the weights of artificial features whereas it is important to weight the real features to have readable models. The accuracy of the ranking is also an important aspect; the *heuristics methods*, another major family of ranking methods based on generate-and-test procedures, are by definition approximate although they produce readable models. The ranking method proposed here combines the advantages of older methods, it has at its core a feature extraction technique based on *Effective Decision Boundary Feature Matrix* (EDBFM), which is extended to calculate the total *weight* of the real features through a procedure geometrically justified. The modular design of the new method allows to include any FE technique referable to the EDBFM model; a thorough benchmarking of the various solutions has been conducted.

**Keywords** Feature ranking · Feature weight · Effective decision boundary feature matrix · Classification

## 4.1 Introduction

The recent developments of information technology dramatically increased the capability of gathering information. This information is described by a high number of attributes, observations or measures, generically called *features*. On the one hand this

---

C. Diamantini · A. Gemelli (✉) · D. Potena

Dipartimento di Ingegneria Dell'Informazione, Università Politecnica Delle Marche,  
via Brecce Bianche, 60131 Ancona, Italy  
e-mail: albertogemelli@hotmail.com

C. Diamantini  
e-mail: diamantini@dii.univpm.it

D. Potena  
e-mail: potena@dii.univpm.it

improves our ability to study real phenomena, but on the other hand huge amounts of data produce an “informative overload”, raising data acquisition and processing costs without effective exploitation of information. What is more, most of machine learning techniques suffer from the so called “curse of dimensionality” effect, and human interpretation of models generated by these techniques can be difficult on high dimensional spaces. To address these issues, the adoption of *Feature Selection* (FS) in processes is observing increasing interest and expansion.

Decision making and operations in the modern production contexts require a FS method which is generally valid for all applications, therefore robust and flexible, able to operate interactively in a dynamic information environment, dealing effectively with challenges posed by data heterogeneity, data bandwidth and real-time requirements. The large availability of information represents also a challenge because of the exponential growth of data acquisition costs and, last but not least, energy consumption by computers and acquisition sensor systems. The FS process represents a complex decisional mechanism in which the accuracy of results is equally important as usability, fastness, robustness and scalability. In the scientific literature, the current approaches to FS in the machine learning process show distinct solutions which address specific issues and highlight opposite vintages, though many practical issues have arisen around applications in productive contexts that have never been considered on the whole. This is the context that inspires the invention and validation of our novel Feature Ranking (FR) method that supports the FS. This chapter proposes an innovative approach to FR that detains vintages otherwise dispersed over a variety of distinct methods. Our research is articulated over two main objectives, the first is to obtain feature ranking leading to high accuracy in machine learning goals achievement, the second is to provide an algorithm capable to actively consider cost functions in supporting decision making. These issues have been studied in relation to a machine learning process among the most known: the classification.

## 4.2 Feature Ranking for Classification: The Background Picture

### 4.2.1 *Intrinsic Discriminant Dimension of a Classification Task*

In the literature FS refers to the problem of selecting a subset of relevant features for building robust learning models [19, 27]. The concept of *optimal feature subset* has been refined during the years by the comprehension of the dataset properties that condition the classification performance. As it happens in generic data collections, many of the features are insignificant to reach a learning objective. A definition of *relevant feature* is provided by [3]: a feature  $x_i$  is strongly relevant to dataset X if there exist examples A and B in X that differ only in their assignment to  $x_i$  and have different labels. A feature  $x_i$  is weakly relevant to classification accuracy if it is possible to remove a subset of the features so that becomes strongly relevant.

In a classification task, the FS is used to predict the so called *intrinsic discriminant dimension* of the dataset, which has been defined by Lee and Landgrebe [24] as the smallest dimensional subspace wherein the same classification accuracy can be obtained as could be obtained in the original space. Effects of FS on accuracy have more recently been studied by Sima et al. [34]. In [21, 35], the problem of FS is seen as trade-off between generalization and specialization or, equivalently, a trade-off between bias and variance of the inductive process. A classification algorithm partitions the instance space into regions; when the number of features is relatively small, regions are too large, that causes the partitioning of the instances to be poor in terms of generalization and therefore accuracy decreases, this phenomenon is called *bias*. When the number of features is high, the probability that individual regions are labeled with the wrong class is increased too. This effect is called *variance*. Decision tree and neural network classifiers are particularly sensible to variance. There emerges the concept of irrelevant/redundant features that might cause the classification algorithms loosing efficiency and accuracy, whereas the subset of features that improves the performance of learning algorithms is defined *optimal subset*. All the aspects of the learning algorithm sensitivity to the dataset dimensionality, have been generally named as the *curse of dimensionality* by Kira and Rendell [20].

The optimal subset can be detected on a feature evaluation function [8]. When doing classification, an *Evaluation Function* (EF) expresses for each feature subset its ability to discriminate between classes. The effectiveness of the EF in highlighting the relative importance of feature depends on the search strategy by which the space of all possible subsets is explored, and it has measurable properties: accuracy (how accurate is the prediction of the EF), generality (how suitable is the EF for different classifiers) and time complexity (time taken to calculate the EF). A selection based on classification accuracy can be considered effective if the classifier error rate does not significantly decrease after selection. The authors indicate the *INN classifier* as a convenient algorithm to build the evaluation function since it appears to always provide a reasonable classification performance in most applications.

#### 4.2.2 Classical Feature Selection Strategies

The FS process is divided generally into two phases: FR and FS in the strict sense. It is necessary to rank the relative importance of features before proceeding to an optimal selection and then learning a classification model, although these two phases can be integrated in different modes as it will be discussed in this section. The progress in scientific research almost coincides for ranking and selection. As in the survey of [2, 15], the FS methods are categorized in two main categories: (i) methods that explore the space of possible subsets, searching an optimal subset of features by using heuristics to limit computational complexity, (ii) methods that rank features individually based on properties that good features are presumed to have, such as their contribution to class separability. In the classification learning process, input dataset is arranged in a  $n$  by  $m$  matrix where each row, or record, represents an

object belonging to a class, and each column represents a characterizing feature. In a geometric sight, the objects can be thought as points positioned in an  $m$ -dimensional space of features. The solution to a problem of classification can be thought as the procedure that finds the hyperplanes that, in the feature space, separate the classes of points.

A broad group of FS techniques is based on the construction and ranking of new features [11, 16]. The *Feature Extraction* (FE) process is based on a transformation of the original set of real features by a linear combination of these, by which the power to discriminate among classes is concentrated on a reduced number of extracted features. The relevancy of each individual feature is evaluated, in fact the set of *eigenvalues*, always associated with the transformation process, represents the relative relevancy of each extracted feature and allows ranking them. It is important to notice, however, that FE methodology was conceived primarily to do data compression, therefore it effectively reduces the size of the initial volume of data, but it implies the entire dataset to be available to construct each extracted feature; clearly the FE approach is of no help in an application where the containment of data acquisition costs is important. Furthermore the FE model is very application specific since extracted features are uniquely associated with a dataset.

For FS, three modes of application have been identified by [6, 16, 29] in relation to the dependence on the classification algorithm: in the *wrapper* mode, selection and classification are iterated to refine the selection of features up to achieving an optimal performance of the classification algorithm. The exploration of the solution space can be conducted either with the brute force or the heuristic approaches. The wrapper mode is supervised and is not suitable for applications in real-time, although some solutions have been proposed that increase its performance whilst avoiding its procedural complexity [28]. By contrast, in the *filter* mode the features that respond to a general criterion of relevancy for a classification process are selected. The filter method is applied in a unique step independently of the classification algorithm. In the *Embedded* mode, FS is part of the model training process, and features relevancy is obtained by analyzing their utility for optimizing the objective function of the learning model; an application example is in [30]. From a productive point of view these three methods represent different levels of trade-off between ease of execution and accuracy of the results.

When *heuristic* methods are used in feature selection the search of the optimal subset is done by attempts, by which there is built an evaluation function that provides for each subset of real features its ability to discriminate between classes [8, 36]. The results depend sensibly on the heuristic adopted and the amount of points effectively explored of the solution space. Because of the underlying subjective assumption, the heuristic approach is not fully reliable [1, 33], however it has the vintage to produce a rank model for real world features, therefore retaining a human interpretability. Among the heuristic strategies we would like to describe briefly the following: *Gain Ratio*, *One-Rule* and *Relief-F*.

The *Gain Ratio* algorithm [32] uses information entropy to find out how well a feature separates instances. The goodness of each individual feature depends of how broadly and uniformly it splits the considered data. Features are sorted from the most

relevant (the one with the highest gain ratio) to the least relevant (the one with the lowest gain ratio). Then, a decision tree is created starting with the most relevant feature. This method is computationally efficient because it tests at most a number of cases equal to the number of features. The danger is that if none of the features is significantly better than the others then the method may fail to find a good subset, by contrast if there is a strongly relevant feature the method gives reasonably good results.

The *One-Rule* algorithm [18] ranks the attributes according to the error rate. This method is sensibly affected by overfitting.

The *Relief* algorithm uses a nearest-neighbor approach [20]. The algorithm updates iteratively a *relevance vector* of length equal to the number of features, initially set to zero. In a two-class problem, for a randomly chosen sample, one nearest point is chosen in the same class and one in the opposite class. The squared component distances of these two closest examples are component-wise subtracted from (or added to) the relevance vector depending on whether the closest example was in the same (or different) class. This procedure is repeated for  $m$  (a given parameter) times, and those features whose relevance weight, thus computed, are above a certain threshold are selected. An improvement of the basic algorithm is *Relief-F* [23] that uses  $M$ , instead of just one, nearest hits and ensures greater robustness of the algorithm against noise.

The development in scientific research currently focuses on topics related to *data explosion* phenomenon such as FS for ultrahigh dimensional data [30], and multi-source FS [38]. In [13] there is a case study on feature selection techniques applied to geographic information systems and geospatial decision support, an application domain where the growing availability of data poses several challenges along with important perspectives. There is a growing interest to consider the FS as something more than just a routine to improve machine learning accuracy; the FR model is by itself a knowledge model holding important semantic aspects of the information environment. There have been attempts to further enrich the concept of relevant feature with semantic meanings, such as the contribution of a feature to the knowledge of the physical process underlying the generation of the data. The usefulness of the FR in selecting the variables for modelling dynamic systems has been studied in [5]. A *causal feature selection* is proposed in [17], where the FS is driven by the detection of cause-effect relationships observed in time. This kind of selection process explicitly associates the concept of relevant feature with the concept of control variable. One step forward to the contribution of FS to the modelling of a real system is provided by [12, 33], which in the selection process take into account the interaction of features, acknowledging the fact that features exhibit group properties that cannot be detected on individual features, as they were actual components of a system. More recently there have been attempts to integrate the FS with preexisting basis of knowledge such as ontologies and association rules [7].

### 4.2.3 A Multiple-Challenge Case Study for Feature Ranking

The issues identified in the previous sections have been dealt in the scientific literature in separate ways, but in reality they constitute a complex of challenges to be addressed in an integrated manner, especially when pursuing goals of efficiency and effectiveness as it is in real applications and in production environments. The problem on which we focus our interest is to obtain a new model for ranking features which combines effective FE methods to a representation model that is humanly understandable and can be integrated in domain knowledge. It is also an objective to explore how generalizable is the efficacy of this new method and how it benefits from a modular architecture that allows to choose between alternative methods of feature extraction depending on restrictions imposed by specific applications. In order to compare the quality of the new model, and its possible variants, to the classical methods it is necessary to identify suitable performance metrics and a benchmarking methodology that uses reference datasets. At the same time there has to be explored the possibility to obtain cost-benefit functions of the features for use in decision-making.

## 4.3 Focus on Feature Extraction Based Ranking

### 4.3.1 Linear Models

Many known techniques of Feature Extraction (FE) differ in the principle underlying the detection of an optimal new set of features. However, all of them show an underlying unity in the calculation of geometric transformation, algebraically expressed as projection (or mapping) matrix.

In *Linear Discriminant Analysis* (LDA), where a linear separability of classes is assumed, the principle underlying the detection of a new feature is that of maximizing the ratio of the *between-class variance* to the *within-class variance* on this feature. Therefore a set of new features are obtained by maximizing the ratio of the between-class covariance matrix  $\mathbf{S}_b$  to the within-class covariance matrix  $\mathbf{S}_w$ . The projection matrix is the eigenvector matrix  $\mathbf{U}$  obtained by solving the generalized eigenvalue problem:  $\mathbf{S}_b \cdot \mathbf{U} = \mathbf{S}_w \cdot \mathbf{U} \cdot \Lambda$ , where  $\Lambda$  is a diagonal matrix whose entries are the *eigenvalues* of  $\mathbf{U}$ . Each eigenvalue  $\lambda_i$  measures the relative capability of each new feature  $\mathbf{u}_i$  of separating classes.

A limitation of the classic LDA algorithm is that both  $\mathbf{S}_w$  and  $\mathbf{S}_b$  matrices must be non-singular in order to preserve the orthonormality of the mapping. For this reason several variants of the classic algorithm have been proposed in order to overcome the singularity problem. In particular in this work, we consider the *Orthogonal Linear Discriminant Analysis* (OLDA) algorithm [37]. This algorithm uses *Singular Value Decomposition* to obtain a non-singular approximation of  $\mathbf{S}_w^{-1} \cdot \mathbf{S}_b$ . When  $\mathbf{S}_w$  and  $\mathbf{S}_b$  matrices are non-singular, OLDA and classic LDA give identical results.

### 4.3.2 Feature Extraction Based on Decision Boundary

Another family of FE techniques is based on the properties of decision border [10]. Classes are statistically characterized by the *class-conditional probability density function* (cpdf)  $p_{X|Y}(\mathbf{x}|y_i)$ , where the continuous random vector  $X$  takes values in  $\mathcal{R}^N$  and the discrete random variable  $Y$  takes value in  $y$ . The cumulative probability density function of the random vector  $X$  is:

$$p_X(\mathbf{x}) = \sum_{i=1}^C P_Y(y_i)p_{X|Y}(\mathbf{x}|y_i), \quad (4.1)$$

where  $P_Y(y_i)$  is the a-priori probability of class  $y_i$ .

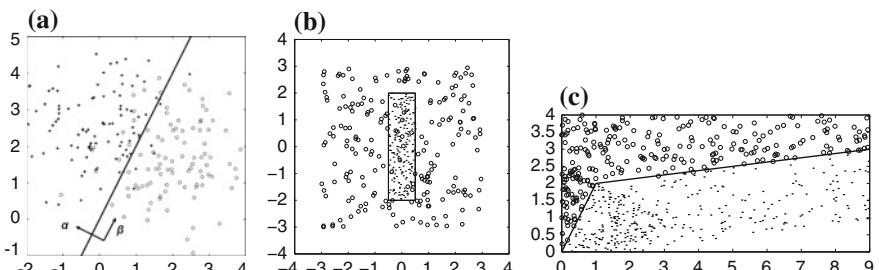
Therefore, a classification or decision rule is a mapping  $\Psi : \mathcal{R}^N \rightarrow Y$  that assigns a class label to data on the basis of the observation of its feature vector. A classification rule determines a partition of the feature space in  $C$  *decision regions*  $D_1, \dots, D_C$  such that  $D_i = \{\mathbf{x} \in \mathcal{R}^N \mid \Psi(\mathbf{x}) = y_i\}$ . The boundary separating decision regions is called the *decision boundary*. Figure 4.1 illustrates an example of decision rule for two Gaussian classes (symbolized by ‘\*’ and ‘o’). The straight line represents the decision boundary: all points at the left of it are assigned by the decision rule to ‘\*’ class, and those at the right to ‘o’ class.

Among all possible classification rules, the rule achieving the minimum *error probability*

$$\varepsilon = \int \sum_{y_i \neq \Psi(\mathbf{x})} p(\mathbf{x}|y_i)P(y_i)d\mathbf{x} \quad (4.2)$$

is the Bayes rule  $\Psi_B(\mathbf{x}) = \arg \text{MAX}_{y_i}[p(\mathbf{x}|y_i)P(y_i)]$ . The corresponding decision boundary is consistently called *Bayes boundary*, which is the theoretically optimal solution that every classification method aims to achieve.

The geometry of the decision boundary has been used in the discriminative feature extraction approach known as *Decision Boundary Feature Extraction* (DBFE) [25]



**Fig. 4.1** Examples of two-classes classification problems in a 2-dimensional space. **a** Linear boundary.  $\alpha$  and  $\beta$  represent the informative direction and the redundant direction respectively, **b** Closed boundary, **c** Piecewise linear boundary

to recognize those informative features allowing to achieve the same classification accuracy as in the original space. The basic idea of DBFE is that moving along the direction of the decision boundary, the classification of each observation will remain unchanged (see Fig. 4.1a). Hence, the direction of the decision boundary is redundant. In contrast, while moving along the direction normal to the decision boundary the classification changes, hence it represents an informative direction. Moreover, the effectiveness of a direction is directly proportional to the area of decision boundary with the same normal vector. To discuss this statement, consider Fig. 4.1b. There, the border is a rectangle parallel to the axes, so the informative directions defined by normal vectors to the border are the  $x$  and  $y$  axes themselves. Although both directions are informative, it is simple to see that the  $x$ -axis is more important since projecting data on it results in less class overlapping than projecting data on the  $y$ -axis.

The idea is formalized by the notion of *Effective Decision Boundary Feature Matrix* (EDBFM):

$$\Sigma_{EDBFM} = \frac{1}{\int_{S'} p(\mathbf{x}) d\mathbf{x}} \int_{S'} \mathbf{N}^T(\mathbf{x}) \mathbf{N}(\mathbf{x}) p(\mathbf{x}) d\mathbf{x}, \quad (4.3)$$

where  $\mathbf{N}(\mathbf{x})$  is the normal vector at a point  $\mathbf{x}$ ,  $\mathbf{N}^T(\mathbf{x})$  denotes the transposed normal vector and  $S'$  is the portion of decision boundary containing most of the training data (the effective decision boundary). It has been proved [25] that:

- the rank of the EDBFM represents the *intrinsic discriminant dimension*, that is the minimum number of feature vectors needed to achieve the same Bayes error probability as in the original space;
- the eigenvectors of EDBFM corresponding to nonzero eigenvalues are the necessary feature vectors.

In order to construct a Bayes decision border, in [25] there has been proposed *SVM Decision Boundary Analysis*, a method that combines DBFE principle and Support Vector Machine algorithm. In [14] the use of *Analytical Decision Boundary Feature Extraction* (ADBFE) is introduced, where the normal vectors are calculated analytically from the equations of the decision border. All methods produce an EDBFM that represents a data projection matrix onto a new feature space.

## 4.4 Feature Ranking Based on Effective Decision Boundary Feature Matrix

### 4.4.1 Geometric Considerations

As it has been introduced in previous sections, it is desirable to obtain a ranking of real features on the basis of information contained in EDBFM. The idea is intuitively

explained by referring again to the examples in Fig. 4.1. Let us consider decision boundaries formed by a unique line, like line  $\beta$  in Fig. 4.1a. In these cases none of the features is redundant, however it is apparent that the relevance of a feature can be stated in terms of the line slope. In order to apply the DBFE method, let us observe that the decision boundary has the form  $y = mx + k$ , hence the normal vector is  $N = [m, -1]$ . The calculus of equation (4.3) is straightforward since the normal vector is constant along  $S'$  and the equation becomes:

$$\Sigma_{EDBFM} = \frac{\mathbf{N}^T \mathbf{N} \int_{S'} p(\mathbf{x}) d\mathbf{x}}{\int_{S'} p(\mathbf{x}) d\mathbf{x}} = \mathbf{N}^T \mathbf{N} = \begin{pmatrix} m^2 & -m \\ -m & 1 \end{pmatrix}. \quad (4.4)$$

Eigenvalues and related eigenvectors are  $\lambda_1 = 0$ ,  $\lambda_2 = m^2 + 1$ ,  $v_1 = [1, m]$ ,  $v_2 = [-m, 1]$ , and only the second eigenvector  $v_2$  is the informative direction. In this case the eigenvector components define the relevance of the real features. For instance, when  $m = 0$  (boundary parallel to the  $x$ -axis) the only informative real feature is the  $y$ -axis, when  $m = 1$  (boundary  $y = x$ ) both features are equally important, finally as  $m \rightarrow \infty$  (boundary tends to the  $y$ -axis) the relevance of  $x$ -axis grows. As a second case, let us consider the border in Fig. 4.1b. In this case, *cpdfs* are taken constant along the boundary and EDBFM is

$$\Sigma_{EDBFM} = \begin{pmatrix} 8 & 0 \\ 0 & 2 \end{pmatrix},$$

with  $\lambda_1 = 8$ ,  $\lambda_2 = 2$ ,  $v_1 = [1, 0]$ ,  $v_2 = [0, 1]$ . This case is somewhat complementary to the former: now, since new features coincide with the real ones, the relevance of the latter is fully expressed by eigenvalues. From the analysis of these two cases we can derive that in the DBFE approach the eigenvector components represent the weight of every real feature locally to the new feature, whereas the eigenvalues represent the discriminative power of each new feature. Hence we can combine these two characteristics in order to define a global ranking of the real features as it is in the objective of the present work. Firstly eigenvectors are weighted by multiplying them by the respective eigenvalues, and then the corresponding components of weighed eigenvectors are summed (in the absolute values). Resulting values are the individual contributions (or weights) of every real feature into the transformation, and represent the discriminative power of each real feature and its relative position in a *rank model*.

Formally, let  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_N$  be the eigenvectors of the EDBFM matrix,  $\lambda_1, \lambda_2, \dots, \lambda_N$  the corresponding eigenvalues, and  $u_{ij}$  the  $j$ th component of the eigenvector  $\mathbf{u}_i$ . The weights of real features are computed as follows:

$$w_j = \sum_{i=1}^N \lambda_i |u_{ij}|, j = 1, \dots, N, \quad (4.5)$$

$w_j > w_k \Rightarrow$  feature  $f_j$  is more important than  $f_k$ .

As a numeric example, let us consider Fig. 4.1c. The equation of the border is  $y = 2x$  for  $x \in [0, 1]$ ,  $y = x/8 + 15/8$  for  $x \in [1, 9]$ . The *cpdfs* are taken constant along the boundary. It turns out that

$$\Sigma_{EDBFM} = \begin{pmatrix} 1.913 & -1.887 \\ -1.887 & 8.385 \end{pmatrix},$$

$\lambda_1 = 1.4$ ,  $\lambda_2 = 8.89$ ,  $v_1 = [0.965, 0.261]$ ,  $v_2 = [-0.261, 0.965]$ . The ranking method leads to the following weights:  $w_1 = 3.68$ ,  $w_2 = 8.95$ , hence the real feature  $y$  turns out to be more discriminant than  $x$  as the figure suggests, since the first piece of boundary is shorter than the second one which is almost parallel to the  $x$ -axis.

#### 4.4.2 The Algorithm

The presented method is based on the calculus of the EDBFM, which in turn needs the knowledge of the decision boundary. In order to apply it to real cases, where the decision boundary, as well as *cpdfs* are typically unknown, non-parametric approaches will be considered. In non-parametric approaches we are given a set of instances of the true phenomenon (training data) only, and no assumption on the form of *cpdfs* is made. In this work we propose the use of *Labeled Vector Quantizer* (LVQ) architectures and the *Bayes Vector Quantizer* (BVQ) learning algorithm. The reason for the choice of BVQ is twofold: (1) it has demonstrated to drive an LVQ toward a (locally) optimal approximation of the Bayes boundary [10]; (2) the approximation is piecewise linear, thus simplifying the calculus of the normal vectors.

An Euclidean nearest neighbor Vector Quantizer (VQ) of dimension  $N$  and order  $Q$  is a function  $\Omega : \mathcal{R}^N \rightarrow \mathcal{M}$ ,  $\mathcal{M} = \{\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_Q\}$ ,  $\mathbf{m}_i \in \mathcal{R}^N$ ,  $\mathbf{m}_i \neq \mathbf{m}_j$ , which defines a partition of  $\mathcal{R}^N$  into  $Q$  regions  $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_Q$ , such that

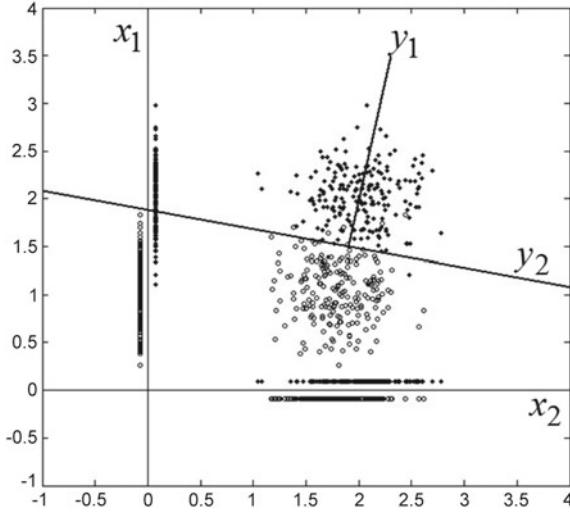
$$\mathcal{V}_i = \{\mathbf{x} \in \mathcal{R}^N : \| \mathbf{x} - \mathbf{m}_i \|^2 < \| \mathbf{x} - \mathbf{m}_j \|^2, j \neq i\}. \quad (4.6)$$

Elements of  $\mathcal{M}$  are called *code vectors*. The region  $\mathcal{V}_i$  defined by (4.6) is called the *Voronoi region* of the code vector  $\mathbf{m}_i$ . Note that the Voronoi region is completely defined by  $\mathcal{M}$ . In particular, the boundary of Voronoi region  $\mathcal{V}_i$  is defined by the intersection of a finite set of hyperplanes  $\mathcal{S}_{i,j}$  with equation

$$(\mathbf{m}_i - \mathbf{m}_j) \cdot (\mathbf{x} - \frac{\mathbf{m}_i + \mathbf{m}_j}{2}) = 0,$$

where  $\mathbf{m}_j$  is a neighbor code vector to  $\mathbf{m}_i$ . The definition of normal vectors to these hyperplanes is thus straightforward and it is  $\mathbf{N}_{ij} = \mathbf{m}_i - \mathbf{m}_j$  (see Fig. 4.2).

By associating with each code vector a class we can define a decision rule. A Labeled Vector Quantizer (LVQ) is a pair  $LVQ = < \Omega, \mathcal{L} >$ , where  $\Omega : \mathcal{R}^N \rightarrow \mathcal{M}$  is a vector quantizer, and  $\mathcal{L} : \mathcal{M} \rightarrow \dagger$  is a labeling function, assigning to each



**Fig. 4.2** A piece of true decision boundary, its linear approximation and the local discriminative direction  $\mathbf{N}_{ij} = \mathbf{m}_i - \mathbf{m}_j$

code vector in  $\mathcal{M}$  a class label. The classification rule associated with an LVQ is:

$$\Psi_{LVQ} : \mathcal{R}^N \rightarrow y, \mathbf{x} \mapsto \mathcal{L}(\mathcal{Q}(\mathbf{x})).$$

Note the Nearest Neighbor nature of this classification rule: each vector in  $\mathcal{R}^N$  is assigned to the same class as its nearest code vector. Thus, decision regions are defined by the union of Voronoi regions of code vectors with the same label. Note also that the decision boundary is defined only by those hyperplanes  $\mathcal{S}_{i,j}$  such that  $\mathbf{m}_i$  and  $\mathbf{m}_j$  have different labels.

An LVQ can be trained to find an approximation of the Bayes boundary. LVQ training algorithms have been originally proposed by Kohonen [22]. Here we use a more recent algorithm known as Bayes VQ (BVQ), formally defined as a gradient descent algorithm for the minimization of the error probability. It strongly resembles Kohonen's LVQ2.1, however, formal derivation introduces also some modifications that improve performances and robustness. The BVQ algorithm is an iterative punishing-rewarding adaptation schema. At each iteration, the algorithm considers a sample randomly picked from the training set. If the sample turns out to fall “on” the decision boundary, then the position of the two code vectors determining the boundary is updated, moving the code vector with the same label of the sample towards the sample itself and moving away that with a different label. Since the decision boundary is a null measure subspace of the feature space, we have zero probability to get samples falling exactly on it. Thus, an approximation of the decision boundary is made, considering those samples falling close to it. Due to lack of space we cannot report the BVQ algorithm here. The algorithm is described in [9].

Having a trained LVQ, the calculus of the feature rank is straightforward and is given by the following BVQ-based Feature Ranking (BVQ-FR) Algorithm 1.

**Algorithm 1** BVQ-FR algorithm

- 
- 1: Train the LVQ  $\{(\mathbf{m}_1, l_1), \dots, (\mathbf{m}_Q, l_Q)\}$ ,  $\mathbf{m}_i \in \mathcal{R}^N$ ,  $l_i \in y$  by using the BVQ algorithm;
  - 2: Set the elements of the matrix  $\Sigma_{BVQFM}$  to 0;
  - 3:  $w_{tot} = 0$ ;
  - 4: For each training sample  $t_k$ 
    - 1: Find the two code vectors  $m_i, m_j$  nearest to  $t_k$ ;
    - 2: If  $l_i \neq l_j$  and  $t_k$  falls at a distance less than  $\Delta$  from the border  $S_{ij}$  then
      - 1: Calculate the unit normal vector to the decision boundary as:  $\mathbf{N}_{ij} = \frac{(\mathbf{m}_i - \mathbf{m}_j)}{\|\mathbf{m}_i - \mathbf{m}_j\|}$ ;
      - 2:  $\Sigma_{BVQFM} = \Sigma_{BVQFM} + \mathbf{N}_{ij}^T * \mathbf{N}_{ij}$ ;
      - 3:  $w_{tot} = w_{tot} + 1$ ;
  - 5:  $\Sigma_{BVQFM} = \frac{\Sigma_{BVQFM}}{w_{tot}}$ ;
  - 6: Calculate eigenvectors  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_N$  and related eigenvalues  $\lambda_1, \lambda_2, \dots, \lambda_N$  of  $\Sigma_{BVQFM}$ ;
  - 7: Set  $\mathbf{W} = \sum_{z=1}^N \lambda_z |\mathbf{u}_z|$ ;
  - 8: Sort features with respect to  $\mathbf{W}$  components.
- 

The core of the BVQ-FR algorithm is at point 4. There, finding the two nearest code vectors to each training sample allows us to define the effective decision boundary of the LVQ. As a matter of fact, testing whether labels are different guarantees that the piece of Voronoi boundary  $S_{ij}$  is actually a part of the decision boundary. Secondly, incrementing the  $\Sigma_{BVQFM}$  each time a pair of code vectors is selected, allows to weight the normal vector  $\mathbf{N}_{ij}$  by the number of samples falling at a distance less than  $\Delta$  from  $S_{ij}$ . It can be proved that this is equivalent to a Parzen estimate of the integral  $\int_{S_{ij}} p(\mathbf{x})$ , while the final value of  $w_{tot}$  represents the Parzen estimate of  $\int_{S'} p(\mathbf{x})$  in Eq. (4.3) [10].

It should be noted that the algorithm BVQ-FR can be transformed by replacing BVQ with other FE algorithm that produces a transformation matrix EDBFM-like. For example there can be used OLDA, SVM and ADBFE algorithms. In the next section an experimental comparison between these alternatives will be made.

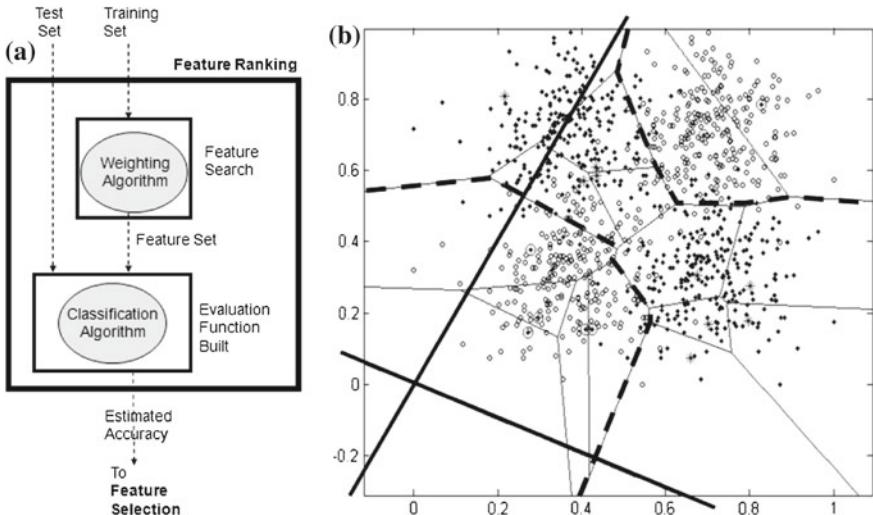
## 4.5 Experiments

### 4.5.1 Experimental Setting

This section is devoted to experimental evaluation of the EDBFM-based feature weighting method. In particular in the present subsection we propose a synthetic experiment which allows us to illustrate the properties of the method. We also describe both the experimental procedure and the evaluation criteria that will be used for all subsequent experiments. In the next subsection various implementations of the method will be tested over real-world datasets and compared with well-known feature weighting algorithms.

As synthetic experiment we draw a dataset from a 22 dimensions two-class problem. The first two dimensions are drawn from the classical XOR problem, while the remaining 20 dimensions are drawn from the normal distribution. The first two dimensions are useful to classify the two classes (i.e. informative dimensions), while the remaining dimensions are noise. The dataset contains 1,000 samples equally distributed over the two classes. In this experiment, as well as in all experiments of the following subsection, we followed a 10-fold cross-validation procedure: in each fold the 90% of the samples are used to build the EDBFM matrix and to weight the original features; the remaining samples are used to evaluate the performance of the method. In particular, for each fold a weight model is calculated on an incrementing number of features taken in the rank order from the test set to extract a projection along the first informative features. Hence we firstly obtain two datasets with the most important feature, then two datasets with the first two most important features, and so forth until the full-dimensional datasets (i.e. the original ones) are returned. For each of these pairs of datasets the Nearest Neighbor algorithm is used to estimate the accuracy. After the tenth fold repetition, the weights and the accuracies are averaged by rank, and curves are built, which represent the average accuracy that the method achieves over all folds as a function of the most informative features.

The experimental work-flow is depicted in Fig. 4.3a, it consists of two phases: first the appliance of the EDBFM based ranking method to the multivariate dataset in the filter mode of [26], and then the validation procedure. The process is sketched in the following pseudocode.



**Fig. 4.3** **a** General work-flow of feature selection techniques. **b** Example of two classes classification problems. *Piecewise lines* represent the approximation of the Bayes boundary found by BVQ.  $y_1$  and  $y_2$  represent the two most important extracted features

---

**Algorithm 2** *First phase:* a FE algorithm (BVQ in this example) is applied to the training set, and then the feature ranking algorithm is executed

---

- 1: Let  $X = \{x_1, x_2, \dots, x_m\}$  be the  $m$ -dimensional normalized dataset.
  - 2: Apply the BVQ algorithm to  $X$ . Let  $Y = \{y_1, y_2, \dots, y_n\}$  be the extracted eigenfeatures.
  - 3: Compute the contributive weight  $w_i$  of each feature  $x_i$  to the eigenfeatures of  $Y$ .
  - 4: Sort the features of  $X$  such that  $x_a < x_b$  if  $w_a < w_b$ . Let  $X^s = \{x_1^s, x_2^s, \dots, x_m^s\}$  be the sorted dataset and  $m$  the rank index.
- 

**Algorithm 3** *Second phase:* on an incrementing number of features, taken in the rank order from the test set, the 1NN classification process is run and the accuracy calculated

---

- 1: The dataset  $X^s$  is input.
  - 2: Apply 1NN to whole  $X^s$ , let  $A_m$  be the returned accuracy.
  - 3: For rank  $i=1$  to  $m$  (where  $m = 22$  for this dataset):
    - let  $X_i^s = \{x_1^s, x_2^s, \dots, x_i^s\}$  be a subset of  $X^s$  with selected features up to rank  $i$ .
    - compute accuracy  $A_i$  using 1NN with 10-fold cross-validation.
- 

For the first fold, the decision boundary depicted by BVQ is reported in Fig. 4.3b, altogether with features extracted on the basis of the DBFE method.

The *BVQ* setting: Optimal values for  $\Delta$  and local region  $r$  have been found by a manually conducted search assuming the classification error rate as objective function. The parameters were fixed to  $\Delta = 0.4$  and  $r = 0.5$ ; 16 code vectors have been detected. The choice of the classification algorithm is unimportant to our purpose since we are interested only in study of the relative performance of ranking algorithms. The *1NN* is a non-parametric classifier among the simplest of all machine learning algorithms, the object is simply assigned to the class of its nearest neighbour on the basis of the Euclidean distance, it does not require settings. In [21] the *1NN* classifier is indicated as a convenient algorithm to build the evaluation function, since it appears to always provide a reasonable classification performance in most applications.

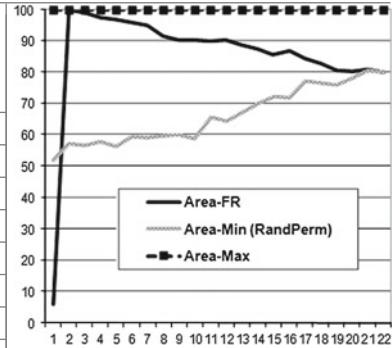
For this experiment the resulting accuracies, in the order they were calculated, are reported in Table 4.1 and plotted aside. The curve shows a steep rise which expresses the high contribution to classification accuracy by the two highest rank features. Beyond a critical point, which in this example occurs on the second feature, the curve tends to decrease because irrelevant features (low rank) are added, which only cause curse of dimensionality. By a way of comparison, a random sorting of features has been used, to which the same validation procedure is applied. The fifth column of Table 4.1 and the corresponding plot represent the average accuracy achieved by 20 different 1NN classifiers, where the features are selected according to 20 different ranks obtained by means of trivial random permutations.

As a figure of merit to characterize the performance of the ranking method we define an empirical *Performance Index*  $\phi$ :

$$\text{Performance Index}(\phi) = \frac{\text{AreaFR} - \text{AreaRP}}{\text{AreaMax} - \text{AreaRP}} \quad (4.7)$$

**Table 4.1** The features (first column) sorted by weight (second column); cumulative percentage of weight (third column); the accuracy, by subset, of EDBFM method (fourth column); the accuracy, by subset, on a random weight model (fifth column)

Rank	Weight	Weight %Cum.	1NN Acc.% Cum. Norm (BVQ-FR)	1NN Acc.% Cum. Norm (rand. rank)
Feat. 1	0.557	27.41	6.08	51.79
1 to 2	0.461	50.10	100	57.27
1 to 3	0.081	54.07	99.09	56.57
1 to 4	0.075	57.76	97.40	57.82
1 to 5	0.063	60.82	95.94	56.26
1 to 6	0.062	63.88	95.15	59.22
1 to 7	0.058	66.74	91.54	59.16
1 to 8	0.057	69.53	90.30	59.54
1 to 9	0.055	72.18	90.41	58.88
1 to 10	0.054	74.83	88.61	58.66
...	...	...	...	
1 to 22	0.035	100.00	79.93	79.93



The accuracies of fourth and fifth columns, normalized to 100 %, are also plotted aside

where *AreaFR* is the area underneath the accuracy curve relative to BVQ-FR, obtained by summing the accuracy values at each feature subset, namely the accuracy value in the BVQ-FR column of Table 4.1. Analogously *AreaRP* is the area underneath the curve obtained by random permutation of features. The *AreaMax* is the area underneath a theoretical curve reaching the 100 % possible accuracy with the top rank feature, thereafter remaining constant up to full dataset. The *AreaFR* is expected to be geometrically bounded between the other two curves, mathematically  $0 \leq \phi \leq 1$ , where  $\phi$  represents a *relative area*. When *AreaFR* approximates *AreaMax*,  $\phi = 1$ , the ranking model approximates an ideal order of the features, where the first feature is the most significant and contains all the weight to discriminate between classes. Conversely, when *AreaFR* approximates *AreaRP*,  $\phi = 0$ , the ranking model approximates to a random ordering of the features and is therefore useless.

#### 4.5.2 Benchmarking the EDBFM Ranking Method

In this section the EDBFM ranking method is tested on complex and real world datasets and the rank models are compared to other methods. Testing includes two phases:

- studying EDBFM performance when different FE algorithms are included;
- comparing EDBFM ranking and heuristic methods.

As data testbed of the experiments, 13 multivariate datasets have been considered. Eight of these datasets (*Heart*, *HeartStat*, *Australian*, *Ionosphere*, *Waveform*, *Segment*, *CoverType*, *Letter*) have been drawn from the UCI repository [31], selected for their large number of instances, classes and features as it is appropriate when testing ranking algorithms. Five more datasets (*Urban*, *Wildfire*, *Landslide*, *Corine*, *Gottigen*) have been extracted from large geographic data collections. These datasets, which include both discrete and continuous variables, are heterogeneous collections of data, excellent to challenge the selective capability of our method and to highlight the properties of the ranking model. The datasets: *Urban*, *Wildfire*, *Landslide*, *Corine* originated from the same data collection, they differ from each other by a different feature chosen as class attribute. *Urban*, *Wildfire* and *Landslide* have balanced classes, namely in these datasets all classes are represented by an equal number of instances. The geographic dataset named *Göttingen* comes from a different collection [4], its features correspond to Earth observation imagery from satellite on different wavelength band. The characteristics of all the datasets are resumed in Table 4.2, where the datasets are sorted by number of classes, then by number of features, and by number of instances. Such a sorting also represents an increasing complexity of dataset, ranging from a simple two-class perfectly balanced dataset with relatively few instances, such is the Urban, up to the Corine dataset which is a 26 class large dataset. All datasets have gone through a common preprocessing step where each feature has been normalized in the range [0; 1], to give equal importance to each feature during learning.

The first set of experiments aims to highlight how the FR algorithm performance varies when different FE built-in algorithms are used. As already mentioned, the algorithm BVQ-FR can be transformed by changing the FE algorithm,

**Table 4.2** Testbed datasets

Origin	Dataset name	# Classes	# Features	# Instances
UCI	HeartStat	2	13	270
UCI	Heart	2	13	293
UCI	Australian	2	14	690
GIS	Urban	2	18	3,972
GIS	Wildfires	2	18	5,359
GIS	Landslides	2	18	23,663
UCI	Ionosphere	2	34	351
UCI	Waveform	3	40	5,000
UCI	CoverType	7	12	58,104
UCI	Segment	7	19	2,310
GIS	Gottigen	14	8	28,083
UCI	Letter	26	16	20,000
GIS	Corine	26	18	48,379

Datasets are sorted by *number of classes*, by *number of features*, and finally by *number of instances*

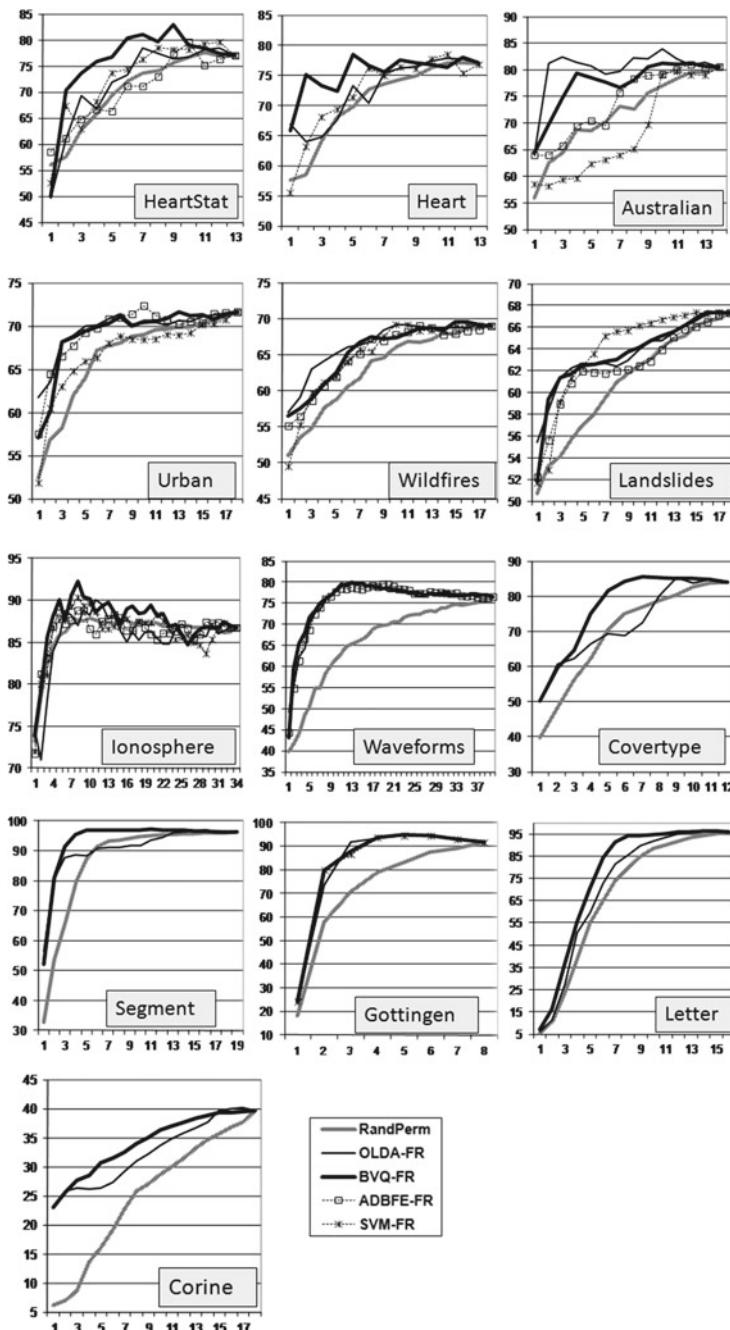
e.g. using *OLDA*, *SVM*, *ADB*. In subsequent experiments we will denote these variants respectively with the acronyms *OLDA-FR*, *SVM-FR*, *ADBFE-FR*. These algorithms, along with *BVQ-FR*, will be tested on the datasets listed above, following the experimental procedure described in the previous section. Notice in the parameter setting for *BVQ-FR*, the number of code vectors has been set to a multiple of the number of classes in the dataset, with 200000 *BVQ* iterations, whereas  $\Delta$  and  $r$  come from a manual refinement in three steps. In *SVM-FR*, we employ a *Gaussian radial basis kernel* to train the *SVM*, and we set  $r$  to 0.2.

In the Fig. 4.4, the accuracy curves are grouped by dataset to compare the performance of EDBFM Ranking algorithms. For each dataset the accuracy curve obtained by means of random permutation of features is also displayed. Notice the curves of EDBFM ranking are always located above the random ranking curve, that reveals the general efficacy of EDBFM ranking. The qualitative comparison between the curves is difficult because of the irregular pattern and their overlaps. The Performance Index  $\phi$  is of help in the analysis. In Table 4.3  $\phi$  calculated for each curve is shown. Note that missing values in the Table 4.3 are due to the impossibility to perform computationally expensive algorithms, such as *SVM* and *ADBFE*, on datasets with large number of classes and instances. We can observe in Table 4.3, where rows are sorted by increasing complexity of the dataset, *OLDA-FR* and *BVQ-FR* have, together, a dominance in the values of  $\phi$  when applied to datasets with two classes *Heart-Stat*, *Heart*, *Australian*, *Urban*, *Wildfire*, *Landslide* whereas *BVQ-FR* has a relative dominance on complex datasets *Ionosphere*, *Waveform*, *CoverType*, *Segment*, *Gottigen*, *Letter*, *Corine*. This is due to the fact that *BVQ-FR*, based on nonparametric model, has a superior performance when working on non-linearly separable classes of objects.

In the second set of experiments we compare the performance of *OLDA-FR* and *BVQ-FR* with other ranking methods known in literature such as *Relief*, *Gain Ratio* and *One-Rule*. Also heuristic methods calculate a weight for each individual real feature, which allows to rearrange the features by decreasing weights and to submit dataset to the 1NN classification algorithm using the same procedure as for the models based on EDBFM. Accuracies calculated in the previous experiment for *OLDA-FR* and *BVQ-FR* are now compared with accuracies obtained using the *Relief*, *Gain Ratio* and *One Rule*. The accuracy curves gathered by dataset are shown in Fig. 4.5. The criterion of comparison of curves is the same than in the previous experiment.

The general picture of performances is rather complex, but trends are evidenced by the analysis of the index  $\phi$ . For each dataset the best ranker is highlighted in the Table 4.4; there are also reported some statistics of the Performance Index: the mean value of  $\phi$  for *BVQ-FR* is the highest, and the variance has the lowest value. The statistics indicate a low dispersion of  $\phi$  for *BVQ-FR* algorithm, that reveals a relatively stable behaviour in comparison to *Relief*, *GainRatio* and *One-Rule* rankers and *OLDA-FR* as well.

In the star plot (see Fig. 4.6) the index values are shown as a radial line from a common centre point. Points corresponding to the same algorithm are connected by a common-style line. In the clockwise direction the datasets are sorted by increasing complexity. Notice in the part of the diagram where two-classes datasets are



**Fig. 4.4** Feature Ranking experiments. Comparing the performance of FE filter algorithms. On the horizontal axis the features sorted by rank and in vertical the percentage of accuracy

**Table 4.3** EDBFM ranking: comparison of filter FE algorithms

	OLDA-FR ( $\phi$ )	BVQ-FR ( $\phi$ )	ADBFE-FR ( $\phi$ )	SVM-FR ( $\phi$ )
HeartStat	0.116	0.411	0.015	0.188
Heart	0.187	0.471	—	0.144
Australian	0.685	0.463	0.180	-0.298
Urban	0.543	0.465	0.502	0.075
Wildfires	0.474	0.354	0.277	0.265
Landslides	0.400	0.388	0.224	0.446
Ionosphere	-0.096	0.207	0.019	0.017
Waveform	0.651	0.670	0.641	—
CoverType	0.125	0.373	—	—
Segment	0.348	0.595	—	—
Gottigen	0.445	0.456	—	0.447
Letter	0.133	0.287	—	—
Corine	0.484	0.592	—	—

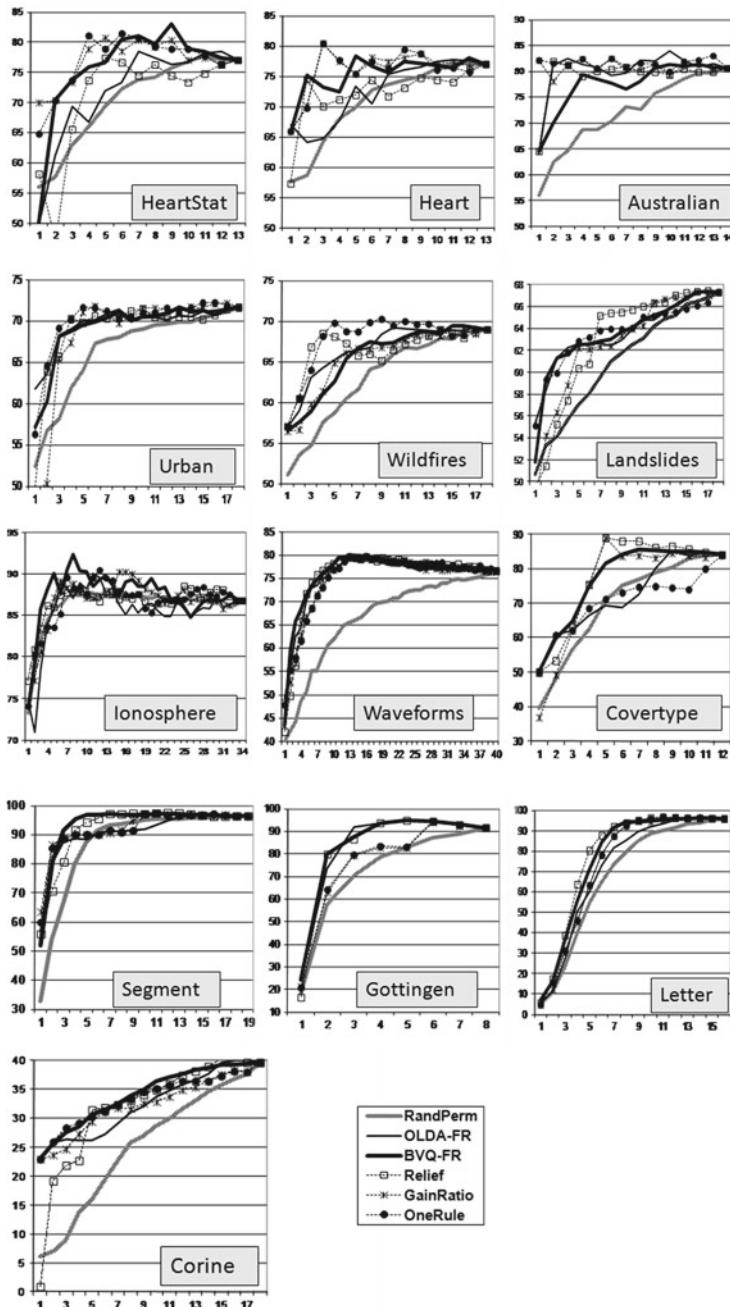
The *Performance Index* ( $\phi$ ) for each of the accuracy curves in Fig. 4.4

concentrated, from Heart to Ionosphere, there is an evident superiority of One Rule over the other rankers. By contrast where more complex datasets are concentrated, from Waveform to Corine, BVQ-FR tends to outperform the other rankers whose performance decreases more rapidly as the dataset complexity increases.

Another comparative indicator of performance is the number of features needed to reach 90 % of total accuracy, see Table 4.5. This indicator represents a relative measure of the steepness of the curve; it indicates the ranker's ability to lead to higher accuracies with relatively small subsets. On this indicator BVQ-FR outperforms all other rankers.

Let us observe in more detail a ranking model to highlight its usefulness in supporting cost-benefit informed decision making. In Fig. 4.7 left, for the Wildfire dataset, the curve of accuracy obtained for BVQ-FR is overlaid with the curve of cumulative weights, the horizontal axis represents the features sorted by rank. Notice that the first nine features, which are 50 % of total, represent half the cost of the entire dataset, but detain over 70 % of the total weight and over 98 % of the total accuracy achievable. Analogously, in Fig. 4.7 right, for CoverType dataset, the first feature holds 17 % of the total weight of the features, whereas the first six features (50 % of total features) detain over 70 % of the total weight and over 80 % of the accuracy achievable on the full dataset. If the individual costs of the features are given, it is possible to construct a detailed cost function. As a consequence, it is evident that the proposed methodology can guarantee the best ratio between cost of features acquisition and informative power.

As it was described above, the index  $\phi$  has been used to compare the relative performance of ranking algorithms on a dataset. To assess the overall performance for each algorithm the number of times that the algorithm has had the highest  $\phi$  was counted. These aleatory results, however, require a test of statistical significance

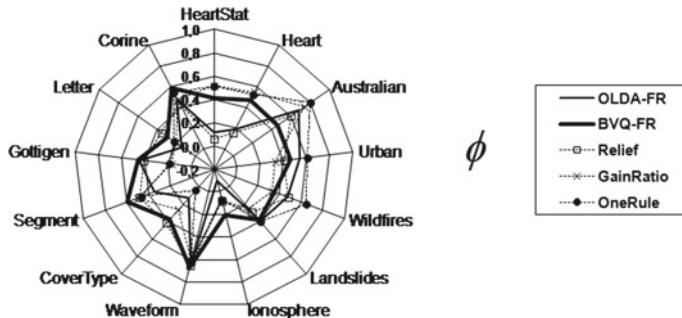


**Fig. 4.5** Feature Ranking experiments. Comparing the performance of Ranking Algorithms. On the *horizontal axis* the features are sorted by rank and in *vertical* the percentage of accuracy

**Table 4.4** The EDBFM ranking is compared to other methods (Relief, Gain Ratio, OneRule); the *Performance Index* ( $\phi$ ) is calculated for each of the accuracy curves in Fig. 4.5

	Goal oriented ranking				
	OLDA-FR ( $\phi$ )	BVQ-FR ( $\phi$ )	Relief ( $\phi$ )	Gain-ratio ( $\phi$ )	One-rule ( $\phi$ )
HeartStat	0.116	0.411	0.059	0.516	0.513
Heart	0.187	0.471	0.151	0.548	0.521
Australian	0.685	0.463	0.602	0.744	0.806
Urban	0.543	0.465	0.408	0.331	0.609
Wildfires	0.474	0.354	0.483	0.336	0.647
Landslides	0.400	0.388	0.292	0.251	0.402
Ionosphere	-0.096	0.207	0.083	0.075	0.072
Waveform	0.651	0.670	0.654	0.614	0.621
CoverType	0.125	0.373	0.418	0.264	0.040
Segment	0.348	0.595	0.483	0.501	0.467
Gottigen	0.445	0.456	0.407	0.169	0.187
Letter	0.133	0.287	0.349	0.198	0.217
Corine	0.484	0.592	0.438	0.474	0.545
<i>Mean</i>	0.346	0.441	0.371	0.386	0.434
<i>Variance</i>	0.055	0.016	0.034	0.039	0.056

The two bottom rows are descriptive statistics of the Performance Index computed values



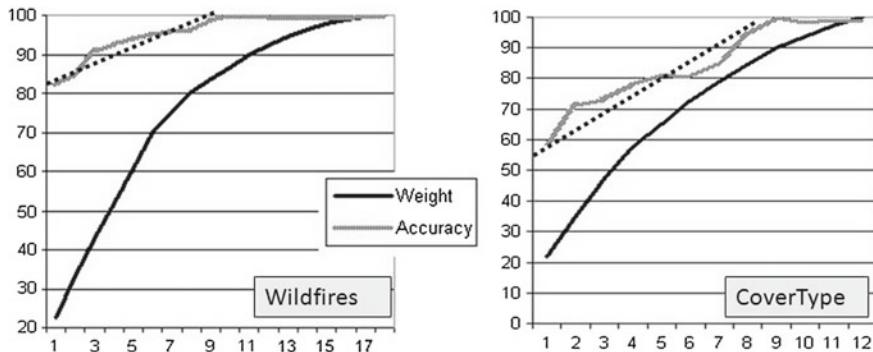
**Fig. 4.6** Each piecewise line represents a method of ranking, each *radial line* represents a dataset. Datasets are radially ordered by increasing complexity. The intersections represent the values of performance index  $\phi$

to support the observations made. The probability of success of an algorithm over another is calculated with the *binomial distribution*, from the count of victories and defeats, or the number of times that the algorithm outperformed the others on the basis of the index of performance. Assuming the *null hypothesis* is that the frequency of success of the two algorithms is the same, with the *two-tailed test* it can be seen how much we deviate from “null hypothesis” assumption.

**Table 4.5** Number of features needed to reach 90 % of total accuracy

	Rand rank	Goal oriented ranking				
		OLDA-FR	BVQ-FR	Relief	Gain-ratio	One-rule
HeartStat	9	7	4	11	4	4
Heart	6	5	2	8	3	3
Australian	9	2	4	2	N/D	N/D
Urban	6	3	3	3	3	3
Wildfires	8	4	6	3	5	3
Landslides	8	3	3	6	5	4
Ionosphere	4	4	3	3	4	4
Waveforms	24	6	6	6	8	8
Covertype	9	9	5	5	5	12
Segment	6	4	3	4	3	3
Gottingen	6	3	3	3	6	6
Letter	10	9	7	6	7	7
Corine	16	13	10	12	14	12

Dataset sorted by increasing complexity



**Fig. 4.7** Performance Indices cost-benefit of features of EDBFM based ranking. Left Wildfire dataset. Right CoverType dataset. On the horizontal axis the features sorted by rank and in vertical the values in percentage normalized to 100 %

In the Table 4.6 (top), the significance test is performed on all ranking experiments. The table does not allow us to assert the superiority of a method over another; pointing out that more experiments are needed. However, we have that BVQ-FR overcomes Relief and Gain Ratio with statistical significance greater than 0.9, while there is condition of parity with One Rule that is expressed by null statistic significance. It is noteworthy that BVQ-FR results have been obtained without stressing the parameters setup of the BVQ algorithm.

**Table 4.6** Overall comparison of the five algorithms

	Sign test—all datasets				
$\phi$	BVQ-FR	Relief	G.Ratio	Onerule	
OLDA-FR	4/9 0.733	7/6 0.0	6/7 0.0	3/10 0.908	w/l P
		10/3 0.908	10/3 0.908	7/6 0.0	w/l P
Relief			8/5 0.419	6/7 0.0	w/l P
				5/8 0.419	w/l P
GainRatio					

## 4.6 Conclusions

This chapter focuses on a novel ranking procedure. We considered that the premise for integrating the feature ranking models into domain knowledge is their representation in terms of real world features. This principle is the fundamental premise of the study conducted which leads to a computational model that is accurate and humanly understandable. A new approach to Feature Ranking (FR) based on features extraction (FE) and properties of the decision border has been discussed. This method uses Effective Decision Boundary Feature Matrix (EDBFM) to measure the relevance of the real world features thus maintaining the readability of the knowledge model extracted. The method has been tested on classification problems and cost-benefit analysis of features. While maintaining the geometric procedure which yields the ranking of features, this method allows to choose between alternative core FE algorithms, such as BVQ, when extracting the EDBFM, that allows to optimize the method application on datasets with different complexity. In particular BVQ-FR has proven to be more effective in applications to dataset of non-linearly separable points. Benchmarking tests, supported by the calculation of index of performance, show that BVQ-FR and OLDA-FR are generally more effective than other solutions. Furthermore, the comparison with known heuristic techniques of ranking confirms the robustness and the superiority of the EDBFM based method on complex dataset.

## References

1. Alelyani, S., Liu, H., Wang, L.: The effect of the characteristics of the dataset on the selection stability. In: Proceedings of the 23rd IEEE International Conference on Tools with Artificial Intelligence (ICTAI), pp. 970–977 (2011)
2. Araujo-Azofra, A., Aznarte, A.L., Benitez, J.M.: Empirical study of feature selection methods based on individual feature evaluation for classification problems. Expert Syst. Appl. **37**(3), 8170–8177 (2011)

3. Blum, A., Langley, P.: Selection of relevant features and examples in machine learning. *Artif. Intell.* **97**(1), 245–271 (1997)
4. Bock, M., Bohner, J., Conrad, O., Kothe, R., Ringler, A.: Saga, system for automated geo-scientific analysis. Technical Report Saga Users Group Association, University of Gottingen, <http://www.saga-gis.org> (2000)
5. Cantu-Paz, E., Newsam, S., Kamath, C.: Feature selection in scientific applications. In: Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 788–793 (2004)
6. Chandrashekhar, G., Sahin, F.: A survey on feature selection methods. *Comput. Electr. Eng.* **40**(1), 16–28 (2014)
7. Chawla, S.: Feature selection, association rules network and theory building. In: Proceedings of the Fourth Workshop on Feature Selection in Data Mining, pp. 14–21 (2010)
8. Dash, M., Liu, H.: Feature selection for classification. *Intell. Data Anal.* **97**(11), 131–156 (1997)
9. Diamantini, C., Panti, M.: An efficient and scalable data compression approach to classification. *ACM SIGKDD Explor.* **2**(2), 54–60 (2000)
10. Diamantini, C., Potena, D.: A study of feature extraction techniques based on decision border estimate. In: Liu, H., Motoda, H. (eds.) *Computational Methods of Feature Selection*, pp. 109–129. Chapman & Hall/CRC, Boca Raton (2007)
11. Ding, S., Zhu, H., Jia, W., Su, C.: A survey on feature extraction for pattern recognition. *Artif. Intell. Rev.* **37**(3), 169–180 (2012)
12. Escalante, H.J., Montes, M., Sucar, E.: An energy-based model for feature selection. In: Proceedings of the 2008 IEEE World Congress on Computational Intelligence (WCCI), pp. 1–8 (2008)
13. Gemelli, A., Mancini, A., Diamantini, C., Longhi, S.: *GIS to Support Cost-Effective Decisions on Renewable Sources: Applications for Low Temperature Geothermal Energy*. Springer, New York (2013)
14. Go, J., Lee, C.: Analytical decision boundary feature extraction for neural networks. In: Proceedings of the IEEE 2000 International Geoscience and Remote Sensing, pp. 3072–3074 (2000)
15. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. *J. Mach. Learn. Res.* **2003**(3), 1157–1182 (2003)
16. Guyon, I., Elisseeff, A.: An introduction to feature extraction. In: Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L. (eds.) *Feature Extraction, Foundations and Applications*, pp. 1–25. Springer, New York (2006)
17. Guyon, I., Aliferis, C., Elisseeff, A.: Causal feature selection. In: Liu, H., Motoda, H. (eds.) *Computational Methods of Feature Selection*, pp. 1–40. Chapman and Hall, London (2007)
18. Holte, R.C.: Very simple classification rules perform well on most commonly used datasets. *Mach. Learn.* **11**(1), 63–90 (1993)
19. John, G.H., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In: Proceedings of the 11th International Conference on Machine Learning, pp. 121–129 (1994)
20. Kira, K., Rendell, L.A.: The feature selection problem: traditional methods and a new algorithm. In: Proceedings of the Ninth National Conference on Artificial Intelligence, pp. 129–132 (1992)
21. Kohavi, R., John, G.H.: Wrappers for feature subset selection. *Artif. Intell.* **97**(1), 273–324 (1997)
22. Kohonen, T.: The self-organizing map. *Proc. IEEE* **78**(9), 1464–1480 (1990)
23. Kononenko, P.C.: Estimating attributes: analysis and extensions of relief. In: Proceedings of the European Conference on Machine Learning '94, pp. 171–182 (1994)
24. Lee, C., Landgrebe, D.A.: Feature selection based on decision boundaries. In: Proceedings of the IEEE 1991 International in Geoscience and Remote Sensing Symposium—IGARSS, pp. 1471–1474 (1991)
25. Lee, C., Landgrebe, D.A.: Feature extraction based on decision boundaries. *IEEE Trans. Pattern Anal. Mach. Intell.* **15**(4), 388–400 (1993)
26. Liu, H., Yu, L.: Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. Knowl. Data Eng.* **17**(4), 491–502 (2005)

27. Liu, H., Motoda, H.: Less is more. In: Liu, H., Motoda, H. (eds.) Computational Methods of Feature Selection, pp. 3–12. Chapman and Hall, London (2007)
28. Liu, H., Suna, J., Liu, L., Zhang, H.: Feature selection with dynamic mutual information. *Pattern Recognit.* **42**(7), 1330–1339 (2009)
29. Liu, H., Motoda, H., Setiono, R., Zhao, Z.: Feature selection: an ever evolving frontier in data mining. *J. Mach. Learn. Res.- Proc.* **10**(1), 4–13 (2010)
30. Monteiro, S.T., Murphy, R.J.: Embedded feature selection of hyperspectral bands with boosted decision trees. In: Proceedings of the IEEE 2011 International in Geoscience and Remote Sensing Symposium, IGARSS, pp. 2361–2364 (2011)
31. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: Repository of machine learning databases. University of California, Technical Report (1998)
32. Quinlan, J.R.: Improved use of continuous attributes in C4.5. *J. Artif. Intell. Res.* **4**(1), 77–90 (1996)
33. Senoussi, H., Chebel-Morello, B.: A new contextual based feature selection. In: Proceedings of the 2008 International Joint Conference on Neural Networks (IJCNN), pp. 1265–1272 (2008)
34. Sima, C., Attoor, S., Brag-Neto, U., Lowey, J., Suh, E., Dougherty, E.R.: Impact of error estimation on feature selection. *Pattern Recognit.* **38**(12), 2472–2482 (2005)
35. Singhi, K.S., Liu, H.: Feature subset selection bias for classification learning. In: Proceedings of the 23rd International Conference on Machine Learning, pp. 849–856 (2006)
36. Wang, L., Zhou, N., Chu, F.: A general wrapper approach to selection of class-dependent features. *IEEE Trans. Neural Netw.* **19**(7), 1267–1278 (2008)
37. Ye, J.: Characterization of a family of algorithms for generalized discriminant analysis on undersampled problems. *J. Mach. Learn. Res.* **6**, 483–502 (2005)
38. Zhao, Z., Wang, J., Sharma, S., Agarwal, N., Liu, H., Chang, Y.: An integrative approach to identifying biologically relevant genes. In: Proceedings of SIAM International Conference on Data Mining, pp. 838–849 (2010)

# Chapter 5

## Weighting of Features by Sequential Selection

Urszula Stańczyk

**Abstract** Constructing a set with characteristic features for supervised classification is a task which can be considered as preliminary for the intended purpose, just a step to take on the way, yet with its significance and bearing on the outcome, the level of difficulty and computational costs involved, the problem has evolved in time to constitute by itself a field of intense study. We can use statistics, available expert domain knowledge, specialised procedures, analyse the set of all accessible features and reduce them backward, we can examine them one by one and select them forward. The process of sequential selection can be conditioned by the performance of a classification system, while exploiting a wrapper model, and the observations with respect to selected variables can result in assignment of weights and ranking. The chapter illustrates weighting of features with the procedures of sequential backward and forward selection for rule and connectionist classifiers employed in the stylometric task of authorship attribution.

**Keywords** Weighting · Ranking of features · Sequential selection · Forward selection · Backward selection · DRSA · ANN · Stylometry · Authorship attribution

### 5.1 Introduction

In order to arrive at a set of characteristic features which are relevant for a task and give some satisfactory predictive accuracy for a classification system employed in supervised pattern recognition [20], we can either start with the empty set and then in the process of forward selection add some number of attributes to it, or we can execute backward elimination of features from some original set, chosen basing on expert domain knowledge, some other algorithms or measures, or even randomly. We can also attempt to go in both directions at the same time, mixing elimination with selection of features [19].

---

U. Stańczyk (✉)  
Institute of Informatics, Silesian University of Technology,  
Akademicka 16, 44-100 Gliwice, Poland  
e-mail: urszula.stanczyk@polsl.pl

In forward selection the initial dimensionality of the inducers used is low and it gradually increases when the numbers of considered variables get higher. Yet in such limited context, without the presence of other features, without observing interactions among them, any conclusions with regard to importance and relevance of attributes could be unreliable and misleading [24].

For rule classifiers the low dimensionality means quick induction of rules and relatively short decision algorithms, with few constituent rules, which seems to be an advantage, however, with not enough data to mine, the constructed rules tend to be approximate rather than certain and do not necessarily help in classification [33]. On the other hand, when a classification system is of a connectionist type, the learning stage is much more problematic when there are only few inputs to induce knowledge from. Artificial neural networks with insufficient number of input nodes can have significant trouble converging and training them takes much more time as more runs are needed to learn anything from the training facts [11].

When the approach is that of backward reduction, we start with induction process while dealing with some high number of attributes, and computational costs needed for inferring decision rules in such case are much higher, could even be unfeasible, depending on the induction algorithm. But, if it is still manageable, studying features in much wider context can bring additional information resulting in better performance of the classifier. Also, connectionist classification systems with more inputs converge faster because with many neurons and interconnections there is simply more room for adjustments of weights which minimises the error on the output.

The chapter illustrates a comparison of the two approaches of sequential selection with a case of a binary classification task of authorship attribution with balanced data [31, 32]. The characteristic features observed refer to textual markers of lexical and syntactic type, which enable definition and recognition of writing styles [25]. The procedures of sequential selection serve as a means to an end of assignment of weights to variables, depending on how their presence or absence in a considered feature subset influences predictive accuracy of the classification system.

The text is organised as follows. Section 5.2 contains a brief introduction to approaches to variable selection, exploited classifiers, and stylometric analysis. In Sect. 5.3 there is described a framework for conducted experiments. Sections 5.4 and 5.5 show results from tests focused respectively on forward and backward selection procedures. Concluding remarks are included in Sect. 5.6.

## 5.2 Background

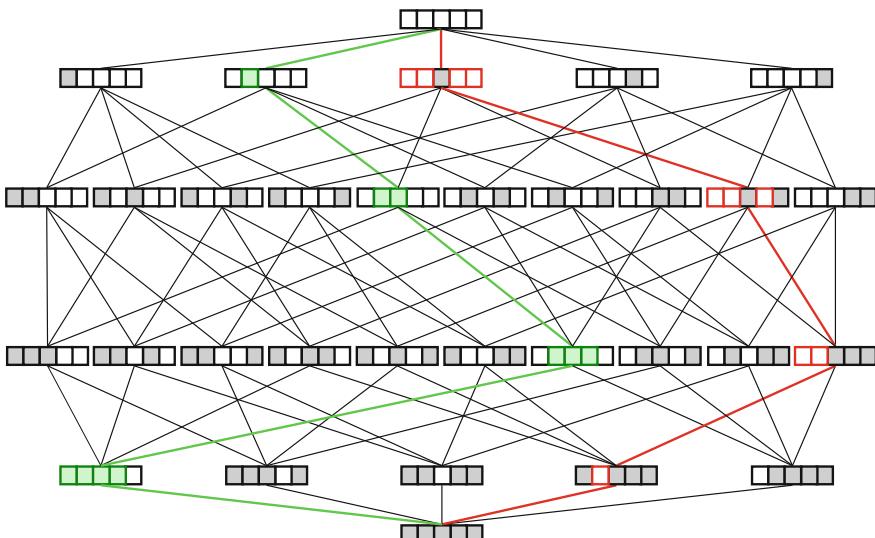
In the research presented in this chapter there are combined three issues, namely approaches to feature selection, connectionist and rule-based classifiers employed in pattern recognition, and stylometric processing of texts as the application domain, which are briefly described in this section.

### 5.2.1 Algorithms for Feature Selection

Any inducer used in data mining incorporates in fact some elements of feature selection in the learning phase, by the way in which it exploits some of the considered variables in higher while others in lower degree. The results of this processing are a part of the obtained solution, depending on its type either directly visible, for example through specific conditions included in decision rules, or hidden in the internal structure, as in artificial neural networks. Apart from those inherent mechanisms there are also many procedures and algorithms dedicated to the aim of selection and reduction of attributes [17].

In feature selection algorithms the decision with respect to the starting point in the feature space, and determined by it possible search directions, bears significantly upon other parameters of the process and its execution [9, 10]. Although theoretically possible, typically the exhaustive search and evaluation of all candidate subsets of features is unfeasible as for  $N$  attributes there are as many as  $2^N$  possible subsets, and even for relatively small  $N$  this number quickly becomes impractical under tests.

Imagine the feature space with five possible attributes to choose from, which corresponds to  $2^5 = 32$  candidate subsets of variables. We can start the search for a good subset, however this “goodness” is defined, with the empty set, then add to it some variables in the forward manner, basing on some evaluation criteria for each selection. Or, we can begin the procedure with the entire set of features, from which the elements are next rejected backward. This feature space and two exemplary search paths are illustrated in Fig. 5.1. Going top to bottom with each level we add



**Fig. 5.1** Points in the exemplary feature space for five attributes, with possible directions for search and either selection or reduction. In *top to bottom* direction there is indicated an example for forward selection path, and from *bottom to the top*, another example for backward elimination

one feature till we have the entire set. Going from bottom to the top at each step one variable is eliminated and at the end of this search path the set becomes empty. The former algorithm is called sequential forward selection, while the latter sequential backward reduction or elimination (or selection). A variation of the two approaches requires commencing with some non-empty set, and adding to it as well as reducing.

Whichever starting point is selected, we need to decide on a search direction, forward or backward, and some limitations that are imposed on the procedure. Instead of checking all available options, more popular and realistic approach is to apply some greedy methodology, where feature selection is executed stepwise—at each stage evaluation of a considered subset of attributes is based on this local context, and addition or removal of features depends on the fact whether this action results in increased performance. With this kind of processing if we conduct it from the beginning to the end without introducing any other stopping criteria the number of tested subsets equals to:

$$N + (N - 1) + (N - 2) + \cdots + 2 + 1 = \sum_{i=0}^{N-1} (N - i) = \frac{(N - 1)N}{2} \quad (5.1)$$

which is in a more manageable form of a polynomial than the exponential relationship given before.

The condition of increased performance uses the concept of relevance by incremental usefulness for attributes. This requirement could be considered as too strong, especially in case of backward elimination. It could be argued that if the predictive accuracy is the same regardless of presence or absence of some variable in the considered subset, then this variable is irrelevant for the task and can be disregarded, thus making the condition weaker.

When subsets of features are evaluated using the quality of prediction in a direct manner, it means employing the wrapper approach [18]. Another alternative is to use some measures, separate and independent from the system responsible for discriminating classes present in the training data, for example exploiting elements from information theory such as information gain, entropy, consistency [9].

Sequential selection procedures, whether forward or backward, executed in the wrapper mode explicitly return the information on how useful individual attributes are for the employed inducer, show how it prefers some variable over others, which can be interpreted as a scoring function assigning specific weights and organising features into specific ordering, which is in fact their ranking. In forward selection the first to be selected are the most important variables, in backward reduction the least important features are the first to be discarded. This importance of attributes is always considered in the local context, from the current perspective of the confines of the search path.

### 5.2.2 Connectionist Classifier

Artificial neural networks (ANN) are widely used in classification tasks due to their ability to generalise: they draw conclusions from the available training data and modify their topology by adjusting weights associated with interconnections in such way that leads to increased performance [11].

Multilayer Perceptron (MLP), which is a popular example of feed-forward unidirectional network topology, in the learning phase most often uses some version of backpropagation training rule that aims at minimisation of the error at the network output, for all outputs and all learning facts. Initial weights assigned to connections strongly influence the training procedure and can cause distinctively different predictive accuracy. To minimise that effect, multi-starting procedure is used, in which there is employed repetitive learning after randomisation of weights, and calculating average classification ratio. The performance of a connectionist classifier depends also on the number of hidden layers and neurons comprising them, and these parameters are usually established in tests.

One of disadvantages of ANNs lies in knowledge representation: even though the networks learn from the input data sets, the relationships detected are hidden in the internal structure of the solution and cannot help in understanding of information.

When there are no significant inconsistencies in the training sets, neural networks usually perform better for higher rather than lower numbers of inputs. For just few inputs a network has trouble converging and learning can require many more runs and still low classification accuracy is obtained. Choosing the best from generally poor solutions, without clear understanding of patterns, can be next to impossible and then still cause some inferior results.

Because of the general idea behind the concept of artificial neural networks, it is more natural to establish irrelevance of some inputs by observing their connection weights adjusted to some negligible values in learning, which can be then deleted in pruning [21, 23]. Thus backward elimination of features seems a better approach than forward selection.

### 5.2.3 Rule-Based Classification

Rule classifiers enable very clear and straightforward expression of available knowledge through decision rules of *IF...THEN...* type. The premise (or condition) parts specify the conditions on attributes that, when met, indicate a particular decision class (or a group of classes) to which the considered object should belong.

In multicriteria decision making [12, 13] better results are obtained while employing approaches that allow for not only nominal but also ordinal classification, possible by detecting and exploiting partial orderings of values for all variables [16]. One of such methodologies is Dominance-based Rough Set Approach (DRSA), which is a modification of classical rough set processing [29, 30], replacing the indiscernibility relation with dominance [35].

Rough set processing possesses an inherent mechanism for dimensionality reduction in the concept of *relative reducts* [27]. Relative reducts are such subsets of attributes which offer the same predictive accuracy as the entire set of attributes for the considered samples. If a reduct is activated, some of variables are excluded from the rule induction phase. If the intersection of all reducts, called the *core*, is non-empty, it includes all features that are necessary for classification, yet they are not necessarily sufficient. Also it often happens that there are many reducts and no indicators as to which one should be activated [26]. Reducts can be used indirectly, as a source of additional information on individual attributes, reflecting their importance for a task [36, 41].

Predictive accuracy of a rule classifier depends not only on the input data basing on which the constituent decision rules are inferred, but also, in the very high degree, on a selected approach to rule induction [5]. Possibly the quickest (yet not the simplest) is induction of a minimal cover—there is found only such small number of rules that are sufficient to classify correctly all learning samples. However, rules inferred with this approach are not necessarily the best. Taking under consideration for example a value of rule support, which is a parameter stating for how many training samples a rule is valid, it may turn out that other algorithms for rule induction can find some more interesting rules [34]. Generation of all rules on examples is the opposite approach to minimal cover and enables calculation of good, bad, and average rules, but at the cost of higher computational complexity and extended processing. If it can be afforded, induction of all rules and their analysis enables to tailor the decision algorithm to specific requirements [37, 38]. Once a set of rules is induced, we can filter some elements using quality measures.

Calculation of all rules on examples for sequential backward elimination of variables even for relatively small their number is a task of unmanageable proportions.

When the number of attributes is low, inferring rules takes distinctively less time which allows to employ sequential forward selection procedure. However, the differences in performance for algorithms found in initial stages can be so small that to choose the best one not only its predictive accuracy is taken into account but also other parameters, for example the number of rules in the algorithm and their type. The exact (certain) rules are the most useful for classification as they classify unambiguously. Possible and approximate rules point to possible inclusion in some class or a union of classes which do not help in increasing recognition without any further processing.

Classification results for rule classifiers are given in three groups of decisions: correct, incorrect, and ambiguous. The last of these is dedicated for cases when there are several rules with contradicting verdicts or no rules matching. In situation of contradicting verdicts the popular attitude is to execute some kind of voting, either by simple majority or with weighting of rules, for example by their support as it can be argued that rules with higher support can be considered as more important [39].

### 5.2.4 Textual Analysis

Some input data sets directly determine data mining techniques that can be applied to them while for others many alternative approaches can be used. Stylometry, which was the application domain in the research on weighting characteristic features in forward and backward selection illustrated in this chapter, refers to stylistic textual descriptors, reflecting individual linguistic preferences of writers [8]. In processing there are employed either computer-aided and statistic-oriented computations [22], or methodologies from machine learning area [1, 42].

While text categorisation with respect to a subject content uses some key words and phrases of specific significance [6], categorisation by text authors, which is considered as the most important of stylometric tasks, needs to detect more subtle linguistic elements because we want to recognise who has written a text regardless of what it is about [7].

In stylometric processing typically there are exploited textual descriptors employed rather subconsciously, based on common parts of speech. Under more detailed analysis they reveal patterns corresponding to individual habits and preferences, invisible to the bare eye, which makes them hard to imitate [2].

Even though linguists agree that we have individual writing styles, they cannot really help when asked for style definitions. Since styles are unique, they cannot be expressed by any general rule that would be universal and applicable to all writers and all texts [3]. Instead for any author a set of discriminating features needs to be established by tests.

The markers the most popularly used in authorship attribution come from either lexical or syntactic group. Lexical descriptors give such numerical statistics as frequencies of occurrences, distributions of frequencies, and averages for characters, words, and phrases [28]. Syntactic markers express organisation of a text in units such as sentences and paragraphs by punctuation marks [4].

## 5.3 Experimental Setting

To be reliable, all numerical characteristics need to be calculated over some sufficient number of representative writing samples. In fact, the bigger the corpus, the higher chance at good recognition ratio. That is why in experiments there were used novels written by two writers, Henry James and Thomas Hardy, divided into smaller parts of comparable length. All texts used in the experiments performed are available in electronic formats for download and on-line reading thanks to Project Gutenberg (<http://www.gutenberg.org>).

To avoid the problems that can result from imbalanced data sets used in classification, in both groups of samples exactly one half corresponds to one author and the other half to the second one, making the classification binary.

By referring to frequencies of usage for certain punctuation marks and elements from the list of the most common words in English language, 17 lexical and 8 syntactic descriptors were arbitrarily selected as follows:

- lexical: but, and, not, what, that, with, in, on, of, as, at, by, for, to, if, this, from,
- syntactic: a comma, a fullstop, a colon, a semicolon, a bracket, a hyphen, a question mark, an exclamation mark,

giving together a search space of 25 characteristic features (which have shown their usefulness in authorship studies [36, 38]), tested in both forward and backward selection procedures.

The frequencies used as features are continuous values with natural ordering. To use Dominance-based Rough Set Approach to this kind of data definitions of either increasing or decreasing preference orderings are required. This choice can be based on domain knowledge, but in its absence the problem can be treated as another aspect of knowledge discovery process and established in tests.

4eMka Software used for induction of DRSA decision rules was developed at the Laboratory of Intelligent Decision Support Systems, (<http://www-idss.cs.put.poznan.pl/>), Poznań University of Technology, Poland [14, 15].

At each stage of forward selection there were inferred two types of algorithms, minimal cover and all rules on examples, for two possible preference orderings of values for a considered variable, increasing and decreasing. From the sets of all generated rules only exact certain rules were used in classification. From the presented test results ambiguous decisions were excluded and they were treated as incorrect to avoid additional processing.

In case of that many input variables backward elimination of attributes while inducing all rules on examples algorithm would take much too long, and minimal cover algorithms typically perform unsatisfactorily. Instead, there was executed another methodology. Firstly all rules on examples algorithm for the complete set of 25 attributes was induced and to this algorithm backward reduction approach applied by rejecting rules containing conditions on evaluated features.

Artificial neural networks were simulated with California Scientific Brainmaker software. In the preliminary phase there were executed tests to establish a topology, in particular the number of hidden layers and the number of neurons in those layers. As a result, in all experiments dedicated to backward elimination of inputs the networks contained two hidden layers and the total number of hidden neurons equal to the number of network inputs. The two outputs corresponded to two recognised authors. Each network was trained 20 times with randomisation of weights between subsequent learning stages.

## 5.4 Sequential Forward Selection

Forward selection procedures started with induction of decision algorithms for 25 single-attribute subsets, from which the one performing best was chosen, which completed the first stage of selection. At the second stage to the already chosen

attribute another one was added and 24 two attribute subsets were prepared and rules for them generated. Again selection of the best algorithm ended the processing at this stage. The analogous procedure was executed at all stages that followed.

At the first and all subsequent stages there were generated four algorithms for each considered subset: two with the conditional attribute of *cost* (decreasing) type, and two for *gain* (increasing) type, for both minimal cover and all rules on examples algorithm. From inferred algorithms all possible and approximate rules were next excluded, then an algorithm was tested with respect to the maximal predictive accuracy. To this aim for all algorithms there were introduced hard constraints on rules with respect to minimal support required of rules to be taken under consideration in classification. In most cases these requirements resulted in increased performance. The details of conducted experiments are listed in Table 5.1.

As ambiguous cases of no rule matching the testing samples or contradicting decisions were always treated as incorrect, the performance of these rule classifiers in the initial phase, when there are only few considered features, is rather poor. However, it increases quickly and gradually with each added attribute. For just few conditional attributes from which rules are inferred, the two types of algorithms, minimal cover and all rules on examples, are not that different, with similar numbers of constituent rules and close performance level. Once there are more features the differences are more distinct.

The first local maximum is detected for the subset of just five attributes, for which all rules on examples algorithm limited by rejecting rules with support lower than 7 classifies correctly 91.67 % of samples. The best performance for six variables for this type of algorithm is lower, 88.33 %. Yet for the same subset the minimal cover algorithm has predictive accuracy of 91.67 %, which is kept at the same level also for seven features before it decreases to 83.33 % for eight attributes. The performance of all best rule classifiers at each stage in shown in Fig. 5.2 for both minimal cover and all rules on examples decision algorithms, denoted as MCDA and FDA respectively.

In forward selection approach with each iterative step of the procedure we deal with more and more variables and at each step we can ask the question whether it is enough, whether we have the set of features that satisfy our requirements. The answer is not straightforward. Even when the predictive accuracy is considered as the most important factor on which the decision is based, it is not a simple task of reaching some maximum, as upon finding it we cannot possibly know if this is of local character or global, and after some decreased performance for another subset in the search path we can encounter another local maximum. We know what the maximum is only when all possible subsets of attributes are tested (all possible on the selected search path, which is not exhaustive), that is including the entire set of available variables.

When we can afford the extended processing of search procedures executed without additional stopping criteria, the observed performance for subsets of variables, with gradually increasing cardinalities, can be used as means to feature weighting and ranking, to be employed for another inducer as a kind of filter. Or, we can finish the variable selection procedure by choosing such subset of features for which the classification accuracy was the highest when compared to all tested alternatives.

**Table 5.1** Forward selection of attributes basing on performance of minimal cover and all rules on examples decision algorithms generated

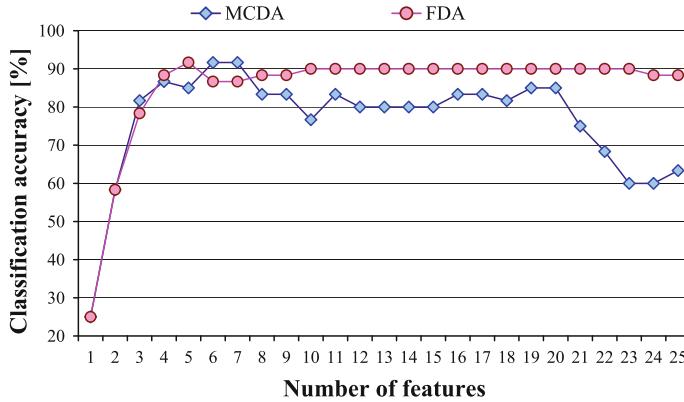
(a)	(b)	(c)	Minimal cover algorithm				All rules on examples algorithm				
			(d)	(e)	(f)	(g)	(h)	(d)	(e)	(f)	(g)
1	that	gain	3	2			25,00	3	2		25,00
2	by	cost	29	8	8	7	58,33	27	15	6	13
3	from	cost	38	15	20	9	81,67	71	37	16	22
4	as	cost	38	17	9	11	86,67	154	54	11	35
5	what	gain	30	20	19	8	85,00	241	119	7	103
6	of	cost	32	24	9	16	91,67	328	191	7	146
7	not	gain	34	27	5	23	91,67	468	237	11	150
8	if	gain	34	32	4	22	83,33	453	384	12	230
9	!	gain	37	35	6	17	83,33	569	453	12	258
10	.	gain	33	31	6	15	76,67	756	628	30	67
11	this	cost	34		9	12	83,33	878		31	55
12	in	cost	32		9	11	80,00	1,155		31	66
13	(	gain	44		2	31	80,00	1,424		31	73
14	:	cost	50		2	36	80,00	2,004		31	77
15	-	cost	52		2	38	80,00	3,167		31	80
16	with	cost	59		2	45	83,33	4,238		31	82
17	,	cost	61		2	43	83,33	6,307		31	89
18	for	cost	65		2	48	81,67	7,567		31	89
19	but	cost	63				85,00	9,524		31	89
20	at	cost	63				85,00	11,503		31	89

(continued)

**Table 5.1** (continued)

(a)	(b)	(c)	Minimal cover algorithm				All rules on examples algorithm					
			(d)	(e)	(f)	(g)	(h)	(d)	(e)	(f)	(g)	(h)
21	?	cost	65		4	19	75.00	15,315		31	92	90.00
22	and	gain	65		2	47	68.33	17,279		31	92	90.00
23	to	cost	71		2	43	60.00	23,865		31	94	90.00
24	on	cost	66		4	14	60.00	30,503		36	44	88.33
25	;	gain	62		2	36	63.33	43,096		36	44	88.33

Columns present parameters: (a) selection stage equal to the number of considered characteristic features, (b) the most recently selected variable, (c) preference order for the most recently selected attribute, (d) number of rules in a decision algorithm without any constraints, (e) number of exact rules when they are fewer than the total number, (f) minimal support required of rules resulting in maximal classification accuracy, (g) number of rules meeting constraints on support, (h) maximal predictive accuracy of the classifier (%)



**Fig. 5.2** DRSA classification accuracy observed in sequential forward selection process, in relation to the number of considered features, for both minimal cover (MCDA) and all rules on examples (FDA) decision algorithms

This approach is also the one to be used when the number of possible features is practically infinitive—as the search cannot be endless, we need to limit the number of subsets to be tested somehow and from them select the best.

Once the search path of forward sequential selection is completed the goal of feature weighting is achieved by observing the order in which attributes were chosen, as listed in the (b) column of Table 5.1, with the frequency of occurrence of “that” at the top of the list and “;” at the bottom.

In the presented research to the central search for important attributes one more element was introduced, which added one more dimension to the search space at each stage, and it was a preference order for an attribute. In the stylometric domain preference orders should be understood as associating certain, higher or lower, frequencies of usage of linguistic elements with specific authors. Although undeniably such relationships exist and enable authorship attribution in the first place, a priori knowledge about them, ready and applicable to any writers, any texts, any samples, does not exist. In its absence preference orders can be arbitrarily assumed, or they can be adjusted through some processing [40]. But when we actively search for subsets of relevant attributes it is more natural to extend this search to include not only selection of variables but their preference orders as well. As a result the obtained solutions are closer tailored to specifics of the classification task, which is visible in higher predictive accuracy when we compare them to the case of rule classifiers induced for attributes with arbitrarily assigned preference orders, tested within sequential backward elimination procedures described in the next section.

## 5.5 Sequential Backward Selection

Backward elimination of variables starts with inducing a solution for all available or considered features to be treated as the reference point. Then there are tested  $N$  subsets of variables, each with a single variable rejected, and the performance of the classifiers compared. We would prefer the predictive accuracy to increase, however, already one goal of backward selection is achieved through reduction of dimensionality, so as long as the performance is not worse, we can still consider it as satisfying requirements. From all tested subsets the one for which classification accuracy is the highest is selected and the whole procedure repeated for  $N - 1$  attributes, then for  $N - 2$ , and so on. Backward selection procedures were employed for both connectionist and rule classifiers.

As mentioned before, for ANN backward elimination is better suited than forward selection. Naturally all networks during the training phase learn to some degree the relevance of particular input features and this learned knowledge is expressed by adjusting weights of interconnections. It is also possible to exploit some input pruning algorithms, which, however, involves rather complex calculations and processing, whereas the general steps of backward selection are straightforward. We need to test relatively many networks with many inputs but in such cases the classifiers converge quickly and typically without trouble. When the numbers of inputs get lower the training takes more time, but there are also significantly fewer such networks to be tested.

The details from the conducted experiments in which artificial neural networks were used as an inducer are given in Table 5.2, where the right-most (e) column presents the order reflecting the weights assigned to attributes by the sequential backward selection. The elimination of “not” begins the list, and “but” was kept to the very end of the search, which indicates its high importance. The column specifying the classification accuracy displays median performance, because to minimise the influence of the initial weights associated with interconnections on the results of the learning phase the multi-starting approach was employed with repeating the learning phase several times and accumulating the minimal, median, and maximal predictive accuracies, plotted in Fig. 5.3.

It can be observed that in the initial 7 steps the performance is increased with each reduced variable to stabilise at the level of 96.67 % for the next 9, then to decrease when the number of remaining features falls below 10. Yet when compared to the performance of the network for all 25 input variables, only in the last two steps, for classifiers referring to respectively just two and one input, the obtained results are worse.

The additional parameters of minimal and maximal performance can be used not only to achieve higher reliability of obtained classification results, but also as factors helping in selection of features. It may happen (and actually in the research is did happen many times) that at some elimination stage several subsets of features give the same results when only median classification accuracy is compared. Then we can analyse the results from each step of multi-starting in more detail, check

**Table 5.2** Sequential backward elimination of attributes basing on the performance of ANN classifiers

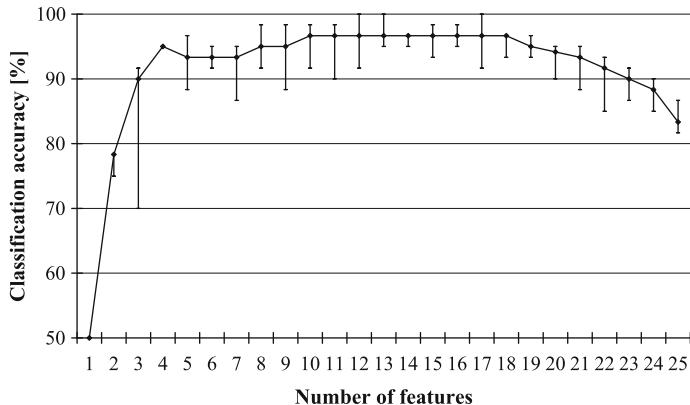
(a)	(b)	(c)	(d)	(e)
0	25	but and not in with on at of as this that by for to if what from . , ; : ! ? ( -	83.33	not
1	24	but and in with on at of as this that by for to if what from . , ; : ! ? ( -	88.33	(
2	23	but and in with on at of as this that by for to if what from . , ; : ! ? -	90.00	in
3	22	but and with on at of as this that by for to if what from . , ; : ! ? -	91.67	and
4	21	but with on at of as this that by for to if what from . , ; : ! ? -	93.33	,
5	20	but with on at of as this that by for to if what from . , ; : ! ? -	94.17	-
6	19	but with on at of as this that by for to if what from . , ; : ! ?	95.00	with
7	18	but on at of as this that by for to if what from . ; : ! ?	96.67	on
8	17	but at of as this that by for to if what from . ; : ! ?	96.67	what
9	16	but at of as this that by for to if from . ; : ! ?	96.67	to
10	15	but at of as this that by for if from . ; : ! ?	96.67	of
11	14	but at as this that by for if from . ; : ! ?	96.67	.
12	13	but at as this that by for if from ; : ! ?	96.67	:
13	12	but at as this that by for if from ; ! ?	96.67	?
14	11	but at as this that by for if from ; !	96.67	;
15	10	but at as this that by for if from !	96.67	this
16	9	but at as that by for if from !	95.00	as
17	8	but at that by for if from !	95.00	at
18	7	but that by for if from !	93.33	for
19	6	but that by if from !	93.33	!
20	5	but that by if from	93.33	if
21	4	but that by from	95.00	from
22	3	but that by	90.00	by
23	2	but that	78.33	that
24	1	but	50.00	but

Columns present parameters: (a) elimination stage, (b) number of characteristic features left, (c) set of currently considered variables, (d) median predictive accuracy of the classifier (%), (e) attribute selected to be eliminated

the distributions and dispersions of specific classification accuracies, we can assign higher priority to these networks that have especially good results such as for example 100 % recognition.

As before for forward selection, for backward elimination the search for subsets of features is not exhaustive. Commencing with the entire set of variables we reject one variable at a time with the decision based on the local context, and once some variable is reduced it is not taken under consideration for the second time.

For the set of attributes with cardinality of 25 with arbitrarily assigned preference orders, the decision algorithm generated within the approach of finding only minimal cover performs rather poorly, correctly recognising barely a half of the testing samples. We can try to increase this accuracy by adjusting preference orders, yet there are no quick procedures that could be employed to this end. When we induce all rules



**Fig. 5.3** ANN classification accuracy observed in sequential backward elimination process, in relation to the number of considered features, for each average there is indicated maximal and minimal performance

on examples, the full algorithm with hard constraints on minimal rule support gives much better results of 76.67 %. Yet this full algorithm contains over 46 thousands of constituent decision rules and calculation takes a noticeable amount of time. Generation of this type of algorithm at all stages of backward elimination of features would be far too much time-consuming. Instead, another approach is employed, focused on the previously inferred all rules on examples algorithm for the entire set of attributes.

Decision rules included in the full algorithm have varying lengths equal to the numbers of conditions in the premise part, varying supports, refer to various attributes. If we were to employ backward elimination procedures and attempted to induce all rules on examples algorithms for subsets of variables in subsequent reduction stages it is reasonable to expect that at least a part of newly inferred rules would be the same as those already found. Therefore, rather than waste time on such simply repetitive computations we can apply the backward selection to the full algorithm itself in the following manner.

In the first step we disregard all rules with conditions on a specific variable, iteratively for all variables. As a result we obtain 25 different reduced decision algorithms, which are then tested and the one with the highest classification accuracy is selected. We reject the attribute, elimination of which have resulted in this algorithm, and use this limited set of decision rules as the input to the second stage of processing, where 24 reduced algorithms are tested, and so on, with reduction of one feature and all rules referring to it at each stage. The process can continue till the point of detecting some significantly worse performance, rejecting all rules from the algorithm (even though there are still some features left to be considered), or reducing all conditional attributes. With this last stopping criterion there is obtained weighting of variables which can be used for other purposes. The procedure of discarding rules governed by included attributes can be perceived as rule filtering.

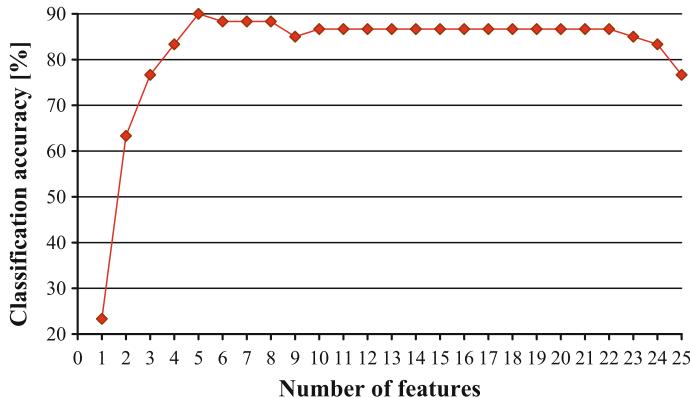
The detailed results from execution of described methodology are shown in Table 5.3, in which the ranking of features is given in the (h) column.

**Table 5.3** Backward elimination of attributes basing on the performance of reduced all rules on examples decision algorithm

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
0	25	but and not in with on at of as this that by for to if what from . , ; : ! ? ( -	46, 191	41	80	76.67	and
1	24	but not in with on at of as this that by for to if what from . , ; : ! ? ( -	42, 018	33	19	83.33	(
2	23	but not in with on at of as this that by for to if what from . , ; : ! ? -	34, 390	30	38	85.00	on
3	22	but not in with at of as this that by for to if what from . , ; : ! ? -	24, 732	29	35	86.67	as
4	21	but not in with at of this that by for to if what from . , ; : ! ? -	19, 869	29	33	86.67	this
5	20	but not in with at of that by for to if what from . , ; : ! ? -	15, 072	29	32	86.67	-
6	19	but not in with at of that by for to if what from . , ; : ! ?	12, 407	29	31	86.67	.
7	18	but not in with at of that by for to if what from . , ; : ! ?	9, 065	29	31	86.67	:
8	17	but not in with at of that by for to if what from . , ; : ! ?	6, 597	29	31	86.67	with
9	16	but not in at of that by for to if what from . , ; : ! ?	4, 939	29	31	86.67	:
10	15	but not in at of that by for to if what from . , ! ?	3, 740	29	31	86.67	for
11	14	but not in at of that by to if what from . , ! ?	3, 031	29	31	86.67	if
12	13	but not in at of that by to what from . , ! ?	2, 456	29	31	86.67	what
13	12	but not in at of that by to from . , ! ?	2, 131	29	31	86.67	that
14	11	but not in at of by to from . , ! ?	1, 841	29	31	86.67	but
15	10	not in at of by to from . , ! ?	1, 659	29	31	86.67	in
16	9	not at of by to from . , ! ?	1, 085	29	21	85.00	?
17	8	not at of by to from . , !	861	21	55	88.33	at
18	7	not of by to from . , !	649	21	54	88.33	!
19	6	not of by to from . ,	407	16	88	88.33	to
20	5	not of by from . ,	311	13	106	90.00	,
21	4	not of by from . ,	172	13	84	83.33	of
22	3	not by from . ,	100	11	71	76.67	not
23	2	by from . ,	34	16	22	63.33	by
24	1	from . ,	4	8	4	23.33	from

Columns present parameters: (a) elimination stage, (b) number of characteristic features left, (c) set of currently considered variables, (d) number of rules in a decision algorithm without any constraints, (e) minimal support required of DRSA rules resulting in maximal classification accuracy, (f) number of exact DRSA rules meeting constraints on support, (g) maximal predictive accuracy of the classifier (%), (h) attribute selected to be eliminated

When we compare predictive accuracies of rule classifiers tested at each reduction stage, plotted in Fig. 5.4, not by exact numbers but perceivable trends, to those previously studied in forward selection approach, it is immediately apparent that they



**Fig. 5.4** Classification accuracy observed for decision algorithms in sequential backward elimination process of attributes and rules, in relation to the number of considered features

are very close. The fact that here the correct classification ratio is lower than before results from the different attitude to preference orders of features. In forward selection it was treated as one more element to be established, thus better adjusted to the task under study, while for backward elimination in the initial stage the preference orders are arbitrarily assigned and remain unchanged throughout all processing.

The two types of classifiers used in experiments have very distinctive properties, and the differences between them are also visible in obtained by them rankings of considered variables, which shows bias that all wrappers are prone to. For example, for ANN the feature “not” is rejected as the first one, while for DRSA classifier it is kept almost to the end. When we compare orderings of variables from forward with backward selection for rule classifiers, even though the type of the inducer employed is the same, we cannot say that they are reversed. All these observations illustrate how different perspectives in which attributes are considered can change relationships and dependencies detectable among studied elements, to the point of completely different relevance of the same features and their resulting weightings.

## 5.6 Concluding Remarks

Backward and forward sequential search are two approaches to feature selection, with the opposite starting points in the feature space. In forward selection we commence with the empty set of variables to which we add one element at a time. In backward elimination we begin with the entire set of attributes, which are reduced one by one. The two procedures are relatively simple to apply, even though could be time consuming, depending on the number of available variables to be tested, and a type of inducer used in validation of candidate feature subsets.

In the experiments dedicated to selection of variables two distinctively different classifiers were employed, connectionist approach of artificial neural networks and rule-based exploiting rough set theory incorporating dominance relation.

For ANN classifiers it is more natural to apply backward elimination, because networks with excessive numbers of inputs still can perform better than those with insufficient features. In the training phase networks can detect just by themselves which input variables are less important and assign to them low weights of interconnections, minimising their influence on the outcome. On the other hand, when there are not enough of characteristic features, the network can try and generalise, yet the conclusions cannot be drawn from nothing. As a result neural networks with only few inputs typically need more time to be trained, can have trouble converging and then generalising for unknown samples.

Induction of decision rules takes significantly less time for fewer attributes. However, they do not necessarily contain information required to infer rules with good quality, resulting in high predictive accuracy. Applying forward selection procedures we can not only choose the attributes that are the most beneficial to rule induction process, but also adjust their preference orders which further increases performance. Typically minimal cover decision algorithms give worse results than rule classifiers constructed in different approaches, for example all rules on examples with some hard constraints on constituent rules such as minimal support required. Yet inferring all rules when there are many attributes requires a lot of computations and takes time. Since in subsequent stages of backward elimination many of generated rules would be the same, as the studied subsets of features are overlapping, we can employ another methodology, in which backward reduction is in fact applied to rules referring to rejected features.

For all search paths tested one of the important elements to consider is the stopping criterion, answering the question when or where the selection procedures should end. The response is not trivial as it depends to a high degree on the purpose of applying the search procedures in the first place. When the goal is just to find a good subset of features, that is resulting in an induced solution with satisfactorily high predictive accuracy, we can stop the search once we detect a maximum in correct recognition ratio. However, if we do it too quickly, before checking alternative subsets, it may turn out that a maximum is only local and not global, and for some other candidate subset of variables the predictive accuracy is better.

If extended processing is acceptable, or with the goal of weighting available variables we test all possible subsets of variables in a search path. We do observe the performance (after all the choice is conditioned by it), but we also study the order in which all features are organised. This order reflects their weighting from the perspective of applied search procedure and inducer employed. As classifiers have different characteristics and the selection of variables is wrapped around their performance, the same search direction applied for another classification system, with distinctively different properties may return completely different ranking of attributes. From all validated subsets we can choose the best, or we can impose the obtained ranking of features on a separate classification process and test its usefulness as a filter.

All attribute selection procedures were illustrated for a binary classification task with balanced data, for the problem of authorship attribution from stylometric processing of texts. The most important aim of textual analysis is to find definitions

of individual writing styles by referring to such linguistic features which betray individual preferences and are employed rather subconsciously. Lexical and syntactic descriptors exploited enable definition of individual styles and categorisation of texts by their authors.

## References

1. Ahonen, H., Heinonen, O., Klemettinen, M., Verkamo, A.: Applying data mining techniques in text analysis. Technical Report C-1997-23, Department of Computer Science, University of Helsinki, Finland (1997)
2. Argamon, S., Burns, K., Dubnov, S. (eds.): *The Structure of Style: Algorithmic Approaches to Understanding Manner and Meaning*. Springer, Berlin (2010)
3. Argamon, S., Karlgren, J., Shanahan, J.: Stylistic analysis of text for information access. In: Proceedings of the 28th International ACM Conference on Research and Development in Information Retrieval, Brazil (2005)
4. Baayen, H., van Haltern, H., Tweedie, F.: Outside the cave of shadows: using syntactic annotation to enhance authorship attribution. *Lit. Linguist. Comput.* **11**(3), 121–132 (1996)
5. Bayardo Jr., R., Agrawal, R.: Mining the most interesting rules. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 145–154 (1999)
6. Berber Sardinha, T.: Using key words in text analysis: practical aspects (1999). Available on-line from <ftp://ftp.liv.ac.uk/pub/linguistics>
7. Burrows, J.: Textual analysis. In: Schreibman, S., Siemens, R., Unsworth, J. (eds.) *A Companion to Digital Humanities*. Blackwell, Oxford (2004)
8. Craig, H.: Stylistic analysis and authorship studies. In: Schreibman, S., Siemens, R., Unsworth, J. (eds.) *A Companion to Digital Humanities*. Blackwell, Oxford (2004)
9. Dash, M., Liu, H.: Feature selection for classification. *Intell. Data Anal.* **1**, 131–156 (1997)
10. Dash, M., Liu, H.: Consistency-based search in feature selection. *Artif. Intell.* **151**, 155–176 (2003)
11. Fiesler, E., Beale, R.: *Handbook of Neural Computation*. Oxford University Press, Oxford (1997)
12. Greco, S., Matarazzo, B., Słowiński, R.: Advances in multiple criteria decision making. In: Gal, T., Hanne, T., Stewart, T. (eds.) *The Use of Rough Sets and Fuzzy Sets in Multi Criteria Decision Making* Chap. 14, pp. 14.1–14.59. Kluwer Academic Publishers, Boston (1999)
13. Greco, S., Matarazzo, B., Słowiński, R.: Rough set theory for multicriteria decision analysis. *Eur. J. Oper. Res.* **129**(1), 1–47 (2001)
14. Greco, S., Matarazzo, B., Słowiński, R.: Dominance-based rough set approach as a proper way of handling graduality in rough set theory. *Trans. Rough Sets* **7**, 36–52 (2007)
15. Greco, S., Słowiński, R., Stefanowski, J.: Evaluating importance of conditions in the set of discovered rules. *Lect. Notes Artif. Intell.* **4482**, 314–321 (2007)
16. Greco, S., Słowiński, R., Stefanowski, J., Żurawski, M.: Incremental versus non-incremental rule induction for multicriteria classification. *Trans. Rough Sets* **2**, 33–53 (2004)
17. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. *J. Mach. Learn. Res.* **3**, 1157–1182 (2003)
18. Jelonek, J., Krawiec, K., Stefanowski, J.: Comparative study of feature subset selection techniques for machine learning tasks. In: Proceedings of the 7th Workshop on Intelligent Information Systems (1998)
19. Jensen, R., Shen, Q.: *Computational Intelligence and Feature Selection*. Wiley, Hoboken (2008)
20. John, G., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In: Cohen, W., Hirsh, H. (eds.) *Machine Learning: Proceedings of the 11th International Conference*, pp. 121–129. Morgan Kaufmann Publishers (1994)

21. Kavzoglu, T., Mather, P.: Assessing artificial neural network pruning algorithms. In: Proceedings of the 24th Annual Conference and Exhibition of the Remote Sensing Society, pp. 603–609. Greenwich (2011)
22. Khmelev, D., Tweedie, F.: Using Markov chains for identification of writers. *Lit. Linguist. Comput.* **16**(4), 299–307 (2001)
23. Kingston, G., Maier, H., Lambert, M.: A statistical input pruning method for artificial neural networks used in environmental modelling. In: Transactions of the 2nd Biennial Meeting of the International Environmental Modelling and Software Society, pp. 87–92. Osnabrueck (2004)
24. Liu, H., Motoda, H.: Computational Methods of Feature Selection. Chapman & Hall/CRC, Boca Raton (2008)
25. Lynam, T., Clarke, C., Cormack, G.: Information extraction with term frequencies. In: Proceedings of the Human Language Technology Conference, pp. 1–4. San Diego (2001)
26. Moshkov, M., Piliszczuk, M., Zielosko, B.: On partial covers, reducts and decision rules with weights. *Trans. Rough Sets* **6**, 211–246 (2006)
27. Moshkov, M., Skowron, A., Suraj, Z.: On covering attribute sets by reducts. In: Kryszkiewicz, M., Peters, J., Rybinski, H., Skowron, A. (eds.) Rough Sets and Emerging Intelligent Systems Paradigms. LNCS (LNAI), vol. 4585, pp. 175–180. Springer, Berlin (2007)
28. Munro, R.: A Queuing-theory model of word frequency distributions. In: Proceedings of the 1st Australasian Language Technology Workshop, pp. 1–8. Melbourne (2003)
29. Pawlak, Z.: Rough sets. *Int. J. Comput. Inform. Sci.* **11**(5), 341–356 (1982)
30. Pawlak, Z.: Rough sets and intelligent data analysis. *Inf. Sci.* **147**, 1–12 (2002)
31. Peng, R.: Statistical aspects of literary style. Bachelor's Thesis, Yale University (1999)
32. Peng, R., Hengartner, H.: Quantitative analysis of literary styles. *Am. Stat.* **56**(3), 15–38 (2002)
33. Shen, Q.: Rough feature selection for intelligent classifiers. *Trans. Rough Sets* **7**, 244–255 (2006)
34. Sikora, M.: Rule quality measures in creation and reduction of data rule models. In: Greco, S., Hata, Y., Hirano, S., Inuiguchi, M., Miyamoto, S., Nguyen, H., Słowiński, R. (eds.) Rough Sets and Current Trends in Computing. Lecture Notes in Computer Science, vol. 4259, pp. 716–725. Springer (2006)
35. Słowiński, R., Greco, S., Matarazzo, B.: Dominance-Based Rough Set Approach to Reasoning About Ordinal Data. LNCS (LNAI), vol. 4585, pp. 5–11 (2007)
36. Stańczyk, U.: Relative reduct-based selection of features for ANN classifier. In: Cyran, K., Kozielski, S., Peters, J., Stańczyk, U., Wakulicz-Deja, A. (eds.) Man-Machine Interactions. AISC, vol. 59, pp. 335–344. Springer, Berlin (2009)
37. Stańczyk, U.: DRSA decision algorithm analysis in stylometric processing of literary texts. In: Szczuka, M., Kryszkiewicz, M., Ramanna, S., Jensen, R., Hu, Q. (eds.) Rough Sets and Current Trends in Computing. LNCS (LNAI), vol. 6086, pp. 600–609. Springer, Berlin (2010)
38. Stańczyk, U.: Reduct-based analysis of decision algorithms: application in computational stylistics. In: Corchado, M., Kurzyński, E., Woźniak, M. (eds.) Hybrid Artificial Intelligence Systems. Part 2. LNCS (LNAI), vol. 6679, pp. 295–302. Springer (2011)
39. Stańczyk, U.: Rule-based approach to computational stylistics. In: Bouvry, P., Kłopotek, M., Marciniak, M., Mykowiecka, A., Rybiński, H. (eds.) Security and Intelligent Information Systems. LNCS (LNAI), vol. 7053, pp. 168–179. Springer, Berlin (2012)
40. Stańczyk, U.: On preference order of DRSA conditional attributes for computational stylistics. In: Decker, H., Lhotska, L., Link, S., Tjoa, B.J.A. (eds.) Database and Expert Systems Applications. LNCS, pp. 26–33. Springer, Berlin (2013)
41. Stańczyk, U.: Relative reduct-based estimation of relevance for stylometric features. In: Cataño, B., Guerrini, G., Pokorný, J. (eds.) Advances in Databases and Information Systems. LNCS, vol. 8133, pp. 135–147. Springer, Berlin (2013)
42. Waugh, S., Adams, A., Tweedie, F.: Computational stylistics using artificial neural networks. *Lit. Linguist. Comput.* **15**(2), 187–198 (2000)

**Part II**

**Rough Set Approach**

**to Attribute Reduction**

# Chapter 6

## Dependency Analysis and Attribute Reduction in the Probabilistic Approach to Rough Sets

Wojciech Ziarko

**Abstract** Two probabilistic approaches to rough sets are discussed in this chapter: the variable precision rough set model and the Bayesian rough set model, as they apply to data dependencies detection, analysis and their representation. The focus is on the analysis of data co-occurrence-based dependencies appearing in classification tables and probabilistic decision tables acquired from data. In particular, the notion of attribute reduct, in the framework of probabilistic approach, is of interest in the chapter. The reduct allows for information-preserving elimination of redundant attributes from classification tables and probabilistic decision tables. The chapter includes two efficient reduct computation algorithms.

**Keywords** Variable precision rough set model · Bayesian rough set model · Dependency analysis · Reduct

### 6.1 Introduction

The chapter reviews the basics of the variable precision rough set [2, 3, 7, 13, 15, 26, 28, 30, 32, 34, 35] and the Bayesian rough set [13] approaches to data dependencies detection, analysis and their optimal representation. The variable precision rough set and the Bayesian rough set theories are extensions of the rough set theory, as introduced by Pawlak [10, 11]. They are among many extensions and generalizations of the rough set approach, which inspired significant research interest worldwide (see, for example [5, 12, 17, 18, 22]). The primary motivation behind the research aimed at extending rough set approach is the imperfections of gathered practical application data. In particular, application data often suffer from presence of measurement noise, leading to lack of consistency and resulting difficulty to form data classifications and set approximations of the rough set model. In addition, the data often are real-valued, for example in pattern recognition or control applications, requiring

---

W. Ziarko (✉)

Department of Computer Science, University of Regina, Regina S4S 0A2, Canada  
e-mail: ziarko@sasktel.net

initial preprocessing via a discretization procedure to make it applicable to rough set methodology. This pre-processing however leads to a loss of information and introduces a subjective factor into the method.

The variable precision and Bayesian rough set models are focused on the recognition and modelling of set overlap-based, also referred to as probabilistic, relationships between sets, which are most useful when dealing with noisy data. In this approach, the set-overlap relationships are used to construct approximations of undefinable sets [11]. The primary application of the approach is to the analysis of data co-occurrence-based dependencies in classification tables and probabilistic decision tables derived from data, as discussed in the following sections. Both, the probabilistic decision tables and classification tables are normally “learned” from data to represent some inter-data item connections, typically for the purpose of their analysis or data value prediction. The probabilistic decision tables can also be used as a basis of generalized probabilistic rule induction algorithms [29], but this topic is outside the scope of this chapter.

In practical applications of the data-acquired decision tables, one of the main issues is the identification of a minimal subset of attributes, which are discrete functions of measured features, to represent an identified data dependency without any loss, or with minimal loss, of information. The original general idea of attribute reduct, as introduced by Pawlak [10, 11], is applicable here. However, the original specific notion of reduct is applicable only to functional, or partial functional, data dependencies. In this chapter, we discuss an extended notion of reduct, as defined in the contexts of variable precision and Bayesian rough set models. The notion of reduct in these contexts allows for information-preserving identification of minimal subsets of attributes, in the presence of probabilistic dependencies between attributes.

The chapter is organized as follows. In the next section, we review the fundamentals of the variable precision rough set approach, which include the introduction of set approximations and the presentation of the basics of the related Bayesian rough set model. In Sect. 6.3, we discuss different kinds of probabilistic dependencies occurring between a “target set” and a partition of the universe of interest. The partition is assumed to represent our classification knowledge. The target set is our learning goal, whose approximate classification in terms of the classification knowledge we are trying to learn. The dependencies in question reflect our overall ability to create such a classification. In Sect. 6.4, the probabilistic attribute value-based decision tables are introduced, along with related classification tables. Both kinds of these tables represent our classification knowledge with respect to the target set.

The probabilistic decision tables additionally represent rough approximations of the target set, as defined in the framework of the variable precision rough set theory. The inter-attribute dependencies occurring in both, the probabilistic decision tables and classification tables, are subject of Sect. 6.5. All the discussed dependencies are of probabilistic nature and are either defined in the contexts of variable precision or Bayesian rough set models. They generalize and expand the attribute dependencies introduced by Pawlak in the original rough set theory [11]. Attribute reduction with respect to introduced dependencies is a subject of Sect. 6.6. The monotonicity property of the introduced  $\lambda$ —dependency measure allows for a definition of the notion of

information-preserving reduct with respect to this dependency. Couple of efficient, linear-time algorithms for computing single attribute reducts, either in classification tables or probabilistic decision tables, are presented. The ability to compute reducts allows us also to determine the importance, or significance of attributes. This is the subject of Sect. 6.7. Finally, in Sect. 6.8, we discuss the concept of generalized core attributes, the extension of the original core attributes introduced by Pawlak [10, 11]. The core attributes are the fundamental ones, which are preserved in every attribute reduction.

## 6.2 Variable Precision Rough Sets

In the rough set approach to data analysis, the crucial aspect is the existence of an ability, or knowledge, to form the *prior* classification of the universe of objects of interest into distinct classes. This ability, or *classification knowledge*, is usually associated with an external agent, such as medical professional for example, who is assumed to *know how* to classify objects (for example patients) into categories (for example, into health condition groups). However, in automated systems such an expert typically is not available. Instead, the system has to rely on measurements taken by system sensors (for example, temperature, blood pressure etc.) to perform the classification. In the rough set approach, the measurements are converted into discrete features called *attribute values*, which are then used to classify objects. We elaborate in detail about the attribute value-based classifications in Sect. 6.4.

The general variable precision rough set (VPRS) model does not make any assumptions how the prior classification was performed. It just assumes that some kind of prior knowledge exists and is represented in mathematical form by an equivalence relation, referred to as an indiscernibility relation  $IND$  on the universe  $U$ ,  $IND \subseteq U \times U$ . The relation is assumed to have a finite number of equivalence classes, i.e. classification categories, called elementary sets. It should be noted that the assumption of finite number of classes may not be satisfied in general, but in attribute-value systems, which are the focus of this chapter, it is always the case. The collection of elementary sets of the  $IND$  relation will be denoted as  $IND^*$ . The pair  $(U, IND)$  is called an approximation space.

Let  $X$  be an arbitrary subset, referred to as the *target set*, of the universe  $U$ ,  $X \subseteq U$ . In practice, the universe is a finite non-empty collection of objects of interest, such as medical patients, and the target set is our “goal” class, for example representing the class of patients suffering from a specific disease. Our objective is to create a system which would allow us to classify arbitrary objects into the “goal” class, or its complement, with an error rate which we would consider acceptable in the context of our criteria (which are domain-specific and, consequently, outside of the rough set model), but lower, on average, than in the case of random classification. For example, the objective may be to predict (diagnose) the presence, or absence, of a specific disease based on the results of medical tests, which are supposed to increase the accuracy of such predictions (if tests are properly designed) in comparison to predictions based solely on the frequency of occurrence of the disease in the population.

In the VPRS approach, each equivalence class  $E$  of the indiscernibility relation  $IND$  is assigned two measures which are: the relative “size” of the class  $E$  within universe  $U$ , referred to as the probability  $P(E)$  of  $E$ , and the relative “size” of the target set  $X$  within an elementary set  $E$ , referred to as the conditional probability  $P(X|E)$ . The conditional probability, in this context, is just a measure of the degree of overlap between the target set  $X$  and the elementary set  $E$ . These two measures can be approximated from data respectively by:

$$P(E) = \frac{\text{card}(E)}{\text{card}(U)} \quad (6.1)$$

and

$$P(X|E) = \frac{\text{card}(X \cap E)}{\text{card}(E)} \quad (6.2)$$

where  $\text{card}$  denotes set cardinality.

The target set  $X$  may be undefinable [11], which informally means that, in general, it cannot be expressed as a set union of some elementary sets forming our classification knowledge. That is, in general, the set definability criterion:

$$X = \cup\{E \in IND^* : E \subseteq X\} \quad (6.3)$$

is not satisfied.

This lack of definability is more common than not in applications. The original rough set theory, as introduced by Pawlak [10, 11], deals with this problem via the notions of lower and upper set approximations. However, in many applications, when the target set is not definable, this approach is not sufficient due to the absence of numeric assessments of the degree of association of elementary sets with the target set  $X$ .

The VPRS approach extends the rough set model to make it more flexible, by replacing the full inclusion relation with the overlap relation in the definitions of set approximations. Two precision control parameters called lower limit  $l$  and upper limit  $u$  are used in the definition of lower approximation of the target set  $X$ , or its complement. In this way, one can control the process of computation of approximations of the target set to identify such approximations which satisfy user-imposed criteria, such as for example, characterizing classes of patients with an elevated (or reduced) risk of a disease.

### 6.2.1 Set Approximations in the VPRS Approach

The approximations of the target set in the VPRS approach are defined in terms of unions of some elementary sets, as controlled by lower limit  $l$  and upper limit  $u$  precision parameters.

The notion of prior probability  $P(X)$  plays also an essential role in the definitions of approximations, also called *approximation regions*: it represents the likelihood that a random object  $e \in U$  is a member of the target set  $X$  in the absence of any classification knowledge about the object. If the classification knowledge is available, as represented by the equivalence relation  $IND$ , the likelihood of membership in the set  $X$  of objects belonging to different elementary sets can either increase, or decrease, or stay approximately the same as the prior probability  $P(X)$ . These variations in the set  $X$  membership likelihood across different elementary sets are reflected in the definitions of set approximation regions, which characterize areas of the universe  $U$  with significantly increased, significantly decreased, or approximately unchanged target set  $X$  membership probability.

Each elementary set is classified either into one of approximation regions of the set  $X$ , i.e. a positive region  $POS_u$ , a negative region  $NEG_l$ , or a boundary region  $BND_{l,u}$ . The upper limit  $u$  defines the positive region, or lower approximation, of the target set  $X$ , with the constraint  $0 < P(X) < u \leq 1$ . It represents the least acceptable degree of the conditional probability  $P(X|E)$ , or the set overlap degree, to include the elementary set  $E$  in the positive region. The positive region, or the lower approximation of the target set  $X$ , denoted as  $POS_u$ , is a collection of objects for which the probability of membership in the target set  $X$  is significantly higher than the prior probability  $P(X)$ , where the term *significantly higher* is precisely specified by the parameter  $u$  (as defined by some external criteria):

$$POS_u(X) = \cup\{E : P(X|E) \geq u\}. \quad (6.4)$$

The lower limit  $l$  defines the negative region of the target set  $X$ , with the constraint  $0 \leq l < P(X) < 1$ . It is the highest acceptable degree of the conditional probability  $P(X|E)$  to include the elementary set  $E$  in the negative region. The negative region of the target set  $X$ , denoted as  $NEG_l$  is a collection of objects for which the probability of membership in the target set  $X$  is significantly lower than the prior probability  $P(X)$ , where the term *significantly lower* is precisely specified by the parameter  $l$  (as defined by some external, application-related, criteria):

$$NEG_l(X) = \cup\{E : P(X|E) \leq l\}. \quad (6.5)$$

The boundary region denoted as  $BND_{l,u}$ , is a collection of remaining objects which cannot be classified with sufficient certainty into either positive or negative regions. For the boundary area objects, the probability of membership in the target set  $X$  is not significantly different from the prior probability  $P(X)$ , that is:

$$BND_{l,u}(X) = \cup\{E : l < P(X|E) < u\}. \quad (6.6)$$

Regardless of the choice of lower and upper limit control parameters, the positive and negative approximation regions are subsets of *absolute approximation regions*, as described in the next subsection.

In the Pawlak's rough set model [11], the notion of upper approximation of a set is defined as a union of all elementary sets which have non-empty intersection with the set. The generalized definition of upper approximation  $UPP_l(X)$  in the VPRS approach, as in the original rough set model, is a set union of the positive region and of the boundary region giving:

$$UPP_l(X) = \cup\{E : P(X|E) > l\}. \quad (6.7)$$

Note that the generalized definition coincides with the Pawlak's definition of upper approximation when  $l = 0$ . In addition, when  $u = 1$ , it can be easily demonstrated that the VPRSM definitions of positive, negative and boundary regions, become equivalent to the original rough set model's definitions of lower approximation, negative and boundary regions [11].

One can also note that, in general, as opposed to Pawlak's rough sets, it is not true that  $POS_u(X) \subseteq X$  and it is not true that  $X \subseteq UPP_l(X)$ . Consequently, the rough set cannot be defined in the VPRSM as a pair consisting of upper and lower approximation, as it is done in Pawlak's rough sets [11].

A frequently asked question is to how to set, or tune, the values of the precision control parameters  $l$  and  $u$ . The author's point of view is that apart from the general constraint  $0 \leq l < P(X) < u \leq 1$ , the settings of the parameters are entirely dependent on the requirements of a practical application, while being likely subjective or obtained via the cost-benefit analysis [27].

### 6.2.2 Absolute Set Approximation Regions

To describe the areas of the universe characterized by an unconstrained increase, or decrease of the set  $X$  membership probability, the following definitions of absolute approximation regions are applicable. In this case, no parameters to specify "sufficiently" high increase, or decrease of the set membership probability in those areas are used. We call these areas *absolute approximation regions*.

The *absolute boundary region* of the target set  $X$  is a definable region of the universe  $U$  consisting of those elementary sets which are characterized by the unchanged probability of membership in the target set  $X \subseteq U$ , that is:

$$BND^*(X) = \cup\{E : P(X|E) = P(X)\}. \quad (6.8)$$

As it can be easily verified, in the absolute boundary region, each elementary set  $E$  is probabilistically independent from the target set  $X$ , i.e.  $P(X \cap Y) = P(X)P(Y)$ . Consequently, the whole boundary region is independent from the target set  $X$ . In other words, the objects in the absolute boundary regions can be considered entirely unrelated with the target set.

The region of the universe  $U$  that is characterized by an increased probabilistic connection with the target set  $X \subseteq U$ , relative to the prior probability  $P(X)$ , is called *the absolute positive region* of the set  $X$ , denoted as  $POS^*(X)$ :

$$POS^*(X) = \cup\{E : P(X|E) > P(X)\}. \quad (6.9)$$

In the absolute positive region of  $X$ , the likelihood of an object belonging to the target set is higher than in the whole universe  $U$ , but in practice that increase may be not sufficient from an application perspective.

Similarly, the *absolute negative region*,  $NEG^*(X)$ , of the target set  $X$  is an area of the universe  $U$  characterized by reduced likelihood of an object being a member of the target set  $X$ :

$$NEG^*(X) = \cup\{E : P(X|E) < P(X)\}. \quad (6.10)$$

The above definitions provide the basis of the Bayesian rough set model [13, 30].

## 6.3 Dependencies in Approximation Spaces

The probabilistic connections between elementary sets and the target set, and between definable sets and the target set in the approximation spaces can be quantified by using different dependency measures [24, 33]. Some of these measures are reviewed below.

### 6.3.1 Absolute Certainty Gain

*Absolute certainty gain*, denoted as  $gabs$ , evaluates the degree of one-directional dependency between any two sets. In the simplest case, it is a single-directional dependency measure representing the degree of change of the probability of membership in the set  $X$  for an object belonging to the elementary set  $E$ . The absolute certainty gain is defined by:

$$gabs(X|E) = |P(X|E) - P(X)|, \quad (6.11)$$

where  $|.|$  is the absolute value function.

The above definition can be extended to any definable set  $Y$ . The absolute certainty gain between the subsets  $X$  and  $Y$  can be computed directly from the available probabilistic knowledge based on the formula below, where the summation is over all elementary sets forming the definable set  $Y$ :

$$gabs(X|Y) = \frac{|\sum_{E \subseteq Y} P(E)P(X|E) - P(X)\sum_{E \subseteq Y} P(E)|}{\sum_{E \subseteq Y} P(E)}. \quad (6.12)$$

### 6.3.2 Absolute Dependency Gain

Another dependency measure is an *absolute dependency gain*, which is a bi-directional dependency measure used to evaluate the degree of the two-way connection between any two sets. Given two arbitrary subsets  $X$  and  $Y$  of the universe  $U$ , the absolute dependency gain, denoted as  $dabs(X, Y)$ , is defined by:

$$dabs(X, Y) = |P(X \cap Y) - P(X)P(Y)|. \quad (6.13)$$

The absolute dependency gain reflects the degree of probabilistic dependency between sets  $X$  and  $Y$  by quantifying the amount of deviation from the probabilistic independence between sets  $X$  and  $Y$ , as represented by the product  $P(X)P(Y)$ .

Similar to the absolute certainty gain, in an approximation space  $(U, IND)$ , if a subset  $Y$  is definable, then the absolute dependency gain between the subsets  $X$  and  $Y$  can be computed directly from the available probabilistic knowledge based on the following formula:

$$dabs(X, Y) = |\sum_{E \subseteq Y} P(E)P(X|E) - P(X)\sum_{E \subseteq Y} P(E)|. \quad (6.14)$$

The absolute boundary region of the target set  $X$  can alternatively be expressed by the absolute dependency gain as:

$$BND^*(X) = \cup\{E : dabs(X, E) = 0\}. \quad (6.15)$$

In other words, the absolute boundary region is an area with no dependency gain.

### 6.3.3 Average Dependency Gain

The average, or expected gain function, denoted as  $egabs(X|IND)$ , is a measure of the degree of probabilistic dependency between classification represented by the indiscernibility relation  $IND$  and the classification  $(X, \neg X)$  of the universe  $U$  induced by the target set  $X$ , and its complement  $\neg X$ . It is a measure of dependency between two partitions of the universe  $U$ :

$$egabs(X|IND) = \sum_{E \in IND^*} |P(X \cap E) - P(X)P(E)| = \sum_{E \in IND^*} dabs(X, E). \quad (6.16)$$

When the dependency is functional, i.e. when set  $X$  is definable in Pawlak's sense [11], we have:

$$egabs(X|IND) = \sum_{E \in IND^*} |P(X \cap E) - P(X)P(E)| \quad (6.17)$$

that is:

$$egabs(X|IND) = \sum_{E \in IND^*} P(E)(1 - P(X)) = 1 - P(X) = P(\neg X). \quad (6.18)$$

Similarly,  $egabs(\neg X|IND) = P(X)$  in the functional case.

In the case when  $egabs = 0$ ,  $P(X \cap E) = P(X)P(E)$ , for every elementary set  $E$ . This means that for every elementary set  $E$ ,  $P(X|E) = P(X)$  and  $P(E|X) = P(E)$ . This is equivalent to saying that all elementary sets are probabilistically independent from the target set  $X$ . In practical terms, it means that the occurrence of an object belonging to any of the elementary sets does not affect in any way our ability to guess whether the object is the member of the set  $X$ , or of its complement  $\neg X$ .

## 6.4 Probabilistic Decision Tables

*Probabilistic decision tables* describe classes of approximation space and their probabilistic relations with a target set. They are composed of combinations of attribute values, probability values and approximation region designations.

### 6.4.1 Attributes

In many applications, the information about objects is expressed in terms of values of observations or measurements, often real-valued, referred to as *features*. For the purpose of rough set-based analysis and classifier construction, the feature values are typically mapped into finite-valued numeric or symbolic domains to form composite mappings, referred to as *attributes*. A common kind of mapping is dividing the range of values of a feature into a number of suitably chosen disjoint subranges via a discretization procedure (see, for example, [9]). Formally, an attribute  $a$  is a function on the universe  $U$ ,  $a : U \rightarrow a(U) \subseteq V_a$ , where  $V_a$  is a finite set of values called the *domain* of the attribute  $a$ .

Based on combinations of attributes and their values, a structure of approximation space can be created and analyzed using general notions and results of rough set theory and of the VPRSM. Each attribute defines a classification of the universe  $U$  into classes corresponding to different values of the attribute. Each attribute value  $v \in a(U)$ , corresponds to a set of objects  $E_v^a \subseteq U$  such that  $E_v^a = a^{-1}(v) = \{e \in U : a(e) = v\}$ . The classes  $E_v^a$ , referred to as *a-elementary sets*, form a partition of  $U$ . The equivalence relation corresponding to this partition will be denoted as  $IND_a$ . Similarly, an equivalence relation  $IND_B$ , and the corresponding approximation space, can be defined on the basis of any non-empty set of attributes  $B$ .

### 6.4.2 Decision Tables

A knowledge representation system [11] is a pair  $(U, A)$ , where  $U$  is a universe and  $A$  is a nonempty and finite set of attributes defined on  $U$ . In the context of rough set approach, decision tables are constructed in terms of knowledge representation systems as follows.

Let  $C, D \subset A$  be two disjoint subsets of attributes, called condition and decision attributes, respectively. The condition attributes generate the partitioning of the universe  $U$  into classes of objects having identical values of attributes belonging to  $C$ , thus forming the structure of approximation space on  $U$ . The corresponding collection of elementary sets of this approximation space is denoted by  $U/C$ . Similarly, the decision attributes  $D$  induce a structure of approximation space on  $U$ , with  $U/D$  denoting its elementary sets. The knowledge representation system with defined condition and decision attributes is called a decision table [11]. Decision tables fall into two broad groups: *deterministic decision tables* and *non-deterministic decision tables*.

Deterministic decision tables describe the functional relation between a set of observations (inputs, conditions) and the corresponding decisions (outcomes). In practice, deterministic decision knowledge is not always available. When only some, but not all, decisions can uniquely be determined by combinations of attribute values, the decision table is called non-deterministic. In a non-deterministic decision table, the relationship between conditions and decisions is only partially functional.

Compared to the previous two types of decision tables, which are based on the original rough set theory, a *probabilistic decision table* is developed within the framework of the variable precision rough set theory. It contains some built-in probabilistic measures to help in the process of decision making or prediction in non-deterministic cases.

When defining the probabilistic decision tables, we focus on elementary sets (our target sets) of the decision attribute  $D$ ,  $X \in U/D$ , of the partition generated by the decision attributes.

For a given target set  $X$ , the probabilistic decision table can be defined as a mapping associating each combination of condition attribute values, corresponding to an elementary set  $E \in U/C$ , with a triple of values representing:

1. the unique designation of the rough approximation region (positive, negative, or boundary region),
2. the respective values of the elementary set probability  $P(E)$ , and
3. the conditional probability  $P(X|E)$ .

In practice, when deriving a probabilistic decision table, the measures of  $P(E)$  and  $P(X|E)$  are usually computed based on available data. An example probabilistic decision table is shown in Table 6.1. It should be noted at this point, that while probabilistic decision tables are containing information about set approximation regions of the variable precision rough set model, and consequently depend on the settings of the parameters  $l$  and  $u$ , similar decision tables can be constructed based on Bayesian

**Table 6.1** Probabilistic decision table

$E_i$	$a_1$	$a_2$	$a_3$	Region	$P(E_i)$	$P(X E_i)$
$E_0$	1	1	1	BND	0.0520	0.78
$E_1$	1	1	0	NEG	0.1354	0.02
$E_2$	1	0	1	POS	0.1562	0.99
$E_3$	1	0	0	BND	0.1562	0.36
$E_4$	0	1	1	NEG	0.1406	0.11
$E_5$	0	1	0	BND	0.1093	0.41
$E_6$	0	0	1	NEG	0.1562	0.27
$E_7$	0	0	0	POS	0.0941	0.85

rough set model, using absolute approximation regions. Another related issue is that the probabilistic decision tables can be structured into parent-child linear hierarchies, in which a parent boundary region provides a basis to form an approximation space for the child decision table [31]. In this way, the exponential growth of decision tables caused by the increase in the number of attributes can be effectively controlled without reducing the quality of rough approximations.

#### 6.4.3 Classification Tables

An intermediate step leading to the probabilistic decision table is the creation of the *classification table*, as illustrated in Table 6.2. The classification table associates combinations of condition attribute values, for each elementary set  $E \in U/C$ , with a pair of corresponding  $P(E)$  and  $P(X|E)$  probability measures. In the example Table 6.2, the partitioning of  $U$  is obtained in terms of conditional attributes  $C = \{a_1, a_2, a_3\}$ , with the connected probabilistic measures. The information contained in the classification table can then be used to build rough approximations of any target set  $X \in U/D$ , based on pre-set values of the precision control lower and upper limit parameters  $l$  and  $u$ .

**Table 6.2** Classification table

$E_i$	$a_1$	$a_2$	$a_3$	$P(E_i)$	$P(X E_i)$
$E_0$	1	1	1	0.0520	0.78
$E_1$	1	1	0	0.1354	0.02
$E_2$	1	0	1	0.1562	0.99
$E_3$	1	0	0	0.1562	0.36
$E_4$	0	1	1	0.1406	0.11
$E_5$	0	1	0	0.1093	0.41
$E_6$	0	0	1	0.1562	0.27
$E_7$	0	0	0	0.0941	0.85

Once the approximation region of each elementary set  $E$  was determined, the classification table can be converted into a probabilistic decision table. The creation of the probabilistic decision table involves adding an extra column, technically of a new decision attribute called *Region*, to mark the approximation region designation of each elementary set. The decision table created in that way is fully deterministic with respect to the new *Region* decision attribute which is representing the corresponding three approximation regions: *POS*, *NEG* and *BND*. This is illustrated in the example probabilistic decision Table 6.1, derived from the classification Table 6.2, with  $l = 0.3$  and  $u = 0.8$ .

## 6.5 Dependencies in Decision Tables

In this section, dependencies between attributes occurring in classification tables and probabilistic decision tables are discussed. Specifically, our interest is in the dependencies occurring between condition attributes  $C$ , or their subset, and the two-class classification  $(X, \neg X)$  formed by the target set  $X$  and its complement  $\neg X$ . This classification is numerically represented in both classification and probabilistic decision tables, by values of the conditional probability  $P(X|E)$ . Technically, the columns  $P(E)$  and  $P(X|E)$  can be treated as extra “attributes” associating some real values with elementary sets of the classification generated by condition attributes. In particular, the attribute  $P(X|E)$  describes the distribution of the degrees of association across different elementary sets  $E$  with the target set  $X$ . Consequently, it can be used, in conjunction with the attribute  $P(E)$ , for computing the overall degree of association of the set of condition attributes, or of its subset, with the binary classification of the universe  $U$ , as defined by the target set  $X$  and its complement  $\neg X$ .

In our research, we identified two dependencies, called  $\gamma$ —dependency and  $\lambda$ —dependency, which provide useful measures for evaluating probabilistic decision tables. They also provide criteria for decision table optimization through reduction of redundant condition attributes.

### 6.5.1 Functional and Partial Functional Dependencies

Functional dependencies and partially functional dependencies between attributes of decision tables were originally explored in [11]. We will refer to them as  $\gamma$ —dependencies. They capture the quality of approximation of the target set  $X \in U/D$  in terms of the elementary sets of the approximation space induced by condition attributes. We generalize them within the framework of the VPRS model by defining the  $\gamma$ —dependencies [33] as a relative size of the positive region of the two class partition  $(X, \neg X)$ , subject to prior setting of the values of the control parameters  $l$  and  $u$ :

$$\gamma_{l,u}(X|C) = P(POS_u(X|C) \cup NEG_l(X|C)), \quad (6.19)$$

where  $POS_u(X|C)$  and  $NEG_l(X|C)$ , respectively are positive and negative regions of  $X$  in the approximation space induced on  $U$  by the set of condition attributes  $C$ . This dependency measure reflects the proportion of objects in the universe  $U$  that can be classified as members of the target set  $X$ , or a complement of the target set  $X$ , with sufficient certainty, as given by the parameters  $l$  and  $u$ .

The  $\gamma_{l,u}(X|C)$  measure was inspired by the partial functional dependency measure  $\gamma(D|C)$  introduced by Pawlak [11], which is given as a fraction of objects of the universe  $U$  that can be uniquely classified, based on their condition attributes value combinations, as members of some classes of the decision attribute  $D$ . More precisely, in the VPRS model terms:

$$\gamma(D|C) = \sum_{F \in U/D} P(POS_1(F|C)). \quad (6.20)$$

The above measures play useful role in decision table analysis and reduction of condition attributes.

### 6.5.2 $\lambda$ —Dependency Measure

Another kind of dependency, unrelated to the  $\gamma$ —dependencies measure and conveying different kind of information, is a parametric  $\lambda$ —dependency, denoted as  $\lambda_{l,u}(X|C)$  [33]. It captures the average, or expected degree of the probabilistic connection between elementary sets  $E$  ( $E \in U/C$ ) and the binary classification  $(X, \neg X)$  corresponding to the target set  $X$  and its complement  $\neg X$ . The dependency is defined as a normalized expected degree of deviation of the conditional probability  $P(X|E)$  from the prior probability  $P(X)$ :

$$\lambda_{l,u}(X|C) = \frac{\sum_{E \subseteq POS_u(X|C) \cup NEG_l(X|C)} P(E)|P(X|E) - P(X)|}{2P(X)(1 - P(X))}, \quad (6.21)$$

where  $2P(X)(1 - P(X))$  is a normalization factor equal to the theoretically maximum value of the numerator summation, achievable only when  $X$  is definable in Pawlak's rough set's sense, independent of settings of the parameters  $l$  and  $u$ . The higher the deviation, the stronger the probabilistic connection between conditional attributes  $C$  and the decision partition  $(X, \neg X)$ , and vice versa, with the total probabilistic independence occurring at  $\lambda_{l,u}(X|C) = 0$ .

In the framework of the Bayesian rough set model, the parametric  $\lambda$ —dependency reduces to non-parametric  $\lambda$ —dependency defined as:

$$\lambda(X|C) = \frac{\sum_{E \in U/C} P(E)|P(X|E) - P(X)|}{2P(X)(1 - P(X))}. \quad (6.22)$$

The non-parametric  $\lambda$ —dependency  $\lambda(X|C)$  is a normalized expected degree of deviation of the conditional probability  $P(X|E)$  from the prior probability  $P(X)$ . The main practical advantage of the non-parametric  $\lambda$ —dependency is the absence of any external parameters, which may be difficult to obtain, to compute the dependency. Another useful advantage is its *monotonicity* with respect to condition attributes, as explained in the next section.

## 6.6 $\lambda$ —Dependency-Based Reduct

The application of idea of *reduct*, introduced by Pawlak [10, 11], allows for optimization of representation of classification knowledge by providing a technique for removal of redundant attributes. The concept of reduct generated considerable amount of research interest, primarily as a method for feature selection [1, 2, 6, 8, 12–14, 16, 19–21, 23–25]. The general notion of reduct is applicable to the optimization of classification tables and probabilistic decision tables. The following theorem [13] demonstrates that the  $\lambda$ —dependency measure is *monotonic*, which means that expanding the set of condition attributes  $B \subseteq C$  will not result in the decrease of the dependency level  $\lambda(X|B)$ .

**Theorem 1** *Let  $B \subseteq C$  be a subset of condition attributes on  $U$  and let “ $a$ ” be any condition attribute. Then the following relation holds:*

$$\lambda(X|B) \leq \lambda(X|B \cup \{a\}). \quad (6.23)$$

As a consequence of the Theorem, the notion of the *probabilistic reduct* of attributes  $RED \subseteq C$  can be defined as a minimal subset of attributes preserving the  $\lambda$ —dependency with the target classification  $(X, \neg X)$ .

The reduct satisfies the following two important properties:

$$\lambda(X|RED) = \lambda(X|C) \quad (6.24)$$

and for any attribute  $a \in RED$ :

$$\lambda(X|RED - \{a\}) < \lambda(X|RED). \quad (6.25)$$

The probabilistic reducts, called  $\lambda$ —reducts, can be computed using any methods available for reduct computation in the framework of the Pawlak’s original rough set approach, and in particular, a single  $\lambda$ —reduct can be easily computed from a classification table using the following  $\lambda$ —Reduction algorithm:

**Algorithm 1**  $\lambda$ —*Reduction*:**Step 1:** Let  $Initial\ Dependency \leftarrow \lambda(X|C)$ ;**Step 2:** Arrange condition attributes  $a \in C$  in descending order based on the degree of  $\lambda$ —dependency measure  $\lambda(X|\{a\})$ ;**Step 3:** Starting with the attribute with the lowest  $\lambda$ —dependency degree and proceeding in ascending order, perform the following two steps for all condition attributes:**Step 3.1:** Test the condition  $Initial\ Dependency = \lambda(X|C - \{a\})$ ;**Step 3.2:** If  $Initial\ Dependency = \lambda(X|C - \{a\})$  then eliminate the attribute  $a$  from the set of condition attributes  $C$ ;**Step 4:** The remaining set of condition attributes at the end of the process is a  $\lambda$ —reduct of the initial collection of condition attributes.

In the above algorithm, the condition attributes with the weakest connection with the target classification are eliminated first. Although this technique does not guarantee finding the shortest reduct, it appears to be a reasonable heuristic to find best attributes in the reduct. It should also be noted that the  $\lambda$ —reduct, in general, does not preserve the approximation regions of a target set  $X$ . This means that after computing the  $\lambda$ —reduct of a condition attributes, the approximation regions of a probabilistic decision table have to be re-computed again.

If the preservation of the approximation regions of a probabilistic decision table is of interest, the reduction of condition attributes can be conducted using  $\gamma$ —dependencies measure (Eq. 6.19), which is also monotonic. In this case, any reduct, referred to as  $\gamma$ —reduct, of condition attributes preserving the functional dependency between the condition attributes and the attribute *Region* indicating the approximation region of each elementary set, can be computed. A single  $\gamma$ —reduct can be identified using a variant of  $\lambda$ —Reduction algorithm, referred to as  $\gamma$ —Reduction algorithm:

**Algorithm 2**  $\gamma$ —*Reduction*:**Step 1** Let  $Initial\ Dependency \leftarrow 1$ ;**Step 2** Arrange condition attributes  $a \in C$  in descending order based on the degree of  $\lambda$ —dependency measure  $\lambda(X|\{a\})$ ;**Step 3** Starting with the attribute with the lowest  $\lambda$ —dependency degree and proceeding in ascending order, perform the following two steps for all condition attributes:**Step 3.1** Test the condition  $Initial\ Dependency = \gamma(Region|C - \{a\})$ ;**Step 3.2** If  $Initial\ Dependency = \gamma(Region|C - \{a\})$  then eliminate the attribute  $a$  from the set of condition attributes  $C$ ;**Step 4** The remaining set of condition attributes at the end of the process equals to a  $\gamma$ —reduct of the initial collection of condition attributes of a probabilistic decision table.

## 6.7 Probabilistic Decision Rules

Once the attribute reduct was computed, corresponding classification and decision tables can be formed based on the reduced set of condition attributes. Each row of either of such tables is a probabilistic decision rule with probabilistic “confidence factor” given by  $P(X|E_i)$  attached to it. The “strength” of such a rule is given by the fraction of “supporting” cases, that is,  $P(E_i)$ . For example, the row for the elementary set  $E_2$  of the classification Table 6.2, can be interpreted as a rule:

*if*  $(a_1 = 1) \wedge (a_2 = 0) \wedge (a_3 = 1)$  *then*  $X$  with *confidence* = 0.99 and *strength* = 0.1562.

The rule of this kind gives the likelihood that a new object matching the rule’s preconditions will belong to the target set  $X$ .

Similarly, the probabilistic rules can be computed from probabilistic decision tables. In this case, the target set  $X$  is replaced by either positive, negative or boundary regions. For example, the row for the elementary set  $E_2$  of the classification Table 6.2, can be interpreted as a rule:

*if*  $(a_1 = 1) \wedge (a_2 = 0) \wedge (a_3 = 1)$  *then*  $POS$  with *confidence* = 0.99 and *strength* = 0.1562.

This rule specifies the likelihood that a new object matching the rule’s preconditions will belong to the positive region of the target set  $X$ . Clearly, these rules are dependent on the settings of the precision parameters  $l$  and  $u$ .

If required, the rules based on the probabilistic decision tables can be further simplified (or “generalized”, using machine learning terminology) by removing some unnecessary attribute-value pairs from their preconditions, without affecting their confidence factors. This objective can be accomplished by computing a *value reduct* of attributes [11]. Value reduct was used in some machine learning algorithms based on the rough set theory [31]. However, we will not elaborate more about this comprehensive topic in this chapter as it deserves another chapter of its own.

## 6.8 Significance of $\lambda$ —Reduct Attributes

The  $\lambda$ —*Reduct* provides a method for computing fundamental factors of the  $\lambda$ —dependency.

The attributes appearing in a  $\lambda$ —*reduct* can be evaluated with respect to their contribution to the dependency with the target classification by adopting the notion of a *significance factor*. The significance factor  $sig_{RED}(a)$  of an attribute  $a \in RED$  is a relative decrease of the dependency  $\lambda(X|RED)$  caused by removal of the attribute “ $a$ ” from the reduct:

$$sig_{RED}(a) = \frac{\lambda(X|RED) - \lambda(X|RED - \{a\})}{\lambda(X|RED)}. \quad (6.26)$$

Similarly, the significance of attributes in a probabilistic decision table can be assessed within any  $\gamma$ —*reduct*, using the approach given above.

## 6.9 $\lambda$ —Core Collection of Attributes

As in the original rough set approach [11], one can easily identify the set of most essential condition attributes with respect to the  $\lambda$ —dependency. These attributes, called the  $\lambda$ —core, are the ones which would never be eliminated in the process of any  $\lambda$ —Reduct computation. They are included in all  $\lambda$ —reducts i.e. their collection is equal to the intersection of all  $\lambda$ —reducts.

Any core attribute  $\{a\}$  satisfies the following inequality:

$$\lambda(X|C) > \lambda(X|C - \{a\}). \quad (6.27)$$

The above inequality demonstrates that there is no need to compute all  $\lambda$ —reducts, which is NP-hard, to identify the  $\lambda$ —core as the core attributes can be found by simple linear testing procedure.

As in the case of  $\lambda$ —core attributes,  $\gamma$ —core attributes can also be computed in a probabilistic decision table with respect to the dependency  $\gamma(Region|C)$  by testing the effect of removal of each condition attribute.

## 6.10 Final Remarks

The chapter reviews results of our long-term research on data dependencies, within the frameworks of the variable precision and Bayesian rough set models, occurring in approximation spaces and in both, classification and decision tables. These probabilistic dependencies are defined based on the degrees of overlap between sets. The primary dependency measures discussed in the chapter are  $\gamma$ —dependency and  $\lambda$ —dependency. They generalize and expand the attribute functional and partial functional dependency measures introduced by Pawlak [10, 11]. The applicability of the measures to creation, analysis and optimization of classification and decision tables, via the concept of attribute reduct, was also discussed and two reduct computation algorithms were presented. The variable precision rough set approach was used in many applications since its introduction in 1990s. To our best knowledge, the most comprehensive application, involving the use of hierarchies of probabilistic decision tables and the attribute dependency measures presented in this chapter, were the experiments with face recognition [4]. It is our belief that the theory and methods presented in the chapter will find additional useful applications in areas dealing with large amounts of data such as, for example, in medicine, pattern classification, market analysis and prediction, machine learning and data mining in general, just to mention a few areas where in our opinion this theory is applicable.

**Acknowledgments** Thanks are due to anonymous referees for their detailed and inspiring comments. The research reported in the chapter was supported by research grants from Natural Sciences and Engineering Research Council of Canada.

## References

1. Bac, L., Tuan, N.: Using rough set in feature selection and reduction in face recognition problem. In: Proceedings of the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining PAKDD. Lecture Notes in Artificial Intelligence, vol. 3518, pp. 226–233 (2005)
2. Beynon, M.: Reducts within the variable precision rough sets model: a further investigation. *Eur. J. Oper. Res.* **134**(3), 592–605 (2001)
3. Beynon, M., Peel, M.: Variable precision rough set theory and data discretization: an application to corporate failure prediction. *Int. J. Manag. Sci.* **29**, 561–576 (2001)
4. Chen, X., Ziarko, W.: Rough set-based incremental learning approach to face recognition. In: Proceedings of the International Conference on Rough Sets and Current Trends in Computing. Lecture Notes in Artificial Intelligence, vol. 6086, pp. 356–365 (2010)
5. Greco, S., Matarazzo, B., Slowinski, R.: Multicriteria classification by dominance-based rough set approach. In: Kloesgen, W., Zytkow, J. (eds.) *Handbook of Data Mining and Knowledge Discovery*, chap. C5.1.9. Oxford University Press, New York (2002)
6. Inuiguchi, M., Yoshioka, Y., Kusunoki, Y.: Variable-precision dominance-based rough set approach and attribute reduction. *Int. J. Approx. Reason.* **50**, 1199–1214 (2009)
7. Katzberg, J., Ziarko, W.: Variable precision rough sets with asymmetric bounds. In: Ziarko, W. (ed.) *Proceedings of the International Workshop on Rough Sets, Fuzzy Sets and Knowledge Discovery RSKD*, pp. 167–177. Springer, London (1994)
8. Mi, J., Leung, Y., Wu, W.: Approaches to attribute reduction in concepts lattices induced by axialities. *Knowl. Based Syst.* **23**(6), 504–511 (2010)
9. Nguyen, H.: On exploring soft discretization of continuous attributes. In: Pal, S.K., Polkowski, L., Skowron, A. (eds.) *Rough-Neural Computing: Techniques for Computing with Words, Cognitive Technologies*, pp. 333–350. Springer (2003)
10. Pawlak, Z.: Rough sets. *Int. J. Comput. Inf. Sci.* **11**, 341–356 (1982)
11. Pawlak, Z.: *Rough Sets: Theoretical Aspects of Reasoning About Data*. Kluwer, The Netherlands (1991)
12. Peters, J.F., Ramanna, S.: Feature selection: near set approach. In: Proceedings of the 3rd ECML/PKDD International Workshop on Mining Complex Data MCD, pp. 57–71 (2007)
13. Slezak, D., Ziarko, W.: Attribute reduction in the Bayesian version of variable precision rough set model. *Electron. Notes Theor. Comput. Sci.* **82**(4), 263–273 (2003)
14. Swiniarski, R., Skowron, A.: Rough set methods in feature selection and recognition. *Pattern Recognit. Lett.* **24**(6), 833–849 (2003)
15. Wei, L., Zhang, W.: Probabilistic rough sets characterized by fuzzy sets. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.* **12**, 47–60 (2004)
16. Xia Wang, X., Zhang, W.: Relations of attribute reduction between object and property oriented concept lattices. *Knowl. Based Syst.* **21**(5), 398–403 (2008)
17. Yao, Y.: Decision theoretic rough set models, rough sets and knowledge. In: Proceedings of the 2nd International Conference on Rough Sets and Knowledge Technology RSKT. Lecture Notes in Artificial Intelligence, vol. 4481, pp. 1–12 (2007)
18. Yao, Y., Lin, T.: Generalization of rough sets using modal logic. *Intell. Autom. Soft Comput.* **2**(2), 103–120 (1996)
19. Yao, Y., Zhao, Y.: Discernibility matrix simplification for constructing attribute reducts. *Inf. Sci.* **179**(5), 867–882 (2009)
20. Yao, Y., Zhao, Y., Wang, J.: On reduct construction algorithms. In: Proceedings of the 1st International Conference on Rough Sets and Knowledge Technology RSKT. Lecture Notes in Artificial Intelligence, vol. 4062, pp. 297–304 (2006)
21. Zhang, W., Mi, J., Wu, W.: Approaches to knowledge reductions in inconsistent systems. *Int. J. Intell. Syst.* **18**(9), 989–1000 (2003)
22. Zhang, H., Leung, Y., Zhou, L.: Variable precision-dominance based rough set approach to interval-valued information systems. *Inf. Sci.* **244**, 75–272 (2013)

23. Zhang, J., Wang, J., Li, D., He, H., Sun, J.: A new heuristic reduct algorithm based on rough sets theory. In: Proceedings of the 4th International Conference on Advances in Web-Age Information Management WAIM. Lecture Notes on Computer Science, vol. 2762, pp. 247–253 (2003)
24. Zhao, Y., Luo, F., Wong, S., Yao, Y.: A general definition of an attribute reduct. In: Proceedings of the 2nd International Conference on Rough Sets and Knowledge Technology RSKT, Lecture Notes in Artificial Intelligence, vol. 4481, pp. 101–108 (2007)
25. Zhong, N., Dong, J.: Using rough sets with heuristics for feature selection. *J. Intell. Inf. Syst.* **16**, 199–214 (2001)
26. Ziarko, W.: Variable precision rough sets model. *J. Comput. Syst. Sci.* **46**(1), 39–59 (1993)
27. Ziarko, W.: Decision making with probabilistic decision tables. In: Proceedings of the 7th International Workshop on Rough Sets. Fuzzy Sets, Data Mining and Granular Computing RSFDGrC. Lecture Notes on Artificial Intelligence, pp. 463–471. Springer, Yamaguchi (1999)
28. Ziarko, W.: Probabilistic decision tables in the variable precision rough set model. *Comput. Intell.* **17**(3), 593–603 (2002)
29. Ziarko, W.: Rough set approaches for discovery of rules and attribute dependencies. In: Kloesgen, W., Zytkow, J. (eds.) *Handbook of Data Mining and Knowledge Discovery*, pp. 328–339. Oxford University Press, New York (2002)
30. Ziarko, W.: Set approximation quality measures in the variable precision rough set model. In: Proceedings of the 2nd International Conference on Hybrid Intelligent Systems HIS. Soft Computing Systems, Management and Applications, vol. 87, pp. 442–452. IOS Press (2002)
31. Ziarko, W.: Acquisition of hierarchy—structured probabilistic decision tables, and rules from data. *Expert Syst., Int. J. Knowl. Eng. Neural Netw.* **20**(5), 10–305 (2003)
32. Ziarko, W.: Probabilistic rough sets. In: Proceedings of the 10th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing RSFDGrC. Lecture Notes in Computer Science, vol. 3641, pp. 283–293 (2005)
33. Ziarko, W.: Partition dependencies in hierarchies of probabilistic decision tables. In: Proceedings of the 1st International Conference on Rough Sets and Knowledge Technology RSKT. Lecture Notes in Artificial Intelligence, vol. 4062, pp. 42–49 (2006)
34. Ziarko, W.: Probabilistic approach to rough sets. *Int. J. Approx. Reason.* **49**(2), 272–284 (2008)
35. Ziarko, W.: Probabilistic Dependencies in Linear Hierarchies of Decision Tables. *Transactions on Rough Sets 9*, vol. 5390, pp. 444–454 (2008)

# Chapter 7

## Structure-Based Attribute Reduction: A Rough Set Approach

**Yoshifumi Kusunoki and Masahiro Inuiguchi**

**Abstract** We provide an introduction to a rough set approach to attribute reduction. Analyzed data sets consist of objects which are described by attributes and partitioned into decision classes. Rough set theory deals with uncertainty decision classes with respect to attributes by approximating them to precise sets. The aim of attribute reduction is to remove redundant attributes as well as find important ones for classification. Several types of attribute reduction have been proposed especially according to preserving structures of approximated decision classes. We introduce definitions and theoretical results about structures-based attribute reduction.

**Keywords** Rough set model · Reduct · Boolean function · Structure-based reduct

### 7.1 Introduction

We provide an introduction to attribute reduction or feature selection based on rough set theory [35, 36, 39]. Rough set theory approaches uncertainty or inconsistency of membership for sets due to incomplete or granular information. In a rough set approach for data analysis, data sets are usually given by decision tables which consist of objects (items) described by attributes. Moreover, each object in decision tables is classified into decision classes. Because of incompleteness of given attributes, some objects are indiscernible to each other by the attributes, and that causes uncertainty of decision classes. Such an uncertain decision class is approximated by two precise sets, called lower and upper approximations. The difference of the upper and lower approximations is called a boundary.

---

Y. Kusunoki (✉)

Graduate School of Engineering, Osaka University, 2-1 Yamadaoka,  
565-0871 Suita, Osaka, Japan  
e-mail: kusunoki@eei.eng.osaka-u.ac.jp

M. Inuiguchi

Graduate School of Engineering Sciences, Osaka University, 1-3 Machikaneyama,  
560-8531 Toyonaka, Osaka, Japan  
e-mail: inuiguti@sys.es.osaka-u.ac.jp

One of the major topics for rough set based data analysis is (relative) attribute reduction [37, 39]. Attribute reduction is a problem to delete redundant condition (explanatory) attributes for the classification of the decision classes. Minimal sets of attributes preserving a part of information of the classification are called reducts. Reducts can be interpreted as important sets of attributes for the classification. Several types of reducts have been proposed according to a part of the information which should be preserved [23, 36, 43, 45]. Originally, Pawlak proposed reducts preserving the positive region [36, 43], which is the union of all lower approximations of decision classes, in other words, the set of all certainly classified objects. Ślęzak proposed ones preserving all boundaries of decision classes [45]. One of the authors also proposed two types of reducts, which preserve all lower approximations and all upper approximations of decision classes, and show that they are equivalent to reducts preserving the positive region and all boundaries, respectively [23].

Inspired by the above studies, we provide a framework to discuss attribute reduction in the rough set theory. We regard attribute reduction as removing condition attributes with preserving some part of the lower/upper approximations of the decision classes, because the approximations summarize the classification ability of the condition attributes. Hence, we define several types of reducts according to structures of the approximations [23, 24]. They are called “structure-based” reducts.

When several types of structure-based reducts are defined, we would be interested in whether one reduct is stronger/weaker than another reduct, in other words, one preserves more/less structure than the other. Therefore, we have investigated the strong-weak relation among different types of structure-based reducts. As a result of the investigation, we obtain a strong-weak hierarchy of structure-based reducts. The strong-weak hierarchy is useful when we search the best reduct for an application, because it provides a trade-off between the size (cost for precise classification) of a reduct and its classification ability. It is an advantage of the variations of structure-based reducts.

The rough set model is extended to apply to various kinds of data sets [12, 16, 22, 29, 38, 43, 47, 53, 54]. Two important extensions of the rough set model are the variable precision rough set model [53, 54] and the dominance-based rough set model [16]. The variable precision rough set model is a probabilistic extension. Given precision parameters, requirements for lower and upper approximations are relaxed to tolerate errors in decision tables. The dominance-based rough set model is applied to decision tables with ordinal attributes, where decision classes are ordered and monotonically depend on the ordinal attributes. It deals with inconsistency between the classification of the ordinal decision classes and the monotonic dependence. Instead of decision classes, upward unions and downward unions of decision classes are approximated. In the extended rough set models, we have studied structure-based reducts [20, 21, 25, 26, 31].

In the classical rough set model, it is well-known that reducts are associated with prime implicants of a Boolean function [37, 43]. We can efficiently enumerate reducts by converting it to enumerating prime implicants of the Boolean function. Like that conversion, the methodology solving a problem by solutions of a Boolean equation is called Boolean reasoning [37, 43]. In this chapter, we propose a unified

formulation of several Boolean functions corresponding to several types of reducts in the classical and extended rough set models.

In this chapter, we show our theoretical results of structure-based reducts in the classical and extended rough set models, including definitions of reducts and their strong-weak hierarchy. The results consist of our papers [20, 23, 25, 31]. Our main contributions are to propose structure-based reducts, investigate strong-weak relations of reducts, and connect reducts with prime implicants of Boolean functions in a unified formulation in the variable precision and dominance-based rough set models. For the structure-based reducts in the variable precision rough set model, we revise their definitions from our previous work [20]. Parts of the results were independently developed by other authors [33, 45, 49, 52].

This chapter is organized as follows. In Sect. 7.2, we study structure-based reducts in the classical rough set models. Firstly, we define a decision table and the rough set model of the decision table. Then, we introduce several types of reducts including structure-based reducts and others. We show that all types of reducts are reduced to two different types. Finally, we connect all reducts of each type with the prime implicants of a specific Boolean function. Sections 7.3 and 7.4 are devoted to structure-based reducts in the variable precision rough set model and those in the dominance-based rough set model, respectively. Those sections have almost the same organization as that of Sect. 7.2, namely, defining a rough set model and reducts, investigating strong-weak relations of reducts, and connecting reducts with prime implicants of Boolean functions. Concluding remarks are given in Sect. 7.5.

## 7.2 Structure-Based Attribute Reduction in Rough Set Models

### 7.2.1 Decision Tables

In rough set theory, analysed data sets form decision tables [36, 39]. A decision table is defined by  $\mathbb{D} = (U, AT = C \cup \{d\}, \{V_a\}_{a \in AT})$ .<sup>1</sup>  $U$  is a finite set of objects.  $AT$  is a finite set of attributes.  $V$  is a set of attribute values. Each attribute  $a \in AT$  is a function  $a : U \rightarrow V_a$ , where  $V_a \subseteq V$  is a set of values for  $a$ . For an object  $u \in U$  and an attribute  $a \in AT$ ,  $a(u)$  is the value of  $u$  with respect to  $a$ . For  $A = \{a_{i_1}, a_{i_2}, \dots, a_{i_k}\} \subseteq AT$ ,  $V_A$  is the Cartesian product of  $\{V_{a_{i_l}}\}_{l=1,2,\dots,k}$ , namely,  $V_A = \prod_{a_{i_l} \in A} V_{a_{i_l}} = \{(v_{i_1}, v_{i_2}, \dots, v_{i_k}) \mid v_{i_l} \in V_{a_{i_l}}, l = 1, 2, \dots, k\}$ .  $A(u)$  is the tuple of the values of  $u$  with respect to  $A$ , namely,  $A(u) = (a_{i_1}(u), a_{i_2}(u), \dots, a_{i_k}(u))$ . The attribute set  $AT$  is divided into a condition attribute set  $C$  and a decision attribute  $d$  to investigate the dependency of the decision attribute on condition attributes or the causal effect of condition attributes on the decision attribute. Throughout this chapter, we consider that the objects and the condition attributes are indexed by

---

<sup>1</sup> A decision table is often defined by the finite set of objects  $U$  and the finite set of attributes  $AT$ , i.e.,  $(U, AT)$ , however we use that definition to clarify the sets of values for the attributes.

$U = \{u_1, u_2, \dots, u_n\}$  and  $C = \{c_1, c_2, \dots, c_m\}$ , where  $n = |U|$  and  $m = |C|$ . Moreover, we define the decision attribute values as  $V_d = \{1, 2, \dots, p\}$ .

*Remark 1* Decision tables are identical to data sets or data tables for the classification problem or the supervised learning in the data mining or machine learning literature, in which condition attributes are called attributes or independent variables, the decision attribute is a class attribute or dependent variable, and objects are tuples or samples. In that literature, each object is given by a tuple of attribute values with a class label (decision attribute value). However, we use the form of decision tables in this chapter by two reason. One is that we often deal with subsets of the attributes, so we prefer to let the symbols of the attributes be explicit. The other is to emphasise the view that a relation (e.g. the equivalence relation) on the object set is induced from a structure of the attribute value space (e.g. equivalence of values) through the attributes (functions).

*Example 1* Consider a decision table  $\mathbb{D} = (U, C \cup \{d\}, \{V_a\})$  about car evaluations in Table 7.1, where  $U = \{u_1, u_2, \dots, u_7\}$ ,  $C = \{\text{Pr, Ma, Sa}\}$  and  $d = \text{Ev}$ . The attribute value sets are given by  $V_{\text{Pr}} = V_{\text{Ma}} = V_{\text{Sa}} = \{\text{low, med, high}\}$ ,  $V_{\text{Ev}} = \{\text{unacc, acc, good}\}$ . Condition attributes Pr, Ma, and Sa indicate price, maintenance cost, and safety of a car, respectively, by values high, med (medium), and low. Decision attribute Ev means evaluation of a car by some customer(s).

The value of  $u_1$  with respect to Pr is  $\text{Pr}(u_1) = \text{high}$ , and that of  $u_2$  with respect to Ev is  $\text{Ev}(u_2) = \text{unacc}$ . The value tuple of  $u_4$  with respect to  $C = \{\text{Pr, Ma, Sa}\}$  is  $C(u_4) = (\text{med, high, low})$ .

Given an attribute subset  $A \subseteq AT$ , we define an indiscernibility relation on  $U$  with respect to  $A$ , denoted by  $R_A$ , as follows:

$$R_A = \{(u, u') \in U^2 \mid a(u) = a(u'), \text{ for any } a \in A\}.$$

$R_A$  is the set of the object pairs each of which is indiscernible by the given attributes  $A$ . Obviously,  $R_A$  is an equivalence relation, which is reflexive, symmetric, and transitive. From  $R_A$ , we define the equivalence class of an object  $u \in U$ , denoted by  $R_A(u)$ , as follows:

$$R_A(u) = \{u' \in U \mid (u', u) \in R_A\}.$$

**Table 7.1** Decision table of car evaluations

Car	Pr	Ma	Sa	Ev
$u_1$	High	High	Low	Unacc
$u_2$	Med	Med	Med	Unacc
$u_3$	Med	Med	Med	Acc
$u_4$	Med	High	Low	Acc
$u_5$	Med	Med	High	Acc
$u_6$	Med	Med	High	Good
$u_7$	Low	Med	Med	Good

$R_A(u)$  is the set of objects which have the same values as  $u$  for all attributes in  $A$ . We denote the set of all equivalence classes with respect to  $R_A$  by  $U/R_A = \{R_A(u) \mid u \in U\}$ . Every equivalence class with respect to the decision attribute  $d$  is called a decision class. For each value of the decision attribute  $i \in V_d$ , we define the corresponding decision class  $X_i = \{u \in U \mid d(u) = i\}$ . Clearly,  $\mathcal{X} = \{X_1, X_2, \dots, X_p\}$  forms a partition of  $U$ .

*Example 2* Remember  $\mathbb{D} = (U, C \cup \{d\}, \{V_d\})$  in Table 7.1. Let  $A = \{\text{Pr}, \text{Ma}\}$  be an attribute subset. The discernibility relation  $R_A$  is described as the following matrix. Symbol \* indicates that the corresponding object pair  $u_i$  and  $u_j$  is in the discernibility relation, i.e.,  $(u_i, u_j) \in R_A$ .

	$u_1$	$u_2$	$u_3$	$u_4$	$u_5$	$u_6$	$u_7$
$u_1$	*						
$u_2$	*	*		*	*		
$u_3$	*	*		*	*		
$u_4$			*				
$u_5$	*	*		*	*		
$u_6$	*	*		*	*		
$u_7$						*	

From the matrix, we can easily see that the equivalence classes by  $R_A$  form a partition of  $U$ , namely,  $U/R_A = \{\{u_1\}, \{u_4\}, \{u_7\}, \{u_2, u_3, u_5, u_6\}\}$ .

The decision classes of the decision table  $\mathbb{D}$  are obtained as  $X_{\text{unacc}} = \{u_1, u_2\}$ ,  $X_{\text{acc}} = \{u_3, u_4, u_5\}$ ,  $X_{\text{good}} = \{u_6, u_7\}$ .

### 7.2.2 Rough Set Models

Let  $A$  be a subset of the attribute set  $AT$  and  $X$  be a subset of the object set  $U$ . When  $X$  can be represented by a union of elements in  $U/R_A$ , we can say that the classification by  $X$  is consistent with the information of  $A$ . Such subsets of objects are called definable sets with respect to  $A$ . On the other hand, considering an object subset  $X$  which cannot be represented by any union of elements in  $U/R_A$ , the classification of  $X$  is inconsistent with  $A$ . The classical Rough Set Model (RSM) [35, 36, 39] deals with the inconsistency by two operators for object sets, called lower and upper approximations. For  $A \subseteq AT$  and  $X \subseteq U$ , the lower approximation  $\text{LA}_A(X)$  and the upper approximation  $\text{UA}_A(X)$  of  $X$  with respect to  $A$  is defined by:

$$\begin{aligned}\text{LA}_A(X) &= \{u \in U \mid R_A(u) \subseteq X\}, \\ \text{UA}_A(X) &= \{u \in U \mid R_A(u) \cap X \neq \emptyset\}.\end{aligned}$$

The difference between  $\text{UA}_A(X)$  and  $\text{LA}_A(X)$  is called the boundary of  $X$  with respect to  $A$ , which is defined by:

$$\text{BN}_A(X) = \text{UA}_A(X) \setminus \text{LA}_A(X).$$

$\text{LA}_A(X)$  is interpreted as the set of objects which are certainly classified to  $X$  in view of  $A$ . While,  $\text{UA}_A(X)$  is the set of objects which are possibly classified to  $X$  in view of  $A$ .  $\text{BN}_A(X)$  is a set of objects whose membership to  $X$  is doubtful.

The approximations are a definable set with respect to  $A$ , where a definable set with respect to  $A$  is a set defined by the union of elements in  $U/R_A$ :

$$\begin{aligned}\text{LA}_A(X) &= \bigcup_{R_A(u) \subseteq X} R_A(u) = \bigcup_{u \in \text{LA}_A(X)} R_A(u), \\ \text{UA}_A(X) &= \bigcup_{R_A(u) \cap X \neq \emptyset} R_A(u) = \bigcup_{u \in \text{UA}_A(X)} R_A(u).\end{aligned}$$

The boundary is necessarily definable because  $U/R_A$  is the partition of  $U$ . In fact,  $\text{LA}_A(X)$  and  $\text{UA}_A(X)$  are “lower” and “upper” approximations of  $X_i$ :

$$\text{LA}_A(X) \subseteq X \subseteq \text{UA}_A(X). \quad (7.1)$$

By the above inclusion relations and the definition of the boundary, it holds that

$$\text{LA}_A(X) = X \setminus \text{BN}_A(X), \quad (7.2)$$

$$\text{UA}_A(X) = X \cup \text{BN}_A(X). \quad (7.3)$$

For  $B \subset A \subseteq AT$ , we have,

$$\text{LA}_B(X) \subseteq \text{LA}_A(X) \text{ and } \text{UA}_B(X) \supseteq \text{UA}_A(X). \quad (7.4)$$

When  $B$  is included in  $A$ , the approximations with respect to  $B$  are coarser than those with respect to  $A$ . It means that dropping some attributes, i.e., information, decline the accuracy of RSM.

So far, we have defined approximations of  $X$  from the lower and upper definable sets. We can approximate the partition  $X$  and  $U \setminus X$  by three definable sets. They are called positive, boundary, and negative regions of  $X$  with respect to  $A$ , denoted by  $\text{POS}_A(X)$ ,  $\text{BND}_A(X)$ , and  $\text{NEG}_A(X)$ , respectively:

$$\text{POS}_A(X) = \bigcup \{E \in U/R_A \mid E \subseteq X\},$$

$$\text{BND}_A(X) = \bigcup \{E \in U/R_A \mid E \cap X \neq \emptyset \text{ and } E \cap U \setminus X \neq \emptyset\},$$

$$\text{NEG}_A(X) = \bigcup \{E \in U/R_A \mid E \subseteq U \setminus X\}.$$

$\text{POS}_A(X)$  is the union of elements in  $U/R_A$  which are completely included in  $X$ , while  $\text{NEG}_A(X)$  is the union of elements in  $U/R_A$  which are completely excluded from  $X$ .  $\text{BND}_A(X)$  is the union of the rest of elements in  $U/R_A$ . Clearly,  $\text{POS}_A(X)$ ,  $\text{BND}_A(X)$ , and  $\text{NEG}_A(X)$  form a partition of  $U$ . We can easily see the following correspondence:

$$\begin{aligned}\text{POS}_A(X) &= \text{LA}_A(X), \\ \text{BND}_A(X) &= \text{BN}_A(X), \\ \text{NEG}_A(X) &= U \setminus \text{UA}_A(X).\end{aligned}$$

In the rest of this section, we consider RSM for decision tables, namely, we only deal with approximations of decision classes  $\mathcal{X} = \{X_1, X_2, \dots, X_p\}$  with respect to subsets of condition attributes  $A \subseteq C$ .

*Example 3* Remember the decision classes  $X_{\text{unacc}} = \{u_1, u_2\}$ ,  $X_{\text{acc}} = \{u_3, u_4, u_5\}$  and  $X_{\text{good}} = \{u_6, u_7\}$  of the decision table in Table 7.1. The lower and upper approximations with respect to  $C$  of  $X_{\text{unacc}}$ ,  $X_{\text{cc}}$  and  $X_{\text{good}}$  are obtained as follows:

$$\begin{aligned}\text{LA}_C(X_{\text{unacc}}) &= \{u_1\}, \quad \text{UA}_C(X_{\text{unacc}}) = \{u_1, u_2, u_3\}, \\ \text{LA}_C(X_{\text{acc}}) &= \{u_4\}, \quad \text{UA}_C(X_{\text{acc}}) = \{u_2, u_3, u_4, u_5, u_6\}, \\ \text{LA}_C(X_{\text{good}}) &= \{u_7\}, \quad \text{UA}_C(X_{\text{good}}) = \{u_5, u_6, u_7\}.\end{aligned}$$

We can see that  $\text{LA}_C(X_i) \subseteq X_i \subseteq \text{UA}_C(X_i)$  for each  $i = \text{unacc}, \text{acc}, \text{good}$ . Moreover, we can also see that each approximation is the union of equivalence classes included in the approximation, e.g.,  $\text{UA}_C(X_{\text{acc}}) = \{u_2, u_3\} \cup \{u_4\} \cup \{u_5, u_6\}$ .

We reduce condition attributes to  $A = \{\text{Pr}\}$ . The approximations become:

$$\begin{aligned}\text{LA}_A(X_{\text{unacc}}) &= \{u_1\}, \quad \text{UA}_A(X_{\text{unacc}}) = \{u_1, u_2, u_3, u_4, u_5, u_6\}, \\ \text{LA}_A(X_{\text{acc}}) &= \emptyset, \quad \text{UA}_A(X_{\text{acc}}) = \{u_2, u_3, u_4, u_5, u_6\}, \\ \text{LA}_A(X_{\text{good}}) &= \{u_7\}, \quad \text{UA}_A(X_{\text{good}}) = \{u_2, u_3, u_4, u_5, u_6, u_7\}.\end{aligned}$$

The approximations with respect to  $A$  are coarser than those with respect to  $C$ , namely,  $\text{LA}_A(X_i) \subseteq \text{LA}_C(X_i)$  and  $\text{UA}_A(X_i) \supseteq \text{UA}_C(X_i)$  for each  $i = \text{unacc}, \text{acc}, \text{good}$ .

For every  $X_i$ , the lower approximation  $\text{LA}_A(X_i)$  and the boundary  $\text{BN}_A(X_i)$  can be represented using all upper approximations of decision classes  $\text{UA}_A(X_1), \text{UA}_A(X_2), \dots, \text{UA}_A(X_p)$ :

$$\text{LA}_A(X_i) = \text{UA}_A(X_i) \setminus \bigcup_{j \in V_d \setminus \{i\}} \text{UA}_A(X_j), \tag{7.5}$$

$$\text{BN}_A(X_i) = \text{UA}_A(X_i) \cap \bigcup_{j \in V_d \setminus \{i\}} \text{UA}_A(X_j). \tag{7.6}$$

All upper approximations form a cover of  $U$ :

$$U = \bigcup_{i \in V_d} \text{UA}_A(X_i). \tag{7.7}$$

A positive region with respect to  $A \subseteq C$  is also defined for the decision attribute  $d$  or equivalently for the decision table  $\mathbb{D}$ . It is the union of all positive regions of decision classes, i.e., the set of objects which are certainly classified to exactly one of the decision classes:

$$\text{POS}_A(d) = \bigcup_{i \in V_d} \text{POS}_A(X_i).$$

A generalized decision function [1, 45] with respect to  $A \subseteq C$ , denoted by  $\partial_A : U \rightarrow 2^{V_d}$ , provides a useful representation of RSM. For  $u \in U$ ,  $\partial_A(u)$  is a set of decision attribute values or decision classes to which  $u$  is possibly classified:

$$\partial_A(u) = \{i \in V_d \mid X_i \cap R_A(u) \neq \emptyset\}.$$

The generalized decision function gives an object-wise view of RSM. The lower and upper approximations can be expressed by the generalized decision function:

$$\begin{aligned}\text{LA}_A(X_i) &= \{u \in U \mid \partial_A(u) = \{i\}\}, \\ \text{UA}_A(X_i) &= \{u \in U \mid \partial_A(u) \ni i\}.\end{aligned}$$

Because  $\partial_A(u)$  is defined based on  $R_A(u)$ , we have

$$\partial_A(u) = \partial_A(u') \text{ if } (u, u') \in R_A,$$

and because each object  $u$  is included in at least one upper approximation, we have

$$\partial_A(u) \neq \emptyset.$$

The monotonic property of upper approximations is represented as:

$$B \subseteq A \Rightarrow \partial_B(u) \supseteq \partial_A(u) \quad \text{for all } u \in U.$$

*Example 4* Remember  $\mathbb{D} = (U, C \cup \{d\}, \{V_a\})$  in Table 7.1. The generalized decision function  $\partial_C$  is obtained as follows.

$$\begin{aligned}\partial_C(u_1) &= \{\text{unacc}\}, & \partial_C(u_2) &= \partial_C(u_3) = \{\text{unacc, acc}\}, \\ \partial_C(u_4) &= \{\text{acc}\}, & \partial_C(u_5) &= \partial_C(u_6) = \{\text{acc, good}\}, \\ \partial_C(u_7) &= \{\text{good}\}.\end{aligned}$$

For  $A \subseteq C$ , a quality of classification (or quality of approximation) of the decision attribute  $d$  with respect to  $A$  is defined by:

$$\gamma_A(d) = \frac{|\text{POS}_A(d)|}{|U|}. \tag{7.8}$$

It measures to what degree objects are correctly classified by RSM.

### 7.2.3 Reducts in Rough Set Models

#### 7.2.3.1 Preserving Positive Region, Quality, and Generalized Decisions

Attribute reduction is to find important subsets of condition attributes by dropping as many as possible other condition attributes while preserving some specific information of RSM for a decision table  $\mathbb{D}$ . A minimal subset of condition attributes preserving the information is called a relative (or decision) reduct. In this chapter, we call it “reduct” for short. Reducts are originally defined to preserve the positive region  $\text{POS}_C(d)$  [36, 43].

**Definition 1** ([36, 43]) A reduct is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\text{POS}_A(d) = \text{POS}_C(d). \quad (\text{P})$$

Here, the minimality is defined in terms of the set inclusion, i.e., there is no proper subset  $A' \subset A$  satisfying (P).

A condition attribute subset  $A$  satisfying (P) preserves the information of the certain classification in the decision table. Generally, there exist more than one reduct in a decision table. The intersection of all reducts is called the core. Every element in the core is an essential condition attribute to preserve the information of  $\text{POS}_C(d)$ . The core can be empty. On the other hand, the condition attributes which do not belong to any reducts can be dropped without deterioration of the information. We call the original reduct a P-reduct.

*Remark 2* Condition (P) is monotonic with respect to the set inclusion of condition attributes, i.e., for  $A' \subseteq A \subseteq C$  we have  $\text{POS}_{A'}(d) = \text{POS}_C(d)$  implies  $\text{POS}_A(d) = \text{POS}_C(d)$ . Hence, the above minimality condition for  $A$  is equivalent to that there is no condition attribute  $a \in A$  such that  $A \setminus \{a\}$  satisfies (P).

*Example 5* Remember  $\mathbb{D} = (U, C \cup \{d\}, \{V_a\})$  in Table 7.1. The set of all condition attributes  $C$  obviously satisfies condition (P), but it is not a P-reduct because a proper subset  $A = \{\text{Pr}, \text{Ma}\}$  satisfies (P). On the other hand,  $A$  is a P-reduct because all of proper subsets of  $A$  do not preserve the positive region:  $\text{POS}_{\{\text{Pr}\}}(d) = \{u_1, u_7\}$  and  $\text{POS}_{\{\text{Ma}\}}(d) = \text{POS}_{\emptyset}(d) = \emptyset$ .

We can define another kind of reducts preserving the quality of classification [23, 39, 40].

**Definition 2** ([40]) A Q-reduct is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\gamma_A(d) = \gamma_C(d). \quad (\text{Q})$$

Clearly, condition (P) implies (Q). In RSM, the inverse is also true, because the monotonic property of  $\text{POS}(d)$  holds, namely, for  $A' \subseteq A$ ,  $\text{POS}_{A'}(d) \subseteq \text{POS}_A(d)$ . We also call Q-reducts measure-based reducts, because it preserves a predefined measure for information of RSM.

Bazan et al. [1] and Ślęzak [45] proposed reducts preserving the generalized decision function  $\partial$ .

**Definition 3** ([1, 45]) A G-reduct is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\partial_A(u) = \partial_C(u) \text{ for all } u \in U. \quad (\text{G})$$

As shown in the next section, condition (G) implies (P).

### 7.2.3.2 Structure-Based Reducts

Structure-based reducts, proposed by one of the authors [20, 23], are defined to preserve families of object sets (structures) which are composed of lower and upper approximations, or positive, boundary, and negative regions. Hence, reducts of condition (P) can be seen as structure-based.

Now, we introduce structure-based reducts proposed in [23]. First, we define a reduct preserving all lower approximations. The preservation of the lower approximations implies the sustenance of certain classification ability.

**Definition 4** ([23]) An L-reduct is a minimal condition attribute subset  $A \subseteq C$  preserving the following condition:

$$\text{LA}_A(X_i) = \text{LA}_C(X_i) \text{ for all } i \in V_d. \quad (\text{L})$$

Clearly, condition (L) implies (P) as well as (Q). In RSM, the inverse is also true, because the lower approximations  $\text{LA}(X_i)$ ,  $i = 1, 2, \dots, p$  have the empty intersection with each other, and they are monotonically decreasing with respect to the set inclusion of condition attributes.

However, even if we preserve lower approximations  $\text{LA}_C(X_i)$ ,  $i = 1, 2, \dots, p$ , we may lose the information of boundaries  $\text{BN}_A(X_i)$ ,  $i = 1, 2, \dots, p$  and the information of upper approximations  $\text{UA}_A(X_i)$ ,  $i = 1, 2, \dots, p$ .

Ślęzak [45] proposed a type of reducts preserving all boundaries.

**Definition 5** ([45]) A B-reduct is a minimal condition attribute subset  $A \subseteq C$  preserving the following condition:

$$\text{BN}_A(X_i) = \text{BN}_C(X_i) \quad \text{for all } i \in V_d. \quad (\text{B})$$

The preservation of boundaries implies the protection against uncertainty expansion. Ślęzak [45] also showed that condition (B) is equivalent to (G). Hence, we have that a B-reduct is a G-reduct and vice versa.

On the other hand, we proposed a reduct preserving all upper approximations [23].

**Definition 6** ([23]) A U-reduct is a minimal condition attribute subset  $A \subseteq C$  preserving the following condition:

$$\text{UA}_A(X_i) = \text{UA}_C(X_i) \quad \text{for all } i \in V_d. \quad (\text{U})$$

By definition, the classification ability of the upper approximations is equal to that of the generalized decision function. Hence, condition (U) is equivalent to (G), and we have that a U-reduct is a G-reduct and vice versa.

From Eqs. (7.2) and (7.5), we know that lower approximations are obtained from upper approximations as well as from boundaries. This fact implies that each of the preservation of all upper approximations or the preservation of all boundaries entails the preservation of all lower approximations.

To sum up the above discussion, we have the next theorem.

**Theorem 1** ([23, 45]) Let  $A$  be a subset of  $C$ . We have the following statements.

- (a)  $A$  is a  $Q$ -reduct if and only if  $A$  is an  $L$ -reduct.
- (b)  $A$  is a  $P$ -reduct if and only if  $A$  is an  $L$ -reduct.
- (c)  $A$  is a  $G$ -reduct if and only if  $A$  is a  $U$ -reduct.
- (d)  $A$  is a  $B$ -reduct if and only if  $A$  is a  $U$ -reduct.
- (e)  $A$  is a  $U$ -reduct as well as  $B$ -reduct, then  $A$  satisfies condition (L).

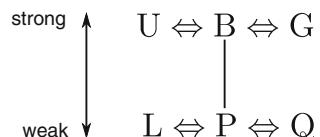
All statements in the theorem can be easily proved by the equations which appeared in Sect. 7.2.2. For example, to prove Theorem 1(d), we show that preserving all boundaries implies preserving all upper approximations by Eq. (7.3), and show the converse by Eq. (7.6).

From Theorem 1(e), if  $A$  is a  $U$ -reduct then there exists an  $L$ -reduct  $B \subseteq A$ . Note that the converse is not always true, i.e., for an  $L$ -reduct  $B$ , there is no guarantee that there exists a  $U$ -reduct  $A \supseteq B$ .

The relations of 6 types of reducts are depicted in Fig. 7.1. Reducts located in the upper part of the figure preserve regions much more. Therefore, such reducts are larger in the sense of the set inclusion than the other reducts located in the lower part. A line segment connecting two types of reducts implies that, for each reduct of the upper type say  $A$  satisfies the preserving condition of the reduct of the lower one. From the figure, we know that there are 2 different types of reducts:  $U$ -reducts and  $L$ -reducts, and  $U$ -reducts are stronger than  $L$ -reducts.

**Remark 3** As shown in Theorem 1, the six types of reducts are reduced to two types. However, it is important to define all possible types of reducts and organize them because of two reasons. One is that when we should mention different definitions of

**Fig. 7.1** Strong-weak hierarchy of 6 types of reducts in RSM



reducts (e.g. different authors would give different definitions), we can easily quote the equivalent of them from here. The other is that equivalent reducts (e.g. U-reducts and B-reducts) in RSM could become different in an extended RSM (e.g. variable precision RSM).

*Remark 4* From the discussion above, we know that a U-reduct preserves more information than an L-reduct. However, when  $p = 2$ , we have the following relation:

$$\text{UA}_A(X_1) = U \setminus \text{LA}_A(X_2), \quad \text{UA}_A(X_2) = U \setminus \text{LA}_A(X_1).$$

Namely, we obtain upper approximations from lower approximations. Hence, in that case, an L-reduct is a U-reduct.

*Remark 5* From Theorem 1, we see that preserving the measure  $\gamma$  is equivalent to preserving the lower approximations. Contrary, we can define a measure preserving which is equivalent to preserving the upper approximations. For example [23], we define,

$$\sigma_A(d) = \frac{\sum_{i \in V_d} |U \setminus \text{UA}_A(X_i)|}{(p-1)|U|},$$

then  $\sigma_A(d) = \sigma_C(d)$  is same as condition (U).

### 7.2.4 Boolean Functions Representing Reducts

Boolean reasoning [37] is a methodology where solutions of a given problem is associated with those of Boolean equations. In this section, we develop positive (monotone) Boolean functions whose solutions are given by condition attribute subsets satisfying the preserving conditions (L) or (U). Moreover, prime implicants of the Boolean functions exactly correspond to L-reducts or U-reducts. The Boolean functions are useful for enumerating reducts.

The results of this section are well-known and appeared in many papers e.g. [1, 43, 45, 50], but in slightly different expressions from ours. A unified formulation of Boolean functions of different types of reducts is provided using the generalized decision function.

Here, we briefly introduce Boolean functions and Boolean formulas [9, 14]. Let  $q$  be a natural number. A Boolean function is a mapping  $f : \{0, 1\}^q \rightarrow \{0, 1\}$ , where  $w \in \{0, 1\}^q$  is called a Boolean vector whose  $i$ th component is  $w_i$ . Let  $x_1, x_2, \dots, x_q$  be Boolean variables. A Boolean formula in the Boolean variables  $x_1, x_2, \dots, x_q$  is a composition of 0, 1, the variables and operators of conjunction  $\wedge$ , disjunction  $\vee$ , complementation  $\bar{\cdot}$ , such as  $x_1 \wedge (x_2 \vee \bar{x}_3)$ ,  $(x_1 \wedge \bar{x}_2) \vee x_3$ , and so on (for complete definition, see e.g. [9]). The Boolean formula is a Boolean function of the variables  $x_1, x_2, \dots, x_q$ . Conversely, any Boolean function can be expressed by a Boolean formula. For two Boolean functions  $f$  and  $g$ ,  $g \leq f$  means that  $f$  and  $g$  satisfy

$g(w) \leq f(w)$  for all  $w \in \{0, 1\}^q$ , and  $g < f$  means that  $g \leq f$  and  $g \neq f$ . A Boolean function  $f$  is positive or monotone, if  $w \leq w'$  implies  $f(w) \leq f(w')$  for all  $w, w' \in \{0, 1\}^q$ .

Boolean variables  $x_1, x_2, \dots$  and the complements  $\bar{x}_1, \bar{x}_2, \dots$  are called literals. A clause (resp., term) is a disjunction (resp., conjunction) of at most one of  $x_i$  and  $\bar{x}_i$  for each variable. The empty disjunction (resp., conjunction) is denoted by  $\perp$  (resp.,  $\top$ ). A clause  $c$  (resp., term  $t$ ) is an implicate (resp., implicant) of a function  $f$ , if  $f \leq c$  (resp.  $t \leq f$ ). Moreover, it is prime if there is no implicate  $c' < c$  (resp., no implicant  $t' > t$ ) of  $f$ . A conjunction normal form (CNF) (resp., disjunction normal form (DNF)) of a function  $f$  is a Boolean formula of  $f$  which is expressed by a conjunction of implicates (resp. disjunction of implicants) of the function, and it is prime if all its members are prime. The complete CNF (resp. DNF) of  $f$  is the conjunction of all prime implicates (resp. disjunction of all prime implicants) of  $f$ . When  $f$  is positive, there is the unique CNF (resp. DNF) of  $f$  which is the complete CNF (resp. DNF) of  $f$ .

First, we associate conditions (L) (or (P)) and (U) (or (B)) with the conditions of the generalized decision function. As mentioned in the previous section, condition (U) is equivalent to (G).

**Lemma 1** ([23, 50]) *Let  $A$  be a subset of  $C$ . We have the following statements.*

- Condition (L) is equivalent to:

$$\partial_A(u) = \partial_C(u) \quad \text{for all } u \in U \quad \text{such that } |\partial_C(u)| = 1. \quad (\text{LG})$$

- Condition (U) is equivalent to (G), i.e.,

$$\partial_A(u) = \partial_C(u) \quad \text{for all } u \in U. \quad (\text{G})$$

The next lemma is the heart of the Boolean reasoning, which connects two notions: “preserving” and “discerning”.

**Lemma 2** *For  $u \in U$ , the following assertions are equivalent.*

- $\partial_A(u) = \partial_C(u)$ .
- $\forall u' \in U, (\partial_C(u') \neq \partial_C(u) \Rightarrow \exists a \in A, (u', u) \notin R_{\{a\}})$

Hence, to preserve the generalized decision of an object  $u$ , we should discern  $u$  from other objects  $u'$  having different generalized decisions from that of  $u$ .

Using Lemmas 1 and 2, we define two Boolean formulas, called discernibility functions. First, we define a discernibility matrix by  $M = (m_{ij})_{i,j=1,2,\dots,n}$ , where  $ij$ -entry  $m_{ij}$  is a set of condition attributes which discern objects  $u_i$  and  $u_j$ ,

$$m_{ij} = \{c \in C \mid c(u_i) \neq c(u_j)\}.$$

Then, we define discernibility functions.

**Definition 7** Discernibility functions  $F^U$  and  $F^L$  are defined as follows:

$$F^U(\tilde{c}_1, \dots, \tilde{c}_m) = \bigwedge_{i, j | \partial_C(u_j) \neq \partial_C(u_i)} \bigvee_{c \in m_{ij}} \tilde{c},$$

$$F^L(\tilde{c}_1, \dots, \tilde{c}_m) = \bigwedge_{i | |\partial_C(u_i)|=1} \bigwedge_{j | \partial_C(u_j) \neq \partial_C(u_i)} \bigvee_{c \in m_{ij}} \tilde{c},$$

where  $\tilde{c}_i$  is a Boolean variable corresponding to  $i$ th condition attribute  $c_i$ .

For  $A \subseteq C$ , we consider a Boolean vector  $\tilde{c}^A = (\tilde{c}_1^A, \dots, \tilde{c}_m^A)$ , where,

$$\tilde{c}_i^A = \begin{cases} 1 & \text{if } c_i \in A, \\ 0 & \text{if } c_i \notin A. \end{cases}$$

Let  $F^U(\tilde{c}^A) = 1$ . Then, for each pair  $u_i$  and  $u_j$  such that  $\partial_C(u_i) \neq \partial_C(u_j)$ , the intersection of  $A$  and  $m_{ij}$  should not be empty by the definition of  $F^U$ . By Lemma 2, in that case,  $\partial_A(u) = \partial_C(u)$  for each  $u$  holds. We have the similar consequence when  $F^L(\tilde{c}^A) = 1$ . Therefore, the following theorem holds. Let  $\phi_A = \bigwedge \{\tilde{c} | c \in A\}$ .

**Theorem 2** ([43, 45, 50]) *Let  $A$  be a subset of  $C$ . We have the following equivalences:*

- *$A$  satisfies (G), i.e., (U) if and only if  $F^U(\tilde{c}^A) = 1$ . Moreover,  $A$  is a U-reduct in RSM if and only if  $\phi_A$  is a prime implicant of  $F^U$ ,*
- *$A$  satisfies (LG), i.e., (L) if and only if  $F^L(\tilde{c}^A) = 1$ . Moreover,  $A$  is an L-reduct in RSM if and only if  $\phi_A$  is a prime implicant of  $F^L$ .*

Definition 7 shows CNFs of  $F^U$  and  $F^L$ . The prime CNFs of the functions can be easily obtained. Because the functions are positive, the prime implicants of the prime DNF of each function are all of the prime implicants of the function. Therefore, all reducts of each type appear in the prime DNF of the corresponding function. The problem which converts the prime CNF of a positive Boolean function to its prime DNF is called the dualization problem [14]. We show an example for enumerating reducts by solving the dualization problems of the discernibility functions.

*Example 6* Remember the decision table  $\mathbb{D} = (U, C \cup \{d\}, \{V_a\})$  in Table 7.1. In Table 7.2, we show again the decision table  $\mathbb{D}$  with the generalized decision function  $\partial_C$ .

The discernibility matrix is obtained as below. Sign \* attached to objects  $u_i$  means that the generalized decision of  $u_i$  is a singleton, or equivalently,  $u_i$  is in the positive region  $\text{POS}_C(d)$ .

**Table 7.2** The decision table in Table 7.1 with the generalized decision function

	Car	Pr	Ma	Sa	Ev	$\partial_C$
$u_1$	High	High	Low	Unacc	{unacc}	
$u_2$	Med	Med	Med	Unacc	{unacc, acc}	
$u_3$	Med	Med	Med	Acc	{unacc, acc}	
$u_4$	Med	High	Low	Acc	{acc}	
$u_5$	Med	Med	High	Acc	{acc, good}	
$u_6$	Med	Med	High	Good	{acc, good}	
$u_7$	Low	Med	Med	Good	{good}	

	$u_1$	$u_2$	$u_3$	$u_4$	$u_5$	$u_6$	$u_7$
$u_1^*$	$\emptyset$	$C$	$C$	{Pr}	$C$	$C$	$C$
$u_2$	$C$	$\emptyset$	$\emptyset$	{Ma, Sa}	{Sa}	{Sa}	{Pr}
$u_3$	$C$	$\emptyset$	$\emptyset$	{Ma, Sa}	{Sa}	{Sa}	{Pr}
$u_4^*$	{Pr}	{Ma, Sa}	{Ma, Sa}	$\emptyset$	{Ma, Sa}	{Ma, Sa}	$C$
$u_5$	$C$	{Sa}	{Sa}	{Ma, Sa}	$\emptyset$	$\emptyset$	{Pr, Sa}
$u_6$	$C$	{Sa}	{Sa}	{Ma, Sa}	$\emptyset$	$\emptyset$	{Pr, Sa}
$u_7^*$	$C$	{Pr}	{Pr}	$C$	{Pr, Sa}	{Pr, Sa}	$\emptyset$

The discernibility functions  $F^L$  and  $F^U$  for the decision table are calculated as:

$$F^L(\tilde{P}r, \tilde{M}a, \tilde{S}a) = \bigwedge_{i=1,4,7} \bigwedge_{j \neq i} \bigvee_{c \in m_{ij}} \tilde{c} = (\tilde{P}r) \wedge (\tilde{M}a \vee \tilde{S}a) = (\tilde{P}r \wedge \tilde{M}a) \vee (\tilde{P}r \wedge \tilde{S}a),$$

$$F^U(\tilde{P}r, \tilde{M}a, \tilde{S}a) = \bigwedge_{i,j | i \neq j, (i,j) \neq (2,3), (3,2), (5,6), (6,5)} \bigvee_{c \in m_{ij}} \tilde{c} = (\tilde{P}r) \wedge (\tilde{S}a) = (\tilde{P}r \wedge \tilde{S}a).$$

Therefore, there are two L-reducts {Pr, Ma} and {Pr, Sa}, and one U-reduct {Pr, Sa}.

In this case, we would select the U-reduct {Pr, Sa}, because we obtain the same size of reducts even if we select the other L-reduct.

### 7.3 Structure-Based Attribute Reduction in Variable Precision Rough Set Models

#### 7.3.1 Rough Membership Function

The reason why decision tables are inconsistent is not only lack of knowledge (condition attributes) related to the decision attribute but also noise in observation of attribute values. In the latter case, the classical RSM would not be very useful because it does not permit any errors in the classification of objects into the lower approximations.

To overcome such shortcoming of the classical RSM, the variable precision rough set model (VPRSM) was proposed [53, 54]. Let  $\mathbb{D} = (U, AT = C \cup \{d\}, \{V_a\}_{a \in AT})$  be a decision table. In definitions of lower and upper approximations in VPRSM, the following rough membership function of an object  $u$  with respect to an object set  $X \subseteq U$  and an attribute set  $A \subseteq AT$  plays an important role:

$$\mu_X^A(u) = \frac{|R_A(u) \cap X|}{|R_A(u)|}.$$

The value  $\mu_X^A(u)$  gives the degree to which the object  $u$  belongs to the set  $X$  under the attribute set  $A$ . It can be interpreted as the conditional probability of  $u \in X$  under  $u \in R_A(u)$ .

Because the rough membership function of an object is defined based not on the object but its equivalence class, we define a rough membership function of an equivalence class  $E \in U/R_A$  for  $X$ :

$$\mu_X(E) = \frac{|E \cap X|}{|E|}.$$

An important property of the function is that given two equivalence classes  $E, E' \in U/R_A$  the rough membership of the union  $E \cup E'$  falls between those of  $E$  and  $E'$ , namely,

$$\min\{\mu_X(E), \mu_X(E')\} \leq \mu_X(E \cup E') \leq \max\{\mu_X(E), \mu_X(E')\}. \quad (7.9)$$

### 7.3.2 Variable Precision Rough Set Models

Given precision parameters  $0 \leq \beta < \alpha \leq 1$ , lower and upper approximations of  $X$  with respect to  $A$  in VPRSM are defined as:

$$\begin{aligned} \text{LA}_A^\alpha(X) &= \{u \in U \mid \mu_X^A(u) \geq \alpha\}, \\ \text{UA}_A^\beta(X) &= \{u \in U \mid \mu_X^A(u) > \beta\}. \end{aligned}$$

The boundary of  $X$  is defined by  $\text{BN}_A^{\alpha,\beta}(X) = \text{UA}_A^\beta(X) \setminus \text{LA}_A^\alpha(X)$ . When  $\alpha = 1$  and  $\beta = 0$ , the approximations of  $X$  are the same as those of the classical RSM.  $\text{LA}_A^\alpha(X)$  is the set of objects whose degrees of membership to  $X$  are not less than  $\alpha$ . On the other hand,  $\text{UA}_A^\beta(X)$  is the set of objects whose degrees of membership to  $X$  are more than  $\beta$ . In this chapter, we restrict our discussion to the situation that  $\alpha = 1 - \beta$  and  $\beta \in [0, 0.5]$ . Under that situation, we have the dual property  $\text{LA}_A^\alpha(X) = U \setminus \text{UA}_A^\beta(U \setminus X)$ , because  $\mu_X^A(u) = 1 - \mu_{U \setminus X}^A(u)$ . We call  $\beta$  an admissible error rate. We denote  $\text{LA}_A^{1-\beta}(X)$  and  $\text{BN}_A^{1-\beta,\beta}(X)$  by  $\text{LA}_A^\beta(X)$  and  $\text{BN}_A^\beta(X)$ , respectively.

Differently from (7.1) of RSM, we do not always have  $\text{LA}_A^\beta(X) \subseteq X$  and  $\text{UA}_A^\beta(X) \supseteq X$ . However, we have

$$\text{LA}_A^\beta(X) \subseteq \text{UA}_A^\beta(X), \quad (7.10)$$

because  $1 - \beta > \beta$  when  $\beta < 0.5$ . Moreover, we also have

$$\text{LA}_A^\beta(X) \cap \text{LA}_A^\beta(X') = \emptyset, \quad (7.11)$$

for any disjoint subsets  $X, X' \subseteq U$ ,  $X \cap X' = \emptyset$ , because  $\beta < 0.5$ . Because the inclusion relation of (7.10), each of the lower and upper approximations, and the boundary is represented by the other two sets:

$$\begin{aligned} \text{UA}_A^\beta(X) &= \text{LA}_A^\beta(X) \cup \text{BN}_A^\beta(X), \\ \text{LA}_A^\beta(X) &= \text{UA}_A^\beta(X) \setminus \text{BN}_A^\beta(X). \end{aligned}$$

The monotonic property (7.4) does not hold either. It causes difficulties of defining and enumerating reducts in VPRSM.

We can define positive, boundary, and negative regions in the same manner of the classical RSM:

$$\begin{aligned} \text{POS}_A^\beta(X) &= \bigcup \{R_A(u) \mid \mu_X^A(u) \geq 1 - \beta\}, \\ \text{BND}_A^\beta(X) &= \bigcup \{R_A(u) \mid \mu_X^A(u) \in [\beta, 1 - \beta]\}, \\ \text{NEG}_A^\beta(X) &= \bigcup \{R_A(u) \mid \mu_X^A(u) > \beta\}. \end{aligned}$$

Clearly, we have,

$$\begin{aligned} \text{POS}_A^\beta(X) &= \text{LA}_A^\beta(X), \\ \text{BND}_A^\beta(X) &= \text{BN}_A^\beta(X), \\ \text{NEG}_A^\beta(X) &= U \setminus \text{UA}_A^\beta(X). \end{aligned}$$

In the rest of this section, we consider VPRSM under a decision table  $\mathbb{D} = (U, C \cup \{d\}, \{V_d\})$ . For each decision attribute value  $i \in V_d$ , the decision class  $X_i = \{u \in U \mid d(u) = i\}$ . The set of all decision classes are denoted by  $\mathcal{X} = \{X_1, X_2, \dots, X_p\}$ .

*Example 7* Consider a decision table  $\mathbb{D} = (U, C \cup \{d\}, \{V_d\})$  given in Table 7.3. The decision table composed of 40 objects with a condition attribute set  $C = \{c_1, c_2, c_3, c_4\}$  and a decision attribute  $d$ . Each condition attribute takes a value bad or good, i.e.,  $V_{c_i} = \{\text{bad, good}\}$  for  $i = 1, 2, 3, 4$ . The decision attribute takes one of three values:  $V_d = \{\text{bad, medium, good}\}$ . Then there are three decision classes  $X_b$ ,  $X_m$  and  $X_g$  whose objects take decision attribute value bad, medium and good,

**Table 7.3** An example of decision table

	$c_1$	$c_2$	$c_3$	$c_4$	$d : (b, m, g)$
$P_1$	Good	Good	Bad	Good	(0, 2, 9)
$P_2$	Good	Good	Good	Bad	(0, 19, 1)
$P_3$	Bad	Good	Bad	Good	(1, 1, 2)
$P_4$	Bad	Bad	Bad	Good	(0, 1, 1)
$P_5$	Good	Bad	Good	Good	(1, 1, 1)

respectively. In Table 7.3, objects are classified into 5 groups  $P_1, P_2, \dots, P_5$  by the condition attributes  $C$ . For example, group  $P_1$  is composed of objects having a condition attribute tuple  $(c_1, c_2, c_3, c_4) = (\text{good}, \text{good}, \text{bad}, \text{good}) \in V_C$ . The number of objects in each class in each group is shown in column  $d : (b, m, g)$  in Table 7.3. For example, (0,2,9) of group  $P_1$  means that no object is in class  $X_b$ , 2 objects are in class  $X_m$  and 9 objects are in class  $X_g$ .

The rough membership of an object  $u$  in each group  $P_i$  to each class  $X_k$  with respect to  $C$  is the number of objects in  $P_i$  and  $X_k$  divided by the number of objects in  $P_i$ . For example,  $\mu_{X_b}^C(P_1) = 0/(0 + 2 + 9) = 0$ ,  $\mu_{X_m}^C(P_1) = 2/(0 + 2 + 9) = 0.1818\dots$ ,  $\mu_{X_g}^C(P_1) = 9/(0 + 2 + 9) = 0.8181\dots$ . Given a condition attribute subset  $A = \{c_1, c_2\}$ , the objects in  $P_1$  and  $P_2$  are indiscernible to each other. Hence, the rough membership of an object  $u$  in  $P_1$  and  $P_2$  with respect to  $A$  becomes  $\mu_{X_b}^A(P_1) = \mu_{X_b}^A(P_2) = 0/(0 + 21 + 10) = 0$ ,  $\mu_{X_m}^A(P_1) = \mu_{X_m}^A(P_2) = 21/(0 + 21 + 10) = 0.6774\dots$ ,  $\mu_{X_g}^A(P_1) = \mu_{X_g}^A(P_2) = 10/(0 + 21 + 10) = 0.3225\dots$ .

Let  $\beta = 0.39$ . The lower approximations and the upper approximations with respect to  $C$  and  $\beta$  are obtained as follows:

$$\begin{aligned} \text{LA}_C^\beta(X_b) &= \emptyset, & \text{UA}_C^\beta(X_b) &= \emptyset, \\ \text{LA}_C^\beta(X_m) &= \{P_2\}, & \text{UA}_C^\beta(X_m) &= \{P_2, P_4\}, \\ \text{LA}_C^\beta(X_g) &= \{P_1\}, & \text{UA}_C^\beta(X_g) &= \{P_1, P_3, P_4\}, \end{aligned}$$

where we express approximations by means of groups, namely, all members of a group  $P$  are members of an approximation  $X$  if  $P \in X$ .

In VPRSM, the properties corresponding to (7.5) and (7.6) are not always satisfied. Consequently, L-reducts, U-reducts, and B-reducts become independent concepts in VPRSM, and there are no strong-weak relations among them.

Additionally, property (7.7) only partially holds:

$$\frac{1}{p} > \beta \Rightarrow U = \bigcup_{i \in V_d} \text{UA}_A^\beta(X_i). \quad (7.12)$$

The union of upper approximations of all decision classes does not always equal to  $U$  but when  $1/p > \beta$ . From this fact we define an unpredictable region of  $d$  with respect to  $\beta$  and  $A$ , denoted by  $\text{UNP}_A^\beta(d)$ , as follows:

$$\text{UNP}_A^\beta(d) = \bigcap_{i \in V_d} \text{NEG}_A^\beta(X_i),$$

equivalently,

$$\text{UNP}_A^\beta(d) = U - \bigcup_{i \in V_d} \text{UA}_A^\beta(X_i).$$

The unpredictable region is the set of all objects which cannot be classified to any decision class.

We can define the positive region of  $d$  with respect to  $\beta$  and  $A$  in the same manner of RSM,

$$\text{POS}_A^\beta(d) = \bigcup_{i \in V_d} \text{POS}_A^\beta(X_i).$$

The quality of classification of  $d$  can be also defined in the same manner,

$$\gamma_A^\beta(d) = \frac{|\text{POS}_A^\beta(d)|}{|U|}.$$

The generalized decision function in RSM can be extended in VPRSM. However, differently from RSM, we define two functions. They are called lower and upper generalized decision functions, denoted by  $\lambda$  and  $v$ , respectively. For each  $u \in U$ ,

$$\begin{aligned}\lambda_A^\beta(u) &= \{i \in V_d \mid \mu_{X_i}^A(u) \geq 1 - \beta\}, \\ v_A^\beta(u) &= \{i \in V_d \mid \mu_{X_i}^A(u) > \beta\}.\end{aligned}$$

The lower generalized decision of  $u$  is the set of the decision values to which the membership degree of  $u$  is more than or equal to  $1 - \beta$ . The upper generalized decision of  $u$  is the set of the decision values to which the membership degree of  $u$  is more than  $\beta$ . The upper generalized decision corresponds to the generalized decision in RSM. By the definitions, the lower and upper generalized decision functions are closely related to the lower and upper approximations,

$$\begin{aligned}i \in \lambda_A^\beta(u) &\Leftrightarrow u \in \text{LA}_A^\beta(X_i), \\ i \in v_A^\beta(u) &\Leftrightarrow u \in \text{UA}_A^\beta(X_i).\end{aligned}$$

So, they have the inclusion relation:

$$\lambda_A^\beta(u) \subseteq v_A^\beta(u). \quad (7.13)$$

Any two objects in the same equivalence class take the same values of generalized decision functions.

$$\text{For each } (u, u') \in R_A, \lambda_A^\beta(u) = \lambda_A^\beta(u') \text{ and } v_A^\beta(u) = v_A^\beta(u'). \quad (7.14)$$

The lower generalized decision function is a singleton or the empty set,

$$|\lambda_A^\beta(u)| \leq 1. \quad (7.15)$$

Unlike RSM, we may have the case that the upper generalized decision of  $u$  is a singleton, i.e.,  $v_A^\beta(u) = \{i\}$  but  $u$  does not belong to the lower approximation  $\text{LA}_A^\beta(X_i)$ . When the lower generalized decision is a singleton, the upper generalized decision is also a singleton, and they are the same,

$$|\lambda_A^\beta(u)| = 1 \Rightarrow v_A^\beta(u) = \lambda_A^\beta(u). \quad (7.16)$$

Property (7.12) can be expressed as:

$$\frac{1}{p} > \beta \Rightarrow v_A^\beta(u) \neq \emptyset. \quad (7.17)$$

Hence,  $v_A^\beta(u)$  may be empty unless  $\beta$  is less than  $\frac{1}{p}$ .

We define a function  $(v \setminus \lambda)_A^\beta(u)$  as:

$$(v \setminus \lambda)_A^\beta(u) = v_A^\beta(u) \setminus \lambda_A^\beta(u).$$

By properties (7.13), (7.15), and (7.16), we have

$$(v \setminus \lambda)_A^\beta(u) = \emptyset \Rightarrow v_A^\beta(u) = \emptyset \text{ or } \lambda_A^\beta(u) \neq \emptyset, \quad (7.18)$$

$$(v \setminus \lambda)_A^\beta(u) \neq \emptyset \Rightarrow (v \setminus \lambda)_A^\beta(u) = v_A^\beta(u). \quad (7.19)$$

By that property, the following equivalence holds:

$$i \in (v \setminus \lambda)_A^\beta(u) \Leftrightarrow u \in \text{BN}_A^\beta(X_i). \quad (7.20)$$

Therefore, we call  $(v \setminus \lambda)$  a boundary generalized decision function.

*Example 8* Remember the decision table  $\mathbb{D} = (U, C \cup \{d\}, \{V_a\})$  in Table 7.3. Let  $\beta = 0.39$ . The lower and upper generalized decision function with respect to  $C$  and  $\beta$  are,

$$\begin{aligned} \lambda_C^\beta(P_1) &= \{g\}, \lambda_C^\beta(P_2) = \{m\}, \lambda_C^\beta(P_3) = \emptyset, \lambda_C^\beta(P_4) = \emptyset, \lambda_C^\beta(P_5) = \emptyset, \\ v_C^\beta(P_1) &= \{g\}, v_C^\beta(P_2) = \{m\}, v_C^\beta(P_3) = \{g\}, v_C^\beta(P_4) = \{m,g\}, v_C^\beta(P_5) = \emptyset, \end{aligned}$$

where  $\lambda_C^\beta(P_i)$  and  $v_C^\beta(P_i)$  indicate the lower and upper generalized decisions of an object in the group  $P_i$ .

### 7.3.3 Structure-Based Reducts in Variable Precision Rough Set Models

Before we define structure-based reducts in VPRSM, we firstly introduce Q-reducts. They preserve the quality of classification with the parameter  $\beta$ .

**Definition 8** ([4, 5]) Let  $\beta \in [0, 0.5]$  be an admissible error rate. A Q-reduct with  $\beta$  in VPRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following conditions:

$$\gamma_A^\beta(d) = \gamma_C^\beta(d), \quad (\text{VPQ1})$$

$$B \text{ satisfies (VPQ1)} \quad \text{for all } B \supseteq A. \quad (\text{VPQ2})$$

*Remark 6* In VPRSM, approximations are no longer monotonic with respect to the set inclusion of condition attributes. Hence, condition (VPQ1) is not monotonic with respect to condition attributes, namely,  $A$  satisfies (VPQ1) but  $B \supset A$  does not. In that case, we modify the preserving condition of reducts by adding a condition like (VPQ2). We notice that Beynon [4, 5] originally proposed Q-reducts (the author called them  $\beta$ -reducts) using only (VPQ1).

We define 4 kinds of structure-based reducts in VPRSM [20], which are already discussed in the classical RSM.

**Definition 9** ([20, 33]) Let  $\beta \in [0, 0.5]$  be an admissible error rate.

- A P-reduct<sup>2</sup> with  $\beta$  in VPRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following conditions:

$$\text{POS}_A^\beta(d) = \text{POS}_C^\beta(d), \quad (\text{VPP1})$$

$$B \text{ satisfies (VPP1)} \quad \text{for all } B \supseteq A. \quad (\text{VPP2})$$

- An L-reduct with  $\beta$  in VPRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\text{LA}_A^\beta(X_i) = \text{LA}_C^\beta(X_i) \quad \text{for all } i \in V_d. \quad (\text{VPL})$$

- A B-reduct with  $\beta$  in VPRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following conditions:

$$\text{BN}_A^\beta(X_i) = \text{BN}_C^\beta(X_i) \quad \text{for all } i \in V_d, \quad (\text{VPB1})$$

$$B \text{ satisfies (VPB1)} \quad \text{for all } B \supseteq A. \quad (\text{VPB2})$$

---

<sup>2</sup> Strictly speaking, P-reducts do not appear in [20].

- A U-reduct with  $\beta$  in VPRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\text{UA}_A^\beta(X_i) = \text{UA}_C^\beta(X_i) \quad \text{for all } i \in V_d. \quad (\text{VPU})$$

Mi et al. [33] independently proposed L-reducts and U-reducts under the names of lower and upper distribution reducts.

Additionally, we can define a reduct preserving the unpredictable region.

**Definition 10** ([20]) Let  $\beta \in [0, 0.5]$  be an admissible error rate. A UN-reduct with  $\beta$  in VPRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following conditions:

$$\text{UNP}_A^\beta(d) = \text{UNP}_C^\beta(d), \quad (\text{VPUN1})$$

$$B \text{ satisfies } (\text{VPUN1}) \text{ for all } B \supseteq A. \quad (\text{VPUN2})$$

*Remark 7* We modify the definitions of B- and UN-reducts from our paper [20], because there are mistakes in Boolean functions for B- and UN-reducts. By adding the second condition, the preserving conditions of B- and UN-reducts become monotone with respect to the set-inclusion of condition attribute sets.

By definitions, (VPL) and (VPU) obviously imply (VPP1,2) and (VPUN1,2), respectively. Moreover, (VPP1,2) also implies (VPQ1,2). Hence, we have the following relations among different types of reducts.

**Theorem 3** ([20]) Let  $A$  be a subset of  $C$ . We have the following statements in VPRSM with a fixed parameter  $\beta \in [0, 0.5]$ ,

- (a)  $A$  is an L-reduct then  $A$  satisfies (VPP1,2),
- (b)  $A$  is a U-reduct then  $A$  satisfies (VPUN1,2),
- (c)  $A$  is a P-reduct then  $A$  satisfies (VPQ1,2).

Contrary to the classical RSM, (VPB1,2) is not equivalent to (VPU). In RSM, preserving boundaries implies preventing ambiguity expansion, namely upper approximations. However, in VPRSM, the ambiguity expansion can be prevented not only by preserving boundaries but by preserving them with the unpredictable region. Furthermore, we can define other compositions of different types of reducts.

Simply combining 5 types of structure-based reducts, we obtain  $2^5 - 1 = 31$  types of reducts (ignoring (a) and (b) of Theorem 3). To reduce the number, we first investigate relationships of preserving conditions of reducts.

**Theorem 4** ([20]) Let  $A$  be a subset of  $C$ . We have the following statements in VPRSM with a fixed parameter  $\beta \in [0, 0.5]$ ,

- The conjunction of (VPB1) and (VPP1) is equivalent to that of (VPB1) and (VPUN1),

- The conjunction of (VPL) and (VPB1) is equivalent to that of (VPL) and (VPU),
- The conjunction of (VPU) and (VPP1) is equivalent to that of (VPL) and (VPU).

We define 4 different types of reducts.

**Definition 11** ([20]) Let  $\beta \in [0, 0.5]$  be an admissible error rate.

- An LU-reduct with  $\beta$  in VPRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\text{LA}_A^\beta(X_i) = \text{LA}_C^\beta(X_i) \text{ and } \text{UA}_A^\beta(X_i) = \text{UA}_C^\beta(X_i) \quad \text{for all } i \in V_d. \quad (\text{VPLU})$$

- An LUN-reduct with  $\beta$  in VPRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following conditions:

$$\text{LA}_A^\beta(X_i) = \text{LA}_C^\beta(X_i) \quad \text{for all } i \in V_d, \text{ and } \text{UNP}_A^\beta(d) = \text{UNP}_C^\beta(d), \quad (\text{VPLUN1})$$

$$B \text{ satisfies (VPLUN1)} \quad \text{for all } B \supseteq A. \quad (\text{VPLUN2})$$

- A BUN-reduct with  $\beta$  in VPRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following conditions:

$$\text{BN}_A^\beta(X_i) = \text{BN}_C^\beta(X_i) \quad \text{for all } i \in V_d, \text{ and } \text{UNP}_A^\beta(d) = \text{UNP}_C^\beta(d), \quad (\text{VPBUN1})$$

$$B \text{ satisfies (VPBUN1)} \quad \text{for all } B \supseteq A. \quad (\text{VPBUN2})$$

- A PUN-reduct with  $\beta$  in VPRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following conditions:

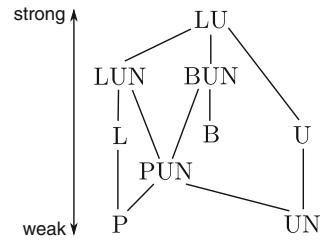
$$\text{POS}_A^\beta(d) = \text{POS}_C^\beta(d) \quad \text{and } \text{UNP}_A^\beta(d) = \text{UNP}_C^\beta(d), \quad (\text{VPPUN1})$$

$$B \text{ satisfies (VPPUN1)} \quad \text{for all } B \supseteq A. \quad (\text{VPPUN2})$$

In Fig. 7.2, we show the relationships among 9 types of reducts. Names of reducts are abbreviated to their first characters. Reducts located in the upper part of Fig. 7.2 preserve regions much more. Therefore, such reducts are larger in the sense of set inclusion than the other reducts located in the lower part. A line segment connecting two types of reducts implies that, for each reduct of the upper type say  $A$  satisfies the preserving condition of a reduct of the lower one. From Fig. 7.2, we know that LU-reducts preserve regions most. On the other hand, UN-reducts and P-reducts do not preserve many regions.

The next proposition says that composite reducts such as LUN- or BUN-reducts can be constructed from their base reducts such as L- and UN-reducts or B- and UN-reducts. The proposition is useful to enumerate the composite reducts.

**Fig. 7.2** Strong-weak hierarchy of 9 types of structure-based reducts in VPRSM



**Proposition 1** Consider two types of reducts,  $\heartsuit$ -reducts and  $\spadesuit$ -reducts, and the composition of them:  $\heartsuit\spadesuit$ -reducts. Let  $\mathcal{H}$  and  $\mathcal{S}$  be the set of all  $\heartsuit$ -reducts and the set of all  $\spadesuit$ -reducts, respectively. Then the set of all  $\heartsuit\spadesuit$ -reducts is the set of all minimal elements of  $\{A \cup B \mid A \in \mathcal{H} \text{ and } B \in \mathcal{S}\}$ .

### 7.3.4 Boolean Functions Representing Reducts

As shown above, L- and U-reducts in the classical RSM are characterized by prime implicants of certain Boolean functions. In this section, we discuss Boolean functions of 9 types of reducts in VPRSM. To do this, we focus on Boolean functions of reducts pertaining to the lower approximations, the upper approximations, the boundaries, the positive region, and the unpredictable region, since the others can be obtained by taking conjunctions of those Boolean functions or using Proposition 1.

First, we represent the preserving conditions by the generalized decision functions.

**Lemma 3** Let  $\beta \in [0, 0.5)$  be an admissible error rate, and  $A$  be a subset of  $C$ . We have the following statements:

- Condition (VPL) with  $\beta$  is equivalent to:

$$\lambda_A^\beta(u) = \lambda_C^\beta(u) \quad \text{for all } u \in U. \quad (\text{VPLG})$$

- Condition (VPU) with  $\beta$  is equivalent to:

$$v_A^\beta(u) = v_C^\beta(u) \quad \text{for all } u \in U. \quad (\text{VPUG})$$

- Condition (VPB1) with  $\beta$  is equivalent to:

$$(v \setminus \lambda)_A^\beta(u) = (v \setminus \lambda)_C^\beta(u) \quad \text{for all } u \in U. \quad (\text{VPBG1})$$

- Condition (VPP1) with  $\beta$  is equivalent to:

$$\lambda_A^\beta(u) = \emptyset \Leftrightarrow \lambda_C^\beta(u) = \emptyset \quad \text{for all } u \in U. \quad (\text{VPPG1})$$

- Condition (VPUN1) with  $\beta$  is equivalent to:

$$v_A^\beta(u) = \emptyset \Leftrightarrow v_C^\beta(u) = \emptyset \quad \text{for all } u \in U. \quad (\text{VPUNG1})$$

The next lemma is the counterpart of Lemma 2 of RSM. However, only the sufficient condition of the lemma holds in VPRSM.

**Lemma 4** Let  $u \in U$  be an object,  $\beta \in [0, 0.5)$  be an admissible error rate, and  $A$  be a subset of  $C$ .

- The following assertion is a sufficient condition of  $v_A^\beta(u) = v_C^\beta(u)$ :

$$\forall u' \in U, (v_C^\beta(u') \neq v_C^\beta(u)) \Rightarrow \exists a \in A, (u', u) \notin R_{\{a\}}.$$

- The following assertion is a sufficient condition of  $\lambda_A^\beta(u) = \lambda_C^\beta(u)$ :

$$\forall u' \in U, (\lambda_C^\beta(u') \neq \lambda_C^\beta(u)) \Rightarrow \exists a \in A, (u', u) \notin R_{\{a\}}.$$

This lemma holds due to property (7.9). Then, we have the following corollary.

**Corollary 1** We have the following equivalences:

$$\begin{aligned} & \forall u \in U, v_A^\beta(u) = v_C^\beta(u) \\ & \Leftrightarrow \forall u, u' \in U, (v_C^\beta(u') \neq v_C^\beta(u)) \Rightarrow \exists a \in A, (u', u) \notin R_{\{a\}}, \\ & \forall u \in U, \lambda_A^\beta(u) = \lambda_C^\beta(u) \\ & \Leftrightarrow \forall u, u' \in U, (\lambda_C^\beta(u') \neq \lambda_C^\beta(u)) \Rightarrow \exists a \in A, (u', u) \notin R_{\{a\}}. \end{aligned}$$

It says that all L-reducts and all U-reducts can be enumerated by discernibility functions. The similar result is shown in [33]. However, we do not have the same result for  $(\nu \setminus \lambda)$  and conditions (VPPG1) and (VPUNG1).

We introduce a discernibility matrix  $M = (m_{i,j})_{ij=1,2,\dots,n}$ , where  $ij$ -entry  $m_{ij}$  is defined by:

$$m_{ij} = \{c \in C \mid a(u_i) \neq a(u_j)\}.$$

It is the same as that of RSM. Then, we define discernibility functions corresponding to L-reducts and U-reducts, which are denoted by  $F_\beta^U$  and  $F_\beta^L$ , respectively.

**Definition 12** Let  $\beta \in [0, 0.5)$  be an admissible error rate. Discernibility functions  $F_\beta^U$  and  $F_\beta^L$  are defined as follows:

$$F_\beta^U(\tilde{c}_1, \tilde{c}_2, \dots, \tilde{c}_m) = \bigwedge_{i,j \mid v_C^\beta(u_i) \neq v_C^\beta(u_j)} \bigvee_{c \in m_{ij}} \tilde{c},$$

$$F_\beta^L(\tilde{c}_1, \tilde{c}_2, \dots, \tilde{c}_m) = \bigwedge_{i,j \mid \lambda_C^\beta(u_i) \neq \lambda_C^\beta(u_j)} \bigvee_{c \in m_{ij}} \tilde{c},$$

where,  $\tilde{c}_i$  is a Boolean variable pertaining to a condition attribute  $c_i \in C$ .

Function  $F_\beta^U$  is true if and only if at least one variable  $\tilde{c}$  in  $m_{ij}$  of  $v_C^\beta(u_i) \neq v_C^\beta(u_j)$  is true. While function  $F_\beta^L$  is true if and only if at least one variable  $\tilde{c}$  in  $m_{ij}$  of  $\lambda_C^\beta(u_i) \neq \lambda_C^\beta(u_j)$  is true.

Remember that we associate  $A \subseteq C$  with a Boolean vector  $\tilde{c}^A = (\tilde{c}_1^A, \tilde{c}_2^A, \dots, \tilde{c}_m^A)$  as follows:

$$\tilde{c}_k^A = \begin{cases} 1 & c_k \in A, \\ 0 & \text{otherwise.} \end{cases}$$

Then, we can prove the next theorem from Corollary 1. Remember that  $\phi_A$  is the term  $\bigwedge \{\tilde{a} | a \in A\}$ .

**Theorem 5** ([20, 33]) *Let  $A$  be the subset of  $C$ , and  $\beta \in [0, 0.5]$  be an admissible error rate. We have the following equivalences:*

- *$A$  satisfies (VPUT) as well as (VPU) with  $\beta$  if and only if  $F_\beta^U(\tilde{c}^A) = 1$ . Moreover,  $A$  is a U-reduct with  $\beta$  if and only if  $\phi_A$  is a prime implicant of  $F_\beta^U$ ,*
- *$A$  satisfies (VPLG) as well as (VPL) with  $\beta$  if and only if  $F_\beta^L(\tilde{c}^A) = 1$ . Moreover,  $A$  is an L-reduct with  $\beta$  if and only if  $\phi_A$  is a prime implicant of  $F_\beta^L$ .*

For the preservation of the boundaries, the positive region, and the unpredictable region, we cannot use discernibility function approach. Because we cannot obtain a preserving subset  $A \subseteq C$  by determining which pairs of objects should be discerned. For example, consider a decision table below.

	$c_1$	$c_2$	$c_3$	$X_1$	$X_2$	$X_3$
$P_1$	0	0	0	4	0	0
$P_2$	0	0	1	0	2	0
$P_3$	0	1	0	0	0	1
$P_4$	1	1	0	1	1	1

There are 3 condition attributes  $C = \{c_1, c_2, c_3\}$  with the value set  $V = \{0, 1\}$ , and 3 decision classes  $X_1, X_2, X_3$ .  $P_1, P_2, P_3, P_4$  are sets of objects, where members of each set have the same condition attribute values. The distribution of the decision classes on each set  $P_i$  is shown in the table, for instance the distribution on  $P_1$  forms  $|X_1 \cap P_1| = 4$ ,  $|X_2 \cap P_1| = 0$ , and  $|X_3 \cap P_1| = 0$ . Consider P-reducts with  $\beta = 0.4$ . The positive region of the table is  $\text{POS}_C^\beta(d) = P_1 \cup P_2 \cup P_3$ . When we can make  $P_1$  and  $P_2$  be indiscernible and combine  $P_1 \cup P_2$ , the positive region is still preserved. Because the distribution on  $P_1 \cup P_2$  is  $(X_1, X_2, X_3) = (4, 2, 0)$ , and  $\mu_{X_1}(P_1 \cup P_2) =$

$2/3 \geq 0.6$ . Similarly, we can make the pair of  $P_1$  and  $P_3$  and the pair of  $P_2$  and  $P_3$  be indiscernible. However, when we make all of  $P_1$ ,  $P_2$ , and  $P_3$  indiscernible, and select  $\{c_1\}$  as a reduct,  $P_1 \cup P_2 \cup P_3$  falls outside of  $\text{POS}_{\{c_1\}}^{\beta}(d)$ , because the distribution is  $(X_1, X_2, X_3) = (4, 2, 1)$ , and  $\mu_{X_1}(P_1 \cup P_2 \cup P_3) = 4/7 \leq 0.6$ .

To overcome that difficulty, for each type of reducts, we consider two approximate discernibility functions  $\hat{F}_\beta$  and  $\check{F}_\beta$ :  $\hat{F}_\beta$  characterizes a sufficient condition of the preservation and  $\check{F}_\beta$  characterizes a necessary condition.

First, we discuss discernibility functions characterizing sufficient conditions. By Theorems 3 and 4, we know that  $F_\beta^L$ ,  $F_\beta^U$ , and  $F_\beta^{LU} = F_\beta^L \wedge F_\beta^U$  are discernibility functions of sufficient conditions for B-reducts, P-reducts, and UN-reducts with  $\beta$ .

**Definition 13** Let  $\beta \in [0, 0.5)$  be an admissible error rate. Discernibility functions  $\hat{F}_\beta^B$ ,  $\hat{F}_\beta^P$ , and  $\hat{F}_\beta^{UN}$  are defined as follows:

$$\hat{F}_\beta^B = F_\beta^L \wedge F_\beta^U, \quad \hat{F}_\beta^P = F_\beta^L, \quad \hat{F}_\beta^{UN} = F_\beta^U.$$

Clearly, we have the following proposition.

**Proposition 2** ([20]) Let  $A$  be a subset of  $C$ , and  $\beta \in [0, 0.5)$  be an admissible error rate. We have the following implications:

- If  $\hat{F}_\beta^B(\tilde{c}^A) = 1$  then  $A$  satisfies (VPB1) and (VPB2) with  $\beta$ ,
- If  $\hat{F}_\beta^P(\tilde{c}^A) = 1$  then  $A$  satisfies (VPP1) and (VPP2) with  $\beta$ ,
- If  $\hat{F}_\beta^{UN}(\tilde{c}^A) = 1$  then  $A$  satisfies (VPUN1) and (VPUN2) with  $\beta$ .

Next, let us discuss a discernibility function characterizing a necessary condition. Consider necessary discernibility functions for P-reducts. In the sufficient discernibility function  $\hat{F}_\beta^P = F_\beta^L$ , pairs of objects included in the positive region are discerned when they have different values of  $\lambda_C^\beta$ . However, such pairs are not necessarily discerned because there may be a P-reduct such that some of pairs become indiscernible. On the other hand, for each pair  $u_i$  and  $u_j$ , if they are excluded from the positive region of the common condition attributes, i.e.,  $C \setminus m_{ij}$ , they should be discerned because no subset  $A \subseteq C \setminus m_{ij}$  satisfies (VPP1) and (VPP2). From this consideration, discernibility functions characterizing necessary conditions for preservation of the boundaries, the positive region, and the unpredictable region are obtained as follows.

**Definition 14** Let  $\beta \in [0, 0.5)$  be an admissible error rate. Moreover, let  $\tilde{c}_i$  be a Boolean variable pertaining to a condition attribute  $c_i \in C$ . Discernibility functions  $\check{F}_\beta^B$ ,  $\check{F}_\beta^P$ , and  $\check{F}_\beta^{UN}$  are defined as follows:

$$\check{F}_\beta^B(\tilde{c}_1, \tilde{c}_2, \dots, \tilde{c}_m) = \bigwedge_{(i,j) \in \Delta_\beta^B} \bigvee_{c \in m_{ij}} \tilde{c},$$

$$\check{F}_\beta^P(\tilde{c}_1, \tilde{c}_2, \dots, \tilde{c}_m) = \bigwedge_{(i,j) \in \Delta_\beta^P} \bigvee_{c \in m_{ij}} \tilde{c},$$

$$\check{F}_\beta^{\text{UN}}(\tilde{c}_1, \tilde{c}_2, \dots, \tilde{c}_m) = \bigwedge_{(i,j) \in \Delta_\beta^{\text{UN}}} \bigvee_{c \in m_{ij}} \tilde{c},$$

where,

$$(i, j) \in \Delta_\beta^{\text{B}} \Leftrightarrow \begin{cases} (\nu \setminus \lambda)_C^\beta(u_i) \neq (\nu \setminus \lambda)_C^\beta(u_j), \text{ or} \\ (\nu \setminus \lambda)_C^\beta(u_i) = (\nu \setminus \lambda)_C^\beta(u_j) = \emptyset \\ \text{and } (\nu \setminus \lambda)_{C \setminus m_{ij}}^\beta(u_i) = (\nu \setminus \lambda)_{C \setminus m_{ij}}^\beta(u_j) \neq \emptyset, \end{cases}$$

$$(i, j) \in \Delta_\beta^{\text{P}} \Leftrightarrow \begin{cases} \lambda_C^\beta(u_i) \neq \emptyset \text{ and } \lambda_C^\beta(u_j) = \emptyset, \text{ or} \\ \lambda_C^\beta(u_i) = \emptyset \text{ and } \lambda_C^\beta(u_j) \neq \emptyset, \text{ or} \\ \lambda_C^\beta(u_i) \neq \emptyset, \lambda_C^\beta(u_j) \neq \emptyset, \text{ and } \lambda_{C \setminus m_{ij}}^\beta(u_i) = \lambda_{C \setminus m_{ij}}^\beta(u_j) = \emptyset, \end{cases}$$

$$(i, j) \in \Delta_\beta^{\text{UN}} \Leftrightarrow \begin{cases} \nu_C^\beta(u_i) \neq \emptyset \text{ and } \nu_C^\beta(u_j) = \emptyset, \text{ or} \\ \nu_C^\beta(u_i) = \emptyset \text{ and } \nu_C^\beta(u_j) \neq \emptyset, \text{ or} \\ \nu_C^\beta(u_i) \neq \emptyset, \nu_C^\beta(u_j) \neq \emptyset, \text{ and } \nu_{C \setminus m_{ij}}^\beta(u_i) = \nu_{C \setminus m_{ij}}^\beta(u_j) = \emptyset. \end{cases}$$

**Proposition 3** Let  $A$  be a subset of  $C$ , and  $\beta \in [0, 0.5)$  be an admissible error rate. We have the following implications:

- If  $\check{F}_\beta^{\text{B}}(\tilde{c}^A) = 0$  then  $A$  does not satisfy (VPB1) or (VPB2) with  $\beta$ ,
- If  $\check{F}_\beta^{\text{P}}(\tilde{c}^A) = 0$  then  $A$  does not satisfy (VPP1) or (VPP2) with  $\beta$ ,
- If  $\check{F}_\beta^{\text{UN}}(\tilde{c}^A) = 0$  then  $A$  does not satisfy (VPUN1) or (VPUN2) with  $\beta$ .

From Proposition 3, we know that any prime implicant of each of  $\check{F}_\beta^{\text{B}}$ ,  $\check{F}_\beta^{\text{P}}$ , and  $\check{F}_\beta^{\text{UN}}$  can be a subset of some reduct of the corresponding type.

Combining Propositions 2 and 3, we have the following theorem.

**Theorem 6** Let  $A$  be a subset of  $C$ , and  $\beta \in [0, 0.5)$  be an admissible error rate. Let  $\hat{\mathcal{P}}_\beta^{\text{B}}$ ,  $\hat{\mathcal{P}}_\beta^{\text{P}}$  and  $\hat{\mathcal{P}}_\beta^{\text{UN}}$  be the sets of condition attribute subsets corresponding to the prime implicants of  $\hat{F}_\beta^{\text{B}}$ ,  $\hat{F}_\beta^{\text{P}}$ , and  $\hat{F}_\beta^{\text{UN}}$ , respectively. Moreover, let  $\check{\mathcal{P}}_\beta^{\text{B}}$ ,  $\check{\mathcal{P}}_\beta^{\text{P}}$  and  $\check{\mathcal{P}}_\beta^{\text{UN}}$  be the sets of condition attribute subsets corresponding to the prime implicants of  $\check{F}_\beta^{\text{B}}$ ,  $\check{F}_\beta^{\text{P}}$ , and  $\check{F}_\beta^{\text{UN}}$ , respectively. Then, we have the following implications:

- If  $A$  is a  $B$ -reduct with  $\beta$  then  $A \in \{B \subseteq C \mid B \supseteq B' \text{ for some } B' \in \check{\mathcal{P}}_\beta^{\text{B}} \text{ and } B \not\supseteq B'' \text{ for any } B'' \in \hat{\mathcal{P}}_\beta^{\text{B}}\}$ ,
- If  $A$  is a  $P$ -reduct with  $\beta$  then  $A \in \{B \subseteq C \mid B \supseteq B' \text{ for some } B' \in \check{\mathcal{P}}_\beta^{\text{P}} \text{ and } B \not\supseteq B'' \text{ for any } B'' \in \hat{\mathcal{P}}_\beta^{\text{P}}\}$ ,
- If  $A$  is a  $UN$ -reduct with  $\beta$  then  $A \in \{B \subseteq C \mid B \supseteq B' \text{ for some } B' \in \check{\mathcal{P}}_\beta^{\text{UN}} \text{ and } B \not\supseteq B'' \text{ for any } B'' \in \hat{\mathcal{P}}_\beta^{\text{UN}}\}$ .

**Table 7.4** Discernibility functions related to 9 kinds of reducts

Reduct	Discernibility function(s)	Exact/approximate
L	$F_\beta^L$	Exact
U	$F_\beta^U$	Exact
B	$(\check{F}_\beta^B, F_\beta^L \wedge F_\beta^U)$	Approximate
P	$(\check{F}_\beta^P, F_\beta^L)$	Approximate
UN	$(\check{F}_\beta^{UN}, F_\beta^U)$	Approximate
LU	$F_\beta^L \wedge F_\beta^U$	Exact
PUN	$(\check{F}_\beta^P \wedge \check{F}_\beta^{UN}, F_\beta^L \wedge F_\beta^U)$	Approximate
LUN	$(F_\beta^L \wedge \check{F}_\beta^{UN}, F_\beta^L \wedge F_\beta^U)$	Approximate
BUN	$(\check{F}_\beta^B \wedge \check{F}_\beta^P, F_\beta^L \wedge F_\beta^U)$	Approximate

The obtained discernibility functions are shown in Table 7.4. In the case of approximate discernibility functions, the first function in the parenthesis characterizes the necessary condition of the preservation and the second function characterizes the sufficient condition. The discernibility functions related to LU-reducts, LUN-reducts and BUN-reducts can be obtained by taking the conjunctions of discernibility functions related to L-reducts, U-reducts, B-reducts and UN-reducts. Note that  $\hat{F}_\beta^B \wedge \hat{F}_\beta^{UN} = (F_\beta^L \wedge F_\beta^U) \wedge F_\beta^U = F_\beta^L \wedge F_\beta^U$ . This is why we have  $F_\beta^L \wedge F_\beta^U$  as the discernibility function characterizing a sufficient condition for the preservation of BUN-reducts.

*Example 9* Remember the decision table  $\mathbb{D} = (U, C \cup \{d\}, \{V_a\})$  in Table 7.3. Let an admissible error rate be  $\beta = 0.39$ . In Table 7.5, we show the decision table with three generalized decision functions  $\lambda_C^{0.39}, v_C^{0.39}, (v \setminus \lambda)_C^{0.39}$  with respect to  $C$  and  $\beta = 0.39$ .

Now let us enumerate reducts as prime implicants of discernibility functions. First let us discuss L-, U- and LU-reducts with  $\beta = 0.39$ . The discernibility matrix of the decision table is shown as below.

	$P_1$	$P_2$	$P_3$	$P_4$	$P_5$
$P_1$	$\emptyset$	$\{c_3, c_4\}$	$\{c_1\}$	$\{c_1, c_2\}$	$\{c_2, c_3\}$
$P_2$	$\{c_3, c_4\}$	$\emptyset$	$\{c_1, c_3, c_4\}$	$C$	$\{c_2, c_4\}$
$P_3$	$\{c_1\}$	$\{c_1, c_3, c_4\}$	$\emptyset$	$\{c_2\}$	$\{c_1, c_2, c_3\}$
$P_4$	$\{c_1, c_2\}$	$C$	$\{c_2\}$	$\emptyset$	$\{c_1, c_3\}$
$P_5$	$\{c_2, c_3\}$	$\{c_2, c_4\}$	$\{c_1, c_2, c_3\}$	$\{c_1, c_3\}$	$\emptyset$

**Table 7.5** The decision table in Table 7.3 with the generalized decision functions

	$c_1$	$c_2$	$c_3$	$c_4$	$d:(b,m,g)$	$\lambda_C^\beta$	$v_C^{0.39}$	$(v \setminus \lambda)_C^{0.39}$
$P_1$	Good	Good	Bad	Good	(0,2,9)	{g}	{g}	$\emptyset$
$P_2$	Good	Good	Good	Bad	(0,19,1)	{m}	{m}	$\emptyset$
$P_3$	Bad	Good	Bad	Good	(1,1,2)	$\emptyset$	{g}	{g}
$P_4$	Bad	Bad	Bad	Good	(0,1,1)	$\emptyset$	{m,g}	{m,g}
$P_5$	Good	Bad	Good	Good	(1,1,1)	$\emptyset$	$\emptyset$	$\emptyset$

From the table, we obtain  $F_{0.39}^L$  and  $F_{0.39}^U$  as follows:

$$\begin{aligned} F_{0.39}^L(\tilde{c}_1, \tilde{c}_2, \tilde{c}_3, \tilde{c}_4) &= \bigwedge_{i=1,2, j=3,4,5} \bigvee_{c \in m_{ij}} \tilde{c} \wedge \bigvee_{c \in m_{12}} \tilde{c} \\ &= (\tilde{c}_1) \wedge (\tilde{c}_2 \vee \tilde{c}_3) \wedge (\tilde{c}_2 \vee \tilde{c}_4) \wedge (\tilde{c}_3 \vee \tilde{c}_4) \\ &= (\tilde{c}_1 \wedge \tilde{c}_2 \wedge \tilde{c}_3) \vee (\tilde{c}_1 \wedge \tilde{c}_2 \wedge \tilde{c}_4) \vee (\tilde{c}_1 \wedge \tilde{c}_3 \wedge \tilde{c}_4), \\ F_{0.39}^U(\tilde{c}_1, \tilde{c}_2, \tilde{c}_3, \tilde{c}_4) &= \bigwedge_{(i,j) \in \{(k,l) | k \neq l\} \setminus \{(1,3), (3,1)\}} \bigvee_{c \in m_{ij}} \tilde{c} \\ &= (\tilde{c}_2) \wedge (\tilde{c}_1 \vee \tilde{c}_3) \wedge (\tilde{c}_3 \vee \tilde{c}_4) = (\tilde{c}_2 \wedge \tilde{c}_3) \vee (\tilde{c}_1 \wedge \tilde{c}_2 \wedge \tilde{c}_4). \end{aligned}$$

We obtain  $F_{0.39}^{LU}$  as:

$$\begin{aligned} F_{0.39}^{LU}(\tilde{c}_1, \tilde{c}_2, \tilde{c}_3, \tilde{c}_4) &= F^L(\tilde{c}_1, \tilde{c}_2, \tilde{c}_3, \tilde{c}_4) \wedge F^U(\tilde{c}_1, \tilde{c}_2, \tilde{c}_3, \tilde{c}_4) \\ &= (\tilde{c}_1) \wedge (\tilde{c}_2) \wedge (\tilde{c}_3 \vee \tilde{c}_4) = (\tilde{c}_1 \wedge \tilde{c}_2 \wedge \tilde{c}_3) \vee (\tilde{c}_1 \wedge \tilde{c}_2 \wedge \tilde{c}_4). \end{aligned}$$

Therefore, L-reducts are obtained as  $\{c_1, c_2, c_3\}$ ,  $\{c_1, c_2, c_4\}$  and  $\{c_1, c_3, c_4\}$ . U-reducts are obtained as  $\{c_2, c_3\}$  and  $\{c_1, c_2, c_4\}$ . LU-reducts are obtained as  $\{c_1, c_2, c_3\}$  and  $\{c_1, c_2, c_4\}$ . Note that  $\{c_2, c_3\}$  is not an L-reduct but a U-reduct. This is very different from the relation between L- and U-reducts in the classical RSM, i.e., in the classical RSM, a U-reduct includes an L-reduct but an L-reduct never includes a U-reduct.

Now let us discuss B-, P-, UN-reducts with  $\beta = 0.39$ . We can obtain only approximations of those reducts. To this end, let us get discernibility functions  $\check{F}_{0.39}^B$ ,  $\check{F}_{0.39}^P$ , and  $\check{F}_{0.39}^{UN}$ . For B-reducts, considering the second condition of  $\Delta_{0.39}^B$ , check each pair  $P_i$  and  $P_j$  such that  $(v \setminus \lambda)_C^{0.39}(P_i) = \emptyset$  and  $(v \setminus \lambda)_C^{0.39}(P_j) = \emptyset$ .

$$\begin{aligned} (v \setminus \lambda)_{C \setminus m_{12}}^{0.39}(P_1) &= (v \setminus \lambda)_{C \setminus m_{12}}^{0.39}(P_2) = \emptyset, \quad (v \setminus \lambda)_{C \setminus m_{15}}^{0.39}(P_1) = (v \setminus \lambda)_{C \setminus m_{15}}^{0.39}(P_5) = \emptyset, \\ (v \setminus \lambda)_{C \setminus m_{25}}^{0.39}(P_2) &= (v \setminus \lambda)_{C \setminus m_{25}}^{0.39}(P_5) = \emptyset. \end{aligned}$$

For P-reducts, check each pair such that  $\lambda_C^{0.39}(P_i) \neq \emptyset$  and  $\lambda_C^{0.39}(P_j) \neq \emptyset$ .

$$\lambda_{C \setminus m_{12}}^{0.39}(P_1) = \lambda_{C \setminus m_{12}}^{0.39}(P_2) = \{\text{medium}\}.$$

Finally, for UN-reducts, check each pair such that  $v_C^{0.39}(P_i) \neq \emptyset$  and  $v_C^{0.39}(P_j) \neq \emptyset$ .

$$\begin{aligned} v_{C \setminus m_{12}}^{0.39}(P_1) &= v_{C \setminus m_{12}}^{0.39}(P_2) = \{\text{medium}\}, \quad v_{C \setminus m_{13}}^{0.39}(P_1) = v_{C \setminus m_{13}}^{0.39}(P_3) = \{\text{good}\}, \\ v_{C \setminus m_{14}}^{0.39}(P_1) &= v_{C \setminus m_{14}}^{0.39}(P_4) = \{\text{good}\}, \quad v_{C \setminus m_{23}}^{0.39}(P_2) = v_{C \setminus m_{23}}^{0.39}(P_3) = \{\text{medium}\}, \\ v_{C \setminus m_{24}}^{0.39}(P_2) &= v_{C \setminus m_{24}}^{0.39}(P_4) = \{\text{medium}\}, \quad v_{C \setminus m_{34}}^{0.39}(P_3) = v_{C \setminus m_{34}}^{0.39}(P_4) = \{\text{good}\}. \end{aligned}$$

Therefore, discernibility functions  $\check{F}_{0.39}^B$ ,  $\check{F}_{0.39}^P$ , and  $\check{F}_{0.39}^{UN}$  are obtained as:

$$\begin{aligned}\check{F}_{0.39}^B(\tilde{c}_1, \tilde{c}_2, \tilde{c}_3, \tilde{c}_4) &= \bigwedge_{i=1,2,5} \bigvee_{j=3,4} \tilde{c} \wedge \bigvee_{c \in m_{ij}} \tilde{c} = \tilde{c}_1 \wedge \tilde{c}_2, \\ \check{F}_{0.39}^P(\tilde{c}_1, \tilde{c}_2, \tilde{c}_3, \tilde{c}_4) &= \bigwedge_{i=1,2} \bigvee_{j=3,4,5} \tilde{c} = (\tilde{c}_1) \wedge (\tilde{c}_2 \vee \tilde{c}_3) \wedge (\tilde{c}_2 \vee \tilde{c}_4) \\ &= (\tilde{c}_1 \wedge \tilde{c}_2) \vee (\tilde{c}_1 \wedge \tilde{c}_3 \wedge \tilde{c}_4), \\ \check{F}_{0.39}^{UN}(\tilde{c}_1, \tilde{c}_2, \tilde{c}_3, \tilde{c}_4) &= \bigwedge_{i=1,2,3,4} \bigvee_{j=5} \tilde{c} = (\tilde{c}_1 \vee \tilde{c}_3) \wedge (\tilde{c}_2 \vee \tilde{c}_3) \wedge (\tilde{c}_2 \vee \tilde{c}_4) \\ &= (\tilde{c}_1 \wedge \tilde{c}_2) \vee (\tilde{c}_2 \wedge \tilde{c}_3) \vee (\tilde{c}_3 \wedge \tilde{c}_4).\end{aligned}$$

Because  $\hat{F}_{0.39}^B = F_{0.39}^L \wedge F_{0.39}^U = (\tilde{c}_1 \wedge \tilde{c}_2 \wedge \tilde{c}_3) \vee \tilde{c}_1 \wedge \tilde{c}_2 \wedge \tilde{c}_4$ , the candidates of B-reducts are,

$$\{c_1, c_2\}, \quad \{c_1, c_2, c_3\}, \quad \{c_1, c_2, c_4\}.$$

We can see that all of those satisfy (VPB1), hence,  $\{c_1, c_2\}$  is the unique B-reduct. Because  $\hat{F}_{0.39}^P = F_{0.39}^L = (\tilde{c}_1 \wedge \tilde{c}_2 \wedge \tilde{c}_3) \vee (\tilde{c}_1 \wedge \tilde{c}_2 \wedge \tilde{c}_4) \vee (\tilde{c}_1 \wedge \tilde{c}_3 \wedge \tilde{c}_4)$ , the candidates of P-reducts are,

$$\{c_1, c_2\}, \quad \{c_1, c_2, c_3\}, \quad \{c_1, c_2, c_4\}, \quad \{c_1, c_3, c_4\}.$$

Also, in that case, all candidates satisfy (VPP1), hence,  $\{c_1, c_2\}$  and  $\{c_1, c_3, c_4\}$  are P-reducts. Similarly, the candidates of UN-reducts are,

$$\{c_1, c_2\}, \quad \{c_2, c_3\}, \quad \{c_3, c_4\}, \quad \{c_1, c_2, c_4\}, \quad \{c_1, c_3, c_4\},$$

and all candidates satisfy (VPUN1), hence,  $\{c_1, c_2\}$ ,  $\{c_2, c_3\}$ , and  $\{c_3, c_4\}$  are UN-reducts.

All reducts are arranged in Table 7.6. We can observe that several kinds of reducts are different. In this example, each L-reduct is also an LUN-reduct and vice versa. Such an equivalence holds in this example but not always.

In this example, we would select  $\{c_1, c_2, c_3\}$  or  $\{c_1, c_2, c_4\}$  to preserve all structures. Additionally,  $c_1$  and  $c_2$  appear in many other reducts. Whereas, we would select U-reduct  $\{c_2, c_3\}$  to reduce the size of the reduct.

**Table 7.6** All obtained reducts with  $\beta = 0.39$  in Table 7.3

Type	Reducts
L-reduct	$\{c_1, c_2, c_3\}, \{c_1, c_2, c_4\}, \{c_1, c_3, c_4\}$
U-reduct	$\{c_2, c_3\}, \{c_1, c_2, c_4\}$
LU-reduct	$\{c_1, c_2, c_3\}, \{c_1, c_2, c_4\}$
B-reduct	$\{c_1, c_2\}$
P-reduct	$\{c_1, c_2\}, \{c_1, c_3, c_4\}$
UN-reduct	$\{c_1, c_2\}, \{c_2, c_3\}, \{c_3, c_4\}$
LUN-reduct	$\{c_1, c_2, c_3\}, \{c_1, c_2, c_4\}, \{c_1, c_3, c_4\}$
BUN-reduct	$\{c_1, c_2\}$
PUN-reduct	$\{c_1, c_2\}, \{c_1, c_3, c_4\}$

## 7.4 Structure-Based Attribute Reduction in Dominance-Based Rough Set Models

### 7.4.1 Decision Tables Under Dominance Principle and Dominance-Based Rough Set Models

In Dominance-based Rough Set Model (DRSM), known as Dominance-based Rough Set Approach [16, 18, 49], decision tables with order relations are analyzed. Let  $\mathbb{D} = (U, AT = C \cup \{d\}, \{V_a\}_{a \in AT})$  be a decision table. The attribute set  $AT$  is partitioned into  $AT_N$  and  $AT_C$ , where  $AT_N$  is the set of nominal attributes and  $AT_C$  is the set of criteria (ordinal attributes). For a criterion  $a \in AT_C$ , we suppose a total order  $\geq$  on its value set  $V_a$ . Moreover, all criteria are of the gain-type, i.e., the greater the better. We assume that the decision attribute  $d$  is a criterion.

In DRSM, it is supposed that if an object  $u$  is better than or equal to another object  $u'$  with respect to all condition attributes, then the class of  $u$  should not be worse than that of  $u'$ . This is called the dominance principle [16].

*Remark 8* The setting of DRSM is considered as the monotone or ordinal classification problem [2, 3, 32], where classifiers are restricted to be monotonic. Let  $f$  be a classifier, which assigns to each object  $u$  a class label (decision class value)  $f(u)$ . The classifier  $f$  is monotonic if for any object pair  $u$  and  $u'$ , we have  $u \leq u'$  implying  $f(u) \leq f(u')$ . In this chapter, however, we do not discuss classifiers nor algorithms for building classifiers.

*Remark 9* We assume the total order, i.e., antisymmetry, transitivity, and, comparability, on the value set  $V_a$  of each condition criteria  $a \in AT_C \cap C$ . However, regardless of comparability, the result of this section can be applied without modification. Additionally, we assume that all criteria are of the gain-type. However, in applications, we may encounter cost-type criteria, i.e., the smaller the better. For a cost-type criterion, we can deal with it as the gain-type by reversing the order of its values.

*Remark 10* Generally, there is more than one decision attribute in a decision table. In such a case, the set of decision classes (the partition of objects by the decision attributes) is partially ordered, while it is totally ordered in the case of a single decision attribute. In this section, we focus on the case of a single decision attribute (more generally, the case when the decision classes form a totally ordered set), however, the results of this section could be straightforwardly extended to that of multiple decision attributes.

For  $A \subseteq C$ , a dominance relation  $D_A$  on  $U$  is defined by:

$$D_A = \left\{ (u, u') \in U^2 \mid a(u) \geq a(u'), \forall a \in AT_C \cap A \right. \\ \left. \text{and } a(u) = a(u'), \forall a \in AT_N \cap A \right\}.$$

$D_A$  satisfies reflexivity and transitivity. When  $(u, u') \in D_A$ , we say that  $u$  dominates  $u'$  with respect to  $A$ . The relation  $(u, u') \in D_A$  means “ $u$  is better than or equal

to  $u'$  with respect to criteria  $A''$ . For  $u \in U$ , its dominating set and its dominated set with respect to  $A$  are defined, respectively, by:

$$\begin{aligned} D_A^+(u) &= \{u' \in U \mid (u', u) \in D_A\}, \\ D_A^-(u) &= \{u' \in U \mid (u, u') \in D_A\}. \end{aligned}$$

The dominating set  $D_A^+(u)$  (resp. the dominated set  $D_A^-(u)$ ) is the set of the objects dominating (resp. dominated by)  $u$  under  $A$ .

Since decision classes are ordered  $X_1 < X_2 < \dots < X_p$ , one can define an upward union of decision classes  $X_i^\geq$  and a downward union of decision classes  $X_i^\leq$  with respect to each class  $X_i$ ,  $i \in V_d$ , as follows:

$$X_i^\geq = \bigcup_{j \geq i} X_j, \quad X_i^\leq = \bigcup_{j \leq i} X_j.$$

For convenience,  $X_0^\leq = X_{p+1}^\geq = \emptyset$ . We have  $X_i^\geq = U \setminus X_{i-1}^\leq$ .

*Example 10* Consider a decision table  $\mathbb{D} = (U, C \cup \{d\}, \{V_a\})$  given in Table 7.7. This table shows student evaluation in a school. The objects are seven students, i.e.,  $U = \{u_1, u_2, \dots, u_7\}$ . The condition attributes are scores of mathematics (Ma), physics (Ph) and literature (Li), while the decision attribute ( $d$ ) is a comprehensive evaluation (E). Namely,  $C = \{\text{Ma}, \text{Ph}, \text{Li}\}$  and  $d = \text{E}$ . We may assume that the better scores in all subjects student takes, the better comprehensive evaluation he/she gets.

Let  $A = \{\text{Ma}, \text{Ph}\}$ . The dominance relation  $D_A$  is described as the following matrix. Symbol \* indicates that the corresponding row object  $u_i$  and column object  $u_j$  is in the dominance relation, i.e.,  $(u_i, u_j) \in D_A$ .

	$u_1$	$u_2$	$u_3$	$u_4$	$u_5$	$u_6$	$u_7$
$u_1$	*	*	*	*	*	*	*
$u_2$	*	*	*	*	*	*	*
$u_3$		*	*	*	*	*	
$u_4$			*		*		
$u_5$				*	*	*	
$u_6$				*	*	*	
$u_7$						*	

**Table 7.7** A decision table of student records

Student	Ma	Ph	Li	E
$u_1$	Good	Good	Good	Good
$u_2$	Good	Good	Med	Med
$u_3$	Med	Good	Med	Good
$u_4$	Bad	Med	Good	Med
$u_5$	Med	Bad	Med	Bad
$u_6$	Med	Bad	Bad	Med
$u_7$	Bad	Bad	Bad	Bad

For each object  $u_i \in U$ , symbols \* in the row of  $u_i$  indicate the objects in  $D^-(u_i)$ , while symbols \* in the column indicate the objects in  $D^+(u_i)$ . For example,  $D^-(u_3) = \{u_3, u_4, u_5, u_6, u_7\}$  and  $D^+(u_3) = \{u_1, u_2, u_3\}$ .

There are three decision classes  $X_b = \{u_5, u_7\}$ ,  $X_m = \{u_2, u_4, u_6\}$  and  $X_g = \{u_1, u_3\}$  for bad, med and good, respectively. The upward and downward unions of those decision classes are,

$$\begin{aligned} X_b^\geq &= U, \quad X_m^\geq = \{u_1, u_2, u_3, u_4, u_6\}, \quad X_g^\geq = \{u_1, u_3\}, \\ X_b^\leq &= \{u_5, u_7\}, \quad X_m^\leq = \{u_2, u_4, u_5, u_6, u_7\}, \quad X_g^\leq = U. \end{aligned}$$

Given a decision table, the inconsistency with respect to the dominance principle is captured by the difference between upper and lower approximations of the unions of decision classes. Given a condition attribute set  $A \subseteq C$ , and  $i \in V_d$ , the lower approximation  $\text{LA}_A(X_i^\geq)$  and the upper approximation  $\text{UA}_A(X_i^\geq)$  of  $X_i^\geq$  are defined, respectively, by:

$$\begin{aligned} \text{LA}_A(X_i^\geq) &= \{u \in U \mid D_A^+(u) \subseteq X_i^\geq\}, \\ \text{UA}_A(X_i^\geq) &= \{u \in U \mid D_A^-(u) \cap X_i^\geq \neq \emptyset\}. \end{aligned}$$

Similarly, the lower approximation  $\text{LA}_A(X_i^\leq)$  of  $X_i^\leq$  and upper approximation  $\text{UA}_A(X_i^\leq)$  are defined, respectively, by:

$$\begin{aligned} \text{LA}_A(X_i^\leq) &= \{u \in U \mid D_A^-(u) \subseteq X_i^\leq\}, \\ \text{UA}_A(X_i^\leq) &= \{u \in U \mid D_A^+(u) \cap X_i^\leq \neq \emptyset\}. \end{aligned}$$

If  $u$  belongs to  $\text{LA}_A(X_i^\geq)$  then all objects dominating  $u$  do not belong to  $X_{i-1}^\leq$ , i.e., there exists no evidence for  $u \in X_{i-1}^\leq$  in view of the monotonicity assumption. Therefore, we can say that  $u$  certainly belongs to  $X_i^\geq$ . On the other hand if  $u$  belongs to  $\text{UA}_A(X_i^\geq)$  then  $u$  is dominating an object belonging to  $X_i^\geq$ , i.e., there exists evidence for  $u \in X_i^\geq$  in view of the monotonicity assumption. Therefore, we can say that  $u$  possibly belongs to  $X_i^\geq$ . The similar interpretations can be applied to  $\text{LA}_A(X_i^\leq)$  and  $\text{UA}_A(X_i^\leq)$ .

The difference between the upper and lower approximations is called a boundary. The boundaries of an upward union  $X_i^\geq$  and a downward union  $X_i^\leq$ , denoted by  $\text{BN}_A(X_i^\geq)$  and  $\text{BN}_A(X_i^\leq)$ , are defined by:

$$\begin{aligned} \text{BN}_A(X_i^\geq) &= \text{UA}_A(X_i^\geq) \setminus \text{LA}_A(X_i^\geq), \\ \text{BN}_A(X_i^\leq) &= \text{UA}_A(X_i^\leq) \setminus \text{LA}_A(X_i^\leq). \end{aligned}$$

Objects in the boundary region of an upward or downward union are classified neither to that union nor to the complement with certainty.

*Example 11* Remember the decision table  $\mathbb{D} = (U, C \cup \{d\}, \{V_d\})$  in Table 7.7. Let  $A = \{\text{Ma, Ph}\}$ . The lower and upper approximations of the upward and downward unions with respect to  $A$  are obtained as follows.

$$\begin{aligned} \text{LA}_A(X_b^{\geq}) &= U, \quad \text{LA}_A(X_{\bar{m}}^{\geq}) = \{u_1, u_2, u_3, u_4\}, \quad \text{LA}_A(X_g^{\geq}) = \emptyset, \\ \text{LA}_A(X_b^{\leq}) &= \{u_7\}, \quad \text{LA}_A(X_{\bar{m}}^{\leq}) = \{u_4, u_5, u_6, u_7\}, \quad \text{LA}_A(X_g^{\leq}) = U, \\ \text{UA}_A(X_b^{\geq}) &= U, \quad \text{UA}_A(X_{\bar{m}}^{\geq}) = U \setminus \{u_7\}, \quad \text{UA}_A(X_g^{\geq}) = \{u_1, u_2, u_3\}, \\ \text{UA}_A(X_b^{\leq}) &= \{u_5, u_6, u_7\}, \quad \text{UA}_A(X_{\bar{m}}^{\leq}) = U, \quad \text{UA}_A(X_g^{\leq}) = U. \end{aligned}$$

Now, we remember properties of approximations [16, 18, 31]. By the boundary conditions of  $X^{\geq}$  and  $X^{\leq}$ ,

$$\begin{aligned} \text{UA}_A(X_1^{\geq}) &= \text{LA}_A(X_1^{\geq}) = U, \quad \text{UA}_A(X_p^{\leq}) = \text{LA}_A(X_p^{\leq}) = U, \\ \text{UA}_A(X_{p+1}^{\geq}) &= \text{LA}_A(X_{p+1}^{\geq}) = \emptyset, \quad \text{UA}_A(X_0^{\leq}) = \text{LA}_A(X_0^{\leq}) = \emptyset. \end{aligned} \quad (7.21)$$

Let  $A \subseteq C$  and  $i \in V_d$ . Similarly to RSM, there exist inclusion relations between each union of decision classes and its lower and upper approximations.

$$\text{LA}_A(X_i^{\geq}) \subseteq X_i^{\geq} \subseteq \text{UA}_A(X_i^{\geq}), \quad \text{LA}_A(X_i^{\leq}) \subseteq X_i^{\leq} \subseteq \text{UA}_A(X_i^{\leq}). \quad (7.22)$$

Approximations are expressed by unions of dominating or dominated sets,

$$\begin{aligned} \text{LA}_A(X_i^{\geq}) &= \bigcup_{D_A^+(u) \subseteq X_i^{\geq}} D_A^+(u) = \bigcup_{u \in \text{LA}_A(X_i^{\geq})} D_A^+(u), \\ \text{UA}_A(X_i^{\geq}) &= \bigcup_{D_A^-(u) \cap X_i^{\geq} \neq \emptyset} D_A^-(u) = \bigcup_{u \in \text{UA}_A(X_i^{\geq})} D_A^-(u), \\ \text{LA}_A(X_i^{\leq}) &= \bigcup_{D_A^-(u) \subseteq X_i^{\leq}} D_A^-(u) = \bigcup_{u \in \text{LA}_A(X_i^{\leq})} D_A^-(u), \\ \text{UA}_A(X_i^{\leq}) &= \bigcup_{D_A^+(u) \cap X_i^{\leq} \neq \emptyset} D_A^-(u) = \bigcup_{u \in \text{UA}_A(X_i^{\leq})} D_A^-(u). \end{aligned}$$

There exists duality of lower and upper approximations.

$$\text{UA}_A(X_i^{\geq}) = U \setminus \text{LA}_A(X_{i-1}^{\leq}), \quad \text{UA}_A(X_i^{\leq}) = U \setminus \text{LA}_A(X_{i+1}^{\geq}). \quad (7.23)$$

So, the upper approximations of the pair of complementary unions of decision classes form a cover of  $U$ :

$$\text{UA}_A(X_i^{\geq}) \cup \text{UA}_A(X_{i-1}^{\leq}) = U. \quad (7.24)$$

By the duality of lower and upper approximations, the boundaries of the pair of complementary unions are the same,

$$\text{BN}_A(X_i^{\geq}) = \text{BN}_A(X_{i-1}^{\leq}). \quad (7.25)$$

Lower and upper approximations can be expressed by boundaries. That is useful for investigating relations between different types of reducts:

$$\text{UA}_A(X_i^{\geq}) = \text{BN}_A(X_i^{\geq}) \cup X_i^{\geq}, \quad \text{UA}_A(X_i^{\leq}) = \text{BN}_A(X_i^{\leq}) \cup X_i^{\leq}, \quad (7.26)$$

$$\text{LA}_A(X_i^{\geq}) = X_i^{\geq} \setminus \text{BN}_A(X_i^{\geq}), \quad \text{LA}_A(X_i^{\leq}) = X_i^{\leq} \setminus \text{BN}_A(X_i^{\leq}). \quad (7.27)$$

Let  $A, B \subseteq C$  and  $i, j \in V_d$ . Then, we have the following monotonicity properties:

$$j \geq i \Rightarrow \text{LA}_A(X_j^{\geq}) \subseteq \text{LA}_A(X_i^{\geq}), \quad \text{UA}_A(X_j^{\geq}) \subseteq \text{UA}_A(X_i^{\geq}), \quad (7.28)$$

$$j \leq i \Rightarrow \text{LA}_A(X_j^{\leq}) \subseteq \text{LA}_A(X_i^{\leq}), \quad \text{UA}_A(X_j^{\leq}) \subseteq \text{UA}_A(X_i^{\leq}), \quad (7.29)$$

$$B \subseteq A \Rightarrow \text{LA}_B(X_i^{\geq}) \subseteq \text{LA}_A(X_i^{\geq}), \quad \text{LA}_B(X_i^{\leq}) \subseteq \text{LA}_A(X_i^{\leq}), \quad (7.30)$$

$$B \subseteq A \Rightarrow \text{UA}_B(X_i^{\geq}) \supseteq \text{UA}_A(X_i^{\geq}), \quad \text{UA}_B(X_i^{\leq}) \supseteq \text{UA}_A(X_i^{\leq}). \quad (7.31)$$

Those are important for defining and enumerating reducts.

Furthermore, the authors proposed lower and upper approximations and boundary regions of decision classes [31]. For  $A \subseteq C$  and  $i \in V_d$ , lower and upper approximations of  $X_i$  and the boundary region of  $X_i$  are defined by:

$$\text{LA}_A(X_i) = \text{LA}_A(X_i^{\geq}) \cap \text{LA}_A(X_i^{\leq}),$$

$$\text{UA}_A(X_i) = \text{UA}_A(X_i^{\geq}) \cap \text{UA}_A(X_i^{\leq}),$$

$$\text{BN}_A(X_i) = \text{UA}_A(X_i) \setminus \text{LA}_A(X_i).$$

This definition is an analogy to  $X_i = X_i^{\geq} \cap X_i^{\leq}$ .

Let  $A \subseteq C$  and  $i \in V_d$ . The upper approximations of  $X_i^{\geq}$  and  $X_i^{\leq}$  are represented by upper approximations of decision classes:

$$\text{UA}_A(X_i^{\geq}) = \bigcup_{j \geq i} \text{UA}_A(X_j), \quad (7.32)$$

$$\text{UA}_A(X_i^{\leq}) = \bigcup_{j \leq i} \text{UA}_A(X_j). \quad (7.33)$$

The boundary of  $X_i$  is the union of the boundaries of  $X_i^{\geq}$  and  $X_i^{\leq}$ ,

$$\text{BN}_A(X_i) = \text{BN}_A(X_i^{\geq}) \cup \text{BN}_A(X_i^{\leq}). \quad (7.34)$$

Approximations of decision classes have similar properties as those of unions of decision classes:

$$\text{LA}_A(X_i) \subseteq X_i \subseteq \text{UA}_A(X_i), \quad (7.35)$$

$$\text{UA}_A(X_i) = \text{BN}_A(X_i) \cup X_i, \quad (7.36)$$

$$\text{LA}_A(X_i) = X_i \setminus \text{BN}_A(X_i). \quad (7.37)$$

The next properties are analogies to (7.6) and (7.5) of the classical RSM.

$$\text{BN}_A(X_i) = \text{UA}_A(X_i) \cap \bigcup_{j \neq i} \text{UA}_A(X_j), \quad (7.38)$$

$$\text{LA}_A(X_i) = U \setminus \bigcup_{j \neq i} \text{UA}_A(X_j). \quad (7.39)$$

We define the positive region for the decision table in DRSM:

$$\text{POS}_A(d) = \bigcup_{i \in V_d} \text{LA}_A(X_i).$$

The complement of the positive region is exactly the union of all boundaries,

$$U \setminus \text{POS}_A(d) = \bigcup_{i \in V_d} \text{BN}_A(X_i). \quad (7.40)$$

Moreover, the approximations are also monotone with respect to the inclusion relation between condition attribute sets. Let  $A, B \subseteq C$  and  $i \in V_d$ .

$$B \subseteq A \Rightarrow \text{LA}_B(X_i) \subseteq \text{LA}_A(X_i), \quad \text{UA}_B(X_i) \supseteq \text{UA}_A(X_i). \quad (7.41)$$

The generalized decision function proposed by Dembczyński et al. [10] also plays an important role for Boolean reasoning in DRSM. It provides an object-wise view of DRSM. Let  $A \subseteq C$  and  $u \in U$ , generalized decision of  $u$  with respect to  $A$  is defined by  $\delta_A(u) = \langle l_A(u), u_A(u) \rangle$ , where

$$\begin{aligned} l_A(u) &= \min\{i \in V_d \mid D_A^+(u) \cap X_i \neq \emptyset\}, \\ u_A(u) &= \max\{i \in V_d \mid D_A^-(u) \cap X_i \neq \emptyset\}. \end{aligned}$$

$\delta_A(u)$  shows the interval of decision classes to which  $x$  may belong.  $l_A(u)$  and  $u_A(u)$  are the lower and upper bounds of the interval. Obviously, we have

$$l_A(u) \leq u_A(u). \quad (7.42)$$

$l_A(u)$  and  $u_A(u)$  are monotone with respect to the inclusion relation between condition attribute sets. Namely, for  $B, A \subseteq C$  and  $u \in U$ , we have

$$B \subseteq A \Rightarrow l_B(u) \leq l_A(u), \quad u_B(u) \geq u_A(u). \quad (7.43)$$

Let  $i \in V_d$ , using the generalized decision function, the lower and upper approximations of unions are represented as:

$$\text{LA}_A(X_i^{\geq}) = \{u \in U \mid l_A(u) \geq i\}, \quad \text{UA}_A(X_i^{\geq}) = \{u \in U \mid u_A(u) \geq i\}, \quad (7.44)$$

$$\text{LA}_A(X_i^{\leq}) = \{u \in U \mid u_A(u) \leq i\}, \quad \text{UA}_A(X_i^{\leq}) = \{u \in U \mid l_A(u) \leq i\}. \quad (7.45)$$

We can represent approximations of classes using the generalized decision,

$$\text{LA}_A(X_i) = \{u \in U \mid l_A(u) = u_A(u) = i\}, \quad (7.46)$$

$$\text{UA}_A(X_i) = \{u \in U \mid l_A(u) \leq i \leq u_A(u)\}, \quad (7.47)$$

$$\text{BN}_A(X_i) = \{u \in U \mid l_A(u) \leq i \leq u_A(u), l_A(u) < u_A(u)\}. \quad (7.48)$$

*Example 12* Remember the decision table  $\mathbb{D} = (U, C \cup \{d\}, \{V_a\})$  in Table 7.7. Let  $A = \{\text{Ma}, \text{Ph}\}$ . The generalized decision function  $\delta_A$  with respect to  $A$  is obtained as follows:

$$\begin{aligned} \delta_A(u_1) &= \langle \text{med}, \text{good} \rangle, & \delta_A(u_2) &= \langle \text{med}, \text{good} \rangle, & \delta_A(u_3) &= \langle \text{med}, \text{good} \rangle, \\ \delta_A(u_4) &= \langle \text{med}, \text{med} \rangle, & \delta_A(u_5) &= \langle \text{bad}, \text{med} \rangle, & \delta_A(u_6) &= \langle \text{bad}, \text{med} \rangle, \\ \delta_A(u_7) &= \langle \text{bad}, \text{bad} \rangle. \end{aligned}$$

#### 7.4.2 Structure-Based Reducts in Dominance-Based Rough Set Models

Before defining structure-based reducts in DRSM, we introduce a notion of reducts preserving the quality of sorting, proposed by Susmaga et al. [49]. For  $A \subseteq C$ , the quality of sorting  $\gamma_A(d)$ , which is the counterpart of the quality of classification in the classical RSM, is defined by:

$$\gamma_A(d) = \frac{|U - \bigcup_{i \in V_d} \text{BN}_A(X_i^{\leq})|}{|U|} = \frac{|U - \bigcup_{i \in V_d} \text{BN}_A(X_i^{\geq})|}{|U|}.$$

By (7.34) and (7.40), we can see that  $\gamma_A(d)$  is related to the positive region of DRSM,

$$\gamma_A(d) = \frac{|U - \bigcup_{i \in V_d} \text{BN}_A(X_i^{\leq})|}{|U|} = \frac{|U - \bigcup_{i \in V_d} \text{BN}_A(X_i)|}{|U|} = \frac{|\text{POS}_A(d)|}{|U|}.$$

We call this type of reducts Q-reducts.

**Definition 15** ([16, 49]) A Q-reduct in DRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\gamma_A(d) = \gamma_C(d). \quad (\text{DQ})$$

Now, we introduce structure-based reducts in DRSM. Lower and upper approximations and boundary regions of upward and downward unions can be considered as a structure over a given object set  $U$ . From this point, we define 7 union-structure-preserving reducts. The following reducts are conceivable.

**Definition 16** ([25, 52]) We define 7 types of reducts as follows.

- An  $L^{\geq}$ -reduct in DRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\text{LA}_A(X_i^{\geq}) = \text{LA}_C(X_i^{\geq}) \quad \text{for all } i \in V_d. \quad (\text{DL}^{\geq})$$

- An  $L^{\leq}$ -reduct in DRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\text{LA}_A(X_i^{\leq}) = \text{LA}_C(X_i^{\leq}) \quad \text{for all } i \in V_d. \quad (\text{DL}^{\leq})$$

- A  $U^{\geq}$ -reduct in DRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\text{UA}_A(X_i^{\geq}) = \text{UA}_C(X_i^{\geq}) \quad \text{for all } i \in V_d. \quad (\text{DU}^{\geq})$$

- A  $U^{\leq}$ -reduct in DRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\text{UA}_A(X_i^{\leq}) = \text{UA}_C(X_i^{\leq}) \quad \text{for all } i \in V_d. \quad (\text{DU}^{\leq})$$

- An  $L^{\diamond}$ -reduct in DRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\text{LA}_A(X_i^{\geq}) = \text{LA}_C(X_i^{\geq}) \quad \text{and} \quad \text{LA}_A(X_i^{\leq}) = \text{LA}_C(X_i^{\leq}) \quad \text{for all } i \in V_d. \quad (\text{DL}^{\diamond})$$

- A  $U^{\diamond}$ -reduct in DRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\text{UA}_A(X_i^{\geq}) = \text{UA}_C(X_i^{\geq}) \quad \text{and} \quad \text{UA}_A(X_i^{\leq}) = \text{UA}_C(X_i^{\leq}) \quad \text{for all } i \in V_d. \quad (\text{DU}^{\diamond})$$

- A  $B^{\diamond}$ -reduct in DRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\begin{aligned} \text{BN}_A(X_i^{\geq}) &= \text{BN}_C(X_i^{\geq}) && \text{for all } i \in V_d, \text{ or equivalently,} \\ \text{BN}_A(X_i^{\leq}) &= \text{BN}_C(X_i^{\leq}) && \text{for all } i \in V_d. \end{aligned} \quad (\text{DB}^{\diamond})$$

Yang et al. [52] independently proposed four kinds of reducts in DRSM with unknown attribute values, which are application of distribution reducts of Mi et al. [33]. Those reducts preserve lower/upper approximations of upward/downward unions. Hence, they correspond to  $L^{\geq}$ -,  $L^{\leq}$ -,  $U^{\geq}$ -, and  $U^{\leq}$ -reducts of ours. However, Yang et al. did not consider boundaries and combinations of different types of reducts.

From (7.23), we know that  $(DL^{\geq})$  and  $(DU^{\leq})$  are equivalent. Similarly,  $(DL^{\leq})$  and  $(DU^{\geq})$  are also equivalent. Therefore,  $(DL^{\diamond})$  is equivalent to  $(DU^{\diamond})$ . Moreover, since condition  $(DL^{\diamond})$  implies conditions  $(DL^{\geq})$  and  $(DL^{\leq})$ , any  $L^{\diamond}$ -reduct satisfies  $(DL^{\geq})$  and also  $(DL^{\leq})$ . Similarly, since condition  $(DU^{\diamond})$  implies conditions  $(DU^{\geq})$  and  $(DU^{\leq})$ , any  $U^{\diamond}$ -reduct satisfies  $(DU^{\geq})$  and also  $(DU^{\leq})$ . Therefore, we have the following theorem.

**Theorem 7** ([25, 52]) Let  $A$  be a subset of  $C$ . The following statements hold.

- $A$  is a  $U^{\geq}$ -reduct if and only if  $A$  is an  $L^{\leq}$ -reduct.
- $A$  is a  $U^{\leq}$ -reduct if and only if  $A$  is an  $L^{\geq}$ -reduct.
- $A$  is a  $U^{\diamond}$ -reduct if and only if  $A$  is an  $L^{\diamond}$ -reduct.
- $A$  is a  $B^{\diamond}$ -reduct if and only if  $A$  is an  $L^{\diamond}$ -reduct.
- If  $A$  is an  $L^{\diamond}$ -reduct then  $A$  satisfies  $(DL^{\geq})$ ,  $(DL^{\leq})$ ,  $(DU^{\geq})$ , and  $(DU^{\leq})$ .

As the result of the discussion, we obtain 3 different types of reducts based on the structure induced from rough set operations on unions. They are represented by  $L^{\geq}$ -reduct,  $L^{\leq}$ -reduct and  $L^{\diamond}$ -reduct.

Now, we are ready to define other types of structure-based reducts, considering approximations of decision classes. The first kind of reducts, called L-reduct, preserves the lower approximations of decision classes, the second kind of reducts, called U-reduct, preserves the upper approximations of decision classes, the third kind of reduct, called B-reduct, preserves the boundary regions of decision classes, and the fourth kind of reduct, called P-reduct, preserves the positive region. They are parallel to L-, U-, B-, P-reducts discussion in the classical RSM.

**Definition 17** ([31]) We define four types of reducts as follows.

- An L-reduct in DRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\text{LA}_A(X_i) = \text{LA}_C(X_i) \quad \text{for all } i \in V_d. \quad (\text{DL})$$

- A U-reduct in DRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\text{UA}_A(X_i) = \text{UA}_C(X_i) \quad \text{for all } i \in V_d. \quad (\text{DU})$$

- A B-reduct in DRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\text{BN}_A(X_i) = \text{BN}_C(X_i) \quad \text{for all } i \in V_d. \quad (\text{DB})$$

- A P-reduct in DRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\text{POS}_A(d) = \text{POS}_C(d). \quad (\text{DP})$$

From the properties of approximations of decision classes, we have the following theorem.

**Theorem 8** ([31]) *Let  $A$  be a subset of  $C$ . We have the following assertions:*

- $A$  is a B-reduct if and only if  $A$  is a U-reduct,*
- $A$  is a P-reduct if and only if  $A$  is an L-reduct,*
- If  $A$  is a U-reduct then  $A$  satisfies (DL).*

Consequently, we have only 2 kinds of class-structure-based reducts: L-reducts and U-reducts (or B-reducts). This result is also parallel to the result in RSM.

Let us discuss relations of the union-based reducts, the class-based reducts, the Q-reducts. We have the following theorems.

**Theorem 9** ([31]) *Let  $A$  be a subset of  $C$ . We have the following assertions:*

- $A$  is an  $L^\diamond$ -reduct if and only if  $A$  is a U-reduct,*
- $A$  is a Q-reduct if and only if  $A$  is an L-reduct.*

Additionally, we propose two more types of reducts, which are compounds of L- with  $L^\geq$ - and  $L^\leq$ -reducts, respectively.

**Definition 18** We define two types of reducts as follows.

- An  $LL^\geq$ -reduct in DRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

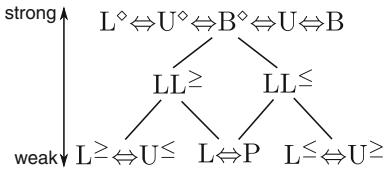
$$\text{LA}_A(X_i) = \text{LA}_C(X_i) \text{ and } \text{LA}_A(X_i^\geq) = \text{LA}_C(X_i^\geq) \quad \text{for all } i \in V_d. \quad (\text{DLL}^\geq)$$

- A  $LL^\leq$ -reduct in DRSM is a minimal condition attribute subset  $A \subseteq C$  satisfying the following condition:

$$\text{UA}_A(X_i) = \text{UA}_C(X_i) \text{ and } \text{LA}_A(X_i^\leq) = \text{LA}_C(X_i^\leq) \quad \text{for all } i \in V_d. \quad (\text{DLL}^\leq)$$

As a result, all types of reducts proposed in DRSM are arranged in Fig. 7.3. Consequently, there exist six different kinds of reducts, i.e., U-reducts (B-reducts,  $L^\diamond$ -reducts,  $U^\diamond$ -reducts,  $B^\diamond$ -reducts),  $LL^\geq$ -reducts,  $LL^\leq$ -reducts, L-reducts (P-reducts),  $L^\geq$ -reducts ( $U^\leq$ -reducts) and  $L^\leq$ -reducts ( $U^\geq$ -reducts) in DRSM.

**Fig. 7.3** Strong-weak hierarchy of reducts in DRSM



### 7.4.3 Boolean Functions Representing Reducts

Because Boolean reasoning is a popular approach to enumerate all reducts of each type in rough set literature, some authors already showed Boolean functions representing their own types of reducts [49, 52]. On the other hand, the authors proposed a unified formulation of Boolean functions for all types of reducts using the generalized decision function in [31]. We only discuss Boolean functions for  $L^{\geq}$ -,  $L^{\leq}$ - and  $L$ -reducts, because  $U$ -reducts,  $LL^{\geq}$ -reducts,  $LL^{\leq}$ -reducts, and their equivalences can be computed from  $L^{\geq}$ - and  $L^{\leq}$ -reducts or their Boolean functions.

We represent preserving conditions of reducts by those of the generalized decision function.

**Lemma 5** ([31]) Let  $A$  be a subset of  $C$ . We have the following assertions.

- Condition  $(DL^{\geq})$  is equivalent to:

$$l_A(u) = l_C(u) \quad \text{for all } u \in U. \quad (\text{DlG})$$

- Condition  $(DL^{\leq})$  is equivalent to:

$$u_A(u) = u_C(u) \quad \text{for all } u \in U. \quad (\text{DuG})$$

- Condition  $(DL)$  is equivalent to:

$$\delta_A(u) = \delta_C(u) \quad \text{for all } u \in U \text{ such that } l_C(u) = u_C(u). \quad (\text{DLG})$$

The next lemma is parallel to Lemma 2 of RSM. It also connects two notions: “preserving” and “non-dominating”.

**Lemma 6** ([31]) Let  $u \in U$ . The following assertions are equivalent.

- $l_A(u) = l_C(u)$ .
- $\forall u' \in U, (l_C(u') < l_C(u)) \Rightarrow \exists a \in A, (u', u) \notin D_{\{a\}}$ .

Moreover, the following assertions are also equivalent.

- $u_A(u) = u_C(u)$ .
- $\forall u' \in U, (u_C(u') > u_C(u)) \Rightarrow \exists a \in A, (u, u') \notin D_{\{a\}}$ .

Now we are ready to define a non-domination matrix, instead of the discernibility matrix of RSM. The non-domination matrix  $M = (m_{ij})_{i,j=1,2,\dots,n}$  in DRSM is defined as follows:

$$m_{ij} = \{c \in C \mid (u_j, u_i) \notin D_{\{c\}}\}$$

Based on  $M$ , we define four non-domination functions.

**Definition 19** Non-domination functions  $F^{\geq}$ ,  $F^{\leq}$  and  $F^L$  are defined as follows.

$$\begin{aligned} F^{\geq}(\tilde{c}_1, \dots, \tilde{c}_m) &= \bigwedge_{i,j \mid l_C(u_j) < l_C(u_i)} \bigvee_{c \in m_{ij}} \tilde{c}, \\ F^{\leq}(\tilde{c}_1, \dots, \tilde{c}_m) &= \bigwedge_{i,j \mid u_C(u_j) > u_C(u_i)} \bigvee_{c \in m_{ji}} \tilde{c}, \\ F^L(\tilde{c}_1, \dots, \tilde{c}_m) &= \bigwedge_{i: l_C(x_i) = u_C(x_i)} \left( \bigwedge_{j \mid l_C(u_j) < l_C(u_i)} \bigvee_{c \in m_{ij}} \tilde{c} \wedge \bigwedge_{j \mid u_C(u_j) > u_C(u_i)} \bigvee_{c \in m_{ji}} \tilde{c} \right), \end{aligned}$$

where  $\tilde{c}_i$  is a Boolean variable corresponding to  $i$ th condition attribute  $c_i$ .

From Lemma 6, we have the following theorem. Let  $A \subseteq C$ . Remember that  $\tilde{c}^A$  is a Boolean vector such that  $i$ th element  $\tilde{c}_i^A$  is true iff  $c_i \in A$ , and  $\phi_A$  is the term  $\bigwedge \{\tilde{c} \mid c \in A\}$ .

**Theorem 10** ([31, 49, 52]) *Let  $A$  be a subset of  $C$ . We have the following equivalences:*

- *$A$  satisfies (DLG), i.e.,  $(DL^{\geq})$  if and only if  $F^{\geq}(\tilde{c}^A) = 1$ . Moreover,  $A$  is an  $L^{\geq}$ -reduct in DRSM if and only if  $\phi_A$  is a prime implicant of  $F^{\geq}$ ,*
- *$A$  satisfies (DUG), i.e.,  $(DL^{\leq})$  if and only if  $F^{\leq}(\tilde{c}^A) = 1$ . Moreover,  $A$  is an  $L^{\leq}$ -reduct in DRSM if and only if  $\phi_A$  is a prime implicant of  $F^{\leq}$ ,*
- *$A$  satisfies (DLG), i.e.,  $(DL)$  if and only if  $F^L(\tilde{c}^A) = 1$ . Moreover,  $A$  is an  $L$ -reduct in DRSM if and only if  $\phi_A$  is a prime implicant of  $F^L$ .*

From Theorem 10, all  $L^{\geq}$ -,  $L^{\leq}$ - and  $L$ -reducts can be obtained as all prime implicants of Boolean functions  $F^{\geq}$ ,  $F^{\leq}$  and  $F^L$ , respectively.

The proposed non-domination matrices have an advantage when compared with the previous ones. We need to calculate neither lower, upper approximations nor boundary regions of unions but only the lower bounds  $l_C$  and the upper bounds  $u_C$  of all objects. Namely, the computation of the proposed approach is free from the number of decision classes.

**Example 13** Remember the decision table given  $\mathbb{D} = (U, C \cup \{d\}, \{V_a\})$  in Table 7.7. In Table 7.8, we show again  $\mathbb{D}$  with the lower bounds  $l_C$  and the upper bounds  $u_C$  of the generalized decisions of the objects which appear in the rightmost two columns of the table. To obtain  $l_C(u_i)$  and  $u_C(u_i)$ , we search the minimum class in  $D_C^+(u_i)$  and the maximum class in  $D_C^-(u_i)$ , respectively.

**Table 7.8** The decision table in Table 7.7 with the generalized decision function

Student	Ma	Ph	Li	E	$\delta_C = \langle l_C, u_C \rangle$
$u_1$	Good	Good	Good	Good	$\langle \text{good}, \text{good} \rangle$
$u_2$	Good	Good	Med	Med	$\langle \text{med}, \text{good} \rangle$
$u_3$	Med	Good	Med	Good	$\langle \text{med}, \text{good} \rangle$
$u_4$	Bad	Med	Good	Med	$\langle \text{med}, \text{med} \rangle$
$u_5$	Med	Bad	Med	Bad	$\langle \text{bad}, \text{med} \rangle$
$u_6$	Med	Bad	Bad	Med	$\langle \text{bad}, \text{med} \rangle$
$u_7$	Bad	Bad	Bad	Bad	$\langle \text{bad}, \text{bad} \rangle$

Non-domination matrices  $M$  are obtained as follows.

	$u_1$	$u_2$	$u_3$	$u_4$	$u_5$	$u_6$	$u_7$
$u_1^*$	$\emptyset$	$\{ \text{Li} \}$	$\{ \text{Ma}, \text{Li} \}$	$\{ \text{Ma}, \text{Ph} \}$	$C$	$C$	$C$
$u_2$	$\emptyset$	$\emptyset$	$\{ \text{Ma} \}$	$\{ \text{Ma}, \text{Ph} \}$	$\{ \text{Ma}, \text{Ph} \}$	$C$	$C$
$u_3$	$\emptyset$	$\emptyset$	$\emptyset$	$\{ \text{Ma}, \text{Ph} \}$	$\{ \text{Ph} \}$	$\{ \text{Ph}, \text{Li} \}$	$C$
$u_4^*$	$\emptyset$	$\{ \text{Li} \}$	$\{ \text{Li} \}$	$\emptyset$	$\{ \text{Ph}, \text{Li} \}$	$\{ \text{Ph}, \text{Li} \}$	$\{ \text{Ph}, \text{Li} \}$
$u_5$	$\emptyset$	$\emptyset$	$\emptyset$	$\{ \text{Ma} \}$	$\emptyset$	$\{ \text{Li} \}$	$\{ \text{Ma}, \text{Li} \}$
$u_6$	$\emptyset$	$\emptyset$	$\emptyset$	$\{ \text{Ma} \}$	$\emptyset$	$\emptyset$	$\{ \text{Ma} \}$
$u_7^*$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

For example, the entry corresponding to row  $u_1$  and column  $u_3$  on  $M$  contains Ma and Li, because  $u_3$  is worse than  $u_1$  with respect to Ma and Li but not worse with respect to Ph. Symbol  $C$  at some entries means  $\{ \text{Ma}, \text{Ph}, \text{Li} \}$ . The rows with symbol \* show objects  $u_i$  such that  $l_C(u_i) = u_C(u_i)$ .

The Boolean function  $F^\geq$  is obtained from  $M$  as

$$F^\geq(\tilde{\text{Ma}}, \tilde{\text{Ph}}, \tilde{\text{Li}}) = \bigwedge_{i=1, j=2, 3, \dots, 7} \bigvee_{c \in m_{ij}} \tilde{c} \wedge \bigwedge_{i=2, 3, 4, j=5, 6, 7} \bigvee_{c \in m_{ij}} \tilde{c} = \tilde{\text{Ph}} \wedge \tilde{\text{Li}}$$

From the last equation,  $F^\geq(\tilde{\text{Ma}}, \tilde{\text{Ph}}, \tilde{\text{Li}})$  = true only when  $\tilde{\text{Ph}} = \text{true}$  and  $\tilde{\text{Li}} = \text{true}$ . This implies that only  $\{ \text{Ma}, \text{Ph}, \text{Li} \}$  and  $\{ \text{Ph}, \text{Li} \}$  satisfy  $(\text{DL}^\geq)$  owing to Theorem 10. An  $L^\geq$ -reduct is a minimal set of condition attributes that satisfies  $(\text{DL}^\geq)$ . Therefore,  $\{ \text{Ph}, \text{Li} \}$  is a unique  $L^\geq$ -reduct. Moreover, the  $L^\geq$ -reduct corresponds to a unique prime implicant of  $F^\geq$ , i.e.,  $\tilde{\text{Ph}} \wedge \tilde{\text{Li}}$ .

Similarly, Boolean functions  $F^\leq$ ,  $F^U$  and  $F^L$  are

$$F^\leq(\tilde{\text{Ma}}, \tilde{\text{Ph}}, \tilde{\text{Li}}) = \bigwedge_{i=4, 5, 6, 7, j=1, 2, 3} \bigvee_{c \in m_{ji}} \tilde{c} \wedge \bigwedge_{i=7, j=1, 2, \dots, 6} \bigvee_{c \in m_{ji}} \tilde{c} = \tilde{\text{Ma}} \wedge \tilde{\text{Ph}},$$

$$\begin{aligned} F^L(\tilde{\text{Ma}}, \tilde{\text{Ph}}, \tilde{\text{Li}}) = & \bigwedge_{i=1, j=2, 3, \dots, 7} \bigvee_{c \in m_{ij}} \tilde{c} \wedge \bigwedge_{i=4, j=5, 6, 7} \bigvee_{c \in m_{ij}} \tilde{c} \wedge \bigwedge_{i=4, j=1, 2, 3} \bigvee_{c \in m_{ji}} \tilde{c} \\ & \wedge \bigwedge_{i=7, j=1, 2, \dots, 6} \bigvee_{c \in m_{ji}} \tilde{c} = \tilde{\text{Ma}} \wedge \tilde{\text{Li}}. \end{aligned}$$

Consequently, we obtain  $\{\text{Ph}, \text{Li}\}$  as the unique  $L^{\geq}$ -reduct,  $\{\text{Ma}, \text{Ph}\}$  as the unique  $L^{\leq}$ -reduct and  $\{\text{Ma}, \text{Li}\}$  as the unique  $L$ -reduct. Moreover,  $\{\text{Ph}, \text{Li}\} \cup \{\text{Ma}, \text{Ph}\} = \{\text{Ma}, \text{Ph}, \text{Li}\} = C$  is the unique  $U$ -reduct,  $\{\text{Ph}, \text{Li}\} \cup \{\text{Ma}, \text{Li}\} = \{\text{Ma}, \text{Ph}, \text{Li}\} = C$  is the unique  $LL^{\geq}$ -reduct, and  $\{\text{Ma}, \text{Ph}\} \cup \{\text{Ma}, \text{Li}\} = \{\text{Ma}, \text{Ph}, \text{Li}\} = C$  is the unique  $LL^{\leq}$ -reduct.

## 7.5 Concluding Remarks

In this chapter, we have studied structure-based attribute reduction as a rough set approach to the attribute selection/reduction problem. We have proposed several concepts of structure-based reducts. In the rough set model, there are 2 different types of reducts,  $U$ -reducts and  $L$ -reducts.  $U$ -reducts preserve generalized decisions  $\partial_C(u)$  for all objects  $u \in U$ , while  $L$ -reducts do so for all certain classified objects  $u$ , namely,  $|\partial_C(u)| = 1$ . The authors studied refinement of the hierarchy of structure-based reducts (Fig. 7.1) by interpolating reducts which preserve objects  $u$  whose generalized decisions are at most  $k$ , namely,  $|\partial_C(u)| \leq k$  [24]. The parameter  $k$  provides a trade-off between the size of a reduct and preserved information.

In VPRSM, because approximations may not be monotone with respect to the set inclusion of condition attributes, classifications of some objects become precise by reducing condition attributes. From that viewpoint, the authors have proposed enhancing reducts [21], which do not preserve but make classification more precise than that of all condition attributes.

Attribute reduction have been also studied in other extensions of the rough set model, e.g. tolerance-based RSM [44], RSM for decision tables with missing values [29, 30], Bayesian RSM [47], fuzzy RSM [27, 28], and variable precision DRSM [26]. However, in general, extensions of the rough set model drop some important properties of approximations. Therefore, in such models, reducts may not be represented by Boolean functions.

When a measure  $\gamma$  (e.g. Eq. 7.8) representing a part of consistency of a rough set model is given, we can define approximate measure-based reducts as follows:  $A \subseteq C$  is an approximate reduct if  $\gamma_A \geq (1 - \varepsilon)\gamma_C$  or  $\gamma_A \geq \gamma_C - \varepsilon$  for a small  $\varepsilon \geq 0$ . Several measures used for approximate measure-based reducts have been proposed, e.g. based on the number of discerned object pairs or the information entropy [45, 46, 51]. Comparing with structure-based reducts, approximate measure-based approach can easily control size of reducts, but we cannot expect which parts of the structure of the rough set model deteriorate by reduction.

We show that reducts are (approximately) represented by prime implicants of Boolean functions (or pairs of Boolean functions). To compute all reducts of a particular type, we solve the dualization problem (more precisely, positive DNF (or CNF) dualization) of the corresponding Boolean function. It probably cannot be solved in polynomial time (it can be solved in quasi-polynomial time with respect to the sizes of the input and the output [9]).

To apply attribute reduction of this chapter to real-world data sets, we notice the following three points. Firstly, we need additional measures to select the best reducts for applications, for example, minimizing the size of the reduct or the number of the equivalence classes given by the reduct and so on [1, 13, 27, 42, 48, 50]. Such an optimization problem cannot be generally solved in polynomial time. Therefore, there are heuristic methods computing one or a number of reducts which are near to optimal [1, 27, 42, 48]. It does not mean that the Boolean functions studied in this chapter are useless for applications. They can be incorporated into heuristic methods.

Secondly, when data sets include numerical or continuous attribute values, the approach of this chapter does not work well, because the order of values or the degree of difference between values are not considered (except for criteria in DRSM). There are two approaches to overcome the drawback. One is discretization [7, 15] where the domain of a numerical attribute is partitioned to lower number of values. After discretization, we can apply attribute reduction to the data set without modification. The other is to use a similarity relation [12, 28] instead of the indiscernibility relation or a fuzzy partition [12, 22, 27] instead of the equivalence classes and define extensions of RSM. In that case, we can define structure-based reducts for the extended RSMs in the same way as those of this chapter.

Thirdly, reducts could suffer from overfitting because of rigid definitions of their preserving conditions. One technique to avoid overfitting is dynamic reducts [1], where decision tables with object subsets of a given cardinality are randomly and repeatedly selected, and reducts which appear in more decision tables than a given threshold are chosen as dynamic reducts.

In this chapter, we did not discuss algorithms to compute reducts and numerical experiments, whereas they are found in [5, 6, 8, 11, 13, 17, 19, 34, 41, 50, 51]. The references show how to select a desirable reduct or find an optimal reduct, and how to use the selected reduct for building classifiers. Additionally, they also show experimental results for benchmark or real-world data sets. The references do not include some types of reducts of this chapter, especially most types of reducts in VPRSM, however, from their results we hope that the proposed reducts would be useful in applications.

Proofs of theoretical results of this chapter are not so difficult. Parts of proofs are found in our papers [20, 23, 25, 31].

## References

1. Bazan, J.G., Nguyen, H.S., Nguyen, S.H., Synak, P., Wróblewski, J.: Rough set algorithms in classification problem. In: Polkowski, L., Tsumoto, S., Lin, T.Y. (eds.) *Rough Set Methods and Applications*, pp. 49–88. Physica-Verlag, New York (2000)
2. Ben-David, A.: Monotonicity maintenance in information-theoretic machine learning algorithms. *Mach. Learn.* **19**, 29–43 (1995)
3. Ben-David, A., Sterling, L., Pao, Y.H.: Learning and classification of monotonic ordinal concepts. *Comput. Intell.* **5**(1), 45–49 (1989)

4. Beynon, M.: Reducts within the variable precision rough sets model: a further investigation. *Eur. J. Oper. Res.* **134**(3), 592–605 (2001)
5. Beynon, M.J., Peel, M.J.: Variable precision rough set theory and data discretisation: an application to corporate failure prediction. *Omega* **29**, 561–576 (2001)
6. Chen, D., Hu, Q., Yang, Y.: Parameterized attribute reduction with Gaussian kernel based fuzzy rough sets. *Inf. Sci.* **181**, 5169–5179 (2011)
7. Chmielewski, M.R., Grzymala-Busse, J.W.: Global discretization of continuous attributes as preprocessing for machine learning. *Int. J. Approx. Reason.* **15**, 319–331 (1996)
8. Cornelis, C., Jensen, R., Hurtado, G., Ślęzak, D.: Attribute selection with fuzzy decision reducts. *Inf. Sci.* **180**, 209–224 (2010)
9. Crama, Y., Hammer, P.L.: Boolean Functions: Theory, Algorithms, and Applications. Cambridge University Press, New York (2011)
10. Dembczyński, K., Greco, S., Kotłowski, W., Słowiński, R.: Quality of rough approximation in multi-criteria classification problems. In: Greco, S., Hata, Y., Hirano, S., Inuiguchi, M., Miyamoto, S., Nguyen, H.S., Słowiński, R. (eds.) 5th International Conference on Rough Sets and Current Trends in Computing, RSCTC 2006. LNCS (LNAI), vol. 4259, pp. 318–327. Springer, Heidelberg (2006)
11. Dimitras, A.I., Slowinski, R., Susmaga, R., Zopounidis, C.: Business failure prediction using rough sets. *Eur. J. Oper. Res.* **114**, 263–280 (1999)
12. Dubois, D., Prade, H.: Rough fuzzy sets and fuzzy rough sets. *Int. J. Gen. Syst.* **17**(2–3), 191–209 (1990)
13. Düntsch, I., Gediga, G.: Uncertainty measures of rough set prediction. *Artif. Intell.* **106**, 109–137 (1998)
14. Eiter, T., Makino, K., Gottlob, G.: Computational aspects of monotone dualization: a brief survey. *Discret. Appl. Math.* **156**, 2035–2049 (2011)
15. Fayyad, U.M., Irani, K.B.: On the handling of continuous-valued attributes in decision tree generation. *Mach. Learn.* **8**(1), 87–102 (1992)
16. Greco, S., Matarazzo, B., Slowinski, R.: Rough set theory for multicriteria decision analysis. *Eur. J. Oper. Res.* **129**(1), 1–47 (2001)
17. Greco, S., Matarazzo, B., Slowinski, R.: Multicriteria classification by dominance-based rough set approach. In: Kloesgen, W., Zytkow, J.M. (eds.) *Handbook of Data Mining and Knowledge Discovery*. Oxford University Press, New York (2002)
18. Greco, S., Matarazzo, B., Słowiński, R.: Decision rule approach. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) *Multiple Criteria Decision Analysis: State of the Surveys*, pp. 507–561. Springer, New York (2005)
19. Hassanien, A.E.: Rough set approach for attribute reduction and rule generation: a case of patients with suspected breast cancer. *J. Am. Soc. Inf. Sci. Technol.* **55**(11), 954–962 (2004)
20. Inuiguchi, M.: Attribute reduction in variable precision rough set model. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.* **14**(4), 461–479 (2006)
21. Inuiguchi, M.: Structure-based attribute reduction in variable precision rough set models. *J. Adv. Comput. Intell. Intell. Inf.* **10**(5), 657–665 (2006)
22. Inuiguchi, M., Tanino, T.: New fuzzy rough sets based on certainty qualification. In: Pal, S.K., Polkowski, L., Skowron, A. (eds.) *Rough-Neural Computing: Techniques for Computing with Words*, pp. 277–296. Springer, Berlin (2004)
23. Inuiguchi, M., Tsurumi, M.: Measures based on upper approximations of rough sets for analysis of attribute importance and interaction. *Int. J. Innov. Comput. Inf. Control* **2**(1), 1–12 (2006)
24. Inuiguchi, M., Matsumoto, Y.: Refinement of attribute reduction in the classical rough sets toward decision analysis. In: *International Workshop on Soft Computing for Knowledge Technology* (2008)
25. Inuiguchi, M., Yoshioka, Y.: Several reducts in dominance-based rough set approach. In: Huynh, V.N., Nakamori, Y., Ono, H., Lawry, J., Kreinovich, V., Nguyen, H.T. (eds.) *Interval/Probabilistic Uncertainty and Non-classical Logics. Advances in Soft Computing*, vol. 46, pp. 163–175. Springer, Berlin (2008)

26. Inuiguchi, M., Yoshioka, Y., Kusunoki, Y.: Variable-precision dominance-based rough set approach and attribute reduction. *Int. J. Approx. Reason.* **50**(8), 1199–1214 (2009)
27. Jensen, R., Shen, Q.: Semantics-preserving dimensionality reduction: rough and fuzzy-rough-based approaches. *IEEE Trans. Knowl. Data Eng.* **16**(12), 1457–1471 (2004)
28. Jensen, R., Tuson, A., Shen, Q.: Finding rough and fuzzy-rough set reducts with SAT. *Inf. Sci.* **255**, 100–120 (2014)
29. Kryszkiewicz, M.: Rough set approach to incomplete information systems. *Inf. Sci.* **112**, 39–49 (1998)
30. Kryszkiewicz, M.: Comparative study of alternative types of knowledge reduction in inconsistent systems. *Int. J. Intell. Syst.* **16**, 105–120 (2001)
31. Kusunoki, Y., Inuiguchi, M.: A unified approach to reducts in dominance-based rough set approach. *Soft Comput.* **14**, 507–515 (2010)
32. Lievens, S., Baets, B.D., Cao-Van, K.: A probabilistic framework for the design of instance-based supervised ranking algorithms in an ordinal setting. *Ann. Oper. Res.* **163**, 115–142 (2008)
33. Mi, J., Wu, W., Zhang, W.: Approaches to knowledge reduction based on variable precision rough set model. *Inf. Sci.* **159**, 255–272 (2004)
34. Nguyen, L.G., Nguyen, H.S.: On elimination of redundant attributes in decision tables. In: Proceedings of the Federated Conference on Computer Science and Information Systems, pp. 317–322 (2012)
35. Pawlak, Z.: Rough sets. *Int. J. Inf. Comput. Sci.* **11**(5), 341–356 (1982)
36. Pawlak, Z.: *Rough Sets: Theoretical Aspects of Reasoning about Data*. Kluwer Academic Publishers, Dordrecht (1991)
37. Pawlak, Z., Skowron, A.: Rough sets and boolean reasoning. *Inf. Sci.* **177**, 41–73 (2007)
38. Pawlak, Z., Skowron, A.: Rough sets: some extensions. *Inf. Sci.* **177**, 28–40 (2007)
39. Pawlak, Z., Skowron, A.: Rudiments of rough sets. *Inf. Sci.* **177**, 3–27 (2007)
40. Pawlak, Z., Słowiński, R.: Rough set approach to multi-attribute decision analysis. *Eur. J. Oper. Res.* **72**, 443–459 (1994)
41. Sawicki, P., Źak, J.: Technical diagnostic of a fleet of vehicles using rough set theory. *Eur. J. Oper. Res.* **193**, 891–903 (2009)
42. Shen, Q., Jensen, R.: Rough sets, their extensions and applications. *Int. J. Autom. Comput.* **4**(3), 217–228 (2007)
43. Skowron, A., Rauszer, C.: The discernibility matrix and function in information systems. In: Słowiński, R. (ed.) *Intelligent Decision Support: Handbook of Application and Advances of Rough Set Theory*, pp. 331–362. Kluwer Academic Publishers, Dordrecht (1992)
44. Skowron, A., Stepaniuk, J.: Tolerance approximation spaces. *Fundam. Inform.* **27**(2–3), 245–253 (1996)
45. Ślęzak, D.: Various approaches to reasoning with frequency based decision reducts: a survey. In: Polkowski, L., Tsumoto, S., Lin, T.Y. (eds.) *Rough Set Methods and Applications*, pp. 235–285. Physica-Verlag, New York (2000)
46. Ślęzak, D.: Approximate entropy reducts. *Fundam. Inform.* **53**, 365–390 (2002)
47. Ślęzak, D., Ziarko, W.: The investigation of the Bayesian rough set model. *Int. J. Approx. Reason.* **40**, 81–91 (2005)
48. Ślęzak, D., Janusz, A.: Ensembles of bireducts: towards robust classification and simple representation. In: Kim, T.H., Adeli, H., Slezak, D., Sandnes, F.E., Song, X., Chung, K.I., Arnett, K.P. (eds.) *Future Generation Information Technology: Third International Conference, FGIT 2011. LNCS*, vol. 7105, pp. 64–77. Springer, Berlin (2011)
49. Susmaga, R., Słowiński, R., Greco, S., Matarazzo, B.: Generation of reducts and rules in multi-attribute and multi-criteria classification. *Control Cybern.* **29**(4), 969–988 (2000)
50. Swiniarski, R.W., Skowron, A.: Rough set methods in feature selection and recognition. *Pattern Recognit. Lett.* **24**, 833–849 (2003)
51. Wróblewski, J.: Ensembles of classifiers based on approximate reducts. *Fundam. Inform.* **47**, 351–360 (2001)
52. Yang, X., Yang, J., Wu, C., Yu, D.: Dominance-based rough set approach and knowledge reductions in incomplete ordered information system. *Inf. Sci.* **178**, 1219–1234 (2008)
53. Ziarko, W.: Variable precision rough set model. *J. Comput. Syst. Sci.* **46**(1), 39–59 (1993)
54. Ziarko, W.: Probabilistic approach to rough sets. *Int. J. Approx. Reason.* **49**, 272–284 (2008)

## **Part III**

# **Rule Discovery and Evaluation**

# Chapter 8

## A Comparison of Rule Induction Using Feature Selection and the LEM2 Algorithm

Jerzy W. Grzymała-Busse

**Abstract** The main objective of this chapter is to compare a strategy of rule induction based on feature selection, exemplified by the LEM1 algorithm, with another strategy, not using feature selection, exemplified by the LEM2 algorithm. The LEM2 algorithm uses all possible attribute-value pairs as the search space. It is shown that LEM2 significantly outperforms LEM1, a strategy based on feature selection in terms of an error rate (5 % significance level, two-tailed test). At the same time, the LEM2 algorithm induces smaller rule sets with the smaller total number of conditions as well. The time complexity for both algorithms is the same.

**Keywords** Rough set theory · Feature selection · LERS data mining system · LEM1 and LEM2 rule induction algorithms

### 8.1 Introduction

In 1982 an approach to feature selection, under the name of attribute reduction, using rough set theory, was introduced in [26], see also [27, 28]. In the rough set community reducing the original attribute set of attributes is one of the main and frequently used techniques.

Feature selection is the process of selecting a subset of relevant features. Research on feature selection, see, e.g., [2, 6, 20–23, 29, 31], includes finding the smallest set of features, improving this way the efficiency of data processing. Data are presented in tables, with rows labeled as cases (examples or entries) and columns labeled as features (variables or attributes).

---

J.W. Grzymała-Busse (✉)

Department of Electrical Engineering and Computer Science, University of Kansas,  
Lawrence 66045, USA  
e-mail: jerzy@ku.edu

J.W. Grzymała-Busse

Department of Expert Systems and Artificial Intelligence, University of Information  
Technology and Management, 35-225 Rzeszów, Poland

An introduction to feature selection is presented in [17]. Recently two books were published [18, 24], summarizing research in this area. There is active research on feature selection in statistics, data mining, and soft computing.

The main objective of this chapter is to compare, in terms of an error rate, rule complexity, and time complexity of two approaches: an approach to rule induction based on feature selection with another approach to rule induction, based on the LEM2 algorithm, without any feature selection. In the former approach computations are conducted on the entire attributes, so it is also called *global* [14]. To be more specific, for every attribute a corresponding partition on the set of all cases, implied by the indiscernibility relation [26–28] is computed and feature selection is conducted by computation on such partitions. On the other hand, the LEM2 algorithm works on attribute values, instead on entire attributes, so it is called *local* [14]. The search space of the LEM2 algorithm is the set of all blocks of attribute-value pairs. A block of an attribute-value pair  $(a, v)$  is the set of all cases with the value of  $a$  equal to  $v$ .

A preliminary version of this chapter was presented at IPMU 2012, the 14th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Catania, Italy, July 9–13, 2012 [15] (Table 8.1).

**Table 8.1** Acronyms and symbols used in the chapter and their meaning

Acronym or symbol	Meaning
$A$	Set of all attributes
$\text{appr}(X)$	Lower approximation of $X$
$\overline{\text{appr}}(X)$	Upper approximation of $X$
$B$	Subset of the set $A$ of all attributes
$B^*$	Partition on $U$ defined by $B$
$C$	Concept of the data sets
$d$	Decision
$\{d\}^*$	Partition on $U$ , the set of all concepts
$G$	Goal of the LEM2
$IND(B)$	Indiscernibility relation of $B$
$LEM1$	Learning from Examples Module version 1
$LEM2$	Learning from Examples Module version 2
$LERS$	Learning from Examples based on Rough Sets data mining system
$t$	Attribute-value pair $(a, v)$
$T$	Complex, i.e., a set of attribute-value pairs
$T(G)$	Set of attribute-value pairs relevant with $G$
$\mathcal{T}$	Local covering
$U$	Universe, the set of all cases of data set
$x$	Element of $U$
$X$	Subset of $U$
$ X $	Cardinality of the set $X$
$y$	Element of $U$

## 8.2 Rule Induction Based on Feature Selection

We will discuss rule induction which belongs to *supervised learning*, i.e., we will assume that all cases are preclassified by an expert. In the data set one of variables is called a *decision* and the decision value is assigned by an expert to each case. A very simple example of such a table is presented as Table 8.2, in which attributes are: *Temperature*, *Headache*, *Nausea*, and the decision is *Flu*. The set of all cases labeled by the same decision value is called a *concept*. For Table 8.2, case set {1, 2} is a concept of all cases affected by flu (for each case from this set the corresponding value of *Flu* is yes).

Note that in Table 8.2 the attribute *Nausea* is redundant (irrelevant). Remaining two attributes (*Temperature* and *Headache*) distinguish all six cases. Let us make it more precise using fundamental definitions of rough set theory [26–28]. Let  $B$  be a nonempty subset of the set  $A$  of all attributes. Let  $U$  denote the set of all cases. The indiscernibility relation  $IND(B)$  is a relation on  $U$  defined for  $x, y \in U$  by  $(x, y) \in IND(B)$  if and only if for both  $x$  and  $y$  the values for all attributes from  $B$  are identical.

The indiscernibility relation  $IND(B)$  is an equivalence relation. Equivalence classes of  $IND(B)$  are called *elementary sets* of  $B$ . Any union of elementary sets of  $B$  is called a *definable set* in  $B$ . For Table 8.2, and  $B = \{\text{Temperature}, \text{Headache}\}$ , elementary sets of  $IND(B)$  are {1}, {2}, {3}, {4}, {5}, and {6}.

The family of all  $B$ -elementary sets is a partition on  $U$ . This set will be denoted by  $B^*$ , for example, in Table 8.2,

$$\{\text{Temperature}, \text{Headache}\}^* = \{\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}\}.$$

For a decision  $d$  we say that  $\{d\}$  depends on  $B$  if and only if  $B^* \leq \{d\}^*$ , i.e., for any elementary set  $X$  in  $B$  there exists a concept  $C$  from  $\{d\}^*$  such that  $X \subseteq C$ . A *global covering* (or *relative reduct*) of  $\{d\}$  is a subset  $B$  of  $A$  such that  $\{d\}$  depends on  $B$  and  $B$  is minimal in  $A$ . The algorithm to compute a single global covering is presented below.

**Table 8.2** A consistent data set

Case	Attributes			Decision
	Temperature	Headache	Nausea	
1	High	Yes	No	Yes
2	Very_high	No	No	Yes
3	Very_high	Yes	No	No
4	Normal	No	No	No
5	High	No	Yes	Maybe
6	Normal	Yes	Yes	Maybe

### LEM1 algorithm for computing a single global covering

(**input:** the set  $A$  of all attributes, partition  $\{d\}^*$  on  $U$ ;

**output:** a single global covering  $R$ );

**begin**

compute partition  $A^*$ ;

$P := A$ ;

$R := \emptyset$ ;

**if**  $A^* \leq \{d\}^*$

**then**

**begin**

**for** each attribute  $a$  in  $A$  **do**

**begin**

$Q := P - \{a\}$ ;

compute partition  $Q^*$ ;

**if**  $Q^* \leq \{d\}^*$  **then**  $P := Q$

**end** {for}

$R := P$

**end** {then}

**end** {algorithm}.

The time complexity of the algorithm for computing a single global covering is polynomial. For a set  $X$ ,  $|X|$  denotes the cardinality of  $X$ . Let  $m$  be the number of all cases, i.e.,  $|U| = m$ , and  $n$  be the number of all attributes, i.e.,  $|A| = n$ . The time complexity of the algorithm, using a “brute force” approach, in the worst case scenario, and assuming symbolic attributes, is  $O(mn^2)$ . The time complexity of the algorithm for the attributes with the number of values depending on  $m$  is  $O(m^2n^2)$ .

For the data set from Table 8.2, the global covering is  $\{\text{Temperature}, \text{Headache}\}$ . The above algorithm is implemented as LEM1 (Learning from Examples Module version 1). It is a component of the data mining system LERS (Learning from Examples using Rough Sets). Another, similar approach to rule induction based on feature selection was presented in [1].

The rule set, induced by LEM1 from the global covering  $\{\text{Temperature}, \text{Headache}\}$  for the concept  $(\text{Flu, maybe})$ , is:

$(\text{Temperature, high}) \text{ and } (\text{Headache, no}) \rightarrow (\text{Flu, maybe}),$

$(\text{Temperature, normal}) \text{ and } (\text{Headache, yes}) \rightarrow (\text{Flu, maybe}).$

## 8.3 LEM2

An idea of blocks of attribute-value pairs is used in the LEM2 rule induction algorithm (Learning from Examples Module, version 2), another component of LERS. LEM2 explores the search space of attribute-value pairs. We will quote a few definitions to describe the LEM2 algorithm [4, 11, 12, 14].

For an attribute-value pair  $(a, v)$ , a *block* of  $(a, v)$ , denoted by  $[(a, v)]$ , is the following set

$$\{x \mid x \in U, a(x) = v\},$$

where  $a(x)$  is the value of the attribute  $a$  for the case  $x$ . Let  $B$  be a nonempty lower or upper approximation of a concept represented by a decision-value pair  $(d, w)$ . Let  $T$  be a *complex* of  $B$ , i.e., the set of attribute-value pairs  $t = (a, v)$  with  $a \in B$ . A block of  $T$ , denoted by  $[T]$ , is the following set

$$\cap\{[t] \mid t \in T\}.$$

Set  $B$  depends on a set  $T$  of attribute-value pairs if and only if

$$\emptyset \neq [T] = \bigcap_{t \in T} [t] \subseteq B.$$

Set  $T$  is a *minimal complex* of  $B$  if and only if  $B$  depends on  $T$  and no proper subset  $T'$  of  $T$  exists such that  $B$  depends on  $T'$ . Let  $\mathcal{T}$  be a nonempty collection of nonempty sets of attribute-value pairs. Then  $\mathcal{T}$  is a *local covering* of  $B$  if and only if the following conditions are satisfied:

1. each member  $T$  of  $\mathcal{T}$  is a minimal complex of  $B$ ,
2.  $\cup\{[T] \mid T \in \mathcal{T}\} = B$ , and

$\mathcal{T}$  is minimal, i.e.,  $\mathcal{T}$  has the smallest possible number of elements.

The LEM2 algorithm is presented below.

#### LEM2 algorithm for computing a single local covering

(**input**: a set  $B$ ,

**output**: a single local covering  $\mathcal{T}$  of set  $B$ );

**begin**

$G := B$ ;

$\mathcal{T} := \emptyset$ ;

**while**  $G \neq \emptyset$

**begin**

$T := \emptyset$ ;

$T(G) := \{t \mid [t] \cap G \neq \emptyset\}$ ;

**while**  $T = \emptyset$  **or**  $[T] \not\subseteq B$

**begin**

                    select a pair  $t \in T(G)$  such that  $|[t] \cap G|$  is

                    maximum; if a tie occurs, select a pair  $t \in T(G)$  with the smallest cardinality of  $[t]$ ;

                    if another tie occurs, select first pair;

$T := T \cup \{t\}$ ;

$G := [t] \cap G$ ;

```

 $T(G) := \{t | [t] \cap G \neq \emptyset\};$ 
 $T(G) := T(G) - T;$ 
end {while}
for each  $t \in T$  do
    if  $[T - \{t\}] \subseteq B$  then  $T := T - \{t\}$ ;
     $\mathcal{T} := \mathcal{T} \cup \{T\}$ ;
     $G := B - \bigcup_{T \in \mathcal{T}} [T]$ ;
end {while};
for each  $T \in \mathcal{T}$  do
    if  $\bigcup_{S \in \mathcal{T} - \{T\}} [S] = B$  then  $\mathcal{T} := \mathcal{T} - \{T\}$ ;
end {algorithm}.

```

With the same assumptions as for the algorithm for computing a single global covering, the time complexity of the algorithm for computing a single local covering is also  $O(mn^2)$  for symbolic attributes and  $O(m^2n^2)$  for attributes with the number of values depending on  $m$ .

The LERS data mining system also includes the LEM2 algorithm.

The first step of the algorithm LEM2 is to compute all attribute-value pair blocks. For Table 8.2, these blocks are

```

 $[(Temperature, very\_high)] = \{2, 3\},$ 
 $[(Temperature, high)] = \{1, 5\},$ 
 $[(Temperature, normal)] = \{4, 6\},$ 
 $[(Headache, yes)] = \{1, 3, 6\},$ 
 $[(Headache, no)] = \{2, 4, 5\},$ 
 $[(Nausea, no)] = \{1, 1, 3, 4\},$ 
 $[(Nausea, yes)] = \{5, 6\}.$ 

```

Let us induce rules for the concept  $\{5, 6\}$ . It is immediately clear that  $\{(Nausea, yes)\}$  is the only required minimal complex and that the local covering consists of only this single minimal complex.

The corresponding rule set, induced by LEM2, contains just one rule  
 $(Nausea, yes) \rightarrow (Flu, maybe)$ .

Obviously, in general, rule sets induced by LEM2 are simpler than rule sets induced by LEM1 from the same data sets. This observation is confirmed by our experiments.

## 8.4 Inconsistent Data

An example of the inconsistent data set is presented in Table 8.3. In inconsistent data some cases may conflict with each other. Conflicting cases have the same attribute values yet different decision values. In Table 8.3 cases 1 and 7 are conflicting. A *level of consistency* is the ratio of the cardinality of the set of all cases not involved in any conflict to the cardinality of  $U$ . For the data set from Table 8.3, the *level of consistency* is  $\frac{5}{7} = 71.4\%$ .

**Table 8.3** An inconsistent data set

Case	Attributes			Decision
	Temperature	Headache	Nausea	
1	High	Yes	No	Yes
2	Very_high	No	No	Yes
3	Very_high	Yes	No	No
4	Normal	No	No	No
5	High	No	Yes	Maybe
6	Normal	Yes	Yes	Maybe
7	High	Yes	No	No

The LERS data mining system uses rough set approach to inconsistent data, i.e., it computes lower and upper approximations for all concepts before applying LEM1 or LEM2 algorithm. Let  $X$  be a concept. In general,  $X$  is not definable in  $A$ . However,  $X$  may be approximated by two definable sets in  $A$ , the first one is called a *lower approximation* of  $X$ , denoted by  $\underline{appr}(X)$  and defined as follows

$$\{\{x\} \mid x \in U, \{x\} \subseteq X\}.$$

The second set is called an *upper approximation* of  $X$ , denoted by  $\overline{appr}(X)$  and defined as follows

$$\cup \{\{x\} \mid x \in U, \{x\} \cap X \neq \emptyset\}.$$

For example, for the concept  $\{(Flu, yes)\} = \{1, 2\}$ ,

$$\underline{appr}(\{1, 2\}) = \{2\},$$

and

$$\overline{appr}(\{1, 2\}) = \{1, 2, 7\}.$$

Rules induced from lower approximations are called *certain*, rules induced from upper approximations are called *possible*.

Note that even though the data set from Table 8.3 is inconsistent, the attribute *Nausea* is still redundant (irrelevant), since

$$\begin{aligned} \{Temperature, Headache\}^* &= \{Temperature, Headache, Nausea\}^* \\ &= \{\{1, 7\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}\}. \end{aligned}$$

The LERS system computes, for every concept, a pair of data sets, based on lower and upper approximations to induce certain and possible rule sets, respectively. For example, for the concept  $\{1, 2\}$ , certain rule sets are induced from the data set presented in Table 8.4 and possible rule sets from Table 8.5.

Obviously, the final rule set, certain or possible, is a union of rule sets induced for all concepts, from data sets based on lower or upper approximations, respectively, with all rules for SPECIAL values removed.

**Table 8.4** A data set based on the lower approximation  $\{2\}$  of the concept  $\{1, 2\}$

Case	Attributes			Decision
	Temperature	Headache	Nausea	
1	High	Yes	No	SPECIAL
2	Very_high	No	No	Yes
3	Very_high	Yes	No	SPECIAL
4	Normal	No	No	SPECIAL
5	High	No	Yes	SPECIAL
6	Normal	Yes	Yes	SPECIAL
7	High	Yes	No	SPECIAL

**Table 8.5** A data set based on the upper approximation  $\{1, 2, 7\}$  of the concept  $\{1, 2\}$

Case	Attributes			Decision
	Temperature	Headache	Nausea	
1	High	Yes	No	Yes
2	Very_high	No	No	Yes
3	Very_high	Yes	No	SPECIAL
4	Normal	No	No	SPECIAL
5	High	No	Yes	SPECIAL
6	Normal	Yes	Yes	SPECIAL
7	High	Yes	No	Yes

Thus, if we are going to use the strategy of rule induction based on feature selection, possible rules induced from Table 8.3 are:

- $(Temperature, \text{high}) \text{ and } (Headache, \text{yes}) \rightarrow (Flu, \text{yes}),$
- $(Temperature, \text{very\_high}) \text{ and } (Headache, \text{no}) \rightarrow (Flu, \text{yes}),$
- $(Temperature, \text{high}) \text{ and } (Headache, \text{yes}) \rightarrow (Flu, \text{no}),$
- $(Temperature, \text{very\_high}) \text{ and } (Headache, \text{yes}) \rightarrow (Flu, \text{no}),$
- $(Temperature, \text{normal}) \text{ and } (Headache, \text{no}) \rightarrow (Flu, \text{no}),$
- $(Temperature, \text{high}) \text{ and } (Headache, \text{no}) \rightarrow (Flu, \text{maybe}),$
- $(Temperature, \text{normal}) \text{ and } (Headache, \text{yes}) \rightarrow (Flu, \text{maybe}).$

At the same time, the LEM2 algorithm will induce the flowing set of possible rules from the same data set:

- $(Nausea, \text{no}) \text{ and } (Temperature, \text{high}) \rightarrow (Flu, \text{yes}),$
- $(Temperature, \text{very\_high}) \text{ and } (Headache, \text{no}) \rightarrow (Flu, \text{yes}),$
- $(Nausea, \text{no}) \text{ and } (Headache, \text{yes}) \rightarrow (Flu, \text{no}),$
- $(Temperature, \text{normal}) \text{ and } (Headache, \text{no}) \rightarrow (Flu, \text{no}),$
- $(Nausea, \text{yes}) \rightarrow (Flu, \text{maybe}).$

Again, it is quite clear that LEM2 produces simpler rules (and fewer rules).

## 8.5 LERS Classification System

There is a few existing classification systems, e.g., associated with rule induction systems LERS or AQ [25]. A classification system used in LERS is a modification of the well-known bucket brigade algorithm [3, 19, 30]. In the LERS classification system the decision to which concept a case belongs is made on the basis of three factors: *strength*, *specificity*, and *support*. These factors are defined as follows: *strength* is the total number of cases correctly classified by the rule during training. *Specificity* is the total number of attribute-value pairs on the left-hand side of the rule. The matching rules with a larger number of attribute-value pairs are considered more specific. The third factor, *support*, is defined as the sum of products of strength and specificity for all matching rules indicating the same concept. The concept  $C$  for which the support, i.e., the following expression

$$\sum_{\text{matching rules } r \text{ describing } C} \text{Strength}(r) * \text{Specificity}(r)$$

is the largest is the winner and the case is classified as being a member of  $C$ .

In the classification system of LERS, if complete matching is impossible, all partially matching rules are identified. These are rules with at least one attribute-value pair matching the corresponding attribute-value pair of a case. For any partially matching rule  $r$ , the additional factor, called *Matching\_factor*( $r$ ), is computed. *Matching\_factor*( $r$ ) is defined as the ratio of the number of matched attribute-value pairs of  $r$  with a case to the total number of attribute-value pairs of  $r$ . In partial matching, the concept  $C$  for which the following expression is the largest

$$\sum_{\substack{\text{partially matching} \\ \text{rules } r \text{ describing } C}} \text{Matching\_factor}(r) * \text{Strength}(r) * \text{Specificity}(r)$$

is the winner and the case is classified as being a member of  $C$ .

## 8.6 Experiments

In our experiments we used 14 data sets that are available on the Machine Learning Repository at the University of California at Irvine, see Table 8.6. Some of these data sets were incomplete (*Breast Cancer-Slovenia*, *Soybean*, *Postoperative Patient* and *Primary Tumor*).

For incomplete data sets missing attribute values were replaced by specified attribute values using an imputation method called the *most common value of an attribute restricted to a concept* [16].

**Table 8.6** Data sets used for experiments

Data set	Number of			Consistency (%)
	Cases	Attributes	Concepts	
Australian Credit Approval	690	14	2	100
Breast Cancer—Slovenia	286	9	2	95.45
Breast Cancer—Wisconsin	625	9	9	94.08
Bupa Liver Disorders	345	6	2	100
Glass	214	9	6	100
Hepatitis	155	19	2	100
Image segmentation	210	19	7	100
Iris	150	4	3	100
Lymphography	148	18	4	100
Pima	768	8	2	100
Postoperative patients	90	8	3	84.44
Soybean	307	35	19	100
Primary Tumor	339	17	21	76.40
Wine Recognition	178	13	3	100

Let us say that attribute  $a$  has missing attribute value for case  $x$  from concept  $C$  and that the value of  $a$  for  $x$  is missing. This missing attribute value is exchanged by the known attribute value for which the conditional probability of  $a$  for case  $x$  given  $C$  is the largest.

Some of these data sets had numerical attributes (*Australian Credit Approval*, *Bupa Liver Disorders*, *Primary Tumor* and *Wine Recognition*). Numerical attributes were discretized using cluster analysis methods of discretization [5].

The data mining system LERS uses for discretization a number of discretization algorithms [13]. In our experiments we used two approaches to discretization based on cluster analysis. First, all numerical attributes were normalized [7] (attribute values were divided by the attribute standard deviation).

In our first discretization technique, based on agglomerative cluster analysis [7], initially each case is a single cluster, then clusters are fused together, forming larger and larger clusters. In remaining four cluster analysis discretization methods, where we used divisive techniques, initially all cases are grouped in one cluster, then this cluster is gradually divided into smaller and smaller clusters. In both methods, during the first step of discretization, *cluster formation*, cases that exhibit the most similarity are fused into clusters.

Once clusters are formed the postprocessing starts. Initially clusters are projected on all attributes. Then the resulting intervals are merged to reduce the number of intervals and, at the same time, preserving consistency. Merging of intervals begins from *safe merging*, where, for each attribute, neighboring intervals labeled by the same decision value are replaced by their union. The next step of merging intervals is based on checking every pair of neighboring intervals whether their merging will result in preserving consistency. If so, intervals are merged permanently. If not, they are marked as un-mergeable.

Thus, all data sets used for experiments were complete and symbolic.

Our experiments were conducted on a machine with 34 GB of RAM with Intel(R) Xeon Processor X5650 (12 MB cache, 2.66 GHz, 6 Cores) under Fedora 17 Linux operating system.

In our experiments, for any data set, for both algorithms, LEM1 and LEM2, the same ten pairs of training and testing data sets were used during ten-fold cross validation. Hence, for any fold, the same training data sets were used for induction and the same testing data sets were used for computing an error rate. Additionally, the same LERS classification method was used for computing errors. The only difference was in different strategies of rule induction used in LEM1 and LEM2. Additionally, for both algorithms, we used only *certain* rule sets for inconsistent data sets. Results of our experiments are presented in Tables 8.7, 8.8 and 8.9.

**Table 8.7** Results of experiments—an error rate

Data set	LEM1 (%)	LEM2 (%)
Australian Credit Approval	21.74	16.67
Breast Cancer—Slovenia	36.71	34.62
Breast Cancer—Wisconsin	19.68	22.24
Bupa Liver Disorders	36.52	36.81
Glass	33.18	31.31
Hepatitis	21.94	16.77
Image segmentation	17.62	18.10
Iris	3.33	4.67
Lymphography	31.08	18.24
Pima	33.46	30.73
Postoperative patients	41.11	35.36
Soybean	23.45	14.98
Primary tTumor	60.18	62.64
Wine Recognition	8.43	5.06

**Table 8.8** Results of experiments—rule set complexity

Data set	LEM1		LEM2	
	Number of			
	Rules	Conditions	Rules	Conditions
Australian Credit Approval	317	1,380	115	559
Breast Cancer—Slovenia	160	518	92	319
Breast Cancer—Wisconsin	255	638	164	421
Bupa Liver Disorders	234	684	164	487
Glass	111	334	82	262
Hepatitis	47	191	21	88
Image segmentation	71	273	38	144
Iris	18	43	13	30
Lymphography	77	217	26	74
Pima	444	1,547	287	1,025
Postoperative patients	50	175	28	98
Soybean	101	411	42	139
Primary Tumor	171	829	125	638
Wine Recognition	32	111	16	52

**Table 8.9** Results of experiments—run time, in milliseconds

Data set	LEM1	LEM2
Australian Credit Approval	457	273
Breast Cancer—Slovenia	38	54
Breast Cancer—Wisconsin	146	126
Bupa Liver Disorders	30	69
Glass	31	30
Hepatitis	33	26
Image segmentation	65	39
Iris	3	1
Lymphography	36	27
Pima	257	407
Postoperative patients	7	5
Soybean	415	158
Primary Tumor	341	134
Wine Recognition	25	9

## 8.7 Conclusions

As follows from our experiments presented in Table 8.7, rule sets induced by the LEM2 algorithm outperform rule sets induced by using feature selection (the LEM1 algorithm) in terms of the error rate. The Wilcoxon matched-pairs signed-rank test indicates that the LEM2 algorithm is better with 5 % of significance level (two-tailed test).

Moreover, the LEM2 algorithm induces much simpler rule sets. As follows from Table 8.8, for all 14 data sets, rule sets induced by the LEM2 algorithm are smaller and the total number of conditions in these rule sets is smaller as well. Simpler rules are easier to interpret.

Both algorithms, LEM1 and LEM2, are of polynomial time complexity. It is confirmed by Table 8.9. The Wilcoxon matched-pairs signed-rank test indicates that there is no significant difference in run time between the two algorithms. LEM2 can induce rule sets from data sets with tens of thousands of attributes, such as microarray data sets, see, e.g., [8–10]. Therefore we may conclude that the LEM2 algorithm, with the space search of all attribute-value pairs, is better than LEM1 based on feature selection.

## References

1. Bazan, J.G., Szczuka, M.S., Wojna, A., Wojnarski, M.: On the evolution of rough set exploration system. In: Proceedings of the Rough Sets and Current Trends in Computing Conference, pp. 592–601 (2004)
2. Blum, A., Langley, P.: Selection of relevant features and examples in machine learning. *Artif. Intell.* **97**, 245–271 (1997)
3. Booker, L.B., Goldberg, D.E., F, H.J.: Classifier systems and genetic algorithms. In: Carbonell, J.G. (ed.) *Machine Learning: Paradigms and Methods*, pp. 235–282. MIT, Boston (1990)
4. Chan, C.C., Grzymala-Busse, J.W.: On the attribute redundancy and the learning programs ID3, PRISM, and LEM2. Technical Report, Department of Computer Science, University of Kansas (1991)
5. Chmielewski, M.R., Grzymala-Busse, J.W.: Global discretization of continuous attributes as preprocessing for machine learning. *Int. J. Approx. Reason.* **15**(4), 319–331 (1996)
6. Dash, M., Liu, H.: Feature selection for classification. *Intell. Data Anal.* **1**, 131–156 (1997)
7. Everitt, B.: *Cluster Analysis*. Heinemann Educational Books, London (1980)
8. Fang, J., Grzymala-Busse, J.: Leukemia prediction from gene expression data—a rough set approach. In: Proceedings of the Eighth International Conference on Artificial Intelligence and Soft Computing, pp. 899–908 (2006)
9. Fang, J., Grzymala-Busse, J.: Mining of microRNA expression data—a rough set approach. In: Proceedings of the First International Conference on Rough Sets and Knowledge Technology, pp. 758–765 (2006)
10. Fang, J., Grzymala-Busse, J.: Predicting penetration across the blood-brain barrier—a rough set approach. In: Proceedings of the IEEE International Conference on Granular Computing, pp. 231–236 (2007)
11. Grzymala-Busse, J.W.: LERS—a system for learning from examples based on rough sets. In: Slowinski, R. (ed.) *Intelligent Decision Support: Handbook of Applications and Advances of the Rough Set Theory*, pp. 3–18. Kluwer Academic Publishers, Dordrecht, Boston (1992)

12. Grzymała-Busse, J.W.: A new version of the rule induction system LERS. *Fundam. Inform.* **31**, 27–39 (1997)
13. Grzymała-Busse, J.W.: Mining numerical data—a rough set approach. In: Proceedings of the RSEISP'2007, the International Conference of Rough Sets and Emerging Intelligent Systems Paradigms, pp. 12–21 (2007)
14. Grzymała-Busse, J.W.: Rule induction. In: Maimon, O., Rokach, L. (eds.) *Data Mining and Knowledge Discovery Handbook*, 2nd edn, pp. 249–265. Springer, Berlin (2010)
15. Grzymała-Busse, J.W.: An empirical comparison of rule induction using feature selection with the LEM2 algorithm. In: Greco, S., Bouchon-Meunier, B.B., Coletti, G., Fedrizzi, M.M., Matarazzo, B., Yager, R.R. (eds.) *Communications in Computer and Information Science*, vol. 297, pp. 270–279. Springer (2012)
16. Grzymała-Busse, J.W., Grzymała-Busse, W.J.: Handling missing attribute values. In: Maimon, O., Rokach, L. (eds.) *Data Mining and Knowledge Discovery Handbook*, 2nd edn, pp. 33–51. Springer, Berlin (2010)
17. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. *J. Mach. Learn. Res.* **3**, 1157–1182 (2003)
18. Guyon, I., Gunn, S., Nikravesh, M., Zadeh, L.A.: *Feature Extraction. Foundations and Applications*. Springer, Berlin (2006)
19. Holland, J.H., Holyoak, K.J., Nisbett, R.E.: *Induction: Processes of Inference, Learning, and Discovery*. MIT, Boston (1986)
20. Jain, A., Zongker, D.: Feature selection: evaluation, application, and small sample performance. *IEEE Trans. Pattern Anal. Mach. Intell.* **19**, 153–158 (1997)
21. Kohavi, R., John, G.: Wrappers for feature selection. *Artif. Intell.* **97**, 273–324 (1997)
22. Lei, Y., Huan, L.: Feature selection for high-dimensional data: a fast correlation-based filter solution. In: Proceedings of the 20-th International Conference on Machine Learning, p. 8 (2003)
23. Liu, H., Yu, L.: Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. Knowl. Data Eng.* **17**, 491–502 (2005)
24. Liu, H., Motoda, H.: *Computational Methods of Feature Selection*. Chapman and Hall/CRC, Boca Raton (2007)
25. Michalski, R.S., Mozetic, I., Hong, J., Lavrac, N.: The multi-purpose incremental learning system AQ15 and its testing application on three medical domains. In: Proceedings of the National Conference on Artificial Intelligence, pp. 1041–1045. Morgan Kaufmann, San Mateo (1986)
26. Pawlak, Z.: Rough sets. *Int. J. Comput. Inf. Sci.* **11**, 341–356 (1982)
27. Pawlak, Z.: *Rough Sets: Theoretical Aspects of Reasoning About Data*. Kluwer Academic Publishers, Dordrecht (1991)
28. Pawlak, Z., Grzymała-Busse, J.W., Slowinski, R., Ziarko, W.: Rough sets. *Commun. ACM* **38**, 89–95 (1995)
29. Peng, H., Fuhui, L., Chris, D.: Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.* **27**, 1226–1238 (2005)
30. Stefanowski, J.: *Algorithms of Decision Rule Induction in Data Mining*. Poznan University of Technology Press, Poznan (2001)
31. Swiniarski, R.W., Skowron, A.: Rough set methods in feature selection and recognition. *Pattern Recognit. Lett.* **24**, 833–849 (2003)

# **Chapter 9**

## **Meta-actions as a Tool for Action Rules Evaluation**

**Hakim Touati, Zbigniew W. Raś and James Studnicki**

**Abstract** Action rules extraction is a field of data mining used to extract actionable patterns from large datasets. Action rules present users with a set of actionable tasks to follow to achieve a desired result. An action rule can be seen as two patterns of feature values (classification rules) occurring together and having the same features. Action rules are evaluated using their supporting patterns occurrence in a measure called support. They are also evaluated using their confidence defined as the product of the two patterns confidences. Those two measures are important to evaluate action rules; nonetheless, they fail to measure the feature values transition correlation and applicability. This is due to the core of the action rules extraction process that extracts independent patterns and constructs an action rule. In this chapter, we present the benefits of meta-actions in evaluating action rules in terms of two measures, namely likelihood and execution confidence. In fact, in meta-actions, we extract real feature values transition patterns, rather than composing two feature values patterns. We also present an evaluation model of the application of meta-actions based on cost and satisfaction. We extracted action rules and meta-actions and evaluated them on the Florida State Inpatient Databases that is a part of the Healthcare Cost and Utilization Project.

---

H. Touati (✉)

College of Computing and Informatics, The University of North Carolina at Charlotte,  
Charlotte, NC 28223, USA  
e-mail: htouati@uncc.edu

Z.W. Raś

The University of North Carolina at Charlotte, Charlotte, NC 28223, USA  
e-mail: ras@uncc.edu

Z.W. Raś

Warsaw University of Technology, 00-665 Warsaw, Poland  
e-mail: ras@ii.pw.edu.pl

J. Studnicki

College of Public Health, The University of North Carolina at Charlotte,  
Charlotte, NC 28223, USA  
e-mail: jstudnic@uncc.edu

**Keywords** Action rules · Rule evaluation · Evaluation measure · Rule support · Meta-actions

## 9.1 Introduction

Action rules are used in several industries such as banking and healthcare. They provide decision makers with a tool to build strategies to drive their business toward a more profitable outcome. Action rules were applied previously to a considerable number of research areas such as Music Information Retrieval (MIR) [5] and healthcare [15, 17]. There has been an increasing interest on action rule discovery algorithms since their creation by Raś and Wiecezorkowska [10]. Some of the new algorithms are Association Action Rules [9] and ARED [4] that extract action rules directly from the dataset, among other ones [6, 12, 14, 16].

In addition to action rules, a new concept called meta-actions allows deciders to extract the triggers that provoke the necessary transitions to execute action rules. In fact, meta-actions are the core triggers of action rules, and allow us to select the action rules that are more likely to take effect. Meta-actions is a relatively new concept that was defined in [14] and further explored in [7, 13].

In the healthcare research, action rules have been used to understand experts practices and improve patients care [9, 15, 17]. However, treatment patterns under the form of meta-actions were not applied previously to evaluate action rules. Meta-actions allow us to mine the diagnosis transitions caused by applying treatments. In this context, we can use meta-actions to evaluate action rules that model treatments. In this chapter, we propose the application of meta-actions as an evaluation tool for action rules. We strive to develop evaluation metrics for meta-actions and action rules.

The 2010 Florida State Inpatient Databases (SID) that is a part of the Healthcare Cost and Utilization Project (HCUP) [2] was used as a dataset to extract meta-actions and action rules. We propose several evaluation metrics such as the likelihood of execution of an action term, and the execution confidence of an action rule and applied them to the 2010 Florida SID. The main contributions of this chapter are summarized as follows:

- We presented meta-actions discovery methodology.
- We developed several evaluation metrics for action rules and meta-actions.
- We extracted meta-actions and action rules from the 2010 Florida SID, and evaluated them.

In what follows, we start by defining the terminology used in this chapter with regards to meta-actions and action rules. We then present the meta-actions extraction process and evaluation metrics. We compare briefly action rules and meta-actions and propose action rules evaluation metrics. Finally, we evaluate our approach and present our findings.

## 9.2 Decision Systems

Action rules and meta-action elementary rules are extracted from decision systems. A decision system is an information system, where features are partitioned into two groups: the first group is composed of classification features, and the second of one particular feature called the decision that models the outcome. An object instance is represented as a row, called transaction instance, in terms of a set of features.

More formally, by a decision system we mean  $S = (X, F \cup \{d\}, V)$ , where:

1.  $X$  is a set of objects instances,  $F$  is a set of classification features,  $d$  is a decision feature, and  $V$  is the domain of these feature values.
2.  $f : X \rightarrow V_f$  is a function for any feature  $f \in F$ , where  $V_f$  is called the domain of  $f$ .
3.  $d : X \rightarrow V_d$  is a function, where  $V_d$  is called the domain of  $d$ .
4.  $V = V_F \cup V_d$ , where  $V_F = \bigcup\{V_f : f \in F\}$ .

Also, for each  $x \in X$  and  $f \in F$ , we assume that value  $f(x) \in V_f$  is classified either as positive (normal) or negative (abnormal). To be more precise, we assume that  $F(x)$  denotes the set  $\{f(x) : f \in F\}$  which represents the state of the object instance  $x$ , and that  $F(x) = E_n(x) \cup E_p(x)$ , where  $E_p(x)$  is a set of positive values and  $E_n(x)$  is a set of negative values for  $x \in X$ . If  $f(x) \in E_n(x)$ , then the value  $f(x)$  is interpreted as abnormal (for instance: high temperature, cough, headache). If  $f(x) \in E_p(x)$ , then value  $f(x)$  is interpreted as normal.

## 9.3 Action-Rules

Action rules are rules that provide a set of actionable patterns to follow in order to transition the objects population from a certain state to a more profitable state with respect to the decision feature. They allow users to understand the correlations between transition patterns in the decision system, and construct actionable tasks that lead to a desirable outcome. Action rules are composed of a decision feature  $d$  and classification features that are in turn divided into two sets: stable features  $F_{st}$ , and flexible features  $F_{fl}$  such that  $F = F_{fl} \cup F_{st}$ . Stable features are object properties that we do not have control over in the context of our information system. For example, age and gender are stable features. Flexible features are object properties that can transition from one value to another value triggering a change in the object state. For instance, salary and benefits are flexible features since their values can change.

**Definition 1** (*Atomic action term in S*) also called elementary action term in  $S$ , is an expression that defines a change of state for a distinct feature in  $S$ . For example,  $(f, v_1 \rightarrow v_2)$  is an atomic action term which defines a change of value for the attribute  $f$  in  $S$  from  $v_1$  to  $v_2$ , where  $v_1, v_2 \in V_f$ . In the case when there is no change, we omit the right arrow sign, so for example,  $(f, v_1)$  means that the value of attribute  $f$  in  $S$  remains  $v_1$ , where  $v_1 \in V_f$ .

Atomic action terms model a single feature values transition pattern, but it does not model the association between feature values transition patterns.

**Definition 2** (*Action terms*) are defined as the smallest collection of expressions for a decision system  $S$  such that:

- If  $t$  is an atomic action term in  $S$ , then  $t$  is an action term in  $S$ .
- If  $t_1, t_2$  are action terms in  $S$  and  $\wedge$  is a 2-argument functor called composition, then  $t_1 \wedge t_2$  is a candidate action term in  $S$ .
- If  $t$  is a candidate action term in  $S$  and for any two atomic action terms  $(f, v_1 \rightarrow v_2), (g, w_1 \rightarrow w_2)$  contained in  $t$  we have  $f \neq g$ , then  $t$  is an action term in  $S$ .

Assuming that  $S$  is given, we will say from now on, *action term* instead of *action term in  $S$* .

**Definition 3** (*Domain of an action term*) The domain  $Dom(t)$  of an action term  $t$  is the set of features listed in the atomic action terms contained in  $t$ . For example,  $t = [(f, v_1 \rightarrow v_2) \wedge (g, w_1)]$  is an action term that consists of two atomic action terms, namely  $(f, v_1 \rightarrow v_2)$  and  $(g, w_1)$ . Therefore,  $Dom(t) = \{f, g\}$ .

Action rules are expressions that take the following form:  $r = [t_1 \Rightarrow t_2]$ , where  $t_1, t_2$  are action terms. The interpretation of the action rule  $r$  is that by triggering the action term  $t_1$ , we would get, as a result, the changes of states in action term  $t_2$ . We also assume that  $Dom(t_1) \cup Dom(t_2) \subseteq F$ , and  $Dom(t_1) \cap Dom(t_2) = \emptyset$ .

For example,  $r = [[(f, v_1 \rightarrow v_2) \wedge (g, w_2)] \Rightarrow (d, d_1 \rightarrow d_2)]$  means that by changing the state of feature  $f$  from  $v_1$  to  $v_2$ , and by keeping the state of feature  $g$  as  $w_2$ , we would observe a change in attribute  $d$  from the state  $d_1$  to  $d_2$ , where  $d$  is commonly referred to as the decision attribute.

### 9.3.1 Action Rules Evaluation

In [8] it was observed that each action rule can be seen as a composition of two classification rules. For instance, the rule  $r = [[(f, v_1 \rightarrow v_2) \wedge (g, w_2)] \Rightarrow (d, d_1 \rightarrow d_2)]$  is a composition of  $r_1 = [(f, v_1) \wedge (g, w_2)] \rightarrow (d, d_1)$  and  $r_2 = [(f, v_2) \wedge (g, w_2)] \rightarrow (d, d_2)$ . This fact can be recorded by the equation  $r = r(r_1, r_2)$ . Also, the definition of support (*Sup*) and confidence (*Conf*) of an action rule is based on support and confidence of classification rules (see below).

Assume that action rule  $r$  is a composition of two classification rules  $r_1$  and  $r_2$ . Then [8]:

- $Sup(r) = \min\{card(sup(r_1)), card(sup(r_2))\}$ ,
- $Conf(r) = conf(r_1) \cdot conf(r_2)$ ,

where  $conf(r_1)$  and  $conf(r_2)$  are the respective confidences of classification rules  $r_1$  and  $r_2$ .

In this context, the support of a classification rule  $r = [[(f_1, f_{11}) \wedge (f_2, f_{21}) \wedge \dots \wedge (f_k, f_{k1})] \rightarrow (d, d_1)]$  in a decision system  $S = (X, F \cup \{d\}, V)$ , where  $(\forall i \leq k)(f_i \in F \text{ and } f_{i1} \in V_{f_i}), d_1 \in V_d$  is defined as  $\text{sup}(r) = \{x \in X : (\forall i \leq k) [f_i(x) = f_{i1}] \text{ and } d(x) = d_1\}$ .

## 9.4 Meta-actions

Action rules are the perfect analysis of transition patterns that inform deciders about the possible changes to perform to reach a desired outcome. However, deciders still need to acquire additional knowledge on how to perform the necessary changes, and what are the object's states changes that actually occurred in the system. To build the strategies on the top of the actionable tasks that action rules provide, we use meta-actions that are defined as follows.

**Definition 4** (*Meta-actions*) associated with a decision system  $S$  are defined as higher level concepts used to model certain generalizations of actions rules [14]. Meta-actions, when executed, trigger changes in values of some flexible features in  $S$ .

Meta-actions are actions, outside of the features  $F$ , taken by deciders to transition objects from an initial known state with specific preconditions to different state with known postconditions. The changes in flexible features, triggered by meta-actions, are represented by atomic action terms for the respective features, and reported by the influence matrix presented in [14].

*Example 1* Let us take market segmentation for automobiles as an example of domain. Then, an automotive company, say company  $X$ , would divide its customers into: sedan cars seekers, sport cars seekers, wagon car seekers, all roads car seekers, and hatchback cars seekers. An extensive list of classification features can be used to classify them. For instance, age would be a good feature that would inform us that a young person would prefer a hatchback, and an older person would rather have a sedan. Another feature such as number of kids would inform us that bigger families would prefer wagons or all roads rather than smaller cars, marital status could be a good indication of sport cars preference, and so on. Other feature can be analyzed for the purpose of customer satisfaction and segmentation. However, all features cited earlier are stable features and would not allow values transitions regardless of meta-actions applied.

Let us assume another company  $Y$ , a luxurious car company, would like to acquire new customers, then their market segmentation would differ from  $X$ 's market segmentation. In fact, the new segmentation would be based on new classification features, some of them are: customers income, car price range, customer functional needs, car comfort, car quality, and customers favorite brand. Given those features, the company can classify the customers based on their favorite brand, and would like to attract customers from other less luxurious brands. Based on those features,  $Y$  can apply a meta-action  $M$  that transitions car price range to match what customers can

afford based on their income. This meta-action  $M$  can also affect car comfort and quality negatively to reduce production price while meeting customers functional needs. In this example  $Y$ 's segmentation is based on luxurious cars, functional cars, powerful sport cars, and so on. A new customer segment, called entry luxury cars, can join the brand with the help of meta-action  $M$  transitioning them from functional cars to entry-luxury cars.

*Example 2* To give a new example, let us assume that classification features in  $S$  describe teaching evaluations at some school and the decision feature represents their overall score. Explain difficult concepts effectively, Speaks English fluently, Stimulate student interest in the course, and Provide sufficient feedback are examples of classification features scored in the system. Then, examples of meta-actions associated with  $S$  will be: Change the content of the course, Change the textbook of the course, Post all material on the Web. Clearly, those meta-actions will trigger changes in some of the features described such as Provide sufficient feed-back and Stimulate student's interest in the course; however none of these three meta-actions will influence the feature Speaks English fluently and its values will remain unchanged [14].

*Example 3* Another example would be using Hepatitis as the application domain. Then the increase in blood cell plagues and the decrease in level of alkaline phosphatase are examples of atomic action terms. Drugs like Hepatil or Hepargen are seen as meta-actions triggering changes described by these two atomic action terms [11, 15]. It should be noted that Hepatil is also used to get rid of obstruction, eructation, and bleeding. However, Hepargen is not used to get rid of obstruction but it is used to get rid of eructation and bleeding. Some of the effects of those meta-actions are not necessary and can be seen as side effects. At the same time some needed changes are not triggered by the meta actions used and require the use of additional meta-actions.

#### 9.4.1 Meta-actions Influence Matrix

Consider several meta-actions, denoted  $M_1, M_2, \dots, M_n$ . Each one can invoke changes within values of some classification features in  $F = \{f_1, f_2, \dots, f_m\}$ . The expected changes of values of classification features on objects from  $S$  triggered by these meta-actions are described by the influence matrix  $\{E_{ij} : 1 \leq i \leq n \text{ and } 1 \leq j \leq m\}$ . The table below gives an example of an influence matrix associated with six meta-actions and three features:  $a$ ,  $b$ , and  $c$ .

For instance, let us describe the meta-action  $M_2$ . The influence matrix in Table 9.1 says that by executing  $M_2$  on objects in  $S$ , two atomic action terms are triggered. They are:  $(a, a_2 \rightarrow a_1)$  and  $(b, b_2 \rightarrow b_2)$ . It means that objects in  $S$  satisfying the description  $(a, a_2) \wedge (b, b_2)$  are expected to change their description to  $(a, a_1) \wedge (b, b_2)$ .

**Table 9.1** Meta-actions influence matrix

Meta-actions	<i>a</i>	<i>b</i>	<i>c</i>
$M_1$	—	$b_1$	$c_2 \rightarrow c_1$
$M_2$	$a_2 \rightarrow a_1$	$b_2$	—
$M_3$	$a_1 \rightarrow a_2$	—	$c_2 \rightarrow c_1$
$M_4$	—	$b_1$	$c_1 \rightarrow c_2$
$M_5$	—	—	$c_1 \rightarrow c_2$
$M_6$	$a_1 \rightarrow a_2$	—	$c_1 \rightarrow c_2$

Let us define  $\mathbf{M}(S)$  as a set of meta-actions associated with a decision system  $S$ . Let  $f \in F$ ,  $x \in X$ , and  $M \subset \mathbf{M}(S)$ , then, applying the meta-actions in the set  $M$  on an object  $x$  will result in  $M(f(x)) = f(y)$ , where object  $x$  is converted to object  $y$  by applying all meta-actions in  $M$  to  $x$ . Similarly,  $M(F(x)) = F(y)$ , where  $x$  is converted to the state of  $y$  by applying all meta-actions in  $M$  to  $x$  for all  $f \in F$ . Also, by  $F(Y)$ , where  $Y \subseteq X$ , we mean  $\{F(y) : y \in Y\}$ .

It should be mentioned here that an expert knowledge concerning meta-actions involves only classification features. Now, if some of these features are correlated with the decision feature, then the change of their values will cascade to the decision through the correlation. The goal of action rule discovery is to identify possibly all such correlations. At the same time, the goal of meta-actions discovery is to identify all the effects triggered by the application of such meta-actions.

#### 9.4.2 Mining Meta-actions Effects

The application of meta-actions on a set of objects triggers a set of effects within the closed information system (new objects can not be added). Those effects translate into changes in some flexible features values. Meta-actions are applied by practitioners without certainty on the effects for all objects. Each object might react differently to each meta-action, thus practitioners shall explore the personalized effects of applying meta-actions. In this section we present the methodology used to extract meta-actions effects on a closed information system.

Commonly, an information system is represented in a tabular format, where features represent the columns as attributes and rows represent the individual transactions. Each transaction is uniquely identified by a transaction ID,  $TID$ . In the context of meta-actions, objects might have one or more transactions and each object is identified by an Object ID,  $OID$ . In addition, mining meta-actions effects translates into mining the objects state transitions. In other words, mining the objects' transitions from an initial transaction, where meta-actions were applied, to another transaction belonging to the same object. Meta-actions transitions are mined after the application of meta-actions, and thus, it is important to keep track of the transactions' order. Therefore, an additional feature representing the order of the sequence of transactions is necessary. In addition, in a decision system, a particular feature called decision

feature is included, and represents the decision values for the particular transaction  $TID$  and object  $OID$ .

Mining meta-actions transformations requires the study of the transactional datasets in hand. Commonly, transactional datasets do not represent the objects temporal transformations resulting from applying meta-actions. To be able to mine meta-action's transformations (atomic action terms), objects have to be uniquely identified along with their transactions and clustered by their identifier. Object's transactions should be ordered based on temporal sequential order. Every two sequential object's transactions will be paired for every meta-action based on a temporal precedence relationship. The resulting pairwise partition will model the atomic action terms transitions for each object given the meta-action applied. For instance, given a patient visits recorded in our dataset with high blood pressure, high fever, and headaches for his first visit to the doctor, who gave him/her a treatment  $m$ , at the second visit the patient diagnosis displays no fever, high blood pressure and no headache. In such a case we can extract the following atomic action terms for this patient pair of visits:  $(fever, high \rightarrow no)$ ,  $(blood\ pressure, high \rightarrow high)$ ,  $(headaches, yes \rightarrow no)$ .

Now, we introduce the set of transactions  $T$ . Let us assume that  $S = (X, F \cup \{d\}, V)$  is a decision system, where  $X$  is a set of objects,  $F$  is a set of classification attributes,  $d$  is the decision attribute, and  $V$  is a set of values of attributes in  $F \cup \{d\}$ , such that  $f(X) \subseteq V$ , for any  $f \in F$ . Also, let us assume that  $\mathbf{M}(S)$  is a set of meta-actions associated with  $S$ . In addition, we define the set  $\{s_{i,j} : j \in J_i\}$  of ordered transactions  $J_i$  associated with  $x_i \in X$ , such that  $s_{i,j} = [(x_i, F(x_i)_j)]$ , where  $(\forall i, j) [s_{i,j} \in T]$ . The set  $F(x_i)_j$  is defined as the set of attribute values  $\{f(x_i) : f \in F\}$  of the object  $x_i$  in the transaction uniquely represented by the transaction identifier  $j$ . Each transaction represents the current state of the object when recorded with respect to a temporal order based on  $j$  for all  $s_{i,j} \in T$ .

We define a precedence relationship denoted as  $>_p$  on the system  $S$  to help locate the position of each transaction within each object's ordered transaction set. Given two transactions  $s_{i,j}$  and  $s_{i,k}$  for an object  $x_i \in X$ , the precedence relationship  $s_{i,j} >_p s_{i,k}$  represents the order of the recorded transactions for the object  $x_i$ , and says that the transaction  $s_{i,j}$  was recorded before the transaction  $s_{i,k}$ .

To strengthen this relationship, we define the set  $P(S)$  of pairs  $(s_{i,j}, s_{i,k})$  such that  $(s_{i,j}, s_{i,k}) \in P(S)$  if and only if  $s_{i,k}$  occurred directly after  $s_{i,j}$  (there is no other transaction between them in the system  $S$ ).

It should be observed that any pair  $(s_{i,j}, s_{i,k}) = ([x_i, F(x_i)_j], [x_i, F(x_i)_k])$  in  $P(S)$  represents a set of atomic action terms  $\{(f, f(x_i)_j \rightarrow f(x_i)_k) : f \in F\}$ .

We assume that there is always a set of meta-actions in  $M$  applied before any transaction  $s_{i,k}$  with the exception of the very first transaction  $s_{i,1}$  for each object  $x_i \in X$ . This suggests that the transaction  $s_{i,k}$  in the pair  $(s_{i,j}, s_{i,k})$  is a direct consequence of applying some set of meta-actions  $m$  ( $m \subset M$ ) to the object  $x_i \in X$ , and supports the assumption that the state of the object  $x_i$  is being affected by these meta-actions.

As we already observed, each transaction pair  $(s_{i,j}, s_{i,k})$  encompasses the set of atomic action terms  $A_{j,k}$  of the form  $\{(f, f(x_i)_j \rightarrow f(x_i)_k) : f \in F\}$ , that defines

a change of value  $f(x_i)_j$  derived from the transaction  $s_{i,j}$  to  $f(x_i)_k$  derived from the transaction  $s_{i,k}$ , for each attribute  $f \in F$  and  $x_i \in X$ .

Now, we can use the sets  $A_{j,k}$  to build the influence matrix [7, 13] covering all  $m \in M$ . Sets of action terms representing meta-actions in  $M$  can be built from sets of pairs in  $P(S)$ . Depending on the objects' states, some of the atomic action terms in  $A_{j,k}$  may not be triggered by meta-actions. To be more precise, for a given meta-action  $m_j$ , only objects  $x_l \in X$  that satisfy the following condition will be affected:

$$\exists(s_{i,j}, s_{i,k}) \in P(S) \text{ such that } F(x_l) \cap F(x_i)_j \neq \emptyset, \text{ where } s_{i,j} = (x_i, F(x_i)_j).$$

This way, by applying the meta-action  $m$  on  $x_i$  we will cover the set of attribute values  $\{F(x_l) \cap F(x_i)_j\}$ , thus the underlying subset of atomic action terms. This subset can be seen as an action term  $t$  containing a set of atomic actions with the domain  $Dom(t) = \{a \in F : a(x_l) = a(x_i)_j\}$ . Multiple action terms can be formed this way, however, not all possible action terms are applicable to a given dataset. The number of possible action terms for each meta-action grows monotonously with the number of extracted pairs, the attributes' domains sizes, and the number of transactions for each object in  $X$ .

Ultimately, in the worst case scenario every object is different and reacts differently to each specific meta-action. This might result in a large number of action terms for each meta-action. Possible conflicts within the same meta-action scope such as  $(a, a_j \rightarrow a_k)$  and  $(a, a_j \rightarrow a_l)$ , or non useful action terms such as  $(a, a_j \rightarrow a_j)$  for  $a \in F$  and  $a_j, a_k, a_l \in V_a$  might be extracted. Not all atomic action terms are useful for all objects; however, it is important to keep a record of the different transitions for the sake of object personalized meta-actions.

We can evaluate the different action terms composing each given meta-action to avoid conflicts and use the more appealing ones to the treated object. Similarly the frequent itemsets used in the Apriori [1] algorithm, frequent action terms can be extracted from multiple pairs. Multiple action terms of different sizes can be formed from the resulting atomic action terms (pairs). Frequent action terms are characterized by their frequency of occurrence throughout all the meta-action partition of the dataset (all the meta-action pairs).

#### 9.4.3 Meta-action Evaluation

To evaluate the meta-actions, we need to evaluate the action terms composing them. A simple evaluation metric consist of the frequency of occurrence (or support) for each action term. Pairs extracted from the data share common atomic action terms transitions, thus, they share common action terms. For each action term  $t_j$ , we define its likelihood support  $Like(t_j)$  as:

$$rLike(t_j) = card \left( \{(s_{i,k}, s_{i,l}) \in P(S) : Left(t_j) \subseteq F(x_i)_k \text{ and } Right(t_j) \subseteq F(x_i)_l\} \right), \quad (9.1)$$

where  $(s_{i,k}, s_{i,l}) = ([x_i, F(x_i)_k], [x_i, F(x_i)_l])$ ,  $x_i \in X$ ,  $Left(t_j)$  is defined as the feature values in the left-hand side of the frequent action term  $t_j$ , and  $Right(t_j)$  is the right-hand side of  $t_j$  such that if  $t_j = [(a_{l1} \rightarrow a_{r1}) \wedge (a_{l2} \rightarrow a_{r2}) \wedge \dots \wedge (a_{ln} \rightarrow a_{rn})]$  then  $Left(t_j) = \{a_{l1}, a_{l2}, \dots, a_{ln}\}$ , and  $Right(t_j) = \{a_{r1}, a_{r2}, \dots, a_{rn}\}$ .

The likelihood support of action term measures the transition likelihood of their attribute values but it neither takes into consideration the conflicts of action terms nor handles the meta-actions comparison in a normalized way. A more sophisticated way to evaluate action terms is by computing their likelihood confidence, and thus a possible meta-action confidence metric. The likelihood confidence of an action term  $t_j$  is computed as follow:

$$TermConf(t_j) = Like(t_j) / sup(Left(t_j)). \quad (9.2)$$

Given the set of atomic action terms  $\{t_i : 1 \leq i \leq n\}$  composing a meta-action  $m_j$ , we can define the confidence of  $m_j$  as the weighted sum of its atomic action terms likelihood confidence where the weights represent atomic action terms likelihood support. To be more precise, the meta-action confidence  $MetaConf(m_j)$  is computed as follows:

$$MetaConf(m_j) = \frac{\sum_{i=1}^n Like(t_i) \cdot TermConf(t_i)}{\sum_{i=1}^n Like(t_i)}, \quad (9.3)$$

where  $n$  is the number of atomic action terms in  $m_j$ .

Note that some action terms will have the likelihood support below the required threshold value, therefore, they will not be considered as frequent action terms. However, those action terms are considered as outliers and it is important to keep track of them in the meta-actions for objects personalized meta-actions.

## 9.5 Meta-actions Versus Action Rules

Meta-actions and action rules are similar concepts since they aim at extracting transition patterns from information systems. However, the meta-action extraction process extracts a subset of the action terms extracted by the action rule extraction process.

**Table 9.2** Meta-actions versus action rules

Action rules	Meta-actions
One objects is one instance	One objects is a set of instances
No instance order	Temporal instance order
Support is minimum support of classification rule	Likelihood is the number of real transitions
Confidence independent probability	Term confidence and execution confidence
Possible transitions	Real transitions
Rules	Collection of action terms

The similarities and differences of both concepts are discussed in this section and reported in Table 9.2.

Action rules are commonly used by decision makers to discover possible changes in objects' state, which would ultimately transition the objects' overall state to a more desirable state with regards to the decision feature. Several techniques exist to extract action rules. The datasets used to extract action rules are composed of instances, where each object is described by one instance. In other words, each object is recorded once in the dataset, and the object state is represented by its instance features values. All possible transitions from any instance to any other instance in the dataset are discovered in the action rules discovery process regardless of the order in which they were recorded. This suggests that there is no temporal ordering relation amongst instances. Therefore, some transitions discovered in the action rules discovery process may not be applicable in a real life scenario. For example, let us assume that an information system models cancer patient visits, where each patient visit describes the patient pathological state in term of diagnoses. In this example dataset, each patient visit represents an instance. Now, let us assume that each visit record includes three diagnoses as features, namely: tumor size, chemo level, and number of chemotherapy performed. Also, it includes the information whether the patient is stable or unstable stored as decision feature, at the visit time. The diagnosis features are recorded as *TSize* for tumor size, *CLevel* for chemo level and *NChemo* for the number of chemotherapy already performed on the patient, and the patient overall state recorded as *State*. Note here that each patient visit is considered as an object for the action rule discovery process. The following action rule may be discovered by the system:  $(TSize, 7 \rightarrow 4) \wedge (CLevel, 4 \rightarrow 3) \wedge (NChemo, 5 \rightarrow 4) \Rightarrow (State, Unstable \rightarrow Stable)$ . In this action rule, we are interested in the patient overall state *State*, and we would like to move patients from an unstable state to a stable one. Nonetheless, we cannot transition the *NChemo* from 5 to 4 since we already performed 5 chemo on this patient. Therefore, this action rule would not be applicable in real life. It was discovered using all the visits in our dataset, where each visit is considered as an object, regardless of the fact that a patient may have several visits to the hospital. In other words, patients are not taken into consideration since the object instances are visits and not patients.

Meta-actions on the other hand, represent transitions from instances that occurred in a specific order for specific objects in real life scenarios, which insures their applicability. Meta-actions do not model rules, they rather model transition effects that are represented in an influence matrix. By using the previous example and adding a new feature that describes the visit number, we could sort the visits by their temporal order. In addition, we treat our system as multiple subsystems of visits identified by the patient ID. This way, we could mine real transitions in terms of patients pathological state. As a result the transition  $(NChemo, 5 \rightarrow 4)$  would not be extracted since the chemo number 4 would have taken place before chemo number 5, and the meta-action mining process takes the visit order into consideration. In addition, different patients might react differently to the chemotherapies with regards to their cancer level and tumor size. Using meta-actions, we extract transitions that occurred in real life for each patient. For instance, we may extract the following transitions:

$(TSize, 7 \rightarrow 6) \wedge (CLevel, 4) \wedge (NChemo, 5 \rightarrow 6) \wedge (State, Unstable \rightarrow Stable)$ , where the left hand side of the action term took place at a visit that occurred prior to the right hand side. Furthermore, the transitions were mined from instances belonging to a unique patients. In summary, the introduction of the instances order and their identifier with regard to the object eliminated any confusion.

The purpose of meta-actions is to trigger transitions in feature values that will change the object state. Eventually, the transitions triggered by meta-actions will trigger an action rule and will cascade into transitioning the decision feature value. As mentioned earlier, meta-actions' transitions are a subset of the transitions extracted in the action rules extraction process. In addition, meta-actions play a passive role, in the sense that they do not change decision feature values. They rather inform decision makers of possible transitions. On the contrary, action rules play an active role that help decision makers drive their strategy, and explicitly look into the transitions that affect the decision feature. In other words, meta-actions are not replacing action rules; instead, enhancing the process of selecting the best action rules.

### 9.5.1 Action Rules Selection by Meta-actions

In the effort of selecting the best action rules, meta-actions play an important evaluation role. After extracting action rules, meta-actions inform us, amongst other things, on whether the extracted action rules are applicable. Meta-actions also provide decision makers with the confidence of executing the antecedent side of an action rule. Given a set of action rules, and an influence matrix describing meta-actions transitions in our system, we strive to select the best applicable action rules and their respective triggers. This is done by first dividing the set of action rules into applicable and non-applicable action rules. Applicable action rules are the rules, which antecedent sides are covered by the influence matrix. Then for each action rule, we select the best coverings of the action rule from the influence matrix. The best coverings of an action rule are the maximal action terms in the influence matrix. By maximal action term, we mean the action term with the largest number of atomic terms covering the action rule. This is performed by intersecting the antecedent side of the action rule with all the action terms in the influence matrix. It is important to note that the action terms with higher number of atomic action terms will have a higher confidence. As one can guess, two or more action terms may be required to cover an action rule. Similarly, two or more meta-actions may be required to cover an action rule.

We described earlier how to compute the confidence of an action term  $TermConf$  in 9.2; however, we still need to define how to compute the confidence of multiple action terms, namely global confidence  $GlobConf$ . Computing the confidence of multiple action terms depends on whether the action terms belong to the same meta-action or not. In fact, action terms  $\{t_1, t_2, \dots, t_n\}$  that belong to different meta-actions are independent since they are extracted from different object pop-

ulations; therefore, their confidences are independent and their global confidence is the product probability of independent confidences, as shown in the following:

$$GlobConf(\{t_1, t_2, \dots, t_n\}) = \prod_{i=1}^n TermConf(t_i). \quad (9.4)$$

On the contrary, action terms that belong to the same meta-action are extracted from the same object population, and might be extracted from the same objects. Therefore, such action terms are dependent on the objects and their global confidence is dependent on their transition probability. To avoid confusion, and for the sake of simplicity, we will define the global confidence of action terms from the same meta-action as follows:

$$GlobConf(\{t_1, t_2, \dots, t_n\}) = Like\left(\bigcup_{i=1}^n \{t_i\}\right) / sup\left(Left\left(\bigcup_{i=1}^n \{t_i\}\right)\right). \quad (9.5)$$

Global confidence will inform us on how well we can trigger the antecedent side of an action rule; however, it does not inform us about how well the features values transitioned by the meta-action will cascade into the desired decision feature value. For this reason, we define a new metric called execution confidence *ExConf* that computes the confidence of execution of an action rule. The execution confidence of an action rule  $r$  triggered by the set of meta-actions  $m$  is as follows:

$$ExConf(m, r) = GlobConf(m(r)) \cdot Conf(r), \quad (9.6)$$

where  $m(r)$  represents the set of action terms used in triggering the antecedent side of the action rule  $r$ .

With the introduction of the *ExConf*, one could select the action rules with the highest execution confidence, along with their corresponding meta-actions. Doing so would insure that the action rules chosen are more likely to be accurate. However, decision makers may also be interested in the highest return on investment solution, which would be good enough to solve the issue in hand. In fact, meta-actions are commonly associated with a cost based on the domain of interest. For example, in the healthcare domain, each treatment is associated with a cost, and patients are discharged with a bill including all their medical expenses.

Let us assume that cost  $C_i$  is associated with each meta-action  $m_i \in M(S)$  used. Then a good measure associating the cost  $C_i$  and the execution confidence *ExConf* would be the satisfaction rate noted as *SatRate*. The satisfaction rate gives a pointer to which action rule  $r$  is good enough; in other words, which action rule and corresponding meta-actions incurs the minimum cost while returning an acceptable execution confidence. The satisfaction rate for a rule  $r$  and a set of corresponding meta-actions  $M_r$  is computed as follows:

$$SatRate(M_r, r) = \frac{ExConf(M_r, r)}{\lambda \sum_{i=1}^n C_i}, \quad (9.7)$$

where  $n$  is the number of meta-actions used, and  $0 < \lambda \leq 1$  is a cost coefficient chosen by decision makers.

## 9.6 Side Effects

The main goal of meta-actions is to trigger action rules. However, it is often the case that when applying meta-actions for the purpose of executing a specific action rule, a set of unrelated additional and potentially harmful atomic action terms are triggered. The additional action terms resulting from the meta-action application are called side effects. Meta-actions might move some object features values from negative to positive values  $f(x) \in E_n(x)$  and  $f(y) \in E_p(y)$  (desirable positive side effects), and some object features values from positive to negative values  $f(x) \in E_p(x)$  and  $f(y) \in E_n(y)$  (undesirable negative side effects). Even though the features transitioning from positive to negative values might result in catastrophic situations, they were not fully investigated in previous work involving action rules discovery. In the following, we depict two types of side effects and we give a brief description for each type.

### 9.6.1 *Meta-actions Side Effects*

Side effects in the context of meta-actions are the effects that occur for specific small clusters of objects. This type of side effects is discovered in the meta-action extraction process. It is represented by the action terms, extracted in the meta-action extraction process, with very low or unusual likelihood of occurrence. In fact, this type of transition is very rare in our dataset, and was extracted from a very small number of objects. We can think of this type of effect as minor effects of a meta-action, that does not represent the core goal or trigger of this meta-action. Detecting this type of side effects is done by setting a minimum likelihood for the action terms, or setting a minimum jump in the likelihood between the action terms.

### 9.6.2 *Action Rules Side Effects*

Side effects in the context of action rules are the unintended changes in some flexible features values that meta-actions trigger on objects. In other words, those effects are triggered by meta-actions but are outside of the intended action rule scope. To discover those side effects, we can perform two set operations. We first start by

performing a minus set operation between the antecedent side of the action rule and the meta-actions action terms reported in the influence matrix. The result is then intersected with the objects precondition to get the final set of side effects.

## 9.7 Experiments

We performed a set of experiments on the Florida State Inpatient Database using our meta-action and association action rules extraction system. We finally evaluated the action rules extracted using the meta-actions applied.

### 9.7.1 Dataset Description

In the research, we used the Florida State Inpatient Databases (SID) that is part of the Healthcare Cost and Utilization Project (HCUP). The Florida SID dataset contains records from several hospitals in the Florida State. It contains over 2.5 million visit discharges from over 1.5 million patients. The dataset is composed of five tables, namely: AHAL, CHGH, GRPS, SEVERITY, and CORE.

The main table used in this chapter is the *Core* table. The *Core* table contains over 280 attributes; however, many of those attributes are repeated with different codification schemes. In the following experiments, we used the Clinical Classifications Software (CCS) that consists of over 260 diagnosis categories, and 231 procedure categories. This system is based on ICD-9-CM codes. In our experiments, we used fewer attributes that are described in this section. Each record in the Core table represents a visit discharge. A patient may have several visits in the table. One of the most important attributes of this table is the *VisitLink* attribute, which describes the patient's ID. Another important attribute is the *Key*, which is the primary key of the table that identifies unique visits for the patients and links to the other tables. As mentioned earlier, a *VisitLink* might map to multiple *Key* in the database.

The *Core* table reports up to 31 diagnoses per discharge as it has 31 diagnosis columns. However, patient's diagnoses are stored in a random order in this table. For example, if a particular patient visits the hospital twice with heart failure, the first visit discharge may report a heart failure diagnosis at diagnosis column number 10, and the second visit discharge may report a heart failure diagnosis at diagnosis column number 22. Furthermore, it is worth mentioning that it is often the case that patients examination returns less than 31 diagnoses.

The *Core* table also contains 31 columns describing up to 31 procedures that the patient went through. Even though a patient might go through several procedures in a given visit, the primary procedure that occurred at the visit discharge is assumed to be the first procedure column. The *Core* table also contains an attribute called *DaysToEvent*, which describes the number of days that passed between the admission

**Table 9.3** Mapping between attributes and concepts features

Attributes	Concepts
VisitLink	Patient Identifier
DaysToEvent	Temporal visit ordering
DXCCSn	$n$ th Diagnosis, flexible attributes
PRCCSn	$n$ th Procedure, meta-actions
Race, Age Range, Sex, ...	Stable attributes
DIED	Decision attribute

to the hospital and the procedure day. This field is anonymized in order to hide the patients' identity.

Furthermore, the *Core* table also contains a feature called *DIED*, that informs us on whether the patient died or survived in the hospital for a particular discharge. In addition, there are several demographic data that are reported in this table as well, such as: Race, Age Range, Sex, living area, ... The following Table 9.3 maps the attributes from the *Core* table to the concepts and notations used in this chapter.

### 9.7.2 Meta-action Extraction

We extracted meta-actions action terms from the Florida SID dataset, and computed their likelihood and term confidence using the method described in Sect. 9.4.2. In these experiments, we extracted action terms for 231 procedures, considered here as meta-actions; however, for the sake of this chapter, we report the results for five meta-actions described in Table 9.4 with their CCS procedure codes. We built an influence matrix represented by Table 9.6 that describes our findings in terms of meta-actions' action terms. Table 9.6 reports few examples of action terms of size 1–4 extracted for each meta-action; however, we extracted action terms with more than 4 atomic terms and we may extract action terms of size up to the number of features. We also included the description of the CCS diagnoses codes for some of the most significant action terms' features in Table 9.5.

**Table 9.4** Mapping between CCS single level procedure codes and their description [3]

PRCCS1	Description
43	Heart valve procedures
44	Coronary artery bypass graft (CABG)
45	Percutaneous transluminal coronary angioplasty (PTCA)
47	Operations on lymphatic system
48	Insertion; revision, replacement; removal of cardiac pacemaker or cardioverter/defibrillator

**Table 9.5** Mapping between CCS single level diagnosis codes and their description [3]

Diagnosis code	Description
101	Coronary atherosclerosis and other heart disease
55	Fluid and electrolyte disorders
62	Coagulation and hemorrhagic disorders
106	Cardiac dysrhythmias
99	Hypertension with complications and secondary hypertension
158	Chronic kidney disease
114	Peripheral and visceral atherosclerosis
108	Congestive heart failure; nonhypertensive
59	Deficiency and other anemia
58	Other nutritional; endocrine; and metabolic disorders
117	Other circulatory disease
105	Conduction disorders
155	Other gastrointestinal disorders
663	Screening and history of mental health and substance abuse codes
257	Other aftercare
259	Residual codes; unclassified
96	Heart valve disorders
253	Allergic reactions
211	Other connective tissue disease

As mentioned in the dataset description, the flexible attributes represented by diagnosis are not ordered by attribute column (there are no fixed attribute columns). For this reason, we represented each visit with a set of diagnoses instead of a set of fixed attributes. However, as you can note in Table 9.6, to simplify the reporting, we assume the domain of each diagnosis is in {0, 1}, where 1 means that the patient is diagnosed with that particular diagnosis at the current visit, and 0 means that the patient is not diagnosed with that specific diagnosis at the current visit.

We first grouped our dataset by procedures that are represented here by the first primary procedure attribute *PRCCS1*. We also grouped the visits by patient ID *VisitLink* and ordered each patient's visits by the *DaysToEvent* attribute. We then built pairs of visits for the meta-action applied, and extracted the action terms from the pairs. Finally, we computed the likelihood and term confidence for each action term. As you can see from Table 9.6, the likelihood and term confidence of the action terms extracted are very high. The higher the likelihood of the action term, the more important the diagnoses and the more likely the diagnoses involved are the main reason of the procedure. In addition, the higher the term confidence the more stable the meta-action and procedure result.

**Table 9.6** Influence matrix like table reporting support and confidence

Meta-actions	Action terms with CCS codes	Likelihood	Term Conf(%)
43	(101, 1 → 0)	167	84
	(55, 1 → 0)	165	91
	(62, 1 → 0)	146	96
	(106, 1 → 0)	135	88
	(99, 1 → 0)	66	96
	(55, 1 → 0) ∧ (62, 1 → 0)	46	90
	(55, 1 → 0) ∧ (106, 1 → 0)	44	90
	(158, 1 → 0) ∧ (106, 1 → 0) ∧ (99, 1 → 0)	14	87.5
	(55, 1 → 0) ∧ (158, 1 → 0) ∧ (101, 1 → 0) ∧ (99, 1 → 0)	8	100
44	(101, 1 → 0)	181	85
	(108, 1 → 0)	92	91
	(62, 1 → 0)	97	98
	(114, 1 → 0)	86	93
	(58, 1 → 0)	193	91
	(108, 1 → 0) ∧ (106, 1 → 0)	32	94
	(55, 1 → 0) ∧ (106, 1 → 0)	42	91
	(62, 1 → 0) ∧ (55, 1 → 0) ∧ (106, 1 → 0)	16	94
	(55, 1 → 0) ∧ (108, 1 → 0) ∧ (58, 1 → 0) ∧ (59, 1 → 0)	8	100
45	(101, 1 → 0)	262	78
	(117, 1 → 0)	97	86
	(105, 1 → 0)	85	83
	(155, 1 → 0)	81	88
	(663, 1 → 0)	201	85
	(55, 1 → 0) ∧ (59, 1 → 0)	33	89
	(58, 1 → 0) ∧ (257, 1 → 0)	40	77
	(259, 1 → 0) ∧ (663, 1 → 0) ∧ (101, 1 → 0)	26	76
	(58, 1 → 0) ∧ (259, 1 → 0) ∧ (663, 1 → 0) ∧ (101, 1 → 0)	6	67
47	(101, 1 → 0)	347	81
	(117, 1 → 0)	135	88
	(105, 1 → 0)	135	82
	(96, 1 → 0)	99	93
	(155, 1 → 0)	97	94
	(117, 1 → 0) ∧ (257, 1 → 0)	45	80
	(259, 1 → 0) ∧ (253, 1 → 0)	41	90
	(117, 1 → 0) ∧ (257, 1 → 0) ∧ (101, 1 → 0)	24	83
	(117, 1 → 0) ∧ (105, 1 → 0) ∧ (257, 1 → 0) ∧ (101, 1 → 0)	9	75

(continued)

**Table 9.6** (continued)

Meta-actions	Action terms with CCS codes	Likelihood	Term Conf(%)
48	(257, 1 → 0)	335	87
	(96, 1 → 0)	152	90
	(211, 1 → 0)	115	86
	(106, 1 → 0)	147	94
	(155, 1 → 0)	137	91
	(59, 1 → 0) ∧ (58, 1 → 0)	69	86
	(55, 1 → 0) ∧ (58, 1 → 0)	69	87
	(58, 1 → 0) ∧ (59, 1 → 0) ∧ (55, 1 → 0)	27	80
	(55, 1 → 0) ∧ (58, 1 → 0) ∧ (158, 1 → 0) ∧ (99, 1 → 0)	11	100

### 9.7.3 Action Rules Extraction and Evaluation

We extracted action rules using the association action rules extraction method described in [9]. We commonly extract action rules from the whole dataset; however, for the sake of this chapter, we extracted action rules from a subset of the dataset that contains patient visit records involving the meta-actions reported in Table 9.6. It is important to note that the meta-action attribute, *PRCCS1*, is not involved in the action rules extraction process. In addition, neither the patient ID, *VisitLink*, nor the ordering attributes, *DaysToEvent*, are used in the action rules extraction process.

Following this setting, we extracted the following two rules with their respective support and confidence, where patients stay alive; more precisely, the decision feature *DIED* stays at 0:

$$r_1 = (58, 1 \rightarrow 0) \wedge (59, 1 \rightarrow 0) \wedge (55, 1 \rightarrow 0) \Rightarrow (\text{DIED}, 0 \rightarrow 0)$$

where *sup* = 334, and *conf* = 85 %,

$$r_2 = (62, 1 \rightarrow 0) \wedge (55, 1 \rightarrow 0) \wedge (106, 1 \rightarrow 0) \Rightarrow (\text{DIED}, 0 \rightarrow 0)$$

where *sup* = 101, and *conf* = 78 %.

Now, we would like to apply meta-actions to trigger those rules. Therefore, we need to pick up the action terms that cover the antecedent side of each rule and their corresponding meta-actions. If we apply the set of meta-actions  $M_{\{48\}} = \{48\}$  to action rule  $r_1$ , we will get the following global confidence and execution confidence:  $\text{GlobConf}(r_1) = 80\%$  and  $\text{ExConf}(M_{\{48\}}(r_1)) = 68\%$ . Whereas, if we apply the meta-actions  $M_{\{44\}} = \{44\}$  we will get:  $\text{GlobConf}(r_1) = 100\%$  and  $\text{ExConf}(M_{\{48\}}(r_1)) = 85\%$ . In other words, if a patient has the following diagnoses  $\{55, 59, 58\} = \{\text{Fluid and electrolyte disorders, Deficiency and other anemia, Other nutritional; endocrine; and metabolic disorders}\}$  it is better to perform a Coronary artery bypass graft (CABG), CCS code of 44, rather than Insertion; revision,

replacement; removal of cardiac pacemaker or cardioverter/defibrillator, CCS code of 48. However, depending of the cost of the meta-action and the satisfaction rate, a practitioner may make a different decision.

Similarly, we can apply meta-actions  $M_{\{44\}} = \{44\}$ , Coronary artery bypass graft (CABG), to trigger  $r_2$  and resolve the diagnoses  $\{62, 55, 106\} = \{\text{Coagulation and hemorrhagic disorders, Fluid and electrolyte disorders, Cardiac dysrhythmias}\}$  and we will get:  $\text{GlobConf}(r_1) = 94\%$  and  $\text{ExConf}(M_{\{48\}}(r_1)) = 73.32\%$ .

We lack the cost of meta-actions in our dataset; hence, we cannot compute the *SatRate*. Nonetheless, this information can be obtained via consultation with a practitioner. If we assume that the cost of any given meta-action is the same and that  $\lambda$  is selected as a constant for each meta-action, then the *SatRate* will essentially be equal to the *ExConf* and practitioners can base their decision on the best execution confidence.

## 9.8 Conclusion

Nowadays, action rules are used in several industries and the healthcare industry among others is a very sensitive area. Results from action rules extraction process have to be thoroughly evaluated and analyzed to be used in such industries. Action rules are commonly constructed from feature values patterns and not from transition patterns. In this chapter, we used meta-actions to evaluate action rules and we introduced new evaluation metrics. We used the 2010 Florida State Inpatient Databases (SID), and extracted meta-actions and action rules from this dataset. We evaluated meta-actions applied to action rules with the different metrics and compared the results to traditional metrics. Our results show the effectiveness of meta-actions in evaluating action rules and the rigorousness of our evaluation metrics. In future work, we will explore the meta-actions' extra action terms, considered as side effects, that covered patients' preconditions and did not cover action rules when applying meta-actions.

## References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Proceedings of the 20th International Conference on Very Large Data Bases. VLDB'94, pp. 487–499. Morgan Kaufmann Publishers Inc., San Francisco (1994)
2. Cost, H., (HCUP), U.P.: HCUP state inpatient databases (SID), agency for healthcare research and quality, rockville, md. [www.hcup-us.ahrq.gov/sidoverview.jsp](http://www.hcup-us.ahrq.gov/sidoverview.jsp) (2005–2009)
3. Cost, H., (HCUP), U.P., for Healthcare Research, A., Quality: Clinical classifications software (CCS) for ICD-9-CM. Website. <http://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp>
4. Im, S., Raš, Z.: Action rule extraction from a decision table: ARED. In: Foundations of Intelligent Systems. Proceedings of ISMIS'08, pp. 160–168. Springer, Toronto (2008)

5. Kohli, D., Raś, Z., Thompson, P., Jastreboff, P., Wieczorkowska, A.: From music to emotions and tinnitus treatment, initial study. In: Foundations of Intelligent Systems. Proceedings of ISMIS 2012 Symposium, pp. 244–253. Springer (2012)
6. Qiao, Y., Zhong, K., Wang, H., Li, X.: Developing event-condition-action rules in real-time active database. In: Proceedings of the 2007 ACM Symposium on Applied Computing. SAC '07, pp. 511–516. ACM, New York (2007)
7. Raś, Z., Dardzińska, A.: Action rules discovery based on tree classifiers and meta-actions. In: Proceedings of the 18th International Symposium on Foundations of Intelligent Systems. ISMIS'09, pp. 66–75. Springer, Berlin (2009)
8. Raś, Z., Dardzinska, A.: From data to classification rules and actions. *Int. J. Intell. Syst.* **26**(6), 572–590 (2011)
9. Raś, Z., Dardzinska, A., Tsay, L., Wasyluk, H.: Association action rules. In: Proceedings of IEEE International Conference on Data Mining Workshops. ICDMW '08, pp. 283–290 (2008)
10. Raś, Z., Wieczorkowska, A.: Action-rules: how to increase profit of a company. In: Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery. PKDD'00, pp. 587–592. Springer, London (2000)
11. Raś, Z., Wyrzykowska, E., Wasyluk, H.: ARAS: action rules discovery based on agglomerative strategy. In: Proceedings of the 3rd ECML/PKDD International Conference on Mining Complex Data. MCD'07, pp. 196–208. Springer, Berlin (2008)
12. Rauch, J., Šimůnek, M.: Action rules and the Guha method: preliminary considerations and results. In: Proceedings of the 18th International Symposium on Foundations of Intelligent Systems. ISMIS'09, pp. 76–87. Springer, Berlin (2009)
13. Tzacheva, A., Raś, Z.: Association action rules and action paths triggered by meta-actions. In: Proceedings of the 2010 IEEE International Conference on Granular Computing. GRC'10, pp. 772–776. IEEE Computer Society, Washington (2010)
14. Wang, K., Jiang, Y., Tuzhilin, A.: Mining actionable patterns by role models. In: Proceedings of the 22nd International Conference on Data Engineering. ICDE '06, pp. 16–16 (2006)
15. Wasyluk, H., Raś, Z., Wyrzykowska, E.: Application of action rules to Hepat clinical decision support system. *J. Exp. Clin. Hepatol.* **4**(2), 46–48 (2008)
16. Zhang, H., Zhao, Y., Cao, L., Zhang, C.: Combined association rule mining. In: Proceedings of the 12th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining. PAKDD'08, pp. 1069–1074. Springer, Berlin (2008)
17. Zhang, X., Raś, Z., Jastreboff, P., Thompson, P.: From Tinnitus data to action rules and Tinnitus treatment. In: Proceedings of IEEE International Conference on Granular Computing (GrC), pp. 620–625 (2010)

# **Chapter 10**

## **Irrelevant Feature and Rule Removal for Structural Associative Classification Using Structure-Preserving Flat Representation**

**Izwan Nizal Mohd Shaharanee and Fedja Hadzic**

**Abstract** Practical applications of association rule mining often suffer from overwhelming number of rules that are generated, many of which are not interesting or useful for the application in question. Removing irrelevant features and/or rules comprised of irrelevant features can significantly improve the overall performance. Many statistical and constraint based measures are used to discard unnecessary and irrelevant features and rules when vectorial or tabular data is in question. In contrast, the use of such measures is limited in the tree-structured data domain, due to the structural aspects that are not easily incorporated. In this chapter, we explore the use of a feature subset selection measure as well as a number of common statistical interestingness measures via a recently proposed structure-preserving flat representation for tree-structured data such as XML. A feature subset selection is used prior to association rule generation. Once the initial set of rules is obtained, irrelevant rules are determined as those that are comprised of attributes not determined to be statistically significant for the classification task. The experiments are performed using real world web access trees and property management dataset. The results indicate that where the dataset has more standard structure a large number of insignificant rules will be discarded and accuracy will increase. However, where the tree instances can vary greatly in terms of structure and label distribution among nodes, while many rules are removed and the accuracy increases, there is a significant reduction in coverage rate of the rule set.

**Keywords** Tree-structured data · Association rule based classification · Feature subset selection · Statistical interestingness

---

I.N.M. Shaharanee (✉)

School of Quantitative Sciences, Universiti Utara Malaysia, Sintok, Malaysia  
e-mail: nizal@uum.edu.my

F. Hadzic

Department of Computing, Curtin University, Perth, Australia  
e-mail: fedja.hadzic@curtin.edu.au

## 10.1 Introduction

Real world datasets often contain attributes that are irrelevant or redundant for the classification problem at hand. These features can degrade the performance and interfere with the learning mechanism typically resulting in a reduction in the quality and generality of the discovered patterns/model and overfitting of the model to the train data. The basic principle of feature subset selection is to find the necessary and sufficient subset of features or attributes which results in simplification of the discovered knowledge model, better generalisation power, while at the same time the accuracy for classification tasks is not compromised.

Association rule mining, being one of the most popular techniques for discovering interesting associations among data objects, has also been utilized for the classification task, where it can contribute to discovering strong associations between occurring attribute and class values [26]. An associative classification framework was first proposed in [28], which consists of an algorithm to generate all class association rules from which a classifier is constructed. Many works [10, 45, 49, 50] have developed various extensions and refinements to this initially proposed framework and the results reported high accuracy and efficiency for the classification problem. Similarly in tree-structured data domain, the XRules structural classifier [52], is based on association rules generated from the ordered embedded subtree mining algorithm [51].

When dealing with pattern selection, one faces the quantity problem due to large volume of output as well as the quality assurance problem of rules reflecting real, significant associations in the domain under investigation [25]. In a recent work presented in [24] the search space of Apriori-like algorithms is pruned so that discovered rules are interesting with respect to the Jaccard measure, rather than the support constraint for which an optimal threshold is often unknown. To deal with the quality problem many interestingness measures have been developed and utilized in various knowledge discovery tasks [12, 29]. In one train of thought, since the nature of data mining techniques is data-driven, the generated rules can often be effectively validated by a statistical methodology in order for them to be useful in practice [13, 22]. Interesting rules could then be interpreted as those rules that have a sound statistical basis and are neither redundant nor contradictory. The aforementioned works [12, 13, 22, 29] have mainly focused on relational data. There is relatively less work in this area when it comes to tree-structured data (an overview is given in the next section). Tree-structured data has underlying complex structural characteristics which typically need to be preserved in the knowledge patterns discovered during a data mining task [17, 52]. The structural characteristics of data pose difficulties in application of traditional classifiers and interestingness measures, whose mechanism typically does not take structural aspects of data into account.

In [38], a unified framework was proposed that systematically combines several statistical/heuristic techniques to assess the rule quality and remove any redundant and unnecessary rules for the classification problem. In this chapter, the focus is on

exploring the application of this framework to tree-structured data, enabled by the recently proposed structure-preserving flat data format for tree-structured data [14]. The work presented in [14] is based on the extraction of a *database structure model* (henceforth DSM) within which every tree instance from the database can be matched to and which keeps the structural information of the flat representation generated. The implications of the representation in contrast to traditional tree mining field is that every subtree pattern or a rule, will be constrained by the pre-order position of the constituting tree nodes of the subtree w.r.t the DSM. In this work, we explore the application of a feature subset selection measure and statistical interestingness measures via this method to filter out unnecessary and irrelevant subtree patterns for the structural classification task. A feature subset selection method is used prior to association rule generation. Once the initial set of rules is obtained, irrelevant rules are determined as those that are comprised of attributes not determined to be statistically significant for the classification task. The experiments are performed using real world web access tree dataset and a property management dataset from a real estate company. The results indicate that where the dataset has more standard structure the use of statistical measures will discard a large number of insignificant rules and at the same time increase the accuracy of the rule sets. On the other extreme, where the tree instances can vary greatly in terms of structure and label distribution among nodes, as is the case in the web access tree dataset, while many rules are removed and the accuracy increases, there is a significant reduction in coverage rate of the rule set. Furthermore, we compare some of the results with a structural classifier based on traditional subtrees, and highlight some important differences and implications when subtree based rules are constrained by their position. The results also show that including the associations that do not necessary result in connected trees can be important, while such associations are typically ignored within the tree mining field. These findings indicate that structural classifier could be improved and complemented by including disconnected subtrees and constraining the subtrees by their exact occurrence in the database. However, more work is required to identify the domains and application where including such association rules can be beneficial and the right way to combine them with traditional subtree patterns for optimal performance.

The rest of the chapter is organized as follows. The related works are given in Sect. 10.2, while Sect. 10.3 defines the concepts and the rule set optimization problem focused on in this study. In Sect. 10.4, we describe the steps involved in the proposed approach which is evaluated using real-world datasets and experimental findings are discussed in Sect. 10.5. Section 10.6 concludes the chapter.

## 10.2 Related Work

To date, limited work has been done on the feature selection, rule evaluation and interestingness measures for tree-structured data. Many of the well developed rule interestingness measures are in relational data and they have had great success in

evaluating rule interestingness as discussed in [44]. Several works on the evaluation of discovered patterns based on statistical significance [2, 22, 46] are limited to relational data. The existence of vast well-developed measuring techniques to evaluate interestingness of rules from relational data, offers great opportunities for adapting these techniques for verifying significant subtrees from semi-structured data. The applicability of these interestingness measures needs to be explored in the context of frequent subtree mining, where necessary adjustments and extensions need to be made to ascertain the validity of the methods given the more complex structured aspects in the data, which often need to be preserved in the rules.

One line of work focusing on more interesting subtree patterns aims to reduce the patterns through the application of plausible constraints. The problem of mining mutually dependent ordered subtrees has been addressed in [32]. The proposed algorithm utilizes the hyperclique method [47] in the tree mining context so that all the components of a subtree are highly correlated together. These hyperclique subtree patterns are discovered using an h-confidence measure which is the minimum probability of an item from a pattern in one transaction implying the presence of all other items in the same transaction. Hence, the extracted hyperclique subtree patterns will satisfy the minimum h-confidence threshold. The work done in [3] uses the method proposed for database compression in regards to item set mining in [39] to demonstrate how the same minimum description length principle can yield good results for sequential and tree-structured data. The work presented in [31] extends the idea of the item constraint [41] to that of a node-inclusion constraint in subtrees. Furthermore, Knijf and Feelders [20] proposed the use of monotone constraints in frequent subtree mining, namely monotone, anti-monotone, convertible and succinct constraints. Using these constraints, the frequent subtrees are mined using an opportunistic pruning strategy, and the set of frequent subtrees are reduced to only those satisfying the specific user pre-defined constraints.

Besides the aforementioned constraint-based techniques, to the best of our knowledge, there are limited works on verifying the significance of discovered frequent subtrees. Hashimoto et al. [19] proposed and developed an application of statistical hypothesis testing to re-rank the significant frequent subtrees. This approach ranks the significant patterns according to P-values obtained from the Fisher's Exact test of significance. The significant patterns were then used for Glycan classifications problems. Recently Yan et al. [48], proposed a mining framework called LEAP (Descending Leap Mine) for checking and mining significant frequent subgraphs which helps to discard redundant frequent subgraphs. For a predefined class label in XML documents, an efficient XRules classifier has been proposed in [52]. This approach offers promising results in terms of a structural classifier for semi-structured data, but utilizes standard measures of interestingness based on support and confidence.

### 10.2.1 Relationship Between Feature Subset Selection and Frequent Subtree Interestingness

In general, the objective of feature subset selection as defined in [18] is to find a minimum set of attributes such that the resulting probability distribution of the data classes is as close as possible to the original distribution obtained using all attributes. Han and Kamber in [18] asserted that domain expertise can be employed in order to pick out useful attributes. However, because the data mining task involves a large volume of data and unpredictable behaviour of data during data mining, this task is often expensive and time consuming.

The test of statistical significance is one of the prominent approaches in evaluating attributes/features usefulness. Stepwise forward selection, stepwise backward selection and a combination of both are three commonly used heuristic techniques utilized in statistical significance tests such as linear regression and logistic regression [18]. Moreover, the application of correlation analysis such as the chi-square test is also valuable in identifying redundant variables for feature subset selection. Another powerful technique for this purpose is the Symmetrical Tau [54], which is a statistical-heuristic feature selection criterion. It measures the capability of an attribute in predicting the class of another attribute. Additionally, information gain is another attributes relevance analysis method employed in the popular ID3 [33] and C4.5 [34] as reported in [18]. An extensive overview and comparison of the different approaches to the feature subset selection problem has been provided in [6, 11, 21, 30].

While the original purpose of feature subset selection is to reduce the number of attributes to only those attributes relevant for a certain data mining task, they nevertheless can be utilized to measure the interestingness of rules/pattern generated. For example, if the rule/pattern consists of irrelevant attributes, the aforementioned measure can also give some indication that the rule/pattern is not interesting. Moreover, [12] stated that there are three roles of interestingness measures. The first is their ability to discard uninteresting patterns during the mining process, thereby narrowing the search space and improving the mining efficiency. The second role is to calculate the interestingness scores for each pattern, which allows the ranking of patterns according to specific needs. The final role is the use of interestingness measures during the post-processing stage to select interesting patterns. Interestingness measures such as the chi-square test [8], Symmetrical Tau [54] and Mutual Information [44], are capable of measuring the interestingness of rules and at the same time identifying useful features for frequent patterns.

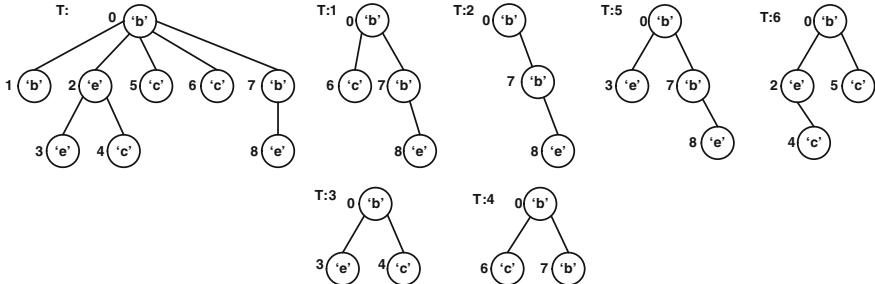
Since frequent patterns are generated based solely on frequency without considering their predictive power, the use of frequent patterns without selecting appropriate features will still result in a huge feature space which leads to larger volume and complexity of rules. This might not only slow down the model learning process, but even worse, the classification accuracy deteriorates (another kind of overfitting issue since the features are numerous) [9].

### 10.3 Problem Background

The problem of finding association rules  $x \rightarrow y$  was first introduced in [1] as a data mining task for finding frequently co-occurring items in large databases. Let  $I = \{i_1, i_2, \dots, i_{|I|}\}$  be a set of items. Let  $D$  be a transactions database for which each record/transaction  $R$  is a set of items, such that  $R \subseteq I$ . An association rule is an implication of the form  $x \rightarrow y$  where  $x \subseteq I$  and  $y \subseteq I$  and  $x \cap y = \emptyset$ . The absolute support of a rule  $x \rightarrow y$  is the number of transactions that contain both  $x$  and  $y$ . Typically, the relative support is used, where given the support of rule  $x \rightarrow y$  (denoted as  $\sigma(x \rightarrow y)$ ) be  $s\%$ , there are  $s\%$  of transactions in  $D$  that contain items (itemsets)  $x$  and  $y$ . In other words, the probability  $P(x \cup y) = s\%$ . An itemset is frequent if it satisfies the user-specified minimum support threshold. The confidence of a rule  $x \rightarrow y$ , is the estimate of conditional probability of a transaction containing the consequent ( $y$ ) if the transaction contains the antecedent ( $x$ ), and is calculated as  $\sigma(x \rightarrow y)/\sigma(x)$ .

Association rule discovery finds all rules that satisfy specific constraints such as the minimum support and confidence threshold, as is the case in the Apriori algorithm [1]. When tree-structured data such as XML is in question, the underlying associations are tree-structured by nature. Thus, the pre-requisite for the discovery of (structural) association rules becomes the task of frequent subtree mining. A tree-structured document can be modeled as a rooted ordered labeled tree. A *rooted ordered labeled tree* can be denoted as  $T = (v_0, V, L, E)$ , where (1)  $V_0 \in V$  is the root vertex; (2)  $V$  is the set of vertices or nodes; (3)  $L$  is a labelling function that assigns a label  $L(v)$  to every vertex  $v \in V$ ; (4)  $E = \{(v_1, v_2) | v_1, v_2 \in V \text{ AND } v_1 \neq v_2\}$  is the set of edges in the tree, and (5) for each internal nodes, the children are ordered from left to right.

This problem is generally defined as: given a database of trees  $T_{db}$  and minimum support threshold  $\sigma$ , find all subtrees that occur at least  $\sigma$  times in  $T_{db}$ . Most commonly considered subtrees are induced and embedded. The formal definitions of induced and embedded subtrees are as follows [42]: Given a tree  $S = (vs_0, V_S, L_S, E_S)$  and tree  $T = (vt_0, V_T, L_T, E_T)$ ,  $S$  is an ordered *induced* subtree of  $T$  iff (1)  $V_S \subseteq V_T$ ; (2)  $L_S \subseteq L_T$ , and  $L_S(v) = L_T(v)$ ; (3)  $E_S \subseteq E_T$ ; (4) the left-to-right ordering of sibling nodes in the original tree is preserved. Moreover,  $S$  is an ordered *embedded* subtree of  $T$  iff (1)  $V_S \subseteq V_T$ ; (2)  $L_S \subseteq L_T$ , and  $L_S(v) = L_T(v)$ ; (3) if  $(v_1, v_2) \in E_S$  then  $parent(v_2) = v_1$  in  $S$  and  $v_1$  is ancestor of  $v_2$  in  $T$ , and (4) the left-to-right ordering of sibling nodes in the original tree is preserved. If  $S = (vs_0, V_S, L_S, E_S)$  is an embedded subtree of  $T = (vt_0, V_T, L_T, E_T)$ , and two vertices  $v_1 \in V_S$  and  $v_2 \in V_S$  form ancestor-descendant relationship, the *level of embedding* (LoE) [42], between  $v_1$  and  $v_2$ , denoted by  $\Delta(v_1, v_2)$ , is defined as the length of the path between  $v_1$  and  $v_2$  in  $T$ . Hence, a *maximum level of embedding constraint* (MaxLoE)  $M_\Delta$  can be imposed on the subtrees extracted from  $T$ , such that any two connected nodes in an embedded subtree of  $T$  will be connected in  $T$  by a path that has the maximum length of  $M_\Delta$ . Examples of induced and embedded subtree are given in Fig. 10.1 (the number on the left of the nodes indicate its pre-order position in the original tree  $T$ ).



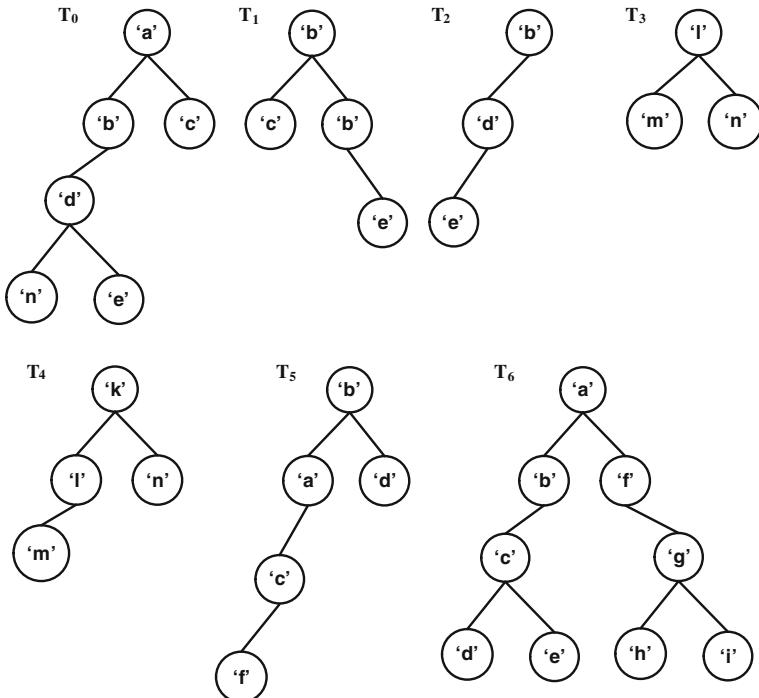
**Fig. 10.1** Example of induced/embedded subtrees ( $T_1, T_2, T_4, T_6$ ) and embedded subtrees ( $T_3, T_5$ ) of tree  $T$

In this chapter, the focus is on evaluating rules based on embedded and induced subtrees that satisfy minimum support and confidence thresholds, and discarding any rules determined to be irrelevant to the classification task at hand. Let us denote the subtree patterns from the frequent subtree set  $SF$  that have a class label (value), as  $SFC$ , their accuracy as  $ac(SFC)$  and coverage rate as  $cr(SFC)$ . The problem focused on in this work can then be generally defined as follows: Given  $SFC$  with accuracy  $ac(SFC)$ , obtain  $SFC' \subseteq SFC$ , such that  $ac(SFC') \geq (ac(SFC) - \varepsilon)$  and  $cr(SFC') \geq (cr(SFC) - \varepsilon)$  ( $\varepsilon$  is an arbitrary user defined small value used to reflect the noise that is often present in real-world data).

In what follows we discuss the common way of representing trees. This will lay the necessary ground for understanding the positional constraint imposed by the DSM approach [14]. A pre-order traversal can be computed as follows: If an ordered tree  $T$  consists only of a root node  $r$ , then  $r$  is the pre-order traversal of  $T$ . Otherwise let  $T_1, T_2, \dots, T_n$  be the subtrees occurring at  $r$  from left to right in  $T$ . The pre-order traversal begins by visiting  $r$  and then traversing all the remaining subtrees in pre-order starting from  $T_1$  and finishing with  $T_n$ . The string encoding ( $\varphi$ ) can be generated by adding vertex labels in the pre-order traversal of a tree  $T = (v_0, V, L, E)$  and appending a backtrack symbol (e.g., ' $'$ , ' $'$   $\notin L$ ) whenever we backtrack from a child node to its parent node. Figure 10.2 and Table 10.1 depict a tree database consisting of 7 tree instances (or transactions) and the string encoding for tree database, respectively.

### 10.3.1 Feature Subset Selection

Feature subset selection is an important pre-processing step in the data mining process. If the irrelevant attributes are left in the dataset, they can interfere with the data mining process and the quality of the discovered patterns may deteriorate, creating problems such as overfitting [9]. It is in particular the case in associative classifiers, since frequent patterns are typically generated without considering their predictive power [9], resulting in a huge feature space for possible frequent patterns. The removal of irrelevant attributes will result in a much smaller dataset, thereby



**Fig. 10.2** Example of a tree-structured database  $T_{db}$  consisting of 7 transactions

**Table 10.1** Example of tree transactions

Tree database ( $T_{db}$ )	Pre-order string encoding
$T_0$	'a b d n -1 e -1 -1 -1 c -1'
$T_1$	'b c -1 b e -1 -1'
$T_2$	'b d e -1 -1'
$T_3$	'l m -1 n -1'
$T_4$	'k l m -1 -1 n -1'
$T_5$	'b a c f -1 -1 -1 d -1'
$T_6$	'a b c d -1 e -1 -1 -1 f g h -1 i -1 -1 -1'

reducing the number of rules that need to be generated from the association rule mining algorithm, while closely maintaining the integrity of the original data [18]. Additionally, rules described with fewer attributes are also expected to perform better when classifying future cases; hence, they will have better generalization power than do the more specific rules that take many attributes into account. Besides, the patterns extracted will also be simpler and easier to analyse and understand. Determining the relevant and irrelevant attributes poses a great challenge to many data mining algorithms [36].

The feature subset selection problem can be more formally described as: Given a relational database  $D$ ,  $AT = \{at_1, at_2, \dots, at_{|AT|}\}$  the set of distinct items in  $D$ , and  $Y = \{y_1, y_2, \dots, y_{|Y|}\}$  the class attribute with a set of class labels in  $D$ . Let an association rule mining algorithm be denoted as  $AR_{AL}$ , the set of association rules for predicting the value of a class attribute  $Y$  from  $D$  extracted using  $AR_{AL}$  as  $AR(D)$ , and accuracy of  $AR(D)$  as  $ac(AR(D))$ . The problem of feature subset selection is to reduce  $D$  into  $D'$  such that  $AT' \subseteq AT$  and  $ac(AR(D')) \geq ac(AR(D)) - \varepsilon$ , where  $\varepsilon$  is an arbitrary user defined small value to reflect noise present in real-world data. In other words, the task is to find the optimal set of attributes,  $AT_{OPT} \subseteq AT$ , such that the accuracy of the association rule set using  $AR_{AL}$  is maximized.

### 10.3.2 Modeling Tree-Structured Data

An example of three user sessions logged into an academic institution website server is depicted here to represent the process of tree-structured data representation for data mining purposes. Table 10.3 is an example of a string to integer mapping from the user sessions in Table 10.2. The mapping process from string to integer can be done with a hash function as discussed in [51]. Representing a label as an integer instead of a string label has considerable performance and space advantages [42].

As mentioned earlier, a common way of representing trees is to use the pre-order (depth-first) string encoding ( $\varphi$ ) as described in [51]. For example, the pre-order string encoding representation of the underlying tree structure of the user navigation of Table 10.2 is transformed to  $(\varphi)(\text{session 0}) = '0 1 2 3 -1 4 -1 -1 5 6 -1 -1 -1 7 8 9 -1 -1 -1 10 11 -1 12 -1 -1'$  and  $(\varphi)(\text{session 1}) = '0 1 13 14 -1 15 -1 -1 16 -1 -1 17 -1'$  and  $(\varphi)(\text{session 2}) = '0 1 18 19 -1 -1 -1 20 21 -1 -1 7 22 -1 -1'$ . The access sequence of web pages from Table 10.2 can be represented in a tree-structured way as shown in Fig. 10.3. The order of pages accessed is reflected by the pre-order traversal of the tree. The corresponding tree structure is more informative than just a sequence of pages accessed as it captures the structure of the web site, and navigational patterns over this website, and the discovered knowledge patterns will as a result be more informative and useful, as already elaborated on in works such as [16, 17, 51, 52]. With this approach, specific pages can be considered within the same context. An example of this is the two pages being grouped under the ‘centres and labs’ parent node with label 13 in the tree of session 1, and 2 pages under the ‘research’ parent node with label 1 in the tree of session 0. Session 0 has come from an IP within the university and is most likely an example of a student acquiring some general information about the institute and then seeking information related to postgraduate study. The first session came from an IP internal to the university, where the user was interested in looking for jobs by browsing institutional centres and labs, and contacted the institute for more information. While session two may come from a potential external student who is searching for a potential supervisor by browsing some related conference papers and is interested in finding a research training program.

**Table 10.2** Example of user session

Session 1:
/
/research.html
/research/topics.html
/research/topics/51-business-intelligence.html
/research/topics/55-e-education-ecosystems.html
/research/seminars.html
/research/seminars/413-presentation-by-eric-feinberg.html
/phd-a-msc.html
/phd-a-msc/scholarships.html
/phd-a-msc/scholarships.html#debbi
/about.html
/about/objectives.html
/about/mission-and-vision.html
Session 2:
/
/research.html
/research/centres-and-labs.html
/centres-and-labs/217-anti-spam-research-lab-asrl.html
/centres-and-labs/214-centre-for-stringology-a-applications
-csa-.html
/research/jobs.html
/contact-us.html
Session3:
/
/research.html
/research/publications.html
/research/publications/conf-a-journal-papers.html
/allstaff.html
/allstaff/Research Professors & Fellows.html
/exchange-students.html
/phd-a-msc.html
/phd-a-msc/research-training.html

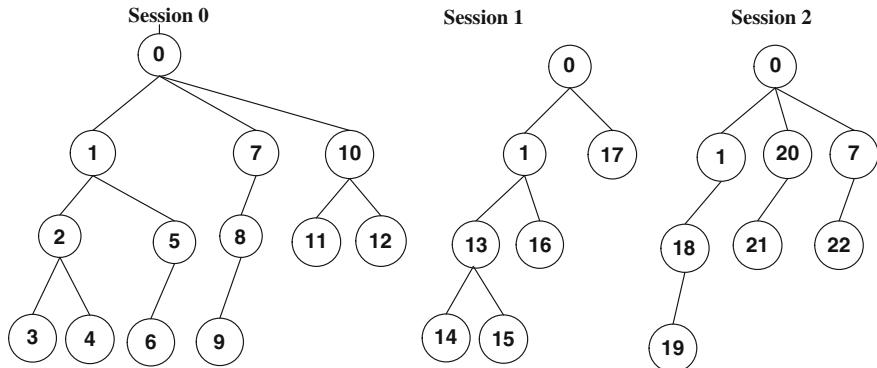
The integer-indexed tree is then formatted as shown in Table 10.4. This dataset format representation was proposed by [51]. Please note that the second column (*cid*) could be used to refer to a specific entity which the record describes (e.g. User id). However, in many domains such information is often unavailable, or it has been intentionally omitted or related through the transaction id (*tid*). Hence, in most of the tree databases represented in this format, the *cid* column will simple be a repetition of the *tid* column. This is the common format used in the frequent subtree mining field [17].

**Table 10.3** Integer mapping for web pages from Table 10.2

ID	Web page
0	Homepage
1	Research
2	Topics
3	51-business-intelligence
4	55-e-education-ecosystems
5	Seminars
6	413-presentation-by-eric-feinberg
7	phd-a-msc
8	Scholarships
9	scholarships.html#debii
10	About
11	Objectives
12	Mission-and-vision
13	Centres-and-labs
14	217-anti-spam-research-lab-asrl
15	214-centre-for-stringoLogsy-a-applications-csa-
16	Jobs
17	Contact-us
18	Publications
19	Conf-a-journal-papers
20	Allstaff
21	Research Professors & Fellows
22	Research-training

### 10.3.3 Database Structure Model (DSM)

The definition given by [14] is utilized here to describe the *Database Structure Model* (DSM). Generally, the string-like representation of a tree database (example given in Table 10.4), is converted into a flat data format while preserving the ancestor-descendant and sibling node relationships. Henceforth, this structure-preserving flat data representation will be simply referred to as ‘table’. The header of the table contains the DSM without any specific attribute names. It represents only the most general structure where every instance from the tree database can be matched to. This will ensure that when the labels of a particular transaction from the tree database are processed, they are placed in the correct column, corresponding to the position in the DSM that this label matches. To illustrate the complete conversion process using DSM, please refer to Fig. 10.2. Using the string encoding format representation [51], the tree database  $T_{db}$  from Fig. 10.2 would be represented as is shown in Table 10.1, where the left column corresponds to the transaction identifiers, and the right column is the string encoding of each subtree.



**Fig. 10.3** Integer-indexed tree of user sessions in Table 10.2

**Table 10.4** An integer-indexed tree in Fig. 10.3 formatted as a string-like representation as used in [51]

<i>tid</i>	<i>cid</i>	$ S $	Pre-order(depth-first) encoding
0	0	25	0 1 2 3 -1 4 -1 -1 5 6 -1 -1 -1 7 8 9 -1 -1 -1 10 11 -1 12 -1 -1
1	1	25	0 1 13 14 -1 15 -1 -1 16 -1 -1 17 -1
2	2	25	0 1 18 19 -1 -1 -1 20 21 -1 -1 7 22 -1 -1
-	-	-	-

*tid* transaction-id, *cid* omitted (i.e. equal to *tid*),  $|S|$  size of string

In this example, the DSM is reflected in the structure of  $T_6$  in Fig. 10.2 and it becomes the header of the table to reflect the attribute names as explained previously. The string encoding is used to represent this uniform structure and since the order of the nodes (and backtracks ('-1')) is important, the nodes and backtracks are labeled sequentially according to their occurrence in the string encoding. For nodes (labels in the string encoding),  $x_i$  is used as the attribute name, where  $i$  corresponds to the pre-order position of the node in the tree, while for backtracks,  $b_j$  is used as the attribute name, where  $j$  corresponds to the backtrack number in the string encoding. Hence, from our example in Fig. 10.2 and Table 10.1,  $(DSM) = 'x_0, x_1, x_2, x_3, b_0, x_4, b_1, b_2, b_3, x_5, x_6, x_7, b_4, x_8, b_5, b_6, b_7'$ .

To fill in the remaining rows, every transaction from  $T_{db}$  is scanned and when a label is encountered, it is placed in the matching column (i.e. under the matching node ( $x_i$ ) in the DSM), and when a backtrack ('-1') is encountered, a value '1' (or 'y') is placed in the matching column (i.e. matching backtrack ( $b_j$ ) in DSM). The remaining entries are assigned values of '0' (or 'No', indicating non existence). The flat data format of  $T_{db}$  from Table 10.1 (and Fig. 10.2) is illustrated in Table 10.5.

The conversion process can be formalized as follows. Let the tree database consisting of  $n$  transactions be denoted as  $T_{db} = \{tid_0, tid_1, \dots, tid_{n-1}\}$ , and let the string encoding of the tree instance at transaction  $tid_i$  be denoted as  $\varphi(tid_i)$ . The DSM is extracted from  $T_{db}$  using the procedure explained earlier. Further, let  $|\varphi(tid_i)|$  denote the number of elements in  $\varphi(tid_i)$ , and  $\varphi(tid_i)_k$  ( $k = \{0, 1, \dots, |\varphi(tid_i)| - 1\}$ ) denote

**Table 10.5** Flat representation of  $T_{db}$  in Fig. 10.2 and Table 10.1

$x_0$	$x_1$	$x_2$	$x_3$	$b_0$	$x_4$	$b_1$	$b_2$	$b_3$	$x_5$	$x_6$	$x_7$	$b_4$	$x_8$	$b_5$	$b_6$	$b_7$
a	b	d	n	1	e	1	1	1	c	0	0	0	0	0	0	1
b	c	0	0	0	0	0	0	1	b	e	0	0	0	0	1	1
b	d	e	0	0	0	0	1	1	0	0	0	0	0	0	0	0
l	m	0	0	0	0	0	0	1	n	0	0	0	0	0	0	1
k	l	m	0	0	0	0	1	1	n	0	0	0	0	0	0	1
b	a	c	f	1	0	0	1	1	d	0	0	0	0	0	0	1
a	b	c	d	1	e	1	1	1	f	g	h	1	i	1	1	1

**Table 10.6** Flat representation of  $T_{db}$  in Fig. 10.2 and Table 10.1 when minimum support = 3

$x_0$	$x_1$	$x_2$	$x_3$	$b_0$	$b_1$	$b_2$	$x_4$	$b_3$
a	b	c	n	1	1	1	c	1
b	c	0	0	0	0	1	b	1
b	d	e	0	0	1	1	0	0
l	m	0	0	0	0	1	n	1
k	l	m	0	0	1	1	n	1
b	a	c	f	1	1	1	d	1
a	b	c	d	1	1	1	f	1

the  $k$ th element (a label or a backtrack ‘-1’) of  $\varphi(tid_i)$ . The flat data format or table  $F_T(C, R)(C = \text{columns}, R = \text{rows})$  is set up where  $C = \{c_0, c_1, \dots, c_{m-1}\}$  ( $m = |C| = |\varphi(DSM)|$ ), and  $R = \{r_0, r_1, \dots, r_{p-1}\}$  ( $p = |R| = n + 1$ ) (i.e. extra column for attribute names). The value in column number  $x$  and row number  $y$  is denoted as  $F_T(c_x, r_y)$ . Hence, to set the attribute names  $F_T(c_i, r_0) = \varphi(DSM)_k$  where  $i = k = \{0, 1, \dots, (|\varphi(DSM)| - 1)\}$ .

In addition, during the conversion process as mentioned in [16], one can incorporate the minimum support threshold  $s$  so that the DSM captures only those structural characteristics that have occurred in at least  $s\%$  of the tree database. Hence, in some cases only a fraction of a tree instance can be matched to the DSM due to low occurrences in the tree database, but the partial information still needs to be included in the resulting flat table. As an example, refer to the tree database  $T_{db}$  in Table 10.5 and Fig. 10.2, in mining the subtrees with minimum support threshold of 3, the resulting DSM would be as follows: ‘ $x_0, x_1, x_2, x_3, b_0, b_1, b_2, x_4, b_3$ ’ and the new table is shown in Table 10.6.

#### 10.3.4 Tree to Flat Conversion Example Using Academic Institution WebLogs Data

Referring to the an Academic Institution WebLogs data example in Sect. 10.3.2, the pre-order encoding format of the tree database needs to be converted into a flat representation as proposed by [14]. The DSM applications were described earlier in

Sect. 10.3.3. In this section, an illustrative example is provided using an Academic Institution WebLogs example as reference. The DSM is reflected in the structure of  $T_0$  in Table 10.4 and the corresponding tree is shown in Fig. 10.3 (Session 0). Transaction  $T_0$  becomes the general structure of DSM and the header in Table 10.7 to reflect the attributes names. Every transaction that remains in the  $T_{db}$  will be matched against the DSM and every node label placed in the matching column (i.e. under the matching node ( $x_i$ ) in the DSM). The flat data format of  $T_{db}$  from Table 10.4 is illustrated in Table 10.7.

### 10.3.5 Representing Disconnected Trees w.r.t. DSM

As discussed earlier in Sect. 10.3.3, the rules from DSM can be converted into pre-order string encoding of the subtrees, and hence are represented as subtrees of the tree database. However, some rules may not be representatives of valid subtrees. For example, it is possible that some items in the rules correspond to sibling nodes in the original tree, while the parent or any ancestor node connecting those in the original tree is not present in the rules discovered using DSM approach. Hence, this would result in an invalid subtree as the nodes are disconnected. In addressing this matter, one can add the other nodes that make it into a valid subtree but flag them as irrelevant. The process consists of sequentially listing the values of each matched node in DSM, while retaining the level of embedding information of each current node in DSM and in the subtree pattern. Since the DSM itself is ordered according to the pre-order traversal, this results in pre-order string encodings of the subtrees.

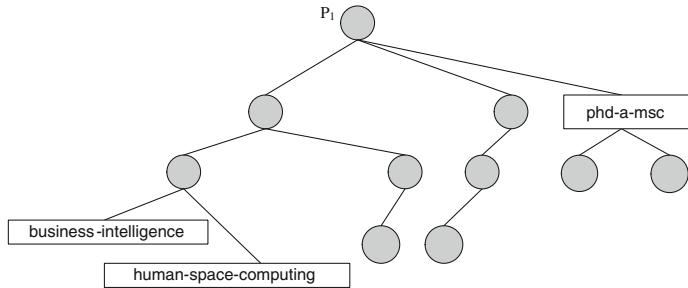
As a simple illustrative example, consider the following associations/patterns extracted from an Academic Institution WebLogs Data:

$P_1$ : business-intelligence human-space-computing phd-msc,

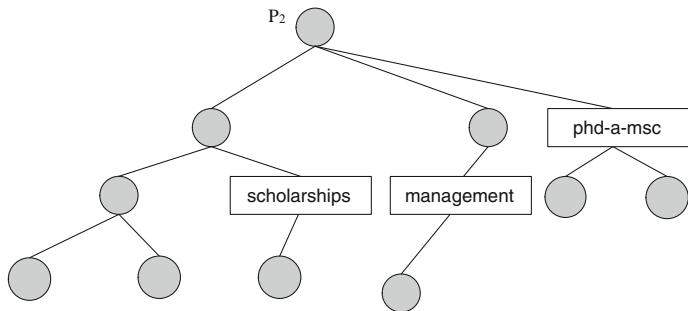
$P_2$ : scholarships management phd-a-msc.

With respect to pattern ( $P_1$ ) in Fig. 10.4 and pattern ( $P_2$ ) in Fig. 10.5, the items (nodes) in the rule correspond to sibling nodes in the original tree, while the parent or any ancestor node connecting those in the original tree is not present in the rule. Hence, this would result in an invalid subtree as the nodes are disconnected. This is illustrated in both Figs. 10.4 and 10.5, where irrelevant nodes are shaded grey. One can also choose to display the labels of nodes that are there to contextualize the information, i.e. scholarships and management and phd-a-msc, which would essentially contextualize the specific rule constraints. Additionally, the labels of nodes can be displayed in order to contextualize the information in the tree. In this work, these rules are recognized as FullTree rules.

**Table 10.7** Flat representation of an Academic Institution WebLogs  $T_{db}$  in Table 10.4



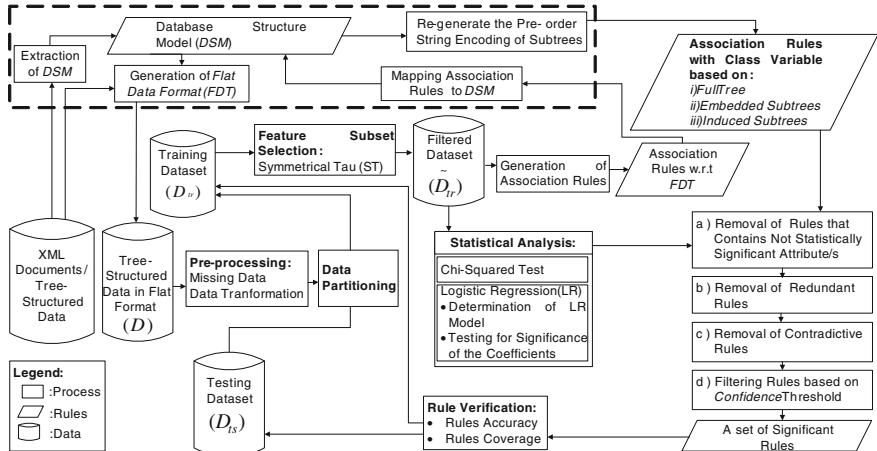
**Fig. 10.4** Displaying pattern ( $P_1$ ) w.r.t. DSM in Table 10.7



**Fig. 10.5** Displaying pattern ( $P_2$ ) w.r.t. DSM in Table 10.7

## 10.4 Method and Experimental Setup

The method used here is the integration of rule optimization framework presented in [37, 38] and structure-preserving flat representation of tree-structured data presented in [14], which as a result will allow direct application of standard statistical measures to tree-structured data. Figure 10.6 shows the proposed framework which in itself describes the experimental process. The database structure model (DSM) [14] is extracted from the tree-structured data/XML documents to preserve the structural characteristics of the data. The extracted DSM is used to create the flat representation of the tree structured data (shown in Fig. 10.6 with the square dash line region). An example of the conversion process is given in Sect. 10.3. Once the tree-structured dataset has been converted to a flat table format (FDT), the dataset is then divided into two parts. The first part is used for frequent pattern mining, statistical evaluation and rule filtering process, while the second part acts as sample data drawn from the dataset used to verify the accuracy and coverage of the discovered rules. In the pre-processing phase, missing values are handled using common distribution-based missing value imputation [27] and equal width binning approach is utilised to discretise the values of any continuous attributes. The equal-width binning approach groups the data into several buckets or bins of the same interval size. The equal width binning will be implemented based on the following steps [35]: (1) Calculate the range of variable



**Fig. 10.6** Method and experimental setup

to be binned; (2) Using the specified number of bins, calculate the boundary (width) of each bin; (3) Using specified boundaries, assign each value of the variable to a bin for each record. The data partitioning, missing value imputation and discretization were performed using the SAS Enterprise Miner software (please refer to [35] for further detail on the use of software for data pre-processing). Secondly, feature subset selection based on attribute ranking according to Symmetrical Tau measure [54] of predictive capability is performed as described in [15].

The association rule mining algorithm is utilized to discover frequent rules from the FDT and rule filtering process based on sequence of chi-square test, Logistic Regression model selection, redundant rule removal (based on minimum improvement redundant rule constraint [4]) and optional filtering based on higher confidence threshold is performed. The extracted association rules are mapped onto the DSM (by the pre-order position of each item) to re-generate the pre-order string encoding of subtrees, thereby representing them as subtrees of the tree database.

These rules may contain both valid and invalid subtrees (disconnected subtrees), and we will refer to these as *FullTree*. In addition, the rules based on embedded subtrees and the rules based on induced subtrees (the rule sets that exclude disconnected subtrees) have also been revealed within the extracted rules. Finally the rule accuracy and coverage rate is calculated for all rule sets at different stages. The extracted frequent rules are mapped onto the DSM to re-generate the pre-order string encoding of subtrees, thereby representing them as subtrees of the tree database.

**Tree-Structured Data Format Conversion:** For given tree-structured data, the enumeration of all possible subtrees in a complete, non-redundant and efficient way is the major problem one needs to tackle [43]. A significant delay in the subtree patterns analysis and interpretation process may occur at lower support thresholds. Additionally, as a large number of frequent subtree patterns may be discovered, many of which may not be useful, one needs to filter out many of the irrelevant/uninteresting patterns.

The flat data format (relational or vectorial data) was proven to be acceptable and successful when utilized with many well-established data mining techniques. Thus, an effective way proposed in [14] known as Database Structure Model (DSM) is utilized in this research to represent tree-structured data in a structure-preserving flat data format. This approach offers a way of preserving tree-structured and attribute-value information. With the application of DSM, the structural characteristics are preserved during the data mining process. The extracted rules from the data mining application can be mapped onto the DSM to re-generate the pre-order string encoding of subtrees.

Let a tree structure data in flat table format (FDT) dataset be denoted as  $D$ ,  $I = \{i_1, i_2, \dots, i_{|I|}\}$  the set of distinct items in  $D$ ,  $AT = \{at_1, at_2, \dots, at_{|AT|}\}$  the set of input attributes in  $D$ , and  $Y = \{y_1, y_2, \dots, y_{|Y|}\}$  the class attribute with a set of class labels in  $D$ . Assume that  $D$  contains a set of  $n$  records  $D = \{x_r, y_r\}_{r=1}^n$ , where  $x_r \subseteq I$  is an item or a set of items and  $y_r \in Y$  is a class label, then  $|x_r| = |AT|$  and  $x_r = \{at_1val_r, at_2val_r, \dots, at_{|AT|}val_r\}$  contains the attribute names and corresponding values for record  $r$  in  $D$  for each attribute  $at$  in  $AT$ . The training dataset is denoted as  $D_{tr} \subseteq D$  and the testing dataset as  $D_{ts} \subseteq D$ , and filtered database after feature selection as  $D'_{tr}$  where  $I' \subseteq I$ .

We extracted the rule sets extracted from the flat table format (FDT) satisfying the minimum support and confidence threshold (denoted as  $F(A)$ ). Individual rules are denoted as  $fA \in F(A)$ , of the form  $x \rightarrow y$ , where  $x$  is the antecedent and  $y$  the consequent,  $\exists\{x_r, y_r\} \in D'_{tr}, x \subseteq x_r, x_r = \{at_1val_r, at_2val_r, \dots, at_{|AT|}val_r\}$  and  $y \in Y$  is a class label. For generating  $F(A)$ , SAS Enterprise Miner software was used.

*Feature Subset Selection:* The Symmetrical Tau (ST) measure [54] was derived from the Goodman's and Kruskal's Asymmetrical Tau measure of association for cross-classification tasks in the statistical domain. Zhou and Dillon [54] have used the Asymmetrical Tau measure as feature selection during decision tree building, and have found that it tends to favour attributes with more values. When the classes of an attribute A are increased by class subdivision, more is known about attribute A and the probability error in predicting the class of another attribute B may decrease. On the other hand, attribute A becomes more complex, potentially causing an increase in the probability error in predicting its category according to the category of B. This trade off effect inspired Zhou and Dillon [54] to combine the two asymmetrical measures in order to obtain a balanced feature selection criterion which is in turn symmetrical. However, note that in case of Boolean variables, symmetrical and asymmetrical tau will have the same value. Some powerful properties of ST, as reported in [54], are noise handling through built-in statistical strength, potential classification uncertainties are conveyed through dynamic error estimation, no bias towards multi-valued attributes, not proportional to sample size, proportional-reduction-in-error nature allows measuring of sequential variation in predictive capability, and handling of Boolean combinations of logical features.

Let there be  $R$  rows and  $C$  columns in the contingency table for attributes  $at_i$  and  $Y$ . The probability that an individual belongs to row category  $r$  and column category  $c$  is represented as  $P(rc)$ , and  $P(r+)$  and  $P(+c)$  are the marginal probabilities in row category  $r$  and column category  $c$  respectively. The measure is

based on the probabilities of one attribute value occurring together with the value of the second attribute, and for the classification task the second attribute will correspond to a special attribute in the dataset defined as class. The ST measure for the capability of input attribute  $at_i$  in predicting the class attribute  $Y$  is defined in [54] as follows.

$$\text{Tau}(at_i, Y) = \frac{\sum_{c=1}^C \sum_{r=1}^R \frac{P(rc)^2}{P(+c)} + \sum_{r=1}^R \sum_{c=1}^C \frac{P(rc)^2}{P(r+)} - \sum_{r=1}^R P(r+)^2 - \sum_{c=1}^C P(+c)^2}{2 - \sum_{r=1}^R P(r+)^2 - \sum_{c=1}^C P(+c)^2} \quad (10.1)$$

The higher values of the ST measure would indicate better discriminating criteria (features) for the class that is to be predicted in the domain. As performed in [15], the attributes are ranked according to their decreasing ST values and a relevance cut-off point is chosen at and below which all attributes are considered as irrelevant and are discarded. The relevance cut-off was selected based on the significant difference (less than half of the previous value in the ranking) between the ST values in decreasing order. This will prevent the generation of rules which would then need to be discarded when found that they were comprised of some irrelevant attributes. In accordance with [5] we have found that mutual information typically ranks attributes with more values higher than the ST measure does.

*Chi-square:* A natural way to express the dependence between antecedent and the consequence of an association rule is the correlation based on the chi-square test for independence [7]. The chi-square test is defined as follows: For a given  $D'_{tr}$ , the occurrence of  $at_i$  where  $at_i \in AT$ , ( $i = (1, \dots, |AT|)$ ) is independent of the occurrence of  $y_r \in Y$  if  $P(at_i \cup y_r) = P(at_i)P(y_r)$ ; otherwise  $at_i$  and  $y_r$  are dependent and correlated. The correlation between  $at_i$  and  $y_r \in Y$  is measured using Eq. 10.2. For a given lift measure [40] based on Eq. 10.2, the chi-square  $\chi^2$  statistic value was utilised to determine whether the correlation is statistically significant.

$$\text{lift}(at_i, y_r) = \frac{P(at_i \cup y_r)}{P(at_i)P(y_r)} \quad (10.2)$$

Hence, the chi-square test discards any  $fA_k \in F(A)$  for which  $\exists at_i$  contained in  $x$  of  $x \rightarrow y$ , the  $\chi^2$  value is not significant for  $y \in Y$  (correlation analysis in Eq. 10.2).

*Logistic Regression:* Another form of statistical analysis applied was the logistic regression. The relationship between the antecedent and consequent in association rule mining can be presented as a relationship between a target variable and the input variables in logistic regression. The following is the definition of the logistic regression model involved in the framework. For a given  $D'_{tr}$ , several logistic regression models were developed based on  $\ln(Y) = \beta_0 + \beta_1 at_1 + \beta_2 at_2 + \dots + \beta_{|AT|} at_{|AT|} + e$ , where  $\ln(Y)$  is the natural logarithm of the odds ratio,  $\beta_0, \beta_1, \dots, \beta_{|AT|}$  are the coefficients of the input attributes  $at_i$ ,  $e$  is the error variable and  $Y$  the dichotomous class attribute. The coefficient  $\beta_i$  of  $at_i$  is determined based on the log likelihood value given in Eq. 10.3, where  $at_i val_r$  denotes the value of attribute  $at_i$  occurring in record  $r$ .

$$\beta_i at_i = \sum_{r=1}^n \{y_r \ln[\pi(at_i val_r)] + (1 - y_r) \ln[1 - \pi(at_i val_r)]\} \quad (10.3)$$

The statistical hypothesis is then used to determine whether the input attributes are significantly related to the class attribute. A number of models can be developed from logistic regression analysis, and each produces a different selection of attributes. The model that fits the data well and has the highest predictive capability is selected. Hence, logistic regression is used to discard any  $fA_k \in F(A), fB_k \in F(B), fC_k \in F(C)$  for which  $\exists at_i$  contained in  $x$  of  $x \rightarrow y$ , the  $\beta_i at_i$  value is not significant towards the class attribute  $Y$  (logistic regression analysis in Eq. 10.3).

*Redundant and Contradictive Rule Removal:* To remove redundant rules, we utilize the concept of productive rules [4]. This approach is based on minimum improvement redundant rule constraint [4], which discards any rule  $x \rightarrow y$  if confidence  $(x \rightarrow y) \leq \max(\text{confidence}(z \rightarrow y)) \forall z \subset x$ . In other words, a rule  $x \rightarrow y$  with confidence value  $c1$  is considered as redundant if there exists another rule  $z \rightarrow y$  with confidence value  $c2$ , where  $z \subset x$  and  $c1 \leq c2$ . The contradictory rule constraint [53] is then utilised to discard two or more rules that have the same precedent but imply a different class value.

*Rules Accuracy and Rules Coverage:* A measure needs to be applied to verify whether the removal of a large volume of rules based on statistical analysis, and redundancy and contradictory assessment methods, will enable the discovery of all the interesting and significant subtree patterns. As such, the quality of the subtree pattern will be demonstrated based on their accuracy and coverage values. The values for rule accuracy and coverage will be measured at every stage and sequence of this task. This measure is crucial as it can determine the quality of the discovered rules. Additionally, this analysis will reveal the balancing/optimization issues with regards to the trade-off between accuracy rate and coverage rate.

## 10.5 Experimental Evaluation

In this section we present the experiments performed using the CRM dataset (real estate property management records in XML), CSLOGS dataset (web access trees) and an academic institution dataset (web access trees), structural characteristics of which are shown in Table 10.8, and the following notation is used:  $|Tr|$ —Number of transactions (independent tree instances);  $|L|$ —Number of unique labels;  $|T|$ —Number of nodes (size) in a transaction;  $|D|$ —Depth;  $|F|$ —Fan-out-factor (or degree). Please note, that in [52] where the structural/XML classification was first proposed, it was demonstrated that a simpler classifier that does not take the structure into the account cannot achieve equally good results. Similarly, in [51] it was empirically shown, that tree-structured web-browsing patterns are more informative and useful than, their itemset/sequential pattern counter part. Hence, this study is not repeated in this work, but rather an experimental study is presented on the use of

**Table 10.8** Structural characteristics of the data

	$ Tr $	$ L $	$\text{Avg} T $	$\text{Avg} D $	$\text{Avg} F $	$\text{Max} T $	$\text{Max} D $	$\text{Max} F $
<i>CRM</i>	1,181	10,611	52.97	4.89	8	533	5	46
<i>CSLOGS</i>	68,302	16,207	7.8	3.45	1.82	313	123	137
<i>Academic Institution Website</i>	18,836	34,052	9.63	4.98	1.56	60	59	37

standard statistical techniques to reduce the huge number of rules typically generated during frequent subtree mining, in the context of associative classification. As such, the focus is on the use of basic accuracy and coverage rate rule evaluation measures to observe the gradual difference in the rule set accuracy and coverage as different feature/rule filtering techniques are applied.

Each dataset underwent conversion into a structure-preserving flat data format (henceforth FDT) using the DSM approach. The backtrack attributes information was kept in DSM as this is important for preserving the structural information. Hence, this can be used to represent the resulting rules as trees/subtrees. The backtrack attributes can be optionally kept in the FDT as when present in rules, they indicate the existence/non-existence of a node irrespective of the label as discussed in [16]. We have compared the results when rules are generated from itemsets including the backtrack attributes and without, and the difference was not substantial to make it worth reporting. Inclusion of backtrack attributes typically resulted in slightly better results, in terms of increased rule set coverage rate and thus all experiments presented are done using this option. When reporting the results, the following notation will be used ST—Symmetrical Tau, AR—accuracy rate, CR—coverage rate, *FullTree*—the initial rule set containing disconnected subtree and backtrack attribute based rules, *Embedded*—after itemsets have been mapped to DSM (by pre-order positions) to generate valid connected subtrees, and *Induced*—only subtrees where maximum level of embedding is limited to 1 (i.e. parent-child relationships among the nodes, see Sect. 10.3).

### 10.5.1 Experiment Set 1—CRM Data

CRM data is a real-world dataset relating to the handling of complaints in the area of real estate. Each complaint relates to a particular defect in the property, and a property manager will assign a case to each defect, containing information such as case managers, contractors, areas of defect, district and building type. The classification problem considered corresponds to the “WorkCompletion”, with 2 possible values (within a month and more than a month duration. The attributes containing similar information or referring to work/task completion duration have then been removed. The dataset consists of 1,181 instances with 675 attributes, of which 66 % was used

**Table 10.9** Subtree association rule evaluation for CRM data

Type of analysis	Data partition	<i>FullTree</i>			<i>Induced</i>		
		# of Rules	AR %	CR %	# of Rules	AR %	CR %
# of Rules after ST	Training	27116	83.02	100	5270	81.56	100
	Testing		83.74	100		83.4	100
Logistic regression	Training	91	79.85	100	17	68.54	100
	Testing		80.95	100		70.57	100
Redundancy removal	Training	51	76.78	100	17	68.54	100
	Testing		77.72	100		70.57	100
Min. Conf. 60 %	Training	44	83.82	95.50	12	77.20	91.53
	Testing		84.57	96.15		79.18	93.59

for training and 34 % for testing. However, there are many complex classes within this CRM data which may interest the users of the data. Nevertheless in this case, as our main purposes is not to analyse the problem of CRM itself, but to look at the CRM data as an example of tree-structured data, the attention is confined to the aforementioned class. The resulting DSM based flat data format contains 675 attributes (including the class), 586 selected attributes based on Symmetrical Tau(ST) feature selection. The rules are then generated based on support of 5 % and confidence of 50 %. Note that initially the dataset with backtrack attributes was used, which caused memory issues in the SAS software and hence we applied the ST feature selection prior to generating association rules which removed all of the backtrack attributes in this dataset. Furthermore, for this dataset, all subtrees generated are of induced type, and hence we do not report any results for the embedded subtree variation as it is identical to induced for this data.

Table 10.9 shows the results as the statistical analysis and the redundancy assessment have progressively been utilized to evaluate the interestingness of rules. Note that chi-square analysis is not presented as it did not result in any rule removal at that stage, and all of the connected subtrees were of induced subtree type in this dataset. As one can see a significant number of rules was removed by applying the logistics regression analysis, and in *FullTree* rule set further 40 rules were detected as redundant. This has reduced the AR % by about 3 %, but after rules whose minimum confidence is below 60 % have been removed (last row) the accuracy has increased with the cost of not covering around 5 % of the instances. In this experiment, *FullTree* rule set is the most optimal one, as it is not only more accurate in classifying/predicting specific instances in the database, but also achieves a higher coverage rate in the final step compared to *Induced* rule set. The *FullTree* rule set can contain rules that do not convert to valid (connected) subtrees when matched to DSM. Nevertheless, these are important to include as they may represent important associations that should not be lost because they do not convert to connected valid subtrees. Note that we have tried to run the XRules structural classifier [52] on this data, but since there are quite a few repeating node labels in single tree instances, caused by repetition of defects and individual cases within a single record, the tree

**Table 10.10** CSLOGS flattened data characteristics and initial number of rules for varying support

Support threshold (%)	Attr. # DSM flat	# Selected attr. Sym. Tau	# of Rules with target attr.		
			FullTree	Embedded	Induced
1	222	217	13835	13833	13809
5	64	52	920	919	918
10	40	29	216	215	215
20	24	11	48	47	47
30	16	7	32	31	31

mining algorithm [51] on which the XRules is based on, has difficulties in extracting subtrees at required low support thresholds.

### 10.5.2 Experiment Set 2—CSLOGS Data

The CSLogs data comprises the web access trees from the computer science department of the Rensselaer Polytechnic Institute previously used in [52] to evaluate the XRules structural classifier. All of the three datasets (US1924, US2430, and US304) were combined and instances were replicated (in both training and test data) to make the class distribution even. The tree instances are labelled according to two classes, namely the internal and external web site access. The total number of combined instances is 68302. The training set was comprised of 66 % of the data and the remainder was left as the test set. Since different support thresholds were used, in our approach the flat data representation of the dataset is done separately for each support threshold, as the extracted database structure model (DSM) varies; hence, the number of attributes used during frequent pattern generation. The general characteristics of the flat data format (including backtrack attributes) and initial number of rules extracted for CSLogs data (50 % minimum confidence) at varying support thresholds is provided in Table 10.10. Note, that when using the association rules for classification task it is natural that performance will vary depending on the support threshold used. Hence, different support thresholds were tried from a larger to a smaller extreme, and as expected for larger support thresholds there will be a trade-off for limited coverage as only the very frequent subtrees will be extracted to form part of the model.

For this dataset, the best results were achieved for the lowest examined support threshold of 1 %, and detailed results of progressively filtered rules based on statistical analysis and redundancy removal are presented in Table 10.11 for support 1 % (at the end of this subsection we present the performance of final rule sets for all the support thresholds). The number of rules are shown in brackets below each AR and CR values reported. The results reveal that by selecting important input attributes with ST and evaluating the rules with statistical analysis and redundancy assessment method,

**Table 10.11** Subtree association rule evaluation for CSLOG data (1% support 50% confidence)

Type of analysis	Data partition	<i>FullTree</i>		<i>Embedded</i>		<i>Induced</i>	
		AR %	CR %	AR %	CR %	AR %	CR %
Initial rules	Training	68.09 (13835)	98.59 (13835)	68.12 (13834)	98.59 (13834)	68.11 (13810)	98.59 (13810)
	Testing	69.94	98.6	69.94	98.6	69.94	98.6
Rules after ST	Training	69.94 (6084)	98.59 (6084)	70.02 (6083)	98.59 (6083)	70.02 (6081)	98.59 (6081)
	Testing	72.01	98.6	72.1	98.6	72.1	98.6
Chi-Square	Training	79.22 (73)	48.97 (73)	79.02 (72)	48.39 (72)	78.41 (65)	48.39 (65)
	Testing	78.78	48.77	78.57	48.25	78.06	48.25
Logistic regression	Training	79.22 (73)	48.97 (73)	79.02 (71)	48.39 (71)	78.41 (64)	48.39 (64)
	Testing	78.78	48.77	78.57	48.25	78.06	48.25
Redundancy removal	Training	79.02 (61)	48.97 (61)	78.71 (54)	48.97 (54)	78.71 (54)	48.97 (54)
	Testing	78.53	48.77	78.53	48.77	78.53	48.77

there is a significant reduction in the number of rules. While an increase in AR can be observed, this is at the cost of reduced CR capabilities. The characteristics of the *FullTree* rule set are similar to those of the *Embedded* and *Induced* rule sets, and the AR and CR are very similar or the same for the different rule sets. This is because the rules from *Embedded* and *Induced* rule sets are subsets of *FullTree*, and in this dataset there were not so many variations among the rule sets w.r.t the level of embedding in subtrees or frequent patterns that produce disconnected subtrees. To conclude, the increase in prediction/classification accuracy comes with a trade-off since fewer instances are captured from the datasets. On the positive side, a smaller number of rules is expected to have better generalization power and are easier for the user to understand and utilize for decision support purposes.

**Comparison with XRules for varying support thresholds.** In Table 10.12 we compare the AR and CR of the final rule sets of *FullTree* with XRules approach for varying support thresholds. Note that the approaches are fairly different in terms of the rule filtering performed in the process. Nevertheless, the comparison performed serves mainly as a benchmark for the kind of accuracy and coverage rate that is to be obtained when basing the classification on frequent patterns/subtrees extracted using the support and confidence thresholds. As such, in no way do the results indicate that one approach performs better than the other, as the internal mechanism is rather incompatible. The XRules approach is based on the TreeMiner [51] algorithm for extracting ordered embedded subtrees, and hence the number of rules extractd at varying support thresholds is larger (shown in brackets), since the likelihood that a subtree will be frequent when it does not need to occur at the same position is much

**Table 10.12** Comparison of rules accuracy and coverage rate for CSLogs data using the XRules and *FullTree* final rule set

Support	1 %		5 %		10 %	
	AR %	CR %	AR %	CR %	AR %	CR %
XRules	72.72 (298)	66.04 (298)	61.74 (20)	40.7 (20)	56.9 (3)	23.21 (3)
	78.53 (61)	48.77 (61)	78.73 (4)	20.35 (4)	76.9 (2)	20.3 (2)

**Table 10.13** Rule sets at support 10 %

#	XRules	#	<i>FullTree</i>
1	$1 \rightarrow \text{Class}(0)$	1	$\text{X1}(1) \rightarrow \text{Class}(0)$
2	$12811 \rightarrow \text{Class}(1)$	2	$\text{X1}(12811) \rightarrow \text{Class}(1)$
3	$6 \rightarrow \text{Class}(0)$		

higher. On this note, the rule sets of the XRules approach will typically have higher coverage rate, especially in the CSLOGS dataset, where subtrees do in fact occur at many different positions due to variations in website navigation. However, one can see that this is at times at a cost of reduction in AR, and constraining the subtrees by position could be seen as more precise, but naturally would cover less cases. To give a simple example, please refer to Table 10.13 where we show rule sets for the support value of 10 %. One can observe that the *FullTree* rule set does not contain a rule that corresponds to rule number 3 in XRules even though it was considered frequent by XRules. The reason for this is that the particular node with label “6” with “Class(0)”, where “6” occurs at the same node/position in DSM did not occur in 10 % of the instances to be considered frequent and part of the *FullTree* rule set. The two matching rules correspond to the first page accessed during the site navigation session, as it is labelled with pre-order position 1, namely X1 in our approach (note that X0 is a virtual node in the CSLOGS dataset always labelled with 0 and is removed in both approaches). For support threshold of 20 and 30 % no rules were extracted in our approach, while XRules only had the single default rule for majority class.

### 10.5.3 Experiment Set 3—Academic Institution Web Log Data

Academic Institution WebLogs data is an apache2 (v2.2.3) web server logs files. The WebLogs data was initially used in [16] in utilizing the DSM application. For the purpose of the work in this research, the similar setting of the WebLogs data as described in [16] has been utilized. The data was collected for a four-month period in its native (default) format. During this period, all access to the website was stored in logs files, while messages stored in the normal error message logs were excluded. The access to the website was then classified as “internal” (within the university) and “external” (outside the university). The grouped user sessions were converted

**Table 10.14** Academic Institution flattened data characteristics and initial number of rules for varying support

Support threshold (%)	Attr. # DSM flat	# Selected attr. Sym. Tau	# of Rules with target attr.		
			FullTree	Embedded	Induced
1	442	217	–	–	–
5	126	123	28282	28282	28282
10	70	63	234	234	234
20	36	29	50	49	49
30	26	19	14	13	13

to trees as was explained with the illustrative example in Sect. 3.1. The resulting dataset had 18,836 instances, of which 66 % was used for training and the remainder for testing. The details of the setting of the WebLogs access can be found in [16]. The general characteristics of the flat data format (including backtrack attributes) and initial number of rules extracted for education institution data (50 % minimum confidence) at varying support thresholds is provided in Table 10.14.

In this dataset, similar to the experiments described in Sect. 10.5.2, rules from *FullTree*, Embedded and Induced rule sets have been progressively assessed with statistical analysis and redundancy assessment method. The results demonstrate that the conversion of the original tree-structured data into the flat data format representation, created a very large number of input attributes, especially at lower support thresholds. By utilizing the Apriori algorithm to generate all frequent rules, one might encounter difficulties in analyzing all rules given certain support and confidence constraints.

By referring to the Table 10.15, even with the given support constraint, the number of extracted rules (Initial Rule Set) is large. A large volume of rules may be discovered due to the presence of irrelevant attributes in the dataset. The capabilities of ST in selecting appropriate attributes, thereby removing irrelevant attributes, are shown in our previous experiments for relational data problems. For this particular task of evaluating tree-structured rules, similar experiments were conducted. The attributes for each different support were ranked according to their decreasing ST and a relevance cut-off point was chosen.

Table 10.15 indicates the differences between the number of initial input attributes and the number of attributes after applying Symmetrical Tau (ST) with their respective rule number (below) for each dataset for each different support. All attributes that have been removed from the WebLogs data are backtrack attributes. This indicates that the inclusion of these backtrack nodes may not be useful or have low capabilities in predicting the class attributes in this dataset. The input variable that contains a single value is unable to distinguish the class variables. Such input attributes have been discarded as they are considered irrelevant based on the ST value calculated. With the application of ST feature selection technique, rules that contain attributes that failed the ST measure are discarded. The large number of rules were managed to be reduced

**Table 10.15** Subtree association rule evaluation for Academic Institution data (10 % support 50 % confidence)

Type of analysis	Data partition	<i>FullTree</i>		<i>Embedded</i>		<i>Induced</i>	
		AR %	CR %	AR %	CR %	AR %	CR %
Initial Rules	Training	64.27 (232)	100.00 (232)	64.54 (232)	100.00 (232)	64.54 (232)	100.00 (232)
	Testing	70.06 (232)	100.00 (232)	70.55 (232)	100.00 (232)	64.54 (232)	100.00 (232)
Rules after ST	Training	75.19 (43)	73.95 (43)	74.94 (43)	73.95 (43)	74.94 (43)	73.95 (43)
	Testing	74.94 (43)	74.09 (43)	74.84 (43)	74.09 (43)	74.84 (43)	74.09 (43)
Chi-Square	Training	78.21 (11)	64.47 (11)	77.56 (10)	64.47 (10)	77.56 (10)	64.47 (10)
	Testing	74.96 (11)	60.12 (11)	74.58 (10)	61.02 (10)	74.58 (10)	61.02 (10)

with a proper sequence of usage of parameters including the ST feature selection, statistical analysis and the redundancy assessment method. According to Table 10.15, with the reduction of number of rules for *FullTree*, Embedded and Induced rule sets for Academic Institution Weblogs (10 % Support) the AR are increased but at the cost of a decrease in CR. One can also notice that the AR for the *FullTree* rule set is initially slightly lower than the AR of the Embedded and Induced rule set, but after Symmetrical Tau is applied, the accuracy of *FullTree* is higher and remains higher after chi-square rule filtering. Note that for this data there were no further rules removed via logistic regression and redundancy check, and hence these stages are not shown in Table 10.15.

## 10.6 Conclusion and Future Work

The work presented in this chapter has explored the application of a number of statistical methods to optimize the subtree based associative classification for tree-structured data. It has utilized a structure-preserving flat format representation, to progressively apply a number of statistical methods to first filter out irrelevant attributes followed by the removal of irrelevant and redundant rules. The use of this method has implications that the subtree based association rules are restricted to those that occur at the same position in the original tree database, and that the initial rule (before subtree reconstruction), can contain rules based on disconnected subtrees. Experiments were performed on three real datasets, and using the proposed approach a large number of rules were removed in both cases without negatively affecting the accuracy of the rule set, while for more structurally varied data, this

optimization was at the cost of a large reduction in coverage rate. The results on this data were compared with a structural classifier based on traditional subtrees, and some important differences and implications were highlighted. The results show that associations based on disconnected subtrees can be useful, while the positional constraint can often result in more precise rules for structurally varied data, but at the cost of lower coverage rate. From these findings one can conclude that when forming association rules for tree-structured data, one should not be constrained to a valid and connected subtree because an interesting association can be anywhere in a tree instance, and it does not need to be a connected subtree of that instance. These findings indicate that including disconnected subtrees and constraining the subtrees by their exact occurrence in the database in addition to traditional subtree patterns, could improve the classifiers for tree-structured data. The method used in this chapter is to be seen as complementary and in no way a replacement of the traditional way that subtrees are mined.

Our future work, will investigate the application domains where including such association rules can be beneficial and the right way to combine them with traditional subtree patterns for optimal performance.

Furthermore, the chi-square and the logistic regression measures were used as a case in point for statistic-based rule filtering, while Symmetrical Tau was utilized in the feature subset selection process. However, by no means is any claim being made that these are the most optimal measures to be used for their specific purpose. In fact, we have used the confidence constraint here because of the stronger focus on statistical quality assessment and the difference between the rule sets discovered using the traditional support and confidence framework. However, many other measures could be used and applied instead of the support and or confidence constraint, which, as discussed in several works [12, 23, 29], will yield more interesting rules. Therefore, another future work will evaluate the combinations of other constraints, statistical measures and techniques for rule removal/attribute relevance determination, in context of the tree-structured data domain.

## References

1. Agrawal, R., Imieliski, T., Swami, A.: Mining association rules between sets of items in large databases. ACM SIGMOD Rec. **22**(2), 207–216 (1993)
2. Aumann, Y., Lindell, Y.: A statistical theory for quantitative association rules. Intell. Inf. Syst. **20**(3), 253–283 (2003)
3. Bathoorn, R., Koopman, A., Siebes, A.: Reducing the frequent pattern set. In: Proceedings of the 6th IEEE International Conference on Data Mining—Workshops, pp. 55–59 (2006)
4. Bayardo, R., Agrawal, R., Gunopulos, D.: Constraint-based rule mining in large, dense databases. Data Min. Knowl. Discov. **4**(2–3), 217–240 (2000)
5. Blanchard, J., Guillet, F., Gras, R., Briand, H.: Using information-theoretic measures to assess association rule interestingness. In: Proceedings of the 5th IEEE International Conference on Data Mining, pp. 215–238 (2005)
6. Bolon-Canedo, V., Sanchez-Marono, N., Alonso-Betanzos, A.: A review of feature selection methods on synthetic data. Knowl. Inf. Syst. **34**(3), 483–519 (2013)
7. Brijs, T., Vanhoof, K., Wets, G.: Defining interestingness for association rules. Int. J. Inf. Theor. Appl. **10**(4), 370–376 (2003)

8. Brin, S., Motwani, R., Silverstein, C.: Beyond market baskets: generalizing association rules to correlations. In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 265–276 (1997)
9. Cheng, H., Yan, X., Han, J., Hsu, C.W.: Discriminative frequent pattern analysis for effective classification. In: Proceedings of the 23rd International IEEE Conference on Data Engineering, pp. 716–725 (2007)
10. Cheng, H., Yan, X., Han, J., Yu, P.: Direct discriminative pattern mining for effective classification. In: Proceedings of the 24th International Conference on Data Engineering, pp. 167–178 (2008)
11. Dash, M., Liu, H.: Feature selection for classification. *Intell. Data Anal.* **1**(3), 131–156 (1997)
12. Geng, L., Hamilton, H.: Interestingness measures for data mining: a survey. *ACM Comput. Surv.* **338**(3, Article No. 9) (2006)
13. Goodman, A., Kamath, C., Kumar, V.: Data analysis in the 21st century. *Stat. Anal. Data Min.* **1**(1), 1–3 (2008)
14. Hadzic, F.: A structure preserving flat data format representation for tree-structured data. In: Proceedings of PAKDD Workshops, vol. 2011, pp. 221–233 (2012)
15. Hadzic, F., Dillon, T.: Using the symmetrical tau ( $\tau$ ) criterion for feature selection in decision tree and neural network learning. In: Proceedings of the 2nd SIAM Workshop on Feature Selection for Data Mining: Interfacing Machine Learning and Statistics (2006)
16. Hadzic, F., Hecker, M.: Alternative approach to tree-structured web log representation and mining. In: Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pp. 235–242 (2011)
17. Hadzic, F., Tan, H., Dillon, T.S.: Mining of Data With Complex Structures, 1st edn, Studies in Computational Intelligence, vol. 333,. Springer (2011)
18. Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufmann (2001)
19. Hashimoto, K., Takigawa, I., Shiga, M., Kanehisa, M., Mamitsuka, H.: Mining significant tree patterns in carbohydrate sugar chains. *Bioinformatics* **24**(16), 167–173 (2008)
20. Knijff, J.D., Feelders, A.J.: Monotone constraints in frequent tree mining. In: Proceedings of the 14th Annual Machine Learning Conference of Belgium and the Netherlands, BENELEARN pp. 13–20 (2005)
21. Kudo, M., Sklansky, J.: Comparison of algorithms that select features for pattern classifiers. *Pattern Recognit.* **33**(1), 25–41 (2000)
22. Lallich, S., Teytaud, O., Prudhomme, E.: Association rule interestingness: measure and statistical validation. In: Quality Measures in Data Mining. Studies in Computational Intelligence, vol. 43, pp. 251–275. Springer (2007)
23. Lallich, S., Teytaud, O., Prudhomme, E.: Formal framework for the study of algorithmic properties of objective interestingness measures. In: Data Mining: Foundations and Intelligent Paradigms, vol. 24, pp. 77–98. ISRL (2012)
24. Le Bras, Y., Lenca, P., Lallich, S.: Mining classification rules without support: an anti-monotone property of Jaccard measure. In: Proceedings of the 14th International Conference on Discovery Science, pp. 179–193 (2011)
25. Lenca, P., Meyer, P., Vaillant, B., Lallich, S.: On selecting interestingness measures for association rules: user oriented description and multiple criteria decision aid. *Eur. J. Oper. Res.* **184**(2), 610–626 (2008)
26. Li, J., Shen, H., Topor, R.: Mining the optimal class association rule set. *Knowl.-Based Syst.* **15**(7), 399–405 (2002)
27. Little, R., Rubin, D.: Statistical Analysis with Missing Data, 2nd edn. Wiley, New York (2002)
28. Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In: Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining, pp. 80–86 (1998)
29. McGarry, K.: A survey of interestingness measures for knowledge discovery. *Knowl. Eng. Rev.* **20**(1), 39–61 (2005)
30. Molina, L., Belanche, L., Nebot, A.: Feature selection algorithms: a survey and experimental evaluation. In: Proceedings of IEEE International Conference on Data Mining, pp. 306–313 (2002)

31. Nakamura, A., Kudo, M.: Mining frequent trees with node-inclusion constraints. In: Advances in Knowledge Discovery and Data Mining, vol. 3518, pp. 850–860. Springer (2005)
32. Ozaki, T., Ohkawa, T.: New frontiers in applied data mining, PAKDD 2008 International Workshops. Mining Mutually Dependent Ordered Subtrees in Tree Databases, pp. 75–86. Springer, Heidelberg (2009)
33. Quinlan, J.R.: Induction of decision trees. *Mach. Learn.* **1**(1), 81–106 (1986)
34. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufman (1993)
35. Refaat, M.: Data Preparation for Data Mining Using SAS. Morgan Kaufmann Publishers, San Francisco (2007)
36. Roiger, R., Geatz, M.: Data Mining: A Tutorial-Based Primer. Addison Wesley, Boston (2003)
37. Shaharanee, I., Hadzic, F.: Evaluation and optimization of frequent, closed and maximal association rule based classification. *Stat. Comput.* **23**, 1–23 (2013)
38. Shaharanee, I., Hadzic, F., Dillon, T.: Interestingness measures for association rules based on statistical validity. *Knowl.-Based Syst.* **24**(3), 386–392 (2011)
39. Siebes, A., Vreeken, J., Leeuwen, M.V.: Item sets that compress. In: Proceedings of the SIAM Conference on Data Mining, pp. 393–404 (2006)
40. Siloverstein, C., Brin, S., Motwani, R.: Beyond market baskets: generalizing association rules to dependence rules. *Data Min. Knowl. Disc.* **2**(1), 39–68 (1998)
41. Srikanth, R., Vu, Q., Agrawal, R.: Mining association rules with item constraints. In: Proceedings of the 3rd International Conference on Knowledge Discovery in Databases and Data Mining, pp. 67–73 (1997)
42. Tan, H., Dillon, T., Hadzic, F., Feng, L., Chang, E.: IMB3-Miner: Mining induced/embedded subtrees by constraining the level of embedding. In: Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 450–461 (2006)
43. Tan, H., Hadzic, F., Dillon, T., Chang, E., Feng, L.: Tree model guided candidate generation for mining frequent subtrees from XML documents. *ACM Trans. Knowl. Disc. Data Min.* **2**(2), 1–43 (2008)
44. Tan, P., Kumar, V., Srivastava, J.: Selecting the right interestingness measure for association patterns. In: Proceedings of the 8th ACM Knowledge Discovery and Data Mining Conference, pp. 32–41 (2002)
45. Veloso, A., Meira, W., Zaki, M.: Lazy Associative classification. In: Proceedings of the 6th IEEE International Conference on Data Mining, pp. 645–654 (2006)
46. Webb, G.: Discovering significant patterns. *Mach. Learn.* **68**(1), 1–33 (2007)
47. Xiong, H., Tan, P.N., Kumar, V.: Hyperclique pattern discovery. *Data Min. Knowl. Disc.* **13**(2), 219–242 (2006)
48. Yan, X., Cheng, H., Han, J., Yu, P.S.: Mining significant graph patterns by leap search. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 433–444 (2008)
49. Yan, X., Han, J., Hsu, C.W.: Discriminative frequent pattern analysis for effective classification. In: Proceedings of the 23rd IEEE International Conference on Data Engineering, pp. 716–725 (2007)
50. Yin, X., Han, J.: CPAR: Classification based on predictive association rules. In: Proceedings of the SIAM International Conference on Data Mining, pp. 396–376 (2003)
51. Zaki, M.: Efficiently mining frequent trees in a forest: algorithms and applications. *IEEE Trans. Knowl. Data Eng.* **17**(8), 1021–1035 (2005)
52. Zaki, M.J., Aggarwal, C.: XRules: an effective structural classifier for XML data. In: Proceedings of the 9th ACM Knowledge Discovery and Data Mining Conference, pp. 316–325 (2003)
53. Zhang, C., Zhang, S.: Collecting quality data for database mining. In: AI 2001: Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 2256, pp. 593–604. Springer (2001)
54. Zhou, X., Dillon, T.: A statistical-heuristic feature selection criterion for decision tree induction. *IEEE Trans. Pattern Anal. Mach. Intell.* **13**(8), 834–841 (1991)

**Part IV**

**Data- and Domain-Oriented**

**Methodologies**

# Chapter 11

## Hubness-Aware Classification, Instance Selection and Feature Construction: Survey and Extensions to Time-Series

Nenad Tomašev, Krisztian Buza, Kristóf Marussy and Piroska B. Kis

**Abstract** Time-series classification is the common denominator in many real-world pattern recognition tasks. In the last decade, the simple nearest neighbor classifier, in combination with dynamic time warping (DTW) as distance measure, has been shown to achieve surprisingly good overall results on time-series classification problems. On the other hand, the presence of hubs, i.e., instances that are similar to exceptionally large number of other instances, has been shown to be one of the crucial properties of time-series data sets. To achieve high performance, the presence of hubs should be taken into account for machine learning tasks related to time-series. In this chapter, we survey hubness-aware classification methods and instance selection, and we propose to use selected instances for feature construction. We provide detailed description of the algorithms using uniform terminology and notations. Many of the surveyed approaches were originally introduced for vector classification, and their application to time-series data is novel, therefore, we provide experimental results on large number of publicly available real-world time-series data sets.

**Keywords** Time-series classification · Hubs · Instance selection · Feature construction

---

N. Tomašev  
Institute Jožef Stefan, Artificial Intelligence Laboratory, Jamova 39,  
1000 Ljubljana, Slovenia  
e-mail: nenad.tomasev@gmail.com

K. Buza (✉)  
Faculty of Mathematics, Informatics and Mechanics,  
University of Warsaw (MIMUW), Banacha 2, 02-097 Warszawa, Poland  
e-mail: chrisbuza@yahoo.com

K. Marussy  
Department of Computer Science and Information Theory,  
Budapest University of Technology and Economics, Magyar Tudósok Krt. 2.,  
Budapest 1117, Hungary  
e-mail: marussy@cs.bme.hu

P.B. Kis  
Department of Mathematics and Computer Science, College of Dunaújváros,  
Táncsics M. u. 1/a, Dunaújváros 2400, Hungary  
e-mail: pbkism@yahoo.com

## 11.1 Introduction

Time-series classification is one of the core components of various real-world recognition systems, such as computer systems for speech and handwriting recognition, signature verification, sign-language recognition, detection of abnormalities in electrocardiograph signals, tools based on electroencephalograph (EEG) signals (“brain waves”), i.e., spelling devices and EEG-controlled web browsers for paralyzed patients, and systems for EEG-based person identification, see e.g. [34, 35, 37, 45]. Due to the increasing interest in time-series classification, various approaches have been introduced including neural networks [26, 38], Bayesian networks [48], hidden Markov models [29, 33, 39], genetic algorithms, support vector machines [14], methods based on random forests and generalized radial basis functions [5] as well as frequent pattern mining [17], histograms of symbolic polynomials [18] and semi-supervised approaches [36]. However, one of the most surprising results states that the simple  $k$ -nearest neighbor ( $k$ NN) classifier using dynamic time warping (DTW) as distance measure is competitive (if not superior) to many other state-of-the-art models for several classification tasks, see e.g. [8] and the references therein. Besides experimental evidence, there are theoretical results about the optimality of nearest neighbor classifiers, see e.g. [12]. Some of the recent theoretical works focused on a time series classification, in particular on why nearest neighbor classifiers work well in case of time series data [10].

On the other hand, Radovanović et al. observed the presence of hubs in time-series data, i.e., the phenomenon that a few instances tend to be the nearest neighbor of surprising lot of other instances [43]. Furthermore, they introduced the notion of bad hubs. A hub is said to be bad if its class label differs from the class labels of many of those instances that have this hub as their nearest neighbor. In the context of  $k$ -nearest neighbor classification, bad hubs were shown to be responsible for a large portion of the misclassifications. Therefore, hubness-aware classifiers and instance selection methods were developed in order to make classification faster and more accurate [9, 43, 50, 52–54].

As the presence of hubs is a general phenomenon characterizing many datasets, we argue that it is of relevance to feature selection approaches as well. Therefore, in this chapter, we will survey the aforementioned results and describe the most important hubness-aware classifiers in detail using unified terminology and notations. As a first step towards hubness-aware feature selection, we will examine the usage of distances from the selected instances as features in a state-of-the-art classifier.

The methods proposed in [50, 52–54] were originally designed for vector classification and they are novel to the domain of time-series classification. Therefore, we will provide experimental evidence supporting the claim that these methods can be effectively applied to the problem of time-series classification. The usage of distances from selected instances as features can be seen as transforming the time-series into a vector space. While the technique of projecting the data into a new space is widely used in classification, see e.g. support vector machines [7, 11] and principal component analysis [25], to our best knowledge, the particular procedure we perform is

novel in time-series classification, therefore, we will experimentally evaluate it and compare to state-of-the-art time-series classifiers.

The remainder of this chapter is organized as follows: in Sect. 11.2 we formally define the time-series classification problem, summarize the basic notation used throughout this chapter and shortly describe nearest neighbor classification. Section 11.3 is devoted to dynamic time warping, and Sect. 11.4 presents the hubness phenomenon. In Sect. 11.5 we describe state-of-the-art hubness-aware classifiers, followed by hubness-aware instance selection and feature construction approaches in Sect. 11.6. Finally, we conclude in Sect. 11.7.

## 11.2 Problem Formulation and Basic Notations

The problem of classification can be stated as follows. We are given a set of instances and some groups. The groups are called classes, and they are denoted as  $C_1, \dots, C_m$ . Each instance  $x$  belongs to one of the classes.<sup>1</sup> Whenever  $x$  belongs to class  $C_i$ , we say that the class label of  $x$  is  $C_i$ . We denote the set of all the classes by  $\mathcal{C}$ , i.e.,  $\mathcal{C} = \{C_1, \dots, C_m\}$ . Let  $\mathcal{D}$  be a dataset of instances  $x_i$  and their class labels  $y_i$ , i.e.,  $\mathcal{D} = \{(x_1, y_1) \dots (x_n, y_n)\}$ . We are given a dataset  $\mathcal{D}^{\text{train}}$ , called *training data*. The task of classification is to induce a function  $f(x)$ , called *classifier*, which is able to assign class labels to instances not contained in  $\mathcal{D}^{\text{train}}$ .

In real-world applications, for some instances we know (from measurements and/or historical data) to which classes they belong, while the class labels of other instances are unknown. Based on the data with known classes, we induce a classifier, and use it to determine the class labels of the rest of the instances.

In experimental settings we usually aim at measuring the performance of a classifier. Therefore, after inducing the classifier using  $\mathcal{D}^{\text{train}}$ , we use a second dataset  $\mathcal{D}^{\text{test}}$ , called *test data*: for the instances of  $\mathcal{D}^{\text{test}}$ , we compare the output of the classifier, i.e., the predicted class labels, with the true class labels, and calculate the accuracy of classification. Therefore, the task of classification can be defined formally as follows: given two datasets  $\mathcal{D}^{\text{train}}$  and  $\mathcal{D}^{\text{test}}$ , the task of classification is to induce a classifier  $f(x)$  that maximizes prediction accuracy for  $\mathcal{D}^{\text{test}}$ . For the induction of  $f(x)$ , however, solely  $\mathcal{D}^{\text{train}}$  can be used, but not  $\mathcal{D}^{\text{test}}$ .

Next, we describe the  $k$ -nearest neighbor classifier ( $k$ NN). Suppose, we are given an instance  $x^* \in \mathcal{D}^{\text{test}}$  that should be classified. The  $k$ NN classifier searches for those  $k$  instances of the training dataset that are most similar to  $x^*$ . These  $k$  most similar instances are called the  $k$  nearest neighbors of  $x^*$ . The  $k$ NN classifier considers the  $k$  nearest neighbors, and takes the majority vote of their labels and assigns this label to  $x^*$ : e.g. if  $k = 3$  and two of the nearest neighbors of  $x^*$  belong to class  $C_1$ , while one

---

<sup>1</sup> In this chapter, we only consider the case when each instance belongs to exactly one class. Note, however, that the presence of hubs may be relevant in the context of multilabel and fuzzy classification as well.

**Table 11.1** Abbreviations used throughout the chapter and the sections where those concepts are defined/explained

Abbreviation	Full name	Definition
AKNN	Adaptive $k$ NN	Sect. 11.5.5
$BN_k(x)$	Bad $k$ -occurrence of $x$	Sect. 11.4
DTW	Dynamic Time Warping	Sect. 11.3
$GN_k(x)$	Good $k$ -occurrence of $x$	Sect. 11.4
h-FNN	Hubness-based fuzzy nearest neighbor	Sect. 11.5.2
HIKNN	Hubness information $k$ -nearest neighbor	Sect. 11.5.4
hw- $k$ NN	Hubness-aware weighting for $k$ NN	Sect. 11.5.1
INSIGHT	Instance selection based on graph-coverage and hubness for time-series	Sect. 11.6.1
$k$ NN	$k$ -nearest neighbor classifier	Sect. 11.2
NHBNN	Naive hubness Bayesian $k$ -nearest Neighbor	Sect. 11.5.3
$N_k(x)$	$k$ -occurrence of $x$	Sect. 11.4
$N_{k,C}(x)$	Class-conditional $k$ -occurrence of $x$	Sect. 11.4
$\mathcal{S}_{N_k(x)}$	Skewness of $N_k(x)$	Sect. 11.4
RImb	Relative imbalance factor	Sect. 11.5.5

of the nearest neighbors of  $x$  belongs to class  $C_2$ , then this 3-NN classifier recognizes  $x^*$  as an instance belonging to the class  $C_1$ .

We use  $\mathcal{N}_k(x)$  to denote the set of  $k$  nearest neighbors of  $x$ .  $\mathcal{N}_k(x)$  is also called as the  $k$ -neighborhood of  $x$ .

Abbreviations used throughout this chapter are summarized in Table 11.1.

## 11.3 Dynamic Time Warping

While the  $k$ NN classifier is intuitive in vector spaces, in principle, it can be applied to any kind of data, i.e., not only in case if the instances correspond to points of a vector space. The only requirement is that an appropriate distance measure is present that can be used to determine the most similar train instances. In case of time-series classification, the instances are time-series and one of the most widely used distance measures is DTW. We proceed by describing DTW. We assume that a time-series  $x$  of length  $l$  is a sequence of real numbers:  $x = (x[0], x[1], \dots, x[l - 1])$ .

In the most simple case, while calculating the distance of two time series  $x_1$  and  $x_2$ , one would compare the  $k$ th element of  $x_1$  to the  $k$ th element of  $x_2$  and aggregate the results of such comparisons. In reality, however, when observing the same phenomenon several times, we cannot expect it to happen (or any characteristic pattern to appear) always at exactly the same time position, and the event's duration can also vary slightly. Therefore, DTW captures the similarity of two time series'

shapes in a way that it allows for elongations: the  $k$ th position of time series  $x_1$  is compared to the  $k'$ th position of  $x_2$ , and  $k'$  may or may not be equal to  $k$ .

DTW is an edit distance [30]. This means that we can conceptually consider the calculation of the DTW distance of two time series  $x_1$  and  $x_2$  of length  $l_1$  and  $l_2$  respectively as the process of transforming  $x_1$  into  $x_2$ . Suppose we have already transformed a prefix (possibly having length zero or  $l_1$  in the extreme cases) of  $x_1$  into a prefix (possibly having length zero or  $l_2$  in the extreme cases) of  $x_2$ . Consider the next elements, the elements that directly follow the already-transformed prefixes, of  $x_1$  and  $x_2$ . The following editing steps are possible, both of which being associated with a cost:

1. *replacement* of the next element of  $x_1$  for the next element of  $x_2$ , in this case, the next element of  $x_1$  is matched to the next element of  $x_2$ , and
2. *elongation* of an element: the next element of  $x_1$  is matched to the last element of the already-matched prefix of  $x_2$  or vice versa.

As result of the replacement step, both prefixes of the already-matched elements grow by one element (by the next elements of  $x_1$  and  $x_2$  respectively). In contrast, in an elongation step, one of these prefixes grows by one element, while the other prefix remains the same as before the elongation step.

The cost of transforming the entire time series  $x_1$  into  $x_2$  is the sum of the costs of all the necessary editing steps. In general, there are many possibilities to transform  $x_1$  into  $x_2$ , DTW calculates the one with minimal cost. This minimal cost serves as the distance between both time series. The details of the calculation of DTW are described next.

DTW utilizes the dynamic programming approach [45]. Denoting the length of  $x_1$  by  $l_1$ , and the length of  $x_2$  by  $l_2$ , the calculation of the minimal transformation cost is done by filling the entries of an  $l_1 \times l_2$  matrix. Each number in the matrix corresponds to the distance between a subsequence of  $x_1$  and a subsequence of  $x_2$ . In particular, the number in the  $i$ th row and  $j$ th column, <sup>2</sup>  $d_0^{DTW}(i, j)$  corresponds to the distance between the subsequences  $x'_1 = (x_1[0], \dots, x_1[i])$  and  $x'_2 = (x_2[0], \dots, x_2[j])$ . This is shown in Fig. 11.1.

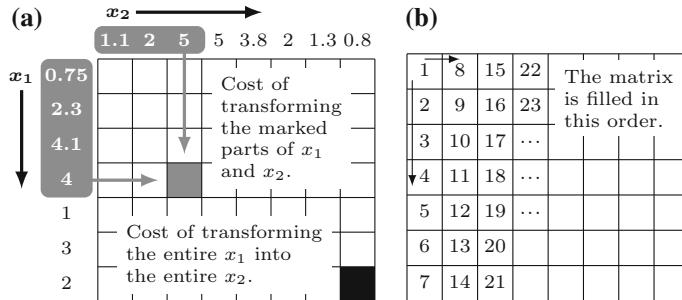
When we try to match the  $i$ th position of  $x_1$  and the  $j$ th position of  $x_2$ , there are three possible cases: (i) elongation in  $x_1$ , (ii) elongation in  $x_2$ , and (iii) no elongation.

If there is no elongation, the prefix of  $x_1$  up to the  $(i - 1)$ th position is matched (transformed) to the prefix of  $x_2$  up to the  $(j - 1)$ th position, and the  $i$ th position of  $x_1$  is matched (transformed) to the  $j$ th position of  $x_2$ .

Elongation in  $x_1$  at the  $i$ th position means that the  $i$ th position of  $x_1$  has already been matched to at least one position of  $x_2$ , i.e., the prefix of  $x_1$  up to the  $i$ th position is matched (transformed) to the prefix of  $x_2$  up to the  $(j - 1)$ th position, and the  $i$ th position of  $x_1$  is matched again, this time to the  $j$ th position of  $x_2$ . This way the  $i$ th position of  $x_1$  is elongated, in the sense that it is allowed to match several positions of  $x_2$ . The elongation in  $x_2$  can be described in an analogous way.

---

<sup>2</sup> Please note that the numbering of the columns and rows begins with zero, i.e., the very-first column/row of the matrix is called in this sense as the 0th column/row.



**Fig. 11.1** The DTW-matrix. While calculating the distance (transformation cost) between two time series  $x_1$  and  $x_2$ , DTW fills-in the cells of a matrix. **a** The values of time series  $x_1 = (0.75, 2.3, 4.1, 4, 1, 3, 2)$  are enumerated on the left of the matrix from *top to bottom*. Time series  $x_2$  is shown on the *top* of the matrix. A number in a cell corresponds to the distance (transformation cost) between two prefixes of  $x_1$  and  $x_2$ . **b** The order of filling the positions of the matrix

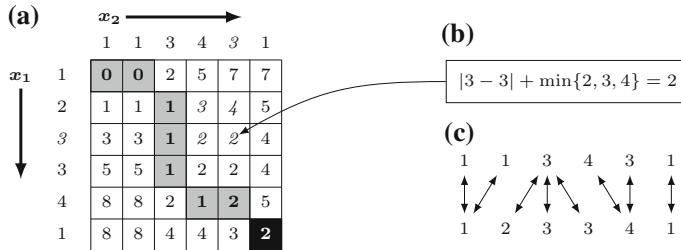
Out of these three possible cases, DTW selects the one that transforms the prefix  $x'_1 = (x_1[0], \dots, x_1[i])$  into the prefix  $x'_2 = (x_2[0], \dots, x_2[j])$  with minimal overall costs. Denoting the distance between the subsequences  $x'_1$  and  $x'_2$ , i.e. the value of the cell in the  $i$ th row and  $j$ th column, as  $d_0^{DTW}(i, j)$ , based on the above discussion, we can write:

$$d_0^{DTW}(i, j) = c_{tr}^{DTW}(x_1[i], x_2[j]) + \min \left\{ \begin{array}{l} d_0^{DTW}(i, j - 1) + c_{el}^{DTW} \\ d_0^{DTW}(i - 1, j) + c_{el}^{DTW} \\ d_0^{DTW}(i - 1, j - 1) \end{array} \right\}. \quad (11.1)$$

In this formula, the first, second, and third terms of the minimum correspond to the above cases of elongation in  $x_1$ , elongation in  $x_2$  and no elongation, respectively. The cost of matching (transforming) the  $i$ th position of  $x_1$  to the  $j$ th position of  $x_2$  is  $c_{tr}^{DTW}(x_1[i], x_2[j])$ . If  $x_1[i]$  and  $x_2[j]$  are identical, the cost of this replacement is zero. This cost is present in all the three above cases. In the cases, when elongation happens, there is an additional elongation cost denoted as  $c_{el}^{DTW}$ .

According to the principles of dynamic programming, Formula (11.1) can be calculated for all  $i, j$  in a column-wise fashion. First, set  $d_0^{DTW}(0, 0) = c_{tr}^{DTW}(x_1[0], x_2[0])$ . Then we begin calculating the very first column of the matrix ( $j = 0$ ), followed by the next column corresponding to  $j = 1$ , etc. The cells of each column are calculated in order of their row-indexes: within one column, the cell in the row corresponding  $i = 0$  is calculated first, followed by the cells corresponding to  $i = 1, 2, \dots$  (see Fig. 11.1). In some cases (in the very-first column and in the very-first cell of each row), in the min function of Formula (11.1), some of the terms are undefined (when  $i - 1$  or  $j - 1$  equals  $-1$ ). In these cases, the minimum of the other (defined) terms are taken.

The DTW distance of  $x_1$  and  $x_2$ , i.e. the cost of transforming the entire time series  $x_1 = (x_1[0], x_1[1], \dots, x_1[l_1 - 1])$  into  $x_2 = (x_2[0], x_2[1], \dots, x_2[l_2 - 1])$  is



**Fig. 11.2** Example for the calculation of the DTW-matrix. **a** The DTW-matrix calculated with  $c_{tr}^{DTW}(v_A, v_B) = |v_A - v_B|$ ,  $c_{el}^{DTW} = 0$ . The time series  $x_1$  and  $x_2$  are shown on the *left* and *top* of the matrix respectively. **b** The calculation of the value of a cell. **c** The (implicitly) constructed mapping between the values of the both time series. The cells are leading to the minimum in Formula (11.1), i.e., the ones that allow for this mapping, are marked in the DTW-matrix

$$d_{DTW}(x_1, x_2) = d_0^{DTW}(l_1 - 1, l_2 - 1). \quad (11.2)$$

An example for the calculation of DTW is shown in Fig. 11.2.

Note that the described method implicitly constructs a mapping between the positions of the time series  $x_1$  and  $x_2$ : by back-tracking which of the possible cases leads to the minimum in the Formula (11.1) in each step, i.e., which of the above discussed three possible cases leads to the minimal transformation costs in each step, we can reconstruct the mapping of positions between  $x_1$  and  $x_2$ .

For the final result of the distance calculation, the values close to the diagonal of the matrix are usually the most important ones (see Fig. 11.2 for an illustration). Therefore, a simple, but effective way of speeding-up dynamic time warping is to restrict the calculations to the cells around the diagonal of the matrix [45]. This means that one limits the elongations allowed when matching the both time series (see Fig. 11.3).

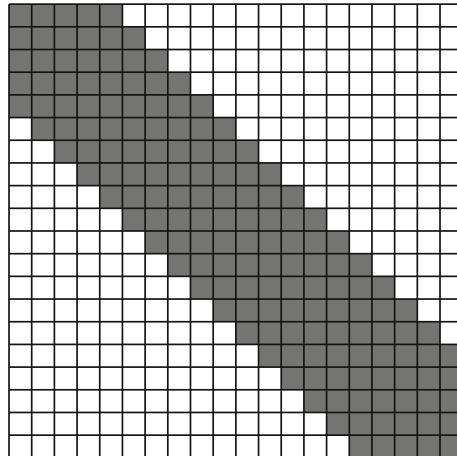
Restricting the warping window size to a pre-defined constant  $w^{DTW}$  (see Fig. 11.3) implies that it is enough to calculate only those cells of the matrix that are at most  $w^{DTW}$  positions far from the main diagonal along the vertical direction:

$$d_0^{DTW}(i, j) \text{ is calculated } \Leftrightarrow |i - j| \leq w^{DTW}. \quad (11.3)$$

The warping window size  $w^{DTW}$  is often expressed in percentage relative to the length of the time series. In this case,  $w^{DTW} = 100\%$  means calculating the entire matrix, while  $w^{DTW} = 0\%$  refers to the extreme case of not calculating any entries at all. Setting  $w^{DTW}$  to a relatively small value such as 5 %, does not negatively affect the accuracy of the classification, see e.g. [8] and the references therein.

In the settings used throughout this chapter, the cost of elongation,  $c_{el}^{DTW}$ , is set to zero:

$$c_{el}^{DTW} = 0. \quad (11.4)$$



**Fig. 11.3** Limiting the size of the warping window: only the cells around the main diagonal of the matrix (*marked cells*) are calculated

The cost of transformation (matching), denoted as  $c_{tr}^{DTW}$ , depends on what value is replaced by what: if the numerical value  $v_A$  is replaced by  $v_B$ , the cost of this step is:

$$c_{tr}^{DTW}(v_A, v_B) = |v_A - v_B|. \quad (11.5)$$

We set the warping window size to  $w^{DTW} = 5\%$ . For more details and further recent results on DTW, we refer to [8].

## 11.4 Hubs in Time-Series Data

The presence of hubs, i.e., some few instances that tend to occur surprisingly frequently as nearest neighbors while other instances (almost) never occur as nearest neighbors, has been observed for various natural and artificial networks, such as protein-protein-interaction networks or the internet [3, 22]. The presence of hubs has been confirmed in various contexts, including text mining, music retrieval and recommendation, image data and time series [43, 46, 49]. In this chapter, we focus on time series classification, therefore, we describe hubness from the point of view of time-series classification.

For classification, the property of hubness was explored in [40–43]. The property of hubness states that for data with high (intrinsic) dimensionality, like most of the time series data,<sup>3</sup> some instances tend to become nearest neighbors much more

---

<sup>3</sup> In case of time series, consecutive values are strongly interdependent, thus instead of the length of time series, we have to consider the *intrinsic* dimensionality [43].

frequently than others. Intuitively speaking, very frequent neighbors, or hubs, dominate the neighbor sets and therefore, in the context of similarity-based learning, they represent the centers of influence within the data. In contrast to hubs, there are rarely occurring neighbor instances contributing little to the analytic process. We will refer to them as *orphans* or *anti-hubs*.

In order to express hubness in a more precise way, for a time series dataset  $\mathcal{D}$  one can define the *k-occurrence* of a time series  $x$  from  $\mathcal{D}$ , denoted by  $N_k(x)$ , as the number of time series in  $\mathcal{D}$  having  $x$  among their  $k$  nearest neighbors:

$$N_k(x) = |\{x_i | x \in \mathcal{N}_k(x_i)\}|. \quad (11.6)$$

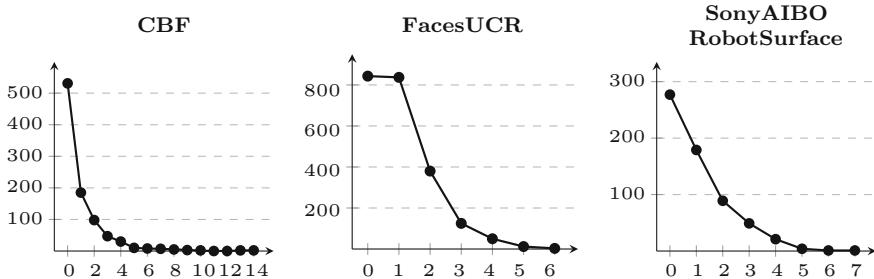
With the term *hubness* we refer to the phenomenon that the distribution of  $N_k(x)$  becomes significantly skewed to the right. We can measure this skewness, denoted by  $\mathcal{S}_{N_k(x)}$ , with the standardized third moment of  $N_k(x)$ :

$$\mathcal{S}_{N_k(x)} = \frac{E[(N_k(x) - \mu_{N_k(x)})^3]}{\sigma_{N_k(x)}^3} \quad (11.7)$$

where  $\mu_{N_k(x)}$  and  $\sigma_{N_k(x)}$  are the mean and standard deviation of the distribution of  $N_k(x)$ . When  $\mathcal{S}_{N_k(x)}$  is higher than zero, the corresponding distribution is skewed to the right and starts presenting a long tail. It should be noted, though, that the occurrence distribution skewness is only one indicator statistic and that the distributions with the same or similar skewness can still take different shapes.

In the presence of class labels, we distinguish between *good hubness* and *bad hubness*: we say that the time series  $x'$  is a *good k-nearest neighbor* of the time series  $x$ , if (i)  $x'$  is one of the  $k$ -nearest neighbors of  $x$ , and (ii) both have the same class labels. Similarly: we say that the time series  $x'$  is a *bad k-nearest neighbor* of the time series  $x$ , if (i)  $x'$  is one of the  $k$ -nearest neighbors of  $x$ , and (ii) they have different class labels. This allows us to define *good (bad) k-occurrence* of a time series  $x$ ,  $GN_k(x)$  (and  $BN_k(x)$  respectively), which is the number of other time series that have  $x$  as one of their good (bad, respectively)  $k$ -nearest neighbors. For time series, both distributions  $GN_k(x)$  and  $BN_k(x)$  are usually skewed, as it is exemplified in Fig. 11.4, which depicts the distribution of  $GN_1(x)$  for some time series data sets (from the UCR time series dataset collection [28]). As shown, the distributions have long tails in which the good hubs occur.

We say that a time series  $x$  is a good (or bad) hub, if  $GN_k(x)$  (or  $BN_k(x)$ , respectively) is exceptionally large for  $x$ . For the nearest neighbor classification of time series, the skewness of good occurrence is of major importance, because some few time series are responsible for large portion of the overall error: bad hubs tend to misclassify a surprisingly large number of other time series [43]. Therefore, one has to take into account the presence of good and bad hubs in time series datasets. While the  $k$ NN classifier is frequently used for time series classification, the  $k$ -nearest neighbor approach is also well suited for learning under class imbalance [16, 20, 21], therefore



**Fig. 11.4** Distribution of  $GN_1(x)$  for some time series datasets. The *horizontal axis* corresponds to the values of  $GN_1(x)$ , while on the *vertical axis* one can see how many instances have that value

hubness-aware classifiers, the ones we present in the next section, are also relevant for the classification of imbalanced data.

The total occurrence count of an instance  $x$  can be decomposed into good and bad occurrence counts:  $N_k(x) = GN_k(x) + BN_k(x)$ . More generally, we can decompose the total occurrence count into the class-conditional counts:  $N_k(x) = \sum_{C \in \mathcal{C}} N_{k,C}(x)$  where  $N_{k,C}(x)$  denotes how many times  $x$  occurs as one of the  $k$  nearest neighbors of instances belonging to class  $C$ , i.e.,

$$N_{k,C}(x) = |\{x_i | x_i \in \mathcal{N}_k(x_i) \wedge y_i = C\}| \quad (11.8)$$

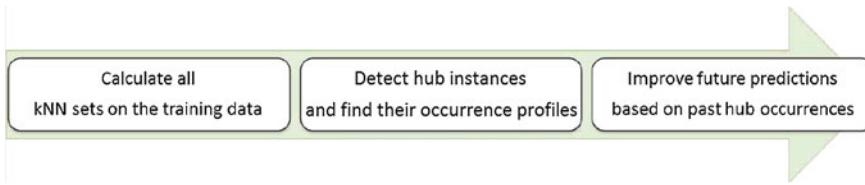
where  $y_i$  denotes the class label of  $x_i$ .

As we mentioned, hubs appear in data with high (intrinsic) dimensionality, therefore, hubness is one of the main aspects of the curse of dimensionality [4]. However, dimensionality reduction can not entirely eliminate the issue of bad hubs, unless it induces significant information loss by reducing to a very low dimensional space—which often ends up hurting system performance even more [40].

## 11.5 Hubness-Aware Classification of Time-Series

Since the issue of hubness in intrinsically high-dimensional data, such as time-series, cannot be entirely avoided, the algorithms that work with high-dimensional data need to be able to properly handle hubs. Therefore, in this section, we present algorithms that work under the assumption of hubness. These mechanisms might be either explicit or implicit.

Several hubness-aware classification methods have recently been proposed. An instance-weighting scheme was first proposed in [43], which reduces the bad influence of hubs during voting. An extension of the fuzzy  $k$ -nearest neighbor framework was shown to be somewhat better on average [54], introducing the concept of *class-conditional hubness* of neighbor points and building an occurrence model which is



**Fig. 11.5** The hubness-aware analytic framework: learning from past neighbor occurrences

used in classification. This approach was further improved by considering the self-information of individual neighbor occurrences [50]. If the neighbor occurrences are treated as random events, the Bayesian approaches also become possible [52, 53].

Generally speaking, in order to predict how hubs will affect classification of non-labeled instances (e.g. instances arising from observations in the future), we can model the influence of hubs by considering the training data. The training data can be utilized to learn a neighbor occurrence model that can be used to estimate the probability of individual neighbor occurrences for each class. This is summarized in Fig. 11.5. There are many ways to exploit the information contained in the occurrence models. Next, we will review the most prominent approaches.

While describing these approaches, we will consider the case of classifying an instance  $x^*$ , and we will denote its nearest neighbors as  $x_i$ ,  $i \in \{1, \dots, k\}$ . We assume that the test data is not available when building the model, and therefore  $N_k(x)$ ,  $N_{k,C}(x)$ ,  $GN_k(x)$ ,  $BN_k(x)$  are calculated on the training data.

### 11.5.1 *hw-kNN: Hubness-Aware Weighting*

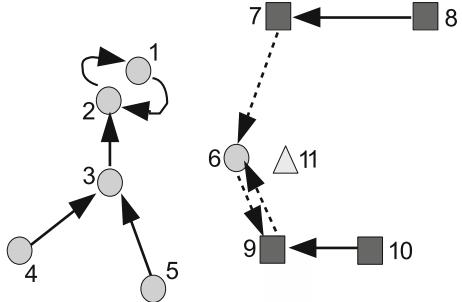
The weighting algorithm proposed by Radovanović et al. [41] is one of the simplest ways to reduce the influence of bad hubs. They assign lower voting weights to bad hubs in the nearest neighbor classifier. In *hw-kNN*, the vote of each neighbor  $x_i$  is weighted by  $e^{-h_b(x_i)}$ , where

$$h_b(x_i) = \frac{BN_k(x_i) - \mu_{BN_k(x)}}{\sigma_{BN_k(x)}} \quad (11.9)$$

is the standardized bad hubness score of the neighbor instance  $x_i \in \mathcal{N}_k(x^*)$ ,  $\mu_{BN_k(x)}$  and  $\sigma_{BN_k(x)}$  are the mean and standard deviation of the distribution of  $BN_k(x)$ .

*Example 1* We illustrate the calculation of  $N_k(x)$ ,  $GN_k(x)$ ,  $BN_k(x)$  and the *hw-kNN* approach on the example shown in Fig. 11.6. As described previously, hubness primarily characterizes high-dimensional data. However, in order to keep it simple, this illustrative example is taken from the domain of low dimensional vector classification. In particular, the instances are two-dimensional, therefore, they can be mapped to points of the plane as shown in Fig. 11.6. Circles (instances 1–6) and

**Fig. 11.6** Running example used to illustrate hubness-aware classifiers. Instances belong to two classes, denoted by *circles* and *rectangles*. The triangle is an instance to be classified



rectangles (instances 7–10) denote the training data: circles belong to class 1, while rectangles belong to class 2. The triangle (instance 11) is an instance that has to be classified.

For simplicity, we use  $k = 1$  and we calculate  $N_1(x)$ ,  $GN_1(x)$  and  $BN_1(x)$  for the instances of the training data. For each training instance shown in Fig. 11.6, an arrow denotes its nearest neighbor in the training data. Whenever an instance  $x'$  is a good neighbor of  $x$ , there is a continuous arrow from  $x$  to  $x'$ . In cases if  $x'$  is a bad neighbor of  $x$ , there is a dashed arrow from  $x$  to  $x'$ .

We can see, e.g., that instance 3 appears twice as good nearest neighbor of other train instances, while it never appears as bad nearest neighbor, therefore,  $GN_1(x_3) = 2$ ,  $BN_1(x_3) = 0$  and  $N_1(x_3) = GN_1(x_3) + BN_1(x_3) = 2$ . For instance 6, the situation is the opposite:  $GN_1(x_6) = 0$ ,  $BN_1(x_6) = 2$  and  $N_1(x_6) = GN_1(x_6) + BN_1(x_6) = 2$ , while instance 9 appears both as good and bad nearest neighbor:  $GN_1(x_9) = 1$ ,  $BN_1(x_9) = 1$  and  $N_1(x_9) = GN_1(x_9) + BN_1(x_9) = 2$ . The second, third and fourth columns of Table 11.2 show  $GN_1(x)$ ,  $BN_1(x)$  and  $N_1(x)$  for each instance and the calculated means and standard deviations of the distributions of  $GN_1(x)$ ,  $BN_1(x)$  and  $N_1(x)$ .

While calculating  $N_k(x)$ ,  $GN_k(x)$  and  $BN_k(x)$ , we used  $k = 1$ . Note, however, that we do not necessarily have to use the same  $k$  for the  $k$ NN classification of the unlabeled/test instances. In fact, in case of  $k$ NN classification with  $k = 1$ , only one instance is taken into account for determining the class label, and therefore the weighting procedure described above does not make any difference to the simple 1 nearest neighbor classification. In order to illustrate the use of the weighting procedure, we classify instance 11 with  $k' = 2$  nearest neighbor classifier, while  $N_k(x)$ ,  $GN_k(x)$ ,  $BN_k(x)$  were calculated using  $k = 1$ . The two nearest neighbors of instance 11 are instances 6 and 9. The weights associated with these instances are:

$$w_6 = e^{-h_b(x_6)} = e^{-\frac{BN_1(x_6) - \mu_{BN_1(x)}}{\sigma_{BN_1(x)}}} = e^{-\frac{2-0.3}{0.675}} = 0.0806$$

**Table 11.2**  $GN_1(x)$ ,  $BN_1(x)$ ,  $N_1(x)$ ,  $N_{1,C_1}(x)$  and  $N_{1,C_2}(x)$  for the instances shown in Fig. 11.6

Instance	$GN_1(x)$	$BN_1(x)$	$N_1(x)$	$N_{1,C_1}(x)$	$N_{1,C_2}(x)$
1	1	0	1	1	0
2	2	0	2	2	0
3	2	0	2	2	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	2	2	0	2
7	1	0	1	0	1
8	0	0	0	0	0
9	1	1	2	1	1
10	0	0	0	0	0
Mean	$\mu_{GN_1(x)} = 0.7$	$\mu_{BN_1(x)} = 0.3$	$\mu_{N_1(x)} = 1$		
Std.	$\sigma_{GN_1(x)} = 0.823$	$\sigma_{BN_1(x)} = 0.675$	$\sigma_{N_1(x)} = 0.943$		

and

$$w_9 = e^{-h_b(x_9)} = e^{-\frac{BN_1(x_9) - \mu_{BN_1(x)}}{\sigma_{BN_1(x)}}} = e^{-\frac{1-0.3}{0.675}} = 0.3545.$$

As  $w_9 > w_6$ , instance 11 will be classified as rectangle according to instance 9.

From the example we can see that in hw- $k$ NN all neighbors vote by their own label. As this may be disadvantageous in some cases [49], in the algorithms considered below, the neighbors do not always vote by their own labels, which is a major difference to hw- $k$ NN.

### 11.5.2 h-FNN: Hubness-Based Fuzzy Nearest Neighbor

Consider the relative class hubness  $u_C(x_i)$  of each nearest neighbor  $x_i$ :

$$u_C(x_i) = \frac{N_{k,C}(x_i)}{N_k(x_i)}. \quad (11.10)$$

The above  $u_C(x_i)$  can be interpreted as the fuzziness of the event that  $x_i$  occurred as one of the neighbors,  $C$  denotes one of the classes:  $C \in \mathcal{C}$ . Integrating fuzziness as a measure of uncertainty is usual in  $k$ -nearest neighbor methods and h-FNN [54] uses the relative class hubness when assigning class-conditional vote weights. The approach is based on the fuzzy  $k$ -nearest neighbor voting framework [27]. Therefore, the probability of each class  $C$  for the instance  $x^*$  to be classified is estimated as:

$$u_C(x^*) = \frac{\sum_{x_i \in \mathcal{N}_k(x^*)} u_C(x_i)}{\sum_{x_i \in \mathcal{N}_k(x^*)} \sum_{C' \in \mathcal{C}} u_{C'}(x_i)}. \quad (11.11)$$

*Example 2* We illustrate h-FNN on the example shown in Fig. 11.6.  $N_{k,C}(x)$  is shown in the fifth and sixth column of Table 11.2 for both classes of circles ( $C_1$ ) and rectangle ( $C_2$ ). Similarly to the previous section, we calculate  $N_{k,C}(x_i)$  using  $k = 1$ , but we classify instance 11 using  $k' = 2$  nearest neighbors, i.e.,  $x_6$  and  $x_9$ . The relative class hubness values for both classes for the instances  $x_6$  and  $x_9$  are:

$$\begin{aligned} u_{C_1}(x_6) &= 0/2 = 0, \quad u_{C_2}(x_6) = 2/2 = 1, \\ u_{C_1}(x_9) &= 1/2 = 0.5, \quad u_{C_2}(x_9) = 1/2 = 0.5. \end{aligned}$$

According to (11.11), the class probabilities for instance 11 are:

$$u_{C_1}(x_{11}) = \frac{0 + 0.5}{0 + 1 + 0.5 + 0.5} = 0.25,$$

and

$$u_{C_2}(x_{11}) = \frac{1 + 0.5}{0 + 1 + 0.5 + 0.5} = 0.75.$$

As  $u_{C_2}(x_{11}) > u_{C_1}(x_{11})$ ,  $x_{11}$  will be classified as rectangle ( $C_2$ ).

Special care has to be devoted to anti-hubs, such as instances 4 and 5 in Fig. 11.6. Their occurrence fuzziness is estimated as the average fuzziness of points from the same class. Optional distance-based vote weighting is possible.

### 11.5.3 NHBNN: Naive Hubness Bayesian $k$ -Nearest Neighbor

Each  $k$ -occurrence can be treated as a random event. What NHBNN [53] does is that it essentially performs a Naive-Bayesian inference based on these  $k$  events

$$P(y^* = C | \mathcal{N}_k(x^*)) \propto P(C) \prod_{x_i \in \mathcal{N}_k(x^*)} P(x_i \in \mathcal{N}_k | C), \quad (11.12)$$

where  $P(C)$  denotes the probability that an instance belongs to class  $C$  and  $P(x_i \in \mathcal{N}_k | C)$  denotes the probability that  $x_i$  appears as one of the  $k$  nearest neighbors of any instance belonging to class  $C$ . From the data,  $P(C)$  can be estimated as

$$P(C) \approx \frac{|\mathcal{D}_C^{train}|}{|\mathcal{D}^{train}|}, \quad (11.13)$$

where  $|\mathcal{D}_C^{train}|$  denotes the number of train instances belonging to class  $C$  and  $|\mathcal{D}^{train}|$  is the total number of train instances.  $P(x_i \in \mathcal{N}_k | C)$  can be estimated as the fraction

$$P(x_i \in \mathcal{N}_k | C) \approx \frac{N_{k,C}(x_i)}{|\mathcal{D}_C^{train}|}. \quad (11.14)$$

*Example 3* Next, we illustrate NHBNN on the example shown in Fig. 11.6. Out of all the 10 training instances, 6 belong to the class of circles ( $C_1$ ) and 4 belong to the class of rectangles ( $C_2$ ). Therefore:

$$|\mathcal{D}_{C_1}^{train}| = 6, \quad |\mathcal{D}_{C_2}^{train}| = 4, \quad P(C_1) = 0.6, \quad P(C_2) = 0.4.$$

Similarly to the previous sections, we calculate  $N_{k,C}(x_i)$  using  $k = 1$ , but we classify instance 11 using  $k' = 2$  nearest neighbors, i.e.,  $x_6$  and  $x_9$ . Thus, we calculate (11.14) for  $x_6$  and  $x_9$  for both classes  $C_1$  and  $C_2$ :

$$\begin{aligned} P(x_6 \in \mathcal{N}_1 | C_1) &\approx \frac{N_{1,C_1}(x_6)}{|\mathcal{D}_{C_1}^{train}|} = \frac{0}{6} = 0, \quad P(x_6 \in \mathcal{N}_1 | C_2) \approx \frac{N_{1,C_2}(x_6)}{|\mathcal{D}_{C_2}^{train}|} = \frac{2}{4} = 0.5, \\ P(x_9 \in \mathcal{N}_1 | C_1) &\approx \frac{N_{1,C_1}(x_9)}{|\mathcal{D}_{C_1}^{train}|} = \frac{1}{6} = 0.167, \quad P(x_9 \in \mathcal{N}_1 | C_2) \approx \frac{N_{1,C_2}(x_9)}{|\mathcal{D}_{C_2}^{train}|} = \frac{1}{4} = 0.25. \end{aligned}$$

According to (11.12):

$$\begin{aligned} P(y_{11} = C_1 | \mathcal{N}_2(x_{11})) &\propto 0.6 \times 0 \times 0.167 = 0 \\ P(y_{11} = C_2 | \mathcal{N}_2(x_{11})) &\propto 0.4 \times 0.5 \times 0.25 = 0.125 \end{aligned}$$

As  $P(y_{11} = C_2 | \mathcal{N}_2(x_{11})) > P(y_{11} = C_1 | \mathcal{N}_2(x_{11}))$ , instance 11 will be classified as rectangle.

The previous example also illustrates that estimating  $P(x_i \in \mathcal{N}_k | C)$  according to (11.14) may simply lead to zero probabilities. In order to avoid it, instead of (11.14), we can estimate  $P(x_i \in \mathcal{N}_k | C)$  as

$$P(x_i \in \mathcal{N}_k | C) \approx (1 - \varepsilon) \frac{N_{k,C}(x_i)}{|\mathcal{D}_C^{train}|} + \varepsilon, \quad (11.15)$$

where  $\varepsilon \ll 1$ .

Even though  $k$ -occurrences are highly correlated, NHBNN still offers some improvement over the basic  $k$ NN. It is known that the Naive Bayes rule can sometimes deliver good results even in cases with high independence assumption violation [44].

Anti-hubs, i.e., instances that occur never or with an exceptionally low frequency as nearest neighbors, are treated as a special case. For an anti-hub  $x_i$ ,  $P(x_i \in \mathcal{N}_k | C)$  can be estimated as the average of class-dependent occurrence probabilities of non-anti-hub instances belonging to the same class as  $x_i$ :

$$P(x_i \in \mathcal{N}_k | C) \approx \frac{1}{|\mathcal{D}_{class(x_i)}^{train}|} \sum_{x_j \in \mathcal{D}_{class(x_i)}^{train}} P(x_j \in \mathcal{N}_k | C). \quad (11.16)$$

For more advanced techniques for the treatment of anti-hubs we refer to [53].

### 11.5.4 HIKNN: Hubness Information $k$ -Nearest Neighbor

In h-FNN, as in most  $k$ NN classifiers, all neighbors are treated as equally important. The difference is sometimes made by introducing the dependency on the distance to  $x^*$ , the instance to be classified. However, it is also possible to deduce some sort of global neighbor relevance, based on the occurrence model—and this is what HIKNN was based on [50]. It embodies an information-theoretic interpretation of the neighbor occurrence events. In that context, rare occurrences have higher self-information, see (11.17). These more informative instances are favored by the algorithm. The reasons for this lie hidden in the geometry of high-dimensional feature spaces. Namely, hubs have been shown to lie closer to the cluster centers [55], as most high-dimensional data lies approximately on hyper-spheres. Therefore, hubs are points that are somewhat less ‘local’. Therefore, favoring the rarely occurring points helps in consolidating the neighbor set locality. The algorithm itself is a bit more complex, as it not only reduces the vote weights based on the occurrence frequencies, but also modifies the fuzzy vote itself—so that the rarely occurring points vote mostly by their labels and the hub points vote mostly by their occurrence profiles. Next, we will present the approach in more detail.

The self-information  $I_{x_i}$  associated with the event that  $x_i$  occurs as one of the nearest neighbors of an instance to be classified can be calculated as

$$I_{x_i} = \log \frac{1}{P(x_i \in \mathcal{N}_k)}, \quad P(x_i \in \mathcal{N}_k) \approx \frac{N_k(x_i)}{|\mathcal{D}^{train}|}. \quad (11.17)$$

Occurrence self-information is used to define the relative and absolute relevance factors in the following way:

$$\alpha(x_i) = \frac{I_{x_i} - \min_{x_j \in \mathcal{N}_k(x_i)} I_{x_j}}{\log |\mathcal{D}^{train}| - \min_{x_j \in \mathcal{N}_k(x_i)} I_{x_j}}, \quad \beta(x_i) = \frac{I_{x_i}}{\log |\mathcal{D}^{train}|}. \quad (11.18)$$

The final fuzzy vote of a neighbor  $x_i$  combines the information contained in its label with the information contained in its occurrence profile. The relative relevance factor is used for weighting the two information sources. This is shown in (11.19).

$$P_k(y^* = C|x_i) \approx \begin{cases} \alpha(x_i) + (1 - \alpha(x_i)) \cdot u_C(x_i), & y_i = C \\ (1 - \alpha(x_i)) \cdot u_C(x_i), & y_i \neq C \end{cases} \quad (11.19)$$

where  $y_i$  denotes the class label of  $x_i$ , for the definition of  $u_C(x_i)$  see (11.10).

The final class assignments are given by the weighted sum of these fuzzy votes. The final vote of class  $C$  for the classification of instance  $x^*$  is shown in (11.20). The distance weighting factor  $d_w(x_i)$  yields mostly minor improvements and can be left out in practice, see [54] for more details.

$$u_C(x^*) \propto \sum_{x_i \in \mathcal{N}_k(x^*)} \beta(x_i) \cdot d_w(x_i) \cdot P_k(y^* = C|x_i). \quad (11.20)$$

*Example 4* Next, we illustrate HIKNN by showing how HIKNN classifies instance 11 of the example shown in Fig. 11.6. Again, we use  $k' = 2$  nearest neighbors to classify instance 11, but we use  $N_1(x_i)$  values calculated with  $k = 1$ . The both nearest neighbors of instance 11 are  $x_6$  and  $x_9$ . The self-information associated with the occurrence of these instances as nearest neighbors:

$$\begin{aligned} P(x_6 \in \mathcal{N}_1) &= \frac{2}{10} = 0.2, \quad I_{x_6} = \log_2 \frac{1}{0.2} = \log_2 5, \\ P(x_9 \in \mathcal{N}_1) &= \frac{2}{10} = 0.2, \quad I_{x_9} = \log_2 \frac{1}{0.2} = \log_2 5. \end{aligned}$$

The relevance factors are:

$$\alpha(x_6) = \alpha(x_9) = 0, \quad \beta(x_6) = \beta(x_9) = \frac{\log_2 5}{\log_2 10}.$$

The fuzzy votes according to (11.19):

$$\begin{aligned} P_k(y^* = C_1|x_6) &= u_{C_1}(x_6) = 0, \quad P_k(y^* = C_2|x_6) = u_{C_2}(x_6) = 1, \\ P_k(y^* = C_1|x_9) &= u_{C_1}(x_9) = 0.5, \quad P_k(y^* = C_2|x_9) = u_{C_2}(x_9) = 0.5. \end{aligned}$$

The sum of fuzzy votes (without taking the distance weighting factor into account):

$$\begin{aligned} u_{C_1}(x_{11}) &= \frac{\log_2 5}{\log_2 10} \cdot 0 + \frac{\log_2 5}{\log_2 10} \cdot 0.5, \\ u_{C_2}(x_{11}) &= \frac{\log_2 5}{\log_2 10} \cdot 1 + \frac{\log_2 5}{\log_2 10} \cdot 0.5. \end{aligned}$$

As  $u_{C_2}(x_{11}) > u_{C_1}(x_{11})$ , instance 11 will be classified as rectangle ( $C_2$ ).

### 11.5.5 Experimental Evaluation of Hubness-Aware Classifiers

Time series datasets exhibit a certain degree of hubness, as shown in Table 11.3. This is in agreement with previous observations [43].

Most datasets from the UCR repository [28] are balanced, with close-to-uniform class distributions. This can be seen by analyzing the relative imbalance factor (RImb) of the label distribution which we define as the normalized standard deviation of the class probabilities from the absolutely homogenous mean value of  $1/m$ , where  $m$  denotes the number of classes, i.e.,  $m = |\mathcal{C}|$ :

**Table 11.3** The summary of relevant properties of the UCR time series datasets

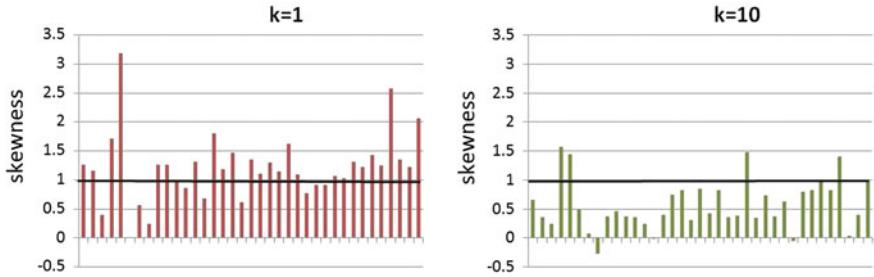
Data set	Size	$ \mathcal{C} $	$\mathcal{S}_{N_1(x)}$	$BN_1^{\%}(\%)$	$\max N_1(x)$	$\mathcal{S}_{N_{10}(x)}$	$BN_{10}^{\%}(\%)$	$\max N_{10}(x)$	Rlmb	$P(c_M) \%$
50words	905	50	1.26	19.6	5	0.66	36.2	33	0.16	4.2
Adiac	781	37	1.16	33.5	6	0.36	51.8	28	0.02	3.7
Beef	60	5	0.40	50	3	0.25	62	18	0.0	20.0
Car	120	4	1.72	24.2	6	1.57	39.2	42	0.0	25
CBF	930	3	3.18	0.0	22	1.44	0.1	57	0.0	33.3
ChlorineConcentration	4307	3	0.0	0	2	0.50	31.6	21	0.30	53.6
CinC-ECG-torso	1420	4	0.57	0.0	5	0.08	0.1	25	0.0	25
Coffee	56	2	0.25	3.6	3	-0.27	31.6	19	0.04	51.8
Cricket-X	780	12	1.26	16.7	6	0.38	33.1	28	0.0	8.3
Cricket-Y	780	12	1.26	18.2	6	0.47	34.9	30	0.0	8.3
Cricket-Z	780	12	1.00	15.9	5	0.37	33.2	27	0.0	8.3
DiatomSizeReduction	332	4	0.86	0.1	5	0.36	0.1	23	0.19	30.7
ECG200	200	2	1.31	12.0	6	0.24	19.7	25	0.33	66.5
ECGFiveDays	884	2	0.68	0.01	4	-0.01	0.04	25	0.0	50
FaceFour	112	4	1.80	4.5	7	0.40	13.5	26	0.09	30.4
FacesUCR	2250	14	1.19	1.4	6	0.75	5.3	36	0.12	14.5
FISH	350	7	1.47	18.6	6	0.83	33.5	36	0.0	14.3
Gun-Point	200	2	0.62	2.0	4	0.31	5.2	23	0.0	50.0
Haptics	463	5	1.36	53.6	6	0.85	60.9	35	0.04	21.6
InlineSkate	650	7	1.11	43.1	6	0.42	59.3	28	0.09	18.0
ItalyPowerDemand	1096	2	1.30	4.5	6	0.83	5.1	46	0.01	50.1

(continued)

**Table 11.3** (continued)

Data set	Size	$ \mathcal{C} $	$\mathcal{S}_{N_1(x)}$	$BN_1\%(\%)$	$\max N_1(x)$	$\mathcal{S}_{N_{10}(x)}$	$BN_{10}\%(\%)$	$\max N_{10}(x)$	Rimb	$P(c_M)(\%)$
Lighting2	121	2	1.15	9.1	5	0.36	28.8	23	0.206	60.3
Lighting7	143	7	1.63	21.0	6	0.39	39.1	26	0.15	26.6
MALLAT	2400	8	1.09	1.3	6	1.48	2.7	53	0.0	12.5
MedicalImages	1141	10	0.78	18.2	4	0.35	31.6	26	0.48	52.1
MoieStrain	1272	2	0.91	5.0	6	0.73	9.3	33	0.08	53.9
OliveOil	60	4	0.92	11.7	4	0.38	28.0	23	0.26	41.7
OSULeaf	442	6	1.07	25.3	5	0.63	44.8	29	0.11	21.9
Plane	210	7	1.03	0.0	5	-0.05	0.1	21	0.0	14.3
SonyAIBORobotSurface	621	2	1.32	1.9	7	0.80	4.4	33	0.12	56.2
SonyAIBORobotSurfaceII	980	2	1.22	1.5	6	0.82	6.5	35	0.23	61.6
SwedishLeaf	1125	15	1.43	14.7	8	0.97	23.5	41	0.0	6.7
Symbols	1020	6	1.25	1.8	6	0.83	3.0	38	0.02	17.7
Synthetic-control	600	6	2.58	0.7	12	1.40	2.0	54	0.0	16.7
Trace	200	4	1.36	0.0	6	0.04	2.5	22	0.0	25
TwoLeadECG	1162	2	1.22	0.1	6	0.40	0.3	33	0.0	50
Two-Patterns	5000	4	2.07	0.0	14	1.01	0.1	46	0.02	26.1
Average	917.6	7.6	1.21	11.7	6.24	0.58	21.2	31.54	0.08	31.0

Each dataset is described both by a set of basic properties (size, number of classes) and some hubness-related quantities for two different neighborhood sizes, namely: the skewness of the  $k$ -occurrence distribution ( $\mathcal{S}_{N_k(x)}$ ), the percentage of bad  $k$ -occurrences ( $BN_k\%$ ), the degree of the largest hub-point ( $\max N_k(x)$ ). Also, the relative imbalance of the label distribution is given, as well as the size of the majority class (expressed as a percentage of the entire dataset).



**Fig. 11.7** The skewness of the neighbor occurrence frequency distribution for neighborhood sizes  $k = 1$  and  $k = 10$ . In both figures, each column corresponds to a dataset of the UCR repository. The figures show the change in the skewness, when  $k$  is increased from 1 to 10

$$\text{RImb} = \sqrt{\frac{\sum_{C \in \mathcal{C}} (P(C) - 1/m)^2}{(m-1)/m}}. \quad (11.21)$$

In general, an occurrence frequency distribution skewness above 1 indicates a significant impact of hubness. Many UCR datasets have  $\mathcal{S}_{N_1(x)} > 1$ , which means that the first nearest neighbor occurrence distribution is significantly skewed to the right. However, an increase in neighborhood size reduces the overall skewness of the datasets, as shown in Fig. 11.7. Note that only a few datasets have  $\mathcal{S}_{N_{10}(x)} > 1$ , though some non-negligible skewness remains in most of the data. Yet, even though the overall skewness is reduced with increasing neighborhood sizes, the degree of major hubs in the data increases. This leads to the emergence of strong centers of influence.

We evaluated the performance of different  $k$ NN classification methods on time series data for a fixed neighborhood size of  $k = 10$ . A slightly larger  $k$  value was chosen, since most hubness-aware methods are known to perform better in such cases, as better and more reliable neighbor occurrence models can be inferred from more occurrence information. We also analyzed the algorithm performance over a range of different neighborhood sizes, as shown in Fig. 11.8. The hubness-aware classification methods presented in the previous sections (hw- $k$ NN, NHBNN, h-FNN and HIKNN) were compared to the baseline  $k$ NN [15] and the adaptive  $k$ NN (AKNN) [56], where the neighborhood size is recalculated for each query point based on initial observations, in order to consult only the relevant neighbor points. AKNN does not take the hubness of the data into account.

The tests were run according to the 10-times 10-fold cross-validation protocol and the statistical significance was determined by employing the corrected re-sampled  $t$ -test. The detailed results are given in Table 11.4.

The adaptive neighborhood approach (AKNN) does not seem to be appropriate for handling time-series data, as it performs worse than the baseline  $k$ NN. While hw- $k$ NN, NHBNN and h-FNN are better than the baseline  $k$ NN in some cases, they do not offer significant advantage overall which is probably a consequence of a relatively low neighbor occurrence skewness for  $k = 10$  (see Fig. 11.7). The hubness

**Table 11.4** Accuracy  $\pm$  standard deviation (in %) for hubness-aware classifiers

Data set	$k$ NN	AKNN	hw- $k$ NN	NHBNN	hFNN	HIKNN
50words	72.4 $\pm$ 3.2	68.8 $\pm$ 3.3 •	71.3 $\pm$ 3.3	74.1 $\pm$ 2.7	76.4 $\pm$ 3.0 ○	<b>80.1</b> $\pm$ <b>2.9</b> ○
Adiac	61.0 $\pm$ 4.1	58.6 $\pm$ 3.3	60.7 $\pm$ 4.0	63.8 $\pm$ 3.9	63.1 $\pm$ 3.8	<b>65.9</b> $\pm$ <b>4.0</b> ○
Beef	<b>51.5</b> $\pm$ <b>15.6</b>	31.8 $\pm$ 15.0 •	43.9 $\pm$ 14.3	50.4 $\pm$ 15.2	47.6 $\pm$ 15.6	47.8 $\pm$ 15.8
Car	71.7 $\pm$ 9.8	73.9 $\pm$ 9.4	76.0 $\pm$ 9.4	76.0 $\pm$ 8.6	76.8 $\pm$ 9.2	<b>79.1</b> $\pm$ <b>8.7</b> ○
CBF	<b>100.0</b> $\pm$ <b>0.0</b>					
Chlorine <sup>a</sup>	87.6 $\pm$ 1.3	68.7 $\pm$ 1.6 •	74.5 $\pm$ 1.6 •	68.7 $\pm$ 1.6 •	80.2 $\pm$ 1.4 •	<b>91.8</b> $\pm$ <b>1.0</b> ○
CinC-ECG-torso	99.8 $\pm$ 0.2	99.6 $\pm$ 0.3	99.7 $\pm$ 0.2	99.6 $\pm$ 0.3	<b>99.9</b> $\pm$ <b>0.1</b>	<b>99.9</b> $\pm$ <b>0.1</b>
Coffee	74.4 $\pm$ 15.9	68.7 $\pm$ 15.4	68.6 $\pm$ 15.1	71.2 $\pm$ 15.3	70.0 $\pm$ 15.2	<b>76.9</b> $\pm$ <b>13.4</b>
Cricket-X	78.0 $\pm$ 2.9	71.0 $\pm$ 2.9 •	77.9 $\pm$ 2.6	78.2 $\pm$ 3.0	79.3 $\pm$ 2.8	<b>81.9</b> $\pm$ <b>2.6</b> ○
Cricket-Y	77.7 $\pm$ 3.1	67.1 $\pm$ 3.1 •	76.7 $\pm$ 3.3	77.5 $\pm$ 3.2	79.9 $\pm$ 3.1	<b>81.6</b> $\pm$ <b>2.9</b> ○
Cricket-Z	78.0 $\pm$ 3.1	70.9 $\pm$ 3.3 •	78.1 $\pm$ 3.1	78.6 $\pm$ 3.0	79.8 $\pm$ 3.0	<b>82.2</b> $\pm$ <b>2.8</b> ○
Diatom <sup>b</sup>	98.5 $\pm$ 1.6	<b>100.0</b> $\pm$ <b>0.0</b>	98.7 $\pm$ 1.4	99.1 $\pm$ 1.1	99.6 $\pm$ 0.8	99.6 $\pm$ 0.7
ECG200	85.9 $\pm$ 5.8	83.0 $\pm$ 4.9	85.3 $\pm$ 5.1	83.9 $\pm$ 4.6	85.2 $\pm$ 5.0	<b>87.1</b> $\pm$ <b>5.0</b>
ECGFiveDays	98.5 $\pm$ 0.7	97.7 $\pm$ 1.0 •	98.3 $\pm$ 1.0	98.3 $\pm$ 0.9	98.8 $\pm$ 0.7	<b>99.0</b> $\pm$ <b>0.6</b>
FaceFour	91.1 $\pm$ 6.3	92.2 $\pm$ 5.7	91.8 $\pm$ 6.7	94.0 $\pm$ 5.1	93.6 $\pm$ 5.4	<b>94.1</b> $\pm$ <b>5.3</b>
FacesUCR	96.4 $\pm$ 0.7	95.9 $\pm$ 0.8	96.7 $\pm$ 0.8	97.1 $\pm$ 0.7	97.5 $\pm$ 0.7	<b>98.0</b> $\pm$ <b>0.6</b>
FISH	77.3 $\pm$ 5.1	72.5 $\pm$ 5.7 •	77.1 $\pm$ 5.3	77.1 $\pm$ 5.5	77.9 $\pm$ 5.1	<b>81.7</b> $\pm$ <b>4.5</b> ○
Gun-Point	<b>99.1</b> $\pm$ <b>1.4</b>	95.4 $\pm$ 3.4 •	98.2 $\pm$ 2.0	97.9 $\pm$ 2.3	98.8 $\pm$ 1.6	99.0 $\pm$ 1.4
Haptics	50.5 $\pm$ 4.8	52.6 $\pm$ 5.5	51.3 $\pm$ 4.8	50.1 $\pm$ 4.8	51.9 $\pm$ 4.7	<b>54.3</b> $\pm$ <b>4.6</b> ○
InlineSkate	55.0 $\pm$ 3.8	44.8 $\pm$ 3.9 •	53.1 $\pm$ 3.7	53.6 $\pm$ 3.8	54.3 $\pm$ 4.0	<b>60.1</b> $\pm$ <b>4.4</b> ○
Italy <sup>c</sup>	96.5 $\pm$ 1.1	<b>96.9</b> $\pm$ <b>1.1</b>	96.7 $\pm$ 1.1	<b>96.9</b> $\pm$ <b>1.0</b>	96.8 $\pm$ 1.0	

(continued)

Table 11.4 (continued)

Data set	<i>k</i> NN	AKNN	hw- <i>k</i> NN	NHBNN	hFNN	HkNN
Lighting2	79.5 ± 8.2	76.5 ± 8.6	78.1 ± 8.2	74.4 ± 8.9	80.0 ± 7.6	<b>83.9 ± 7.2</b> ○
Lighting7	69.2 ± 9.1	65.9 ± 9.1	70.0 ± 9.5	74.3 ± 9.0	71.5 ± 8.7	<b>76.3 ± 7.8</b> ○
MALLAT	98.5 ± 0.5	98.4 ± 0.5	98.5 ± 0.5	98.4 ± 0.5	98.5 ± 0.4	<b>98.8 ± 0.4</b>
MedicalImages	79.2 ± 2.7	73.6 ± 3.1 ●	78.9 ± 2.9	69.2 ± 2.5 ●	79.2 ± 2.8	<b>81.3 ± 2.6</b> ○
MoteStrain	94.0 ± 1.4	94.3 ± 1.3	94.7 ± 1.4	94.3 ± 1.4	94.7 ± 1.3	<b>95.4 ± 1.2</b> ○
OliveOil	76.2 ± 13.7	73.4 ± 12.5	78.9 ± 13.0	<b>87.9 ± 9.9</b> ○	78.6 ± 12.6	87.1 ± 10.7○
OSULeaf	67.0 ± 5.0	60.1 ± 5.4 ●	66.0 ± 5.1	64.1 ± 5.1	66.1 ± 4.5	<b>72.5 ± 4.9</b> ○
Plane	<b>99.7 ± 0.7</b>	98.9 ± 1.6	99.4 ± 1.0	99.5 ± 0.9	<b>99.7 ± 0.7</b>	<b>99.7 ± 0.7</b>
Sony <sup>d</sup>	96.7 ± 1.5	<b>98.9 ± 1.0</b> ○	97.8 ± 1.2 ○	97.4 ± 1.3 ○	97.7 ± 1.3 ○	97.5 ± 1.3 ○
SonyII <sup>e</sup>	95.8 ± 1.4	94.8 ± 1.6	96.0 ± 1.4	95.8 ± 1.4	96.3 ± 1.3	<b>96.5 ± 1.2</b>
SwedishLeaf	82.8 ± 2.1	82.6 ± 2.6	84.8 ± 2.3 ○	85.8 ± 2.2 ○	85.4 ± 2.3 ○	<b>86.0 ± 2.2</b> ○
Symbols	97.4 ± 1.0	97.1 ± 1.1	97.6 ± 0.9	97.6 ± 0.9	98.0 ± 0.8 ○	<b>98.2 ± 0.8</b> ○
Synthetic-control	99.0 ± 0.7	97.2 ± 1.4 ●	<b>99.5 ± 0.6</b>	99.2 ± 0.7	99.2 ± 0.7	99.0 ± 0.7
Trace	98.9 ± 1.6	98.4 ± 2.0	98.4 ± 2.1	98.7 ± 1.8	99.7 ± 0.7	<b>99.9 ± 0.4</b>
TwoLeadECG	99.8 ± 0.2	99.7 ± 0.2	99.8 ± 0.2	99.8 ± 0.2	<b>99.9 ± 0.1</b>	<b>99.9 ± 0.1</b>
Two-Patterns	99.9 ± 0.0	99.9 ± 0.0	99.9 ± 0.0	99.9 ± 0.0	<b>100.0 ± 0.0</b>	<b>100.0 ± 0.0</b>
Average	82.5	79.4	81.9	82.2	82.9	<b>84.9</b>

All experiments were performed for  $k = 10$ . The symbols ●/○ denote statistically significant worse/better performance ( $p < 0.05$ ) compared to  $k$ NN. The best result in each line is in bold.

<sup>a</sup>ChlorineConcentration

<sup>b</sup>DiatomSizeReduction

<sup>c</sup>ItalyPowerDemand

<sup>d</sup>SonyAIBORobotSurface

<sup>e</sup>SonyAIBORobotSurfaceII

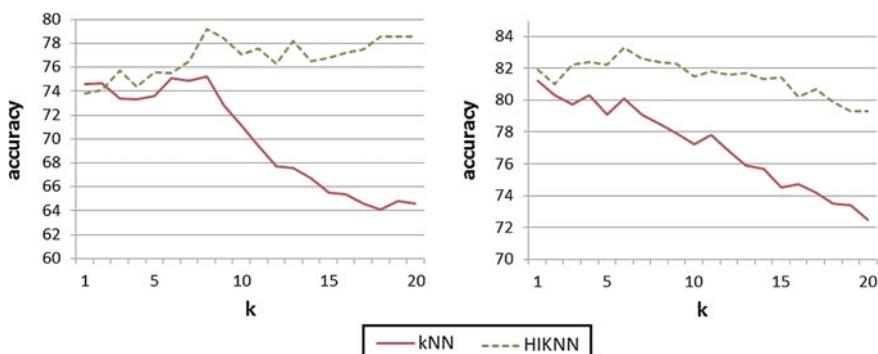
is, on average, present in time-series data to a lower extent than in text or images [49] where these methods were previously shown to perform rather well.

On the other hand, HIKNN, the information-theoretic approach to handling voting in high-dimensional data, clearly outperforms all other tested methods on these time series datasets. It performs significantly better than the baseline in 19 out of 37 cases and does not perform significantly worse on any examined dataset. Its average accuracy for  $k = 10$  is 84.9, compared to 82.5 achieved by  $k$ NN. HIKNN outperformed both baselines (even though not significantly) even in case of the ChlorineConcentration dataset, which has very low hubness in terms of skewness, and therefore other hubness-aware classifiers worked worse than  $k$ NN on this data. These observations reaffirm the conclusions outlined in previous studies [50], arguing that HIKNN might be the best hubness-aware classifier on medium-to-low hubness data, if there is no significant class imbalance. Note, however, that hubness-aware classifiers are also well suited for learning under class imbalance [16, 20, 21].

In order to show that the observed improvements are not merely an artifact of the choice of neighborhood size, classification tests were performed for a range of different neighborhood sizes. Figure 11.8 shows the comparisons between  $k$ NN and HIKNN for  $k \in [1, 20]$ , on Car and Fish time series datasets. There is little difference between  $k$ NN and HIKNN for  $k = 1$  and the classifiers performance is similar in this case. However, as  $k$  increases, so does the performance of HIKNN, while the performance of  $k$ NN either decreases or increases at a slower rate. Therefore, the differences for  $k = 10$  are more pronounced and the differences for  $k = 20$  are even greater. Most importantly, the highest achieved accuracy by HIKNN, over all tested neighborhoods, is clearly higher than the highest achieved accuracy by  $k$ NN.

These results indicate that HIKNN is an appropriate classification method for handling time series data, when used in conjunction with the dynamic time warping distance.

Of course, choosing the optimal neighborhood size in  $k$ -nearest neighbor methods is a non-trivial problem. The parameter could be set by performing cross-validation on the training data, though this is quite time-consuming. If the data is small, using



**Fig. 11.8** The accuracy (in %) of the basic  $k$ NN and the hubness aware HIKNN classifier over a range of neighborhood sizes  $k \in [1, 20]$ , on Car and Fish time series datasets

large  $k$  values might not make a lot of sense, as it would breach the locality assumption by introducing neighbors into the  $k$ NN sets that are not relevant for the instance to be classified. According to our experiments, HIKNN achieved very good performance for  $k \in [5, 15]$ , therefore, setting  $k = 5$  or  $k = 10$  by default would usually lead to reasonable results in practice.

## 11.6 Instance Selection and Feature Construction for Time-Series Classification

In the previous section, we described four approaches that take hubness into account in order to make time-series classification more accurate. In various applications, however, besides classification accuracy, the classification time is also important. Therefore, in this section, we present hubness-aware approaches for speeding-up time-series classification. First, we describe instance selection for  $k$ NN classification of time-series. Subsequently, we focus on feature construction.

### 11.6.1 Instance Selection for Speeding-Up Time-Series Classification

Attempts to speed up DTW-based nearest neighbor classification fall into four major categories: (i) speeding-up the calculation of the distance of two time series (by e.g. limiting the warping window size), (ii) indexing, (iii) reducing the length of the time series used, and (iv) instance selection. The first class of techniques was already mentioned in Sect. 11.3. For an overview of techniques for indexing and reduction of the length of time-series and more advanced approaches for limiting the warping window size, we refer to [8] and the references therein. In this section, we focus on how to speed up time-series classification via instance selection. We note that instance selection is orthogonal to the other speed-up techniques, i.e., instance selection can be combined with those techniques in order to achieve highest efficiency.

Instance selection (also known as  *numerosity reduction* or  *prototype selection*) aims at discarding most of the training time series while keeping only the most informative ones, which are then used to classify unlabeled instances. In case of conventional nearest neighbor classification, the instance to be classified, denoted as  $x^*$ , will be compared to all the instances of the training data set. In contrast, when applying instance selection,  $x^*$  will only be compared to the selected instances of the training data. For time-series classification, despite the techniques aiming at speeding-up DTW-calculations, the calculation of the DTW distance is still relatively expensive computationally, therefore, when selecting a relatively small number of instances, such as 10% of the training data, instance selection can substantially speed-up the classification of time-series.

While instance selection is well explored for general nearest neighbor classification, see e.g. [1, 6, 19, 24, 32], there are only few works for the case of time series. Xi et al. [57] presented the FastAWARD approach and showed that it outperforms state-of-the-art, general-purpose instance selection techniques applied for time-series.

FastAWARD follows an iterative procedure for discarding time series: in each iteration, the rank of all the time series is calculated and the one with lowest rank is discarded. Thus, each iteration corresponds to a particular number of kept time series. Furthermore, Xi et al. argue that the optimal warping window size depends on the number of kept time series. Therefore, FastAWARD calculates the optimal warping window size dependent on the number of kept time series.

In this section, we present a hubness-aware instance selection technique which was originally introduced in [9]. This approach is simpler and therefore computationally much cheaper than FastAWARD while it selects better instances, i.e., instances that allow more accurate classification of time-series than the ones selected by FastAWARD.

In [9] coverage graphs were proposed to model instance selection, and the instance selection problem was formulated as finding the appropriate subset of vertices of the coverage graph. Furthermore, it was shown that maximizing the coverage is NP-complete in general. On the other hand, for the case of time-series classification, a simple approach performed surprisingly well. This approach is called *Instance Selection based on Graph-coverage and Hubness for Time-series* or INSIGHT.

INSIGHT performs instance selection by assigning a score to each instance and selecting instances with the highest scores (see Algorithm 4), therefore the “intelligence” of INSIGHT is hidden in the applied score function. Next, we explain the suitability of several score functions in the light of the hubness property.

- **Good 1-occurrence Score**—INSIGHT can use scores that take into account how many times an instance appears as good neighbor of other instances. Thus, a simple score function is the *good 1-occurrence score*  $GN_1(x)$ .
- **Relative Score**—While  $x$  is being a good hub, at the same time it may appear as bad neighbor of several other instances. Thus, INSIGHT can also consider scores that take bad occurrences into account. This leads to scores that relate the good occurrence of an instance  $x$  to either its total occurrence or to its bad occurrence. For simplicity, we focus on the following *relative score*, however, other variations could be used too: *relative score*  $RS(x)$  of a time series  $x$  is the fraction of good 1-occurrences and total occurrences plus one (to avoid division by zero):

$$RS(x) = \frac{GN_1(x)}{N_1(x) + 1}. \quad (11.22)$$

- **Xi’s score**—Notably,  $GN_k(x)$  and  $BN_k(x)$  allows us to interpret the ranking criterion used by Xi et al. in FastAWARD [57] as another form of score for relative hubness:

$$XI(x) = GN_1(x) - 2BN_1(x). \quad (11.23)$$

**Algorithm 4 INSIGHT**


---

**Require:** Time series dataset  $\mathcal{D}$ , Score Function  $g(x)$  /\* e.g. one of  $GN_1(x)$ ,  $RS(x)$  or  $XI(x)$  \*/,  
Number of selected instances  $n_{sel}$   
**Ensure:** Set of selected instances (time series)  $\mathcal{D}'$

- 1: Calculate score function  $g(x)$  for all  $x \in \mathcal{D}$
- 2: Sort all the time series in  $\mathcal{D}$  according to their scores  $g(x)$
- 3: Select the top-ranked  $n_{sel}$  time series and return the set containing them

---

As reported in [9], INSIGHT outperforms FastAWARD both in terms of classification accuracy and execution time. The second and third columns of Table 11.5 present the average accuracy and corresponding standard deviation for each data set, for the case when the number of selected instances is equal to 10% of the size of the training set. The experiments were performed according to the 10-fold cross-validation protocol. For INSIGHT, the good 1-occurrence score is used, but we note that similar results were achieved for the other scores too.

In clear majority of the cases, INSIGHT substantially outperformed FastAWARD. In the few remaining cases, their differences are remarkably small (which are not significant in the light of the corresponding standard deviations). According to the analysis reported in [9], one of the major reasons for the suboptimal performance of FastAWARD is that the skewness degrades during the FastAWARD's iterative instance selection procedure, and therefore FastAWARD is not able to select the best instances in the end. This is crucial because FastAWARD discards the worst instance in each iteration and therefore the final iterations have substantial impact on which instances remain, i.e., which instances will be selected by FastAWARD.

### 11.6.2 Feature Construction

As shown in Sect. 11.6.1, the instance selection approach focusing on good hubs leads to overall good results. Previously, once the instances were selected, we simply used them as training data for the  $k$ NN classifier. In a more advanced classification schema, instead of simply performing nearest neighbor classification, we can use distances from selected instances as features. This is described in detail below.

First, we split the training data  $\mathcal{D}^{train}$  into two disjoint subsets  $\mathcal{D}_1^{train}$  and  $\mathcal{D}_2^{train}$ , i.e.,  $\mathcal{D}_1^{train} \cap \mathcal{D}_2^{train} = \emptyset$ ,  $\mathcal{D}_1^{train} \cup \mathcal{D}_2^{train} = \mathcal{D}^{train}$ . We select some instances from  $\mathcal{D}_1^{train}$ , denote these selected instances as  $x_{sel,1}, x_{sel,2}, \dots, x_{sel,l}$ . For each instance  $x \in \mathcal{D}_2^{train}$ , we calculate its DTW-distance from the selected instances and use these distances as features of  $x$ . This way, we map each instance  $x \in \mathcal{D}_2^{train}$  into a vector space:

$$x_{mapped} = (d_{DTW}(x, x_{sel,1}), d_{DTW}(x, x_{sel,2}), \dots, d_{DTW}(x, x_{sel,l})) . \quad (11.24)$$

**Table 11.5** Accuracy  $\pm$  standard deviation (in %) for FastAWARD, INSIGHT and feature construction methods

Data set	FastAWARD	INSIGHT	HubFeatures	RndFeatures
50words	52.6 $\pm$ 4.1	64.2 $\pm$ 4.6 $\circ$	<b>65.5 <math>\pm</math> 3.5 <math>\circ</math></b>	65.2 $\pm$ 4.9 $\circ$
Adiac	34.8 $\pm$ 5.8	46.9 $\pm$ 4.9 $\circ$	48.5 $\pm$ 4.8 $\circ$	<b>51.0 <math>\pm</math> 5.2 <math>\circ</math></b>
Beef	35.0 $\pm$ 17.4	33.3 $\pm$ 10.5	<b>38.3 <math>\pm</math> 19.3</b>	36.7 $\pm$ 15.3
Car	45.0 $\pm$ 11.9	<b>60.8 <math>\pm</math> 14.5 <math>\circ</math></b>	47.5 $\pm$ 22.9	55.8 $\pm$ 20.8
CBF	97.2 $\pm$ 3.4	<b>99.8 <math>\pm</math> 0.6 <math>\circ</math></b>	<b>99.8 <math>\pm</math> 0.5 <math>\circ</math></b>	<b>99.8 <math>\pm</math> 0.5 <math>\circ</math></b>
ChlorineConcentration	53.7 $\pm$ 2.3	<b>73.4 <math>\pm</math> 3.0 <math>\circ</math></b>	54.8 $\pm$ 1.4	54.6 $\pm$ 1.7
CinC-ECG-torso	40.6 $\pm$ 8.9	96.6 $\pm$ 1.4 $\circ$	<b>98.7 <math>\pm</math> 1.2 <math>\circ</math></b>	<b>98.7 <math>\pm</math> 1.2 <math>\circ</math></b>
Coffee	56.0 $\pm$ 30.9	60.3 $\pm$ 21.3	<b>66.0 <math>\pm</math> 21.4</b>	59.0 $\pm$ 16.1
DiatomSizeReduction	97.2 $\pm$ 2.6	96.6 $\pm$ 5.8	<b>100.0 <math>\pm</math> 0.0 <math>\circ</math></b>	98.8 $\pm$ 2.2
ECG200	75.5 $\pm$ 11.3	83.5 $\pm$ 9.0	<b>86.0 <math>\pm</math> 7.0 <math>\circ</math></b>	84.5 $\pm$ 7.6
ECGFiveDays	93.7 $\pm$ 2.7	94.5 $\pm$ 2.0	96.5 $\pm$ 2.2 $\circ$	<b>96.7 <math>\pm</math> 1.7 <math>\circ</math></b>
FaceFour	71.4 $\pm$ 14.1	89.4 $\pm$ 12.8 $\circ$	<b>91.1 <math>\pm</math> 8.4 <math>\circ</math></b>	90.2 $\pm$ 8.9 $\circ$
FacesUCR	89.2 $\pm$ 1.9	<b>93.4 <math>\pm</math> 2.1 <math>\circ</math></b>	91.4 $\pm$ 2.2 $\circ$	91.5 $\pm$ 1.8 $\circ$
FISH	59.1 $\pm$ 8.2	<b>66.6 <math>\pm</math> 8.5</b>	59.7 $\pm$ 6.5	62.9 $\pm$ 9.3
Gun-Point	80.0 $\pm$ 12.4	<b>93.5 <math>\pm</math> 5.9 <math>\circ</math></b>	85.5 $\pm$ 6.4	84.5 $\pm$ 3.7
Haptics	30.3 $\pm$ 6.8	<b>43.5 <math>\pm</math> 6.0 <math>\circ</math></b>	33.9 $\pm$ 9.4	35.4 $\pm$ 7.9
InlineSkate	19.7 $\pm$ 5.6	<b>43.4 <math>\pm</math> 7.7 <math>\circ</math></b>	36.3 $\pm$ 7.5 $\circ$	36.5 $\pm$ 8.7 $\circ$
ItalyPowerDemand	<b>96.0 <math>\pm</math> 2.0</b>	95.7 $\pm$ 2.8	95.8 $\pm$ 2.4	95.7 $\pm$ 2.1
Lighting2	69.4 $\pm$ 13.4	67.0 $\pm$ 9.6	67.7 $\pm$ 6.4	<b>75.1 <math>\pm</math> 8.9</b>
Lighting7	44.7 $\pm$ 12.6	51.0 $\pm$ 8.2	<b>61.4 <math>\pm</math> 10.5 <math>\circ</math></b>	60.8 $\pm$ 9.3 $\circ$
MALLAT	55.1 $\pm$ 9.8	<b>96.9 <math>\pm</math> 1.3 <math>\circ</math></b>	96.4 $\pm$ 1.5 $\circ$	96.8 $\pm$ 1.1 $\circ$
MedicalImages	64.2 $\pm$ 3.3	69.3 $\pm$ 4.9 $\circ$	<b>73.4 <math>\pm</math> 3.9 <math>\circ</math></b>	73.0 $\pm$ 3.3 $\circ$
MoteStrain	86.7 $\pm$ 4.2	90.8 $\pm$ 2.7 $\circ$	93.3 $\pm$ 2.4 $\circ$	<b>93.6 <math>\pm</math> 2.2 <math>\circ</math></b>
OliveOil	63.3 $\pm$ 10.0	71.7 $\pm$ 13.0	<b>80.0 <math>\pm</math> 17.2 <math>\circ</math></b>	75.0 $\pm$ 23.9 $\circ$
OSULeaf	41.9 $\pm$ 5.3	53.8 $\pm$ 5.7 $\circ$	<b>57.0 <math>\pm</math> 6.7 <math>\circ</math></b>	54.5 $\pm$ 8.1 $\circ$
Plane	87.6 $\pm$ 15.5	<b>98.1 <math>\pm</math> 3.2</b>	95.7 $\pm$ 5.2	94.8 $\pm$ 6.1
SonyAIBORobotSurface	92.4 $\pm$ 3.2	97.6 $\pm$ 1.7 $\circ$	98.4 $\pm$ 1.3 $\circ$	<b>98.7 <math>\pm</math> 1.3 <math>\circ</math></b>
SonyAIBORobotSurfaceII	91.9 $\pm$ 1.5	91.2 $\pm$ 3.3	94.6 $\pm$ 2.3 $\circ$	<b>95.1 <math>\pm</math> 2.8 <math>\circ</math></b>
SwedishLeaf	68.3 $\pm$ 4.6	75.6 $\pm$ 4.8 $\circ$	77.6 $\pm$ 5.1 $\circ$	<b>77.9 <math>\pm</math> 5.6 <math>\circ</math></b>
Symbols	95.7 $\pm$ 1.8	<b>96.6 <math>\pm</math> 1.6</b>	95.1 $\pm$ 2.1	95.6 $\pm$ 2.2
Synthetic-control	92.3 $\pm$ 6.8	<b>97.8 <math>\pm</math> 2.6 <math>\circ</math></b>	95.3 $\pm$ 2.6	94.0 $\pm$ 3.4
Trace	78.0 $\pm$ 11.7	<b>89.5 <math>\pm</math> 7.2 <math>\circ</math></b>	73.0 $\pm$ 8.6	74.0 $\pm$ 8.1
TwoLeadECG	97.8 $\pm$ 1.3	<b>98.9 <math>\pm</math> 1.2</b>	93.6 $\pm$ 2.8 $\bullet$	93.3 $\pm$ 2.9 $\bullet$
Two-Patterns	40.7 $\pm$ 2.7	<b>98.7 <math>\pm</math> 0.7 <math>\circ</math></b>	98.4 $\pm$ 0.5 $\circ$	98.4 $\pm$ 0.7 $\circ$
Wafer	92.1 $\pm$ 1.2	99.1 $\pm$ 0.2 $\circ$	<b>99.5 <math>\pm</math> 0.2 <math>\circ</math></b>	<b>99.5 <math>\pm</math> 0.2 <math>\circ</math></b>
WordsSynonyms	54.4 $\pm$ 5.8	63.7 $\pm$ 6.6 $\circ$	65.3 $\pm$ 3.9 $\circ$	<b>66.6 <math>\pm</math> 5.3 <math>\circ</math></b>
Yoga	55.0 $\pm$ 1.7	<b>87.7 <math>\pm</math> 2.1 <math>\circ</math></b>	86.4 $\pm$ 2.0 $\circ$	86.7 $\pm$ 1.6 $\circ$
Average	67.5	<b>79.2</b>	78.3	78.4

The symbols  $\bullet/\circ$  denote statistically significant worse/better performance ( $p < 0.05$ ) compared to FastAWARD. The best result in each line is in bold

The representation of the data in a vector space allows the usage of any conventional classifier. For our experiments, we trained logistic regression from the Weka software package.<sup>4</sup> We used the mapped instances of  $\mathcal{D}_2^{train}$  as training data for logistic regression.

When classifying an instance  $x^* \in \mathcal{D}^{test}$ , we map  $x^*$  into the same vector space as the instances of  $\mathcal{D}_2^{train}$ , i.e., we calculate the DTW-distances between  $x^*$  and the selected instances  $x_{sel,1}, x_{sel,2}, \dots, x_{sel,l}$  and we use these distances as features of  $x^*$ . Once the features of  $x^*$  are calculated, we use the trained classifier (logistic regression in our case) to classify  $x^*$ .

We used the 10-fold cross-validation to evaluate this feature construction-based approach. Similarly to the case of INSIGHT and FastAWARD, the number of selected instances corresponds to 10 % of the entire training data, however, as described previously, for the feature construction-based approach, we selected the instances from  $\mathcal{D}_1^{train}$  (not from  $\mathcal{D}_2^{train}$ ).

We tested several variants of the approach, for two out of them, the resulting accuracies are shown in the last two columns of Table 11.5. The results shown in the fourth column of Table 11.5 (denoted as HubFeatures) refer to the case when we performed hub-based instance selection on  $\mathcal{D}_1^{train}$  using good 1-occurrence score. The results shown in the last column of Table 11.5 (denoted as RndFeatures) refer to the case when the instances were randomly selected from  $\mathcal{D}_1^{train}$ .

Both HubFeatures and RndFeatures outperform FastAWARD in clear majority of the cases: while they are significantly better than FastAWARD for 23 and 21 data sets respectively, they are significantly worse only for one data set. INSIGHT, HubFeatures and RndFeatures can be considered as alternative approaches, as their overall performances are close to each other. Therefore, in a real-world application, one can use cross-validation to select the approach which best suits the particular application.

## 11.7 Conclusions and Outlook

We devoted this chapter to the recently observed phenomenon of hubness which characterizes numerous real-world data sets, especially high-dimensional ones. We surveyed hubness-aware classifiers and instance selection. Finally, we proposed a hubness-based feature construction approach. The approaches we reviewed were originally published in various research papers using slightly different notations and terminology. In this chapter, we presented all the approaches within an integrated framework using uniform notations and terminology. Hubness-aware classifiers were originally developed for vector classification. Here, we pointed out that these classifiers can be used for time-series classification given that an appropriate time-series distance measure is present. To the best of our knowledge, most of the surveyed approaches have not yet been used for time-series data. We performed extensive experimental evaluation of the state-of-the-art hubness-aware classifiers on a large

---

<sup>4</sup> <http://www.cs.waikato.ac.nz/ml/weka/>.

number of time-series data sets. The results of our experiments provide direct indications for the application of hubness-aware classifiers for real-world time-series classification tasks. In particular, the HIKNN approach seems to have the best overall performance for time-series data.

Furthermore, we pointed out that instance selection can substantially speed-up time-series classification and the recently introduced hubness-aware instance selection approach, INSIGHT, outperforms the previous state-of-the-art instance selection approach, FastAWARD, which did not take the presence of hubs explicitly into account. Finally, we showed that the selected instances can be used to construct features for instances of time-series data sets. While mapping time-series into a vector space by this feature construction approach is intuitive and leads to acceptable overall classification accuracy, the particular instance selection approach does not seem to play a major role in the procedure.

Future work may target the implications of hubness for feature construction approaches and how these features suit conventional classifiers. One would for example expect that monotone classifiers [2, 13, 23], benefit from hubness-based feature construction: the closer an instance is to a good hub, the more likely it belongs to the same class. Furthermore, regression methods may also benefit from taking the presence of hubs into account: e.g. hw- $k$ NN may simply be adapted for the case of nearest neighbor regression where the weighted average of the neighbors' class labels is taken instead of their weighted vote. Last but not least, due to the novelty of hubness-aware classifiers, there are still many applications in context of which hubness-aware classifiers have not been exploited yet, see e.g. [47] for recognition tasks related to literary texts. Also the classification of medical data, such as diagnosis of cancer subtypes based on gene expression levels [31], could potentially benefit from hubness-aware classification, especially classifiers taking class-imbalance into account [51].

**Acknowledgments** Research partially performed within the framework of the grant of the Hungarian Scientific Research Fund (grant No. OTKA 108947). The position of Krisztian Buza is funded by the Warsaw Center of Mathematics and Computer Science (WCMCS).

## References

1. Aha, D., Kibler, D., Albert, M.: Instance-based learning algorithms. *Mach. Learn.* **6**(1), 37–66 (1991)
2. Altendorf, E., Restificar, A., Dietterich, T.: Learning from sparse data by exploiting monotonicity constraints. In: Proceedings of the 21st Annual Conference on Uncertainty in Artificial Intelligence, pp. 18–26. AUAI Press, Arlington, Virginia (2005)
3. Barabási, A.: *Linked: How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life*. Plume, New York (2003)
4. Bellman, R.E.: *Adaptive Control Processes—A Guided Tour*. Princeton University Press, Princeton (1961)
5. Botsch, M.: *Machine Learning Techniques for Time Series Classification*. Cuvillier, Munchen (2009)

6. Brighton, H., Mellish, C.: Advances in instance selection for instance-based learning algorithms. *Data Min. Knowl. Discov.* **6**(2), 153–172 (2002)
7. Burges, C.J.: A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.* **2**(2), 121–167 (1998)
8. Buza, K.A.: *Fusion Methods for Time-Series Classification*. Peter Lang Verlag, New York (2011)
9. Buza, K., Nanopoulos, A., Schmidt-Thieme, L.: Insight: efficient and effective instance selection for time-series classification. In: Proceedings of the 15th Pacific-Asia Conference on Knowledge Discovery and Data Mining. Lecture Notes in Computer Science, vol. 6635, pp. 149–160. Springer (2011)
10. Chen, G.H., Nikolov, S., Shah, D.: A latent source model for nonparametric time series classification. In: Advances in Neural Information Processing Systems, vol. 26, pp. 1088–1096. Springer (2013)
11. Cortes, C., Vapnik, V.: Support vector machine. *Mach. Learn.* **20**(3), 273–297 (1995)
12. Devroye, L., Györfi, L., Lugosi, G.: *A Probabilistic Theory of Pattern Recognition*. Springer, New York (1996)
13. Duivesteijn, W., Feelders, A.: Nearest neighbour classification with monotonicity constraints. In: Machine Learning and Knowledge Discovery in Databases, pp. 301–316. Springer (2008)
14. Eads, D., Hill, D., Davis, S., Perkins, S., Ma, J., Porter, R., Theiler, J.: Genetic algorithms and support vector machines for time series classification. In: Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation V, Proceedings of SPIE, vol. 4787, pp. 74–85 (2002)
15. Fix, E., Hodges, J.: Discriminatory analysis, nonparametric discrimination: consistency properties. Technical Report, USAF School of Aviation Medicine, Randolph Field (1951)
16. Garcia, V., Mollineda, R.A., Sanchez, J.S.: On the k-NN performance in a challenging scenario of imbalance and overlapping. *Pattern Anal. Appl.* **11**, 269–280 (2008)
17. Geurts, P.: Pattern extraction for time series classification. In: Principles of Data Mining and Knowledge Discovery, pp. 115–127. Springer (2001)
18. Grabocka, J., Wistuba, M., Schmidt-Thieme, L.: Time-series classification through histograms of symbolic polynomials. *Comput. Res. Repos.-arXiv abs/1307.6365* (2013)
19. Grochowski, M., Jankowski, N.: Comparison of instance selection algorithms II. Results and comments. In: International Conference on Artificial Intelligence and Soft Computing. Lecture Notes in Computer Science, vol. 3070, pp. 580–585. Springer, Berlin (2004)
20. Hand, D.J., Vinciotti, V.: Choosing k for two-class nearest neighbour classifiers with unbalanced classes. *Pattern Recognit. Lett.* **24**, 1555–1562 (2003)
21. He, H., Garcia, E.A.: Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.* **21**(9), 1263–1284 (2009)
22. He, X., Zhang, J.: Why do hubs tend to be essential in protein networks? *PLoS Genet.* **2**(6) (2006)
23. Horváth, T., Vojtáš, P.: Ordinal classification with monotonicity constraints. In: Advances in Data Mining. Applications in Medicine, Web Mining, Marketing, Image and Signal Mining, pp. 217–225 (2006)
24. Jankowski, N., Grochowski, M.: Comparison of instance selection algorithms I. Algorithms survey. In: Proceedings of the International Conference on Artificial Intelligence and Soft Computing. Lecture Notes in Computer Science, vol. 3070, pp. 598–603. Springer, Berlin (2004)
25. Jolliffe, I.: *Principal Component Analysis*. Wiley Online Library, New York (2005)
26. Kehagias, A., Petridis, V.: Predictive modular neural networks for time series classification. *Neural Netw.* **10**(1), 31–49 (1997)
27. Keller, J.E., Gray, M.R., Givens, J.A.: A fuzzy k-nearest-neighbor algorithm. *IEEE Trans. Syst., Man Cybern.* **15**(4), 580–585 (1985)
28. Keogh, E., Shelton, C., Moerchen, F.: Workshop and challenge on time series classification. In: International Conference on Knowledge Discovery and Data Mining (KDD) (2007)

29. Kim, S., Smyth, P.: Segmental hidden Markov models with random effects for waveform modeling. *J. Mach. Learn. Res.* **7**, 945–969 (2006)
30. Levenshtein, V.: Binary codes capable of correcting deletions, insertions, and reversals. *Sov. Phys. Dokl.* **10**(8), 707–710 (1966)
31. Lin, W.J., Chen, J.J.: Class-imbalanced classifiers for high-dimensional data. *Brief. Bioinform.* **14**(1), 13–26 (2013)
32. Liu, H., Motoda, H.: On issues of instance selection. *Data Min. Knowl. Discov.* **6**(2), 115–130 (2002)
33. MacDonald, I., Zucchini, W.: *Hidden Markov and Other Models for Discrete-Valued Time Series*, vol. 1. Chapman & Hall, London (1997)
34. Marcel, S., Millan, J.: Person authentication using brainwaves (EEG) and maximum a posteriori model adaptation. *IEEE Trans. Pattern Anal. Mach. Intell.* **29**, 743–752 (2007)
35. Martens, R., Claesen, L.: On-line signature verification by dynamic time-warping. In: *Proceedings of the 13th International Conference on Pattern Recognition*, vol. 3, pp. 38–42 (1996)
36. Marussy, K., Buza, K.: Success: a new approach for semi-supervised classification of time-series. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L., Zurada, J. (eds.) *Artificial Intelligence and Soft Computing. Lecture Notes in Computer Science*, vol. 7894, pp. 437–447. Springer, Berlin (2013)
37. Niels, R.: Dynamic time warping: an intuitive way of handwriting recognition? Master's Thesis. Radboud University Nijmegen, The Netherlands (2004)
38. Petridis, V., Kehagias, A.: *Predictive Modular Neural Networks: Applications to Time Series*. The Springer International Series in Engineering and Computer Science, vol. 466. Springer, Netherlands (1998)
39. Rabiner, L., Juang, B.: An introduction to hidden Markov models. *ASSP Mag.* **3**(1), 4–16 (1986)
40. Radovanović, M.: *Representations and Metrics in High-Dimensional Data Mining*. Izdavačka knjižarnica Zorana Stojanovića, Novi Sad, Serbia (2011)
41. Radovanović, M., Nanopoulos, A., Ivanović, M.: Nearest neighbors in high-dimensional data: the emergence and influence of hubs. In: *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pp. 865–872 (2009)
42. Radovanović, M., Nanopoulos, A., Ivanović, M.: Hubs in space: popular nearest neighbors in high-dimensional data. *J. Mach. Learn. Res. (JMLR)* **11**, 2487–2531 (2010)
43. Radovanović, M., Nanopoulos, A., Ivanović, M.: Time-series classification in many intrinsic dimensions. In: *Proceedings of the 10th SIAM International Conference on Data Mining (SDM)*, pp. 677–688 (2010)
44. Rish, I.: An empirical study of the naive Bayes classifier. In: *Proceedings of the IJCAI Workshop on Empirical Methods in Artificial Intelligence* (2001)
45. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. *Acoust. Speech Signal Process.* **26**(1), 43–49 (1978)
46. Schedl, M.F.A.: A Mirex meta-analysis of hubness in audio music similarity. In: *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR 12)* (2012)
47. Stańczyk, U.: Recognition of author gender for literary texts. In: *Man-Machine Interactions 2*, pp. 229–238. Springer (2011)
48. Sykacek, P., Roberts, S.: Bayesian time series classification. *Adv. Neural Inf. Process. Syst.* **2**, 937–944 (2002)
49. Tomašev, N.: The Role of Hubness in High-Dimensional Data Analysis. Jožef Stefan International Postgraduate School (2013)
50. Tomašev, N., Mladenić, D.: Nearest neighbor voting in high dimensional data: learning from past occurrences. *Comput. Sci. Inf. Syst.* **9**, 691–712 (2012)
51. Tomašev, N., Mladenić, D.: Class imbalance and the curse of minority hubs. *Knowl. Based Syst.* **53**, 157–172 (2013)
52. Tomašev, N., Mladenić, D.: Hub co-occurrence modeling for robust high-dimensional kNN classification. In: *Proceedings of the ECML/PKDD Conference*. Springer (2013)

53. Tomašev, N., Radovanović, M., Mladenović, D., Ivanović, M.: A probabilistic approach to nearest neighbor classification: Naive hubness Bayesian k-nearest neighbor. In: Proceedings of the CIKM Conference (2011)
54. Tomašev, N., Radovanović, M., Mladenović, D., Ivanović, M.: Hubness-based fuzzy measures for high-dimensional k-nearest neighbor classification. *Int. J. Mach. Learn. Cybern.* **5**(3), 445 (2013)
55. Tomašev, N., Radovanović, M., Mladenović, D., Ivanović, M.: The role of hubness in clustering high-dimensional data. *IEEE Trans. Knowl. Data Eng.* **99** (PrePrints), 1 (2013)
56. Wang, J., Neskovac, P., Cooper, L.N.: Improving nearest neighbor rule with a simple adaptive distance measure. *Pattern Recognit. Lett.* **28**(2), 207–213 (2007)
57. Xi, X., Keogh, E., Shelton, C., Wei, L., Ratanamahatana, C.: Fast time series classification using numerosity reduction. In: Proceedings of the 23rd International Conference on Machine Learning (ICML), pp. 1033–1040 (2006)

# **Chapter 12**

## **Selection of Visual Descriptors for the Purpose of Multi-camera Object Re-identification**

**Piotr Dalka, Damian Ellwart, Grzegorz Szwoch, Karol Lisowski,  
Piotr Szczuko and Andrzej Czyżewski**

**Abstract** A comparative analysis of various visual descriptors is presented in this chapter. The descriptors utilize many aspects of image data: colour, texture, gradient, and statistical moments. The descriptor list is supplemented with local features calculated in close vicinity of key points found automatically in the image. The goal of the analysis is to find descriptors that are best suited for particular task, i.e. re-identification of objects in a multi-camera environment. The analysis is performed using two datasets containing images of humans and vehicles recorded with different cameras. For the purpose of descriptor evaluation, scatter and clustering measures are supplemented with a new measure that is derived from calculating direct dissimilarities between pairs of images. In order to draw conclusions from multi-dataset analysis, four aggregation measures are introduced. They are meant to find descriptors that provide the best identification effectiveness, based on the relative ranking, and simultaneously are characterized with large stability (invariance to the selection of objects in the dataset). Proposed descriptors are evaluated practically with object re-identification experiments involving four classifiers to detect the same object after its transition between cameras' fields of view. The achieved results are discussed in detail and illustrated with figures.

**Keywords** Video surveillance · Image descriptors · Multi-camera tracking · Object identification

### **12.1 Introduction**

The popularity of video surveillance systems increases continuously along with greater availability of solutions designed to ensure the safety of the monitored areas. However, due to rapid development of multi-camera surveillance systems [17], they

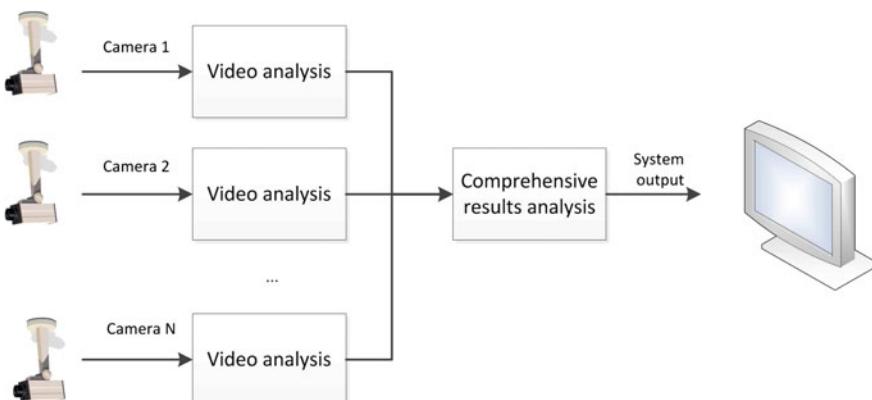
---

P. Dalka (✉) · D. Ellwart · G. Szwoch · K. Lisowski · P. Szczuko · A. Czyżewski  
Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology,  
Narutowicza 11/12, 80-233 Gdańsk, Poland  
e-mail: piotr.dalka@sound.eti.pg.gda.pl

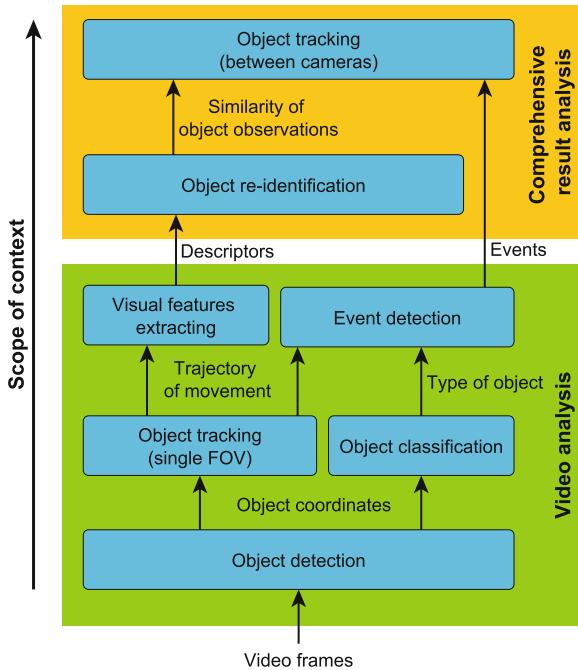
become more difficult to manage manually because of a limited human perception [1, 10, 31]. Therefore, methods and automatized algorithms are being extensively developed to assist the human operator in the monitoring process. This task can be accomplished by applying certain video processing algorithms to the recorded or live video sequences [40]. Such methods can be utilized for threat-related event detection. The event itself can be described diversely, for example as a defined set of rules [46] or as a deviation from typically observed behaviour [48]. The system operator should be alerted in case of an incident, and as a result, his attention should be focused on the relevant camera image where the suspicious event occurs. Additionally, the object that triggered the alert should be tracked in order to facilitate further observation of its behaviour.

The approach described above is applied independently to each camera of the system. As a result, tracks of moving objects in individual cameras are obtained. In order to benefit from the multi-camera system architecture, data acquired from all the logically related video sources should be combined in multi-camera object tracks. The illustration of this approach is presented in Fig. 12.1.

In order to facilitate understanding how such a system works, a certain model (architecture) of video processing might be considered (Fig. 12.2). The video processing task can be divided into layers depending on the context level. Low level algorithms operate on values of particular image pixels in order to detect objects that are not part of a stable background. Object tracking within the camera field of view (FOV) is based on the background subtraction results. In the next stage, the classification and event detection tasks are performed. Simultaneously, calculation of visual feature descriptors is carried out. A combination of results obtained from many cameras, data related to spatio-temporal inter-camera dependencies (referred to as topology), and object behaviour model is performed on the higher level of analysis. On each consecutive layer, the context becomes richer. It starts from determining that moving object appears in camera's FOV and ends with connecting parts of video images related to certain objects and to detected events.



**Fig. 12.1** A general scheme of a smart multi-camera system



**Fig. 12.2** Consecutive stages of video analysis presented as layered model

Although the multi-camera object tracking is, in most of the cases, not a simple task, its completion may allow for a more in-depth analysis of monitored activities. For example, the route of object movement can be obtained for the whole monitored area, presenting a comprehensive overview of the incident. In order to accomplish the multi-camera object tracking it is necessary to recognize the same objects in images from separate cameras. This is not a trivial task due to changes in object appearance between the cameras. Two views of the same object in two cameras may differ in shape (because of different camera angles), colour (different light or camera settings). Therefore, the task of object re-identification requires solving two problems. First, a set of visual descriptors has to be selected in order to provide a distinctive representation of objects appearance. Numerous descriptors related to object shape, colour, texture, etc. are available and in order to perform an efficient re-identification, a set of descriptors has to be selected so that it is possible to match two views of the same object in two cameras and, at the same time, distinguish the object from the others. The second problem is related to a method of testing the similarity between sets of visual descriptors. This may be formulated as follows: given a set of descriptors of each object collected from individual cameras and a set of descriptors computed for an object that appeared in another camera, find a best match of the descriptors. The matching procedure may be realized with trained classifiers and a distance measure.

In this chapter, the authors' approach to the aforementioned problem of object re-identification is described. Section 12.2 presents the algorithms used for the low-level video analysis which extracts moving objects from the individual camera images. In Sect. 12.3, various approaches for multi-camera object tracking are presented. The problem of object re-identification requires a selection of distinctive image features which allow for matching two appearances of the same object in separate cameras and, at the same time, ensure that two different objects will not have similar feature sets. In order to achieve this goal, the most commonly used visual features descriptors and methods of extracting them from the image are discussed in Sect. 12.4. These descriptors are then evaluated in order to select a subset of them optimal for the object re-identification in multiple cameras, as presented in Sect. 12.5. In the following Section, the selected descriptors are applied to the object re-identification task. The authors propose to use a classifier which is trained on feature descriptors of a single object obtained from all its appearances in a single camera, and then use it for comparison with the feature sets obtained from other cameras. Four classification methods are described in Sect. 12.6 and the optimal solution is selected. Finally, the results of experiments performed using the proposed framework are presented and discussed in Sect. 12.7.

## 12.2 Video Preprocessing

Generally, an image contains visual information at several levels of complexity, depending on the application. From a single colour light detection, to analysis of raster binary maps, colour photos of simple geometry, to understanding of complex natural scenes, and finally tracking objects moving on a complex background [4].

Digital images, represented as an RGB pixel matrix, can be processed in many dissimilar ways. Several methods treat each pixel independently of its surroundings (contrast enhancement, gamma correction), while others require the pixel context—colour values of neighbouring pixels (noise reduction, sharpening). Numerous methods of image filtration can be found in the literature [8, 9, 21], employing FFT, DCT and wavelet transformations [42].

For the object re-identification framework described here, the most important objects are these in motion, described with a particular colour distribution and a texture, disturbed by the noise, and changing appearance over time due to the motion and deformations of the object, and the camera motion in 3D space. In order to obtain information on the movement, algorithms for object detection, recognition, and tracking are applied [12, 28]. The purpose of the object detection routine is to select image pixels belonging to moving objects, and to extract image regions representing individual objects. These regions will be later used for calculation of important visual information on these objects. Usually, a background modelling and subtraction approach is used for this task [45]. The background model composes statistical profiles of the most probable values of background pixels, for determination of what colour and brightness of pixel in the current frame should be treated as the

current background. In case of significant discrepancy between the pixel and the model, the pixel is regarded as the foreground. The model is updated continuously to account for lighting condition changes.

The result of the object detection stage for a single video image is composed of image regions representing moving objects. Contours of these objects may be parametrized using descriptors related to its size and shape, while the image section covered by each region provides data for calculation of appearance descriptors (colour, texture, etc.) These descriptors are essential for the object re-identification [16].

## 12.3 Multi-camera Object Tracking

Object tracking performed within a single camera is a matter of finding relations between objects detected in consecutive video frames, so that each moving object is tracked on a frame-to-frame basis. The result of object tracking performed in one camera is the track—a set of consecutive object positions. For a multi-camera approach, these relations are searched among the images from different system cameras, so that the object's tracks from a number of cameras are merged, which makes this task more difficult. There are two general approaches to perform multi-camera object tracking. Provided that the monitored area is observed by cameras with overlapping views, the situation is quite simple. It is enough to define the relations between these areas to re-identify objects in different images on the basis of their positions [38].

For disjointed camera views, the situation is more demanding. In order to recognize objects between cameras, they need to be characterized by a set of features. Again, two basic approaches can be found in the literature. In the first one, objects are described by their biometrical features. For example, re-identification of persons may be performed using features such as the gait or face [20]. However, application of these methods is strictly limited regarding the image quality and camera placement. A more popular approach for multi-camera object tracking involves extraction of image descriptors which depict object appearance [44, 47]. Selection of a proper description method needs to ensure that the selected features are characterized by a high robustness to expected variances in object appearance throughout all cameras.

Surveillance systems can contain miscellaneous camera types (e.g. digital and analogue ones), registering images under unstable lighting conditions. Hence, an object representation in images from these devices can vary. Particularly, this variation can be related to different camera characteristics or their settings (i.e. white balance, exposure), and their position respective to the object (angle of view). Additionally, scenes observed by the cameras can be illuminated differently, resulting in different object appearances. Such differences can also be noticed between the indoor and outdoor surveillance cameras. Therefore, the acquired object description needs to be robust against such changing conditions. This can be achieved by utilizing illumination-invariant features which allow obtaining feature vectors that, in most of the cases, do not suffer from varying conditions. On the other hand, even

simple description methods can be sufficient, provided that camera colour calibration techniques are used [27].

The actual object re-identification task involves implementation of the classification scheme that allows determining which object candidate in a destination camera is the one observed in a source camera. This task may be based on a simple, distance-based measure in order to compare visual feature vectors of objects' images with each other, or it may employ intelligent decision systems techniques, such as Neural Networks or Decision Trees.

Surveillance systems can contain dozens or even hundreds of cameras. Searching for the tracked object in each camera of such systems would be inefficient and could lead to many errors. In order to solve this problem, information about the system topology should be included in the process [18]. This way, if an object leaves the field of view of a certain origin camera, its representation is required to be searched only in the cameras with a proper spatio-temporal relation from the origin. For further improvement of the object re-identification accuracy, time windows representing expected transition times between cameras can be utilized. Their purpose is to define additionally the most probable periods within which the objects are sought after. Such a mechanism is the most beneficial in case when neighbouring cameras are placed at a considerable distance between each other.

The third hint in the re-identification process is a behaviour model that utilizes information on frequency of particular transitions between cameras. Moreover, the complete paths (routes) of objects can be analysed in order to obtain a behaviour model. Such a model contains a statistical description of objects behaviour, so it can be used to predict future movement of the object or allow reconstructing its route in the past. In literature, many methods of building a behaviour model are described. Kettneraker et al. present utilization of Markov models [29]. The number of future steps which can be predicted is related to the order of Markov model. Another method presumes usage of particle filters based on previously collected statistical data about routes of objects [30]. Behaviour model facilitating object re-identification might be also based on the idea of Pawlak's flowgraphs [14]. The latter two algorithms use large amount of statistical data as an input.

To sum up, the multi-camera object tracking process combines many types of meta-data obtained from the consecutive steps of video analysis. Depending on the changing conditions of video acquisition (different illumination types, number of objects tracked by video surveillance system, amount of statistical data used for building behaviour model, modifications in the camera network topology etc.), importance of a particular type of hints can also be modified.

## 12.4 Visual Object Descriptors

A large miscellany of visual object descriptors for image classification are available [2, 3, 24, 35]. However, only some of them can be utilized for the problem considered here [5, 32, 44]. As it was stated in the previous Section, object description for the purpose of re-identification in a multi-camera surveillance system is expected to be

robust against scene illumination changes. Hence, the extracted set of features should be invariant to a set of image transformations. Five general types of such changes related to different lighting conditions are defined in literature [49]. The general pixel color transformation model is defined according to Eq. 12.1

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} o_R \\ o_G \\ o_B \end{bmatrix} \quad (12.1)$$

where RGB represent image values in corresponding channels,  $[o_R, o_G, o_B]$  is the offset and  $a, b$  and  $c$  denote scaling factors. From the relation presented in Eq. 12.1, five different types of illumination changes can be listed:

- Type 1:  $a = b = c$  and  $o_R = o_G = o_B = 0$ —light intensity change
- Type 2:  $a = b = c = 0$  and  $o_R = o_G = o_B$ —light intensity shift
- Type 3:  $a = b = c$  and  $o_R = o_G = o_B$ —light intensity change and shift
- Type 4:  $a \neq b \neq c$  and  $o_R = o_G = o_B = 0$ —light colour change
- Type 5:  $a \neq b \neq c$  and  $o_R \neq o_G \neq o_B$ —light colour change and shift

To show the usefulness of the particular image descriptors they should be tested against the enlisted transformations. Most extensively utilized descriptors in the topic of multi-camera object tracking include SIFT-like features [5, 23, 47].

In this work, a set of feature extraction techniques is chosen, to show efficiency of other solutions. Besides the regular image colour histogram, which is calculated for comparison reasons, each of the descriptors shows resistance to at least one of the illumination transformations. The related information is presented in Table 12.1.

Another important feature property is its geometric invariance. Object dimensions can vary due to different camera placement and perspective. Additionally, for non-levelled cameras and in case of lens distortions, objects visible in the scene can be rotated. Some of the descriptors listed in Table 12.1 would suffer from these conditions. To overcome this problem, during image preprocessing, objects are rotated accordingly and resized to defined dimensions to introduce rotation and scale invariance.

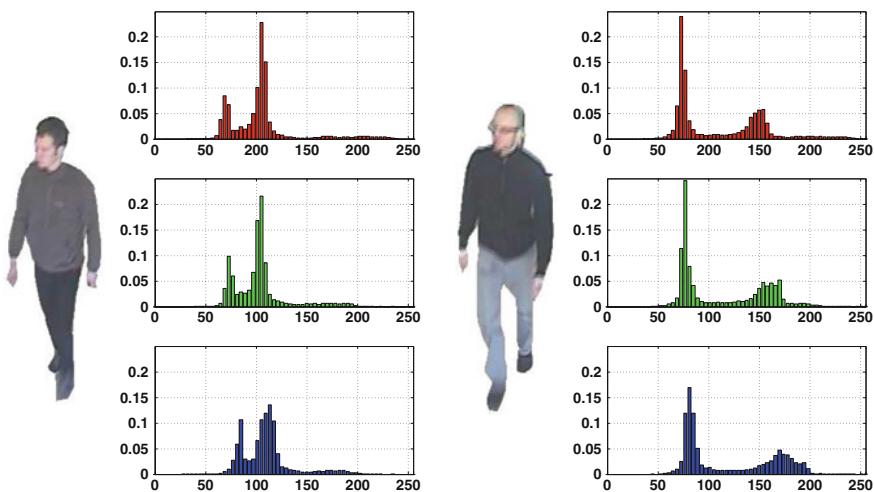
### 12.4.1 Colour Histogram

Two types of colour histograms have been selected as visual image features. The first one is full RGB histogram of an object image (*HistFull* descriptor—Fig. 12.3). The second one is a two-channel histogram using a chromatic space of  $R_cG_c$  colours (*Hist* descriptor). The values of  $R_c$  and  $G_c$  in the chromatic space may be derived from a RGB pixel through the following formulas:

$$R_c = \frac{R}{R + G + B}, \quad G_c = \frac{G}{R + G + B}. \quad (12.2)$$

**Table 12.1** Characteristics of the visual feature descriptors

Feature name	Color space/representation	Illumination inv.	Geometric inv.
MomentCentNorm	RGB	Type 2	Yes
MomentInvGPSO	RGB	Type 1, 2, 3, 4, 5	Yes
VertTrace	Transformed colour	Type 1, 2, 3, 4, 5	No
CLD	RGB	Type 1, 2	No
CLDTrans	Transformed colour	Type 1, 2, 3, 4, 5	No
EHD	Gray-level	Type 1, 2, 3	No
LBPHist	RGB	Type 1, 2, 3	No
Hist	rg-chromatic space	Type 1	Yes
HistFull	RGB	–	Yes
Sift	Gray-level	Type 1, 2, 3, 4, 5	Yes
Surf64	Gray-level	Type 1, 2, 3, 4, 5	Yes
Surf128	Gray-level	Type 1, 2, 3, 4, 5	Yes
OpponentSift	Opponent colour space	Type 1, 2, 3	Yes
OpponentSurf64	Opponent colour space	Type 1, 2, 3	Yes
CMSP	HSV	Type 1, 2, 3	No

**Fig. 12.3** Colour histograms for example objects

This means that the values of  $R_c$  and  $G_c$  represent the share of red and green in the original colour. It is easy to notice that deriving the share of blue is unnecessary, as the three shares sum to one.

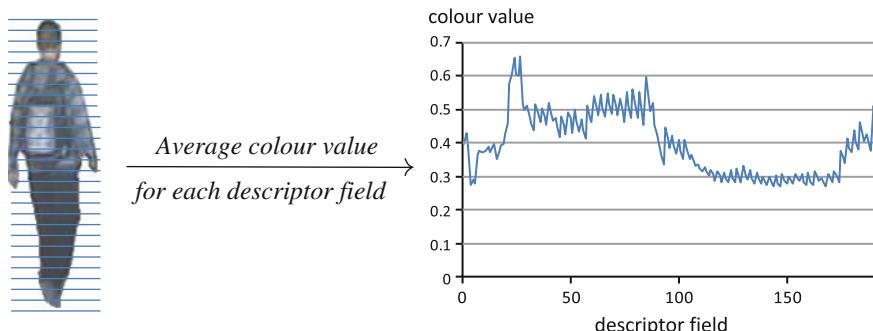
The chromatic space defined in that way introduces some losses (it is not possible to revert to the full RGB space of colours). However, simultaneously the space does not include the information about luminance which causes an object moving in an unevenly illuminated scene to have identical description in the  $R_cG_c$  space all the time. For example, for  $R_c = 1/3$  and  $G_c = 1/3$ , it is known that all the compound colours have the same share in the original RGB space, although it is not possible to determine whether the colour is black, grey, or white. Another advantage of the  $R_cG_c$  colour space is the simplicity of its specification and lower dimension of histograms based on it.

In order to limit dimensions of the histogram descriptors and to reduce their internal redundancy, they are calculated for each colour channel independently. Furthermore, histograms bin size is equal to 4, so each colour channel is represented by 64 bins in the histogram (assuming 8 bits per channel, i.e. 256 levels). Therefore, the `HistFull` descriptor contains 192 elements and `Hist`—128 elements.

#### 12.4.2 Vertical Trace

One of the methods to describe an object is by its colour characteristics. This can be achieved by calculating the image colour trace along a defined direction [26]. In case of images representing people, the most reasonable direction is vertical, since more information about colour distribution can be acquired (Fig. 12.4). However, this descriptor can be utilized for other objects as well, assuming that all of them share the same orientation. Trace can be calculated in several ways, for example as the mean or median of slices along the orientation.

As the trace length is dependent on the described object dimensions, during the preprocessing stage, the image needs to be normalized. With this approach, 192-element feature vector is built, containing 64 values for each of the colour channels (VertTrace descriptor). To introduce description invariance, feature extraction procedure is performed for transformed colour representation, defined by the following equation:



**Fig. 12.4** Vertical trace calculation

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = \begin{bmatrix} \frac{R - \mu_R}{\sigma_R} \\ \frac{G - \mu_G}{\sigma_G} \\ \frac{B - \mu_B}{\sigma_B} \end{bmatrix} \quad (12.3)$$

where  $R, G, B$  denote appropriate channel colour values,  $\mu_{[R,G,B]}$  and  $\sigma_{[R,G,B]}$  are their mean and standard deviation values in the image, respectively.

### 12.4.3 Moment Invariants

Image statistical moments calculation is one of the methods for a general characterization of colour distribution. However, these basic parameters are sensitive to all kinds of geometrical image transformations. Hence, normalized central moments are defined to overcome this limitation. These moments, calculated for each image channel independently, are utilized to form a feature vector consisting of 21 elements (MomentCentNorm).

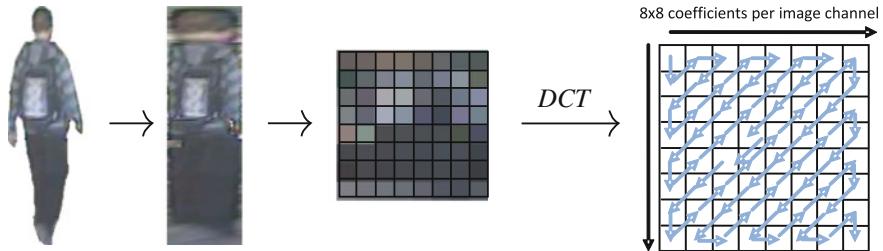
However, such a description is influenced by most of the photometric image transformations. A set of parameters based on image moments which are invariant to such changes is proposed in the literature [37]. They are expressed as a combination of inter-channel image moments, which are defined as:

$$M_{pq}^{abc} = \sum_{x,y} x^q y^p I_R(x, y)^a I_G(x, y)^b I_B(x, y)^c \quad (12.4)$$

where  $I(x, y)$  are the image values for  $(x, y)$  coordinates,  $p + q$  is the moment order and  $a + b + c$  is the moment degree. From the set of moments proposed by Mindru et al. [37], Moment Invariants of type GPSO (Geometric Photometric Scaling Offset) are chosen for later considerations, as they are independent from affine geometric and photometric (scaling and offset) image transformations. Utilizing this approach, a feature vector consisting of 18 elements (MomentInvGPSO descriptor) is built for the purpose of further experiments.

### 12.4.4 Colour Layout Descriptors

Another method utilized for image colour description is called Colour Layout Descriptor (CLD). It is a parametrization method defined as one of the colour descriptors in MPEG-7 standard [36]. It is used to describe the basic colour palette present in the image, including additional information about the colour spatial layout. Preparation of a feature vector regarding this method can be divided into several stages. These steps, including image preprocessing, are presented in Fig. 12.5.



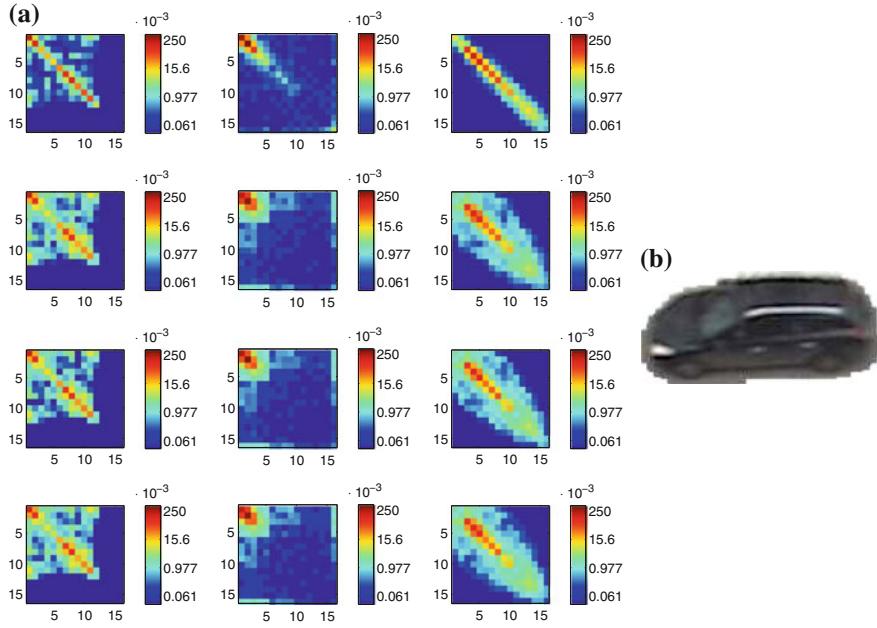
**Fig. 12.5** Consecutive stages of Colour Layout Descriptor calculation. From *left* described object, its stretched representation, image regions mean calculation, and the DCT coefficient scanning result

As the CLD is defined for images of defined rectangular dimensions and the object shape can vary, the visual object representation is stretched in each row to defined dimensions during the preprocessing stage. Afterwards, the analysed image is divided into equal blocks ( $8 \times 8$ ). Next, a representative value for each image region is calculated as the mean channel intensity within a block. As a result, image thumbnail is acquired, which is further transformed using DCT (discrete cosine transform). To form the feature vector, coefficients scanning, along with their normalization is performed. This type of coefficient reading method is utilized to group low frequency components together. The final feature vector acquired this way consists of 192 elements, in which 64 DCT coefficients for each colour channel are included. In MPEG-7 standard it is stated that for CLD descriptor, YCbCr representation should be used (CLD descriptor). For the purpose of experiments, besides YCbCr, additionally Transformed Colour, defined by Eq. 12.3, is utilized as well (CLDTrans descriptor).

#### 12.4.5 Co-occurrence Matrices Statistical Parameters

This image descriptor contains statistical parameters of co-occurrence matrices of an object image (CMSP descriptor). A co-occurrence matrix, also referred to as a co-occurrence distribution, is defined over an image to be the distribution of co-occurring values at a given offset ( $\Delta x, \Delta y$ ) [13, 25]. It is commonly used as a texture description.

A set of symmetrical, normalized, co-occurrence matrices  $P$  is calculated for each channel of the object image in HSV colour space. Each set contains four matrices  $P$  calculated for four different directions of offsets:  $0^\circ$ —offset  $(0,1)$ ,  $45^\circ$ — $(1,1)$ ,  $90^\circ$ — $(1,0)$  and  $135^\circ$ — $(-1,1)$ . Image values are quantized into  $16 \times 16$  equally-spaced levels, therefore each co-occurrence matrix contains  $16 \times 16$  elements. Example co-occurrence matrices  $P$  for two different object images are visualised in Figs. 12.6 and 12.7.



**Fig. 12.6** Co-occurrence matrices **a** calculated for an example vehicle image **b** for each channel (HSV—from *left to right*) and for each direction ( $0^\circ, 45^\circ, 90^\circ, 135^\circ$ —from *top to bottom*)

Five statistical parameters are calculated for every co-occurrence matrix  $P$ . They were chosen from among others to be the least dependent on each other and they are defined as follows [13, 25]:

$$\text{contrast} = \sum_{i,j} P_{i,j} (i - j)^2 \quad (12.5)$$

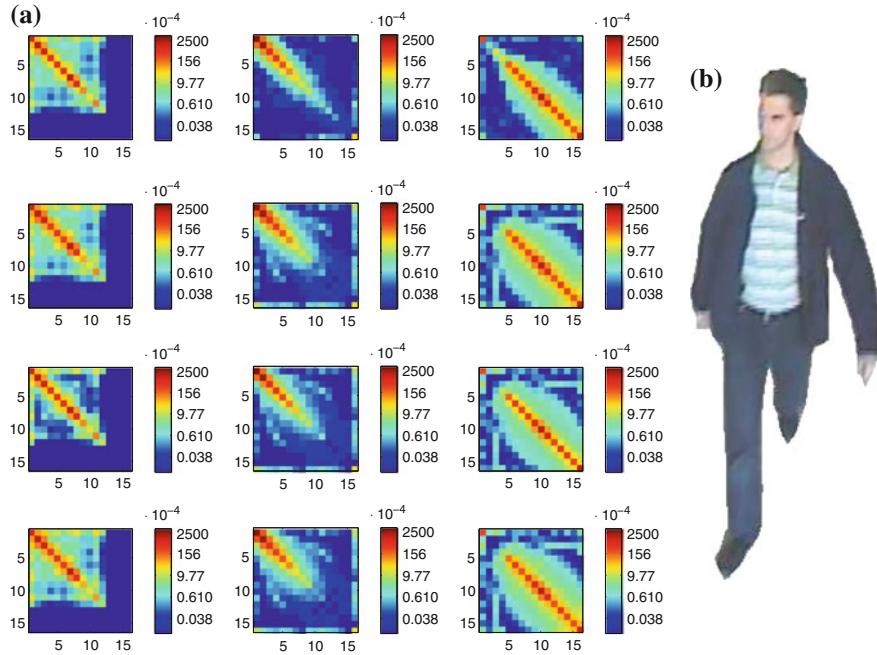
$$\text{energy} = \sqrt{\sum_{i,j} P_{i,j}^2} \quad (12.6)$$

$$\text{mean} = \mu_i = \mu_j = \sum_{i,j} i \cdot P_{i,j} \quad (12.7)$$

$$\text{standard deviation} = \sigma_i = \sigma_j = \sqrt{\sum_{i,j} (i - \mu_i)^2 P_{i,j}} \quad (12.8)$$

$$\text{correlation} = \sum_{i,j} P_{i,j} \frac{(i - \mu_i)(j - \mu_j)}{\sqrt{\sigma_i^2 \sigma_j^2}} \quad (12.9)$$

This gives a total number of 60 elements for CMSP descriptor (3 channels  $\times$  4 co-occurrence matrices  $\times$  5 parameters).



**Fig. 12.7** Co-occurrence matrices **a** calculated for an example person image **b** for each channel (HSV—from left to right) and for each direction ( $0^\circ, 45^\circ, 90^\circ, 135^\circ$ —from top to bottom)

### 12.4.6 Edge Histogram Descriptor

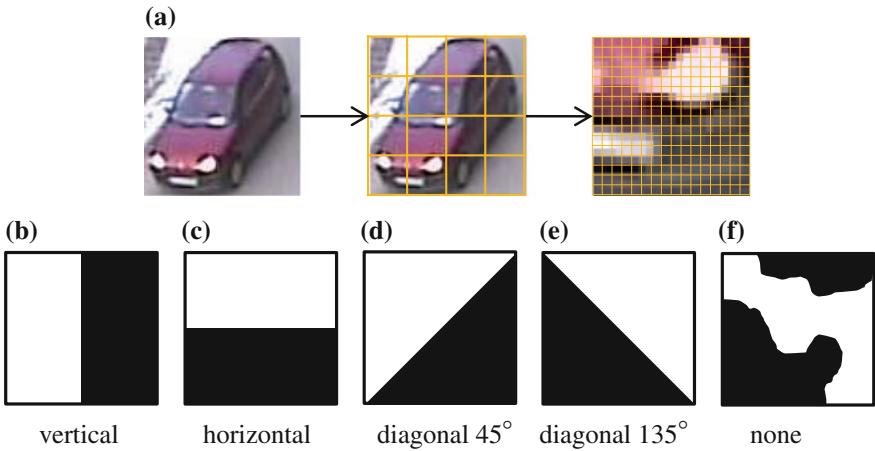
Edge Histogram Descriptor is a technique which is used to characterize image texture (EHD). It is one of the methods defined within the MPEG-7 standard [36]. It describes the texture of a grey-scale image in the form of a histogram. This descriptor extraction process is calculated as follows [41].

First, similarly to CLD extraction, a preprocessing stage, in which the image is re-scaled and rotated accordingly, is carried out. Next, the analysed image is divided into 16 regions ( $4 \times 4$ ). Afterwards, each region is additionally divided into blocks of 4 pixels ( $2 \times 2$ ). For each block, a dominant texture directionality is estimated utilizing a set of filters (Fig. 12.8).

To estimate the main directionality, the strength is calculated for each of its types according to the equation:

$$m_{[dir]} = \left| \sum_{k=0}^3 a_k(i, j) \times f_{[dir]}(k) \right| \quad (12.10)$$

where  $m_{[dir]}$  is the strength of a specific direction,  $a_k$  is the  $k$ th block element at  $(i, j)$  coordinates and  $f_{[dir]}(k)$  corresponds to  $k$ th filter coefficient.



**Fig. 12.8** Edge Histogram Descriptor: **a** Consecutive stages of the image segmentation; **b–f** directions for each bin in the edge histogram

The acquired maximum value is additionally thresholded to remove weak directionalities which could be caused by image noise:

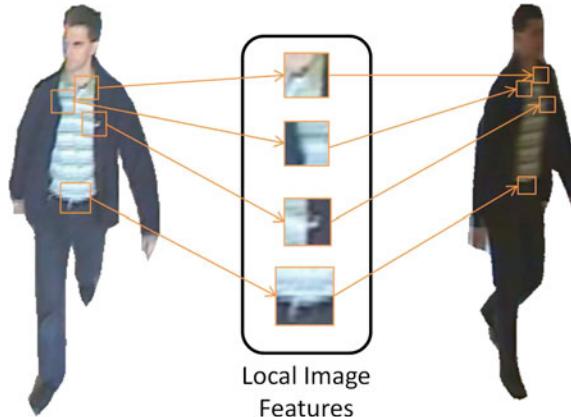
$$\max \{m_{[dir]}\} > T_{edge} \quad (12.11)$$

The applied threshold ( $T_{edge}$ ) is set to 10 during experiments. If this condition is fulfilled, the appropriate histogram bin value, related to the specified directionality, is incremented. In this way, 5 bin histograms for each of the 16 image regions are built. Hence, the feature vector acquired utilizing this method consists of 80 elements.

#### 12.4.7 Local Binary Pattern

Local Binary Pattern (LBP) is a texture related parametrization method which describes neighbourhood of each analysed image point [11, 24]. In most cases, a direct 8-point neighbourhood of an analysed pixel is used. However, it is possible to define it in other words. LBP extraction procedure can be divided into two stages. First, each pixel neighbourhood is analysed to determine the related code word. This is done by comparing neighbouring pixel values against the analysed pixel. For the pixel positions where the analysed point is greater than the related neighbour, the value of 1 is assigned, 0 otherwise. In this way, binary code words are formed for each of the image points. Afterwards, they are used to build a histogram. The description obtained with this approach is rotation-dependent. Therefore, before feature extraction occurs, object images are preprocessed to set their proper orientation.

The LBP histogram is calculated and normalized for each of the colour channels independently. After quantizing each of them to 64 bins, a feature vector consisting of 192 elements is built (LBPHist descriptor).



**Fig. 12.9** Local image feature matching

#### 12.4.8 Local Image Features

Local image features differ significantly from all other image descriptors presented in the chapter. They are calculated in the nearest vicinity of key points that are found automatically in the grey-level image. While dimensionality of the vector derived from each key point is constant for each local image feature, the number of key point varies significantly depending on the image. Therefore, the final local image descriptor is a matrix with vectors for each key point stored in its rows. This means that the width of the matrix remains constant while its height depends on the analysed image (Fig. 12.9).

Key points in the image are found automatically using the Fast-Hessian detector [5]. An adaptive, iterative procedure is used in order to find  $25 \pm 10\%$  key points for each image by changing Hessian threshold properly with each iteration.

Two types of local image features are used: SIFT (Scale Invariant Feature Transform) and SURF (Speeded Up Robust Features). They are scale-, rotation- and translation invariant, show robustness against illumination changes, and their values remain very similar across a substantial range of affine distortions. SIFT features [34] are derived from the image gradient magnitudes and orientations, sampled for each pixel in the key point neighbourhood, while SURF descriptors [5] are based on the sums of Haar wavelet responses in horizontal and vertical directions in the square region centered around the key point. Depending on the number of sums, the SURF vector may contain 64 or 128 elements per one key point.

Typically, local image features are calculated for grey-level images, therefore they lack colour information. In order to overcome this inconvenience, features are additionally calculated for each colour channel independently, thus increasing feature vector dimensionality three times comparing to grey-level images. The opponent colour space was used for this purpose [49]:

**Table 12.2** Local image features listing

Feature symbol	Base local feature	Colour information	Feature vector dimensionality for a key point
Sift	SIFT	Grey-level	128
Surf64	SURF	Grey-level	64
Surf128	SURF	Grey-level	128
OpponentSift	SIFT	Opponent colour space	364
OpponentSurf64	SURF	Opponent colour space	192

$$\begin{bmatrix} O_1 \\ O_2 \\ O_3 \end{bmatrix} = \begin{bmatrix} \frac{R-G}{\sqrt{2}} \\ \frac{R+G-2B}{\sqrt{6}} \\ \frac{R+G+B}{\sqrt{3}} \end{bmatrix} \quad (12.12)$$

where  $R$ ,  $G$ ,  $B$  denote appropriate channel colour values. This space is tuned to mimic human perception of colour. The third channel is proportional to the intensity channel of the HSV colour model while other two channels contain differences of opponent pairs *red–green* and *yellow–blue*.

Table 12.2 lists all local image features used during object re-identification experiments presented further in the chapter.

## 12.5 Descriptor Evaluation Methods

In order to evaluate the set of visual object descriptors objectively, valid measures able to extract and highlight properties of the descriptors significant from the object re-identification point of view, have to be employed. Therefore, two measures are used that analyse inter- and intra-class scatter and clustering properties. They are supplemented with a new, third method that is based on direct image pairs dissimilarity measurements.

### 12.5.1 RS Index

RS index is a parameter that estimates the degree of clusters dissimilarity [33, 43]. This method is based on the Euclidean distances between instances of the dataset. The measure is one of the approaches utilized for verifying the presented object description techniques. The R-squared parameter  $RS$  is calculated utilizing the following relation:

$$RS = \frac{SS_t - SS_w}{SS_t}, \quad (12.13)$$

where  $SS_t$  and  $SS_w$  are the sum of squared distances for the whole dataset and for each of the clusters, respectively. These parameters are defined as follows:

$$SS_t = \sum_{v \in C} \|v - \bar{v}\|^2, \quad (12.14)$$

$$SS_w = \sum_i \sum_{v \in C_i} \|v - \bar{v}_i\|^2, \quad (12.15)$$

where  $\bar{v}$  is the mean of the whole dataset, described as  $C$ , and  $\bar{v}_i$  represents the mean vector for the  $i$ th cluster.  $RS$  parameter ranges from 0, for poorly clustered data, to 1 for the classes which can be characterized as compact and separable. Thus, this term should be maximized.

For the purpose of visual descriptors assessment this measure is calculated in two variants:

- $C$ —for each camera independently, where clusters represent objects,
- $O$ —for each object independently, where clusters represent cameras.

The  $C$  variant is utilized to depict descriptor capabilities to distinguish between objects in the same camera. On the other hand,  $O$  variant is utilized to represent the compactness of object description between different system cameras. Stability of these results, for each of the variants, is calculated as:

$$SNR_i = \frac{\mu_i}{\sigma_i}, \quad (12.16)$$

where  $\mu_i$  and  $\sigma_i$  are mean and standard deviation calculated for  $i$ th variant. Afterwards, the aggregated results are utilized to determine the parameter  $RS_A$  which can be used to compare different descriptors quality:

$$RS_A = \frac{SNR_C}{SNR_O}. \quad (12.17)$$

For the optimal feature extractor, object description between cameras should be as compact as possible and, at the same time, its description within a particular camera should be distant from other objects descriptions. Hence, for a good parametrization technique,  $RS_A$  term should be maximized.

### 12.5.2 SD Index

*SD* validity index is another measure utilized to verify data clustering [22]. This parameter is calculated as a relation of two components:

$$SD = (1 + Scat) \cdot Dis, \quad (12.18)$$

where  $Scat$  represents cluster compactness and  $Dis$  indicates the separation between different classes. Both of these components are calculated according to the following formulas:

$$Dis = \frac{\max_{i,j=1,\dots,n_c} (\|v_j - v_i\|)}{\min_{i,j=1,\dots,n_c} (\|v_j - v_i\|)} \cdot \sum_{i=1}^{n_c} \left( \sum_{j=1,i \neq j}^{n_c} \|v_j - v_i\| \right)^{-1}, \quad (12.19)$$

$$Scat = \frac{\sum_{i=1}^{n_c} \|\sigma(v_i)\|}{n_c \|\sigma(x)\|}, \quad (12.20)$$

where  $n_c$  is the class count,  $v_i$  and  $v_j$  are the corresponding class mean vectors,  $\sigma(v_i)$  denotes  $i$ th cluster standard deviation and  $\sigma(x)$  is the deviation of the whole dataset.

Low  $Scat$  factor value means that clusters are more compact.  $Dis$  parameter is related to the clusters distribution and its value increases with the cluster number. To ensure the best dataset cluster separation and compactness, the  $SD$  index value should be minimum.

This validity measure, similarly to the  $RS$  index, is calculated for two separate cases and aggregated afterwards (Eq. 12.16). From the obtained results,  $SD_A$  term is determined as the inverted relation from Eq. 12.17:

$$SD_A = \frac{SNR_O}{SNR_C}. \quad (12.21)$$

### 12.5.3 Dissimilarity Measure

A new method of descriptor evaluation is proposed. It is based on direct comparison of dissimilarity of all image pairs in the dataset according to a chosen visual descriptor. The goal of the dissimilarity measure is to compare descriptors according to their behaviour in a multi-camera, multi-object environment. The measure highlights the fact that similarity of two images of the same object should be greater than similarity of different objects, regardless of the cameras used to acquire both images. The measure has also one additional advantage: in contrast to  $SD$  and  $RS$  indexes, the dissimilarity measure may be obtained for local image features, because it does not utilize mean and standard deviation values that are not defined for set-of-vectors descriptors.

Given two images  $I_1$  and  $I_2$  and their descriptors  $\mathbf{V}_1$  and  $\mathbf{V}_2$ , the dissimilarity  $d$  for standard (non-local-based) descriptors is calculated as the Euclidean distance:

$$d = \|\mathbf{V}_1 - \mathbf{V}_2\|. \quad (12.22)$$

In case of local image feature-based descriptors,  $\mathbf{V}_1$  and  $\mathbf{V}_2$  are matrices  $m_1 \times n$  and  $m_2 \times n$ , accordingly. Therefore, a matching procedure is used in order to find the best matching vector in matrix  $\mathbf{V}_2$  for each vector in  $\mathbf{V}_1$  and vice versa [50]. For  $k$ th vector (row) of matrix  $\mathbf{V}_i$ , the best matching vector in the matrix  $\mathbf{V}_j$  is the one having the smallest Euclidean distance; this distance is denoted as  $e_{ij}^k$ . The dissimilarity of  $\mathbf{V}_1$  and  $\mathbf{V}_2$  is calculated according to the equation:

$$d = 0.5 \cdot \frac{\sum_{k=1}^{m_1} e_{21}^k}{m_1} + 0.5 \cdot \frac{\sum_{k=1}^{m_2} e_{21}^k}{m_2}. \quad (12.23)$$

The smaller the value of  $d$ , the smaller the dissimilarity and thus the greater similarity of the compared images.

Dissimilarity values  $d$  of image pairs are divided into 4 sets depending on the objects that are depicted in the images and on cameras that were used to capture images. The sets are as follows: the same object in the same camera (SOSC), the same object in different cameras (SODC), different objects in the same camera (DOSC) and different objects in different cameras (DODC). In each set,  $SNR$  measure is calculated as the ratio of the mean value to the standard deviation. This normalizes results and makes it possible to compare visual descriptors according to the dissimilarity measure. Final dissimilarity measure  $DSIM$  for the given visual descriptor is derived from the following equation:

$$DSIM = \frac{SNR_{SOSC} \cdot SNR_{SODC}}{SNR_{DOSC} \cdot SNR_{DODC}}. \quad (12.24)$$

The numerator is related to similarity of image pairs belonging to the same object while the denominator reflects similarity of images of different objects. Therefore, the smaller dissimilarity measure  $DSIM$  for the given visual descriptor, the better the descriptor is suited to identify objects in a multi-camera environment.

### 12.5.4 Result Aggregation

Descriptor evaluation measures (DEMs) discussed in Sects. 12.5.1–12.5.3 are calculated for one set of objects' images only. In order to perform a thorough evaluation of various visual descriptors, they need to be verified using several test datasets. This subsection presents methodology for aggregation of results from such an analysis.

Let  $U = u_i, i = 1, \dots, N_U$  be the set of  $N_U$  datasets used for visual descriptors evaluation. Each set contains images of different objects recorded in different locations. Each dataset  $u_i$  is divided into  $N_i$  subsets  $u_{ij} \in u_i$  containing objects from the dataset  $u_i$ . DEMs are calculated for each subset  $u_{ij}$ , independently. The aim of the approach is to make results of analysis robust against selection of particular objects in the dataset.

We propose two measures useful for analysis of DEM values for different datasets and different visual descriptors: stability and ranking. Stability describes dispersion of DEM values for the given descriptor. It exists in two variants: internal and external. Individual internal stability  $SII$  is defined for the given feature  $D$ , set  $u_i$  and descriptor evaluation measure  $M$  with the following equation:

$$SII_i(D, M) = \frac{\sqrt{\sum_{j=1}^{N_i} (M_{ij}(D) - \mu_i^{MD})^2}}{\mu_i^{MD} \cdot \sqrt{N_i}} = \frac{\sigma_i^{MD}}{\mu_i^{MD}} = CV_i^{MD}, \quad (12.25)$$

$$\mu_i^{MD} = \frac{1}{N} \sum_{j=1}^{N_i} M_{ij}(D),$$

where  $M_{ij}(D)$  is a value of DEM  $M$  of the descriptor  $D$  in the subset  $u_{ij}$ ,  $\mu_i^{MD}$  denotes the mean value of DEM  $M$  of the descriptor  $D$  in the set  $u_i$ ,  $\sigma_i^{MD}$  is the standard deviation of the same data and  $CV$  coefficient of variation.

Stability is given as an average dispersion of DEM values related to their mean value, therefore the smaller its value is, the greater stability of the DEM is and the better the visual descriptor is suited for object re-identification.

Internal stability  $SI$  aggregates partial stabilities of all DEMs of all sets:

$$SI(D) = \sqrt{\sum_{M=1}^{N_M} \sum_{i=1}^{N_U} SII_i(D, M)}, \quad (12.26)$$

where  $N_M$  denotes number of DEMs used.

External stability  $SE$  measures normalized dispersion of DEMs of a particular feature  $D$  across different datasets in  $U$ . It is given with the equation:

$$SE(D) = \sqrt{\sum_{M=1}^{N_M} \sum_{i=1}^{N_U} \left( \frac{\mu_i^{MD} - \mu^{MD}}{\mu^{MD}} \right)^2}, \quad (12.27)$$

$$\mu^{MD} = \frac{1}{N} \sum_{i=1}^{N_U} \mu_i^{MD},$$

where  $\mu^{MD}$  denotes the average of mean values of DEM  $M$  of the descriptor  $D$  in all datasets  $U$ . The smaller the value of  $SE$ , the greater the independence of DEMs of the descriptor  $D$  from dataset selection and the better the descriptor is suited for visual object identification.

Ranking is the second measure developed for analysis of DEM values for different datasets and various visual descriptors. It also exists in two variants: value-based and position-based. Individual value-based ranking  $RVI$  is defined for the given feature

$D$ , set  $u_i$  and DEM  $M$  as the mean value over all subsets of  $u_i$  normalized according to the “best” value of  $M$  for all visual descriptors:

$$RVI_i(D, M) = \frac{\mu_i^{MD}}{\max_D(\mu_i^{MD})}, \text{ or } RVI_i(D, M) = \frac{\min_D(\mu_i^{MD})}{\mu_i^{MD}}. \quad (12.28)$$

$$RVI_i(D, M) \in [0, 1]$$

The formula for  $RVI$  is chosen based on the characteristics of DEM; if its value is directly proportional to the expected results, i.e. the greater value of DEM the better the descriptor is suited for visual object identification, then the first formula is used; otherwise the second equation is employed.

Partial value-based ranking is aggregated in other to obtain the final value-based ranking  $RV$  for the descriptor  $D$ :

$$RV(D) = \frac{\sum_{M=1}^{N_M} \sum_{i=1}^{N_U} RVP_i(D, M)}{N_M \cdot N_U}, \quad RV(D) \in [0, 1]. \quad (12.29)$$

Position-based ranking, in contrast to the value-based one, takes into account only integer position (place) of the DEM of the given descriptor against all other features. For the given feature  $D$ , set  $u_i$  and DEM  $M$ , individual position-based ranking  $RPI$  is defined as an integer value in the range  $[1, N_D]$  ( $N_D$ —number of descriptor used) that positions the descriptor as the best (1) or the worst ( $N_D$ ) among all descriptors. Therefore, for a particular set and DEM,  $RVI$  values in the descending order and  $RPI$  values in the ascending order create the same sequence of descriptors.

Partial position-based ranking is aggregated in other to obtain final, floating-point, position-based ranking  $RP$  for the descriptor  $D$ :

$$RP(D) = \frac{\sum_{M=1}^{N_M} \sum_{i=1}^{N_U} RPP_i(D, M)}{N_M \cdot N_U}, \quad RP(D) \in [1, N_D]. \quad (12.30)$$

## 12.6 Object Identification

The purpose of feature extraction presented in the previous sections is to employ the gathered data on moving objects for the task of object re-identification. A typical scenario of the multi-camera system is that the object (e.g. a vehicle) leaves the camera view and appears in another camera’s view after some time. It is generally not known in which camera the object will re-appear. Therefore, the problem may be defined as follows. A set of feature vectors  $\mathbf{v}_{i,j}$  of the moving object  $O_i$ , gathered from each video frame  $j$  in which a given object was present, comprises a *class*. For a newly appeared object in another camera, a new feature vector  $\mathbf{v}'$  is computed.

The task of a *classifier* is to assign  $\mathbf{v}'$  to the matching class. Each class represents a single moving object. The label of such a class (e.g. the object's ID) is the classification result, allowing for object re-identification.

Diversity of classification algorithms may be found in the literature. For the purpose of the experiments described in the chapter, four classifications algorithms have been selected. Some other algorithms, such as Support Vector Machines or the Bayes classifier, have been rejected after the preliminary experiments due to their unsatisfactory performance. Where applicable, classifiers are set to solve a regression problems in order to obtain both discrete class label and a real number that may be interpreted as similarity of a feature vector to the class recognized.

Parameters of the employed classifiers have been selected based on initial experiments and are meant to be representative for the general problem class (i.e. object re-identification). Usage of many classifiers is motivated by the need to make descriptor evaluation invariant to a classifier selection. The four classifiers used are briefly described below.

**k-Nearest Neighbours (kNN).** This is the simplest one of the four selected classifiers. For each feature vector of the considered objects,  $k$  closest feature vectors are found in the training set, consisting of feature vectors of objects in another cameras ( $k = 3$  value has been used in the experiments as a typical value that has performed well in initial experiments). These nearest neighbours are found by calculating a distance between vectors, with the Euclidean metric. Object re-identification is performed by voting, i.e. an object is assigned to the class to which most of the found nearest neighbours belong.

**Artificial Neural Networks (ANN).** For the purpose of object identification, a feed-forward, multi-layer perceptron (MLP) ANN [39] with one hidden layer is used. This significantly reduces ANN training duration, comparing to larger quantity of hidden layers. The ANN is trained using image features (in case of whole image-based descriptors) or with features calculated for each interest point (for local image features), and with their responses (classes—object IDs). Thus, the number of inputs corresponds with the length of a feature vector. Based on initial experiments, the number of neurons in the hidden layer is set to the half of ANN inputs. This fact and only one hidden layer reduce possibility of ANN overtraining and losing its generalisation properties. Bipolar sigmoid transfer functions are used in all neurons as the most suited one to the ranges of input and output values. There are two outputs from the network. An expected ANN output is equal to  $(1, -1)$  for a vector belonging to the valid object and  $(-1, 1)$  if it belongs to other objects. During classification, two ANN outputs (instead of one) make possible to get information both on the vector similarity and the ANN response reliability.

**Gradient Boosted Trees (GBTree).** This classifier is built on the idea of Classification and Regression Trees [7]. Binary decision trees are trained using the object features and their responses, a tree is constructed by processing each object feature and finding an optimal split. Re-identification is done by processing the new object features with the trained tree, starting from the topmost node, passing through each split, and reaching the leaf which defines the class (ID of the re-identified object). Importance of each object feature for the classification is assessed, so that features

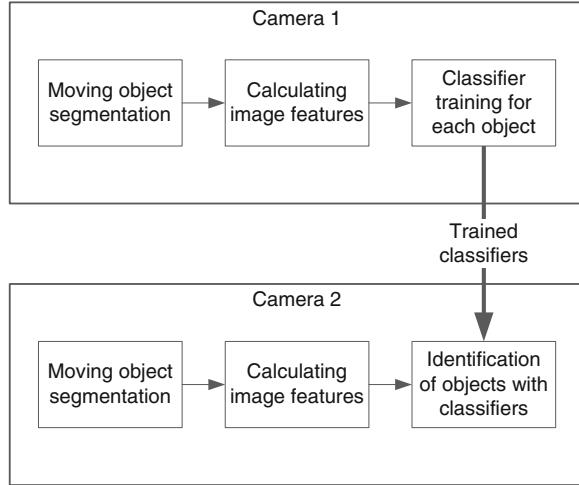
with low importance may be omitted from the classification. The Gradient Boosted Trees (GBTREE) algorithm [19] extends the idea of decision trees by employing a set (an ensemble) of *weak* decision trees. A weighted sum of decisions from each tree in the ensemble provides the final re-identification result. In the experiments described here, the GBTREE classifier is employed to solve the regression problem, returning a value in the range  $[0, 1]$ ; the greater the output value is, the greater similarity of the object feature vectors. A squared loss function is used for the ensemble training as the most suitable one for this setup. 200 iterations of boosting algorithms during GBTREE training has been performed (resulting in 200 trees in the ensemble) and for each iteration, the randomly chosen subset of 80 % training vectors has been used.

**Random Forest (RTree).** This algorithm also uses an ensemble of binary trees (called *a forest* here) for enhanced classification [6]. In contrast to the GBTREE algorithm, each tree is trained with only a subset of object features, different for each tree. The data not used for training of a given tree is then used for its validation. During the re-identification stage, each tree processes the same vector of object features and makes a decision. The final classification result is a class that was chosen by most of the decision trees. The depth of each tree is set to 5, which seems to be an optimal value based on initial experiments using the cross-validation. The minimum samples required at a leaf node for it to be split is equal to 10, which accounts for the small ratio of the total data. The size of the randomly selected subset of features at each tree node that are used to find the best split is equal to the square root of the number of elements in the feature vector. During the re-identification experiments, the RTree classifier was configured to solve a binary classification problem. However, the direct output (a discrete class label) is ignored. Instead, the probability (confidence) of the sample belonging to the object of interest is used. It returns the number between 0 and 1 that is calculated as the proportion of the decision trees that assigned the sample to the class representing an object of interest.

In order to employ diversified classifiers and visual features to object re-identification task, a method of training the classifier and aggregation of the results must be proposed. The solution presented in the chapter resembles the real-life application scenario as close as possible: a classifier is trained with object image features obtained from one camera and then it is used to recognize the same object in video frames acquired from another camera (Fig. 12.10). Therefore, one classifier is trained per each object and each transition between fields of view of two particular cameras.

Positive training samples are formed by the image features of an object of interest in the source camera. Negative samples for training are formed by features of other objects from the dataset that passed the field of view of the source camera. Therefore the classifier is learned to distinguish one particular object from other, possible similar objects that could be found in the same area. Positive samples for validation are created from the images of the object of interest in another, destination camera; negative samples are formed with features of other object images. It is assured that the same objects are not found in negative training and validation sets.

In order to identify the object  $S$  observed in camera  $C_1$  (the source one) in video frames acquired from camera  $C_2$  (the destination one), it is necessary to find the most similar object out of all candidates for matching. Let  $O_i, i = 1 \dots NO$  denote



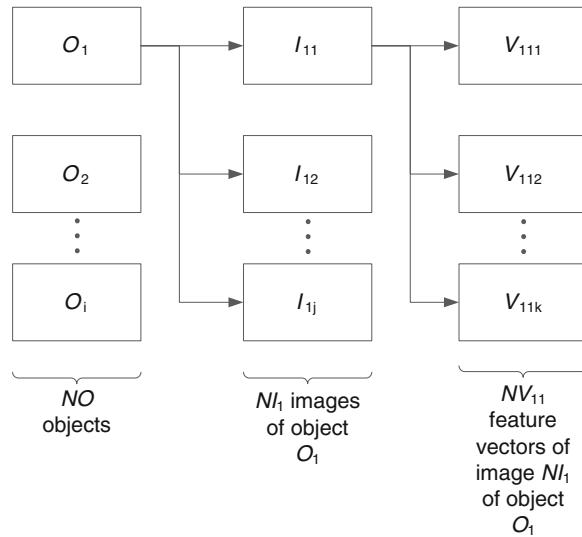
**Fig. 12.10** Scheme of the multi-camera object identification algorithm

*i*th objects for matching from among  $NO$  objects,  $I_{ij}$ ,  $j = 1 \dots NI_i$  represents  $j$ th image of the  $i$ th object out of  $NI_i$  images of  $O_i$  and  $V_{ijk}$ ,  $k = 1 \dots NV_{ij}$  defines  $k$ th image feature vector of the image  $I_{ij}$  from among  $NV_{ij}$  vectors. In case of majority of image features, there is only one feature vector for each object image; however, in case of local image features, there is one feature vector for each of multiple key points found automatically in the image. The relation between objects, images and feature vectors are illustrated in Fig. 12.11. During the re-identification stage, all feature vectors  $V_{ijk}$  of every image of all objects found in the destination camera  $C_2$  are classified with the chosen classifier that represents the object  $S$  observed in the source camera  $C_1$ .

The result of each classification consists of three values: (1) the binary decision  $d_{ijk}$  denoting whether the feature vector  $V_{ijk}$  belongs (positive decision), or not (negative decision), to the object of interest  $S$ ; (2) the response  $r_{ijk} \in [0, 1]$  representing the similarity of vector  $V_{ijk}$  to the object of interest  $S$ ; (3) the weight  $w_{ijk} > 0$  that represents the classifier response reliability. The formulas for obtaining decision, response and weight values for each classifier type are presented in Table 12.3.

Classification results for all vectors of the image  $I_{ij}$  are aggregated in order to obtain the mean result  $\bar{r}_{ij}$  for an image according to the equation:

$$\bar{r}_{ij} = \begin{cases} \frac{\sum\limits_{k=0}^{NV_{ij}} r_{ijk} \cdot w_{ijk}}{NV_{ij}} & \text{for positive } d_{ijk} \\ \frac{\sum\limits_{k=0}^{NV_{ij}} -(1-r_{ijk}) \cdot w_{ijk}}{NV_{ij}} & \text{for negative } d_{ijk} \end{cases} \quad (12.31)$$



**Fig. 12.11** Relation between objects, images and features

**Table 12.3** Formulas for obtaining decision  $d$ , response  $r$  and weight  $w$  values for various classifiers

Classifier	Classifier outputs	Decision $d$ (positive decision condition)	Response $r \in [0, 1]$	Weight $w > 0$
ANN	$(o_1, o_2) \in [-1, 1]$ — values of outputs	$o_1 > o_2$	$0.5 \cdot (\frac{o_1}{2} - \frac{o_2}{2} + 1)$	$\frac{\max(\frac{o_1+1}{2}, \frac{o_2+1}{2})^2}{\frac{o_1+1}{2} + \frac{o_2+2}{2}}$
kNN	$o \in [0, 1]$ —output, $\mathbf{L} = \{l : l \in \{0, 1\}\}$ — vector of $k$ responses, $\mathbf{M} = \{m : m \geq 0\}$ — vector of $k$ distances	$o > 0.5$	$o$	$\frac{\sum_{i=1}^k m_i \cdot l_i}{\sum_{i=1}^k l_i}$
RTree	$o \in [0, 1]$ —probability	$o > 0.5$	$o$	1
GBTTree	$o \in [0, 1]$ —probability	$o > 0.5$	$o$	1

Then, the aggregated result  $R_i$  for each object  $O_i$  is calculated as follows:

$$R_i = \frac{\sum_{j=0}^{N_{I_i}} \bar{r}_{ij}}{N_{I_i}}. \quad (12.32)$$

The object of interest  $S$  is matched with the object  $O_i$  with the highest value of the aggregated result  $R_i$ . If there are more than one object with the same maximum values of  $R_i$ , the object with the highest ratio  $D_i$  of individual positive decisions is chosen:

$$D_i = \frac{\sum_{j=0}^{NI_i} \left( \frac{1}{NV_{ij}} \sum_{j=0}^{NV_{ij}} d_{ijk} \right)}{NI_i}, \quad d_{ijk} = \begin{cases} 1 & \text{for positive decision} \\ 0 & \text{for negative decision} \end{cases} \quad (12.33)$$

## 12.7 Experiments and Results

This section presents experiments carried out and their outcomes. First, datasets used in experiments, involving human and vehicle images, are presented. Then, descriptor evaluation measures are employed to assess particular descriptors in the task of object re-identification. Based on the results, combined feature vectors for both object types are proposed. The vectors (as well as single descriptors) are evaluated with object re-identification experiments involving four classifiers.

### 12.7.1 Datasets

For the purpose of visual descriptor evaluation for multi-camera object identification, two datasets were created. The first one, HUMAN, contains images of persons walking indoors. The second set, VEHICLE, contains images of cars driving in the parking lot near an office building. This location has already been exploited in our previous experiments regarding parking event detection [15]. Both sets contain images of objects acquired from various cameras with different horizontal and vertical angles (Figs. 12.12 and 12.13). Not all objects appear in all cameras, therefore the number of object/camera pairs in the set is lower than the product of numbers of objects and cameras. Datasets also differ significantly by the number of objects and the duration of data acquisition. Furthermore, in case of the VEHICLE set, for the re-identification task a subset of objects (based on the time criterion) has been used comparing with the descriptor evaluation task. Object amount reduction has been made due to the enormous computational cost of the object re-identification experiments (see the Sect. 12.7.3). All these differences (summarized in Table 12.4) are meant to evaluate visual descriptors in the large inconstancy of conditions, including small and large datasets, different object types and recording conditions.

Each object image in the dataset is accompanied with a binary mask that denotes image pixels belonging to the object. The masks have been obtained automatically with moving object detection and tracking algorithms that were developed with the Indect project and deployed successfully in the area of video event detection [16, 46]. Only masked pixels of object are used for visual descriptor calculating.

**Fig. 12.12** Sample images of persons from HUMAN dataset



**Fig. 12.13** Sample images of two vehicles from the VEHICLE dataset moving in opposite direction in the fields of view of all cameras

**Table 12.4** Characteristics of datasets used in the experiments

Dataset	Number of objects	Number of cameras	Number of images	Number of object/camera pairs	Average number of images per object	Average number of images per object/camera pair	Timespan of data acquisition (h)
HUMAN	7	7	265	48	37.9	5.5	1
VEHICLE (descriptor evaluation experiments)	223	8	12,994	1,143	58.3	11.4	9
VEHICLE (object re-identification experiments)	15	8	922	85	61.5	10.9	1

### 12.7.2 Descriptor Evaluation

Both datasets have been divided into several subsets based on objects in order to make analysis results independent of the selection of the specific objects in the dataset. HUMAN set contains few objects, therefore subsets have been created as all combinations of five objects out of seven. This gives the total number of 21 subsets.

The VEHICLE set contains an adequate number of objects, therefore the criterion for dividing is based on the time the object appears for the first time in any camera; time window length of 1h has been chosen and the window moves with the step of

0.5 h. This gives a total number of 17 subsets; each contains from 10 to 39 objects (25 on average).

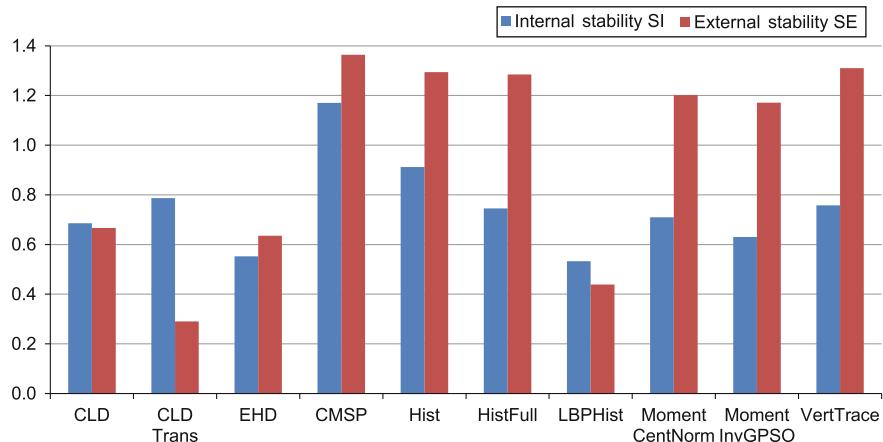
For each image in the datasets, all visual descriptors have been calculated. Then, within each subset, visual descriptor evaluation measures presented in Sects. 12.5.1–12.5.3 have been calculated. Unfortunately, it is not possible to calculate *SD* and *RS* indexes for local image features. Therefore, analysis for these descriptors is limited to *DSIM* measure only.

Next, the results were aggregated for each visual descriptor according to the steps described in Sect. 12.5.4. For the reference, the same procedure has been performed for full HUMAN and VEHICLE datasets, assuming there is only one subset in each set containing all objects. Naturally, in this case it is not possible to calculate internal stabilities *SI*.

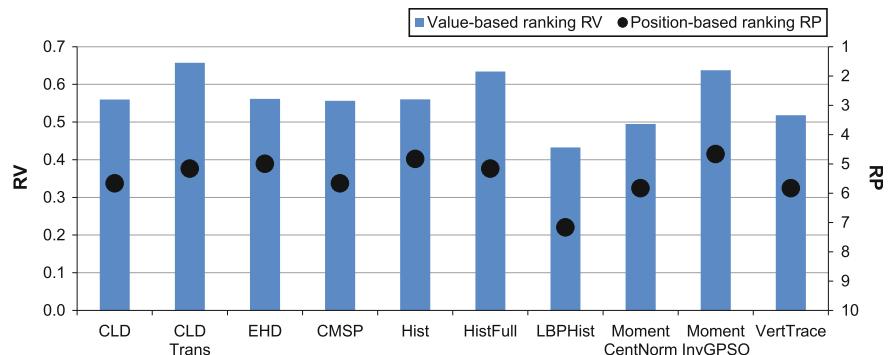
Table 12.5 contains mean and standard deviation values of descriptor evaluation measures (DEMs) over subsets of each dataset, for every visual descriptor. Because for the local image features only dissimilarity measure *DSIM* can be calculated, results for other measures are not available. The data is used to calculate individual aggregation measures presented in Tables 12.6 and 12.7.

Summarized aggregation results (for all datasets and DEMs) are presented in Figs. 12.14 and 12.15, and for each dataset separately—in Figs. 12.16 and 12.17. Based on the data, it is possible to evaluate and compare descriptors with each other based on the data type and different criteria. Among non-local image features, the best results (based on *RV* and *RP*) were achieved for colour-based descriptors (*Hist*, *HistFull*), texture-based ones (*CMSp*) and statistical-based ones (*MomentCentNorm*, *MomentInvGPSO*) in case of VEHICLE dataset, and for colour-based feature (*CLDTrans*), edge-based one (*EHD*) and statistical ones (*MomentCentNorm*, *MomentInvGPSO*) in case of PERSON dataset. It means that the statistical features are universal enough to be suitable for both human and vehicle identification task, however they are not the best in each category, separately. The reason for high effectiveness of histograms for VEHICLE set results from the vehicle nature: they are usually monochromatic, therefore histogram-based vectors are more or less orthogonal. This is not happening in case of objects with more complex colours, like people images. On the other hand, edge orientation-based descriptor *EHD* provides good results (comparing to other descriptors) for PERSON dataset because human pose changes less with different camera viewing angles (the pose remains vertical) while vehicle edges orientation depends heavily on the object rotation angle and camera viewing angle. It seems strange that the *VertTrace* extraction technique provides low descriptor evaluation measures for PERSON. This might be due to characteristics of each image row colour that can vary depending on people outfit. Hence, the mean value utilized to depict this variety might to be inadequate for this task. However, a practical evaluation of this descriptor is performed during re-identification experiments.

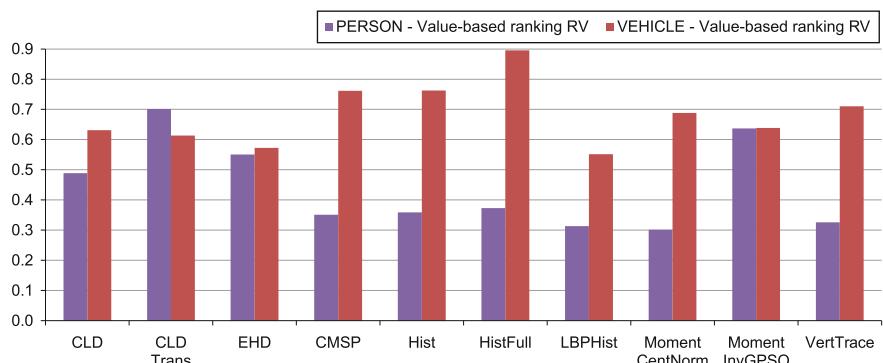
Ranking aggregation measures are not enough to select the best descriptors. It is also important for the descriptor to guarantee stable results in case of different objects in the dataset (it should not be dependent on particular objects in a dataset). Analysis of the stability aggregation measures *SI* and *SE* reveals that the *CMSp* feature has



**Fig. 12.14** Internal stability  $SI$  and external stability  $SE$  aggregation measures (the lower the better)



**Fig. 12.15** Value-based ranking  $RV$  (the higher the better) and position-based ranking  $RP$  (the higher dots and smaller values the better)



**Fig. 12.16** Value-based ranking  $RV$  (the higher the better) calculated for each dataset separately

**Table 12.5** Descriptor evaluation measures statistics (mean values and standard deviations) over subsets of PERSON and VEHICLE datasets

Visual descriptor	<i>DSIM</i> (the lower the better)		<i>SD<sub>A</sub></i> (the higher the better)		<i>RS<sub>A</sub></i> (the higher the better)	
	mean	std	mean	std	mean	std
<b>PERSON set</b>						
CLD	0.746	0.067	1.528	0.313	0.815	0.231
CLDTrans	0.476	0.060	0.668	0.221	1.382	0.524
EHD	0.787	0.043	2.287	0.583	0.956	0.193
CMSP	0.769	0.049	1.714	1.386	0.231	0.069
Hist	0.736	0.056	1.790	0.799	0.208	0.032
HistFull	0.554	0.051	0.867	0.343	0.172	0.056
LBPHist	1.186	0.162	1.095	0.263	0.509	0.139
MomentCentNorm	0.959	0.020	1.775	0.327	0.182	0.044
MomentInvGPSO	0.881	0.041	6.424	1.614	0.511	0.209
VertTrace	0.901	0.049	1.874	0.819	0.218	0.077
OpponentSift	0.628	0.057	n/a		n/a	
OpponentSurf64	0.868	0.032	n/a		n/a	
Sift	0.832	0.032	n/a		n/a	
Surf128	0.890	0.047	n/a		n/a	
Surf64	0.911	0.045	n/a		n/a	
<b>VEHICLE set</b>						
CLD	0.422	0.064	0.690	0.307	0.908	0.313
CLDTrans	0.482	0.051	0.760	0.388	0.931	0.261
EHD	0.726	0.041	0.889	0.278	1.121	0.346
CMSP	0.587	0.132	0.717	0.249	3.134	2.105
Hist	0.505	0.059	0.881	0.364	2.222	1.439
HistFull	0.470	0.065	0.865	0.285	3.437	1.368
LBPHist	0.696	0.046	0.931	0.257	0.687	0.158
MomentCentNorm	0.566	0.075	1.016	0.511	1.349	0.505
MomentInvGPSO	0.772	0.176	1.097	0.306	1.263	0.237
VertTrace	0.840	0.116	0.895	0.283	2.793	1.032
OpponentSift	0.412	0.073	n/a		n/a	
OpponentSurf64	0.548	0.063	n/a		n/a	
Sift	0.350	0.047	n/a		n/a	
Surf128	0.419	0.054	n/a		n/a	
Surf64	0.395	0.051	n/a		n/a	

the worst external (inter-set) stability *SE*. This alone would not disqualify the feature (it provides good results for one object type only), however within dataset stability, *SI* is also the worst which means that descriptor evaluation measures DEMs for CMSP

**Table 12.6** Aggregation measures for PERSON dataset

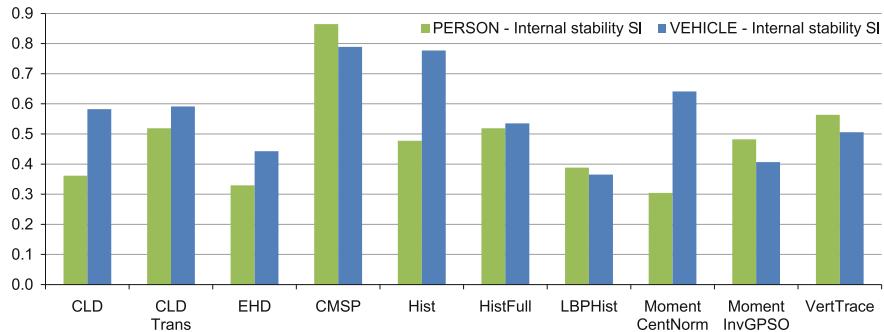
Visual descriptor	Individual internal stability <i>SII</i>			Individual value-based ranking ( <i>RVI</i> ) and individual position-based ranking ( <i>RPI</i> ) in parenthesis		
	<i>DSIM</i>	<i>SD<sub>A</sub></i>	<i>RS<sub>A</sub></i>	<i>DSIM</i>	<i>SD<sub>A</sub></i>	<i>RS<sub>A</sub></i>
CLD	0.090	0.205	0.284	0.638 (4)	0.238 (7)	0.590 (3)
CLDTrans	0.126	0.331	0.379	1.000 (1)	0.104 (10)	1.000 (1)
EHD	0.054	0.255	0.202	0.605 (6)	0.356 (2)	0.692 (2)
CMSP	0.064	0.809	0.300	0.619 (5)	0.267 (6)	0.167 (6)
Hist	0.076	0.446	0.152	0.646 (3)	0.279 (4)	0.151 (8)
HistFull	0.092	0.396	0.323	0.859 (2)	0.135 (9)	0.125 (10)
LBPHist	0.137	0.240	0.273	0.401 (10)	0.170 (8)	0.368 (5)
MomentCentNorm	0.021	0.184	0.241	0.496 (9)	0.276 (5)	0.132 (9)
MomentInvGPSO	0.046	0.251	0.409	0.540 (7)	1.000 (1)	0.370 (4)
VertTrace	0.054	0.437	0.352	0.528 (8)	0.292 (3)	0.158 (7)

**Table 12.7** Aggregation measures for VEHICLE dataset

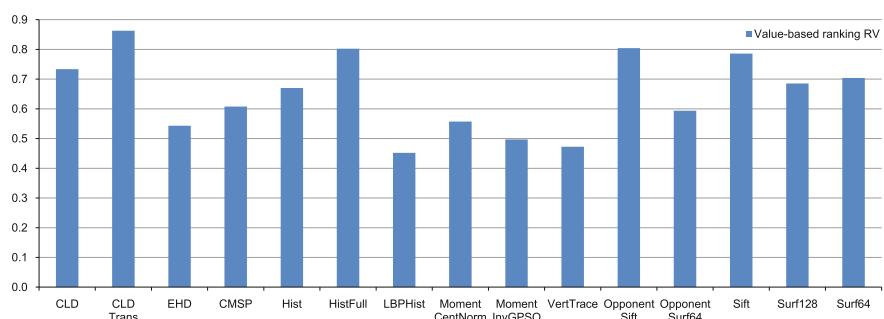
Visual descriptor	Individual internal stability <i>SII</i>			Individual value-based ranking ( <i>RVI</i> ) and individual position-based ranking ( <i>RPI</i> ) in parenthesis		
	<i>DSIM</i>	<i>SD<sub>A</sub></i>	<i>RS<sub>A</sub></i>	<i>DSIM</i>	<i>SD<sub>A</sub></i>	<i>RS<sub>A</sub></i>
CLD	0.151	0.444	0.345	1.000 (1)	0.630 (10)	0.264 (9)
CLDTrans	0.106	0.510	0.280	0.875 (3)	0.693 (8)	0.271 (8)
EHD	0.056	0.313	0.309	0.581 (8)	0.810 (5)	0.326 (7)
CMSP	0.225	0.347	0.672	0.719 (6)	0.654 (9)	0.912 (2)
Hist	0.117	0.413	0.648	0.837 (4)	0.804 (6)	0.646 (4)
HistFull	0.138	0.330	0.398	0.898 (2)	0.788 (7)	1.000 (1)
LBPHist	0.066	0.276	0.229	0.606 (7)	0.849 (3)	0.200 (10)
MomentCentNorm	0.132	0.503	0.375	0.745 (5)	0.926 (2)	0.393 (5)
MomentInvGPSO	0.228	0.279	0.188	0.547 (9)	1.000 (1)	0.368 (6)
VertTrace	0.138	0.317	0.369	0.502 (10)	0.816 (4)	0.813 (3)

change the most depending on the particular objects in the dataset. On the other hand, EHD provides good *SI* and *SE* stabilities, comparing to other features.

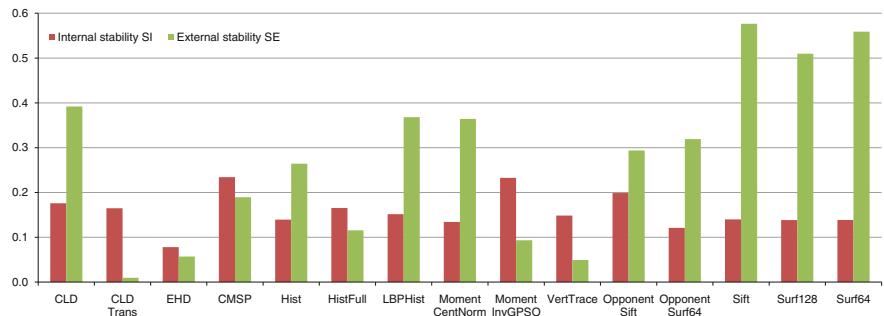
Figures 12.18, 12.19 and 12.20 present aggregation measures for all descriptors, including local ones, calculated for dissimilarity *DSIM* measure only. It may be noticed that SIFT-based feature achieved better results than SURF-based features: they have higher *RV* values and slightly better stabilities. Comparing grey level-based versions with the opponent colour-based ones it is seen that the latter are significantly better, especially in terms of stability, and their rankings are also better in general.



**Fig. 12.17** Internal stability *SI* (the lower the better) calculated for each dataset separately

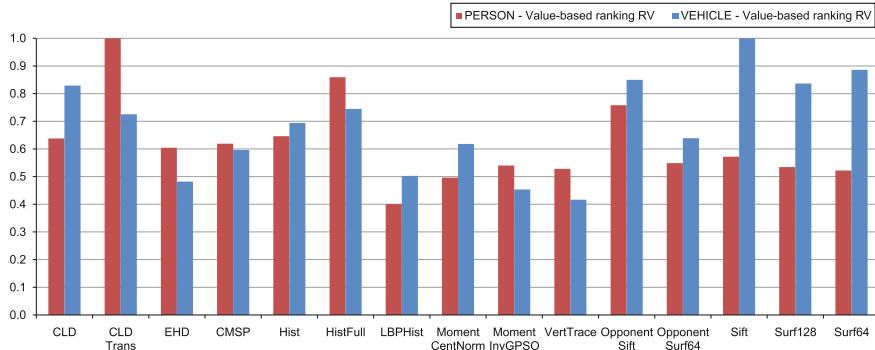


**Fig. 12.18** Value-based ranking *RV* (the higher the better) for *DSIM* measure only with local image descriptors included



**Fig. 12.19** Internal stability *SI* and external stability *SE* (the smaller the better) for *DSIM* measure only with local image descriptors included

Basing on the *DSIM* measure only, it is possible to compare local-based features with other descriptors. Local-based features provide better results for **VEHICLE** (they adopt leading positions). This may be caused by the fact that vehicles provide more distinguishable locations for local feature extraction that are more or less con-



**Fig. 12.20** Value-based ranking  $RV$  (the higher the better) for each dataset separately, for *DSIM* measure only with local image descriptors included

**Table 12.8** Relative differences (%) between aggregation measures calculated for both datasets as wholes and the measures based on mean values of subsets

Visual descriptor	External stability <i>SE</i> difference (%)	Value-based ranking <i>RV</i> difference (%)	Position-based ranking <i>RP</i> difference (%)
CLD	9.8	-6.5	-17.6
CLDTrans	12.2	-3.7	-6.5
EHD	7.5	-9.3	-10.0
CMSP	15.4	-3.7	-5.9
Hist	8.0	-0.2	-3.4
HistFull	5.7	-12.9	9.7
LBPHist	14.6	-14.9	4.7
MomentCentNorm	27.9	-20.1	22.9
MomentInvGPSO	-3.0	-0.1	3.6
VertTrace	6.0	-9.1	0.0

stant for all vehicles (e.g. borders between windows and a vehicle body); there is much more diversity in case of human images. For HUMAN set, the OpponentSift descriptor (the best one of all local image features) occupies the third place.

In the last experiments, it was evaluated how the results calculated for each subset of both datasets independently correlate with results based on the whole sets (Table 12.8). It may be noticed that for majority of descriptors, external stability *SE* is lower (stability value is higher) which seems to be obvious because of a larger quantity of objects in the dataset. At the same time, value-based ranking *RV* values decreased which means that descriptor evaluation measures vary more. However, in majority of cases, difference values are less than 10 %. It proves that the results presented in the chapter are invariant to the number of objects in datasets.

Table 12.9 contains proposed visual descriptors that are better than others (both in case of their effectiveness and stability) for each dataset independently and for both datasets simultaneously. These features are characterized by the highest rank-

**Table 12.9** Visual descriptor candidates suited for multi-camera object identification

Dataset	Best visual descriptors
PERSON	CLDTrans, CLD, EHD, MomentInvGPSO, OpponentSIFT
VEHICLE	HistFull, MomentInvGPSO, VertTrace, OpponentSIFT, SURF64
Both	CLDTrans, MomentInvGPSO, HistFull, EHD, OpponentSIFT

ing values and, simultaneously, by high or average stability. Analysing the results in Fig. 12.20 it can be noticed that for the PERSON category, feature extractors that include spatial dependencies outperform other evaluated descriptors. This relation is reasonable since humans are typically characterized by a higher colour diversity than vehicles. In comparison, the most effective descriptors for VEHICLE dataset are based on local image features as well as on the general image colour representation. This dependency shows that specific description techniques are more suitable for specific object categories. From comprehensive analysis of both datasets, MomentInvGPSO, OpponentSIFT and CLDTrans turned out to be a good choice.

### 12.7.3 Object Identification

Object identification experiments have been performed according to the approach presented in details in Sect. 12.6. A classifier is trained to distinguish visual features of an object of interest from features of other objects in the first (source) camera and then it is used to find the same object in the second (destination) camera from among a few candidates. The classification process is repeated for each object and for all pairs of cameras it appears in; within each pair, two classifications are performed, as each camera is treated as the source and the destination of the transition. The positive training samples are formed by images of the object of interest  $S$  in a camera  $C_1$ . The positive validation samples are formed by images of the vehicle  $S$  in the camera  $C_2$ , where  $C_1 \neq C_2$ . All other objects that appear in cameras  $C_1$  or  $C_2$  are randomly drawn to the negative training set and negative validation set, alternately, until one of object pools is empty. Therefore both negative sets are equipotent or the negative training set has one more element than the validation one. The drawing procedure also assures that the negative training and validation samples do not contain images of the same objects, therefore during re-identification negative validation samples belong to objects whose images were not used for classifier training.

Table 12.10 presents details regarding quantity of objects in training and validation sets and the amount of single identification tasks (for the given classifier and a feature vector) that is equal to the number of object/transition pairs in the dataset. There is always one positive sample (an object of interest  $S$ ) in positive sets and there are approx. 3 persons or 5–6 vehicles in negative training and validation samples on average during each classification. It means that a classifier is trained with more

**Table 12.10** Number of objects (mean values and standard deviations) in training and validation sets for each identification task

Dataset	Number of identification tasks	Training set				Validation set			
		Positive		Negative		Positive		Negative	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std
PERSON	282	1.00	0.00	3.00	0.00	1.00	0.00	2.94	0.24
VEHICLE	452	1.00	0.00	5.50	1.58	1.00	0.00	5.07	1.71

negative than positive samples, and its task is to find one valid object in a set of 4 other persons or 6 other vehicles. This is a very challenging task that is unlikely to be carried out in the real scenarios with the application of spatial and temporal constraints regarding cameras' geographical localizations.

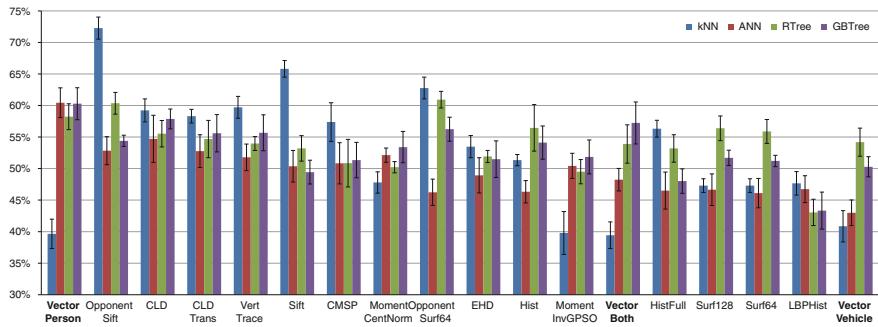
During re-identification experiments, 18 different feature vectors have been used; 15 of them are formed with single descriptors, presented in Sect. 12.4. Additionally, 3 combined vectors have been used that contain descriptors selected as the best ones for re-identification of persons, vehicles and both groups, based on experiments employing descriptor evaluation measures (Sect. 12.7.2). Due to different nature of local image features and other descriptors, both parameter groups cannot be mixed up within the same feature vector. Therefore, comparing to Table 12.9, combined feature vectors VectorPerson, VectorVehicle and VectorBoth contain only standard (non-local) descriptors. They are presented in Table 12.11.

Taking into account the number of feature vectors examined (18) and the number of classifiers used (4), the total quantity of single identification tasks performed is equal to 20,304 for PERSON dataset and 32,544 for VEHICLE dataset. In order to make the results robust against the selection of training and validation samples, all identification tasks are repeated 5 times, with different, random selection of both sets. Furthermore, classifications with ANN and RTree (for the given training and validation sets) are further repeated 5 times due to the stochastic nature of the classifiers' learning. Therefore, the re-identification tasks are computationally very complex and full experiments take a few days to complete on a modern PC. The averaged results from all iterations for PERSON and VEHICLE datasets are reported in Figs. 12.21 and 12.22, separately for each classifier used. Table 12.12 presents results averaged over all classifiers.

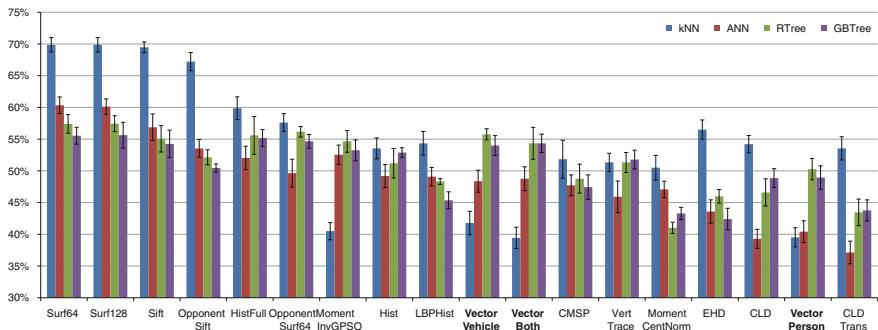
Absolute accuracy values of object re-identification (i.e. the results of finding the one object in the group of a few) are not impressive at the first sight, but taking into

**Table 12.11** Combined visual feature vectors

Feature vector name	Descriptors included in the vector
VectorPerson	CLDTrans, CLD, EHD, MomentInvGPSO
VectorVehicle	HistFull, MomentInvGPSO, VertTrace
VectorBoth	CLDTrans, MomentInvGPSO, HistFull, EHD



**Fig. 12.21** Average object identification results for different feature vectors and classifiers, in descended order of the accuracy averaged over all classifiers—PERSON set



**Fig. 12.22** Average object identification results for different feature vectors and classifiers, in descended order of the accuracy averaged over all classifiers—VEHICLE set

account the large diversity of object poses and camera orientations in the datasets and hard conditions of experiments that are unlikely to happen in real-life scenarios, they could be interpreted as promising. However, the main purpose of the experiments was to find the best descriptors for object re-identification.

Standard deviation values of results obtained for the tested classifiers and feature vectors are similar to each other and low (less than 4 % points). This proves that results obtained are independent of the selection of the specific objects into classifier training and validation sets.

Analysing results for used classifiers it may be noticed that kNN classifier achieved better results using local image features, comparing to other classifiers. In the same time, it performed worse with combined feature vectors. Averaging over all feature vectors, RTree classifier performed best for PERSON dataset (54.02 % accuracy) and kNN—for VEHILCE dataset (54.49 %). On the other hand, ANN classifier performed the worst for both datasets (49.72 % and 48.97 %, respectively). This may be caused by the training set that contained a few object only, the amount too small for proper neural network training. Although the accuracy span among classifiers is small (approx. 5 % point), it is larger than the averaged standard deviation of

**Table 12.12** Object identification results averaged over all classifiers

PERSON dataset		VEHICLE dataset			
Feature vector	Mean (%)	Standard deviation (%)	Feature vector	Mean (%)	Standard deviation (%)
<b>VectorPerson</b>	57.54	2.31	Surf64	60.56	1.31
OpponentSift	56.40	1.65	Surf128	60.44	1.41
CLD	55.77	2.31	Sift	57.88	1.80
CLDTrans	54.05	2.40	OpponentSift	54.68	1.16
VertTrace	53.54	1.95	HistFull	53.85	1.99
Sift	52.54	1.93	OpponentSurf64	52.07	1.37
CMSp	51.72	3.22	MomentInvGPSO	51.38	1.56
MomentCentNorm	51.51	1.55	Hist	50.44	1.63
OpponentSurf64	51.39	1.75	LBPHist	49.17	1.27
EHD	50.19	2.10	<b>VectorVehicle</b>	49.16	1.50
Hist	49.19	2.25	<b>VectorBoth</b>	48.99	1.88
MomentInvGPSO	49.16	2.50	CMSp	48.32	2.21
<b>VectorBoth</b>	48.97	2.57	VertTrace	47.99	1.75
HistFull	48.76	2.09	MomentCentNorm	46.26	1.28
Surf128	48.58	1.69	EHD	45.34	1.53
Surf64	48.12	1.55	CLD	43.26	1.62
LBPHist	45.97	2.25	<b>VectorPerson</b>	42.59	1.69
<b>VectorVehicle</b>	45.04	2.08	CLDTrans	40.80	1.84

results achieved by each classifier (less than 2 % points). Therefore, for object re-identification tasks, kNN and RTree classifiers seem to be the most suited.

Classification results obtained for different descriptors are coherent with the analysis performed with descriptor evaluation measures (Sect. 12.7.2); the best candidate descriptors obtained during this analysis turned out to perform better than majority of other descriptors. Especially, CLD, CLDTrans and OpponentSIFT descriptors provide better results for PERSON dataset and SURF64, OpponentSIFT, HistFull and MomentInvGPSO—for VEHICLE dataset. The only major difference between descriptor evaluation and identification results is caused by VertTrace descriptor that performs very well for PERSON dataset, although descriptor evaluation measures state otherwise. However, such a performance is consistent with the descriptor extraction method that should provide meaningful clues for people images.

Furthermore, it is clear that in case of vehicles, local image features outperform other descriptors, while for people, descriptors based on whole image representations are leading. This is due to the fact that vehicle images are basically texture-less and very similar to each other, thus a proper vehicle identification may be performed by looking at local, distinguishable features only.

Combined feature vectors, stemmed from descriptor evaluation measures (Table 12.11), turned out to outperform single feature based vectors. In case of PERSON dataset, VectorPerson achieved the best results (on average), while VectorVehicle provided the worst outcome. For VEHICLE dataset, VectorVehicle feature vector results are at the top (excluding local image features that perform the best and cannot be mixed with other descriptors), however four single descriptors (HistFull, MomentInvGPSO, Hist and LBPHist) provide some slightly better results; VectorPerson feature vector is penultimate. The vector VectorBoth proposed as the universal one, is characterized with a mediocre performance; it is better than the combined feature vector not designed for the object type, but significantly worse than the suited combined feature vector and many single descriptors. This means that it is hard to provide the content of a feature vector that would be universal and accurate of all types of objects; the feature vector should be adjusted to the object type in order to obtain a high performance.

## 12.8 Conclusions

The problem of object re-identification in multiple cameras required solving the problems of extracting important features of moving objects from a video stream and efficient comparing the descriptor sets in order to match the same object in two cameras. In order to select only distinctive descriptors, a comparative analysis of many visual features was presented. A wide range of descriptors have been studied, including the ones based on colour, texture, gradient and local image features. Evaluation of the descriptors was performed for two image datasets containing persons and vehicles. On the basis of the experiments, features that are best suited for object re-identification in multi-camera environment were selected.

The proposed method of descriptors evaluation is based on the measurement of direct dissimilarity between pairs of images of objects. This method is supplemented with two other methods known from the literature. In order to combine the results achieved for various datasets, four aggregation measures were introduced. Their goal is to find descriptors that provide the best results based on the relative ranking, whereas they are simultaneously characterized with large stability, i.e. their effectiveness does not depend on the selection of particular objects in the dataset. The proposed descriptors are evaluated practically with object re-identification experiments involving four classifiers to detect the same object after its transition between cameras' fields of view.

Results achieved show that there are no visual descriptors that provide the best results for both datasets, simultaneously. In case of human images, the best results were provided by descriptors based on colour, gradient, and image statistical moments. In case of vehicles, the favourable ones are descriptors utilizing colour information, image statistical moments, and local features. The most universal descriptors, providing adequate effectiveness both for humans and vehicles, are the statistical-based ones.

Future work will focus on automatic classification of object type, and shape parametrization methods. Both approaches are expected to improve re-identification results, and to provide means for creating a hybrid “first classify, then re-identify” approach-based solutions.

**Acknowledgments** Research is subsidized by the European Commission within FP7 project “ADDPRI” (“Automatic Data relevancy Discrimination for a PRIVacy-sensitive video surveillance”, Grant Agreement No. 261653). The authors wish to thank the Gdańsk Science and Technology Park for their help in establishing the test bed for the experiments described in the chapter.

## References

1. Allen, R., Mcgeorge, P., Pearson, D., Milne, A.B.: Attention and expertise in multiple target tracking. *Appl. Cogn. Psychol.* **18**(3), 337–347 (2004)
2. Antani, S., Kasturi, R., Jain, R.: A survey on the use of pattern recognition methods for abstraction, indexing and retrieval of images and video. *Pattern Recognit.* **35**(4), 945–965 (2002)
3. Bannour, H., Hlaoua, L., El Ayeb, B.: Survey of the adequate descriptor for content-based image retrieval on the web: global versus local features. In: Conference en Recherche d’Information et Applications CORIA, pp. 445–456. LSIS-USTV (2009)
4. Baxes, G.A.: Digital Image Processing: Principles and Applications. Wiley, New York (1994)
5. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (SURF). *Comput. Vis. Image Underst.* **110**(3), 346–359 (2008)
6. Breiman, L.: Random forests. *Mach. Learn.* **45**(1), 5–32 (2001)
7. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. Chapman and Hall, New York (1994)
8. Burger, W., Burge, M.J.: Principles of Digital Image Processing: Core Algorithms. Springer, Berlin (2009)
9. Burger, W., Burge, M.J.: Principles of Digital Image Processing: Fundamental Techniques. Springer, New York (2009)
10. Cavanagh, P., Alvarez, G.A.: Tracking multiple targets with multifocal attention. *Trends Cogn. Sci.* **9**(7), 349–354 (2005)
11. Chang-yeon, J.: Face detection using LBP features. Final project report—CS 229 machine learning (2008)
12. Cipolla, R., Battiatto, S., Farinella, G.: Computer Vision: Detection, Recognition and Reconstruction. Studies in Computational Intelligence. Springer, New York (2010)
13. Clausi, D.A.: An analysis of co-occurrence texture statistics as a function of grey level quantization. *Can. J. Remote Sens.* **28**(1), 45–62 (2002)
14. Czyżewski, A., Lisowski, K.: Employing flowgraphs for forward route reconstruction in video surveillance system. *J. Intell. Inf. Syst.* **40**, 1–15 (2013)
15. Dalka, P., Szwoch, G., Ciarkowski, A.: Distributed framework for visual event detection in parking lot area. In: Dziech, A., Czyżewski, A. (eds.) *Multimedia Communications, Services and Security. Communications in Computer and Information Science*, vol. 149, pp. 37–45. Springer, Berlin (2011)
16. Dalka, P., Szwoch, G., Szczuko, P., Czyżewski, A.: Video content analysis in the Urban area telemonitoring system. In: Tsirhirtzis, G.A., Jain, L.C. (eds.) *Multimedia Services in Intelligent Environments, Smart Innovation, Systems and Technologies*, vol. 3, pp. 241–261. Springer, Berlin (2010)
17. Doyle, A., Lippert, R., Lyon, D.: Eyes Everywhere : The Global Growth of Camera Surveillance. Routledge, London (2012)

18. Ellis, T.J., Makris, D., Black, J.K.: Learning a multi-camera topology. In: Proceedings of Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, pp. 165–171 (2003)
19. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. *Ann. Stat.* **29**, 1189–1232 (2000)
20. Geng, X., Wang, L., Li, M., Wu, Q., Smith-Miles, K.: Adaptive fusion of gait and face for human identification in video. In: Proceedings of IEEE Workshop on Applications of Computer Vision WACV, pp. 1–6 (2008)
21. Gonzalez, R.C., Woods, R.E.: *Digital Image Processing*, 3rd edn. Prentice-Hall Inc., Upper Saddle River (2006)
22. Halkidi, M., Vazirgiannis, M., Batistakis, Y.: Quality scheme assessment in the clustering process. In: Zighed, D.A., Komorowski, J., Zytkow, J. (eds.) *Principles of Data Mining and Knowledge Discovery. Lecture Notes in Computer Science*, vol. 1910, pp. 265–276. Springer, Berlin (2000)
23. Hamdoun, O., Moutarde, F., Stanciulesscu, B., Steux, B.: Person re-identification in multi-camera system by signature based on interest point descriptors collected on short video sequences. In: Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras ICDSC, pp. 1–6 (2008)
24. Hanbury, A., Kandaswamy, U., Adjeroh, D.A.: Illumination-invariant morphological texture classification. In: Ronse, C., Najman, L., Decencière, E. (eds.) *40 Years On Mathematical Morphology. Computational Imaging and Vision*, vol. 30, pp. 377–386. Springer, Netherlands (2005)
25. Haralick, R., Shanmugam, K., Dinstein, I.: Textural features for image classification. *IEEE Trans. Syst. Man Cybern. SMC* **3**(6), 610–621 (1973)
26. Ilyas, A., Scuturici, M., Miguët, S.: Inter-camera color calibration for object re-identification and tracking. In: Proceedings of International Conference of Soft Computing and Pattern Recognition (SoCPaR), pp. 188–193 (2010)
27. Jeong, K., Jaynes, C.: Object matching in disjoint cameras using a color transfer approach. *Mach. Vis. Appl.* **19**(5–6), 443–455 (2008)
28. Kale, K.: *Advances in Computer Vision and Information Technology*. I.K. International Publishing House Pvt. Limited, New Delhi (2008)
29. Kettnaker, V., Zabih, R.: Bayesian multi-camera surveillance. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, p. 259 (1999)
30. Kim, H., Romberg, J., Wolf, W.: Multi-camera tracking on a graph using Markov chain Monte Carlo. In: Proceedings of the 3rd ACM/IEEE International Conference on Distributed Smart Cameras, ICDSC, pp. 1–8 (2009)
31. Kramer, A.F., Hahn, S.: Splitting the beam: distribution of attention over noncontiguous regions of the visual field. *Psychol. Sci.* **6**(6), 381–386 (1995)
32. Li, J.H., Liu, M.S., Song, P.: A novel modified extraction method of MPEG-7 visual descriptor for image retrieval. In: Proceedings of International Conference on Machine Learning and Cybernetics (ICMLC), vol. 4, pp. 2037–2041 (2010)
33. Liu, Y., Li, Z., Xiong, H., Gao, X., Wu, J.: Understanding of internal clustering validation measures. In: Proceedings of IEEE 10th International Conference on Data Mining (ICDM), pp. 911–916 (2010)
34. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **60**(2), 91–110 (2004)
35. Manjunath, B., Ohm, J.R., Vasudevan, V., Yamada, A.: Color and texture descriptors. *IEEE Trans. Circuits Syst. Video Technol.* **11**(6), 703–715 (2001)
36. Martinez, J.M.: MPEG-7 overview. <http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm> (2004)
37. Mindru, F., Tuytelaars, T., Gool, L.V.: Moment invariants for recognition under changing viewpoint and illumination. *Comput. Vis. Image Underst.* **94**(1–3), 3–27 (2004). Special Issue: Colour for Image Indexing and Retrieval

38. Mittal, A., Davis, L.: Unified multi-camera detection and tracking using region-matching. In: Proceedings of IEEE Workshop on Multi-Object Tracking, pp. 3–10 (2001)
39. Orr, G., Muller, K.: Neural Networks: Tricks of the Trade. Springer, New York (1998)
40. Piciarelli, C., Foresti, G.: Surveillance-oriented event detection in video streams. *IEEE Intell. Syst.* **26**(3), 32–41 (2011)
41. Pinheiro, A.: Image descriptors based on the edge orientation. In: Proceedings of the 4th International Workshop on Semantic Media Adaptation and Personalization SMAP, pp. 73–78 (2009)
42. Pitas, I.: Digital Image Processing Algorithms and Applications. Wiley-Interscience Publication, New York (2000)
43. Sharma, S.: Applied multivariate techniques. Wiley, New York (1996)
44. Spyrou, E., Borgne, H.L., Mailis, T., Cooke, E.: Fusing MPEG-7 visual descriptors for image classification. In: Proceedings of International Conference on Artificial Neural Networks (ICANN), pp. 847–852. Springer (2005)
45. Szeliski, R.: Computer Vision: Algorithms and Applications. Texts in Computer Science. Springer, Heidelberg (2010)
46. Szwoch, G., Dalka, P., Czyżewski, A.: A framework for automatic detection of abandoned luggage in airport terminal. In: Tsihrintzis, G., Damiani, E., Virvou, M., Howlett, R., Jain, L.C. (eds.) Intelligent Interactive Multimedia Systems and Services, Smart Innovation, Systems and Technologies, vol. 6, pp. 13–22. Springer, Berlin (2010)
47. Teixeira, L.F., Corte-Real, L.: Video object matching across multiple independent views using local descriptors and adaptive learning. *Pattern Recognit. Lett.* **30**(2), 157–167 (2009)
48. Tian, Y.L., Hampapur, A., Brown, L., Feris, R., Lu, M., Senior, A., Shu, C.F., Zhai, Y.: Event detection, query, and retrieval for video surveillance, Chap. Artificial Intelligence for Maximizing Content Based Image Retrieval, Information Science Reference (2008)
49. Van de Sande, K.E.A., Gevers, T., Snoek, C.G.M.: Evaluating color descriptors for object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **32**(9), 1582–1596 (2010)
50. Wallraven, C., Caputo, B., Graf, A.: Recognition with local features: the kernel recipe. In: Proceedings of the 9th IEEE International Conference on Computer Vision, vol. 1, pp. 257–264 (2003)

# Chapter 13

## Improving the Recognition Performance of Moment Features by Selection

George A. Papakostas

**Abstract** This chapter deals with the selection of the most appropriate moment features used to recognize known patterns. This chapter aims to highlight the need for selection of moment features subject to their descriptive capabilities. For this purpose, some popular moment families are presented and their properties, making them suitable for pattern recognition tasks, are discussed. Two different types of feature selection algorithms, a simple Genetic Algorithm (GA) and the Relief algorithm are applied to select the moment features that better discriminate human faces and facial expressions, under several pose and illumination conditions. Appropriate experiments using four benchmark datasets have been conducted in order to investigate the theoretical assertions. An extensive experimental analysis has shown that the recognition performance of the moment features can be significantly improved by selecting them from a predefined pool, relative to a specific application.

**Keywords** Moment descriptors · Pattern recognition · Feature selection · Genetic algorithms · Relief algorithm

### 13.1 Introduction

Nowadays, many advanced intelligent systems take part into humans' daily life helping them to satisfy possible professional or entertainment needs. Thus, advanced human computer/machine interaction [3], human identity authentication [10], biometric authentication [30] and surveillance systems [28], have been developed and proposed. Such systems mainly consist of a pattern recognition procedure, which enables the system to interact with the surrounding environment.

---

G.A. Papakostas (✉)

Human Machines Interaction (HMI) Laboratory, Department of Computer and Informatics Engineering, Eastern Macedonia and Thrace (EMT)  
Institute of Technology, Ag. Loukas 65404, Kavala, Greece  
e-mail: gppapak@teikav.edu.gr

The successful operation of the pattern recognition procedure is mainly based on the representation method of the real patterns in a form suitable to be manipulated and managed by the recognition module (classifier). In the case of image based systems the content of an image pattern has to be transformed in a compact numerical format (or other) by applying a feature extraction method (FEM). The role of a FEM is twofold; it performs a dimensionality reduction from the space of image pixels to a small set of numbers and it captures the discriminative characteristics of the patterns in order to distinguish them.

A popular feature extraction method for the case of image patterns is the method of moments. Image moments have proved to be efficient descriptors of the images' content, with many applications in pattern recognition [2, 22, 25, 36], computer vision [12, 23], image analysis [33, 37], image watermarking [27] etc. Among the several moment types, the orthogonal moments [4] constitute the most prominent moment features (discrimination features based on moments) due to their minimum redundancy and high reconstruction capabilities. Additionally, their inherent properties staying invariant under common geometrical transformations (rotation, scaling, translation, flipping) or incorporating such invariances through coordinates transformation, give them all the desirable advantages for any invariant pattern recognition task.

However, a common drawback is the absence of a prior knowledge regarding the number and the suitability of the used moment features being controlled by adjusting the order of the orthogonal polynomial used as kernel function. A common practice is to compute all the moment features up to a certain order and to apply the entire set of moments as discriminative features. This is an "ad hoc" practice in some sense, since the significance of each moment component in discriminating the patterns of the application is not taken into account. A possible solution to this issue is the application of an additional process that selects, from a large pool, the moment features that best perform in terms of recognition accuracy.

The aforementioned issue, of the used moments' appropriateness, constitutes the main subject of this chapter. Initially, the main properties of some representative moment features and their representation capabilities are discussed in Sect. 13.2. Section 13.3 focuses on the justification of the need for selection of the moment features that better describe the distinctive characteristics of the patterns. The selection of moment features by applying two different types of selection algorithms, a Genetic Algorithm and the Relief algorithm, is presented in Sect. 13.4. An extensive experimental study with four benchmark pattern recognition datasets and selected moment features subsets aims to justify the initial assertions in Sect. 13.5. Finally, Sect. 13.6 summarizes and discusses the main conclusions.

## 13.2 Image Moment Features

Geometric moments were the first type of moments introduced in image analysis and pattern recognition [9]. These moments constitute the projection of the image on the monomial  $x^n y^m$  of  $(n + m)$ th order. However, the geometric moments suffer from

high information redundancy making them less efficient in difficult problems where more discriminative information needs to be captured, due to monomials' lack of orthogonality increasing their information redundancy.

This fact has motivated scientists to develop the orthogonal moments, which use as kernel functions orthogonal polynomials that constitute orthogonal basis. The property of orthogonality gives to the corresponding moments the feature of minimum information redundancy, meaning that different moment orders describe different image content.

Initially, the orthogonal moments defined in the continuous space were introduced [26], such as Zernike, Pseudo-Zernike, Fourier-Mellin and Legendre moments. Although these moments were widely applied in many disciplines for a long time, their performance is degraded by several approximation errors [32] generated mainly due to coordinates normalization and space granulation procedures.

Recently, enhanced orthogonal moments free from approximation errors and directly defined inside the discrete coordinate system of the image, were proposed to overcome the disadvantages of the continuous moments. The most representative moment families of discrete form are the Tchebichef [19], Krawtchouk [34] and dual Hahn [14, 37] moments.

It is worth noting that the main research directions across which most scientists work with, in the field of image moments are the following: (1) the development of new algorithms that accelerate the overall moments' computation time, (2) the improvement of the moments' accuracy by reducing the quantization and approximation errors and (3) the embodiment of invariance capabilities into the moments' computation regarding the major linear image's transformations (translation, rotation and scaling). The last direction is relative to the capabilities of the moment features to achieve high recognition rates exploiting invariant behaviour under the aforementioned three basic transformations. Herein, only the description capabilities of the moment features in terms of recognition accuracy will be studied, without paying any attention to the invariant versions of them.

The most representative orthogonal moment families of both continuous and discrete coordinate space are hereafter described and analyzed experimentally.

### 13.2.1 Continuous Orthogonal Moments

In the previous section it has already been mentioned that the first type of orthogonal moments for images (2-D) was defined in the continuous coordinate space of a continuous intensity function  $f(x, y)$ . However, in order to use those moments with digital images, which are defined in the discrete domain, an approximation was applied the so-called zeroth order approximation (ZOA). These two different definitions are as follows:

$$M_{nm} = NF_1 \iint_A Kernel_{nm}(x, y)f(x, y) dx dy \quad (13.1)$$

where  $A$  is the computation coordinate space,  $Kernel_{nm}(\cdot)$  corresponds to the moment's kernel (product of two polynomials) consisting of specific orthogonal polynomials of order  $n$  and  $m$ , which constitute the orthogonal basis and  $NF_1$  is a normalization factor. The type of Kernel's polynomials gives the name to the moment family and thus a wide range of different moment types can be derived.

The zeroth order approximation of Eq. (13.1) for a  $N \times N$  image having intensity function  $f(x, y)$  has the form:

$$(ZOA) : M_{nm} = NF_2 \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} Kernel_{nm}(x, y)f(x, y) \quad (13.2)$$

where  $NF_2$  is a normalization factor and the double integral of Eq. (13.1) is substituted by a double summation, by incorporating some approximation error. The minimization of this error has been the subject of many works [31], which try to propose a discrete computation form that converges to the theoretical values. Three representative moment families of this category will be described in details in the next sections.

### 13.2.1.1 Zernike Moments

Zernike moments are the most widely used family of orthogonal moments due to their inherent property of being invariant to an arbitrary rotation of the image they describe. The main characteristic of this moment family is the usage of a set of complex polynomials as basis, which forms a complete orthogonal set over the interior of the unit circle  $x^2 + y^2 = 1$ . These polynomials have the form:

$$V_{nm}(r, \theta) = R_{nm}(r)e^{jm\theta} \quad (13.3)$$

where  $n$  is a non-negative integer and  $m$  an integer subject to the constraints  $n - |m|$  even and  $|m| \leq n$ ,  $r(r = \sqrt{x^2 + y^2})$  is the length of the vector from the origin to the pixel and  $\theta (\theta = \tan^{-1}(y/x))$  is the angle between the vector  $r$  and  $x$ -axis in counter-clockwise direction.  $R_{nm}(r)$ , are the Zernike radial polynomials [35], in  $(r, \theta)$  polar coordinates defined as:

$$R_{nm}(r) = \sum_{s=0}^{\frac{n-|m|}{2}} (-1)^s \frac{(n-s)!}{s! \left(\frac{n+|m|}{2} - s\right)! \left(\frac{n-|m|}{2} - s\right)!} r^{n-2s} \quad (13.4)$$

Note that  $R_{n(-m)}(r) = R_{nm}(r)$ .

The Zernike moment of order  $n$  with repetition  $m$  for a  $N \times N$  pixels size continuous image function  $f(x, y)$ , that vanishes outside the unit disk, has the form:

$$Z_{nm} = \frac{n+1}{\pi} \int_0^{2\pi} \int_0^1 V_{nm}^*(r, \theta) f(r, \theta) r dr d\theta \quad (13.5)$$

where the symbol (\*) corresponds to conjugate.

For a digital image, the integrals are replaced according to the zeroth order approximation Eq. (13.2) by summations to get:

$$Z_{nm} = \frac{n+1}{\pi} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} V_{nm}^*(r_{ij}, \theta_{ij}) f(i, j). \quad (13.6)$$

The above transformation from continuous to discrete form adds some approximation errors. For this reason, several attempts [32] to decrease these approximation errors have been reported in the literature. Moreover, significant work has been done [7] in the last years towards the fast computation of the radial polynomials (Eq. 13.4) and the moments (Eq. 13.6).

### 13.2.1.2 Legendre Moments

The  $(n+m)$ th order Legendre moment [6] of an intensity function  $f(x, y)$ , is defined in  $[-1, 1]$  as:

$$L_{nm} = \frac{(2n+1)(2m+1)}{4} \int_{-1}^1 \int_{-1}^1 P_n(x) P_m(y) f(x, y) dx dy \quad (13.7)$$

where  $P_n(x)$  is the  $n$ th order Legendre polynomial defined as:

$$P_n(x) = \sum_{k=0}^n \alpha_{k,n} x^k = \frac{1}{2^n n!} \left( \frac{d}{dx} \right)^n (x^2 - 1)^n \quad (13.8)$$

The above Legendre polynomials satisfy the following recursive equation:

$$\begin{aligned} P_n(x) &= [(2n-1)xP_{n-1}(x) - (n-1)P_{n-2}(x)]/n \\ P_0(x) &= 1, \quad P_1(x) = x \end{aligned} \quad (13.9)$$

The recursive formula of Eq. (13.10) permits the fast computation of the Legendre polynomials by using polynomials of lower order. In case of computing the Legendre moments of a  $N \times N$  image, Eq. (13.7) takes the following discrete form:

$$L_{nm} = \frac{(2n+1)(2m+1)}{(N-1)(N-1)} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} P_n(x) P_m(y) f(x, y). \quad (13.10)$$

The computation of the Legendre moments through Eq. (13.10) shows significant approximation errors as discussed lately and the resulted Legendre moments do not satisfy the properties of the theoretical ones, by affecting their ability to describe the image in process. For this reason new algorithms ensuring the accurate computation of the moments have been proposed [6].

### 13.2.1.3 Gaussian-Hermite Moments

Gaussian-Hermite moments are continuous moments that have been introduced in image analysis quite recently by Yang and Dai [33]. The  $(n+m)$ th order Gaussian-Hermite moment is defined in  $(-\infty, +\infty)$  and has the form:

$$GH_{nm} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{H}_n(x; \sigma) \hat{H}_m(y; \sigma) f(x, y) dx dy \quad (13.11)$$

where

$$\hat{H}_n(x; \sigma) = \frac{1}{\sqrt{2^n n! \sigma \sqrt{\pi}}} e^{(-x^2/2\sigma^2)} H_n(x; \sigma) \quad (13.12)$$

is the weighted Hermite orthonormal polynomial of order  $n$ , derived by the ordinary Hermite polynomial  $H_n(x; \sigma)$ , modulated by a Gaussian function with  $\sigma$  variance. The ordinary Hermite polynomial of order  $n$  is defined as:

$$H_n(x) = n! \sum_{k=0}^{\lfloor n/2 \rfloor} \frac{(-1)^k}{k!(n-2k)!} (2x)^{n-2k} \quad (13.13)$$

The recursive computation of the above Hermite polynomials is performed according to:

$$\begin{aligned} H_{n+1}(x) &= 2x H_n(x) - 2n H_{n-1}(x), \quad \text{for } n \geq 1 \\ H_0(x) &= 1, \quad H_1(x) = 2x \end{aligned} \quad (13.14)$$

The Gaussian-Hermite moments have proved to be of higher image representation ability [33], compared to some traditional moment families e.g. Legendre and thus their usage has been rapidly increased in many applications of the engineering life.

Due to this popularity, a faster and more accurate computation algorithm has been proposed recently [8].

### 13.2.2 Discrete Orthogonal Moments

The aforementioned drawback of the continuous orthogonal moments, has motivated scientists in the field of image moments to develop more accurate moment families. This goal has been achieved by the introduction of the discrete orthogonal moments being defined directly on the discrete image coordinates space. Some of the most representative discrete moment families are discussed herein.

#### 13.2.2.1 Tchebichef Moments

This moment family is the first proposed in the literature by Mukundan et al. [19]. The  $(n+m)$ th order Tchebichef moment of a  $N \times N$  image having intensity function  $f(x, y)$  is defined as:

$$T_{nm} = \frac{1}{\tilde{\rho}(n, N)\tilde{\rho}(m, N)} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \tilde{t}_n(x) \tilde{t}_m(y) f(x, y) \quad (13.15)$$

where  $\tilde{t}_n(x)$  is the  $n$ th order normalized Tchebichef polynomial, introduced in order to ensure numerical stability and moments' limited dynamical range, defined as follows:

$$\tilde{t}_n(x) = \frac{t_n(x)}{\beta(n, N)} \quad (13.16)$$

where the ordinary Tchebichef polynomial of  $n$  order has the form:

$$\begin{aligned} t_n(x) &= (1-N)_n {}_3F_2(-n, -x, 1+n; 1, 1-N; 1) \\ &= \sum_{k=0}^n (-1)^{n-k} \binom{N-1-k}{n-k} \binom{n+k}{n} \binom{x}{k}. \end{aligned} \quad (13.17)$$

In the above formulas,  ${}_3F_2$ , is the generalized hypergeometric function,  $n, x = 0, 1, 2, \dots, N-1, N$  is the image size and  $\beta(n, N)$  is a suitable constant independent of  $x$  that serves as a scaling factor, such as  $N^n$ . Moreover,  $\tilde{\rho}(n, N)$  is the normalized norm of the Tchebichef polynomials defined as:

$$\tilde{\rho}(n, N) = \frac{\rho(n, N)}{\beta(n, N)^2} \quad (13.18)$$

with

$$\rho(n, N) = (2n)! \binom{N+n}{2n+1}, \quad n = 0, 1, \dots, N-1. \quad (13.19)$$

The computation speed of Tchebichef moments can be accelerated by using the following recursive formula:

$$\begin{aligned} n\tilde{t}_n(x) &= (2n-1)\tilde{t}_{n-1}(x) - (n-1)\left(1 - \frac{(n-1)^2}{N^2}\right)\tilde{t}_{n-2}(x) \\ \tilde{t}_0(x) &= 1, \quad \tilde{t}_1(x) = \frac{2x+1-N}{N} \end{aligned} \quad (13.20)$$

Tchebichef moments have proved to be superior to Zernike and Legendre moments in describing objects, while their robustness in the presence of high noise levels makes them appropriate to real-time pattern classification and computer vision applications.

### 13.2.2.2 Krawtchouk Moments

The Krawtchouk orthogonal moments are the second type of discrete moments introduced in image analysis by Yap et al. [34]. The  $(n+m)$ th order orthogonal discrete Krawtchouk moment of a  $N \times N$  image having intensity function  $f(x, y)$  is defined as:

$$K_{nm} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \bar{K}_n(x; p_1, N-1) \bar{K}_m(y; p_2, N-1) f(x, y) \quad (13.21)$$

where

$$\bar{K}_n(x; p, N) = K_n(x; p, N) \sqrt{\frac{w(x; p, N)}{\rho(n; p, N)}} \quad (13.22)$$

is the weighted Krawtchouk polynomial of  $n$  order, used to reduce the numerical fluctuations presented in the ordinary Krawtchouk polynomials, defined as:

$$K_n(x; p, N) = {}_2F_1\left(-n, -x; -N; \frac{1}{p}\right) = \sum_{k=0}^N \alpha_{k,n,p} x^k. \quad (13.23)$$

In Eq. (13.22)  $\rho(n; p, N)$  is the norm of the Krawtchouk polynomials having the form:

$$\rho(n; p, N) = (-1)^n \left(\frac{1-p}{p}\right)^n \frac{n!}{(-N)_n}, \quad n = 1, \dots, N \quad (13.24)$$

and  $w(x; p, N)$  is the weight function of the Krawtchouk moments,

$$w(x; p, N) = \binom{N}{x} p^x (1-p)^{N-x} \quad (13.25)$$

In Eq. (13.24) the symbol  $(\cdot)_n$  is the Pochhammer symbol, which for the general case is defined as  $(\alpha)_k = \alpha(\alpha+1)\dots(\alpha+k+1)$ .

In practice, the computation of the weighted Krawtchouk polynomials is not performed through Eq. (13.22), since this is a very time consuming procedure; instead, a recursive algorithm [34] is applied:

$$\begin{aligned} p(N-n)\bar{K}_{n+1}(x; p, N) &= A(Np - 2np + n - x)\bar{K}_n(x; p, N) \\ &\quad - Bn(1-p)\bar{K}_{n-1}(x; p, N) \end{aligned} \quad (13.26)$$

where

$$A = \sqrt{\frac{p(N-n)}{(1-p)(n+1)}}, \quad B = \sqrt{\frac{p^2(N-n)(N-n+1)}{(1-p)^2(n+1)n}} \quad (13.27)$$

and

$$\bar{K}_0(x; p, N) = \sqrt{\frac{w(x; p, N)}{\rho(0; p, N)}}, \quad \bar{K}_1(x; p, N) = \left(1 - \frac{x}{pN}\right) \sqrt{\frac{w(x; p, N)}{\rho(1; p, N)}} \quad (13.28)$$

The Krawtchouk moments proved to be effective local descriptors, since they can describe the local features of an image, unlike the other moment families, which capture only the global features of the objects they describe. This locality property is controlled by appropriate adjustment of the  $p_1, p_2$  parameters of Eq. (13.21).

### 13.2.2.3 Dual Hahn Moments

The  $(n+m)$ th order orthogonal dual Hahn moment [37] of a  $N \times N$  image having intensity function  $f(x, y)$  is defined as:

$$W_{nm} = \sum_{x=a}^{b-1} \sum_{y=a}^{b-1} \hat{W}_n^{(c)}(x, a, b) \hat{W}_m^{(c)}(y, a, b) f(x, y), \quad n, m = 0, 1, \dots, N-1 \quad (13.29)$$

where  $-\frac{1}{2} < a < b$ ,  $|c| < 1 + a$ ,  $b = a + N$  and

$$\hat{W}_n^{(c)}(s, a, b) = W_n^{(c)}(s, a, b) \sqrt{\frac{\rho(s)}{d_n^2} \left[ \Delta x \left( s - \frac{1}{2} \right) \right]} \quad (13.30)$$

is the  $n$ th order weighted dual Hahn polynomial used to reduce the numerical instabilities caused by the ordinary dual Hahn polynomials, defined as:

$$W_n^{(c)}(s, a, b) = \frac{(a-b+1)_n (a+c+1)_n}{n!} {}_3F_2(-n, a-s, a+s+1; a-b+1, a+c+1; 1) \quad (13.31)$$

for  $n = 0, 1, \dots, N-1$ ,  $s = a, a+1, \dots, b-1$ , where  ${}_3F_2$  is the generalized hypergeometric function given by:

$${}_3F_2(a_1, a_2, a_3; b_1, b_2; z) = \sum_{k=0}^{\infty} \frac{(a_1)_k (a_2)_k (a_3)_k}{(b_1)_k (b_2)_k} \frac{z^k}{k!} \quad (13.32)$$

In the above formulas  $\rho(s)$  is the weighting function defined in terms of the gamma function  $\Gamma(\cdot)$  as:

$$\rho(s) = \frac{\Gamma(a+s+1) \Gamma(c+s+1)}{\Gamma(s-a+1) \Gamma(b-s) \Gamma(b+s+1) \Gamma(s-c+1)} \quad (13.33)$$

and

$$d_n^2 = \frac{\Gamma(a+c+n+1)}{n! (b-a-n-1)! \Gamma(b-c-n)}, \quad n = 0, 1, \dots, N-1. \quad (13.34)$$

It is obvious from the above equations that the computation of dual Hahn polynomials is a time consuming task, so efficient recursive algorithms need to be used [37].

### 13.2.3 Image Reconstruction by the Method of Moments

A significant and useful property of the orthogonal moments is their ability to reconstruct the image they describe. The reconstruction of a  $N \times N$  image by using moment orders up to  $n_{max}$  and  $m_{max}$  is described by the following inverse formula of Eq.(13.2):

$$\hat{f}(x, y) = \sum_{n=0}^{n_{max}} \sum_{m=0}^{m_{max}} \overline{Kernel_{nm}}(x, y) M_{nm} \quad (13.35)$$

where  $M_{nm}$  is the  $(n+m)$ th order moment and  $\overline{Kernel_{nm}}(x, y)$  the kernel function of the used polynomial family, which is not always the same as Eq.(13.2), e.g. for the case of the Zernike moments it is equal to the conjugate of  $Kernel_{nm}(x, y)$ .

It is worth to note that the reconstruction ability of each moment family is an indication of its information compactness, which is highly connected with the amount of image's information enclosed by the moment coefficients. An extensive analysis of the reconstruction performance of the orthogonal moments has been reported in [21].

### 13.2.4 Moments Interpretation

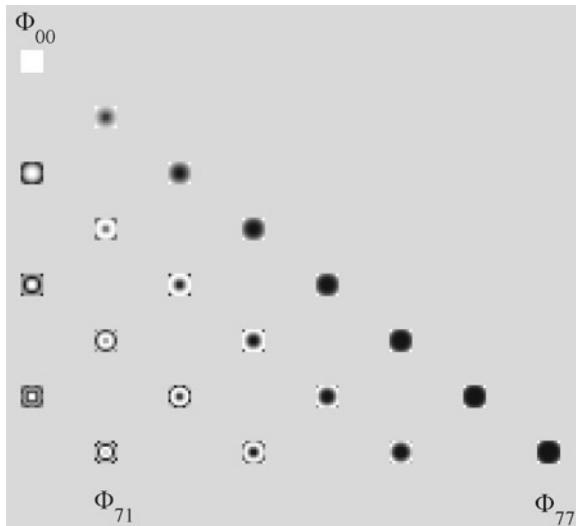
The conventional characterization of the orthogonal image moments defines them as the projections of an image to the orthogonal basis of the used polynomials. However, this definition encloses more mathematical than engineering or computer science oriented knowledge. According to a different perspective from an engineering and computer science point of view, the orthogonal moments represent the similarity between the image and a number of image patterns formed by the kernel function of each moment family.

In order to better understand the latter proposition, the image patterns derived from the kernels of the pre-analyzed moment families need to be calculated. These image patterns are called *basis images* and they are computed by applying the following formula:

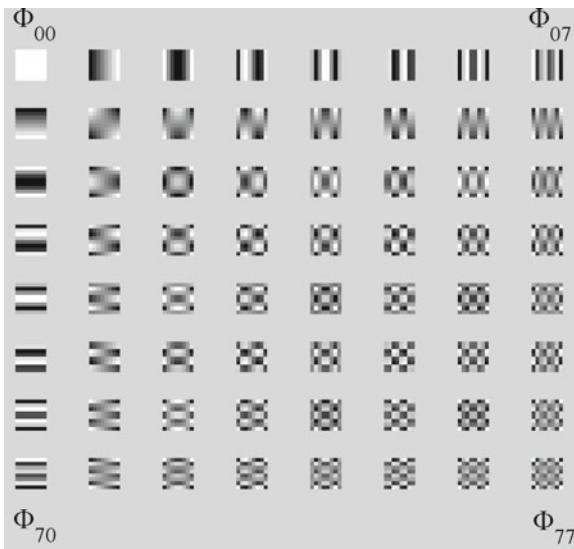
$$\Phi_{nm} = [\mathbf{Poly}_n]^T \mathbf{Poly}_m \quad (13.36)$$

where  $\mathbf{Poly}_n$  and  $\mathbf{Poly}_m$  are vectors with the values of the  $n$ th and  $m$ th order polynomial for each image pixel. The computed basis images for the case of an  $8 \times 8$  square image having constant intensity equal to 1, for the cases of the continuous orthogonal moments (up to order 7) are depicted in Figs. 13.1, 13.2 and 13.3.

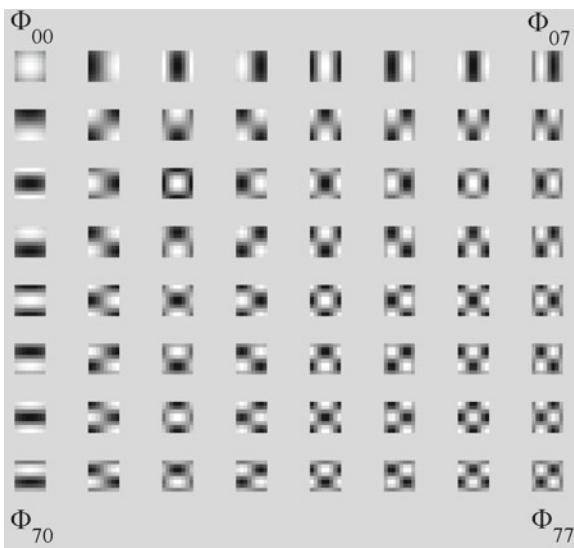
By examining the basis images of Figs. 13.1, 13.2 and 13.3 it is noticeable that the patterns provided by each moment type are totally different. The Zernike moments (Fig. 13.1) generate circular patterns due to the used radial polynomials Eq. (13.2), while some images do not exist due to the constraints hold between the  $n$  and  $m$  parameters.



**Fig. 13.1** Basis images of the continuous orthogonal Zernike moments

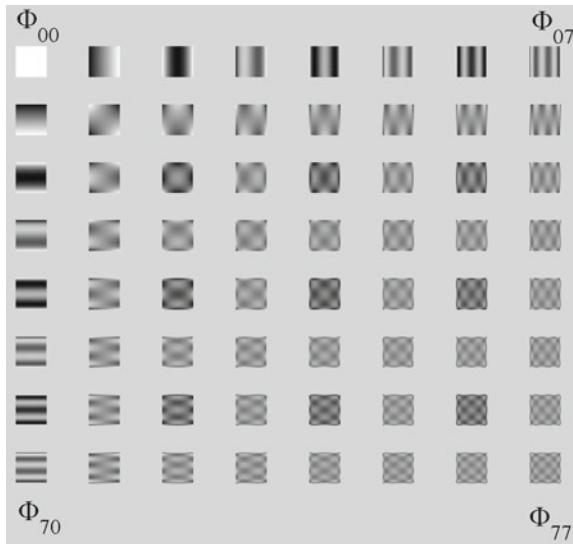


**Fig. 13.2** Basis images of the continuous orthogonal Legendre moments



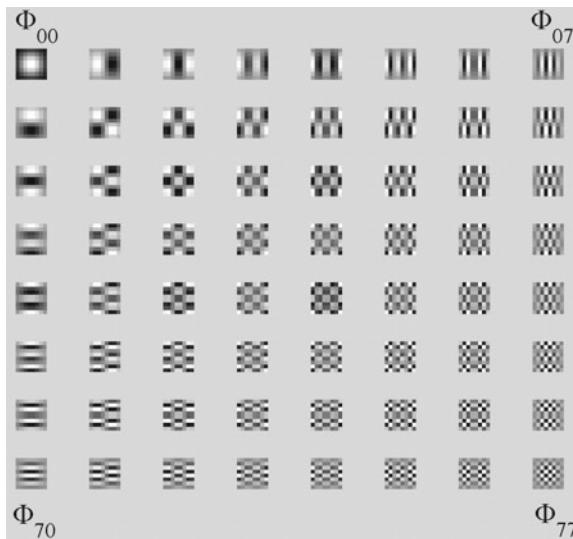
**Fig. 13.3** Basis images of the continuous orthogonal Gaussian-Hermite ( $\sigma = 1$ ) moments

Concerning the basis images of the rest continuous moments it can be noted that the formed patterns include more details as the moment order increases, while for low order these patterns are coarse. This observation deals with what is commonly

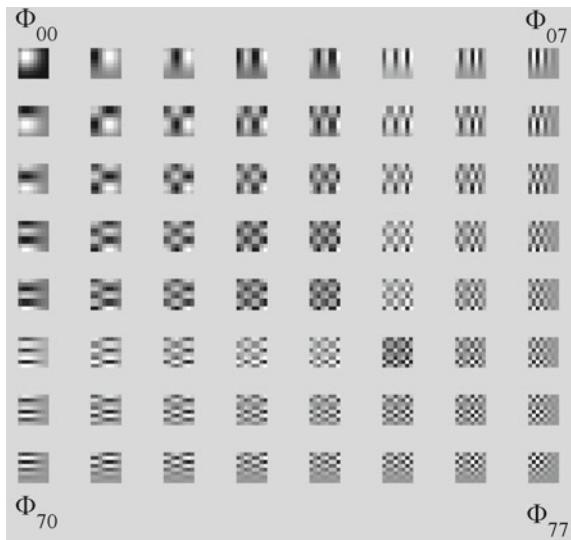


**Fig. 13.4** Basis images of the discrete orthogonal Tchebicichef moments

stated regarding the description capabilities of different moment orders. The corresponding basis images for the case of the discrete orthogonal moments are illustrated in Figs. 13.4, 13.5 and 13.6.



**Fig. 13.5** Basis images of the discrete orthogonal Krawtchouk ( $p_1 = p_2 = 0.5$ ) moments



**Fig. 13.6** Basis images of the discrete orthogonal dual Hahn ( $a = c = 0, b = 8$ ) moments

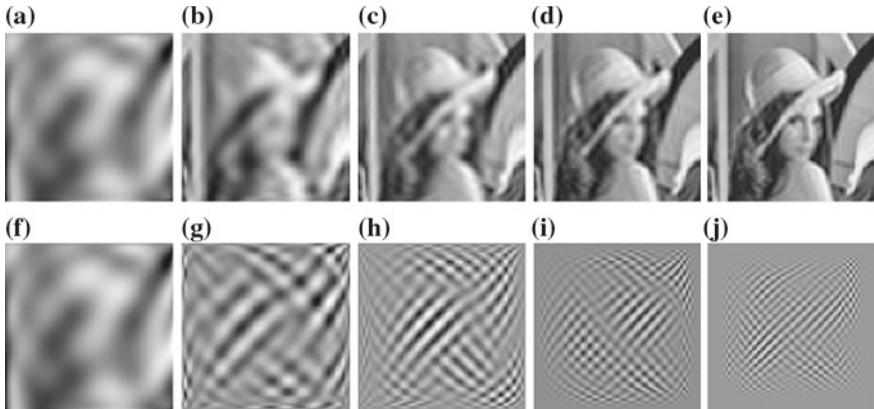
The local behaviour of the Krawtchouk and dual Hahn moments is obvious from the above basis images of the discrete orthogonal moments. This constitutes the most advantageous property of these moments, since they localize the window of interest in a specific image portion. The previous analysis considering the basis images of the orthogonal moments highlights the different representation capabilities of each moment type for different moment orders. According to this study each moment carries different image's content and therefore the selection of the moments that better discriminate some patterns seems to be a reasonable and inevitable process.

### 13.3 Is There a Need for Moments' Selection?

Apart from the previous analysis of moments' representation capabilities, the execution of a certain experiment regarding moments' description ability, would be constructive to further highlight the need for selection of the most appropriate moment sets.

For this purpose the well known Lena benchmark image with  $64 \times 64$  pixels size in grayscale format, is reconstructed with various sets of Tchebichef moments having different orders. The reconstruction results for moment sets consisting of 10 *different* orders are depicted in Fig. 13.7. It has to be noted that in each reconstructed image the intensities are normalized into the  $[0, 255]$  range for illustration purposes.

From the reconstructed images of Fig. 13.7, it can be deduced that as the moment order increases more detailed information of the image's content emerges.



**Fig. 13.7** Lena image reconstruction using several sets of Tchebichef moments of various orders: **a** 0–9, **b** 0–19, **c** 0–29, **d** 0–39, **e** 0–49, **f** 0–9, **g** 10–19, **h** 20–29, **i** 30–39 and **j** 40–49

For example, the orders' range 0–9 (1st row of Fig. 13.7) is able to reconstruct a quite coarse image's content, while by adding the next 10 orders (0–19) some detailed information is incorporated. This observation is in agreement with the image's content described by each moment set (2nd row of Fig. 13.7), where it is obvious that the higher order moments sets model the high frequency pixels variations.

Considering the above analysis, the moments of low orders are not so useful in discriminating patterns which differ slightly, since their differences are described in the high moment orders. For example, if a second image of the above Lena benchmark is constructed with Lena having her eyes closed, the two image patterns could not be discriminated by the low order moments but high orders are needed.

Therefore, it is evident that the appropriate set of moments, better discriminating some specified patterns, depends on the application and thus a selection procedure considering patterns' modalities is inevitable.

## 13.4 Moment Features Selection

By examining the recent literature in the field of image moments, one can reach the conclusion that little work has been done towards the moments' selection [13, 20].

The main selection method applied to all the aforementioned works is the Genetic Algorithm (GA), proved as an efficient wrapper selection technique [24] taking into account the classification model applied to recognize the patterns. Genetic Algorithms are a great example of evolutionary computation mimicking the evolutionary process that characterizes the evolution of living organisms [5]. However, the main disadvantage of the GA-based selection is the high computation time need to converge the algorithm to a suitable solution.

The aforementioned drawback of the GA-based selection makes the filter selection methods [24] an attractive alternative approach. These methods do not use the mining model (they are independent of the classification model), instead the internal data properties/characteristics (dependency, correlation etc.) are taken into consideration.

For the sake of the experiments presented in the next section, the GA-based (wrapper) and the Relief [15] algorithm (filter) selection methods will be applied for the selection of the proper moment subsets that better discriminate the patterns of some benchmark pattern recognition datasets. These two algorithms are briefly discussed in the next sections.

### 13.4.1 GA-Based Selection

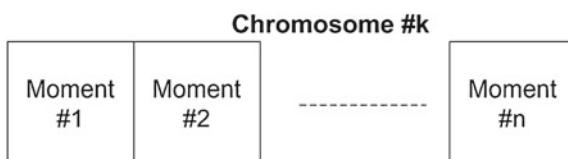
The main operational element of a Genetic Algorithm is the chromosome. The chromosome corresponds to a candidate solution to the problem at hand, consisting of the set of variables appropriately coded. For the case of the GA-based moment selection method, the chromosome consists of the indices (Fig. 13.8) of a predefined number of moments. The indices correspond to the moment id belonging to the moment feature vector, which is constructed by arranging the computing moments according to the zigzag scanning operation [11].

Initially, a pool of 100 moment features is constructed by computing all the moments up to a specific order. Considering that a number of  $n$  moments are required to be selected, the  $k$ th chromosome structure of the GA is depicted in Fig. 13.8.

Furthermore, the objective function being minimized by the GA is equal to the recognition error (Wrong Recognized Patterns/Total Patterns) derived when the selected moment sets are fed to the classifier model.

### 13.4.2 Relief Algorithm

Relief algorithm [15] is a popular feature selection method due to its simplicity. It is based on the computation of the relevance between pairs of feature vectors. The relevance is measured by applying the L-dimensional Euclidean distance, where L is the dimension of the feature vectors being compared. This algorithm selects those features which are relevant subject to a defined threshold in linear time.



**Fig. 13.8** Chromosome structure

## 13.5 Experimental Study

A set of experiments were conducted in order to study the impact of the selected moment subsets on the overall recognition performance in several pattern recognition problems. For this purpose, appropriate software was developed in the MATLAB 2012b environment, while all experiments were executed in an Intel i5 3.3 GHz PC with 8 GB RAM. Moreover, four well known benchmark datasets were used to evaluate the initial assertion of the moments' selection significance, towards the improvement of the recognition accuracy.

### 13.5.1 Benchmark Datasets

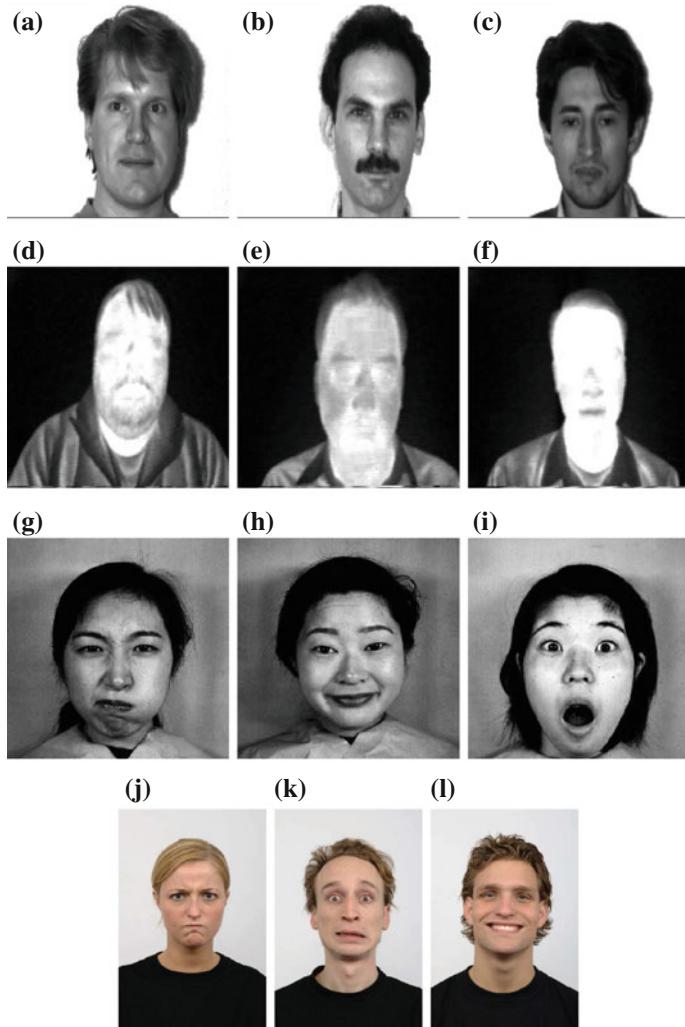
Four benchmark datasets are used in order to investigate the selection performance of the Relief and Genetic Algorithms in selecting moment subsets of various sizes. The considered datasets are the Yale face recognition dataset [1], a subset [22] of the Terravic thermal infrared face recognition [18], the JAFFE [17] and RABDOUD [16] facial expressions datasets. Three sample images (different classes) from each dataset are depicted in Fig. 13.9, while the properties of each dataset are summarized in Table 13.1.

### 13.5.2 Datasets Pre-processing

It is worth noting that before computing the moment features, the images need to be pre-processed in order to remove irrelevant image information (background, hair, ears, etc.) and to isolate the image's part, which includes the main face information. For this purpose, the Viola-Jones face detector [29] is applied being followed by an ellipse masking [11], for the case of Yale, JAFFE and Radboud datasets. The aforementioned face detector fails to detect faces in thermal infrared images and

**Table 13.1** Benchmark datasets properties

Dataset	Type	Number of classes	Samples/class	Total samples
Yale	Face recognition	15	11	165
Terravic	Thermal infrared face recognition	10	70	700
JAFFE	Facial expression recognition	7	30, 29, 32, 31, 30, 31, 30	213
Radboud	–	8	67	536



**Fig. 13.9** Samples of the four benchmarks: **a–c** Yale, **d–f** Terravic, **g–i** JAFFE and **j–l** RADBOUD

thus the procedure proposed in [22] is applied for the Terravic dataset. The outcome of this processing stage is a cropped image of  $128 \times 128$  pixels size, which includes the most relative information to the face image content.

### 13.5.3 Simulations

A large scale experimental study has taken place in order to extract useful conclusions towards the improvement of the moment features' recognition performance through the application of a selection mechanism. The Genetic Algorithm settings are: population size 50, maximum generations 30, crossover with probability 0.8 and 2 points, mutation probability 0.01 and Stochastic Universal Approximation (SUS) selection method. The k-NN classifier ( $k = 1$ ) is selected as the prediction model in the case of the GA-based selection. Moreover, a 10-fold cross-validation technique is applied in all datasets, while the moments are selected from a pool of the first 100 (up to 9 order for all moments, except Zernike computed up to 18 order) computed moments for each moment family.

The corresponding mean recognition rates for each dataset are summarized in Tables 13.2, 13.3, 13.4 and 13.5. The best moment family along with the best selection method is presented in these results.

By examining Tables 13.2, 13.3, 13.4 and 13.5 it is deduced that in almost all cases the selected moment features show better or equal, in the worst case, performance than the non selected (NoSel.) moments. This outcome enforces the initial assertion regarding the needs for moment features selection. Moreover, among the two examined selection methods, the GA-based one seems to be more efficient for small sized moment subsets (up to 25–30), while the Relief algorithm is superior for larger subsets (greater than 30–40). This observation can be justified by the fact that for large moment subsets (greater than 30) the optimization problem, which needs to be solved by the GA, is quite difficult. One solution to this limitation is to use more advanced versions of the algorithm, where adaptive crossover and/or mutation operators could guide the algorithm to more optimum solutions.

**Table 13.2** Recognition performance of moment features subsets for the Yale dataset

Yale dataset			
Number of moments	Moment type	Recognition rate (%)	Selection method
5	Krawtchouk	76.00	GA
10	Krawtchouk	86.66	GA
15	Zernike	88.00	GA
20	Zernike	87.33	GA
25	Dual Hahn	84.66	GA
30	Krawtchouk	72.66	Relief
40	Krawtchouk	75.33	Relief
50	Krawtchouk	74.66	Relief
60	Krawtchouk	76.66	Relief
70	Dual Hahn	76.00	Relief

**Table 13.3** Recognition performance of moment features subsets for the Terravic dataset

Terravic dataset

Number of moments	Moment type	Recognition rate (%)	Selection method
5	All	100.00	NoSel./GA
10	All	100.00	NoSel./GA
15	All	100.00	NoSel./GA
20	All	100.00	NoSel./GA
25	All	100.00	NoSel./GA
30	All	100.00	NoSel./Relief
40	All	100.00	NoSel./Relief
50	All	100.00	NoSel./Relief
60	All	100.00	NoSel./Relief
70	All	100.00	NoSel./Relief

**Table 13.4** Recognition performance of moment features subsets for the JAFFE dataset

JAFFE Dataset

Number of moments	Moment type	Recognition rate (%)	Selection method
5	Legendre	71.66	GA
10	Krawtchouk	79.90	GA
15	Legendre	78.90	GA
20	Krawtchouk	77.47	GA
25	Krawtchouk	69.33	GA
30	Krawtchouk	53.92	GA
40	Dual Hahn	47.35	Relief
50	Zernike	46.88	Relief
60	Gaussian-Hermite	46.80	Relief
70	Legendre	46.88	NoSel.

From the above results, it can be observed that the increase of the number of moments used to discriminate the patterns does not always improve the recognition accuracy. In almost all the cases a subset of 10–25 moment features is able to achieve the highest recognition rate.

In order to draw a conclusion regarding the optimal settings, ensuring the best solution to each dataset, the most effective configuration in each case is summarized in Table 13.6.

The results of Table 13.6 show again the outperformance of the GA-based selection method over the Relief one, while its recognition accuracy is in agreement with the state of the art methods [11, 22, 25]. As far as the performance of the moment families is concerned, it is obvious that Zernike moments are the most efficient family, while a moments' subset of size lower than 25 is adequate to ensure an acceptable recognition

**Table 13.5** Recognition performance of moment features subsets for the Radboud dataset

Radboud dataset				
Number of moments	Moment type	Recognition rate (%)	Selection method	
5	Zernike	48.21	GA	
10	Zernike	55.75	GA	
15	Zernike	61.96	GA	
20	Zernike	61.38	GA	
25	Zernike	62.14	GA	
30	Zernike	54.10	GA	
40	Zernike	53.30	Relief	
50	Zernike	53.66	Relief	
60	Zernike	53.30	Relief	
70	Zernike	51.47	No Sel./Relief	

**Table 13.6** Best configuration for each dataset

Dataset	Number of moments	Moment type	Recognition rate (%)	Selection method
Yale	15	Zernike	88.00	GA
Terravic	5	All	100.00	No Sel./GA
JAFFE	10	Krawtchouk	79.90	GA
Radboud	25	Zernike	62.14	GA

accuracy. However, Krawtchouk moments show a significant performance leading to the conclusion that the locality property plays an important role to capture the local characteristics of the patterns. More work has to be done in this direction of describing the local information by the method of moments.

## 13.6 Conclusions

A detailed discussion of the main properties of the most representative image orthogonal moment families was presented in the previous sections. Through an in depth analysis of the representation capabilities of the orthogonal moments, the need for selection of moment features for improved recognition accuracy is highlighted. Finally, an extensive experimental study on well known benchmark datasets has resulted in useful conclusions regarding the initial assertion of moment's selection and the description capability of each moment family. The GA-based selection method has shown superior performance to the Relief algorithm, mainly for low number of moments, while for high number of features the latter algorithm seems to be the suitable choice.

Moreover, the Zernike moments seem to be the most discriminant moment family compared to other moments, since they recognized the patterns of the three datasets with the highest rate. Also, one additional outcome of the experiments was the out-performance of the Krawtchouk moments to the JAFFE dataset. This result set the basis of a future study regarding the selection of moments belonging to different moment families, in order to take advantage of the properties each family presents.

Conclusively, an initial claim was set and proved both theoretically and experimentally, by establishing the selection of moment features as a mandatory processing step of any modern pattern recognition system.

## References

1. Belhumeur, P.N., Kriegman, D.J.: The Yale face database. <http://cvc.yale.edu/projects/yalefaces/yalefaces.html> (1997)
2. Chen, B.J., Shu, H.Z., Zhang, H., Chen, G., Toumoulin, C., Dillenseger, J.L., Luo, L.M.: Quaternion Zernike moments and their invariants for color image analysis and object recognition. *Signal Process.* **92**(2), 308–318 (2012)
3. Cipolla, R., Pentland, A.: Computer vision for human-machine interaction. Cambridge University Press, Cambridge (1998)
4. Flusser, J., Zitova, B., Suk, T.: Moments and Moment Invariants in Pattern Recognition. Wiley, Chichester (2009)
5. Holland, J.H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence. U Michigan Press, Ann Arbor (1975)
6. Hosny, K.M.: Exact Legendre moment computation for gray level images. *Pattern Recognit.* **40**(12), 3597–3605 (2007)
7. Hosny, K.M.: Fast computation of accurate Zernike moments. *J. Real-Time Image Process.* **3**(1–2), 97–107 (2008)
8. Hosny, K.M.: Fast computation of accurate Gaussian-Hermite moments for image processing applications. *Digit. Signal Process.* **22**(3), 476–485 (2012)
9. Hu, M.K.: Visual pattern recognition by moment invariants. *IRE Trans. Inf. Theory* **8**(2), 179–187 (1962)
10. Jain, A.K., Hong, L., Pankanti, S., Bolle, R.: An identity-authentication system using fingerprints. *Proc. IEEE* **85**(9), 1365–1388 (1997)
11. Kaburlasos, V.G., Papadakis, S.E., Papakostas, G.A.: Lattice computing extension of the FAM neural classifier for human facial expression recognition. *IEEE Trans. Neural Netw. Learn. Syst.* **24**(10), 1526–1538 (2013)
12. Kadir, A., Nugroho, L.E., Santosa, P.I.: Experiments of Zernike moments for leaf identification. *J. Theor. Appl. Inf. Technol.* **41**(1), 82–93 (2012)
13. Kanan, H.R., Faez, K.: GA-based optimal selection of PZMI features for face recognition. *Appl. Math. Comput.* **205**(2), 706–715 (2008)
14. Karakasis, E.G., Papakostas, G.A., Koulouriotis, D.E., Tourassis, V.D.: Generalized dual Hahn moment invariants. *Pattern Recognit.* **46**(7), 1998–2014 (2013)
15. Kira, K., Rendell, L.A.: A practical approach to feature selection. In: Proceedings of the 9th International Workshop on Machine Learning, pp. 249–256. Morgan Kaufmann Publishers Inc. (1992)
16. Langner, O., Dotsch, R., Bijlstra, G., Wigboldus, D.H., Hawk, S.T., van Knippenberg, A.: Presentation and validation of the Radboud faces database. *Cogn. Emot.* **24**(8), 1377–1388 (2010)

17. Lyons, M., Akamatsu, S., Kamachi, M., Gyoba, J.: Coding facial expressions with Gabor wavelets. In: Proceedings of the 3rd IEEE International Conference on Automatic Face and Gesture Recognition, pp. 200–205. IEEE (1998)
18. Miezianko, R.: Terravic research infrared database. In: IEEE OTCBVS WS Series Bench. IEEE (2005)
19. Mukundan, R., Ong, S.H., Lee, P.A.: Image analysis by Tchebichef moments. *IEEE Trans. Image Process.* **10**(9), 1357–1364 (2001)
20. Papakostas, G.A., Boutilis, Y.S., Mertzios, B.G.: Evolutionary selection of Zernike moment sets in image processing. In: Proceedings of the 10th International Workshop on Systems, Signals and Image Processing (IWSSIP'03), pp. 10–11 (2003)
21. Papakostas, G.A., Karakasis, E.G., Koulouriotis, D.E.: Orthogonal image moment invariants: highly discriminative features for pattern recognition applications. In: Cross-Disciplinary Applications of Artificial Intelligence and Pattern Recognition: Advancing Technologies, pp. 34–52. IGI Global (2012)
22. Papakostas, G.A., Kaburlasos, V.G., Pachidis, T.: Thermal infrared face recognition based on lattice computing (LC) techniques. In: 2013 IEEE International Conference on Fuzzy Systems (FUZZ), pp. 1–6. IEEE (2013)
23. Papakostas, G.A., Boutilis, Y.S., Karras, D.A., Mertzios, B.G.: A new class of Zernike moments for computer vision applications. *Inf. Sci.* **177**(13), 2802–2819 (2007)
24. Papakostas, G.A., Koulouriotis, D.E., Polydoros, A.S., Tourassis, V.D.: Evolutionary feature subset selection for pattern recognition applications. In: Evolutionary Algorithms, pp. 443–458. InTech (2011)
25. Papakostas, G.A., Koulouriotis, D.E., Karakasis, E.G., Tourassis, V.D.: Moment-based local binary patterns: a novel descriptor for invariant pattern recognition applications. *Neurocomputing* **99**(1), 358–371 (2013)
26. Teague, M.R.: Image analysis via the general theory of moments. *JOSA* **70**(8), 920–930 (1980)
27. Tsougenis, E.D., Papakostas, G.A., Koulouriotis, D.E., Tourassis, V.D.: Performance evaluation of moment-based watermarking methods: a review. *J. Syst. Softw.* **85**(8), 1864–1884 (2012)
28. Velasin, S.A., Remagnino, P.: Intelligent Distributed Video Surveillance Systems, vol. 5. IET, London (2006)
29. Viola, P., Jones, M.J.: Robust real-time face detection. *Int. J. Comput. Vis.* **57**(2), 137–154 (2004)
30. Wayman, J., Jain, A., Maltoni, D., Maio, D.: An introduction to biometric authentication systems. *Biometric Systems*, pp. 1–20. Springer, London (2005)
31. Wee, C.Y., Paramesran, R.: On the computational aspects of Zernike moments. *Image Vis. Comput.* **25**(6), 967–980 (2007)
32. Xin, Y., Pawlak, M., Liao, S.: Accurate computation of Zernike moments in polar coordinates. *IEEE Trans. Image Process.* **16**(2), 581–587 (2007)
33. Yang, B., Dai, M.: Image analysis by Gaussian-Hermite moments. *Signal Process.* **91**(10), 2290–2303 (2011)
34. Yap, P.T., Paramesran, R., Ong, S.H.: Image analysis by Krawtchouk moments. *IEEE Trans. Image Process.* **12**(11), 1367–1377 (2003)
35. Zernike, F.V.: Beugungstheorie des schneidenverfahrens und seiner verbesserten form, der phasenkontrastmethode. *Physica* **1**(7), 689–704 (1934)
36. Zhang, F., Liu, S.Q., Wang, D.B., Guan, W.: Aircraft recognition in infrared image using wavelet moment invariants. *Image Vis. Comput.* **27**(4), 313–318 (2009)
37. Zhu, H., Shu, H., Zhou, J., Luo, L., Coatrieux, J.L.: Image analysis by discrete orthogonal dual Hahn moments. *Pattern Recognit. Lett.* **28**(13), 1688–1704 (2007)

# Chapter 14

## Signature Selection for Grouped Features with a Case Study on Exon Microarrays

Sangkyun Lee

**Abstract** When features are grouped, it is desirable to perform feature selection groupwise in addition to selecting individual features. It is typically the case in data obtained by modern high-throughput genomic profiling technologies such as exon microarrays, which measure the amount of gene expression in fine resolution. Exons are disjoint subsequences corresponding to coding regions in genes, and exon microarrays enable us to study the event of different usage of exons, called alternative splicing, which is presumed to contribute to development of diseases. To identify such events, all exons that belong to a relevant gene may have to be selected, perhaps with different weights assigned to them to detect most relevant ones. In this chapter we discuss feature selection methods to handle grouped features. A popular shrinkage method, lasso, and its variants will be our focus, that are based on regularized regression with generalized linear models. Data from exon microarrays will be used for a case study.

**Keywords** Penalized regression · Lasso · Group lasso · Sparsity · Convex regularization

### 14.1 Introduction

Group information of features provides us a way to perform feature selection in different resolutions. That is, not only individual features (high resolution) but also groups comprising these features (low resolution) can be considered for selection when they are relevant. Examples of group information include:

- Population census data, where each record consists of demographic, economic and social features. Grouped feature selection may identify that demographic features are important for predicting years in education,

---

S. Lee (✉)

Technische Universität Dortmund, Joseph-von-Fraunhofer-Strasse 23,  
44227 Dortmund, Germany  
e-mail: sangkyun.lee@tu-dortmund.de

- Measurements from sensors deployed in different parts of a machine. A task will be finding parts that affect sensitivity in operation, as well as individual sensors that matter inside,
- Gene expression measured in exon level. Exons correspond to coding regions of genes, and they are translated to proteins that are functional in cells. Identifying clinically important genes, as well as detecting their different usage of exons, is an important task in biomedical studies,
- Genes that belong to different cellular components, different biological functions, or different molecular functions according to the Gene Ontology.<sup>1</sup>

In the examples above, groups of features are used to represent associations of features that come from our prior knowledge.

Another type of groups comes from our design on features, for instance, when we perform feature selection on multinomial covariates. Suppose that a feature  $z \in \{A, B, C\}$  is represented with dummy variables  $x_1$  and  $x_2$ , so that  $(x_1, x_2) = (0, 1)$ ,  $(1, 0)$ , and  $(1, 1)$  represent  $A$ ,  $B$ , and  $C$ , respectively. When  $z$  is relevant, then it would make sense to select both  $x_1$  and  $x_2$ ; otherwise, both variables should not be selected. Therefore dummy variables that correspond to the same multinomial variable have to be considered as a group.

For both scenarios, the same methods can be applied for grouped feature selection. We will focus on the first type where groups represent our knowledge on features.

In the chapter we discuss feature selection methods that can extract features in both individual and group levels. We focus on a popular shrinkage method called *lasso*, and its extensions to handle grouped features. These methods are often referred to as embedded feature selection methods in machine learning, or penalized (regularized) regression methods in statistics. A characteristic of them is that feature selection is integrated with learning predictors, so there is no need to perform each separately.

### 14.1.1 Regularized Regression

The methods we will discuss in this chapter can be described as optimization problems with a canonical convex minimization formulation,

$$\min_{\beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}} f(\beta, \beta_0) + \Psi(\beta). \quad (14.1)$$

Here the first part  $f(\beta, \beta_0)$  of the objective function represents the amount of loss or error by making incorrect predictions. The second part  $\Psi(\beta)$  is called a *regularizer* or a *penalty* term, which is used to induce certain structure (for example, sparsity) on the coefficient vector  $\beta$ .

---

<sup>1</sup> <http://www.geneontology.org>

### 14.1.1.1 Loss Functions

Suppose that we have a data set composed of  $n$  examples  $\{(\mathbf{x}^i, y^i)\}_{i=1}^n$ , where  $\mathbf{x}^i \in \mathbb{R}^p$  is an input vector with  $p$  features and  $y^i$  is a response of  $\mathbf{x}^i$ . We consider loss functions  $f(\boldsymbol{\beta}, \beta_0)$  that involve generalized linear models. Popular examples include:

- **Ordinary least squares.** This loss function is used for fitting a linear model to real-valued responses  $y^i \in \mathbb{R}$ ,  $i = 1, 2, \dots, n$ :

$$f(\boldsymbol{\beta}, \beta_0) = \frac{1}{2} \sum_{i=1}^n (y^i - \boldsymbol{\beta}^T \mathbf{x}^i - \beta_0)^2. \quad (14.2)$$

Here the inner product between two vectors  $\boldsymbol{\beta}$  and  $\mathbf{x}^i$  has been represented as  $\boldsymbol{\beta}^T \mathbf{x}^i = \sum_{j=1}^p \beta_j x_j^i$ , where  $\beta_j$  is the  $j$ th element of  $\boldsymbol{\beta}$ .

- **Logistic regression.** The loss function is for classifying binary responses  $y^i \in \{-1, +1\}$ :

$$f(\boldsymbol{\beta}, \beta_0) = \sum_{i=1}^n \log \left[ 1 + \exp \left\{ -y^i (\boldsymbol{\beta}^T \mathbf{x}^i + \beta_0) \right\} \right].$$

- **Cox regression.** This loss function is for the case where responses are given by  $y^i = (t^i, e^i)$ :  $t^i \in \mathbb{R}_+$  is the survival time of the  $i$ th patient (whose genetic profile is given by a vector  $\mathbf{x}^i$ ) and  $e^i \in \{0, 1\}$  is an indicator variable ( $e^i = 1$  if the  $i$ th patient had an event,  $e^i = 0$  otherwise). It is defined by

$$f(\boldsymbol{\beta}, \beta_0) = - \sum_{i \in E} \log \frac{\exp(\boldsymbol{\beta}^T \mathbf{x}^i)}{\sum_{j \in R_i} \exp(\boldsymbol{\beta}^T \mathbf{x}^j)},$$

where  $E$  is an index set of all patients who have events, and  $R_i$  is an index set of patients who are at risk at the time  $t^i$ . This is the negative partial log-likelihood function due to the proportional hazard model proposed by [6], which typically appears in survival analysis.

A common property of the three loss functions above is the convexity of  $f$  in its both arguments,  $\boldsymbol{\beta} \in \mathbb{R}^p$  and  $\beta_0 \in \mathbb{R}$ . A function  $f(\boldsymbol{\beta})$  is *convex* in  $\mathbb{R}^p$  if for any  $\alpha \in [0, 1]$ ,

$$f(\alpha \boldsymbol{\beta} + (1 - \alpha) \boldsymbol{\beta}') \leq \alpha f(\boldsymbol{\beta}) + (1 - \alpha) f(\boldsymbol{\beta}'), \quad \forall \boldsymbol{\beta}, \boldsymbol{\beta}' \in \mathbb{R}^p.$$

This is a desirable property, together with the convexity of the regularizer  $\Psi$ , since it facilitates finding a minimizer of (14.1).

In our discussion, we only require that  $f$  is convex and continuously differentiable, except that in some derivations we use the  $f$  of the ordinary least squares because it leads to the simplest derivation.

### 14.1.1.2 Regularizers

A popular choice of the regularizer  $\Psi(\boldsymbol{\beta})$  in (14.1) to perform feature selection (without considering feature groups) is the  $\ell_1$  norm of  $\boldsymbol{\beta}$ , which is also known as the *lasso* penalty [23],

$$\Psi(\boldsymbol{\beta}) := \lambda \|\boldsymbol{\beta}\|_1 = \lambda \sum_{j=1}^p |\beta_j|,$$

for a given  $\lambda > 0$ . This is a convex function in  $\boldsymbol{\beta}$ . The bias term  $\beta_0$  is usually not included in regularization. A property of the lasso penalty is that when a feature is not important for fitting responses with respect to a given value of  $\lambda$ , the lasso penalty sets the corresponding coefficient in  $\boldsymbol{\beta}$  to the exactly zero value. So there is no need for thresholding to filter out irrelevant features after finding solutions of (14.1). In fact, the value of  $\lambda$  plays a similar role to a threshold value, as we will see later.

When features are correlated, lasso tends to select only few out of the correlated features (in an unstable way, especially when  $p \gg n$ ). This is not desirable when all correlated features may matter and therefore have to be selected. A remedy for this behavior is to use the *elastic net* regularization [26], which augments  $\Psi$  above as follows,

$$\Psi(\boldsymbol{\beta}) := \lambda \left\{ \alpha \|\boldsymbol{\beta}\|_1 + (1 - \alpha) \|\boldsymbol{\beta}\|_2^2 \right\}. \quad (14.3)$$

Here  $\|\boldsymbol{\beta}\|_2$  is the  $\ell_2$  norm (the Euclidean norm) of  $\boldsymbol{\beta}$ . The parameter  $\alpha \in [0, 1]$  controls the mixing of the  $\ell_1$  and  $\ell_2$  regularizers: the case of  $\alpha = 0$  is often referred to as the ridge regression, and for  $\alpha = 1$  it becomes the lasso penalty. Elastic net tends to select all correlated features when they are relevant. So correlated groups of features will be identified, but they may not correspond to known groups of features.

The rest of this chapter is organized as follows. In Sect. 14.2, the extensions of lasso, namely the group lasso (Sect. 14.2.1), the overlapping group lasso (Sect. 14.2.2), and the sparse group lasso (Sect. 14.2.3) algorithms are introduced, discussing their properties and differences. A case study on exon microarray data follows in Sect. 14.3, demonstrating a possible use of grouped feature selection in bioinformatics. Some technical issues of the methods are discussed in Sect. 14.4, followed by conclusions in Sect. 14.5.

## 14.2 Regularized Regression Methods for Grouped Features

When group information on features is available, we can impose it as an extra constraint for feature selection. Suppose that  $p$  features are grouped into  $K$  groups, where we represent each group of  $G_1, G_2, \dots, G_K$  as a subset of feature indices, that is,  $G_k \subset \{1, 2, \dots, p\}$ . For simplicity we assume that all features have their groups assigned, in other words  $\cup_{k=1}^K G_k = \{1, 2, \dots, p\}$ .

To facilitate our discussion, we denote a subvector of  $p$  dimensional vector  $\beta$  that corresponds to a group  $G_k$  by

$$\beta_{G_k} := (\beta_j)_{j \in G_k} \in \mathbb{R}^{|G_k|}.$$

Here we consider a fixed permutation of elements in  $\beta_{G_k}$  given by the sorted indices in  $G_k$  in an increasing order. For example, if  $\beta = (a, b, c)^T$  and  $G_1 = \{1, 3\}$ , then  $\beta_{G_1} = (a, c)^T$ , not  $(c, a)^T$ . Also, the cardinality of a finite set  $G_k$  has been denoted by  $|G_k|$ . The definitions of norms and inner products apply naturally on these subvectors, for example  $\|\beta_{G_k}\|_2 = \left(\sum_{j \in G_k} (\beta_j)^2\right)^{1/2}$  and  $\beta_{G_k}^T \mathbf{x}_{G_k} = \sum_{j \in G_k} \beta_j x_j$ .

### 14.2.1 Group Lasso

Group lasso [24] is an extension of the lasso penalty discussed in Sect. 14.1.1 for grouped features. In group lasso we solve the following optimization problem,

$$\min_{\beta, \beta_0} f(\beta, \beta_0) + \Psi_G(\beta), \quad \Psi_G(\beta) := \lambda \sum_{k=1}^K \|\beta_{G_k}\|_2. \quad (14.4)$$

Here note that the  $\ell_2$  norm in summation is not squared: otherwise  $\Psi_G$  becomes equivalent to  $\lambda \|\beta\|_2^2$  when groups are not overlapping.

The regularizer  $\Psi_G$  that replaces the  $\ell_1$  norm of  $\beta$  in lasso is often referred to as the group  $\ell_1/\ell_2$  norm. It computes an  $\ell_1$  norm over groups, and an  $\ell_2$  norm for each group. The reason behind the name becomes obvious when we reformulate  $\Psi_G$  defining a new vector  $\mathbf{z}$ ,

$$\mathbf{z} := \begin{bmatrix} \|\beta_{G_1}\|_2 \\ \vdots \\ \|\beta_{G_K}\|_2 \end{bmatrix}, \quad \Psi_G(\beta) = \lambda \sum_{k=1}^K |\|\beta_{G_k}\|_2| = \lambda \|\mathbf{z}\|_1.$$

That is,  $\Psi_G$  is an  $\ell_1$  norm over groups, and an  $\ell_2$  norm for each group. As a result, group lasso selects few groups that are relevant for a given task as in lasso. If a group of features is chosen, then all of the features inside are typically selected. Otherwise, all corresponding features will have zero coefficients assigned.

#### 14.2.1.1 Computation of Solutions

To see how group lasso performs groupwise feature selection, we need to understand the characterizations of its solutions. In this section we consider group lasso defined with the loss function of the ordinary least squares discussed in Sect. 14.1.1.1.

Before discussing the details of the topic, we simplify the expression of  $f$  by defining a collection of responses  $\mathbf{y} = (y^1, y^2, \dots, y^n)^T \in \mathbb{R}^n$  and a collection of input vectors as rows in a matrix  $\mathbf{X} = (\mathbf{x}^1, \mathbf{x}^2, \dots, \mathbf{x}^n)^T \in \mathbb{R}^{n \times p}$ . Omitting the bias term  $\beta_0$  for further simplification, we can rewrite the expression in (14.2) as,

$$f(\boldsymbol{\beta}) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2. \quad (14.5)$$

We denote the column submatrix of  $\mathbf{X}$  corresponding to a group  $G_k$  by  $\mathbf{X}_{G_k}$ . As in [24], we further assume that the columns in  $\mathbf{X}_{G_k}$  are orthonormal, that is,  $\mathbf{X}_{G_k}^T \mathbf{X}_{G_k} = \mathbf{I}_{|G_k|}$ , an identity matrix of dimension  $|G_k| \times |G_k|$ . Then the group lasso problem formulation in (14.4) defines a convex minimization problem for which solutions can be characterized in a closed form by first-order optimality conditions, as shown in the following theorem. The theorem is from [24], except for that our proof provides more rigorous arguments on how optimality conditions lead to the conclusion, compared to the original proof.

**Theorem 1** *For a group lasso problem defined with  $f(\boldsymbol{\beta}) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$  and  $\mathbf{X}_{G_k}^T \mathbf{X}_{G_k} = \mathbf{I}_{|G_k|}$  for  $k = 1, 2, \dots, K$ , an optimal solution  $\boldsymbol{\beta}$  is given by*

$$\boldsymbol{\beta}_{G_k} = \left( 1 - \frac{\lambda}{\|\mathbf{s}_k\|_2} \right)_+ \mathbf{s}_k, \quad k = 1, 2, \dots, K,$$

where  $\mathbf{s}_k := \mathbf{X}_{G_k}^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}_{-G_k})$ ,  $\boldsymbol{\beta}_{-G_k}$  is a vector with the same elements as  $\boldsymbol{\beta}$  except for having zeros in the components at  $j \in G_k$ , and  $(\cdot)_+ := \max(\mathbf{0}, \cdot)$ .

*Proof* For  $\boldsymbol{\beta}_{G_k} \neq \mathbf{0}$  the objective function of (14.4) is differentiable, and therefore the first-order optimality condition is,

$$-\mathbf{X}_{G_k}^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \frac{\boldsymbol{\beta}_{G_k}}{\|\boldsymbol{\beta}_{G_k}\|_2} = \mathbf{0},$$

Since  $\mathbf{X}_{G_k}^T \mathbf{X}_{G_k} = \mathbf{I}_{|G_k|}$ , this is equal to

$$-\mathbf{X}_{G_k}^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}_{-G_k}) + \left( \frac{\lambda}{\|\boldsymbol{\beta}_{G_k}\|_2} + 1 \right) \boldsymbol{\beta}_{G_k} = \mathbf{0}. \quad (14.6)$$

Suppose that  $\boldsymbol{\beta}_{G_k} = \mathbf{0}$  is optimal. We can consider the gradients of  $f(\boldsymbol{\beta}) + \Psi_G(\boldsymbol{\beta})$  at two nonzero points,  $\boldsymbol{\beta} + \boldsymbol{\eta}$  and  $\boldsymbol{\beta} - \boldsymbol{\eta}$  where  $\boldsymbol{\eta} := (0, \dots, 0, \varepsilon, \dots, \varepsilon, 0, \dots, 0)^T$  is a  $p$  dimensional vector with  $\varepsilon > 0$  at components  $j \in G_k$ . Since  $\boldsymbol{\beta}_{G_k} = \mathbf{0}$  is optimal, the two gradients must satisfy

$$-\mathbf{s}_k + \left( \frac{\lambda}{\varepsilon \sqrt{|G_k|}} + 1 \right) \boldsymbol{\eta}_{G_k} > 0, \quad -\mathbf{s}_k - \left( \frac{\lambda}{\varepsilon \sqrt{|G_k|}} + 1 \right) \boldsymbol{\eta}_{G_k} < 0.$$

Since  $f(\beta)$  is continuously differentiable, tending  $\varepsilon \rightarrow 0$  gives us  $|(\mathbf{s}_k)_j| \leq \lambda / \sqrt{|G_k|}$  for  $j = 1, \dots, |G_k|$ . This implies that

$$\|\mathbf{s}_k\|_2 \leq \left( \sum_{j \in G_k} \lambda^2 / |G_k| \right)^{1/2} = \lambda. \quad (14.7)$$

On the other hand, suppose that  $\beta_{G_k} \neq \mathbf{0}$  is optimal. From (14.6), we get

$$\mathbf{s}_k = \left( \frac{\lambda}{\|\beta_{G_k}\|_2} + 1 \right) \beta_{G_k} \Rightarrow \|\beta_{G_k}\|_2^2 + 2\lambda \|\beta_{G_k}\|_2 + \lambda^2 - \|\mathbf{s}_k\|_2^2 = 0.$$

Since  $\|\beta_{G_k}\|_2 \geq 0$  we obtain  $\|\beta_{G_k}\|_2 = \|\mathbf{s}_k\|_2 - \lambda$ . Replacing  $\|\beta_{G_k}\|_2$  above and rearranging terms, we have

$$\beta_{G_k} = \left( 1 - \frac{\lambda}{\|\mathbf{s}_k\|_2} \right) \mathbf{s}_k.$$

Combining with the condition in (14.7), the claim follows.  $\square$

As we can see, the ratio between  $\lambda$  and the norm of a partial gradient vector  $\|\mathbf{s}_k\|_2$  determines whether to select the corresponding group ( $\|\mathbf{s}_k\|_2 > \lambda$ ) or not ( $\|\mathbf{s}_k\|_2 \leq \lambda$ ). That is, the value of  $\lambda$  acts like a threshold for choosing groups.

It is also worthwhile to note that the optimal subvector  $\beta_{G_k}$  for a selected group  $G_k$  is not necessarily sparse, since there is no mechanism that pushes its coefficients to the zero values, as we can see from the theorem above. Therefore group lasso may not be appropriate for finding features not only groupwise but also within groups, for instance to detect alternative splicing events from exon microarrays.

#### 14.2.1.2 Scaling of Regularization

In group lasso (14.4), we can consider rescaled versions of the regularizer  $\Psi_G$ ,

$$\Psi_G(\beta) = \lambda \sum_{k=1}^K w_k \|\beta_{G_k}\|_2,$$

where a scaling factor  $w_k$  is defined for each group  $k = 1, 2, \dots, K$ . Scaling factors can be defined according to applications. Two examples include:

- $w_k = \sqrt{|G_k|}$ : without scaling, each term  $\|\beta_{G_k}\|_2$  in (14.4) penalizes the square root of the degree of freedom to present a group  $G_k$ . This scaling can be used to penalize degrees of freedom, not square roots of them [16, 24]. This setting is suitable when multinomial variables are represented as groups of dummy variables.
- $w_k = 1/\sqrt{|G_k|}$ : without scaling, the order of magnitude of  $\|\beta_{G_k}\|_2$  is  $O(\sqrt{|G_k|})$ , and therefore larger groups tend to be penalized more than smaller ones in (14.4).

In other words, smaller groups are more likely to be selected. When this behavior is not preferable, we can use this scaling to impose equal amounts of regularization on groups of different sizes.

### 14.2.2 Overlapping Group Lasso

So far, there has been no consideration of overlaps in groups. Graph lasso discussed in Sect. 14.2.1, however, may lead to incorrect feature selection when groups overlap.

Let us take a simple example from [12], with  $\beta = (\beta_1, \beta_2, \beta_3)^T$ ,  $G_1 = \{1, 2\}$ , and  $G_2 = \{2, 3\}$ . When we select none or both of  $G_1$  and  $G_2$ , there is no issue. But selecting only one of them is not permitted by the group lasso regularizer  $\Psi_G$ . That is, if we select  $G_1$  (assigning nonzero values to  $\beta_1$  and  $\beta_2$ ), then  $G_2$  must be selected as well (since  $\beta_2 \neq 0$  implies that  $\|\beta_{G_2}\|_2 \neq 0$ ). In fact, selecting either  $G_1$  or  $G_2$  is allowed only for a very rare case when the optimal value for the shared variable is  $\beta_2 = 0$ . Theorem 1 tells that such case could happen when  $G_1$  is selected and  $\frac{\partial f(\beta, \beta_0)}{\partial \beta_2} = 0$ . However, this is not a property of the regularizer  $\Psi_G$ : it is not capable of selecting only one of overlapping groups.

For the case of overlapping groups, we can use an alternative regularizer proposed by [12]. The idea is rather simple: we consider a separate coefficient vector  $\gamma^k \in \mathbb{R}^p$  for each group  $k = 1, 2, \dots, K$ , which can have nonzero values only for components at  $j \in G_k$ . Using these vectors, the regularizer for overlapping groups is defined by

$$\Psi_O(\beta) := \lambda \inf_{\sum_{k=1}^K \gamma^k = \beta} \sum_{k=1}^K \|\gamma^k\|_2. \quad (14.8)$$

Applying this to our simple example above, we have  $\gamma^1 = (\gamma_1^1, \gamma_2^1, 0)^T$  for  $G_1 = \{1, 2\}$  and  $\gamma^2 = (0, \gamma_2^2, \gamma_3^2)^T$  for  $G_2 = \{2, 3\}$ . Now only one of the groups can be selected since we have separate variables  $\gamma_2^1$  and  $\gamma_2^2$  for the second feature, and therefore choosing  $\gamma_2^1$  does not necessarily imply  $\|\gamma^2\|_2 > 0$ , for instance  $\beta$  is determined by  $\beta = \gamma^1 + \gamma^2$ .

#### 14.2.2.1 Structure Induced by $\Psi_G$ and $\Psi_O$

The difference between  $\Psi_G$  and  $\Psi_O$  can be understood clearer by investigating the structure induced by them on the coefficient vector  $\beta$ . In particular, we discuss the set of nonzero components in  $\beta$ ,  $\{j : \beta_j \neq 0\}$ , which is equal to the set of features to be selected.

In case of  $\Psi_G$  from group lasso, we have

$$\{j : \beta_j \neq 0\} \subset \left( \bigcup_{k: \beta_{G_k} = \mathbf{0}} G_k \right)^c.$$

As we discussed, in group lasso it is not allowed to select a group which has any shared feature with a group that is not selected. That is, feature indices in selected groups cannot belong to any of unselected groups, as stated in the expression above. For an example, consider three groups  $G_1 = \{1, 2, 4\}$ ,  $G_2 = \{2, 3\}$  and  $G_3 = \{4, 5\}$  with  $p = 5$ . Suppose that  $G_2$  and  $G_3$  are not selected. Then the only possible option is to select the 1st feature, that is, the set of candidate features is  $\{1\} \subset (G_2 \cup G_3)^c$ .

On the other hand, in case of  $\Psi_O$  for overlapping groups,

$$\{j : \beta_j \neq 0\} \subset \bigcup_{k: \gamma^k \neq 0} G_k.$$

This is indeed obvious since  $\Psi_O$  allows for selecting any group whenever its associated  $\gamma^k$  vector is nonzero. In the example above with three groups, we can select any combination of  $G_1$ ,  $G_2$  and  $G_3$ .

Note that when groups define a partition of features, that is, there is no overlap amongst groups, then the two expressions above become the same. Therefore it is advised to apply group lasso in Sect. 14.2.1 only if there is no overlap, or if there exists overlap but the set of features that can be selected fits particular purposes.

#### 14.2.2.2 Reformulation to Group Lasso

In the definition of the overlapping group lasso regularizer in (14.8), the coefficient vector  $\beta$  is expressed as the summation of all  $\gamma^k$  vectors over group indices  $k = 1, 2, \dots, K$ , that is,

$$\beta = \sum_{k=1}^K \gamma^k.$$

This result can be used in combination with the loss function  $f(\beta, \beta_0)$  discussed in Sect. 14.1.1.1. Taking the loss function for logistic regression, we get

$$f(\gamma^1, \dots, \gamma^k, \beta_0) := \sum_{i=1}^n \log \left[ 1 + \exp \left\{ -y^i \left( \sum_{k=1}^K (\gamma^k)^T \mathbf{x}^i + \beta_0 \right) \right\} \right].$$

This can be further simplified by constructing two new vectors,

$$\begin{aligned} \tilde{\gamma} &:= \left( (\gamma_{G_1}^1)^T, (\gamma_{G_2}^2)^T, \dots, (\gamma_{G_K}^k)^T \right)^T, \\ \tilde{\mathbf{x}}^i &:= \left( (\mathbf{x}_{G_1}^i)^T, (\mathbf{x}_{G_2}^i)^T, \dots, (\mathbf{x}_{G_K}^i)^T \right)^T. \end{aligned}$$

Here  $\tilde{\gamma}$  is a collection of nonzero components in  $\{\gamma^k\}_{k=1}^K$  vectors, and  $\tilde{\mathbf{x}}^i$  is a copy of an input vector  $\mathbf{x}^i$  with features replicated if they belong to multiple groups. Then the loss function can be rewritten as,

$$f(\tilde{\boldsymbol{\gamma}}, \beta_0) := \sum_{i=1}^n \log \left[ 1 + \exp \left\{ -y^i \left( \tilde{\boldsymbol{\gamma}}^T \tilde{\mathbf{x}}^i + \beta_0 \right) \right\} \right].$$

Finally, we define new groups  $H_k := \{j \mid \sum_{s=1}^{k-1} |G_s| + 1 \leq j \leq \sum_{s=1}^k |G_s|\}$  for  $k = 1, 2, \dots, K$ . Then the optimization problem for overlapping group lasso can be written in the same form as that of group lasso in (14.4),

$$\min_{\tilde{\boldsymbol{\gamma}}, \beta_0} f(\tilde{\boldsymbol{\gamma}}, \beta_0) + \lambda \sum_{k=1}^K \|\tilde{\boldsymbol{\gamma}}_{H_k}\|_2.$$

Therefore, the same algorithm for group lasso can be applied for overlapping group lasso.

### 14.2.3 Sparse Group Lasso

In group lasso (Sect. 14.2.1) and overlapping group lasso (Sect. 14.2.2), the coefficient subvectors corresponding to selected groups are not necessarily sparse, as briefly discussed in Sect. 14.2.1.1. This would be undesirable when we try to identify only a few important features inside selected groups.

An alternative is the sparse group lasso proposed by [22]. It tries to induce sparsity in coefficients not only groupwise but also within groups, by using a modified formulation of group lasso in (14.4):

$$\min_{\boldsymbol{\beta}, \beta_0} f(\boldsymbol{\beta}, \beta_0) + \lambda \{\alpha \|\boldsymbol{\beta}\|_1 + (1 - \alpha) \Psi_G(\boldsymbol{\beta})\}. \quad (14.9)$$

Here  $\alpha \in [0, 1]$  is a parameter that determines the mixing of the  $\ell_1$  and the group ( $\Psi_G$ ,  $\ell_1/\ell_2$ ) norms, similarly to a constant in elastic net (14.3). Overlapping group lasso discussed in Sect. 14.2.2 can be extended in a similar fashion, by augmenting its objective function with an  $\ell_1$  term.

#### 14.2.3.1 Computation of Solutions

Similarly to group lasso, sparse group lasso in (14.9) defines a convex minimization problem for which sufficient and necessary conditions can be characterized by optimality conditions.

To understand the optimality conditions, we need the notion of *subgradient*. For a function  $h(\boldsymbol{\beta})$ ,  $\mathbf{g} \in \mathbb{R}^p$  is a subgradient for  $h$  at  $\boldsymbol{\beta} \in \mathbb{R}^p$  if

$$h(\boldsymbol{\beta}') \geq h(\boldsymbol{\beta}) + \mathbf{g}^T (\boldsymbol{\beta}' - \boldsymbol{\beta}), \quad \forall \boldsymbol{\beta}' \in \mathbb{R}^p.$$

If  $h$  is a convex function on  $\mathbb{R}^p$ , it has a subgradient at every point in  $\mathbb{R}^p$ . For example,  $h(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_1$  is a convex function and its subgradient  $\mathbf{g}$  at  $\boldsymbol{\beta}$  is defined by

$$g_j \in \begin{cases} \{+1\} & \text{if } \beta_j > 0, \\ \{-1\} & \text{if } \beta_j < 0, \\ [-1, +1] & \text{if } \beta_j = 0. \end{cases}$$

Note that the expression on the right defines a set of subgradients, not a single subgradient, since when  $\beta_j = 0$  we can select any number in  $[-1, +1]$  for the  $j$ th element of  $\mathbf{g}$ . A set of subgradients is referred to as a *subdifferential*.

For simple notation, we define a vector-valued *sgn* function  $\text{sgn}: \mathbb{R}^p \rightarrow \{-1, 0, +1\}^p$  so that the  $j$ th element of  $\text{sgn}(\boldsymbol{\beta})$  is  $+1$  if  $\beta_j > 0$ ,  $-1$  if  $\beta_j < 0$ , and  $0$  otherwise. We also define a vector-valued *soft threshold* function  $\text{soft}(\mathbf{v}, t)$  for a vector  $\mathbf{v}$  and a scalar  $t$ , so that  $(\text{soft}(\mathbf{v}, t))_j = \text{sgn}(\mathbf{v}_j) \max\{0, |\mathbf{v}_j| - t\}$ .

The next theorem summarizes the optimality conditions for sparse group lasso. The theorem is from [22], however, our proof is much simpler and provides clearer arguments regarding optimality conditions.

**Theorem 2** For a sparse group lasso problem (14.9) defined with  $f(\boldsymbol{\beta}) = \frac{1}{2} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2$  in (14.5) and  $\mathbf{X}_{G_k}^T \mathbf{X}_{G_k} = \mathbf{I}_{|G_k|}$  for  $k = 1, 2, \dots, K$ , an optimal solution  $\boldsymbol{\beta}$  is given by

$$\boldsymbol{\beta}_{G_k} = \left( 1 - \frac{\lambda}{\|\text{soft}(\mathbf{s}_k, \lambda\alpha)\|_2} \right)_+ \text{soft}(\mathbf{s}_k, \lambda\alpha), \quad k = 1, 2, \dots, K,$$

where  $\mathbf{s}_k := \mathbf{X}_{G_k}^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}_{-G_k})$ ,  $\boldsymbol{\beta}_{-G_k}$  is a vector with the same elements as  $\boldsymbol{\beta}$  except for having zeros in the components at  $j \in G_k$ , and  $(\cdot)_+ := \max(\mathbf{0}, \cdot)$ .

*Proof* For  $\boldsymbol{\beta}_{G_k} \neq \mathbf{0}$ , the objective function of (14.9) is differentiable, and therefore the first-order optimality condition can be written using  $\mathbf{X}_{G_k}^T \mathbf{X}_{G_k} = \mathbf{I}_{|G_k|}$  and  $\mathbf{s}_k = \mathbf{X}_{G_k}^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}_{-G_k})$ ,

$$-\mathbf{s}_k + \lambda\alpha \mathbf{g}_{G_k} + \left( \frac{\lambda(1-\alpha)}{\|\boldsymbol{\beta}_{G_k}\|_2} + 1 \right) \boldsymbol{\beta}_{G_k} = \mathbf{0}. \quad (14.10)$$

Here  $\mathbf{g}_{G_k}$  is a subgradient of  $\|\boldsymbol{\beta}\|_1$  for the components in  $G_k$ .

Suppose that  $\boldsymbol{\beta}_{G_k} = \mathbf{0}$  is optimal. Then the gradients of the sparse group lasso objective in (14.9) evaluated at two nonzero points,  $\boldsymbol{\beta} + \boldsymbol{\eta}$  and  $\boldsymbol{\beta} - \boldsymbol{\eta}$ , where  $\boldsymbol{\eta} := (0, \dots, 0, \varepsilon, \dots, \varepsilon, 0, \dots, 0)^T$  is a  $p$  dimensional vector with  $\varepsilon > 0$  at components  $j \in G_k$ , must satisfy

$$\begin{cases} -\mathbf{s}_k + \lambda\alpha \mathbf{g}_{G_k} + \left( \frac{\lambda(1-\alpha)}{\varepsilon\sqrt{|G_k|}} + 1 \right) \boldsymbol{\eta}_{G_k} > 0, \\ -\mathbf{s}_k + \lambda\alpha \mathbf{g}_{G_k} - \left( \frac{\lambda(1-\alpha)}{\varepsilon\sqrt{|G_k|}} + 1 \right) \boldsymbol{\eta}_{G_k} < 0. \end{cases}$$

Tending  $\varepsilon \rightarrow 0$ , we obtain  $|(\mathbf{s}_k + \lambda\alpha \mathbf{g}_{G_k})_j| \leq \lambda(1-\alpha)/\sqrt{|G_k|}$  for  $j = 1, \dots, |G_k|$ . This implies that together with the property of  $\mathbf{g}_{G_k}$  at  $\boldsymbol{\beta}_{G_k} = \mathbf{0}$ ,

$$\|\text{soft}(\mathbf{s}_k, \lambda\alpha)\|_2 \leq \|\mathbf{s}_k + \lambda\alpha \mathbf{g}_{G_k}\|_2 \leq \lambda(1-\alpha). \quad (14.11)$$

On the other hand, suppose that  $\beta_{G_k} \neq \mathbf{0}$  is optimal. From (14.10), we get

$$\mathbf{s}_k - \lambda\alpha \mathbf{g}_{G_k} = \left( \frac{\lambda(1-\alpha)}{\|\boldsymbol{\beta}_{G_k}\|_2} + 1 \right) \boldsymbol{\beta}_{G_k}.$$

Here, we note that  $(\boldsymbol{\beta}_{G_k})_j = 0$  is optimal whenever  $|(\mathbf{s}_k)_j| \leq \lambda\alpha$  since it satisfies an equality in the multiple equations above. Therefore we have

$$\text{soft}(\mathbf{s}_k, \lambda\alpha) = \left( \frac{\lambda(1-\alpha)}{\|\boldsymbol{\beta}_{G_k}\|_2} + 1 \right) \boldsymbol{\beta}_{G_k}.$$

This gives that  $\|\boldsymbol{\beta}_{G_k}\|_2 = \|\text{soft}(\mathbf{s}_k, \lambda\alpha)\|_2 - \lambda(1-\alpha)$ . Replacing  $\|\boldsymbol{\beta}_{G_k}\|_2$  above and rearranging terms, we get

$$\boldsymbol{\beta}_{G_k} = \left( 1 - \frac{\lambda(1-\alpha)}{\|\text{soft}(\mathbf{s}_k, \lambda\alpha)\|_2} \right) \text{soft}(\mathbf{s}_k, \lambda\alpha).$$

Combining with the condition in (14.11), we obtain the claim.  $\square$

Theorem 2 tells that there are two possibilities for a component  $\beta_j$ ,  $j \in G_k$ , to have the zero value:

- $\|\text{soft}(\mathbf{s}_k, \lambda\alpha)\| \leq \lambda$ : in this case, the  $(\cdot)_+$  part of the expression of  $\boldsymbol{\beta}_{G_k}$  in Theorem 2 becomes zero, and therefore the entire subvector  $\boldsymbol{\beta}_{G_k}$  (including  $\beta_j$ ) becomes the zero vector. If this is the case, the group  $G_k$  is not selected,
- $\|\text{soft}(\mathbf{s}_k, \lambda\alpha)\| > \lambda$  and  $|(\mathbf{s}_k)_j| \leq \lambda$ : in this case the group  $G_k$  is selected, but the component  $j \in G_k$  is not selected because  $(\text{soft}(\mathbf{s}_k, \lambda\alpha))_j = 0$ , and therefore  $\beta_j = 0$ .

Comparing to Theorem 1 for group lasso in Sect. 14.2.1, group lasso has a similar property to the first one above for groupwise selection, but lacks the second property for within group feature selection.

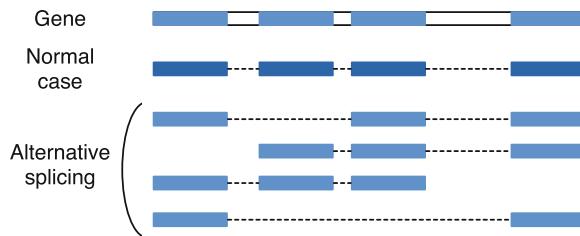
## 14.3 A Case Study on Exon Microarray Data

As a case study of selecting grouped features, we consider identifying alternative splicing of genes from high-throughput genomic data. We use a specific type of data acquired by using the GeneChip Human Exon 1.0 ST Arrays from Affymetrix,<sup>2</sup> a popular platform for profiling gene expression of entire human genome in exon level.

For background information, a gene is a sequence in DNA (deoxyribonucleic acid) composed of four letters A, T, G, and C, and it has coding regions (called exons) and non-coding regions (called introns). After its sequence is copied into a messenger RNA (ribonucleic acid), only coding regions are combined together and later used to

---

<sup>2</sup> <http://www.affymetrix.com>



**Fig. 14.1** An example of alternative splicing. A gene consists of coding regions (*dark exons*) and non-coding regions (*white introns*). Exons are combined together by splicing mechanism to create a normal transcript. In case of alternative splicing, transcripts are created with different use of exons

create functional units, proteins. This process of collecting exons is called *splicing* in genomics.

*Alternative splicing* refers to the events of using exons differently than normal cases in splicing. Figure 14.1 shows a pictorial example of alternative splicing. Classical gene microarrays have probes to measure the amount of transcripts (mRNAs) created as a result of splicing, and therefore cannot capture alternative splicing events. Exon microarrays can capture alternative splicing. Different usage of exons can be caused by several reasons including mutations in exon sequences, and it has implications in development of cancers, for example.

Exon microarrays measure the expression level of individual exons, and these are grouped as genes as we can see in Fig. 14.1. So it would make sense to select all exons that belong to the same gene when they are relevant. But one may also want to assign different weights on exons, to detect alternative splicing events if any exists. Selecting individual exons without considering their grouping as genes also remains as an option, but it is more likely to overfit given data when  $p$  is large since it is not constrained by group information.

### 14.3.1 Data

A combination of two microarray data sets available at Gene Expression Omnibus (GEO),<sup>3</sup> with accession numbers GSE21713 and GSE32664 [9, 19], have been used in our case study. These contain total 113 microarrays from neuroblastoma patients. Neuroblastoma is one of the most common solid cancer in children who are usually younger than two years. Both GEO data sets have been obtained using the same microarray platform, the Affymetrix Human Exon v1.0 ST arrays.

As up to four probes are used to measure the expression of a single exon, raw measurements in microarrays have to be summarized and normalized. For these we apply the frozen RMA (fRMA) algorithm by [14], which processes individual microarrays using information from predefined global reference arrays. Low quality

<sup>3</sup> <http://www.ncbi.nlm.nih.gov/geo>

arrays with the median GNSE error scores [15] larger than a threshold value of 1.2 have been discarded, resulting in total 92 microarrays for analysis.

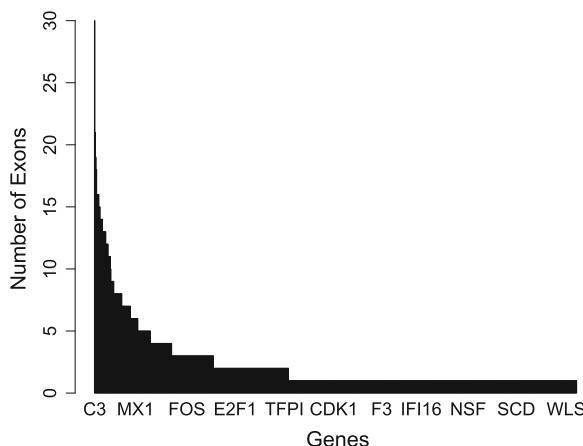
After applying the fRMA algorithm, we choose the “core” exons for which we have the best evidence being a coding region, with unique hybridization, unique localization on human chromosomes, and genes assigned, according to the NetAffx probeset annotation v33.1.<sup>4</sup> This resulted in total 228476 features. Finally, the top 2000 exon features with largest standard deviation values have been chosen for analysis.

To summarize, our data set consist of  $n = 92$  patients with  $p = 2000$  features corresponding to exons. Each patient has a tumor stage out of five stages (1, 2, 3, 4, and 4s) assigned. We categorize the stages into low risk ( $y^i = -1$ , stages 1, 2, and 4s) and high risk ( $y^i = +1$ , stages 3 and 4), so to create a binary classification task. The ratio of the two categories is about 50:50.

In this data set, the 2000 exon features are grouped into  $K = 845$  genes. Figure 14.2 shows the sizes of groups, that is, the number of exons (y-axis) in each gene (a few gene names are on the x-axis). About 88 % of genes consist of 1–4 exons, whereas a gene C8 has the maximal size (30 exons).

### 14.3.2 Algorithms for Comparison

The following three algorithms are to be compared, with the loss function for logistic regression, in order to identify features that are important for classifying high and low risk categories.



**Fig. 14.2** The number of exons in genes. As genes consist of exons, this shows the sizes of groups. Most of the groups have 1–4 features

<sup>4</sup> <http://www.affymetrix.com/analysis/downloads/na33/>

## LASSO

The regularized regression discussed in Sect. 14.1.1 is used to select individual features, with the lasso regularizer. The `glmnet` R package [10] was used for our experiments.

## GL

The group lasso algorithm discussed in Sect. 14.2.1 is used to select grouped features. For solving group lasso problems, the `grplasso` [16] or the `SGL` [22] R packages can be used. The latter is designed for sparse group lasso, but it can solve group lasso problems by specifying the parameter  $\alpha = 0$  so that the sparse group lasso formulation in (14.9) will be optimized without an  $\ell_1$  term. We used the `SGL` package for our experiments.

## SGL

The sparse group lasso discussed in Sect. 14.2.3 is used to perform both groupwise and within-group individual feature selection. For analysis we use the `SGL` package. Note that the parameter  $\alpha$  in (14.9) can be chosen to solve the lasso problem by setting  $\alpha = 1$ , or the group lasso problem by  $\alpha = 0$ . An optimal value of  $\alpha$  can be determined for instance by cross validation, searching on a two dimensional grid for both  $\lambda$  and  $\alpha$ . For the purpose of demonstration, we used a fixed value  $\alpha = 0.95$ .

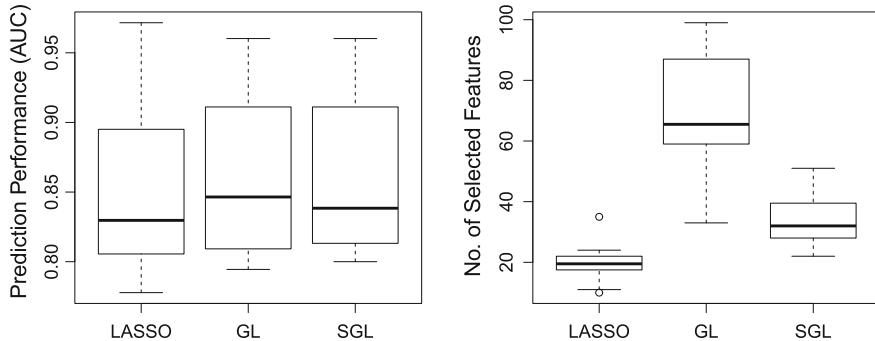
### 14.3.3 Comparison of Performance

#### 14.3.3.1 Prediction Performance

Since the entire data set is rather small ( $n = 92$ ), instead of dividing the set into a training and a test set once, we performed random subsampling: we repeated the process of choosing 70 % of random patient indices (without replacement) for training and taking the rest for testing. For each trial, we measured the prediction performance on a test set of the predictor obtained with a training set.

Figure 14.3 shows the AUC (area under the curve) [11, 20] scores from 20 random subsampling trials. The AUC score (left panel) is improved by performing grouped selection (GL and SGL), compared to the individual selection (LASSO). However, grouped selection resulted in choosing larger number of features than LASSO (right panel). Less number of features were chosen by SGL compared to GL as expected, but with sacrificing a small portion of prediction performance.

In fact, the number of selected features is closely related to the cost of clinical tests built upon the chosen features. All numbers were relatively small (<100) in our case, however some would prefer a smaller number of features to reduce cost if degradation in prediction would not be significant. In this regard, SGL in Fig. 14.3 seemed to provide a good compromise between the number of features and prediction performance.



**Fig. 14.3** Overall prediction performance of three feature selection methods: LASSO, GL, and SGL. *Left* prediction performance in AUC score on test sets over 20 random subsampling trials (train:test = 70:30 %.) *Right* the corresponding number of selected features

#### 14.3.3.2 Probabilistic Prediction

In logistic regression, the probability that an example  $\mathbf{x}^i$  will have the label “1” is modeled by the logistic function

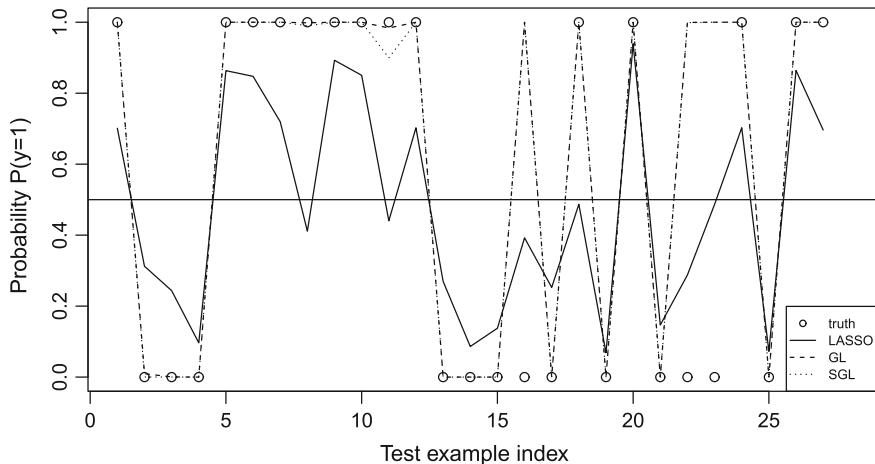
$$P(y^i = 1|\mathbf{x}^i) = \frac{1}{1 + \exp\{-(\boldsymbol{\beta}^T \mathbf{x}^i + \beta_0)\}} \in [0, 1],$$

where  $\boldsymbol{\beta}$  and  $\beta_0$  are coefficients estimated during training. Note that this function always returns a value between zero and one. That is, given  $\boldsymbol{\beta}$  and  $\beta_0$ , logistic regression provides each test point with a probabilistic outcome in addition to a binary prediction. This makes a clear distinction to other classification methods such as the support vector machines [3, 21].

For two logistic regression classifiers with similar binary prediction performance (for example, in terms of AUC scores), a method that gives higher probability for correct predictions would be arguably preferred in practice, since it provides higher confidence on its predictions.

Figure 14.4 compares such probability values for the three feature selection methods LASSO, GL, and SGL. The x-axis shows the indices of test examples (in a test set created by random subsampling), while the y-axis shows the probability values we discussed above. The circles show the true labels, 0 or 1. The probability outcomes from each algorithm are connected by lines only for visual distinction, without any other implication. The decision probability ( $P(y = 1) = 0.5$ ) is shown as a horizontal line.

As we can see, GL and SGL provided higher values of probability outcomes for correct labels ( $P(y = 1)$  or  $P(y = 0) = 1 - P(y = 1)$ ), at least for this particular test set. The characteristics of GL and SGL were similar: on 11th example GL provided slightly higher probability than SGL, and both misclassified 16th, 22nd and 23rd examples that were classified correctly by LASSO.



**Fig. 14.4** Probabilistic outcomes of logistic regression classifiers for a single random test set. The outcomes of LASSO, GL and SGL are compared to the true labels

#### 14.3.3.3 Selected Features

Another important factor of feature selection in this case study is how the three methods (LASSO, GL, and SGL) provide insights on our research question about alternative splicing.

Table 14.1 summarizes the features selected by the three methods in a single random subsampling trial. Here the PID represents a 7-digit unique number assigned by Affymetrix, to a set of probes (also known as probesets) on microarrays to detect exons. As we consider core probesets with unique hybridization, each probeset has a correspondence to a unique exon, a coding subsequence of a gene on DNA. A PID also maps to a feature in our discussion. The “Coef” refers to the value in the coefficient vector  $\beta$  corresponding to each chosen feature: the larger its magnitude is, the more contribution is made by a feature to prediction.

Although some genes (DDR2 and HIST1H1A) were detected in all of the three feature sets, the results were quite different. First, the exons chosen by LASSO were all different except for the gene EPB41L3. As the genetic relations of exons are essentially ignored and any exon with high correlation to labels can be selected by LASSO, it is hard to say if an exon was chosen by a possible change in a single exon or by a new combination of exons (alternative splicing). In fact, the coefficient magnitude values were small except for a single exon from the gene IPW. Since this gene is non-functional (non-protein coding), it is likely that correlations in measurement errors have been captured rather than the true signals in this case. On the other hand, several exons were chosen from the same genes (DDR2 and HIST1H1A) with relatively large coefficient values in case of GL and SGL (a single exon was chosen from HIST1H1A in case of SGL). They could be indicatives of possible alternative

**Table 14.1** Features chosen by LASSO, GL, and SGL methods in a single random subsampling trial

PID	Coef	Gene	Description
<i>LASSO</i>			
2343490	0.10	IFI44L	Interferon-induced protein 44-like
2364269	0.08	DDR2	Discoidin domain receptor tyrosine kinase 2
2417277	0.09	GNG12	Guanine nucleotide binding protein (G protein), gamma 12
2592622	-0.05	TMEFF2	Transmembrane protein with EGF-like and 2 follistatin-like domains 2
2946198	-0.10	HIST1H1A	Histone cluster 1, H1a
3039339	-0.04	DGKB	Diacylglycerol kinase, beta 90kDa
3544537	-0.03	FOS	FBJ murine osteosarcoma viral oncogene homolog
3584453	0.23	IPW	Imprinted in Prader-Willi syndrome (non-protein coding)
3797070	-0.07	EPB41L3	Erythrocyte membrane protein band 4.1-like 3
3797105	-0.10	EPB41L3	Erythrocyte membrane protein band 4.1-like 3
<i>GL</i>			
2364258	1.09	DDR2	Discoidin domain receptor tyrosine kinase 2
2364269	1.04	DDR2	Discoidin domain receptor tyrosine kinase 2
2364272	0.97	DDR2	Discoidin domain receptor tyrosine kinase 2
2413913	0.62	DHCR24	24-dehydrocholesterol reductase
2923918	-0.63	PKIB	Protein kinase (cAMP-dependent, catalytic) inhibitor beta
2946197	-1.01	HIST1H1A	Histone cluster 1, H1a
2946198	-0.68	HIST1H1A	Histone cluster 1, H1a
3584107	6.55	MKRN3	Makorin ring finger protein 3
3618343	-0.90	MEIS2	Meis homeobox 2
3653293	-1.27	CACNG3	Calcium channel, voltage-dependent, gamma subunit 3
<i>SGL</i>			
2343489	1.29	IFI44L	Interferon-induced protein 44-like
2364258	1.23	DDR2	Discoidin domain receptor tyrosine kinase 2
2364269	0.53	DDR2	Discoidin domain receptor tyrosine kinase 2
2413913	1.68	DHCR24	24-dehydrocholesterol reductase
2946197	-2.44	HIST1H1A	Histone cluster 1, H1a
3584107	5.19	MKRN3	Makorin ring finger protein 3
3797060	-1.50	EPB41L3	Erythrocyte membrane protein band 4.1-like 3
3797104	-1.50	EPB41L3	Erythrocyte membrane protein band 4.1-like 3
3797105	-0.48	EPB41L3	Erythrocyte membrane protein band 4.1-like 3
4026080	0.60	GABRA3	Gamma-aminobutyric acid (GABA) A receptor, alpha 3

splicing events, although following-up biochemical experiments would be required for validation.

## 14.4 Discussion

The use of the lasso regularization (Sect. 14.1.1) also appears in many different fields, for instance in compressed sensing [4, 5, 8] which solves an optimization problem of the form of (14.1) with least squares loss function and an  $\ell_1$  regularizer. In compressed sensing, a signal of length  $p$  is recovered from few observations (small  $n$ ), under the assumption that the original signal is sparse. Exact recovery with a high probability is guaranteed under certain conditions.

For group lasso (Sect. 14.2.1) with the ordinary least squares loss function, there are alternative methods to group lasso, including group LARS (Least Angle Regression Selection) and group non-negative garotte [24]. These methods have slightly different characteristics on their solution path. Also, group LARS usually scales much better than group lasso.

In Sect. 14.2.2, we have introduced a naive approach to reformulate overlapping group lasso to group lasso, by replicating features that belong to multiple groups. However, this approach increases the dimension of optimization, and therefore may not be preferable when  $p$  is large. There exist several optimization algorithms that do not require such replication [13, 25, 28].

When the dimension in data is much larger than the size of a sample ( $p \gg n$ ), the solution of the optimization problem in (14.1) can vary even by small changes in the sample. Denoting an estimate by  $\hat{\beta}^n$  which we obtain by solving (14.1) with a sample of size  $n$ , and denoting a true unknown parameter by  $\beta^*$ , we can define the notion of *consistency* in terms of variable selection,

$$P\left(\{j : \hat{\beta}_j^n \neq 0\} = \{j : \beta_j^* \neq 0\}\right) \rightarrow 1, \text{ as } n \rightarrow \infty.$$

When a method is consistent in terms of variable selection and the convergence above is fast enough, then small  $n$  may not matter much as an estimate  $\hat{\beta}^n$  will be close to  $\beta^*$ .

Lasso produces consistent estimates when some strong conditions hold [17, 27]. Unfortunately features from high-throughput genomic profiling are typically highly correlated, for which these conditions often break. Reference [2] has shown that under a fixed  $p$  and a specific choices of  $\lambda$ , the intersection of features selected by bootstrapped lasso estimates is consistent under less restrictive conditions. Reference [18] has proposed the randomized lasso method, which potentially has better consistency. This issue has been studied in bioinformatics in terms of stable feature selection [1, 7]. With the development of high-throughput profiling technologies, the dimension in data keeps growing. Therefore consistency remains as a challenging topic for research.

## 14.5 Conclusion

The rapid growth of dimensionality in modern high-throughput measurement technology requires us to consider extra information on features, in order to avoid averse effects of high dimensionality such as overfitting. Information on groupings

of features are rather easy to acquire in many applications, and therefore can provide a promising way to cope with high dimensionality.

Grouped feature selection based on regularized regression provides an intuitive way to incorporate grouping information of features into feature selection, in a statistically sound and computationally efficient way. The regularizers in these methods are also flexible to change, making it possible to adapt for future applications that would come with different structures of groups such as hierarchies.

A case study of feature selection on exon microarray data has illustrated that grouped feature selection would provide not only better prediction performance, but also potentially new understanding of complex biological systems. Consistency is still an open problem to solve when sample sizes are much smaller than the dimensionality in data, especially for biomedical applications of grouped feature selection methods.

**Acknowledgments** This work has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 “Providing Information by Resource-Constrained Analysis”, project C1.

## References

1. Abeel, T., Helleputte, T., Van de Peer, Y., Dupont, P., Saeys, Y.: Robust biomarker identification for cancer diagnosis with ensemble feature selection methods. *Bioinformatics* **26**(3), 392–398 (2010)
2. Bach, F.R.: Bolasso: model consistent Lasso estimation through the bootstrap. In: The 25th International Conference on Machine Learning, pp. 33–40 (2008)
3. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pp. 144–152 (1992)
4. Candés, E.J., Tao, T.: Decoding by linear programming. *IEEE Trans. Inf. Theory* **51**(12), 4203–4215 (2005)
5. Candés, E.J., Romberg, J., Tao, T.: Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. Inf. Theory* **52**(2), 489–509 (2006)
6. Cox, D.R.: Regression models and life-tables. *J. R. Stat. Soc. Ser. B Stat. Methodol.* **34**(2), 187–220 (1972)
7. Davis, C.A., Gerick, F., Hintermair, V., Friedel, C.C., Fundel, K., Küffner, R., Zimmer, R.: Reliable gene signatures for microarray classification: assessment of stability and performance. *Bioinformatics* **22**(19), 2356–2363 (2006)
8. Donoho, D.L.: Compressed sensing. *IEEE Trans. Inf. Theory* **52**(4), 1289–1306 (2006)
9. Eschenburg, G., Eggert, A., Schramm, A., Lode, H.N., Hundsdorfer, P.: Smac mimetic LBW242 sensitizes XIAP-overexpressing neuroblastoma cells for TNF- $\alpha$ -independent apoptosis. *Cancer Res.* **72**(10), 2645–2656 (2012)
10. Friedman, J., Hastie, T., Tibshirani, R.: Regularization paths for generalized linear models via coordinate descent. *J. Stat. Softw.* **33**(1), 1–22 (2010)
11. Hastie, T., Tibshirani, R., Friedman, J.H.: *The Elements of Statistical Learning*, 2nd ed. 2009. corr. 10th printing 2013 edn. Springer (2009)
12. Jacob, L., Obozinski, G., Vert, J.P.: Group Lasso with overlap and graph Lasso. In: Proceedings of the 26th International Conference on Machine Learning, pp. 433–440. Montreal, Quebec, (2009)

13. Mairal, J., Jenatton, R., Obozinski, G., Bach, F.: Network flow algorithms for structured sparsity. In: Advances in Neural Information Processing Systems, vol. 23, pp. 1558–1566. MIT Press (2010)
14. McCall, M.N., Bolstad, B.M., Irizarry, R.A.: Frozen robust multiarray analysis (fRMA). *Bio-statistics* **11**(2), 242–253 (2010)
15. McCall, M., Murakami, P., Lukk, M., Huber, W., Irizarry, R.: Assessing affymetrix GeneChip microarray quality. *BMC Bioinform.* **12**(1), 137 (2011)
16. Meier, L., van de Geer, S., Bühlmann, P.: The group Lasso for logistic regression. *J. R. Stat. Soc. Ser. B* **70**, 53–71 (2008)
17. Meinshausen, N., Bühlmann, P.: High-dimensional graphs and variable selection with the Lasso. *Ann. Stat.* **34**, 1436–1462 (2006)
18. Meinshausen, N., Bühlmann, P.: Stability selection. *J. R. Stat. Soc. Ser. B* **72**(4), 417–473 (2010)
19. Mestdagh, P., Boström, A.K., Impens, F., Fredlund, E., Peer, G.V., Antonellis, P.D., von Stedingk, K., Ghesquière, B., Schulte, S., Dews, M., Thomas-Tikhonenko, A., Schulte, J.H., Zollo, M., Schramm, A., Gevaert, K., Axelson, H., Speleman, F., Vandesompele, J.: The miR-17-92 microRNA cluster regulates multiple components of the TGF- $\beta$  pathway in neuroblastoma. *Mol. Cell* **40**(5), 762–773 (2010)
20. Robin, X., Turck, N., Hainard, A., Tiberti, N., Lisacek, F., Sanchez, J.C., Müller, M.: pROC: an open-source package for r and s+ to analyze and compare ROC curves. *BMC Bioinform.* **12**(1), 77 (2011)
21. Schölkopf, B., Smola, A.J.: Learning with Kernels. MIT Press, Cambridge (2002)
22. Simon, N., Friedman, J., Hastie, T., Tibshirani, R.: A sparse-group Lasso. *J. Comput. Graph. Stat.* **22**(2), 231–245 (2013)
23. Tibshirani, R.: Regression shrinkage and selection via the Lasso. *J. R. Stat. Soc. Ser. B* **58**, 267–288 (1996)
24. Yuan, M., Lin, Y.: Model selection and estimation in regression with grouped variables. *J. R. Stat. Soc. Ser. B* **68**, 49–67 (2006)
25. Yuan, M., Liu, J., Ye, J.: Efficient methods for overlapping group Lasso. In: Advances in Neural Information Processing Systems, vol. 24, pp. 352–360. MIT Press (2011)
26. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. *J. R. Stat. Soc. B* **67**, 301–320 (2005)
27. Zhao, P., Yu, B.: On model selection consistency of Lasso. *J. Mach. Learn. Res.* **7**, 2541–2563 (2006)
28. Zhao, P., Rocha, G., Yu, B.: The composite absolute penalties family for grouped and hierarchical variable selection. *Ann. Stat.* **37**(6A), 3468–3497 (2009)

# Index

## A

Absolute approximation region, 98  
Absolute certainty gain, 99  
Absolute dependency gain, 100  
Action rule, 177, 180, 186  
Action term, 179–181, 184  
Affine distortion, 277  
Aggregate, 234, 279  
Aggregation, 263, 281, 285, 290  
All-relevant feature selection, 12, 15  
ANN, 29, 33, 35, 38, 71, 75, 78, 83, 238, 284  
Anti-hub, 239, 244  
Approximate rule, 36, 72, 76, 79  
Approximation region, 97  
Approximation space, 95, 99, 101, 103  
Apriori algorithm, 185  
ARED, 178  
Association rule, 49, 178, 200, 204, 221  
Association rule mining, 207, 215  
Associative classification, 200  
Authorship attribution, 29, 34, 38, 71, 77  
Average dependency gain, 100

## B

Backpropagation, 35, 75  
Backward selection, 30, 36, 71, 74, 83, 203  
Balanced, 60, 72, 216, 247  
Bayes, 14, 51, 54, 93, 99, 105, 157, 232, 241, 244, 284  
Benchmark, 50, 158, 222, 305, 318, 321  
BGTree algorithm, 284  
Bias, 17, 30, 32, 47, 87, 216, 334  
Binary, 12, 286  
Binomial distribution, 65  
Biometrical feature, 267  
Block, 164, 167, 273, 275

Boolean, 113, 124, 126, 134, 136, 154, 216  
Bootstrap, 18, 347  
Boruta, 12, 16–18, 20, 21, 26  
Boundary, 45, 51, 97, 113, 118, 129, 146, 151, 215  
BVQ, 54, 55

## C

Causal feature selection, 49  
Certain rule, 36, 72, 76, 78, 169, 173  
Chromatic space, 269  
Chromosome, 320  
Classification knowledge, 95  
Classification system, 32, 71, 88, 171  
Cluster, 23, 172, 190, 246, 278, 280  
Co-occurrence, 93, 273  
Code vector, 54  
Colour distribution, 266  
Colour histogram, 269  
Combination feature, 19  
Complex, 167  
Computational complexity, 47, 76  
Concept, 165, 168  
Confidence, 177, 180, 186, 188, 189, 192, 202, 285  
Consistency, 32, 74, 93, 168, 173, 347  
Content specific, 31  
Contrast enhancement, 266  
Contrast variable, 15  
Converging, 72, 75, 88  
Convex, 329–331  
Core, 76, 95, 109, 121  
Corpus, 77  
Correlation, 16, 177, 179, 183, 203, 217, 274, 320, 345  
Coverage, 199, 214, 215, 218, 255

Covering, 165, 167, 188  
 Crossover, 323  
 Cross-validation, 32, 57, 173, 250, 253, 285,  
   343  
 CRSA, 35  
 Curse of dimensionality, 46, 47, 58, 240

**D**

Data-driven, 200  
 Data explosion, 49  
 Data mining, 164  
 DBFE, 51  
 Decision algorithm, 35, 72  
 Decision border, 51  
 Decision system, 179, 268  
 Decision table, 102, 115  
 Decision tree, 16, 30, 33, 47, 268, 284  
 Definable set, 99, 165  
 Dependency, 93, 94, 99, 104, 108, 115, 246,  
   296, 320  
 Deterministic, 19, 102  
 Diagonal, 50, 237  
 Dimensionality, 31, 33, 36, 72, 76, 83, 238,  
   277  
 Discrete, 51, 60, 94, 273, 307, 311  
 Discretization, 94, 101, 172, 215  
 Dissimilarity, 278, 280, 290, 293  
 Distance, 231, 232, 234, 237, 244, 253, 265  
 Distribution, 12, 31, 104, 239, 250  
 Domain, 30, 31, 49, 71, 101, 179, 180, 189,  
   232  
 Dominance, 35, 61, 75  
 DRSA, 29, 33, 35, 38, 71, 75, 78, 115, 144  
 Dual, 157, 313  
 Dummy variable, 330, 335  
 Dynamic programming, 235, 236  
 Dynamic time warping, 232, 234, 253

**E**

EDBFM, 45  
 Edge histogram descriptor, 275  
 Edit distance, 235  
 Eigenvalue, 48, 50, 52  
 Eigenvector, 50, 52  
 Elastic net, 332  
 Elementary set, 95, 102, 165  
 Elongation, 235  
 Embedded, 29, 33, 36, 38, 40, 48, 200, 204,  
   330  
 Ensemble, 16, 17, 285  
 Entropy, 32, 48, 74, 157  
 Environment, 46, 280, 305

Equivalence, 35, 95, 101, 116, 128, 165  
 Error rate, 47, 49, 58, 95, 164, 173  
 Euclidean distance, 58, 278, 280, 320  
 Evaluation measure, 177, 219, 281  
 Exact rule, 36, 76  
 Execution confidence, 177, 189  
 Exhaustive search, 15, 36, 73  
 Expert, 31, 71, 95, 165, 183

**F**

FastAWARD, 255, 256  
 Feature construction, 232, 256  
 Feature extraction, 48, 269, 306  
 Feed-forward, 35, 75  
 Field of view, 264, 268, 285  
 Filter, 29, 32, 38, 48, 57, 79, 275, 320  
 Flat data format, 216  
 Flexible feature, 179, 181, 183, 190  
 Flowgraph, 268  
 Forward selection, 71, 74, 78, 203  
 Frequency, 31, 65, 77, 95, 185, 203, 245, 250,  
   273, 319  
 Fuzzy, 157, 158, 240, 243, 246

**G**

Gain ratio, 48, 61  
 Gamma correction, 266  
 Gaussian, 19, 51, 61, 310  
 Generative feature, 19  
 Genetic algorithm, 232, 305, 319, 320  
 Geometric invariance, 269  
 Geometric moment, 306  
 Geospatial decision support, 49  
 Global, 36, 53, 79, 164, 165, 188, 246  
 Gradient, 55, 277, 284  
 Granule, 35  
 Grouped feature, 329  
 Group  $\ell_1/\ell_2$  norm, 333

**H**

Hahn, 313  
 Hermite, 310  
 Heterogeneity, 46, 60  
 Heuristic, 17, 31, 47, 48, 61, 107, 158, 200  
 Homogenous, 247  
 Hub, 232, 238, 239  
 Hyperclique, 202  
 Hyper-sphere, 246

**I**

Image classification, 268

Image descriptor, 263  
 Image reconstruction, 314  
 Imbalance, 77, 239, 247, 253  
 Imputation, 171, 214  
 Incomplete, 42, 113, 171  
 Inconsistency, 75, 113, 168, 173  
 Incremental usefulness, 74  
 Indiscernibility, 35, 75, 95, 100, 113, 164, 165  
 Inference, 244  
 Influence matrix, 181, 182, 185, 188  
 Information gain, 32, 74, 203  
 Information system, 14, 49, 179, 183, 186  
 Information theory, 30, 74  
 INSIGHT, 255  
 Instance selection, 232, 254  
 Instance space, 47  
 Interestingness measure, 200  
 Intrinsic, 47, 52, 238  
 Invariant, 267, 269, 272, 277, 306  
 Irrelevant, 14, 74, 165, 169, 200, 205, 217

**J**  
 Jaccard measure, 200

**K**  
 Kernel, 61, 315  
 KNN, 232, 233, 284, 323  
 Knowledge discovery, 30, 78  
 Knowledge representation, 102  
 Krawtchouk, 312

**L**  
 Lasso, 329, 330, 332  
 Legendre, 309  
 LEM1, 163, 166  
 LEM2, 163, 166  
 LERS, 163, 166, 171  
 Lexical, 30, 34, 77  
 Likelihood, 97, 177, 185, 190  
 Linear discriminant analysis, 50  
 Local, 30, 36, 79, 164, 167, 246  
 Lower approximation, 96, 113, 167  
 Low level algorithm, 264  
 LVQ, 54

**M**  
 Machine Learning Repository, 171  
 Mapping, 50, 51, 101, 102, 207, 237  
 Markov model, 232, 268

Matching, 36, 76, 108, 171, 236, 238, 266, 281  
 Matrix, 50, 117, 125, 235–237, 266, 334  
 Median, 38, 41, 83, 271  
 Membership, 97, 113  
 Merging, 173  
 Meta-action, 177, 181  
 Meta-action extraction, 178, 190  
 Minimal cover, 76, 78, 79, 84  
 Missing value, 61, 157, 171, 214  
 MLP, 35, 75, 284  
 Moment descriptor, 305  
 Monotonic, 94, 106, 114, 129, 202, 259  
 Multi-camera tracking, 263  
 Multi-starting, 35, 75, 83  
 Multivariate, 57, 60  
 Music retrieval, 178, 238  
 Mutation, 323

**N**  
 Nearest neighbor, 55, 232, 238  
 Negative region, 97, 105, 118, 129  
 Neural network, 33, 47, 72, 232, 268, 298  
 Noise, 12, 22, 49, 57, 93, 127, 205, 216, 266, 312  
 Nominal, 35, 75, 144  
 Non-deterministic, 102  
 Non-singular approximation, 50  
 Normalization factor, 105, 308  
 Normalized, 58, 105, 106, 172, 186, 247, 271, 273, 307, 318, 341  
 NP-complete, 255  
 NP-hard, 109  
 Null hypothesis, 65  
 Numerical attribute, 158, 172  
 Numerosity reduction, 254

**O**  
 Object identification, 263  
 Object re-identification, 265, 278  
 Object tracking, 267  
 Occurrence, 31, 77, 95, 185, 201, 210, 239, 250  
 Offset, 273  
 OLDA algorithm, 50  
 Ontology, 49, 330  
 Ordinal, 35, 75, 114, 144  
 Orphan, 239  
 Orthogonal, 50, 254, 290, 306, 307  
 Orthonormal, 50, 310, 334  
 Outlier, 186  
 Overfitting, 49, 158, 200, 203, 205, 347

**P**

Particle filter, 268  
 Partition, 51, 54, 100, 164, 165, 184, 185  
 Parzen estimate, 56  
 Pattern recognition, 29, 71, 93, 231, 306  
 Pattern selection, 200  
 Penalty, 330  
 Permutation, 15, 58, 61, 333  
 Pixel, 264, 269, 276, 306  
 Polynomial, 74, 157, 166, 232, 306, 309  
 Population, 95, 179, 189, 323, 329  
 Positive region, 97, 104, 114, 119  
 Possible rule, 36, 76, 79, 169  
 Precedence, 184  
 Preference order, 33, 35, 82  
 Prime implicant, 126  
 Principal component analysis, 232  
 Probabilistic, 93, 94, 101, 114, 344  
 Probability distribution, 15, 203  
 Prototype selection, 254  
 Pruning, 30, 33, 75, 83

**Q**

Quality measure, 29, 33, 76

**R**

R library, 18  
 Random feature, 19  
 Random forest, 12, 13, 15, 17, 18, 232, 285  
 Rank, 12, 34, 38, 47, 57, 255  
 Ranking, 12, 16, 30, 32, 33, 36, 38, 45, 48,  
   50, 71, 79, 255, 282, 283, 290  
 Raster, 266  
 Ratio, 26, 48, 61, 63, 75, 168, 171, 217, 281  
 Reduct, 30, 33, 36, 40, 76, 93, 106, 113, 134,  
   165  
 Redundant, 14, 31, 47, 52, 104, 114, 165,  
   169, 200, 215, 307  
 Regression, 203, 217, 258, 284, 329, 342  
 Regularization, 329  
 Relevance, 12, 14, 15, 31, 32, 36, 46, 48, 72,  
   74, 163, 203, 217, 232, 246, 320  
 Relief algorithm, 32, 36, 38, 49, 61, 305, 320  
 Replacement, 196, 235, 343  
 Resolution, 329  
 Rough approximation, 94  
 Rough membership, 127, 130  
 Rough set, 30, 33, 35, 42, 76, 95, 113, 163,  
   165  
 Rule complexity, 164  
 Rule evaluation, 177, 201  
 Rule induction, 35, 76, 94, 163, 171

**S**

Satisfaction rate, 189  
 Scalability, 46  
 Scatter, 278  
 Score, 32, 33, 182, 241, 255, 342  
 Search space, 36, 78, 164, 166, 200  
 Sensitivity, 15, 16, 21, 23, 47, 330  
 Sequential selection, 36, 71, 74, 78, 83  
 Set approximation, 94  
 Set definability, 96  
 Sharpening, 266  
 Significance factor, 108  
 Signum, 339  
 Skewness, 239, 250, 256  
 Soft computing, 164  
 Space granulation, 307  
 Sparsity, 329  
 Specificity, 171  
 Stability, 279, 282, 292, 311  
 Stable feature, 179, 347  
 Statistical interestingness, 200  
 Statistical significance, 17, 63, 66, 201, 203,  
   250, 252  
 Statistics, 31, 61, 77, 164, 292, 330  
 Strength, 108, 171, 216, 275  
 String encoding, 207  
 Structural, 31, 200, 204  
 Structure, 31, 102, 113, 114, 200  
 Stylistic feature, 30  
 Styliometry, 29, 30, 71, 77  
 Subdifferential, 339  
 Subgradient, 338  
 Supervised, 48, 71, 116, 165, 232  
 Support, 36, 38, 76, 79, 85, 171, 177, 180,  
   185, 202  
 Support vector machine, 52, 232, 284, 344  
 Syntactic, 30, 34, 77

**T**

Tchebichef, 311  
 Text categorisation, 77  
 Text mining, 238  
 Textual descriptor, 77  
 Texture, 266  
 Threshold, 16, 49, 158, 186, 200, 211, 215,  
   216, 276, 320, 332, 339  
 Time complexity, 47, 164, 166  
 Time-series, 232, 240, 250  
 Time window, 268  
 Topology, 35, 38, 75, 78, 264  
 Track, 267  
 Transaction, 179, 183, 202, 208

Transition, 177, 186, 268, 285

Tree instance, 201

Tree-structured data, 200

## U

UCI repository, 19, 26, 60

UCR, 239, 247

Uncertainty, 113, 122, 243

.Undefinable set, 94, 96

Unfeasible, 33, 72, 73

Unidirectional, 35, 75

Unknown instance, 35

Upper approximation, 96, 113, 167

Usefulness, 32, 49, 74, 203, 269

## V

Value reduct, 108

Variable precision rough set (VPRS), 93, 95,

102, 114, 128, 133

Variance, 47, 310

Vector space, 232, 256

Video acquisition, 268

Video surveillance, 263

Visual descriptor, 265, 268

Voronoi region, 54

Vote, 18, 233, 241, 243, 246

Voting, 36, 76, 240, 243, 253, 284

## W

Warping window, 237, 254

Wavelet, 266, 277

Weight, 32, 33, 35, 36, 45, 49, 61, 72, 186, 243, 286, 312

Weighting, 36, 56, 71, 76, 79, 240, 241, 244, 246, 314

Weka, 42, 258

Wrapper, 18, 29, 32, 38, 48, 74, 320

## Z

Zernike, 308