
Aleksander Øhrn

Discernibility and Rough Sets in Medicine: Tools and Applications

Department of Computer and Information Science
Norwegian University of Science and Technology
N-7491 Trondheim, Norway



NTNU Trondheim
Norges teknisk-naturvitenskapelige universitet
Doktor ingeniøravhandling 1999:133

Institutt for datateknikk og informasjonsvitenskap
IDI-rapport 1999:14

ISBN 82-7984-014-1
ISSN 0802-6394

Abstract

This thesis examines how discernibility-based methods can be equipped to possess several qualities that are needed for analyzing tabular medical data, and how these models can be evaluated according to current standard measures used in the health sciences. To this end, tools have been developed that make this possible, and some novel medical applications have been devised in which the tools are put to use.

Rough set theory provides a framework in which discernibility-based methods can be formulated and interpreted, and also forms an appealing foundation for data mining and knowledge discovery. When the medical domain is targeted, several factors become important. This thesis examines some of these factors, and holds them up to the current state-of-the-art in discernibility-based empirical modelling. Bringing together pertinent techniques, suitable adaptations of relevant theory for model construction and assessment are presented. Rough set classifiers are brought together with ROC analysis, and it is outlined how attribute costs and semantics can enter the modelling process.

ROSETTA, a comprehensive software system for conducting data analyses within the framework of rough set theory, has been developed. Under the hypothesis that the accessibility of such tools lowers the threshold for abstract ideas to migrate into concrete realization, this aids in reducing a gap between theoreticians and practitioners, and enables existing problems to be more easily attacked. The ROSETTA system boasts a set of flexible and powerful algorithms, and sets these in a user-friendly environment designed to support all phases of the discernibility-based modelling methodology. Researchers world-wide have already put the system to use in a wide variety of domains.

By and large, discernibility-based data analysis can be varied along two main axes: Which objects in the universe of discourse that we deem it necessary to discern between, and how we define that discernibility among these objects is allowed to take place. Using ROSETTA, this thesis has explored various facets of this also in three novel and distinctly different medical applications:

- A method is proposed for identifying population subgroups for which expensive tests may be avoided, and experiments with a real-world database on a cardiological prognostic problem suggest that significant savings are possible.
- A method is proposed for anonymizing medical databases with sensitive contents via cell suppression, thus aiding to preserve patient confidentiality.
- Very simple rule-based classifiers are employed to diagnose acute appendicitis, and their relative performance is compared to a team of experienced surgeons. The added value of certain biochemical tests is also demonstrated.

Contents

Preface	xiii
I Setting	1
1 Introduction	3
1.1 Introduction	3
1.2 Objectives	4
1.3 Results	5
1.4 Thesis Outline	6
1.4.1 A Roadmap	8
2 Context	11
2.1 Introduction	11
2.2 Data Mining and Knowledge Discovery	12
2.2.1 The KDD Process	12
2.2.2 KDD and Rough Sets	14
2.2.3 Themes in KDD	15
2.3 Medical Informatics	16
2.3.1 Themes in Medical Informatics	17
2.4 Aspects of KDD in Medicine	18
2.4.1 On Data Availability and Quality	18
2.4.2 On Model Induction and Selection	20
2.4.3 On Model Assessment and Deployment	21
3 Rough Sets in Medicine	25
3.1 Introduction	25
3.2 Literature Review	25
3.2.1 Diagnosis and Outcome Prediction	26
3.2.2 Feature Selection	27
3.2.3 Miscellaneous	28
3.3 Appeal	29
3.4 Discussion	30
3.4.1 On Prerequisites and Assumptions	30
3.4.2 On Interpretation and Deployment	31
3.4.3 On Model Maintenance and Assessment	32

II Preliminaries	35
4 Boolean Reasoning	37
4.1 Introduction	37
4.2 Boolean Algebras	37
4.3 Implicants	38
4.4 General Scheme	39
5 Discernibility and Rough Sets	41
5.1 Introduction	41
5.2 Information Systems	41
5.2.1 Discernibility Matrices	42
5.2.2 Indiscernibility Relations	45
5.2.3 Rough Sets	47
5.2.4 Discernibility Functions	48
5.2.5 Reducts	50
5.2.6 Reduct Approximations	53
5.3 Decision Systems	55
5.3.1 Decision Classes	55
5.3.2 Generalized Decisions	56
5.3.3 Discernibility Matrices	56
5.3.4 Discernibility Functions	57
5.3.5 Reducts	58
6 Patterns and Rules	61
6.1 Introduction	61
6.2 Patterns	61
6.2.1 Syntax	61
6.2.2 Semantics	62
6.2.3 Minimal Patterns	62
6.2.4 Numerical Measures	63
6.3 Decision Rules	63
6.3.1 Minimal Decision Rules	64
6.3.2 Numerical Measures	64
6.3.3 Balancing Accuracy and Coverage	66
6.4 Classification	66
6.4.1 Voting	66
6.4.2 Object Tracking	68
7 Classifier Evaluation	71
7.1 Introduction	71
7.2 Partitioning the Examples	72
7.2.1 Systematic Partitioning Methods	73
7.3 Confusion Matrices	75
7.4 Binary Classifiers	75
7.4.1 Realizing ϕ	77

7.4.2	Discrimination and Calibration	77
7.4.3	ROC Curves	78
7.4.4	Calibration Plots	81
7.4.5	Performance Measures	83
7.4.6	Statistical Hypothesis Testing	86
III Tools		91
8	The ROSETTA System	93
8.1	Introduction	93
8.2	System Justification	94
8.3	GUI Overview	95
8.4	Methodological Support	98
8.4.1	Selection	98
8.4.2	Preprocessing	99
8.4.3	Transformation	100
8.4.4	Data Mining	100
8.4.5	Interpretation and Evaluation	101
8.4.6	Automation	101
8.4.7	Deployment	102
8.4.8	Miscellaneous	102
8.5	Miscellaneous	102
9	A ROSETTA Case Study	105
9.1	Introduction	105
9.2	GUI Preliminaries	105
9.3	Data Material	106
9.4	Experimental Setup	106
9.4.1	Comments	109
9.4.2	Parameters	110
9.4.3	Results	112
9.5	Systematic Partitioning	116
9.5.1	Parameters	116
9.5.2	Results	117
9.6	Comments	117
10	Design and Implementation	119
10.1	Introduction	119
10.2	System Requirements	119
10.2.1	Kernel Requirements	121
10.2.2	Front-End Requirements	122
10.3	The Computational Kernel	123
10.3.1	Handles	123
10.3.2	Transitive Self-Identification	123
10.3.3	Legacy Code	124

10.3.4	Creational Patterns	124
10.3.5	Main Object Dichotomy	125
10.3.6	Algorithm Application	126
10.3.7	Miscellaneous	127
10.3.8	A Small Example	127
10.4	The GUI Front-End	127
10.4.1	Workspace	129
10.4.2	Method Invocation	129
10.4.3	Kernel Reflection	130
IV	Applications	131
11	Identifying Population Subgroups	133
11.1	Introduction	133
11.2	Preliminaries	134
11.3	Methodology	134
11.3.1	Approximation Evaluation	136
11.4	Experiments	137
11.5	Results	139
11.5.1	Meta Results	140
11.6	Analysis and Discussion	141
12	Anonymizing Sensitive Data	143
12.1	Introduction	143
12.2	Preliminaries	144
12.3	Main Approaches	145
12.3.1	Removing Outliers	146
12.3.2	Generalization	146
12.3.3	Cell Suppression	146
12.4	Methodology	147
12.4.1	Basic Algorithm	147
12.4.2	Anonymizing a Database	149
12.4.3	Relative Anonymization	150
12.4.4	Hampering Reconstruction	152
12.5	Software Systems	154
12.6	Examples	154
12.6.1	Anonymization	154
12.6.2	Relative Anonymization	156
12.6.3	Remarks	158
12.7	Analysis and Discussion	158
12.7.1	Interpretation and Links	159
12.7.2	Changing Biases	160
12.7.3	Complexity Issues	160
12.7.4	Record Order and Multiplicity	161
12.7.5	Additional Suppression	162

13 Diagnosing Acute Appendicitis	163
13.1 Introduction	163
13.2 Preliminaries	164
13.3 Methodology	164
13.4 Experiments	166
13.5 Results	168
13.6 Analysis	168
13.7 Discussion	172
13.8 Conclusions	174
V Synopsis	177
14 Summary and Results	179
14.1 Summary	179
14.2 Main Results and Contributions	181
15 Further Work	185
15.1 Introduction	185
15.2 General	185
15.3 Tools	187
15.4 Applications	189
VI Appendices	191
A Nomenclature	193
B The ROSETTA C++ Library	195
C The 5x2CV Test	199
D Discretization	201

List of Figures

1.1	Thesis roadmap.	9
2.1	Outline of the overall KDD process.	13
5.1	An example concept hierarchy from the medical domain.	44
5.2	Lower and upper set approximations.	47
5.3	A graphical metaphor between addition and reducts.	51
5.4	Different types of reducts of an information system.	51
5.5	Approximate hitting sets versus dynamic reducts.	54
5.6	Different types of reducts of a decision system.	59
6.1	Conceptual depiction of accuracy and coverage.	65
6.2	The inverse relationship between accuracy and coverage.	66
7.1	Points on an example ROC curve.	78
7.2	Domination among ROC curves.	79
7.3	An example calibration plot.	82
7.4	An example calibration function.	83
7.5	Two-sided statistical hypothesis testing.	86
8.1	An example ROSETTA project tree.	96
8.2	An example ROSETTA workspace.	97
9.1	Experimental setup for the case study.	108
9.2	An IDG file in ROSETTA format.	109
9.3	A file with cost information in ROSETTA format.	111
9.4	Confusion matrix for the Cleveland database.	113
9.5	Performance plots for the Cleveland database.	115
9.6	A CV script in ROSETTA format.	117
9.7	Excerpt of the CV log file.	118
10.1	An example pipeline for producing Prolog rules.	128
10.2	Implementation of Figure 10.1 with the ROSETTA library.	128
11.1	Monitoring the change in the boundary region.	136
12.1	Making an object in a database anonymous.	147
12.2	An order-sensitive method for anonymizing a database.	149

12.3	An order-insensitive way of anonymizing a database.	150
12.4	Anonymizing an object relative to an outcome.	151
12.5	Making an object in a database anonymous, revisited.	161
13.1	ROC curves for patients with suspected acute appendicitis (1/2).	170
13.2	ROC curves for patients with suspected acute appendicitis (2/2).	170
13.3	Situations where a 1R approach will fail.	173

List of Tables

5.1	Defining discernibility by IDGs.	45
7.1	Names of entries and quantities from 2×2 matrices.	76
7.2	Generating points in a calibration plot.	82
8.1	An overview of how each ROSETTA IDG edge specification alters E_a	99
9.1	Summary of the coronary artery disease data material.	107
9.2	Intervals computed from part of the Cleveland database.	113
9.3	Some rules induced from the Cleveland database.	114
11.1	Summary of the hard cardiac event data material.	138
11.2	Approximation results for patients with future hard cardiac events.	139
11.3	Approximation meta-results.	140
12.1	Cell reconstruction using background knowledge.	153
12.2	Example database before and after one anonymization cycle.	155
12.3	Example database before and after one relative anonymization cycle.	157
13.1	Summary of the acute appendicitis data material.	166
13.2	Initial search for “best” attributes.	167
13.3	Formal definitions of the employed 1R classifiers.	167
13.4	Results from a single 10-fold CV run.	169
13.5	Results from five different 2-fold CV runs.	169
13.6	Rules for predicting acute appendicitis.	171
13.7	Pairwise statistical analysis of the results in Table 13.5.	172
B.1	ROSETTA C++ library statistics.	197

Preface

This dissertation is submitted to the Norwegian University of Science and Technology (NTNU) in partial fulfillment of the requirements for the degree *doktor ingeniør*. The work has been carried out at the Knowledge Systems Group [101], Department of Computer and Information Science, NTNU. Parts of the work were done while on a research stay at the Decision Systems Group [39], Harvard Medical School, Boston, USA.

Acknowledgments

This work could not have been carried out without both direct and indirect help and support from others, and many collaborators and friends deserve credit.

I thank my supervisor Professor Jan Komorowski for introducing me to the topic of discernibility and logically based methods for data analysis, for support, and for opening doors.

Thanks to Professor Bob Greenes at Harvard Medical School for hosting my visit to Boston, and to all members of the Decision Systems Group for making my stay an interesting and enjoyable one. Special thanks to Lucila Ohno-Machado for comments and suggestions on model construction and validation, and for being such a friendly and responsive sounding board. Thanks also to Todd Rowland for helping me explain my field of work to a broader audience.

Professor Andrzej Skowron and his group at the Department of Mathematics at the University of Warsaw, Poland, made available the RSES library at an early stage of development of ROSETTA, something that is much appreciated. Special thanks to Piotr Synak for answering all the questions I had when I wrote the wrappers needed to embed portions of RSES as a ROSETTA component.

Thanks to all fellow members, past and present, of the Knowledge Systems Group; in alphabetical order, Tore Amble, Tor-Kristian Jenssen, Mihhail Matskin, Torulf Mollestad, Sobah Abbas Pettersen, Amund Tveit, Staal Vinterbo, Agata Wrzos-Kamińska, Jacek Wrzos-Kamiński and Thomas Ågotnes. Special thanks to my fellow PhD students for many interesting and idea-spawning discussions. I could not have shared an office with a friendlier and more helpful person than Staal.

Contact with Professor Bjørn Angelsen and his group at the Department of Physiology and Biomedical Engineering at NTNU at an early stage of this work was inspiring, and helped me to overcome part of the language barrier that exists between the worlds of technology and medicine.

The usability of ROSETTA has been greatly increased through valuable feedback from the students whom I have supervised and helped in their project or diploma works. In alphabetical order, these include Ulf Carlin, Merete Hvalshagen, Torgeir Rhodén Hvidsten, Tor-Kristian Jenssen, Terje Løken, Jørn Erlend Nygjerd, Daniel Remmem, Knut Magne Risvik, Helge Grenager Solheim, Dyre Tjeldvoll, Ivan Uthus and Thomas Ågotnes. Most helpful was also the assistance from some of these in unraveling a few initial mysteries of MFC and Windows programming.

Several people have played miscellaneous roles that deserve mention. Ron van Domburg at the University Hospital of Rotterdam-Dijkzigt, The Netherlands, supplied the cardiac database. Piotr Szymański from the Postgraduate Medical School in Warsaw, Poland, helped formulate the patient identification procedure. Latanya Sweeney at the Massachusetts Institute of Technology, Cambridge, USA, introduced me to data anonymization and encouraged me to make a contribution in this area. Arne Åsberg and Stein Hallan at the Department of Clinical Chemistry, University Hospital of Trondheim, Norway, made available the appendicitis database.

Thanks also to the administrative and technical support staff at the Department of Computer and Information Science, NTNU, for providing necessary infrastructure.

This research is sponsored in part by the Norwegian Research Council under grant 74467/410. Additional sources of funding have been the European Union under the 4th Framework Telematics project CARDIASSIST, and travel grants from the Department of Computer and Information science at NTNU, Generaldirektør Rolf Østbyes stipendiefond, Norges tekniske høgskoles fond and Det Norske Veritas. Their financial support is greatly appreciated.

My friends and family have given me many enjoyable and much needed distractions and breaks from my work. My parents have always been a continuous source of support, for which I am immensely grateful. Last but not least I thank Janne, without whom nothing would be much worth doing.

In fond memory of my father, Johannes Øhrn.



Aleksander Øhrn
December 16, 1999

Part I

Setting

Chapter 1

Introduction

1.1 Introduction

Medicine has from early on often been employed as a testbed domain for newly developed learning and reasoning techniques from computer science [67, 114, 134, 187, 189]. Not only because the field of medicine has many applications whose solutions are important in a social context, but also because the field is notoriously difficult with a wide range of confounding factors and aspects that demand special considerations, hence forming a technically challenging area. This interplay between computer science and medicine has led to some remarkable work being accomplished in the last 30 years, but major developmental efforts and research is still needed if a significant impact on the practice of medicine is to be realized [35, 218]. The directions to take for construction of the underlying computational models¹ may differ. Traditionally, models from the field of *artificial intelligence* have been very knowledge-intensive in the sense that domain-specific knowledge about such things as etiology and pathophysiology have been carefully encoded by hand into the model. Although much is to be said for such a “top-down” approach to model construction, this is an extremely complex and laborious task and may ultimately lead to problems with model maintenance. Additionally, extensive domain expertise and the ability to produce at least a partial problem formulation and model specification is required. On the other end of the spectrum are “bottom-up” approaches where models are attempted induced or synthesized from low-level data without relying on a priori domain knowledge. A shift in vogue towards such methods has been observed in the last decade, something which is also in line with an increasing emphasis in medicine on evidence-based practice [35]. Naturally, hybrid approaches can be envisioned. For example, some bottom-up methods may employ background domain knowledge to guide the model induction process.

Large databases are often collected for research or business purposes. Often these

¹The term *model* will be used frequently throughout this thesis. Generally speaking, we can say that an entity \mathcal{M} is a model of an entity \mathcal{A} if \mathcal{M} can be used to answer questions concerning some of the characteristic properties of \mathcal{A} [242].

databases grow so large that human inspection and interpretation of the data is not feasible, with a gap between data generation and data understanding as a result. Clearly, tools and techniques that can aid in extracting unknown interesting patterns buried in the data would be useful to help bridge this gap. Classical tools for database querying may be adequate if you know what to look for. For instance, current systems are good for answering questions of the type “How many patients suffered from spinal cord injuries in California last February, who are they, and what was their average length of stay in hospital?” But often the most interesting queries to pose cannot be formulated as straightforward lookups. Consider for instance the following question of interest: “What are the main factors that determine how successful our current rehabilitation program is, and how do these factors interact?” To be able to deal with such advanced queries, more intelligent data analysis tools are needed. Research in computer science has in the last decades spawned a vast multitude of methods that “learn” from examples, and that can be used to extract patterns from empirical data for classification. Such techniques are increasingly being applied to medical data sets.

Pawlak [154, 155] introduced *rough set theory* in the early 1980s as a tool for representing and reasoning about imprecise or uncertain information. Based on the notion of indiscernibility and the inability to distinguish between objects, rough set theory deals with the approximation of sets or concepts by means of binary relations, typically constructed from empirical data. Such approximations can be said to form models of our target concepts, and hence in its typical use falls in under the bottom-up approach to model construction. As the methodology has matured, several interesting applications of the theory have surfaced, also in medicine. For example, in a medical setting, sets of interest to approximate could be the set of patients with a certain disease or outcome, or the set of patients responsive to a certain treatment.

This thesis concerns itself with tools for and applications of discernibility and rough sets in medicine. As such, the context in which to place this work is an intersection of several scientific areas, the perhaps two most relevant being the field of *data mining and knowledge discovery*, and the field of *medical informatics*. This work focuses on the development of tools and techniques from a certain subfield of the former area, and applying them in the latter.

1.2 Objectives

Obviously, targeting the medical domain has an important social dimension since solutions to relevant problems in this domain may have a beneficial impact on human beings and their welfare. As such, this thesis constitutes an incremental contribution towards the overall long-term goal of improving the quality of healthcare and lowering costs. There is also a challenging technical dimension to targeting the medical domain, since the art of medicine is not an exact science in which processes are easily formalized or modelled. Furthermore, these two dimensions intertwine in a way that exerts tight constraints on the solution space.

Many aspects of what we might perceive as intelligent behavior can essentially be re-

duced to the ability to classify. And any classification task crucially relies on an ability to appropriately discern between objects or situations. Discernibility-based methods, and rough set theory in particular, are intuitively appealing and involve, technicalities aside, rather simple ideas. Also, their formal soundness defines a solid theoretical basis for empirical modelling.

The main objectives of this thesis are:

- To discuss and illuminate the usefulness of discernibility-based methods for data mining in the health sciences, and to demonstrate that these ideas are indeed applicable by devising relevant and novel medical applications.
- To furnish the research community with a set of flexible and powerful software tools for conducting data analyses within the framework of rough set theory, and to provide a software environment which facilitates such experimentation. Under the hypothesis that the accessibility of such tools lowers the threshold for abstract ideas to migrate into concrete realization, this aids in reducing a gap between theoreticians and practitioners, and enables existing problems to be more easily attacked.

1.3 Results

The work in this thesis has been carried out as a combination of adaptations of relevant theoretical constructs, programming and computer simulations. Briefly, the main contributions of this thesis are, in no particular order:

- The discernibility-based approach to data mining and knowledge discovery is considered in the light of general demands imposed by the medical domain. Bringing together pertinent techniques, suitable adaptations of relevant theory for model construction and assessment are presented.
- The ROSETTA system for data analysis within the framework for rough set theory has been developed. The system implements features relevant to build and evaluate rough set models in the medical domain, and offers a highly user-friendly environment in which to conduct experiments.

Although by design equipped with several features that are relevant for analysis of medical data, ROSETTA is in itself a general-purpose system that is not geared towards any particular application domain. ROSETTA has been put to use by a large number of researchers world-wide, and has resulted in scientific publications in a wide variety of areas.

- Novel medical applications have been devised, which, although diverse in theme and scope, all share a common discernibility-based foundation:
 - A method is proposed for identifying population subgroups for which expensive tests may be avoided, and experiments with a real-world database

on a cardiological prognostic problem suggest that significant savings are often possible.

- A method is proposed for anonymizing medical databases with sensitive contents via cell suppression, thus aiding to preserve patient confidentiality.
- Very simple rule-based classifiers are employed to diagnose acute appendicitis, and are shown to compare favorably with a team of experienced surgeons. The added value of certain biochemical tests is also demonstrated.

The ROSETTA system has been employed for all examples and experiments throughout.

This thesis draws together various work performed over a period of almost four years. As such, portions of several chapters have, directly or indirectly, previously appeared as [5, 24, 26, 27, 88, 93, 94, 104, 105, 137–151, 181, 232, 233].

1.4 Thesis Outline

The reader of this thesis need not possess any prior expertise of the areas presented, as the relevant ideas and concepts will be introduced and explained in subsequent chapters. In particular, no previous knowledge about rough set theory is required, nor is this a medical dissertation. However, some prior familiarity, even superficial, of empirical modelling would be useful. Some rudimentary knowledge of databases, discrete mathematics, logic and statistics will be assumed.

The remainder of the thesis is structured as follows:

Part I provides the setting of the thesis, outlines the context and reviews previous relevant work.

Chapter 2 presents the context of this work, namely the intersection of the medical informatics field and the field of data mining and knowledge discovery. The two fields are very briefly reviewed on their own, and special considerations about their intersection are discussed.

Chapter 3 concerns itself with the appeal, drawbacks and previous uses of rough sets in medicine. A list of points that make the methodology attractive for practical use is given and subsequently discussed, and the current literature on applications of rough sets in medicine is reviewed.

Part II introduces, defines and explains the theoretical constructs from discrete mathematics, logic, rough set theory and statistics that are relevant for this work.

Chapter 4 briefly outlines some topics from Boolean reasoning. Several of the most practically important concepts in rough set theory can be conveniently

interpreted in the realm of Boolean reasoning, and Boolean reasoning techniques are typically applied to solve minimization problems in rough set theory.

Chapter 5 introduces rough set theory and some of its theoretical foundations. Starting from the notion of discernibility and discernibility matrices, indiscernibility relations are defined and set approximations via these are introduced. Boolean discernibility functions are defined and the notion of minimal discernibility-preserving sets of attributes are explored.

Chapter 6 presents how minimal descriptive patterns and if-then rules can be generated. Furthermore, an overview of numerical measures of patterns and rules is presented, along with an introduction to how ensembles of if-then rules can be used to realize classifiers.

Chapter 7 discusses ways of evaluating classifier performance, in particular classifiers with binary outcomes. The topics of discrimination and calibration are discussed, and some pertinent statistical tests are presented.

Part III presents the software tools that have been developed to facilitate discernibility-based empirical modelling.

Chapter 8 introduces the ROSETTA software system for data analysis, and gives an overview of its main features.

Chapter 9 provides a detailed example of how ROSETTA can be used to analyze a medical database. As a case study, a publicly available database on coronary artery disease is employed.

Chapter 10 discusses aspects of the design and implementation of ROSETTA. A set of system requirements and design parameters are outlined, and a presentation of how these are met is given along with the rationale behind them.

Part IV outlines some medical applications where discernibility and rough sets have been applied using the developed ROSETTA software system.

Chapter 11 reports on a novel application of using the semantics of rough sets for feature extraction. The proposed method can be used to identify certain population subgroups for whom acquiring knowledge of potentially costly information is strictly needed for purposes of classification. An application to a prognostic problem in cardiology is given, where the aim is to identify the subgroup for whom performing a scintigraphic scan can be avoided.

Chapter 12 presents a novel application of using discernibility to anonymize the contents of sensitive medical databases via cell suppression. The proposed method can be used as a subcomponent in a full-fledged system for preserving confidentiality prior to data dissemination, and can be viewed as an instance of data mining applied in reverse. The cell suppression algorithm obfuscates data so that identifying patterns are subsequently being made difficult to discover, and does so in a manner that preserves the "truthfulness" of the data material.

Chapter 13 lays out an application where a collection of if-then rules are used to diagnose the presence or absence of acute appendicitis. Whereas Chapter 11 and Chapter 12 focus on multivariate aspects, Chapter 13 investigates the utility of extremely simple univariate classification rules. The classifier is compared to the performance of a team of physicians and with results from the literature. Furthermore, the issue of additional value of certain biochemical tests to the model is investigated.

Part V collects the threads from the previous parts, and points out some directions for further research.

Chapter 14 contains a detailed summary of the thesis, and lists its main results and contributions.

Chapter 15 outlines some directions for future work.

Appendices of various kinds can be found last.

Appendix A provides a list of the mathematical notation and abbreviations used throughout this thesis.

Appendix B gives a brief outline of the structure of the ROSETTA C++ library presented in Chapter 10.

Appendix C presents the 5x2CV F -test employed in Chapter 13.

Appendix D presents how considerations based on discernibility can be used to discretize numerical attributes.

1.4.1 A Roadmap

Figure 1.1 provides a thesis roadmap, depicting dependencies between the various parts. The logical order to read this thesis is to read the parts in sequence. However, Parts III and IV may be read independently of each other.

Within each part, the chapters should normally be read in consecutive order. However, there are several exceptions that can be made:

- In Part II, Chapter 7 can be read independently of Chapters 4, 5 and 6.
- In Part III, Chapter 10 can be skipped without loss of continuity.
- In Part IV, Chapters 11, 12 and 13 are independent entities.

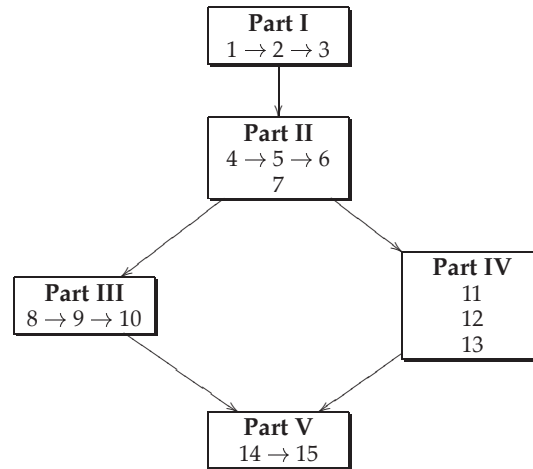


Figure 1.1: The logical order to read this thesis is to read the parts in sequence. However, Parts III and IV may be read independently of each other. Within each part, the chapters should normally be read in consecutive order. Some chapters can be read as separate entities, as indicated.

Chapter 2

Context

2.1 Introduction

A proper context in which to place this thesis might be in the intersection between the following scientific fields:

- *Data mining and knowledge discovery*: A field which revolves around constructing computer models from databases. These models may serve to reveal “nuggets of knowledge” previously hidden in the data, or to predict attribute values for future observations.
- *Medical informatics*: A field which studies the acquisition, encoding, organization, transmittal and deployment of medical data, with the purpose of exploiting computers to improve various aspects of healthcare.

The topic of this work is on the development of tools and techniques from a certain subfield of the former area, and applying them in the latter. This chapter aims to provide a brief glimpse of the nature of these fields, and of issues that should be considered when they are brought together. Due to the enormous scope and myriad of subfields and current activities in these fields, this chapter does not in any way try to give a complete review. Rather, only some chosen topics deemed the most relevant in the context of this thesis are very briefly mentioned. More thorough treatments can be found in the literature, e.g., [53, 100, 167, 191, 228].

Section 2.2 gives a very brief rundown and presentation of the field of data mining and knowledge discovery, and Section 2.3 briefly reviews the medical informatics field. In Section 2.4, some observations about the nature of the intersection of these fields are given.

2.2 Data Mining and Knowledge Discovery

The problem of finding models that describe or classify measurement data is encountered in many situations. This task falls into the extensive category of *empirical modelling*, which, broadly speaking, can be said to be the science of constructing models that describe or classify measurements. Such models may take on a wide variety of forms according to the model construction scheme used. The choice of modelling scheme bears both to the nature of the pattern recognition problem, as well as to the purpose of the modelling task. The purpose of developing such models may be twofold: In some instances, the goal may be to gain insight into the problem at hand by analyzing the constructed model, i.e., the structure of the model is itself of interest. In other applications, the transparency and explainability features of the model is of secondary importance, and the main objective is to construct a classifier of some sort that classifies arbitrary objects well. When constructing models on the basis of labeled training data, a knowledge-based approach to such empirical modelling is to inductively infer a set of general rules that produce the desired class. This is an instance of an activity that is collectively referred to as *knowledge discovery in databases* (KDD), or, sometimes, *data mining*. In the following, the term knowledge discovery will be used to denote a process of which data mining is a component. The term *machine learning* is often used as a common label for many of the techniques involved in learning from examples.

KDD is commonly defined [167] as “the nontrivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data”. The term “data” is here understood as a collection of facts or atomic pieces of information, e.g., records in a database, while “knowledge” is a higher-level concept that says something about the properties of the collection of data as a whole, e.g., dependencies among sets of attributes in a database, or rules for predicting attribute values. The distinctions between “data” via “information” to “knowledge” are somewhat blurry and open to debate, but intuitive notions will suffice in the present context.

2.2.1 The KDD Process

The overall KDD process consists of several steps and phases that are iterated in a waterfall-like cycle [54], and is displayed graphically in Figure 2.1. The exposition is not in any way specific about the technical approaches taken at each step. From a data source containing raw data, all or portions of this is selected for further processing. The selected raw data is then typically preprocessed and transformed in some way, before being passed on to the data mining algorithm itself. The patterns output from the mining procedure are then postprocessed, interpreted and evaluated, hopefully revealing new knowledge previously buried in the data. Along the way, backtracking on each of the steps will in practice inevitably occur. Efforts to define more formal process models are ongoing [29].

The selection phase more or less defines the KDD problem. The saying “garbage in, garbage out” also applies to KDD, but since we are looking for relationships that are

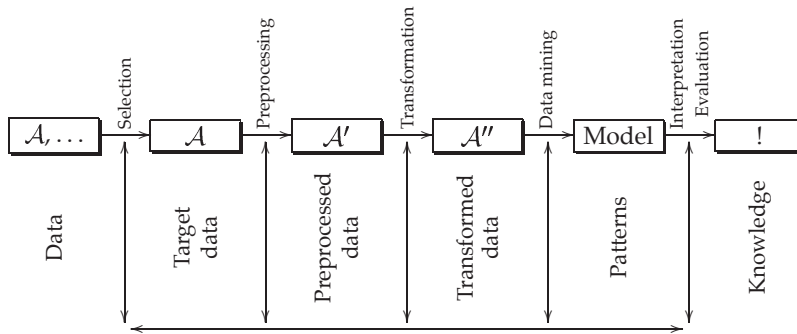


Figure 2.1: Outline of the overall KDD process, adapted from Fayyad et al. [54]. The patterns output from the data mining step do not necessarily have to be on the form of if-then rules, but depend on the selected modelling scheme. For example, if logistic regression is employed, the model to interpret would be a set of numerical coefficients. Note the iterative and nonlinear nature of the overall KDD process.

presumably unknown beforehand, it might not be entirely straightforward to a priori specify what constitutes “garbage” and what constitutes potentially valuable input. Letting a method loose on data material that has not been sufficiently “focused” by narrowing down the number of inputs may not only increase the computational burden in the data mining step, but is also likely to result in spurious and nonsensical correlations being found. In practice, making an effort to ask the right questions and properly formulate the KDD problem may prove more fruitful than spending time optimizing algorithmic details [54].

In the preprocessing and transformation steps, care has to be taken that no unwanted biases are introduced. How these steps are executed depend largely on the method selected for the subsequent data mining step.

In the data mining step, many different learning and modelling paradigms are plausible candidates. Some of these are inherently symbolically oriented and are “trained” through logical and algebraic methods, others have a numerical foundation and are trained by regression and curve-fitting methods. Which methods that are appropriate depends in part on whether the target to learn is continuous or can be treated as a discrete classification task. In the following, unless otherwise stated, it will be assumed that the modelling task can be reduced to a classification problem. Popular methods include:

- *Logistic regression:* The perhaps most popular analysis technique within the health sciences. Adjusts a set of numerical coefficients so that a best-fit is obtained between a sigmoidal function and the data [85].
- *Neural networks:* Simple nonlinear processing elements are interconnected in a large network in which an input signal is propagated towards one or more designated output nodes [80]. Typically, network weights are adjusted through a

gradient descent procedure to optimize a global objective function [182].

- *Belief networks*: Also called Bayesian networks. Each variable or attribute is represented by a node in a directed graph, where the edges in the graph reflect attribute interdependencies [79]. The network structure can either be supplied by a domain expert or attempted induced from the data. Estimates of conditional probabilities at each node are obtained from the data.
- *Decision trees*: Recursively partitions the input space according to various data-driven splitting criteria [175]. Subsequent predictions can be made by propagating a sample from the root towards the leaves, and noting the class distribution of training instances in the destination leaf.
- *Rule-based models*: Collections of if-then rules are produced that fully or approximately describe the example classifications in the training material [32, 124, 192]. Rules can also be induced via decision trees.

2.2.2 KDD and Rough Sets

Rough set theory is suitable for problems that can be formulated as classification tasks, and has gained significant scientific interest as a framework for data mining and KDD [126, 168]. Although the main computational workhorse in many symbolically based methodologies is the well-known propositional calculus, rough sets provide a unifying framework in which the computations can be interpreted.

Adapting the elements of Figure 2.1 to how models are typically constructed in the framework of rough sets, the following items can be noted:

- *Selection*: The basic vehicle for data representation in the rough set framework are flat, two-dimensional data tables. This does not imply that a table has to be a single physical table, it may very well be logical view across several underlying tables. One suitably formed table is selected for subsequent analysis.
- *Preprocessing*: If the selected table contains “holes” in the form of missing values or empty cell entries, the table may be processed in various ways to yield a completed table in which all entries are present.
- *Transformation*: Numerical attributes and attributes that have an ordering on them may have to be discretized, i.e., transformed in such a way that intervals or ranges are used instead of the exact observations themselves. This amounts to defining a coarser view of the world, and to making quantitative data more qualitative.
- *Data mining*: In the rough set approach, conjunctions of elementary propositions or if-then rules are produced. This is done in a two-stage process, in which minimal attribute subsets are first computed before patterns or rules are in turn generated from these.

- *Interpretation and evaluation:* Individual patterns or rules can be ordered by some measure of “goodness” and manually inspected. Ensembles of rules can be employed to classify new cases and noting their classificatory performance.

Note that not all of these steps are necessarily carried out, but vary with the application at hand. In some cases the preprocessing and transformation steps may not be needed or even desirable to execute. Furthermore, many of the steps above are not specific to the rough set approach, but are shared across many different learning approaches. For example, most methods operate on flat data tables, and discretization is a step that almost all symbolically based data mining methods require for them to perform well. If not stated as an explicit requirement, discretization is often performed implicitly behind the scenes.

2.2.3 Themes in KDD

KDD draws on many fields, such as statistics, database theory and artificial intelligence. Current research themes in KDD are therefore scattered across a large array of topics that intertwine, including:

- *Data representation:* There is a growing interest towards investigating going beyond simple two-dimensional data tables, and to try to mine free text, multimedia databases and the world wide web. This last item also bears to the topic of mining distributed data in general.
- *Learning paradigms:* In terms of the number of publications, a large body of the research in data mining and KDD seems to be within incremental improvements of facets of different learning and modelling techniques, and on the empirical verification of these.
- *Use of background knowledge:* If any domain knowledge is present, it would be desirable to employ it. Techniques for incorporating problem-specific knowledge into the model construction process are therefore of great interest to develop. Use of background knowledge may also aid in analyzing tabular data in which many of the entries are missing.
- *Disk-based algorithms:* An implicit assumption made by most common data analysis algorithms is that all the data can simultaneously reside in the computer’s main memory for analysis. For massive amounts of data, this is clearly not a viable assumption, and efforts are made to develop efficient disk-based analysis algorithms.
- *Model pruning and simplification:* Since we ultimately want the models to be understandable, a large body of research has gone into developing methods for simplifying models. Approaches to this include pruning or filtering an existing model, and transforming one type of model into another more interpretable type.

- *Visualization*: Techniques for visualizing relevant portions of data and knowledge summaries are immensely powerful and useful, and may greatly enhance any step of the KDD process.
- *Quality assessment of data mining results*: After a model has been mined it is desirable to assess its qualities. Methods for knowledge evaluation, benchmarks and metrics for system evaluation and statistical tests in KDD applications are therefore needed.
- *Man-machine interaction*: The KDD process is more likely to succeed with human intervention, since it involves many choices that have to be made on the basis of the results that gradually unfold. User interfaces and environments that allow such intervention and that combine querying and KDD are therefore essential.

Other themes in KDD include how to decompose and parallelize steps of the KDD process, development of methods for discretization and other preprocessing techniques, discovery of trends and the mining of time series, ensuring data quality and managing changing data, to name but a few.

2.3 Medical Informatics

Computer science is a field that permeates today's society, with applications in almost every domain. Sometimes hybrid fields emerge that meld disciplines together with special focus on the needs and special considerations of the application domain. *Medical informatics* is such a field. Blois and Shortliffe [191] define medical informatics as "the rapidly developing scientific field that deals with the storage, retrieval and optimal use of biomedical information, data, and knowledge for problem solving and decision making." This is indeed a very wide definition, but Blois and Shortliffe provide some delineations to neighboring fields. For instance, biomedical engineering is in their view more device-oriented than medical informatics, which in turn emphasizes higher-level information processes. Many issues in medical informatics can sometimes be phrased as rather general computer science issues, but that become perhaps more pertinent and pressing in the context of modern healthcare.

In its broadest sense, medical informatics can be said to concern itself with the management of information in the context of healthcare, and undergoes a rapid pace of change and development as computer technology continues to accelerate. This pace of change and development is hardly likely to slow down in the future. A transition towards medicine based on an understanding of the human genome, the expectation of quality healthcare provided at a lower cost and the requirement for improved access to healthcare, are all factors that will contribute to this.

2.3.1 Themes in Medical Informatics

The medical informatics field encompasses a broad range of themes, both in terms of technical issues as well as with respect to the medical context the methods are set in. Current research therefore covers a wide array of topics, including:

- *Data acquisition:* Capturing and recording medical data in electronic form is a major bottleneck in applying computers in healthcare. Medical data include things that are not easily recorded or precisely defined. From the viewpoint of designing intelligent systems that reason about medical data, this constitutes a formidable challenge.
- *Medical vocabularies:* Obviously, data has to be somehow represented in coded and machine-readable form if it is to be made proper use of in a computer system. An important part of the body of medical knowledge are nomenclatures and ontologies for accurately describing, cataloguing and classifying elements in the finding-disease continuum. Such systems are commonly referred to as controlled medical terminologies (CMTs).
- *Electronic medical records:* For an electronic medical record (EMR) to be searchable and its information content harvested, its content should be encoded using some kind of CMT. Furthermore, once the content of a medical record is in place, there is still the issue of how to structure the content internally and how to tailor suitable views of an EMR's content for various user groups.
- *Decision support systems:* Computer programs that help clinicians make clinical decisions typically can typically be divided into three groups [190]: Tools for information management, tools for focusing attention and tools for patient-specific consultation. Ways to build, maintain and evaluate such decision support systems (DSSs) are of interest to develop.
- *Deployment barriers:* Systems that may prove successful in research settings often do not make it into clinical use. Barriers of deployment may be of both technical, operational, organizational and legal nature, and the identification of these and ways to overcome them are active research issues in medical informatics.
- *Confidentiality issues:* Medical information is often sensitive and with a potential for misuse if obtained by malevolent third-parties. The challenge that sound management of confidential data represents has not gone unnoticed in the medical informatics community.

Other themes in medical informatics abound, and include computerization of and tools for care guidelines and protocols, ways of evaluating computer systems in healthcare, elicitation and use of patient preferences in decision analysis, and telemedicine, to name but a few.

2.4 Aspects of KDD in Medicine

Bringing together two fields can elucidate the validity of some previously held assumptions and beliefs, and often results in domain-specific considerations. The intersection between data mining and KDD and medical informatics is no exception. This section discusses some general points to be taken from medical informatics for practitioners of data mining and KDD in the medical domain.

The perhaps most obvious place for data mining techniques in medicine is in conjunction with DSSs. Tools for focusing attention and tools for patient-specific consultation are where data mining and machine learning techniques are most likely to contribute to clinical DSSs. Such techniques may provide focus by discarding superfluous information, and provide predictive models that can be applied for patient-specific consultation. However, the explanatory capabilities of such models vary according to the modelling paradigm. It is generally desirable for any DSS to be able to justify the recommendations it makes, at least in part, something which is often stressed in the context of medicine.

The scope of tools for decision support for patient-specific consultation vary immensely. Although a continuum, two main categories can be identified:

1. Tools designed to provide a probability or scoring index of a fixed outcome or diagnosis.
2. Tools that employ large knowledge-bases designed to suggest a broad range of differential diagnoses.

Current state-of-the-art techniques for empirical modelling are capable of constructing tools of the first type. The more ambitious goal of fully automatic construction of large knowledge-bases for tools of the second type is probably too complex and difficult a task for current technologies.¹ Of course, tools of the first type can be employed as small components of larger DSSs, and thus contribute to making such systems more evidence-based.

2.4.1 On Data Availability and Quality

Practitioners of KDD might tend to take the existence of large amounts of usable data for granted. However, obtaining historical clinical data in electronic form may not always be easy or even possible. Acquisition of such data may in practice often involve laborious and non-automated tasks. Traditionally, most hospitals use transcription services where the dictated notes of the physicians are transcribed into the medical

¹Four carefully crafted broad-ranging systems for computer-based diagnosis were evaluated by Berner et al. [14] in 1994 on a set of diagnostically challenging clinical cases. As a result, the state-of-the-art technology for computer assisted diagnosis was at the time assigned a C grade by Kassirer [97].

records. Moreover, what may be considered as a “large” collection of data in the medical domain is often considerably smaller than the massive amounts that are sometimes reported in the data mining literature.² In medicine, each case typically equals one clinical encounter, an event with a far lower occurrence frequency than in many other domains. Furthermore, often only a small subset of these encounters are of interest to include in a study. Consequently, many medical KDD studies are often run on existing research databases previously analyzed with traditional statistical methods. In such studies, the data material was often collected manually with aspects of a carefully planned future analysis in mind, e.g., for elimination of possible biases. This in contrast to the more common situation for data mining and KDD, where the data to be analyzed is typically historical data that at the time of collection was not intended to be analyzed.

If retrospective analyses of EMRs form the basis for a KDD experiment, some initial data cleansing is likely to be needed. Since the contents of EMRs vary from patient to patient, unless the scope of the study is sufficiently narrow, not all information may have been recorded for every patient in the study, and one may encounter the problem of missing or incomplete data. Values are also often missing because certain tests have been deemed medically irrelevant to perform, due to the results of previous tests. Very many popular data mining techniques are unable to deal with missing values directly. Furthermore, the data material may often be organized and encoded in quite complex fashions, and the tools required for properly dealing with such data may have to be able to interface with CMTs or other sources of ontological information. Few data mining techniques are currently able to cope with such information directly.

With regards to data management, most would agree that medical information concerning a specific patient is private and confidential. Many abuses of sensitive medical data can easily be hindered through simple means such as access control mechanisms and audit trails. But medical data often need to be disseminated for a wide variety of reasons, for instance to be analyzed by KDD practitioners. Although personally identifying information may have been stripped from the data material prior to release, the data may in principle be cross-referenced against other databases, and thus potentially enabling individuals to be re-identified.³ Practitioners of KDD in the medical domain should therefore be careful about distributing medical data or any results from the KDD analysis, since the consequences for the patients involved might prove disastrous if such information is abused. Lastly, from an ethical point of view, medical data should be treated with care and respect, as behind each piece of data may lie human suffering.

²To get a feel for the magnitudes involved, clinical databases with a few thousand records (or even fewer) may very well be said to be “large”, depending on the medical context. In contrast, Fayyad et al. [54] talk about multigigabyte databases with millions of records being commonplace.

³As demonstrated by Sweeney [215], such re-identification is often surprisingly easy to do. Simple demographic or medical information often suffices. Woodward [243] gives several examples of how medical information has been abused due to improper disclosure and cross-referencing against other databases.

2.4.2 On Model Induction and Selection

Occam's razor, also known as the principle of parsimony, is often used as a guide for model selection in the data mining field. In a medical setting, the parsimony of mined models ought to be viewed in the context of costs, both in the economical sense as well as in the abstract sense. Since not all information is equally cheap to obtain, it is wise to view model parsimony in light of the cost of model realization, and not only in terms of model complexity. For example, for the same task, a classifier based on five pieces of readily available clinical information would probably be preferable to a classifier based on two pieces of information that involve invasive medical procedures, unless the latter displayed significantly better performance.

Many challenging diagnostic problems in medicine involve relatively rare diseases or outcomes. As a result, heavily skewed class distributions in the data material might be observed. While popular in the data mining field, induction of classification rules that reflect normal or default relationships between symptoms and diseases or outcomes may not always be fully appropriate in medicine. This because it may indeed be the few, rare cases and not the possibly overwhelmingly prevailing normal situation that comprise what is medically interesting and hence of interest to be captured by the model. Several of the most popular techniques for inducing classifiers experience, in their basic form, problems with detecting rare events.

The computational complexities involved in machine learning are often staggering. Hence, in order to keep the problems tractable, simplifying assumptions are made, sometimes implicitly. Some of the most typical assumptions are clearly unrealistic in a medical context, but may still be necessary to impose for pragmatical reasons. For diagnostic problems, three of the most common simplifying assumptions are:

- *The set of possible diseases are mutually exclusive and exhaustive.* As for exhaustiveness, medicine is a dynamic science with an ever-expanding body of knowledge, and it would be impossible to list everything. The mutual exclusivity hypothesis is clearly not valid, as counterexamples are easily found, e.g., hyperglycemia and diabetes, and AIDS and kaposi sarcoma.
- *Clinical observations are independent of each other when conditioned on disease.* This means that if we fix the disease status, the knowledge of one finding or observation does not influence our knowledge of another finding.

$$\Pr(\text{Finding}_1 \mid \text{Finding}_2, \text{Disease}) = \Pr(\text{Finding}_1 \mid \text{Disease}) \quad (2.1)$$

This assumption eliminates the need to have to consider all possible interactions between findings, a number which grows dramatically with the number of findings. For a counterexample that invalidates this hypothesis, consider the diagnosis of left-sided valvular heart disease. Assuming conditional independence between observations of systolic and diastolic heart murmurs will lead to erroneous conclusions [219].

- *The order of tests and findings does not matter.* This assumption enables one to not have to consider all permutations of tests. However, radiologists often base their interpretations on findings reported by primary care physicians, and the time lag between tests is often of importance.

Szolovits and Pauker [219] discuss these assumptions in more detail.

2.4.3 On Model Assessment and Deployment

Internally, most classifiers realize a two-stage process when presented with an item x to classify. Performance evaluation can be carried out along at least two axes, depending on which of these stages we examine:

1. For each possible class X_i , a numerical value $0 \leq \phi_i(x) \leq 1$ is estimated. The value of $\phi_i(x)$ reflects the degree of confidence or certainty the classifier has in x belonging to X_i .
 \Rightarrow *How close is $\phi_i(x)$ to the true probability of x belonging to X_i ?*
2. On the basis of the $\phi_i(x)$ values, a class X_j is selected as the class to which x should be assigned.
 \Rightarrow *How good is the classifier at selecting the correct outcome class?*

These two issues are called *calibration* and *discrimination*, respectively. The issue of calibration is largely ignored in the machine learning literature, and may for some applications not be very important. However, if a classifier is to be used in an interactive decision-support setting, then the issue of calibration ought to be considered. Even though a model has been induced that discriminates well, that is not necessarily how it might be employed in practice. For example, in a medical setting, a classifier may have been trained to determine whether or not a patient will survive or die. But the actual use of the model by a decision-maker might be to use the certainty coefficient associated with the “death” outcome to aid in deciding if a patient should be transferred to an intensive-care unit (ICU). Such a decision may not only depend on the certainty coefficient, but also on external factors such as the current scarcity of ICU beds. Clearly, a decision-maker will be more comfortable with a tool that produces a probability rather than some abstract index of certainty. Not only because probability is a familiar concept for humans, but also because other analytical frameworks for decision-making take probabilities as inputs. The latter point may be important in practice, since in a full-fledged decision-support environment the value of $\phi_i(x)$ is likely to be considered in combination with costs, patient preferences or utility factors in general. Tools that combine $\phi_i(x)$ with such factors will almost surely expect $\phi_i(x)$ to denote probabilities.

New or unconventional data analysis techniques need time to mature and prove their worth before becoming generally accepted. As part of this process, in order to gain

acceptance and generate the necessary impact, a great deal of empirical studies have to be carried out. However, when evaluating these models, it is imperative to do so according to well-established performance measures in the domain. This so they can be compared with and contrasted to existing and more widely accepted techniques on the same terms. For evaluating classifiers, the data mining and machine learning communities have in the past had an almost exclusive focus on classification accuracy, i.e., the probability of the induced classifier arriving at a correct classification. Medicine has since long employed a more refined spectrum of evaluation measures, since it is not uncommon that one type of classification error might potentially cost lives, while other types of errors are insignificant in comparison. Discriminatory classifier performance in medicine should therefore be evaluated using measures that handle skewed class distributions and error costs, important issues which classification accuracy as a measure does not reflect very well.

Even though there are numerous papers in the medical informatics literature where excellent results are reported using inductive learning techniques, extremely few of these results are subsequently deployed in any real-world clinical applications. One possible way of defining if an induced model is successful, is to see if it is actually deployed outside a pure research setting. There seems to be an inverse relationship between the number of seemingly successful models and the distance between the point of model deployment to the physical patient: The further away from the point of care a model's location of deployment is, the more likely it is that it will be used and thus defined to be a success. This can in part be attributed to the fact that the number of complicating factors grows dramatically the closer to the point of care one gets. For this reason, most successfully deployed DSSs operate on very specific and narrowly defined problems and in environments where data is either already easily electronically accessible, or in situations where a certain amount of manual data entry can be tolerated. Success factors are not only technical, issues regarding work-flow integration, organizational issues and hospital policies play a very important role. Therefore, a system that is successful in one healthcare institution may not necessarily be equally successful in another.

A point that is often emphasized in the context of medicine is that computer models should offer some means of explaining the reasoning behind their decision-making, e.g., for reasons of user validation and acceptance. Types of models used in machine learning vary in their degree of opaqueness, i.e., in how well one can obtain such explanations through inspecting and interpreting the model structure. In some situations such information can indeed be obtained, but in practice even such presumably understandable model types as rule sets or decision trees may often be hard to interpret. This can, e.g., be because one needs to understand how large collections of firing and possibly conflicting rules are combined in order to yield a decision, or because the path leading to a leaf node in a tree leads to an overly complex rule. Another circumstance that may complicate the picture is if the best-performing model is largely "unintelligible" altogether. Rather than abandoning models that are hard to interpret, we may as an alternative choose to consider case-based explanations instead of model-based ones. That is, instead of letting the models themselves provide explanations, we consider explanations that come in the form of sets of cases from the data

material the models were induced from. A new case or example is then explained in terms of a small handful of old cases that are somehow “close to” the new case, where the distance metric on cases is extracted from the model.⁴ Caruana et al. [28] show how neural network models can be employed to enable case-based explanations to be made.⁵ Case evaluation is frequently emphasized in medical training and practice, and medical practitioners are therefore often proficient at understanding explanations provided in the form of cases [28].

Lastly, if an induced model is deployed in a clinical setting, care should be made that the patient population the model is applied to is “compatible” with the population the model was induced from, i.e., that they are drawn from approximately the same distribution. Otherwise, unless the model can be shown to be very robust, results might not be as expected. Ensuring such population compatibility can be difficult, as population profiles may tend to “drift” over time.

⁴Of course, for this to be possible the training data must be available, a requirement that the model-based explanation scheme does not impose. Also note that if we provide the distance metric directly and bypass the model induction, we end up with the well-known k -nearest neighbor method [179].

⁵How rule-based models can be used to a similar effect will be outlined in Section 6.4.2.

Chapter 3

Rough Sets in Medicine

3.1 Introduction

This chapter concerns itself with rough sets in medicine: Its appeal, drawbacks and previous uses. In Section 3.2, the literature is reviewed and an overview is given of previous applications of rough set theory in the medical domain. Then, Section 3.3 outlines some of the points that make the methodology attractive for practical use, both in general in the context of medical informatics. Finally, Section 3.4 discusses some aspects that one is likely to encounter in practice, and comments on some of the issues raised in Section 3.2 and Section 3.3.

3.2 Literature Review

Various applications of rough set theory to medical data are described in the literature. Mostly in the data mining and rough set literature, but also to a certain degree in the medical informatics literature. Several of these studies simply apply rough set methods to data that happen to have a medical origin, without much regard to the underlying medical problem at hand. Even though such purely syntactical studies may be interesting from a technical standpoint, they, as discussed in Chapter 2, are unlikely to have any significant practical impact or awake much clinical interest, since they make little or no semantical considerations nor consider the problem context.

However, several interesting studies are worth reporting. Broadly speaking, previous applications of rough sets in medicine can be classified into two categories, *diagnosis and outcome prediction* and *feature selection*, with an overwhelming majority of papers falling into the former.

3.2.1 Diagnosis and Outcome Prediction

The by far most common application of rough sets in medicine is for diagnosis or prediction of outcomes. This is usually accomplished by synthesizing if-then rules.

Fibak et al. [56] applied rough sets to analyze a database of patients with duodenal ulcer treated by highly selective vagotomy¹ (HSV) at a hospital in Poznań, Poland. From preoperative information, the goal was to predict the long-term success of the operation, as evaluated by a surgeon into four outcome classes. The analysis was done with discretization being carried out using expert medical knowledge, and a semi-manual search for minimal approximate discerning attribute sets. As a result, descriptive models of each outcome class could be formulated in natural language, and a small set of decision rules was also synthesized. Such descriptive models are important for screening purposes, since one can potentially utilize indications for treatment to maximize the number of successful operations. Indeed, the HSV study by Fibak et al. is one of few data analysis studies, regardless of methodology, that has managed to cross the clinical deployment barrier. Słowiński [197] reports that the developed prediction models were consulted for screening a group of subsequent candidates for surgery, with the result of the group accepted for surgery having a clearly more advantageous distribution among the outcome classes than the group from which the models were originally derived. Other papers pertaining to the HSV study include [157, 198, 200, 201].

Vinterbo et al. [231] report on an experiment where a rough set predictor for myocardial infarction was synthesized using data collected at an emergency room in Edinburgh, Scotland, and evaluated on data collected at another emergency room in Sheffield, England. The behavior and stability of classifiers across site boundaries is of great practical interest, and this issue had been previously investigated by others [99, 221] using other methods on the same sets of data. Both these studies reported performance losses when crossing site boundaries, while Vinterbo et al. conclude that the rough set predictor seems to generalize well across sites.

Rowland et al. [181] compared neural networks, logistic regression and rough sets for the prediction of ambulation following spinal cord injuries. All methods performed well, and although the performance of the rough set model was marginally lower than the other models, Øhrn et al. [149] demonstrated that the rule sets could be pruned down to very small and manageable sizes with a very low penalty in performance loss. In some simulations, the models were pruned down to as little as five simple decision rules. Obtaining compact models is important for interpretability.

Breast cancer is the cancer type that most commonly causes death for women in the United States. Wojcik and Ziarko [241] used a rough set approach to analyze a database on women with breast cancer, and induced prognostic rules for determining short-term and long-term follow-up survival. The rules are subsequently analyzed and interpreted from a clinical perspective, and although in general the rules were found

¹Highly selective vagotomy is a surgical procedure which consists of vagal denervation of the stomach area secreting hydrochloric acid.

to support well-known established facts about factors influencing survival, new insight was reported gained on the impact some medical and social factors may have on long-term survival patterns.

Taking a similar approach as in the previously described HSV study, Słowiński et al. [202] construct rough approximations to analyze a small set of data from peritoneal lavage² in acute pancreatitis. The goal of the study was to determine which subsets of pre-lavage attributes that best determine various outcomes, e.g., the required length of the treatment period. Their analysis showed that a significant portion of the information was redundant. A similar study with a larger set of patients was later done by Słowiński and Sharif [199] where peritoneal lavage was employed as a diagnostic aid applied to patients with multiple injuries. Decision rules were synthesized, from which concrete clinical conclusions could be drawn.

Woolery and Grzymala-Busse [244] report on the development of a prototype expert system for assessing preterm birth risk. The goal of the system was to predict preterm deliveries, using a knowledge-base of rules extracted from empirical data by means of a machine learning system based on rough sets. The prototype expert system was verified to be more accurate than traditional manual techniques in predicting preterm birth.

Induction of rules by means of rough sets for subsequent implementation in expert systems has also been investigated by Tsumoto [223], then in the context of diagnosis of congenital malformations. The resulting expert system was evaluated in clinical practice, and found to yield as good performance as a medical expert.

Several other medical applications of rough sets for predictive modelling are reported in the literature, e.g., for screening patients for extracorporeal shock wave lithotripsy [203, 208], diagnosis of progressive encephalopathy in children [237], analysis of structure-activity relationships in pharmacology [109, 110], diagnosis of acute appendicitis [26, 27], risk assessment of developmental toxicity [78], differential diagnosis of headache [222, 224], prediction of heart attacks [63], diagnosing diabetes in children [209, 210], and determining the maturity status of newborn children [30].

3.2.2 Feature Selection

The studies reported in Section 3.2.1 all employ rough sets as the main processing methodology for diagnosis or outcome prediction. However, another use is to employ rough sets as a preprocessing step for other machine learning techniques. The primary application for this is feature selection, i.e., using the concept of minimal discerning subsets to reduce the number of input dimensions for other modelling schemes. Of course, the modelling schemes employing the selected feature subsets should be flexible enough to incorporate at least some of the strong nonlinearities that the reduced modelling problems are likely to exhibit.

²Peritoneal lavage for acute pancreatitis consists of introducing a catheter into the peritoneal cavity and instilling fluid which is subsequently drained, flushing out toxic materials. This procedure can be repeated in multiple stages.

Jelonek et al. [90] used rough sets to select attributes for classification of histological images with neural networks. The goal was to reduce the number of inputs to the neural network so that the training time of the network could be lowered. From a set of images of cells from seven types of brain cancer, a large number of features were extracted from each image. Through combining medical expertise with processing by rough sets, the dimensionality could be reduced to about 11% of the original set. Cross-validation experiments showed that a neural network using only this reduced set of attributes performed slightly better than a network using the full set of inputs. In a follow-up study [91] using alternative image features, a collection of an even larger set of features was reduced to about 4% of the original set, using fully automatic rough set feature selection. Acceptable performance was maintained.

Dreiseitl et al. [49] employ a wide variety of techniques, including rough sets, for selecting relevant features for prediction of myocardial information. The selected feature subsets were validated and visualized with self-organizing maps.

Stability across data from different geographical sites is not the only robustness issue of interest for data-driven modelling, temporal stability is of importance as well. Kandulski et al. [96] focus on synthesizing rough set approximations for establishing a hierarchy of known risk factors for surgical wound infection. Applying the technique on data gathered within a timespan of four years, they report that the hierarchy of features selected on the basis of the first two years of data appears to be similar to that obtained when considering only data from the last two years.

3.2.3 Miscellaneous

There are extremely few papers where rough sets have been applied in the medical domain that do not fit into one of the two outlined mentioned categories. Some noteworthy exceptions exist, though. Øhrn et al. [151] and Komorowski and Øhrn [104] describe an application of rough sets for *feature extraction* in cardiology. By feature extraction is meant the construction of new features from combinations of existing features. Øhrn and Ohno-Machado [148] describe how Boolean reasoning can be employed for *anonymization* of sensitive medical data by means of cell suppression, and provide interpretations of the procedure in terms of rough set ideas. By anonymization is meant a transformation of data such that re-identification of individuals is hampered. These applications are presented in detail in Chapter 11 and Chapter 12, respectively.

If we extend the scope under consideration, some studies can be found in the literature that deal with discretization of medical data with the aim of applying rough sets to analyze the discretized data. Wakulicz-Deja et al. [236] discuss discretization in the context of diagnosing mitochondrial encephalomyopathies, while Słowiński and Słowiński [200,201] investigate how sensitive the HSV study outlined in Section 3.2.1 is to alternative discretizations.

3.3 Appeal

Discernibility and rough set theory constitute an appealing framework for machine learning in medicine for several reasons. Some of the reasons for this appeal are purely technical, while others relate to some of the issues raised in Section 2.4. Furthermore, some of the reasons are shared among several other modelling paradigms, while others are specific to rough sets.

- A guiding philosophy of rough set theory is to let the data material speak for itself. As such, very few assumptions are made about the data, attributes need only have some notion of inequality defined on their domains. In particular, rough set theory does not make assumptions about statistical distributions of the data, nor does external information such as, e.g., membership functions have to be supplied.
- In domains where model deployment involves humans, the purpose of data mining and KDD is just as often to construct interpretable descriptions of the data as it is to build well-performing classifiers. Not only for so that individual classifications can be “explained”, but also to gain insight into the nature of the data. The output of a rough set analysis is usually a collection of if-then rules. An if-then rule is, arguably, as close to a model in natural language as one might expect to obtain, and can be read and interpreted by personnel without expertise in the actual model induction technique.
- Via Boolean minimization techniques, functionally superfluous information can be discarded as part of a rough set analysis. This effectively provides a mechanism for focusing attention on the nature of the classification task at hand. Furthermore, not only may the methodology eliminate complete variables or attributes altogether, but functionally superfluous information within the remaining attributes can also often be discarded. This enables highlighting of such information that, e.g., only low values for an attribute are relevant for classification, while medium and high values do not matter.
- Construction of the knowledge-base is commonly perceived as a major bottleneck in building rule-based expert systems. As rough sets can be used to automatically induce if-then rules from empirical data, this offers the possibility to automate, at least in part, the knowledge acquisition stage when developing such systems.
- Rough set theory is directly equipped to handle inconsistent or seemingly conflicting examples in the data material. Inconsistencies may occur due to, e.g., transcription errors, subjective determination of attribute values or outcomes, lack of information, or if the mutual exclusivity assumption from Section 2.4.2 is violated.
- The theory of rough sets can handle any finite number of outcome categories, and not just dichotomous outcomes.

- Certain types of important background information are possible to incorporate in the modelling phase, if available. For example, information about attribute costs can be embedded in the process of searching for functional dependencies or minimal discerning subsets. Furthermore, extensions of standard rough set theory have the ability to handle missing values and taxonomies or hierarchies among attribute values.
- Since discernibility-based methods are logically based, the problem of having cases or examples that belong to very rare categories numerically “drown” in the remaining data can be controlled or even avoided.

The suitability of rough sets applied in the medical domain has been investigated by, e.g., Tsumoto [222–226], who discusses some characteristics of medical reasoning and argues that rough set representations of diagnostic models is a useful approach to extracting medical knowledge from databases. Tsaptsinos and Bell [220] argue along similar lines.

3.4 Discussion

Section 3.3 lists several appealing characteristics of rough sets for data analysis, and Section 3.2 provides several examples of the methodology having proved its usefulness in medicine. However, a few comments are in order, some of which are rough set specific and some of which are of a more general nature, as discussed in Section 2.4.

3.4.1 On Prerequisites and Assumptions

An implicit assumption underlying the applicability of most techniques for data analysis, is that data is available in electronic form, and that this can be represented in tabular format. If the data does not lend itself to such a flat representation or the problem cannot be expressed as a classification problem, alternative methods than rough sets may be preferable.³ Furthermore, a great deal of ingenuity might be required to represent certain types of data, e.g., temporal data,⁴ in a suitable format.

Several of the appealing features listed in Section 3.3 can be traced back to that so few assumptions about the data are made. However, this is a double-edged sword, since there might equally well be information available that is not being made use of, e.g.,

³In theory, a rough set only requires a binary indiscernibility relation to be defined on the universe of objects, and does in itself not directly rely on a tabular representation of data. In practice, however, this binary relation is always constructed on the basis of discernibility considerations derived from flat data tables.

⁴Approaches to this include devising attributes that encode time lags, first and second time derivatives of attributes evaluated at the time of data acquisition, and using Boolean state attributes that denote whether or not some event has happened at the time of recording the current data point. Yet another approach is to convert complete time evolutions into time-invariant attributes such as Fourier coefficients or regression coefficients of polynomials in t .

in the form of metrics or information about dependencies or attribute semantics. Also, since rough set analysis is data-driven, incorporation of any existing domain theories as background knowledge can be difficult. Two ways to incorporate at least some background knowledge is to extend the data table with hand-crafted problem-specific attributes or features that are combinations of more primitive attributes, or to overload or redefine what is really meant by “discernibility”.

For methods based on discernibility, it is important that an appropriate degree of coarseness is employed when viewing the world. Otherwise, a bias will be made towards selecting many-valued attributes. In concrete terms, this means that for ordinal attributes some kind of preprocessing is likely to be needed. For attributes taking on numerical values, this may translate into defining intervals or bins rather than operating on the original numerical data. A wide variety of algorithms exist for automatically computing the appropriate ranges of these intervals, some of which are also based on discernibility considerations. A rough set analysis might be sensitive to different discretizations.

Proponents of rough sets often argue that the method is objective due to the philosophy of letting the data material speak for itself. This is only true to the extent that the data itself is objective, something it need not necessarily be. For example, a dichotomous attribute may be the result of having binarized a vague concept using a subjectively chosen threshold. This issue is further discussed by Koczkodaj et al. [102].

3.4.2 On Interpretation and Deployment

A rough set analysis may reveal functional or near-functional dependencies between attributes. However, functional dependencies do not necessarily reflect causal relationships.⁵ Whether they do or not is something that can only be decided by a domain expert. Moreover, a model induced by a rough set analysis is the result of a purely syntactical process, and domain expertise is required to assess the semantical bearings of the model.

Although a single rule may in itself be readable and interpretable, it is easy to lose sight of the overall rule-based model if the size of the rule set is too large. Hence, in practice, rule-based models can often be said to be opaque rather than white-box when it comes to the model as a whole. For work on taming large rule sets, see, e.g., [4,5,149]. Strong model components, i.e., individual rules, however, can often be extracted and interpreted simply by sorting the rules according to some measure of goodness, and viewing only the highest-ranking rules. Of course, the interpretability of a single rule depends on the length of its antecedent. As reported by Lavrač [113], medical doctors often tend to prefer rules that are of medium complexity since too short rules, even though they may be strong, may not correlate enough information for them to be of interest for KDD purposes. Tsumoto [222,224] also discusses this aspect.

⁵The same comment applies to other measures of correspondence, e.g., statistical correlation. A famous example illustrating the difference between correlation and causality is the fact that the position of the hands of all clocks are correlated, without one clock being the cause of the position of the others. Another example is the significant correlation between human birth rates and stork population sizes [18].

Many papers on applications of rough sets emphasize that the models produced are inspectable, yet still only use the model as a black-box classifier and do not present any actual rules verified by a domain expert. The reasons for this may be many, the perhaps dominating one being that the persons performing the analysis and the persons that provide the data are seldom the same. Hence, the domain expertise required to be able to perform such a semantical verification may be lacking. Extremely close cooperation between data analysts and domain experts is required for actual knowledge discovery to be done.

Flat data tables can in many ways be said to describe a kind of “one-shot” approach to classification, i.e., where a whole set of observations are collected before a classification is made. Viewing a classification as a diagnosis, this is in contrast with how medical diagnoses are typically made in practice, where arriving at a diagnosis is an iterative and serial decision-making procedure. By this is meant that a diagnosis is often really eventually “converged upon” after acquiring a series of tests in a certain order, and deciding on whether to order more tests in order to strengthen or weaken the current hypothesis. If the chosen machine learning technique has the ability to discard functionally redundant information, some of this seriality may possibly be recovered from the data table. Decision trees and methods that induce if-then rules are examples of such methods.

Rule-based expert systems usually reach conclusions through forming chains of deduction, while if-then rules as output from a rough set analysis relates multiple conditions to a single decision. The decision or outcome in one table may be a condition in another table, so rule induction may have to be performed repeatedly for different decisions if the rules are to be employed in expert systems that support chaining. Otherwise, the induced rules can only form a single link in a reasoning chain.

If-then rules output from a standard rough set analysis relate only positive knowledge, and do not involve negations. Rather, a negated propositional fact is represented as an exhaustive listing of the other possible values for the attribute in question. However, this is partially a matter of data representation and model presentation. If knowledge of the set of values an attribute might take on is employed, a collection of rules can be massaged to incorporate negations before being presented to the user.⁶ Negations can also be obtained in retrospect by considering the contrapositives of rules. This logical equivalence is explored by Tsumoto [223].

3.4.3 On Model Maintenance and Assessment

If more data becomes available after a model has been synthesized, it would be desirable to be able to update the model incrementally such that the new training data is also taken into account. Many common data mining methods simply re-synthesize a model from scratch using all available data. Not only may this be time-consuming, but also assumes that the original training data is available in the first place. Research related to rough sets and incremental learning include work by Tsumoto and

⁶This will be further discussed in Section 8.4.1.

Tanaka [227] and Shan and Ziarko [188].

Traditionally, accuracy or error rate has been the dominating measure for assessing classificatory performance in the machine learning literature. In medicine, ROC analysis is the prevailing measure of discriminatory performance for dichotomous outcomes. The number of studies reported in the literature where rough set predictors are evaluated through an ROC analysis are few, but on the rise.

Part II

Preliminaries

Chapter 4

Boolean Reasoning

4.1 Introduction

In 1854, George Boole [17] set out to develop a framework in which the laws of thought could be formalized, with the ultimate goal of reducing human reasoning to calculation. This seminal work in modern symbolic logic introduced several techniques and concepts that in the past 150 years have been refined and further developed by logicians and mathematicians into the rich body that the topic constitutes today.¹ Although Boole's equation-based logic has been widely abandoned by 20th century logicians in favour of the more powerful predicate calculus, parts of his original framework is still very useful for practical applications.

Boolean reasoning builds on the equation-based Boole-Schröder algebra of logic rather than on the predicate calculus, and is an extremely versatile tool with a wide range of applications. This chapter only briefly touches upon the topics needed in connection with rough set theory. The interested reader is referred to Brown [21] for a more detailed exposition.

4.2 Boolean Algebras

A *Boolean algebra* is a quintuple $(\mathbf{B}, +, \cdot, 0, 1)$ where the four postulates given below are satisfied. The set \mathbf{B} is called the *carrier set*, and “+” and “ \cdot ” are binary operations on \mathbf{B} . The elements 0 and 1 are distinct members of \mathbf{B} . A Boolean algebra is often denoted in shorthand by its carrier set alone.

1. *Commutative laws:* For all $a, b \in \mathbf{B}$, $a + b = b + a$ and $a \cdot b = b \cdot a$.
2. *Distributive laws:* For all $a, b, c \in \mathbf{B}$, $a + (b \cdot c) = (a + b) \cdot (a + c)$ and $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$.

¹On a historical note, many of the proofs that Boole gave to justify his methods are best described as “complete nonsense”. Burris [23] provides a short discussion of this and some examples.

3. *Identities*: For all $a \in \mathbf{B}$, $0 + a = a$ and $1 \cdot a = a$.
4. *Complements*: To any element $a \in \mathbf{B}$ there corresponds an element $a' \in \mathbf{B}$ such that $a + a' = 1$ and $a \cdot a' = 0$.

We define the relation \leq on a Boolean algebra as follows:

$$a \leq b \Leftrightarrow a \cdot b' = 0 \quad (4.1)$$

The relation \leq is a *partial order*. Partial orders, conveniently depicted in *Hasse diagrams*,² have three basic properties:

1. *Reflexivity*: For all $a \in \mathbf{B}$, $a \leq a$.
2. *Antisymmetry*: For all $a, b \in \mathbf{B}$, if $a \leq b$ and $b \leq a$ then $a = b$.
3. *Transitivity*: For all $a, b, c \in \mathbf{B}$, if $a \leq b$ and $b \leq c$ then $a \leq c$.

By the representation theorem of Stone [212], all Boolean algebras with a finite carrier set are isomorphic to the Boolean algebra of subsets $(2^A, \cup, \cap, \emptyset, A)$ of some finite set A . Because the relation \leq in a Boolean algebra \mathbf{B} corresponds to the relation \subseteq in the subset-algebra isomorphic to \mathbf{B} , \leq is called the *inclusion relation*.

4.3 Implicants

An m -variable function $f : \mathbf{B}^m \rightarrow \mathbf{B}$ is called a *Boolean function* if and only if it can be expressed by a Boolean formula. An *implicant* of a Boolean function f is a term p such that $p \leq f$. In other words, an implicant of f is any conjunction of literals (variables or their negations) such that, if the values of these literals are true under an arbitrary valuation v of variables, then the value of f under v is also true. Any term of a sum-of-products (SOP) formula for f is clearly an implicant of f .

A *prime implicant* is an implicant of f that ceases to be so if any of its literals are removed. An implicant p of f is a prime implicant of f in case, for any term q , the implication below holds.

$$p \leq q \leq f \Rightarrow p = q \quad (4.2)$$

The disjunction of all prime implicants of f is called *Blake's canonical form* and is denoted $BCF(f)$. In general, the task of computing all terms of $BCF(f)$ belongs to the

²In general, if \leq is a partial order on a set \mathbf{B} , we construct a Hasse diagram for \leq on \mathbf{B} by drawing a line segment from a up to b , if $a, b \in \mathbf{B}$ with $a \leq b$ and if there is no other element $c \in \mathbf{B}$ such that $a \leq c$ and $c \leq b$. If we adopt the convention of reading the diagram from bottom to top, then it is not necessary to direct any edges.

NP-hard class of problems [62]. For large and complex functions f , therefore, approximation algorithms and heuristics that search for individual terms of $BCF(f)$ are in order.

4.4 General Scheme

Following the presentation of Brown [21], the general scheme of applying Boolean reasoning to solve a problem P can be formulated as follows:

1. Encode the problem P as a system of simultaneously-asserted Boolean equations.

$$P = \begin{cases} g_1 = h_1 \\ \vdots \\ g_k = h_k \end{cases} \quad (4.3)$$

The g_i and h_i are Boolean functions on \mathbf{B} .

2. Reduce the equation system to a single Boolean equation³ $f_P = 0$.

$$f_P = \sum_{i=1}^k (g_i' \cdot h_i + g_i \cdot h_i') \quad (4.4)$$

3. Compute $BCF(f_P)$, the prime implicants of f_P .
4. Solutions to P are obtained by interpreting the prime implicants of f_P .

This general scheme defines a powerful and versatile tool for problem solving, with an infinite array of possible applications. In the field of machine learning this procedure may crop up as a subcomponent in algorithms for rule induction, clustering, feature extraction and discretization of real-valued attributes.

Ledley and Lusted [114] emphasize Boolean reasoning as one of the foundations of medical diagnosis, together with probability and value theory. They state the logical problem of medical diagnosis as follows:⁴

The logical aspect of the medical diagnosis problem is to determine the diseases f such that *if medical knowledge E is known, then: if the patient presents symptoms G , he has diseases f* . In terms of our symbolic language, the problem is to determine a Boolean function f that satisfies the following formula:

$$E \rightarrow (G \rightarrow f)$$

³As discussed by Brown [21], Boole and other 19th century logicians based symbolic reasoning on equations in 0-normal form, i.e., $f_P = 0$. The equivalent 1-normal form, i.e., $f_P' = 1$, may also be used.

⁴The notion that a implies b is typically written as " $a \rightarrow b$ ", and is semantically equivalent to " $d + b$ ".

This is the fundamental problem of medical diagnosis. That this is truly the diagnosis in an intuitive sense can be readily seen. For it is easy to show that the fundamental formula can be equivalently written as

$$E \rightarrow (f' \rightarrow G')$$

which means in a sense that if the diseases f are cured, then the patient's symptoms will disappear. It can be shown that a solution f always exists.

Chapter 5

Discernibility and Rough Sets

5.1 Introduction

This chapter introduces the rough set theory of Pawlak [154, 155] and some of its theoretical foundations. Several of the most practically important concepts in rough set theory can be conveniently interpreted in the realm of Boolean reasoning, and Boolean reasoning techniques are typically applied to solve minimization problems in rough set theory, as shown by Skowron and Rauszer [192, 196].

The exposition in this chapter is slightly non-standard in a couple of ways. The indiscernibility relation, the fundamental building block of rough sets, will be defined in terms of a structure called a discernibility matrix. This in contrast to how the material is typically presented in the literature, where discernibility matrices are mostly considered as a tool for solving minimization problems wrt. the indiscernibility relation. Furthermore, the exposition in this chapter has been generalized to a level that encompasses most popular extensions of standard rough set theory.

The literature is rich with papers that link rough set theory to other theoretical frameworks in computer science and mathematics. However outside the scope of this chapter, these works include connections to Dempster-Shafer evidence theory [193], fuzzy set theory [50, 152], modal logics [247] and mathematical morphology [195], to name a few.

5.2 Information Systems

The basic vehicle for data representation in the rough set framework is an *information system*. An information system is in this context a single flat table, either physically or logically in form of a view across several underlying tables. We can thus define an information system \mathcal{A} in terms of a pair (U, A) , where U is a non-empty finite set of *objects* and A is a non-empty finite set of *attributes*.

$$\mathcal{A} = (U, A) \quad (5.1)$$

Each attribute $a \in A$ can be viewed as a function that maps elements of U into a set V_a . The set V_a is called the *value set* of attribute a .

$$a : U \rightarrow V_a \quad (5.2)$$

The value of attribute a for object x is said to be *missing* if $a(x)$ has not been observed. Values may be missing for a variety of reasons. How missing values should be interpreted and subsequently treated depends on the application domain. In the following, “ \top ” will be used to denote a missing value, and is assumed to be a member of every value set.

5.2.1 Discernibility Matrices

An information system \mathcal{A} defines a matrix M_A called a *discernibility matrix*. Each entry $M_A(x, y) \subseteq A$ consists of the set of attributes that can be used to discern between objects $x, y \in U$:

$$M_A(x, y) = \{a \in A \mid \text{discerns}(a, x, y)\} \quad (5.3)$$

Typically, the *discerns/3* predicate is defined as ordinary inequality on the attribute values as defined in Equation 5.4, but the definition of the *discerns/3* predicate can be tailored to the application at hand.

$$\text{discerns}(a, x, y) \Leftrightarrow a(x) \neq a(y) \quad (5.4)$$

Conceptually, M_A is an $|U| \times |U|$ matrix. More pragmatically, we only need to consider pairs of *distinct* objects during its construction. Also note that if *discerns/3* is symmetric in x and y as well as reflexive, then $M_A(x, y) = M_A(y, x)$ and $M_A(x, x) = \emptyset$ for all x, y . In practice, this means that we can get away with computing less than half the matrix entries when constructing M_A .

Defining Discernibility

Overloading the notion of discernibility on a per attribute basis is one way of letting background domain knowledge enter into a discernibility-based data analysis. This is explored in the following. Note, however, that tailoring the definition of discernibility may have consequences wrt. properties of relations derived from M_A . This will be further discussed in Section 5.2.2.

In some applications the elements of an attribute's value set can be hierarchically organized, e.g., according to an "is-a" relation. We can model such a situation by a partial order \preceq_a on V_a , where $v_1 \preceq_a v_2$ is to be read as " v_1 is an instance of v_2 " or " v_2 is more general than v_1 ". It then seems natural to define that an attribute can only be used to discern between two objects if their corresponding attribute values do not form an inclusion:

$$\text{discerns}(a, x, y) \Leftrightarrow a(x) \not\preceq_a a(y) \text{ and } a(y) \not\preceq_a a(x) \quad (5.5)$$

To gain an intuitive understanding of Equation 5.5, consider the example from the medical domain found in Figure 5.1. As discussed in Section 2.3.1 and Section 2.4.1, medical data on electronic form is often encoded using CMTs. Some of these CMTs define hierarchies, at least in principle. One such CMT is the ICD-9-CM system [59], which assigns three-digit numerical codes to diagnoses, possibly with a fourth modifier digit. Generally, the code prefixes define gradually more specific families or classes of diagnoses. The fragment depicted in Figure 5.1 could be used if attribute a assigned ICD-9-CM diagnosis codes.

Equation 5.5 can be reduced to a plethora of special cases. A *discrete* partial order is one in which no two distinct elements are comparable. If \preceq_a is discrete, Equation 5.5 reduces to the more traditional Equation 5.4. If a discrete partial order is extended with \top as an element that is "more general than" all other elements, we obtain Equation 5.6, which can be viewed as a simple way of handling missing values [111, 112].

$$\text{discerns}(a, x, y) \Leftrightarrow a(x) \neq a(y) \text{ and } a(x) \neq \top \text{ and } a(y) \neq \top \quad (5.6)$$

In a discernibility-based data analysis, the traditional way of handling an attribute with a value set that is totally ordered is to somehow partition its original value set into intervals, and to treat this discretized attribute as a categorical variable using Equation 5.4. As an alternative to this discretization process, we can choose to overload the notion of discernibility. Equation 5.7 displays one way of doing this, where we define that an attribute can only be used to discern between two objects if their corresponding attribute values are "far enough" apart, as determined by the parameter r_a :

$$\text{discerns}(a, x, y) \Leftrightarrow |a(x) - a(y)| > r_a \quad (5.7)$$

In general, complete control of the *discerns/3* predicate for an attribute a can be obtained by means of an *indiscernibility definition graph* (IDG). An IDG for attribute a is a directed graph with the elements of V_a as nodes or vertices, and a set of edges $E_a \subseteq V_a^2$.

$$\text{IDG}_a = (V_a, E_a) \quad (5.8)$$

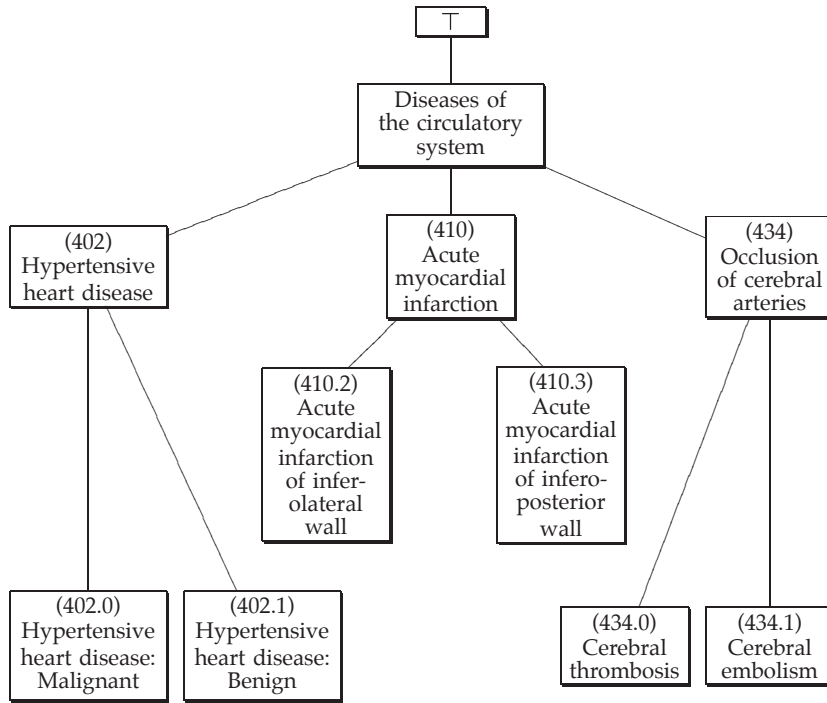


Figure 5.1: An example concept hierarchy from the medical domain, depicting a small fragment of the ICD-9-CM hierarchy. Such partial orders can be used to define the *discerns/3* predicate, as shown in Equation 5.5. For example, this fragment states that we cannot, on the basis of ICD-9-CM diagnosis codes alone, discern between a person x with diagnosis code “410” and a person y with diagnosis code “410.3”, but that we can use the diagnosis code to discern between x and a person z with diagnosis code “434.1”. Note that no information at all, denoted by the missing value symbol “ \top ”, is here defined as a top omnimatching element.

E_a	$discerns/3$
$\{(v, v) \mid v \in V_a\}$	Equation 5.4
$\{(v_1, v_2), (v_2, v_1) \mid v_1, v_2 \in V_a \text{ and } v_1 \preceq_a v_2\}$	Equation 5.5
$\{(v, v), (v, \top), (\top, v) \mid v \in V_a\}$	Equation 5.6
$\{(v_1, v_2) \mid v_1, v_2 \in V_a \text{ and } v_1 - v_2 \leq r_a\}$	Equation 5.7

Table 5.1: An overview of how the set of edges E_a in an IDG translates into various definitions of the $discerns/3$ predicate. By carefully defining E_a , the notion of discernibility can be overloaded on a per attribute basis.

The edges E_a effectively define the $discerns/3$ predicate through Equation 5.9. An edge $(v_1, v_2) \in E_a$ is to be interpreted as that attribute a cannot be used to discern an object with value v_1 from an object with value v_2 .

$$discerns(a, x, y) \Leftrightarrow (a(x), a(y)) \notin E_a \quad (5.9)$$

How Equation 5.9 encompasses all the other definitions of $discerns/3$ can be seen from Table 5.1. Hybrids are of course possible.

When Equation 5.9 is employed, great care should be taken that E_a is correctly specified. At the very least, the IDG should define a reflexive relation. The relation should almost always be symmetric as well. Słowiński and Vanderpooten [207] explore introducing a “directedness” property of discernibility, thus abandoning symmetry.

Lastly, we note that IDGs are flexible enough to also accommodate the situation when there is some uncertainty in the attribute values, e.g., when a can only relate x to a subset of V_a rather than an element of V_a .¹ If we further assign a certainty distribution to the set of possible values for $a(x)$, we obtain an extension reminiscent of that of Słowiński and Stefanowski [206].

5.2.2 Indiscernibility Relations

A discernibility matrix M_A defines a binary relation $R_A \subseteq U^2$. The relation R_A is called an *indiscernibility relation* with respect to A , and expresses which pairs of objects that we cannot discern between.

$$xR_A y \Leftrightarrow M_A(x, y) = \emptyset \quad (5.10)$$

A generalization of Equation 5.10 would be to define two objects to be in R_A if and only if we do not bother to discern between them (even if we can), simply because

¹Technically, we then have 2^{V_a} as the effective value set. The $discerns/3$ predicate might then be implemented as, e.g., a threshold on some measure of set similarity, and \top might be interpreted as V_a .

they do not differ “enough”.²

The properties of R_A vary according to how the *discerns/3* predicate is defined. If *discerns/3* is defined as in Equation 5.4, R_A is an *equivalence relation*. Equivalence relations have three basic properties:

1. *Reflexivity*: For all $x \in U$, xR_Ax .
2. *Symmetry*: For all $x, y \in U$, if xR_Ay then yR_Ax .
3. *Transitivity*: For all $x, y, z \in U$, if xR_Ay and yR_Az then xR_Az .

With alternative definitions of the *discerns/3* predicate such as, e.g., Equation 5.5 or Equation 5.6, transitivity is easily lost. Relations that are reflexive and symmetric but not transitive are sometimes referred to as *similarity relations* or *tolerance relations*. In standard rough set theory the indiscernibility relation is required to be an equivalence relation, but less restrictive extensions of rough set theory do not require the transitivity condition to hold.

The *indiscernibility set* of an object $x \in U$ is denoted $R_A(x)$, and consists of those objects that stand in relation to object x by R_A .

$$R_A(x) = \{y \in U \mid xR_Ay\} \quad (5.12)$$

If R_A is an equivalence relation, then the indiscernibility sets are called *equivalence classes*. Equivalence relations induce a *partition* of the universe, meaning that all equivalence classes are disjoint and their union equals the full universe U . Vice versa, a partition also induces an equivalence relation. In the more general case of tolerance relations, the indiscernibility sets form a covering of U , meaning that the indiscernibility sets are allowed to overlap.

Indiscernibility Graphs

An alternative way to represent R_A is as an *indiscernibility graph*, i.e., a graph G_A with vertices U and edges R_A :

$$G_A = (U, R_A) \quad (5.13)$$

²One way to accomplish this would be to define R_A as follows:

$$xR_Ay \Leftrightarrow \text{cost}(M_A(x, y)) \leq k \quad (5.11)$$

This formulation states that we do not bother to discern between objects that are “cheap enough” to discern between. Note that with $k = 0$, the function *cost* defined as the sum of costs of each member attribute, and with a unit cost for all attributes, Equation 5.11 reduces to Equation 5.10.

Another plausible choice for numerical attributes would be to define *cost* as, e.g., the weighted Manhattan or Mahalanobis distance between x and y wrt. the attributes in $M_A(x, y)$.

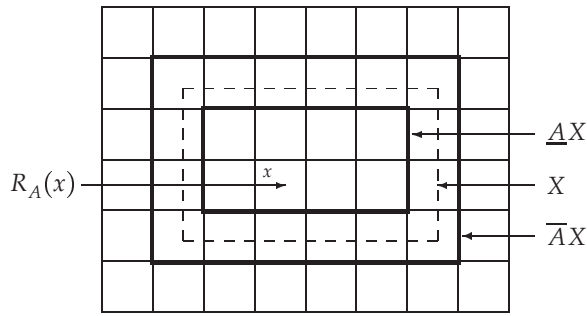


Figure 5.2: We can imagine the indiscernibility relation R_A , here assumed to be an equivalence relation, to define a grid that we overlay our universe U , with each indiscernibility set being displayed as a “pixel” or square in the grid. This grid forms our *approximation space*. The subset of objects $X \subseteq U$ that we want to approximate is drawn as a dashed line that crosses pixel boundaries, and cannot be defined crisply within our approximation space. The lower and upper approximations $\underline{A}X$ and $\overline{A}X$ are drawn as thick gridlines.

G_A is normally only interesting to consider when R_A is a tolerance relation. The graph can then be used for the purpose of clustering or unsupervised learning, with the distance metric being the length of the shortest path between two objects [192].

5.2.3 Rough Sets

The basic idea behind rough sets is to construct approximations of sets using the binary relation R_A . The indiscernibility sets $R_A(x)$ form basic building blocks from which subsets $X \subseteq U$ can be assembled. If X cannot be defined in a crisp manner using attributes A , we can circumscribe them through *lower and upper approximations* $\underline{A}X$ and $\overline{A}X$, defined below.

$$\underline{A}X = \{x \in U \mid R_A(x) \subseteq X\} \quad (5.14)$$

$$\overline{A}X = \{x \in U \mid R_A(x) \cap X \neq \emptyset\} \quad (5.15)$$

The lower approximation consists of those objects that certainly belong to X whereas the upper approximation consists of the objects that possibly belong to X . Note that the upper approximation includes the lower approximation. The *boundary region* is defined as the difference between the upper and the lower approximation, and consists of the objects that we cannot decisively assign as being either a members or non-members of X . The *outside region* is defined as the complement of the upper approximation, and consists of the objects that are definite non-members. A *rough set* is any subset $X \subseteq U$ defined through its lower and upper approximations.

Figure 5.2 displays these ideas graphically.

Rough Membership Functions

The *rough membership function* [156] is a function $\mu_A^X : U \rightarrow [0, 1]$ that, when applied to object x , quantifies the degree of relative overlap between the set X and the indiscernibility set to which x belongs. The rough membership function can be interpreted as a frequency-based estimate of $\Pr(x \in X \mid x, A)$, the conditional probability that object x belongs to set X , given knowledge of the information signature of x with respect to attributes A .

$$\mu_A^X(x) = \frac{|R_A(x) \cap X|}{|R_A(x)|} \quad (5.16)$$

Variable Precision Rough Sets

The formulas for the lower and upper set approximations can readily be generalized to some arbitrary level of precision $\pi \in [0, 0.5)$ by means of the rough membership function, as shown below. Note that the lower and upper approximations as defined in Equation 5.14 and Equation 5.15 are obtained as a special case.

$$\underline{A}_\pi X = \{x \in U \mid \mu_A^X(x) \geq 1 - \pi\} \quad (5.17)$$

$$\overline{A}_\pi X = \{x \in U \mid \mu_A^X(x) > \pi\} \quad (5.18)$$

A rough set X defined through the lower and upper approximations $\underline{A}_\pi X$ and $\overline{A}_\pi X$ is sometimes referred to as a *variable precision rough set* [249, 250], and can be seen as a way of “thinning” the boundary region. Equation 5.17 and Equation 5.18 can easily be generalized to employ asymmetric bounds [98].

5.2.4 Discernibility Functions

A *discernibility function* is a Boolean product-of-sums (POS) function that expresses how an object or a set of objects can be discerned from a certain subset of the full universe of objects.

From a discernibility matrix M_A , we can construct the discernibility function relative to an object $x \in U$ as shown below. The function $f_A(x)$ is a POS function of $|A|$ Boolean variables, where variable a^* corresponds to attribute a . Each conjunction of $f_A(x)$ stems from an object $y \in U$ from which x can be discerned, and each term within that conjunction represents an attribute that discerns between those objects.³

³Relating this to the general Boolean reasoning scheme outlined in Section 4.4 and using the 1-normal form of the resulting Boolean equation, the system P of simultaneously-asserted Boolean equations would consist of one equation for each $y \in U$ such that $M_A(x, y) \neq \emptyset$. Equation 4.3 would then read,

$$f_A(x) = \prod_{y \in U} \left\{ \sum a^* \mid a \in M_A(x, y) \text{ and } M_A(x, y) \neq \emptyset \right\} \quad (5.20)$$

The prime implicants of $f_A(x)$ reveal the minimal subsets of A that are needed to discern object x from the objects in U that are not members of $R_A(x)$.

In addition to defining discernibility relative to a particular object, discernibility can also be defined for the information system \mathcal{A} as a whole. The full discernibility function $g_A(U)$ is defined below, and expresses how all objects in U can be discerned from each other.

$$g_A(U) = \prod_{x \in U} f_A(x) \quad (5.21)$$

The prime implicants of $g_A(U)$ reveal the minimal subsets of A we need to discern all distinct objects in U from each other.

Equation 5.21 includes all non-empty entries in M_A in the construction of $g_A(U)$. In practice, including entries from only the lower or upper triangular matrix of M_A suffices. This is explored below.

Simplification

A Boolean POS function can often be considerably simplified while fully preserving the function's semantics. First of all, duplicate sums can be eliminated since Boolean algebras have the property of multiplicative idempotence, meaning that $a \cdot a = a$ for all members a of the algebra's carrier set \mathbf{B} . If the function has n product terms, this can be done by a simple sort-and-scan procedure which is bounded by the sorting step, typically $O(n \log n)$. Furthermore, a sum that includes ("is a superset of") another sum in the function can be safely eliminated since in Boolean algebras $a \cdot (a + b) = a$ for all members $a, b \in \mathbf{B}$. This property is called *absorption*. Absorption can be carried out naively in $O(n^2)$ time, but subquadratic algorithms exist [171, 172]. Boolean SOP functions can be simplified in a similar manner.

In practical applications, Boolean functions are typically simplified during or after construction before being passed on to other procedures that employ them. The running time of the simplification often heavily dominates the running time of the overall algorithm, but this can be seen as a kind of "insurance" since it reduces the chances that the subsequent procedures that employ the function will have an unacceptably

taking the disjunction over all $a \in M_A(x, y)$:

$$P = \left\{ \begin{array}{l} \vdots \\ \sum a^* \\ \vdots \end{array} \right. = 1 \quad (5.19)$$

high running time. This because many of the procedures that employ and process Boolean functions are NP-hard, and we may hence choose to run an often dominating polynomial algorithm to make the problem smaller before passing it on to a potentially exponential algorithm.

Constructing Binary Information Systems

A discernibility function is constructible from an information system, but an information system is also constructible from a Boolean POS function. This opens up a gateway for using table-based software systems for general Boolean reasoning purposes.

Let h denote any Boolean POS function of m Boolean variables $\{a_1^*, \dots, a_m^*\}$, composed of n Boolean sums $\{s_1, \dots, s_n\}$. Furthermore, let $w_{ij}^* \in \{0, 1\}$ denote an indicator variable that states whether a_i^* occurs in s_j .

$$s_j = \sum_{i=1}^m w_{ij}^* \cdot a_i^* \quad (5.22)$$

$$h = \prod_{j=1}^n s_j \quad (5.23)$$

We can then construct a binary information system composed of a universe $U = \{x_0, \dots, x_n\}$ and a set of attributes $A = \{a_1, \dots, a_m\}$ as shown below.

$$a_i(x_j) = \begin{cases} 0 & \text{if } j = 0 \\ w_{ij}^* & \text{otherwise} \end{cases} \quad (5.24)$$

From the resulting binary information system, we then have that $f_A(x_0) = h$.

5.2.5 Reducts

An important practical issue is whether some of the attributes in an information system are redundant with respect to enabling us to make the same object classifications as with the full set of attributes A .

If an attribute subset $B \subseteq A$ preserves the indiscernibility relation R_A and hence our ability to form set approximations, then the attributes $A - B$ are said to be *dispensable*. Typically, an information system may have many such attribute subsets B . All such subsets that are minimal, i.e., that do not contain any dispensable attributes, are called *reducts*. The set of all reducts of an information system \mathcal{A} is denoted $RED(\mathcal{A})$.

A graphical display of the notion of reducts can be found in Figure 5.3.

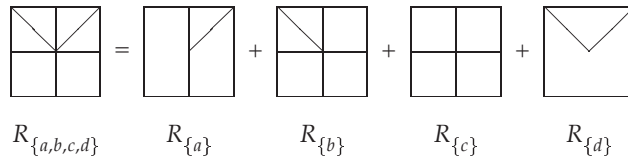
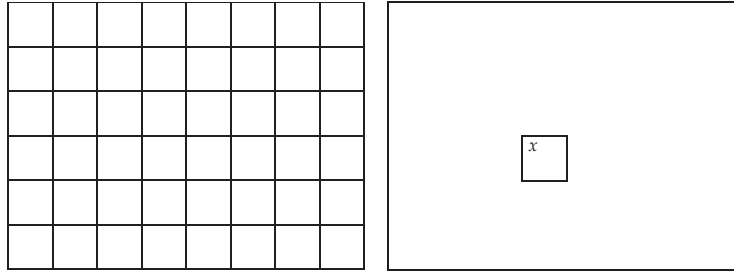


Figure 5.3: The indiscernibility relation R_A can be seen as the superposition of the indiscernibility relations $R_{\{a\}}$ for each of the individual attributes $a \in A$. As such, not all of the attributes might be needed in order to “sum up” to the total observed R_A . A reduct is a minimal subset $B \subseteq A$ such that $R_B = R_A$. In this example, the reducts are $\{a, b\}$, $\{b, d\}$ and $\{c, d\}$.



(a) Reducts in $RED(\mathcal{A})$ preserve the indiscernibility relation R_A , i.e., they preserve the “grid” or all “pixels”. Such reducts are minimal attribute subsets $B \subseteq A$ that enable us to discern all discernable objects from each other.

(b) Reducts in $RED(\mathcal{A}, x)$ preserve at least the indiscernibility set $R_A(x)$, i.e., they preserve the “pixel” containing x . Such reducts are minimal attribute subsets $B \subseteq A$ that enable us to discern object x from all other discernable objects.

Figure 5.4: Different types of reducts of an information system \mathcal{A} .

As noted in Section 5.2.4, different variations and combinations of indiscernibility are possible. In particular, indiscernibility may or may not be relative to a particular object x . If indiscernibility is relative to x , two objects y and z that are different from x are considered to be indiscernible simply on the basis that they differ from x , regardless of whether y and z are actually indiscernible. Reducts that are relative to a particular object reveal the minimum amount of information needed to discern that particular object from all other discernable objects. The set of all reducts of an information system \mathcal{A} that are relative to an object x is denoted $RED(\mathcal{A}, x)$.

A graphical display of which parts of the overall indiscernibility relation that the two types of reducts preserve can be found in Figure 5.4.

A reduct is isomorphic to a prime implicant of a discernibility function [196]. Hence the relationships listed below hold for any information system \mathcal{A} :

$$B \in RED(\mathcal{A}) \Leftrightarrow R_B = R_A \text{ and } B \text{ is minimal} \tag{5.25}$$

$$\Leftrightarrow \prod_{a \in B} a^* \text{ is a prime implicant of } g_A(U)$$

$$\begin{aligned} B \in \text{RED}(\mathcal{A}, x) &\Leftrightarrow R_B(x) = R_A(x) \text{ and } B \text{ is minimal} \\ &\Leftrightarrow \prod_{a \in B} a^* \text{ is a prime implicant of } f_A(x) \end{aligned} \quad (5.26)$$

A reduct $B \in \text{RED}(\mathcal{A})$ of an information system defines a functional dependency $B \rightarrow A - B$, as known from relational database theory. That is, on the basis of the attributes in a reduct we can define a function such that the values of the attributes not in the reduct can be computed. Viewed in the context of relational databases, we may thus use the fields B as a composite, minimal key for table \mathcal{A} .

Reducts and Hitting Sets

A *hitting set* of a given bag or multiset⁴ \mathcal{S} of elements from 2^A is a set $B \subseteq A$ such that the intersection between B and every set in \mathcal{S} is non-empty. The set $B \in \text{HS}(\mathcal{S})$ is a *minimal hitting set* of \mathcal{S} if B ceases to be a hitting set if any of its elements are removed. Let $\text{HS}(\mathcal{S})$ and $\text{MHS}(\mathcal{S})$ denote the sets of hitting sets and minimal hitting sets, respectively.

$$\text{HS}(\mathcal{S}) = \{B \subseteq A \mid B \cap S_i \neq \emptyset \text{ for all } S_i \text{ in } \mathcal{S}\} \quad (5.27)$$

The problem of computing prime implicants of Boolean POS functions is easily transformed into the problem of computing minimal hitting sets [196]. If h is a Boolean POS function composed on n sums as defined in Equation 5.22 and Equation 5.23, we can interpret h as a bag or multiset $\mathcal{S}(h)$ with n elements as follows:⁵

$$\mathcal{S}(h) = [S_i \mid S_i = \{a_j \in A \mid a_j^* \text{ occurs in } s_i\}] \quad (5.28)$$

A hitting set of $\mathcal{S}(h)$ obviously defines an implicant of h , and subsequently, a minimal hitting set corresponds to a prime implicant. Relating this connection to reducts, we thus have the following relationships:

⁴A bag or a multiset is conceptually an unordered collection of elements where the same element may occur more than once. Mathematically, therefore, it is common to define a multiset through a mapping from the element domain into the set of natural numbers, with the mapping defining the occurrence count. Here notation will be abused slightly and set-like syntax will in places be employed for convenience, even though duplicates are allowed. The text should make it clear whether we are dealing with sets or multisets. For additional clarity, a list-like notation with square brackets will be adopted for multisets in lieu of curly braces.

⁵The multiset constructor \mathcal{S} is a trivial matter of reinterpretation, as the following example shows:

$$\mathcal{S}((a^* + b^*) \cdot (a^* + b^*) \cdot (c^*)) = [\{a, b\}, \{a, b\}, \{c\}]$$

$$B \in RED(\mathcal{A}) \Leftrightarrow B \in MHS(\mathcal{S}(g_{\mathcal{A}}(U))) \quad (5.29)$$

$$B \in RED(\mathcal{A}, x) \Leftrightarrow B \in MHS(\mathcal{S}(f_{\mathcal{A}}(x))) \quad (5.30)$$

5.2.6 Reduct Approximations

Real-world data is almost always polluted with noise and other imperfections. Hence, since it only takes a single, noisy object to change the indiscernibility relation, reducts as defined in Section 5.2.5 are prone to incorporate noise and other peculiarities in the data set. Clearly, what is desirable to find are attribute subsets that reveal the underlying, general patterns in the data material. This implies that reduct approximations need to be found instead, i.e., attribute subsets that in a sense “almost” preserve the indiscernibility relation. In Figure 5.3, one could say that $\{b\}$ constitutes such a subset.

Dynamic Reducts

The process of computing *dynamic reducts* [12, 13] from an information system \mathcal{A} can be seen as combining normal reduct computation with resampling techniques. The basic idea is simple:

1. Randomly sample a family of subsystems $\mathcal{S} = \{\mathcal{A}_1, \dots, \mathcal{A}_n\}$ from \mathcal{A} , where each subsystem $\mathcal{A}_i = (U_i, A)$ and $U_i \subseteq U$.
2. From each subsystem $\mathcal{A}_i \in \mathcal{S}$, compute $RED(\mathcal{A}_i)$.
3. Note with which frequency each reduct occurs across all reducts computed in the previous step.

The reducts that occur the most often in the outlined procedure are believed to be the most “stable”, and reveal more general relationships in \mathcal{A} than $RED(\mathcal{A})$ does. The set of ε -dynamic reducts of an information system \mathcal{A} with respect to a family of sampled subsystems \mathcal{S} is denoted $DRED(\mathcal{A}, \varepsilon, \mathcal{S})$, and consists of those attribute subsets that occur “frequently enough” as reducts,⁶ as determined by the parameter ε .

$$DRED(\mathcal{A}, \varepsilon, \mathcal{S}) = \{B \subseteq A \mid \frac{|\{\mathcal{A}_i \in \mathcal{S} \mid B \in RED(\mathcal{A}_i)\}|}{|\mathcal{S}|} \geq 1 - \varepsilon\} \quad (5.31)$$

Similarly, one can also define dynamic reducts relative to an object $x \in U$ by considering $RED(\mathcal{A}_i, x)$ instead of $RED(\mathcal{A}_i)$. However, the sampling of the family of subsystems \mathcal{S} must then be constrained so that $x \in U_i$ for all $\mathcal{A}_i \in \mathcal{S}$.

⁶Equation 5.31 actually yields what Bazan [12] calls *generalized dynamic reducts*, and is slightly less strict in its definition than how dynamic reducts are often defined. To achieve dynamic reducts as defined in [12, 13, 192], the term $B \subseteq A$ in Equation 5.31 can be substituted by $B \in RED(\mathcal{A})$.

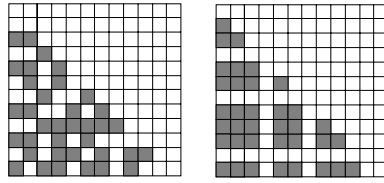


Figure 5.5: Conceptual depiction of the difference between an approximate hitting set formulation and one sample in a dynamic reduct computation framework. The former blocks out or ignores individual entries in the discernibility matrix, while the latter blocks out or ignores complete matrix rows and columns. Only half the matrix is displayed since the matrix is symmetric.

Approximate Hitting Sets

The sampling of subsets of objects $U_i \subseteq U$ done when computing dynamic reducts translates to removing or suppressing those rows and columns in the discernibility matrix that correspond to objects $x \notin U_i$. A natural generalization of this is to suppress individual entries in the matrix rather than complete rows or columns, i.e., ignoring only certain object-object interactions. This is illustrated in Figure 5.5, and can be accomplished through considering approximate solutions to the minimal hitting set problem outlined in Section 5.2.5.

An approximate solution to the hitting set problem is a set that hits “enough” elements of the bag or multiset \mathcal{S} . This can be further generalized to the situation where we have different numerical values or weights $w(S_i)$ associated with each member S_i of \mathcal{S} . The set of ε -approximate hitting sets of \mathcal{S} with respect to w is denoted $AHS(\mathcal{S}, \varepsilon, w)$, where the parameter ε controls the degree of approximation.

$$\sigma_w(\mathcal{S}') = \sum_{S_i \text{ in } \mathcal{S}'} w(S_i) \quad (5.32)$$

$$AHS(\mathcal{S}, \varepsilon, w) = \{B \subseteq A \mid \frac{\sigma_w([S_i \text{ in } \mathcal{S} \mid S_i \cap B \neq \emptyset])}{\sigma_w(\mathcal{S})} \geq \varepsilon\} \quad (5.33)$$

The set $B \in AHS(\mathcal{S}, \varepsilon, w)$ is a minimal ε -approximate hitting set if it ceases to be so if any of its elements are removed. The set of all minimal ε -approximate hitting sets is denoted $MAHS(\mathcal{S}, \varepsilon, w)$.

The purpose of introducing the function w should perhaps be mentioned. When the multiset \mathcal{S} is constructed from a Boolean POS function h as shown in Equation 5.28, w is meant to carry over some original numerical information that might otherwise go lost if h has undergone some kind of transformation, e.g., simplification as described in Section 5.2.4. For example, in an unsimplified Boolean POS function, w would typically equal unity for all members of \mathcal{S} and σ_w would reduce to a cardinality count. If all duplicate sums were eliminated from h , i.e., if \mathcal{S} was transformed from a multiset to an ordinary set, w would be updated to reflect the occurrence counts prior to

removing duplicate elements. The function w would be automatically⁷ updated in a similar manner when supersets are removed. See Vinterbo and Øhrn [233] for details. Without the weighting w and the multiset extension, the outlined approximation scheme is related to what Skowron and Nguyen [194] call α -reducts.

Computing all elements of $MAHS(\mathcal{S}, \varepsilon, w)$ is in general computationally intractable, and heuristics are needed. Vinterbo and Øhrn [233] apply a genetic algorithm to search for solutions. Furthermore, they incorporate the option to add information about attribute costs to the search so that “cheap” solutions can be obtained.

5.3 Decision Systems

It often happens that each entry or object in an information system has some kind of labeling associated with it. For instance, in a medical database, each object may represent a patient that may have a known disease status or treatment outcome. It is typically desirable to incorporate this labeling into the rough set analysis. An important subclass of information systems are therefore *decision systems*, also called *decision tables*.

A decision system is any information system \mathcal{A} of the form below, where $d \notin A$ is a distinguished attribute called the *decision attribute*. The elements of A are called *condition attributes*.

$$\mathcal{A} = (U, A \cup \{d\}) \quad (5.34)$$

Since a decision system is a special kind of information system, the mathematical machinery developed in Section 5.2 is applicable also to decision systems. In some cases, though, we may want to treat the decision attribute in a special manner. This will be further explored in Section 5.3.3.

5.3.1 Decision Classes

The decision attribute d induces a partition of the universe of objects U . Without loss of generality, we may assume that V_d is the set of integers $\{1, \dots, r(d)\}$, where $r(d)$ is said to be the *rank* of d . The induced partition is therefore the collection of equivalence classes $\{X_1, \dots, X_{r(d)}\}$, called *decision classes*, where two objects are said to belong to the same decision class if they have the same value for the decision attribute.

$$X_i = \{x \in U \mid d(x) = i\} \quad (5.35)$$

In most practical applications, it is the decision classes that we typically want to approximate using the framework outlined in Section 5.2.3.

⁷Automatically, but in a way that, for some reason (typically because it is too complicated, or too ugly, or perhaps even too trivial), the speaker doesn't feel like explaining to you [177].

5.3.2 Generalized Decisions

In a decision system \mathcal{A} it may happen that two objects that are indiscernible with respect to attributes A may belong to different decision classes. The decision system \mathcal{A} is said to be *inconsistent* with respect to A if this is the case, and *consistent* otherwise.

The *generalized decision attribute* ∂_A is a function $\partial_A : U \rightarrow 2^{V_d}$ that, when applied to object x , summarizes the set of values that the objects in the indiscernibility set $R_A(x)$ take on for the decision attribute d .

$$\partial_A(x) = \{v \in V_d \mid \exists y \in R_A(x) \text{ such that } d(y) = v\} \quad (5.36)$$

Note that the decision system \mathcal{A} is consistent if and only if $\partial_A(x)$ are singletons for all $x \in U$. Furthermore, note that the decision system obtained from \mathcal{A} by replacing d with ∂_A is necessarily consistent. This property is often exploited in practice, where in some procedures a decision system is assumed to be consistent. An inconsistent decision system can thus always be transformed to one that is consistent as an initial preprocessing step.

5.3.3 Discernibility Matrices

For almost every application where discernibility considerations are useful and each object has a labeling associated with it, a key observation to make is that we do not have to discern between objects that belong to the same decision class.

Since decision systems are so central, it is convenient to define the discernibility matrix *modulo the decision attribute* of a decision system \mathcal{A} . This will be denoted $M_{\mathcal{A}}^d$ and is defined in Equation 5.38 through a simple augmentation of the *discerns/3* predicate from Section 5.2.1. The *discerns/4* predicate is defined in terms of *discerns/3*, but with the generalized decision functioning as an additional “filtering constraint”.

$$\text{discerns}(a, x, y, d) \Leftrightarrow \partial_A(x) \neq \partial_A(y) \text{ and } \text{discerns}(a, x, y) \quad (5.37)$$

$$M_{\mathcal{A}}^d(x, y) = \{a \in A \mid \text{discerns}(a, x, y, d)\} \quad (5.38)$$

To incorporate the fact that we need not discern between objects that belong to the same decision class, we can simply substitute $M_{\mathcal{A}}^d$ for $M_{\mathcal{A}}$ in all the formulae given in Section 5.2. This has the effect of propagating this very consideration throughout the mathematical machinery developed in Section 5.2.⁸ The resulting relations, functions and sets are said to be computed modulo d , and will be denoted with a d superscript.

⁸Here, computation in the context of decision systems is defined as a slight variation of computation in the context of information systems. We could just as well have made the definitions the other way around, since Equation 5.38 reduces to Equation 5.3 as a special case when, e.g., d is an automorphism of U .

Since discernibility functions and reducts are of such central practical importance, they will for clarity be presented again in the following section, but reframed in the setting of decision systems.

Boundary Region Thinning

Just as the *discerns/3* predicate could be tailored to the application at hand, one can employ slightly more complex “filters” in Equation 5.37 than strict inequality of the generalized decision values. For example, distributions of decision values within $R_A(x)$ and $R_A(y)$ could come into play. This idea is sometimes referred to as *boundary region thinning*, of which the variable precision model from Section 5.2.3 is an example.

$$\Delta_A^\pi(x) = \{v \in \partial_A(x) \mid \mu_A^{X_v}(x) \geq \pi\} \quad (5.39)$$

$$\partial_A^\pi(x) = \begin{cases} \Delta_A^\pi(x) & \text{if } \Delta_A^\pi(x) \neq \emptyset \\ \partial_A(x) & \text{otherwise} \end{cases} \quad (5.40)$$

$$\text{discerns}(a, x, y, d) \Leftrightarrow \partial_A^\pi(x) \neq \partial_A^\pi(y) \text{ and } \text{discerns}(a, x, y) \quad (5.41)$$

Hence, with boundary region thinning we look at the distribution of decision values within each indiscernibility set, and exclude those decision values from the generalized decision that occur with a frequency below π . Low-probability decision values are thus treated as “noise”.

5.3.4 Discernibility Functions

Equation 5.20 can readily be computed modulo d as a special case, as shown below.

$$f_A^d(x) = \prod_{y \in U} \left\{ \sum a^* \mid a \in M_A^d(x, y) \text{ and } M_A^d(x, y) \neq \emptyset \right\} \quad (5.42)$$

The prime implicants of $f_A^d(x)$ reveal the minimal subsets of A that are needed to determine which decision class object x belongs to.

A discernibility function can also be defined for a decision system \mathcal{A} as a whole. The discernibility function $g_A^d(U)$ defined below expresses how all decision classes in \mathcal{A} can be discerned from each other.

$$g_A^d(U) = \prod_{x \in U} f_A^d(x) \quad (5.43)$$

The prime implicants of $g_A^d(U)$ reveal the minimal subsets of A we need in order to determine the decision class of every object in U .

5.3.5 Reducts

In Section 5.2.5, the concept of reducts was defined in the context of information systems. Although the same mathematical machinery is applicable to decision systems as noted in Section 5.3.3, reducts are briefly reviewed again here for clarity, but computed modulo d .

Informally, an attribute subset $B \subseteq A$ is a reduct of a decision system \mathcal{A} computed modulo d if B is minimal and preserves at least our ability to discern between objects that have different generalized decisions, i.e., lie in different approximation regions. The set of all such attribute subsets is denoted $RED(\mathcal{A}, d)$. With such a reduct we may often be able to also discern between some objects that lie in the same approximation region, but this is not a requirement.

As was outlined in Section 5.2.5, reducts can also be relative to a particular object $x \in U$. Reducts that are relative to an object x and computed modulo d are minimal attribute subsets $B \subseteq A$ that enable us to identify at least the objects in $R_A(x)$, possibly together with some other objects. These other objects, however, are all required to have the same value for the generalized decision attribute as x . The set of all such reducts is denoted $RED(\mathcal{A}, x, d)$.

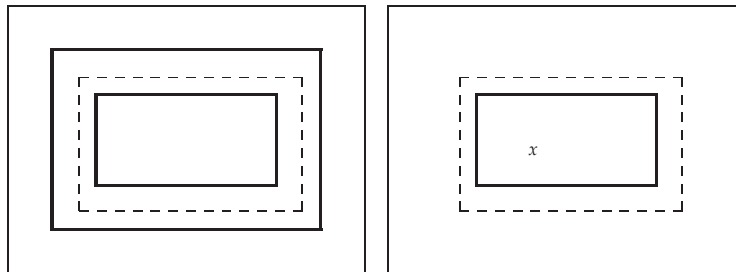
Formally, we have the following relationships:

$$B \in RED(\mathcal{A}, d) \Leftrightarrow \prod_{a \in B} a^* \text{ is a prime implicant of } g_A^d(U) \quad (5.44)$$

$$B \in RED(\mathcal{A}, x, d) \Leftrightarrow \prod_{a \in B} a^* \text{ is a prime implicant of } f_A^d(x) \quad (5.45)$$

A graphical metaphor for which parts of the overall indiscernibility relation that the two types of reducts concentrate on preserving can be found in Figure 5.6.

Reducts reveal redundancies in tabular data. A reduct $B \in RED(\mathcal{A}, d)$ of a decision system defines a functional dependency $B \rightarrow \{d\}$, as known from the theory of relational databases.



(a) Conceptually, a reduct in $RED(\mathcal{A}, d)$ preserves at least the boundaries between the approximation regions. That is, indiscernibility sets within the approximation regions are allowed to coalesce.

(b) Conceptually, a reduct in $RED(\mathcal{A}, x, d)$ preserves at least the indiscernibility set $R_{\mathcal{A}}(x)$, but also allows this to merge with other objects that lie in the same approximation region as x .

Figure 5.6: Different types of reducts of a decision system \mathcal{A} , computed modulo d .

Chapter 6

Patterns and Rules

6.1 Introduction

In practical applications, one of the main purposes of rough set data analysis is to induce or mine patterns or rules from data represented as information or decision systems. This chapter makes the notions of patterns and decision rules more precise, and shows how they can be computed from tabular data using the framework outlined in Chapter 5. Furthermore, an overview of numerical measures of patterns and rules is presented, along with an introduction to how ensembles of decision rules can be employed to realize classifiers.

6.2 Patterns

It is convenient to define a language for describing objects in an information system that have a certain set of properties. A language is defined through a *syntax* that defines the set of valid expressions or formulae in the language, and a *semantics* that maps syntactical entities to their meaning. The meaning of an expression will in our case be a subset of the universe of objects.

6.2.1 Syntax

A formula in our language \mathcal{L} is called a *pattern*. The most primitive pattern and hence the fundamental building block for assembling formulae in our language is called a *descriptor*. A descriptor is simply an expression $a = v$, where $a \in A$ and $v \in V_a$.

Patterns can be combined in a recursive manner in order to form more complex patterns by means of the propositional connectives $\{ \cdot, +, \rightarrow, \neg \}$, denoting conjunction, disjunction, implication and negation respectively. We allow patterns to be parenthesized for clarity.

1. If α is on the form $a = v$ then $\alpha \in \mathcal{L}$.
2. If $\alpha \in \mathcal{L}$ and $\beta \in \mathcal{L}$ then $\alpha \cdot \beta \in \mathcal{L}$.
3. If $\alpha \in \mathcal{L}$ and $\beta \in \mathcal{L}$ then $\alpha + \beta \in \mathcal{L}$.
4. If $\alpha \in \mathcal{L}$ and $\beta \in \mathcal{L}$ then $\alpha \rightarrow \beta \in \mathcal{L}$.
5. If $\alpha \in \mathcal{L}$ then $\neg\alpha \in \mathcal{L}$.
6. If $\alpha \in \mathcal{L}$ then $(\alpha) \in \mathcal{L}$.
7. A pattern α is in language \mathcal{L} if and only if its being so follows from finitely many applications of the rules above.

6.2.2 Semantics

Let α be a pattern in \mathcal{L} . The semantics or meaning of α with respect to an information system \mathcal{A} is denoted $\llbracket \alpha \rrbracket_{\mathcal{A}}$, and is defined recursively as shown below.

1. $\llbracket a = v \rrbracket_{\mathcal{A}} = \{x \in U \mid (v, a(x)) \in E_a\}$
2. $\llbracket \alpha \cdot \beta \rrbracket_{\mathcal{A}} = \llbracket \alpha \rrbracket_{\mathcal{A}} \cap \llbracket \beta \rrbracket_{\mathcal{A}}$
3. $\llbracket \alpha + \beta \rrbracket_{\mathcal{A}} = \llbracket \alpha \rrbracket_{\mathcal{A}} \cup \llbracket \beta \rrbracket_{\mathcal{A}}$
4. $\llbracket \alpha \rightarrow \beta \rrbracket_{\mathcal{A}} = \llbracket \neg\alpha + \beta \rrbracket_{\mathcal{A}}$
5. $\llbracket \neg\alpha \rrbracket_{\mathcal{A}} = U - \llbracket \alpha \rrbracket_{\mathcal{A}}$
6. $\llbracket (\alpha) \rrbracket_{\mathcal{A}} = \llbracket \alpha \rrbracket_{\mathcal{A}}$

The base case of descriptors is here defined in terms of the IDG formulation from Section 5.2.1.

If the information system \mathcal{A} is understood, the subscript may be omitted.

6.2.3 Minimal Patterns

The type of pattern that is most commonly considered in an information system \mathcal{A} is conjunctions of descriptors, formed by overlaying the set of attributes A over an object $x \in U$ and reading off the values of x for every $a \in A$. This defines a characteristic formula for object x with respect to attributes A , and is denoted $F_A(x)$:

$$F_A(x) = \prod_{a \in A} (a = a(x)) \quad (6.1)$$

Note that the typical meaning of the characteristic formula $F_A(x)$ equals the indiscernibility set $R_A(x)$. This means that a reduct $B \in RED(\mathcal{A}, x)$ is a minimal subset of A that preserves the semantics of $F_A(x)$. Hence, reducts form a natural basis for defining sets of minimal patterns:

$$PAT(\mathcal{A}, x) = \{F_B(x) \mid B \in RED(\mathcal{A}, x)\} \quad (6.2)$$

$$PAT(\mathcal{A}) = \bigcup_{x \in U} PAT(\mathcal{A}, x) \quad (6.3)$$

6.2.4 Numerical Measures

Having defined the semantics of the pattern α , we are in the position to define several numerical quantities associated with α . The *support* of pattern α is the number of objects in the information system \mathcal{A} that have the property described by α .

$$support(\alpha) = ||[\alpha]|| \quad (6.4)$$

The *coverage* of pattern α is the support of α divided by the number of objects in our universe U . Hence, $coverage(\alpha)$ denotes the proportion of objects in U that match the description given by α .

$$coverage(\alpha) = \frac{support(\alpha)}{|U|} \quad (6.5)$$

If B is an attribute subset generated by a dynamic reduct computation procedure as described in Section 5.2.6 and x is an object in the decision table B was computed from, then it is possible to define the *stability* of the pattern $F_B(x)$. This issue is further explored by Bazan [12].

6.3 Decision Rules

Similarly as a decision system is a specialized type of information system, a *decision rule* is a specialized type of pattern, as defined in Section 6.2. A decision rule reflects a relationship, possibly a probabilistic one, between a set of conditions and a conclusion or decision.

Let \mathcal{A} denote a decision system, and let α denote a conjunction of descriptors that only involve attributes in A . Furthermore, let β denote a descriptor $d = v$, where v is any allowed decision value. The decision rule read “if α then β ” is denoted by $\alpha \rightarrow \beta$, with its semantics defined in Section 6.2.2. The pattern α is called the rule’s *antecedent*, while the pattern β is called the rule’s *consequent*.

6.3.1 Minimal Decision Rules

Reducts relative to an object define minimal patterns, as noted in Section 6.2. A natural scheme for mining a set of minimal decision rules from a decision system \mathcal{A} is therefore to use the reducts as a defining basis:

$$RUL(\mathcal{A}, x, d) = \{F_B(x) \rightarrow (\partial_B = \partial_B(x)) \mid B \in RED(\mathcal{A}, x, d)\} \quad (6.6)$$

$$RUL(\mathcal{A}, d) = \bigcup_{x \in U} RUL(\mathcal{A}, x, d) \quad (6.7)$$

In practical applications where the rules are used to classify unseen objects, reduct approximations are typically employed instead of proper reducts.

6.3.2 Numerical Measures

Several numerical quantities of interest can be associated with a decision rule $\alpha \rightarrow \beta$. Frequently, when we talk about the *support* of the rule we mean the number of objects in the decision system that possess both properties α and β . Most numerical quantities of interest are derived from support counts.

Accuracy

A decision rule $\alpha \rightarrow \beta$ may only reveal a part of the overall picture of the decision system from which it was derived. It may happen that the decision system contains objects that match the rule's antecedent α , but that have a different value for the decision attribute than the one indicated by the rule's consequent β . Hence, we are interested in the probability of the conclusion β being correct, given pattern α . The quantity *accuracy*($\alpha \rightarrow \beta$) gives a measure of how trustworthy the rule is in drawing conclusion β on the basis of evidence α , and is a frequency-based estimate of the conditional probability $\Pr(\beta \mid \alpha)$.

$$accuracy(\alpha \rightarrow \beta) = \frac{support(\alpha \cdot \beta)}{support(\alpha)} \quad (6.8)$$

Let B denote the set of attributes that appear in the antecedent α of a decision rule $\alpha \rightarrow \beta$, where α is a conjunction and β is assumed to be a simple descriptor. Let X denote the decision class described by β . Relating the accuracy measure to the concepts introduced in Section 5.2.3, we see that *accuracy*($\alpha \rightarrow \beta$) is identical to the value of the rough membership function μ_B^X applied to an object x that matches α . Thus, *accuracy* measures the degree of membership of x in X , using attributes B .

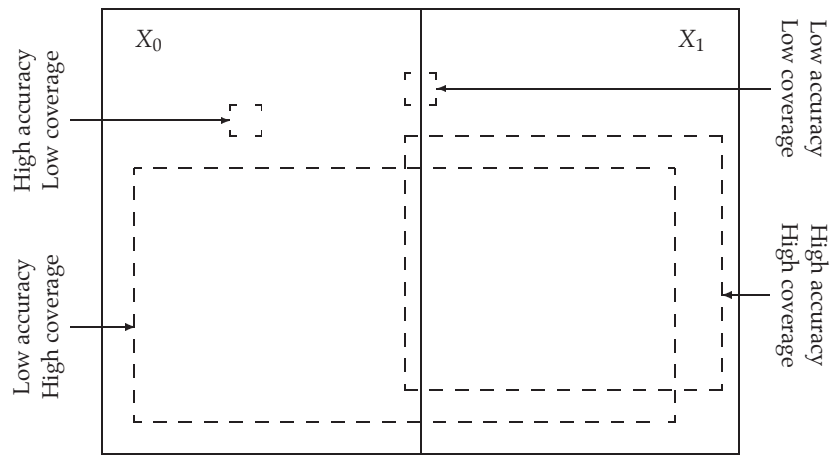


Figure 6.1: Depiction of four decision rules $\alpha_i \rightarrow \beta_i$ over a binary decision domain, i.e., $\beta_i = (d = 0)$ or $\beta_i = (d = 1)$. Each dashed set represents $support(\alpha_i)$, the set of objects that match the rule's antecedent α_i .

Coverage

We want a rule to be strong in the sense that it has a large support basis. However, what we consider to be “large” typically varies with how the decision values are distributed. The quantity $coverage(\alpha \rightarrow \beta)$ gives a measure of how well the pattern α describes the decision class defined through β , and is a frequency-based estimate of the conditional probability $\Pr(\alpha \mid \beta)$.

$$coverage(\alpha \rightarrow \beta) = \frac{support(\alpha \cdot \beta)}{support(\beta)} \quad (6.9)$$

A graphical display of the relationship between accuracy and coverage can be found in Figure 6.1. It is desirable for a rule to be accurate as well as to have a high degree of coverage, although one does not necessarily imply the other.

Stability

If $\alpha \rightarrow \beta$ is a decision rule that is constructed by overlaying a dynamic reduct over an object in an information system as described in Section 6.2, then it is also possible to define the *stability* of the rule. This issue is further explored by Bazan [12].

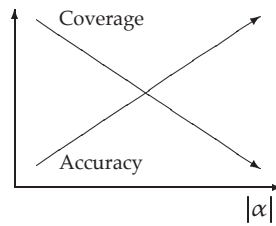


Figure 6.2: As the length of the antecedent of a decision rule increases, the rule becomes more specific and less general. As a result, the coverage decreases while the accuracy increases. Finding a suitable balance between the trade-off between coverage and accuracy can be difficult in practice.

6.3.3 Balancing Accuracy and Coverage

Obviously, we would like to obtain decision rules that are both accurate as well as have a high degree of coverage. It is not difficult to show that as the antecedent of a decision rule grows longer, the coverage decreases while the accuracy increases. This is shown graphically in Figure 6.2. Defining a point that balances the trade-off between these two numerical rule measures can be difficult in practice, and is also a function of the application domain.

For rule evaluation purposes, it would be helpful if we could combine accuracy and coverage into a compound *quality* measure for a decision rule $\alpha \rightarrow \beta$. By counting the number of objects that match or don't match patterns α and β , we can form a contingency table from which quality measures can be defined.

Bruha [22] provides a survey of several quality measures. Some measures are purely ad hoc, while others have more firm theoretical foundations. Approaches range from simply forming a linear combination of the accuracy and coverage measures, to using a contingency table to form more complex statistically founded measures of association and agreement. Ågotnes [3] investigates how these quality measures can be employed for rule filtering purposes.

6.4 Classification

A natural use of a set of induced decision rules is to see how well the ensemble of rules is able to classify new and unseen objects. This is explored in the following section.

6.4.1 Voting

Voting is an ad hoc technique for rule-based classification that works reasonably well in practice. In the following, the rule set will be assumed to be *unordered*, i.e., all rules in the rule set will participate in the classification process.

Let RUL denote an unordered set of decision rules. The process of voting is one way of employing RUL to assign a numerical certainty factor to each decision class for each object. Typically, but not necessarily, an object is classified to the decision class for which the certainty factor is maximized. This is further discussed in Chapter 7.

Presented with a given object x to classify, the voting process goes as follows:

1. The set RUL is scanned for rules that *fire*, i.e., rules that have an antecedent that matches x . Let $RUL(x) \subseteq RUL$ denote the set of rules that fire for object x .
2. If $RUL(x) = \emptyset$, then no classification can be made. Typically, a predetermined fallback classification is then invoked. Alternatives include reverting to a nearest-neighbor method and instead consider a collection of rules “close” to object x in some sense [205].
3. An election process among the rules in $RUL(x)$ is performed in order to resolve conflicts and rank the decisions. The election process is performed as follows:
 - (a) Let each rule $r \in RUL(x)$ cast a number in votes $votes(r)$ in favor of the decision class the rule indicates.¹ The number of votes a rule gets to cast may vary. Typically, $votes(r)$ is based on the support of the rule, but more complex quality-based measures are also possible.
 - (b) Compute a normalization factor $norm(x)$. The normalization factor can be computed in different ways. Typically, $norm(x)$ is simply the sum of all cast votes, and only serves as a scaling factor.
 - (c) Divide the accumulated number of votes for each possible decision class β by the normalization factor $norm(x)$ in order to arrive at a certainty coefficient $certainty(x, \beta)$ for each decision class.

$$R_\beta = \{r \in RUL(x) \mid r \text{ predicts } \beta\} \quad (6.10)$$

$$votes(\beta) = \sum_{r \in R_\beta} votes(r) \quad (6.11)$$

$$certainty(x, \beta) = votes(\beta) / norm(x) \quad (6.12)$$

Most of the steps in the voting process have several degrees of freedom. For instance, how missing values are treated in the firing step may be dealt with in different ways. Furthermore, if two or more firing rules are generalizations² of each other, we may or may not choose to exclude all but the most specific of these from the set of firing rules before proceeding. Various different approaches to defining $votes(r)$ and $norm(x)$ can also be envisioned.

¹If a rule in $RUL(x)$ indicates more than one possible decision, we may imagine this rule to be logically expanded to several rules, each with the same antecedent but with a single descriptor only as a consequent.

²If α_i and α_j are antecedents of rules r_i and r_j and α_j can be written as $\alpha_i \cdot \alpha$ for some combination of descriptors α , we say that r_i is more general than r_j .

Univariate Rules and Support Voting

It is worth analyzing the interesting special case that arises when RUL comprises the set of all *univariate* rules. A rule is said to be univariate if its antecedent consists of a single descriptor only. Univariate rules are sometimes referred to as *1R rules*, and often work fairly well as simple classifiers [82].

$$RUL = 1R(A) = \bigcup_{x \in U} \bigcup_{a \in A} \{(a = a(x)) \rightarrow (d = d(x))\} \quad (6.13)$$

Let $r = \alpha \rightarrow \beta$ denote a 1R rule, as defined above. Furthermore, let the quantity $votes(r)$ be defined as $support(\alpha \cdot \beta)$, i.e., the number of objects that match both the antecedent and the consequent of rule r simultaneously. Note that if $r \in RUL(x)$, then the antecedent of r reads $(a = a(x))$. We then have the following relationship, where all probabilities are as estimated from the set U of training cases:

$$\begin{aligned} certainty(x, \beta) &= votes(\beta) / norm(x) \\ &\propto votes(\beta) \\ &= \sum_{r \in R_\beta} votes(r) \\ &= \sum_{r \in R_\beta} support(\alpha \cdot \beta) \\ &= \sum_{a \in A} \left\{ \frac{accuracy((a = a(x)) \rightarrow \beta) \times}{support((a = a(x)))} \right\} \\ &= \sum_{a \in A} \left\{ \frac{\Pr(\beta \mid (a = a(x))) \times}{\Pr((a = a(x))) \times |U|} \right\} \\ &\propto \sum_{a \in A} \Pr(\beta \mid (a = a(x))) \times \Pr((a = a(x))) \\ &= \sum_{a \in A} \Pr((a = a(x)) \text{ and } \beta) \end{aligned} \quad (6.14)$$

Hence, using 1R rules and support-based voting, the computed certainty coefficient $certainty(x, \beta)$ is proportional to the summed estimated probabilities of each of the observed attribute values of x occurring together with β . It seems natural to classify x to the decision class which maximizes this expression, although this may not necessarily be the optimal approach with respect to some cost measure. This issue is further discussed in Chapter 7.

6.4.2 Object Tracking

Even though standard voting as described in Section 6.4.1 is simple and often works adequately, an objection one might make is that the procedure does not take into account which objects the rules in $RUL(x)$ were originally derived from. It may very

well be the case that the sets of objects that two or more firing rules were originally derived from partially overlap. Hence, the objects that are members of such intersections are given a possibly unfair amount of weight in the voting process, since they effectively participate in the voting process through more than one rule.

Let $\alpha \rightarrow \beta$ denote a rule in $RUL(x)$, and let $\alpha \rightarrow \beta$ have been induced from a decision system \mathcal{A} . We define the set of *tracked objects* for the rule as $\llbracket \alpha \rrbracket_{\mathcal{A}}$, i.e., the set of objects that are included in the support basis of the rule's antecedent.³ The union of all tracked objects for all firing rules then constitutes a natural basis for assigning a "fair" certainty factor to x .

$$tracked(RUL(x)) = \{y \in U_{\mathcal{A}} \mid \exists(\alpha \rightarrow \beta) \in RUL(x) \text{ such that } y \in \llbracket \alpha \rrbracket_{\mathcal{A}}\} \quad (6.15)$$

$$certainty(x, (d = v)) = \frac{|\{y \in tracked(RUL(x)) \mid d(y) = v\}|}{|tracked(RUL(x))|} \quad (6.16)$$

Object tracking can also be viewed as a kind of voting. The participants in the election process are then not the firing rules themselves, but rather the union of all the tracked objects defined through the firing rules. Each tracked object in this union gets to cast one vote, and the results are then normalized to sum to unity.

Lastly, and as noted in Section 2.4.3, it is worth commenting on the potential for using the originating, tracked objects as a case-based explanation to complement the model-based explanation $RUL(x)$. By coupling the sets of tracked objects with Boolean reasoning, minimal case-based explanations can be obtained.⁴

³Note that we do not here make use of the consequent, we can hence just as well use general patterns instead of decision rules.

⁴For example, by considering the shortest prime implicants of the following Boolean function, where α is the antecedent of a rule in $RUL(x)$:

$$h = \prod_{\alpha} \left\{ \sum y^* \mid y \in \llbracket \alpha \rrbracket_{\mathcal{A}} \right\} \quad (6.17)$$

Note that this is not specific to rough sets or rule-based classifiers, but is applicable to all classifiers that are composed of a set of "detectors" and where we can associate a subset of $U_{\mathcal{A}}$ with each detector.

Chapter 7

Classifier Evaluation

7.1 Introduction

Classifiers induced from empirical data can be evaluated along at least two dimensions: Performance and explanatory features. By performance is meant assessment of how well the classifier does in classifying new cases, according to some specified performance criterion. By explanatory features is meant how interpretable the structure of the induced classifier is, so that one might gain some insight into how the classification or decision making process is carried out. How these two evaluation dimensions are to be weighted is a matter of the intended role of the induced classifier. If the classifier is to operate in a fully automated environment, then performance may be the only feature of interest. Conversely, if the classifier induction is part of a larger data mining or knowledge discovery process, then the interpretability of the classifier will be increasingly more important, and a decrease in performance may be acceptable. Classifiers that are to function as parts of interactive decision support systems lie somewhere in the middle of these two extremes.

Different machine learning methods for classification vary in how much they facilitate the knowledge discovery aspect, depending on the type of classifiers they produce. A point that is often held forth in favour of methods that produce decision tree or rule sets is that the models are directly readable and interpretable. In contrast, methods such as logistic regression or artificial neural networks are more difficult to interpret and may require familiarity with sophisticated statistical concepts. This chapter will not touch upon the issue of classifier interpretability, but will concern itself only with the performance evaluation aspect.

A classifier κ is in the following viewed as a realization of a function that, when applied to an object $x \in U$ in an information system \mathcal{A} , assigns a classification $\hat{d}_\kappa(x)$ to x . The true actual classification of x is denoted $d(x)$. In the following, unless otherwise stated and relevant for the exposition, no distinction will be made between a classifier and the function it realizes.

$$\hat{d}_\kappa : U \rightarrow V_d \quad (7.1)$$

We assume in the following that κ is forced to make a classification when presented with an object. In the case that the classifier fails to recognize an object, a default classification is assumed invoked. Extensions that incorporate rejection and degrees of doubt by κ are possible [179].

Section 7.2 discusses how to conduct classification experiments in a structured and systematic way so that reliable performance estimates can be obtained, while Section 7.3 reviews a simple way of collating the produced classifications. Section 7.4 then focuses on evaluation methods for binary classifiers. Two key evaluation dimensions are discussed in Section 7.4.2, and frameworks for graphically assessing these are presented in Section 7.4.3 and Section 7.4.4. Different quantitative performance measures are discussed in Section 7.4.5, while Section 7.4.6 outlines techniques for statistical comparison of such measures.

7.2 Partitioning the Examples

In supervised learning we are given a set of labeled example objects in a decision system \mathcal{A} , and want to construct a mapping \hat{d}_κ that maps elements in U to elements in V_d , using only attributes contained in A . If we use the full set of examples \mathcal{A} to construct the classifier, we may obtain a model that, if chosen the learning paradigm lends itself to it, can be inspected and generate hypotheses that explain the observations. In some applications this might be valuable, but then we do not have any data left to assess the performance of the classifier in an unbiased manner. Applying the classifier to the dataset from which it was induced will obviously give an unfair and biased assessment, since a classifier can always be constructed that reproduces its originating data material perfectly. A perfect (or near-perfect, in the case of inconsistent labels) mapping can easily be constructed through rote learning, for instance by collecting all the provided examples in a lookup table. Since the full set of examples \mathcal{A} was used to construct \hat{d}_κ , we have no way of assessing how well \hat{d}_κ generalizes to new and unseen example objects.

To this end and since in practice \mathcal{A} is almost always a finite and limited collection of possible examples, it is customary to randomly divide the examples in \mathcal{A} into two disjoint subsets, a *training set* and a *test set*. The training set is used to construct κ , while the test set is used to assess its performance. Under the assumption that the two sets comprise independent samples, this ensures us that the performance estimate will be unbiased.

Every algorithm for classifier construction has a set of parameters or options which can be tweaked and tuned, and it is natural to select these settings so that the chosen performance criterion is optimized. Performance assessment for this purpose should *not* be made using the test set, since we will then have no way of estimating the classifier's true performance. And using the training set for parameter tuning may also not

desirable, since we may then be likely to *overfit* our data. By overfitting is meant that the constructed mapping is overly geared towards reproducing the training set, and may have captured noise and other data impurities that may be present. An overfitted model is not likely to generalize well to unseen examples.

These considerations may lead us to divide \mathcal{A} into three disjoint sets of examples instead of two, namely into a training set, a test set and a *hold-out set*. While the classifier κ is induced from the training set, the purpose of the hold-out set is to function as a “test set during training”, i.e., as a set of examples apart from those considered when constructing the mapping defined by κ , used for guiding the training process. The performance assessment on this hold-out set can be used to tune the training parameters, decide on when training should be stopped, or to aid in pruning or simplification of the model derived from the training set. The test set is kept completely separate, and is only used at the last moment to evaluate the performance of the classifier that was constructed from the training and hold-out sets.

7.2.1 Systematic Partitioning Methods

The reliability of the performance estimated from a single partitioning can be questioned. It could be that the random division of examples used for training and testing was a particularly “lucky” or “unlucky” split. One way of reducing the possible impact of the split is to repeat the described training and test process several times with different random splits, and to average the performance estimates from each iteration. However, this way the training and testing sets from iteration i will almost surely partially overlap with the training and testing sets from iteration j . Hence it would be desirable to take a more systematic approach where also the possibility of similar partitionings is taken into account.

Cross-Validation

The process of *cross-validation* (CV) is a way of getting more reliable estimates and more mileage out of possibly scarce data. In *k-fold* CV we randomly divide the set of examples into k disjoint “blocks” of examples, usually of equal size. Then we can apply a classifier trained using $k - 1$ blocks to the remaining block to assess its performance. Repeating this for each of the k blocks enables us to average the estimates from each iteration to obtain an unbiased performance estimate. By this procedure, each example is guaranteed to be in the test set once and in the training size $k - 1$ times.

An extreme variant of selecting k is to choose $k = |U|$, i.e., letting each test set consist of a single example. This is called *leave-one-out* CV, and, although potentially extremely computer-intensive, may be intuitively pleasing as it most closely mimics the true size of the training set.

Bootstrapping

Another approach to systematic partitioning of the training examples is the *bootstrap*. The basic idea of the procedure known as the *0.632-bootstrap* is to randomly sample a training set \mathcal{A}^* of $|U|$ examples from \mathcal{A} with replacement, and to assess the performance of the classifier on \mathcal{A}^* and on the examples in \mathcal{A} and not in \mathcal{A}^* . These two estimates are then weighted by 0.368 and 0.632¹ respectively, and added to obtain a bootstrap performance estimate. The process can then be repeated several times in order to compute the average bootstrap estimate.

Remarks

Of course, systematic partitioning methods can be combined with using hold-out sets, simply by doing a second-level split on the examples originally singled out for training. Salzberg [183] recommends using CV together with hold-out sets as a means of ensuring fair comparisons between classifiers.

If inspection of the model for KDD purposes is a primary goal, then systematic partitioning methods introduces a complicating factor. For what is then the model that goes along with the obtained performance estimate? Instead of a single model, we have instead a plethora of different models. A common way to interpret the performance estimate is as the estimate we would get if we had induced a model from the full database and had had more data with which to test it.

Resampling ideas can be employed not only for systematically partitioning the data into training and test examples, but can also be used in process of inducing a model from the training set. We can thus introduce resampling at two different levels: At the evaluation level and at the model induction level. If the latter is done, the induced model is really an ensemble of several submodels that are ultimately combined. Such meta-methods are sometimes called *bagging*² or *boosting*. In the bagging procedure by Breiman [19], a large number of bootstrapped training sets are sampled from the training set by the previously described bootstrap procedure, and one submodel is induced from each of these bootstrap samples. The final model is then defined as the aggregation of all the submodels by uniform voting, i.e., when an presented with an unseen example, the ensemble labels it with the class that is predicted by the greatest number of submodels. In boosting [60, 186], the training set chosen at any point depends on the performance of earlier classifiers. Examples that are incorrectly classified are then chosen more often than correctly classified examples. Bagging and boosting can often considerably improve classificatory performance [10].

Further general readings on CV, bootstrapping, and other resampling techniques can be found in, e.g., [52, 179].

¹The number 0.632 is shorthand for $(1 - \frac{1}{e})$, the limit for large $|U|$ of the probability that a given example in \mathcal{A} appears in \mathcal{A}^* .

²Short for “bootstrap aggregation”.

7.3 Confusion Matrices

A *confusion matrix* C is a $|V_d| \times |V_d|$ matrix with integer entries that summarizes the performance of a classifier κ , applied to the objects in an information system \mathcal{A} .

Without loss of generality we may assume that V_d is the set of integers $\{0, \dots, r(d) - 1\}$, as defined in Section 5.3.1. The entry $C(i, j)$ counts the number of objects that really belong to class i , but were classified by κ as belonging to class j .

$$C(i, j) = |\{x \in U \mid d(x) = i \text{ and } \hat{d}_\kappa(x) = j\}| \quad (7.2)$$

Obviously, it is desirable for the diagonal entries to be as large as possible. Probabilities are easily estimated from the confusion matrix C by dividing an entry by the sum of the row or column the entry appears in:

$$\Pr(d(x) = i \mid \hat{d}_\kappa(x) = j) = \frac{C(i, j)}{\sum_i C(i, j)} \quad (7.3)$$

$$\Pr(\hat{d}_\kappa(x) = j \mid d(x) = i) = \frac{C(i, j)}{\sum_j C(i, j)} \quad (7.4)$$

$$\Pr(d(x) = \hat{d}_\kappa(x)) = \frac{\sum_i C(i, i)}{\sum_i \sum_j C(i, j)} \quad (7.5)$$

Classification tasks with binary outcomes are so common in practice that the entries of 2×2 confusion matrices and the corresponding values of Equation 7.3 and Equation 7.4 are given special names. Consider the following confusion matrix:

$$\begin{array}{c|cc} & \hat{d}_\kappa & \\ & 0 & 1 \\ \hline d & 0 & \begin{array}{l} TN \\ FP \end{array} \\ & 1 & \begin{array}{l} FN \\ TP \end{array} \end{array}$$

The names given to the entries and derived quantities from 2×2 confusion matrices are listed in Table 7.1 The sensitivity of a classifier thus gives us a measure of how good it is in detecting that an event defined through X_1 has occurred, while the specificity gives us a measure of how good it is in picking up non-events defined through X_0 . The positive and negative predictive values of a classifier gives us a measure of how “trustworthy” it is in its detections of events and non-events.

7.4 Binary Classifiers

In the following, since the situation is so common in practice, we will examine the case where κ is a binary classifier, i.e., where $V_d = \{0, 1\}$. As defined in Section 5.3.1, the

Quantity	Name	Description
$C(0, 0)$	TN	True negatives.
$C(0, 1)$	FP	False positives.
$C(1, 0)$	FN	False negatives.
$C(1, 1)$	TP	True positives.
$TP/(TP + FN)$	Sensitivity	$\Pr(\hat{d}_\kappa(x) = 1 \mid d(x) = 1)$
$TN/(TN + FP)$	Specificity	$\Pr(\hat{d}_\kappa(x) = 0 \mid d(x) = 0)$
$TP/(TP + FP)$	Positive predictive value (PPV)	$\Pr(d(x) = 1 \mid \hat{d}_\kappa(x) = 1)$
$TN/(TN + FN)$	Negative predictive value (NPV)	$\Pr(d(x) = 0 \mid \hat{d}_\kappa(x) = 0)$

Table 7.1: Names given to the entries and derived quantities from 2×2 confusion matrices.

corresponding decision classes will be denoted X_0 and X_1 .

We start by decomposing κ into realizing two functions ϕ and θ so that $\hat{d}_\kappa(x) = \theta(\phi(x))$, and refine Equation 7.1 as shown below. The function ϕ maps an object x to a continuous estimate in the interval $[0, 1]$, where $\phi(x)$ is to be interpreted as the classifier's certainty that x belongs to decision class X_1 . The function θ interprets the output of ϕ , and decides how the mapping into $\{0, 1\}$ from the intermediate representation $[0, 1]$ is to take place.

$$\hat{d}_\kappa : U \xrightarrow{\phi} [0, 1] \xrightarrow{\theta} \{0, 1\} \quad (7.6)$$

The function ϕ is what is usually associated with a given classifier, and is what most machine learning paradigms implement, whether it being neural networks, decision trees, rule sets, logistic regression equations or something else. Letting $certainty(x, X_i)$ denote the classifier's degree of certainty that x belongs to decision class X_i , we have the following:

$$\phi(x) = certainty(x, X_1) \quad (7.7)$$

It is typically desirable for $\phi(x)$ to estimate the probability $\Pr(d(x) = 1 \mid x)$, as will be discussed in Section 7.4.2.

The function θ interprets $\phi(x)$ and decides on how this is to be converted into a final decision value. Here, θ will be assumed to be a simple threshold function that outputs decision value 1 if $\phi(x)$ is above a certain threshold $0 \leq \tau \leq 1$, and decision value 0 otherwise:

$$\theta(\phi(x)) = \begin{cases} 1 & \text{if } \phi(x) \geq \tau \\ 0 & \text{otherwise} \end{cases} \quad (7.8)$$

Other definitions of θ might under some circumstances be appropriate, for instance if our application requires the classifier to express 'doubt' as a separate classification.

7.4.1 Realizing ϕ

As mentioned, different classifiers may realize the certainty function ϕ in radically different fashions. For decision rules as defined in Section 6.3, variants of the voting procedures explored in Section 6.4 may be used. Using, e.g., Equation 6.12 and normalizing by all votes cast, for binary classifiers the certainty function reduces to Equation 7.9.

$$\phi(x) = \frac{\text{votes}((d = 1))}{\text{votes}((d = 0)) + \text{votes}((d = 1))} \quad (7.9)$$

7.4.2 Discrimination and Calibration

Two of the main performance dimensions along which binary outcome classifiers can be evaluated is *discrimination* and *calibration*. Calibration deals with probability estimation, while discrimination deals with classificatory abilities.

Discrimination measures how well the classifier is able to separate objects in decision class X_0 from objects in decision class X_1 . Examples of measures of discrimination are accuracy and the area under ROC curves, further discussed in Section 7.4.5. Discrimination is in many ways a natural and intuitive performance measure to focus on, as it indicates how good a classifier is at doing what it was designed to do, namely at guessing the correct value for $d(x)$ when presented with object x .

So-called *calibration-in-the-large* measures how close the average intermediate model output is to the average actual outcome, computed on the basis of the whole sample. Calibration-in-the-large, also referred to as *bias*, thus gives an overall picture of whether a model is “optimistic” or “pessimistic”, since it signals if the model outputs are systematically high or low. However, calibration-in-the-large is not very useful in practice since a model may be perfectly calibrated-in-the-large, and yet provide no information.³

A classifier is considered well *calibrated-in-the-small* when cases assigned a certainty value of $\phi(x)$ actually yield outcome 1 approximately $\phi(x) \times 100\%$ of the time. Hence, calibration-in-the-small measures how well the function ϕ realizes a probability estimate. In the following, calibration-in-the-small will simply be referred to as calibration.

Note that calibration is different from discrimination. A classifier may discriminate well, but be badly calibrated. Conversely, a classifier may be well calibrated, but discriminate poorly. A good discussion of calibration and discrimination can be found in [136].

A short comment on terminology might be in order. Discrimination and calibration are by no means the only names that these issues are known under. For instance,

³Consider for example the “default” model that for all inputs simply outputs the prevalence of disease.

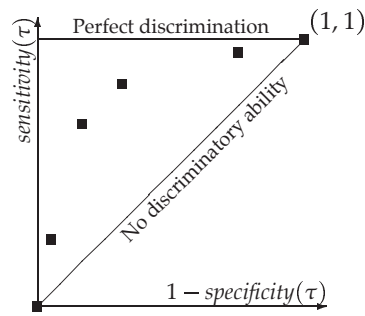


Figure 7.1: Points on an example ROC curve. The points are typically connected with straight line segments to form a curve. Each point corresponds to a different threshold value τ . A straight line from $(0, 0)$ to $(1, 1)$ indicates no discriminatory ability, i.e., the area under the curve is 0.5. The ideal ROC curve has an area under the curve of 1, i.e., it is a step function with segments from $(0, 0)$ to $(0, 1)$ to $(1, 1)$.

discrimination is sometimes referred to as *resolution*, and *imprecision* is some places used instead of calibration. Hand [74] devotes a small section in his book to review and comment on some of the different terms that are employed in the literature.

7.4.3 ROC Curves

A *receiver operating characteristic* (ROC) curve is a graphical representation of discrimination. The analysis of ROC curves is a classic methodology from signal detection theory [216] that is common in medical diagnosis [75], and that has recently begun to be used more generally in the machine learning field [173, 174].

As can be seen from Equation 7.8, the output of a classifier κ depends on a selected threshold value τ . An ROC curve captures the behavior of κ as the threshold τ is varied across the full spectrum of possible values. A very important and desirable feature of an ROC curve is that it describes the predictive behavior of a classifier independent of class distributions and classification error costs. There is a rich literature on the field of ROC analysis.

For each different value of the threshold τ we may obtain a different 2×2 confusion matrix C when we apply κ to the objects in an information system. Let the sensitivity and specificity of C as defined in Section 7.3 be denoted by $sensitivity(\tau)$ and $specificity(\tau)$. An ROC curve is defined through a collection of points as shown in Equation 7.10. The points $(0, 0)$ and $(1, 1)$ are also included as these points correspond to the situations where all objects are blindly classified as belonging to the same decision class. An example ROC curve is drawn in Figure 7.1.

$$ROC = \bigcup_{\tau} \{(1 - specificity(\tau), sensitivity(\tau))\} \cup \{(0, 0), (1, 1)\} \quad (7.10)$$

An ROC curve is said to *dominate* another ROC curve if it lies consistently above

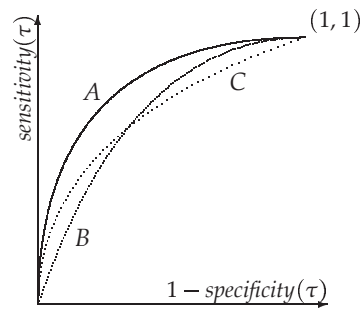


Figure 7.2: ROC curve A dominates both ROC curves B and C . Curve B does not dominate curve C , nor does C dominate B .

it in ROC space, illustrated in Figure 7.2. If the ROC curve for a classifier κ_1 clearly dominates the ROC curve of another classifier κ_2 , then it is safe to say that κ_1 is better than κ_2 . However, this is seldom the case since the ROC curves usually cross each other, in which case the choice of classifier might be unclear.

It should be noted that even though a binary classifier or test is better than another test with respect to an ROC analysis, then the better test is not necessarily the test that would be preferred in practice. The reason for this is that there are additional non-performance aspects of conducting a test that a plain ROC analysis does not take into account, such as the cost of acquiring the information the test is based on. For example, in a medical setting, some attributes may involve expensive drugs or therapy or incur considerable discomfort for the patient.

The ROC Convex Hull Method

When comparing several classifiers and plotting their respective ROC curves, the curves will in practice almost always cross each other somewhere, i.e., none of the classifiers will be optimal under all circumstances. Provost and Fawcett [173] propose a method for determining from ROC curves which classifier is optimal, for various class and cost distributions. Schematically, the process is as follows:

1. Compute the convex hull of all the points on the ROC curves. If a classifier does not contribute to a point on the hull, then that classifier can never be optimal.
2. Encode the class and cost distribution as a line and superimpose this as a tangent on the hull. The classifier that contributes to the line/hull intersection point is the best classifier under that particular class and cost distribution.
3. Tabulate under which class and cost distributions that the hull-contributing classifiers are optimal.

Interpreting the ROC Slope

If we examine how the axes in an ROC curve are defined and substitute Equation 7.8 and Equation 5.35 into their definitions from Section 7.3, we obtain the following:

$$\begin{aligned}
 \frac{\text{sensitivity}(\tau)}{1 - \text{specificity}(\tau)} &= \frac{\Pr(\hat{d}_\kappa(x) = 1 \mid d(x) = 1)}{1 - \Pr(\hat{d}_\kappa(x) = 0 \mid d(x) = 0)} \\
 &= \frac{\Pr(\phi(x) \geq \tau \mid x \in X_1)}{1 - \Pr(\phi(x) < \tau \mid x \in X_0)} \\
 &= \frac{\Pr(\phi(x) \geq \tau \mid x \in X_1)}{\Pr(\phi(x) \geq \tau \mid x \in X_0)} \tag{7.11}
 \end{aligned}$$

An ROC curve is thus a plot of the cumulative distribution function for $\phi(x)$ in the X_0 subpopulation against that of the X_1 subpopulation. Consequently, the local slope of the ROC curve can be interpreted as a likelihood ratio, by computing the derivatives of the counter and denominator of Equation 7.11 with respect to τ .

Hilden [81] exploits this relationship by scaling the ROC coordinate axes by various factors that represent such things as prevalence and measures of utility, and discusses how the local slopes of the scaled ROC curves can then be interpreted. For details, see [81]. Hilden also discusses issues relating to the concavity and convexity of ROC curves.

Interpreting the ROC Integral

The area under the ROC curve (AUC) is a measure of how well the classifier is able to discriminate objects in X_0 from objects in X_1 . AUC is usually computed using the trapezoidal rule for integration, but parametric methods that produce smooth curves and compute areas from these are also in use. An AUC of 0.5 represents no discriminatory ability, while an AUC of 1 represents perfect discrimination. Note that due to the shape of a typical ROC curve, the trapezoidal rule is prone to slightly underestimate the true area.

$$\text{AUC} = \int_0^1 \text{sensitivity}(\tau) d\text{specificity}(\tau) \tag{7.12}$$

The AUC can be shown to have a nice and intuitive probabilistic interpretation: Let $S_\kappa(\square) \subseteq X_0 \times X_1$ be defined as below, where “ \square ” denotes a comparison operator. A pair (x_0, x_1) is said to be *concordant* if it is a member of $S_\kappa(<)$, and *discordant* if it is a member of $S_\kappa(>)$. The set $S_\kappa(=)$ constitutes the set of *ties*. Obviously, it is desirable for a pair to be concordant. Under the assumption that tie-resolution results in half the ties being correctly ranked, the so-called *c-index* [77] equals the AUC estimated using the trapezoidal rule of integration. The *c-index* measures the probability that, given a

pair (x_0, x_1) randomly drawn from $X_0 \times X_1$, the function ϕ realized by classifier κ will rank x_0 and x_1 correctly, i.e., define a concordant pair.

$$S_\kappa(\square) = \{(x_0, x_1) \in X_0 \times X_1 \mid \phi(x_0) \square \phi(x_1)\} \quad (7.13)$$

$$c\text{-index} = \frac{|S_\kappa(<)| + \frac{1}{2}|S_\kappa(=)|}{|X_0||X_1|} \quad (7.14)$$

Statisticians may recognize the c -index as being the parameter of the Mann-Whitney-Wilcoxon rank sum test.

Selecting a Threshold

An ROC curve displays a model's discriminatory performance across a spectrum of thresholds. But if a classifier is to be implemented in practice, a value for the threshold must be decided upon. How this value should be selected is obviously a function of how one weighs the cost of false positives against the cost of false negatives. If the costs are equally weighted, the threshold that produces the "northwestern-most" point on the ROC curve, i.e., the point closest to $(0, 1)$, is an intuitive candidate. However, if we can quantify the costs and know the prevalence of disease, we can compute the threshold τ^* that minimizes the expected total cost as shown below. If π_i denotes the a priori probability that x is a member of X_i and misclassification incurs a cost c_i , the optimal threshold τ^* can be computed as follows [74]:

$$\tau^* = \arg \min_{\tau} \{ \pi_0 c_0 (1 - \text{sensitivity}(\tau)) + \pi_1 c_1 (1 - \text{specificity}(\tau)) \} \quad (7.15)$$

Hand [74] briefly relates how τ^* can be found by considering the local slope of the ROC curve, and how this can be used to identify a range of "good" threshold values.

7.4.4 Calibration Plots

A *calibration plot* is a graphical method for assessing how well calibrated a binary classifier is. Points in a calibration plot are generated as described below. Table 7.2 gives a small example of this process.

1. Sort the pairs $(d(x), \phi(x))$ according to the value of $\phi(x)$.
2. Partition the sorted vector of pairs into g groups.
3. Compute the average actual outcome \bar{d}_i and the average classifier output $\bar{\phi}_i$ for each group i , and add $(\bar{d}_i, \bar{\phi}_i)$ to the plot.

Group	$d(x)$	$\phi(x)$
	0	0.04
1	0	0.21
	0	0.41
$(\bar{d}_1, \bar{\phi}_1)$	0.00	0.22
	1	0.46
2	0	0.51
	0	0.59
$(\bar{d}_2, \bar{\phi}_2)$	0.33	0.52
	1	0.61
3	1	0.75
	1	0.80
	1	0.92
$(\bar{d}_3, \bar{\phi}_3)$	1.00	0.77

Table 7.2: Generating points in a calibration plot. In this small example there are 10 $(d(x), \phi(x))$ pairs that are sorted according to $\phi(x)$ and form $g = 3$ groups. Each group defines a plot point by computing the averaged values within each group. Sometimes group sums are plotted instead of group averages.

If the classifier is well calibrated, the points in the calibration plot should tend to lie around the 45-degree line that crosses through $(0, 0)$. An example calibration plot can be found in Figure 7.3.

How to select the number of groups g is not entirely apparent. Usually, a predetermined number is simply chosen at will. Furthermore, all groups are usually set to have approximately the same size, but having variable-sized groups is also a possibility [84].

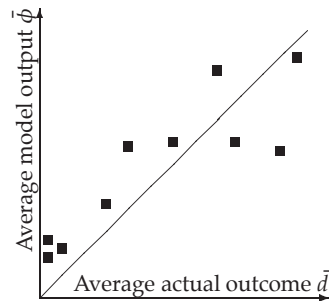


Figure 7.3: An example calibration plot with $g = 10$ groups. For a well calibrated classifier, the points should be close to the 45-degree line through $(0, 0)$.

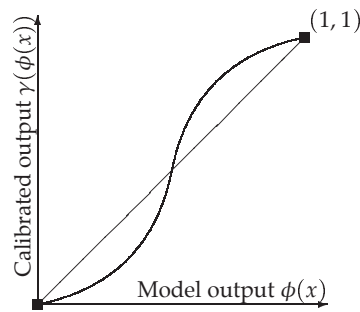


Figure 7.4: An example calibration function γ . Here, γ pushes the original model output $\phi(x)$ away from the center and towards the extreme values 0 and 1. If ϕ has a tendency to densely cluster the object certainties around a mean value, such a choice of γ may be appropriate.

Calibrating a Model

A model that may not be very well calibrated can in principle almost always be transformed into a model that is fairly well calibrated, while preserving the model’s discriminatory abilities. This can be done by introducing a monotone strictly increasing function γ as shown in Equation 7.16, and using this to remap $\phi(x)$. Hence, we can imagine $\gamma(\phi(x))$ as being a calibrated version of our classifier.

$$\hat{d}_\kappa : U \xrightarrow{\phi} [0, 1] \xrightarrow{\gamma} [0, 1] \xrightarrow{g} \{0, 1\} \quad (7.16)$$

The purpose of the “calibration function” γ is to stretch or compress the function ϕ in such a way that the transformation results in a better calibrated model, under the constraint that the discriminatory ability of the calibrated model does not worsen. If γ is a monotone strictly increasing function, then the ROC curve will remain unchanged. Figure 7.4 displays an example calibration function γ .

Although in principle a model can be calibrated a posteriori, it may in practice not be clear how to construct a suitable calibration function γ . An intuitive approach is to compute the linear regression line of the points in the calibration plot, and to let γ be the simple function that would “tilt” this line to equal the 45-degree line that crosses through $(0, 0)$. This approach is explored by Vinterbo and Ohno-Machado [230].

7.4.5 Performance Measures

Section 7.2 discussed *how* reliable performance estimates could be gathered, but was not specific as to *which* performance measure that should be gathered. This section discusses some popular performance measures and dimensions along which classifier performance may be assessed.

Accuracy and Risk

Two measures for assessing a classifier's discriminatory performance are *accuracy* and its generalization, *risk*. The proportion of correctly classified objects, defined by Equation 7.5, is called the accuracy of the classifier, and is by far the most popular performance measure in the machine learning literature. More often than not, accuracy (or *error rate*, defined as one minus accuracy) is the only performance measure reported in comparison studies. Accuracy is an intuitive quantity to relate to, but its use as the only performance measure is questionable [174]. The main reason for this is that the measure does not capture situations where different types of classification errors have different costs, nor does it adjust for skewed outcome class distributions.

Bayes decision rule, given by Equation 7.18, states that an object x should be assigned to the most probable decision class when x is given. Bayes decision rule is intuitive, and can be shown to be theoretically optimal with respect to maximizing the accuracy, or, equivalently, minimizing the error rate [179, 185]. However, in practice, the Bayes error rate can rarely be obtained. Doing so would require perfect knowledge of all distributions and conditional probabilities involved, something that is not practically achievable.⁴

$$\hat{d}_\kappa(x) = \arg \max_k \Pr(d(x) = k | x) \quad (7.18)$$

An aspect that Equation 7.18 does not take into account is that not all errors are equal. Making a wrong decision of a certain kind may be dramatically more costly than making a different type of decision error. A common way of making up for this is to introduce a *loss function* L , where $L(j, k)$ denotes the loss or cost incurred by making decision j when the true decision class is k . The risk function for a classifier κ is the expected loss when using it, as a function of the unknown decision class k . Bayes decision rule for a general loss function is given by Equation 7.19. Again, since a classifier κ can only hope to form a fair approximation of $\Pr(d(x) = k | x)$, the theoretically minimal risk is rarely obtainable in practice.

$$\hat{d}_\kappa(x) = \arg \min_k \sum_j L(j, k) \Pr(d(x) = k | x) \quad (7.19)$$

With a loss function such that $L(k, k) = 0$ and 1 otherwise, Equation 7.19 reduces to Equation 7.18. Such a loss function is called a *0/1 loss function*.

Note that a binary classifier as defined in Section 7.4 with a fixed threshold for θ of $\tau = 0.5$ amounts to emulating Equation 7.18. Other threshold values amounts to

⁴The so-called *naive Bayes rule* approximates $\Pr(d(x) = k | x)$ as below. The naive Bayes classifier often works very well in practice, and excellent classification may be obtained even when the probability estimates contain large errors [46].

$$\Pr(d(x) = k | x) = \frac{\Pr(d(x) = k)}{\Pr(x)} \prod_{a \in \mathcal{A}} \Pr((a = a(x)) | d(x) = k) \quad (7.17)$$

incorporating some kind of loss information, and hence emulates the more general Equation 7.19.

Domingos [45] proposes a method for using the loss function L together with a bagging procedure to relabel the training examples in such a way that the relabeled set can be trained by a procedure that focuses on minimizing error rate. The error rate of the relabeled data is then hopefully similar to the risk of the original data. An obvious advantage of such a “wrapper” approach is that the actual model induction technique does not have to be modified or tailored to take loss information into account.

The Area Under the ROC Curve

The AUC value derived from an ROC curve is a measure of a model’s discriminatory performance. Collapsing a full ROC curve into a single-number statistic necessarily results in a loss of distinction of some kind. For example, several different curves may have the same AUC value, and a focus on the AUC alone glosses over the fact that the classifiers may both be optimal under different operating conditions. Still, the AUC is generally accepted as the best ROC-derived single-number statistic to use for performance assessment.⁵

The Brier Score

The *Brier score* [20] is defined as the average squared difference between the classifier’s raw output value $\phi(x)$ and the actual binary decision value $d(x)$. The Brier score is also referred to as the *probability score*.

$$B = \frac{1}{|U|} \sum_{x \in U} (\phi(x) - d(x))^2 \quad (7.20)$$

The Brier score B carries information about both calibration and discrimination, and can be decomposed in several ways. A popular decomposition is given in Equation 7.21, usually attributed to Murphy [128]. The term \bar{d} denotes the average actual outcome of objects in U , while CI and DI denote indices of calibration and discrimination respectively.

$$B = \bar{d}(1 - \bar{d}) - \text{CI} + \text{DI} \quad (7.21)$$

Arkes et al. [9] discuss the Murphy decomposition, and propose an alternative decomposition of the Brier score that lends itself nicely to graphical visualization in a *covariance graph*. Their decomposition involves readily interpretable components.

⁵However, the unquestioned use of the area under ROC curves is not without its critics. Hilden [81] argues that, for clinicians, the *c*-index or AUC may give “the right answer to the wrong question” since they are in practice almost never asked to decide which of two cases is diseased and which is non-diseased.

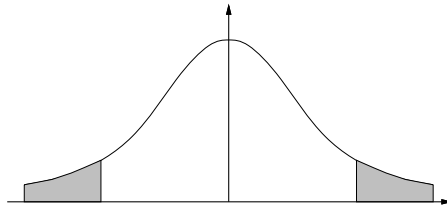


Figure 7.5: Statistical hypothesis testing (two-sided). Here, the test statistic Z is assumed to follow a standard Normal distribution under the null hypothesis H_0 that the two observed performance estimates are equal. For a two-sided test, the alternative hypothesis H_1 is that the two observed performance estimates are not equal. If we observe a realization z of Z that deviates significantly from 0, we reject H_0 and conclude that the two performance estimates are indeed different. The p -value indicates the probability that we erroneously reject H_0 , i.e., the probability of a type I error.

7.4.6 Statistical Hypothesis Testing

In typical classifier comparison experiments, two or more classifiers are trained on the same training set and applied to the same test set, and a performance measure for each classifier is estimated. But if one classifier results in one performance estimate and another classifier results in another performance estimate, how much must the two estimates differ before one can say that the performances are significantly different in a statistical sense? Any such claim should always be accompanied by a p -value that indicates how statistically valid the claim is.

The starting point for performing hypothesis testing is a test statistic that under some null hypothesis H_0 has a known distribution. For example, H_0 could be that two performance estimates really are equal, with the alternative hypothesis H_1 being that they are different. The test statistic Z is a stochastic variable, and from our observations we can compute a realization z of Z . If z has a value that under H_0 would be very unlikely to observe, then we are prone to reject H_0 . A p -value denotes the probability that we erroneously reject H_0 .⁶ Obviously, we would like this probability p to be small before we conclude that H_0 is to be rejected, typically $p < 0.05$ by convention. Figure 7.5 displays p -values graphically. A brief discussion of the use of p -values in decision making can be found in [239].

This section reviews a handful of statistical tests that can be used to compare discrimination and calibration between pairs of binary classifiers.

McNemar's Test

With two classifiers κ_1 and κ_2 applied to the same set of labeled objects using fixed suitable threshold values τ_1 and τ_2 for θ for each classifier, *McNemar's test* is the ap-

⁶The situation of erroneously rejecting H_0 is called a *type I error* in the statistical literature. The other situation, where we erroneously accept H_0 , is called a *type II error*.

appropriate statistical test to perform to detect statistically significant differences in accuracy [8, 43, 44, 58, 179, 183].

McNemar's test counts the number of objects that κ_1 classified wrongly but that κ_2 classified correctly, and the number of objects that κ_1 classified correctly but that κ_2 classified wrongly. The following small table is then obtained:

		\hat{d}_{κ_1}	
		error	correct
\hat{d}_{κ_2}	error	n_a	n_b
	correct	n_c	n_d

Under the null hypothesis H_0 that there is no difference between the accuracies of κ_1 and κ_2 , we would expect n_b and n_c to be equal. An exact⁷ test can be made since then n_b would follow a Binomial distribution with parameters $(n_b + n_c, \frac{1}{2})$. However, such a distribution can be very well approximated with the more easily manageable Normal distribution, even for quite small samples. McNemar's test uses the continuity-corrected Z statistic below, which under H_0 follows a standard Normal distribution. In some texts the χ_1^2 -distributed quantity Z^2 is employed instead.

$$Z = \frac{|n_b - n_c| - 1}{\sqrt{n_b + n_c}} \sim N(0, 1) \quad (7.22)$$

Computing the observed value for Z enables us to perform a hypothesis test in the usual manner.

Altman [8] discusses McNemar's test in greater detail, and Salzberg [183] recommends that McNemar's test (or the exact Binomial variant on that test) should always be used when comparing the accuracy of classification algorithms.

Hanley-McNeil's Test

To statistically compare AUC values of two ROC curves derived from the same set of cases, *Hanley-McNeil's test* can be used.

Computations of the AUC of a classifier should always be accompanied by an estimate of the variability of that estimate. Hanley and McNeil [75] provide the following formula for the standard error $SE(AUC)$ of the AUC estimate computed using the trapezoidal rule of integration:

$$SE(AUC) = \sqrt{\frac{1}{|X_0||X_1|} \sqrt{\frac{AUC(1 - AUC) + (|X_1| - 1)(Q_1 - AUC^2) + (|X_0| - 1)(Q_2 - AUC^2)}{2}}} \quad (7.23)$$

⁷An exact test has precisely the distribution claimed under H_0 .

The quantities Q_1 and Q_2 in Equation 7.23 are distribution-specific quantities. Hanley and McNeil claim that very good approximations can be obtained by Equation 7.24, and they also outline a more complex and non-parametric way of computing $SE(AUC)$.

$$Q_1 = \frac{AUC}{2 - AUC} \quad Q_2 = \frac{2AUC^2}{1 + AUC} \quad (7.24)$$

Using Equation 7.23, one can compute the smallest sample size one needs for obtaining a sufficient degree of statistical precision.

When we want to test the hypothesis that two estimated AUC values derived from the same cases are really different, we have to take into account that the AUC estimates are correlated since the data material is the same. If we know that one AUC value for one classifier is very low, this affects our knowledge of the AUC estimates for the other classifiers. Hanley and McNeil [76] provide a framework for statistically comparing two different AUC values derived from the same cases. The main contribution of their paper is a tabulation of values of the correlation quantity r in Equation 7.25, for different values of AUC_1 , AUC_2 and the computed correlations⁸ between the two classifiers' raw output values $\phi(x)$ for objects in the two decision classes X_0 and X_1 .

$$SE(AUC_1 - AUC_2) = \sqrt{\frac{SE^2(AUC_1) + SE^2(AUC_2) - 2rSE(AUC_1)SE(AUC_2)}{2}} \quad (7.25)$$

The perhaps most important purpose of assessing the variability of the observed AUC differences, is so that we can construct a statistic to use for hypothesis testing. With the above definition of $SE(AUC_1 - AUC_2)$, Hanley and McNeil use the statistic Z defined below.

$$Z = \frac{AUC_1 - AUC_2}{SE(AUC_1 - AUC_2)} \sim N(0, 1) \quad (7.26)$$

Under the null hypothesis H_0 that the two ROC areas really are equal (and that the true distribution actually is Normal), the test statistic Z follows a standard Normal distribution. We can then reject H_0 , i.e., conclude that the two ROC areas are not equal, if our observed value for Z is such that it does not lie within the allowed range we set up according to our desired level of significance.

Quantitative Assessment of Calibration

Calibration is usually only assessed qualitatively by visually inspecting calibration plots or covariance graphs, but quantitative assessment of calibration is also possible.

⁸Computed using either Pearson's product-moment correlation or Kendall's tau. See Hanley and McNeil [76] for details.

However, methods for this have often originally been developed with logistic regression models in mind, and carrying such statistics directly over to other model types may yield unpleasant surprises in applied research. For instance, undefined division-by-zero situations may occur. This because formulae may involve dividing by $\phi(x)$ or its complement, and logistic regression models can never output identically 0 or 1 but approach these extreme values asymptotically. For other model paradigms such as rule sets or decision trees, these extreme values are indeed possible outputs.

Hosmer-Lemeshow's test [83], originally intended for use with logistic regression models, constructs a statistic from the points in the calibration plot.⁹ Simulations indicate that this statistic is well approximated by a χ^2 -distribution with $g - 2$ degrees of freedom under the null hypothesis that the model is appropriate calibration-wise. The Hosmer-Lemeshow statistic can thus be used to assess quantitatively whether or not the points in the calibration plot deviate from the 45-degree line through $(0, 0)$ in a statistically significant fashion.

However, there are several pitfalls with using Hosmer-Lemeshow's test other than the ones already described. The statistic is only applicable if the groups sizes are above 5, and is undefined if the sum of model outputs within a group is zero. Furthermore, if the model outputs tend to be either small (< 0.1) or large (> 0.9), then the test should be used with caution [84].

Brier Score Tests

Statistics for assessing Brier scores often suffer from the same drawback as the Hosmer-Lemeshow statistic, namely that they have been developed with methods in mind that cannot output identically 0 or 1 but approach these extreme values asymptotically. However, if a model does not produce such extreme values on an analyzed dataset, such statistics may be useful after all. Bloch [16] reviews several statistics that can be used to test whether the actual outcomes $d(x)$ are compatible with the set of model outputs $\phi(x)$. Probabilities of some events must be explicitly chosen or known to use such approaches, however.

Statistics for comparing two Brier scores obtained from the same set of cases against each other exist, enabling one to test whether or not one model's Brier score is better than the Brier score of another model. Bloch [16] and Redelmeier et al. [178] discuss such statistics and the corresponding tests in detail.

Miscellaneous

Both Hanley-McNeil's and McNemar's tests assume that the two classifiers that are to be statistically compared have been applied to the same set of cases, and that we have their corresponding observed outputs available. This might not always be the

⁹Actually, the Hosmer-Lemeshow statistic operates on the groups sums instead of on the group averages.

situation, and if the datasets only partially overlap the methods are not applicable. Metz et al. [121] propose an algorithm that allows an ROC analysis to be done, even with only partially-paired data.

Other approaches, including non-parametric methods, than the one described for comparing correlated AUC estimates are possible, although such methods are less simple and immediate. A non-parametric alternative to Hanley-McNeil's test is provided by DeLong et al. [40].

AUC estimates are uncertain since the computed points on the ROC curve have an inherent degree of uncertainty associated with them. Another approach to ascertaining the uncertainty of the AUC estimate is to compute confidence intervals for the sensitivities and specificities, and thus obtain a "confidence envelope" for the ROC curve, i.e., a pair of curves that surround the ROC curve from above and below for which we can be certain "enough" that the true ROC curve lies within. Bounds for the AUC can thus be assessed via the bounds of the ROC curve. Computing confidence bounds for ROC curves is explored by Schäfer [184].

ROC analysis for more than two outcome classes can be imagined. Then, in the three-class case, an ROC curve would generalize to an ROC surface, with the volume under this surface corresponding to the AUC. This is further explored by, e.g., Dreiseitl et al. [48].

Lastly, it should be pointed out that combining statistical hypothesis testing with systematic partitioning methods as described in Section 7.2.1 is not entirely straightforward and should be done with caution. One reason for this is that the training and/or testing sets for the splits may partially overlap, thus introducing dependencies across the statistics computed per split. Dietterich [43, 44] analyzes this scenario for some popular statistics for comparing error rates, and shows through simulations that erroneous conclusions are likely to be drawn if inter-split dependencies are ignored. Dietterich also proposes a statistic that may be used for combining error rate comparisons with systematic partitioning methods, based on doing 2-fold CV five times. Alpaydin [7] improves this statistic further.

Part III

Tools

Chapter 8

The ROSETTA System

Sections of this chapter have previously appeared as [145, 147].

8.1 Introduction

Fields concerned with empirical modelling necessarily have a high experimental content, both because the sought after relationships are unknown in advance and because real-world data is often noisy and imperfect. The overall modelling process thus typically consists of a sequence of several steps that all require various degrees of tuning and fine-adjustments. Moreover, it may not beforehand be obvious which steps in the modelling pipeline that are required, nor which of several alternative algorithms that should be chosen for each step. As a result, the process of constructing a model is an iterated waterfall cycle with possible backtracking on the individual steps. It is therefore important to have a set of tools available that render possible this type of flexible experimentation. However, a complete model construction and experimentation tool must comprise more than a collection of clever algorithms in order to be fully useful. It is needed to set the tools in an environment such that intermediate results can be viewed and analyzed, and decisions for further processing made. Basically, an environment to interactively manage and process data is required.

This chapter introduces the *ROSETTA*¹ *system*, a comprehensive set of software components for discernibility-based data analysis. Rough sets and methods based on discernibility have gained significant scientific interest as a framework for data mining and KDD [126, 168–170], but successful research in this field undoubtedly requires good cooperation between theoreticians and practitioners. This interface can be enhanced by providing sophisticated tools and environments that support all aspects of the iterative nature of model construction and assessment. In response to these needs,

¹The *Rosetta stone* is a stone slab found in 1799 near the Egyptian town of Rosetta. Bearing parallel inscriptions in Greek, Egyptian hieroglyphic and demotic characters, it made possible the decipherment of ancient Egyptian hieroglyphics. The name ROSETTA can also be construed as an acronym, e.g., for a *Rough Set Toolkit for Analysis of Data*.

the process of using rough sets for KDD has been investigated and process patterns typical to rough set KDD experiments been established. The ROSETTA system has been designed and implemented as a result.

It is practical to differentiate between the *computational kernel* and the *front-end* of ROSETTA. The computational kernel is a general C++ class library for KDD within the rough set methodology, and offers an advantageous code base for researchers to quickly assemble and try out new algorithms and ideas. The front-end is a state-of-the-art graphical user interface (GUI) running under Windows NT/98/95. Together, the kernel and the front-end constitute a powerful vehicle for practical discernibility-related research and applications.

The ROSETTA kernel can be employed in two modes: Together with the GUI front-end, and as a stand-alone command-line program. The former enables access to the computational engine in a user-friendly environment, while the latter enables ROSETTA to be used as a computational engine called from elsewhere, e.g., from Perl scripts.

ROSETTA is probably the most complete, flexible and advanced rough set software system of its kind currently available. The system also encompasses several stand-alone utility programs that operate directly on output from the main ROSETTA program, e.g., programs for statistical hypothesis testing.

This chapter gives a brief rundown of ROSETTA and some of its features. Section 8.2 outlines some of the background for initiating development of the ROSETTA system, while Section 8.3 gives a brief overview of the ROSETTA GUI. Section 8.4 presents how the typical discernibility-based KDD modelling methodology is supported by ROSETTA.

8.2 System Justification

The initial impetus to develop a discernibility-based data analysis tool at this time came from a perceived discrepancy between the theoretical developments in the rough set field and the scope and availability of tools for model realization. A set of software components was called for to help bridge that gap. Furthermore, for the purpose of this thesis, a flexible tool was needed to conduct experiments with medical applications in mind.

Software systems for performing rough set computations exist, but are scarce as the field is still relatively young. Lists of known systems are compiled by Polkowski and Skowron [170] and by Komorowski et al. [105]. Rather than basing ourselves on one or more of these, the need to develop ROSETTA arose for the following main reasons:

- Most existing systems tended to be highly specialized towards a particular rough set algorithm or stage in the KDD process, and hence did not allow the degree of flexibility and generality that was desired.
- Source code was generally not available, hence making any novel or non-standard

analyses, applications and extensions difficult. For the few systems where source code could be obtained, these were often written in interpreted languages of commercial systems. The efficiency of a compiled language was essential, and becoming dependent on a third-party run-time environment was not desirable.

- The front-ends of most existing systems did not match up to current expectations of user-friendliness in interfaces.
- Very few of the existing systems allowed considerations to be made that are of importance when working within the medical domain.

Additionally, some C++ code comprising the core of the *Rough Set Expert System* (RSES) had been made available courtesy of the Group of Logic at the University of Warsaw, Poland [69,217]. A lot of resources had gone into developing this, and it was natural to exploit its existence to the extent possible. Using this as an initial core, the time to develop an initial operational prototype could be brought down. Today, the RSES library functions as an optional and value-adding ROSETTA component.

8.3 GUI Overview

The ROSETTA front-end runs under 32-bit Windows operating systems on Intel platforms, and offers an environment in which the user in a simple way can view and keep track of the individual data items in an analysis project. Emphasis has also been put on making it easy to manipulate the data items and initiate computations. The GUI is designed according to a strict object-oriented philosophy.

Briefly, features include:

- *Project trees*: Each item in a data analysis project is represented by its own icon specific to its type, and each project organizes these icons in a tree. The topology of the tree conveys how the data items relate to each other in an intuitive and immediate way. A branch from a parent item to a child item signifies a direct derivation relationship, while siblinghood typically represents alternative hypotheses being explored in the KDD process. A snapshot of a sample project tree is provided in Figure 8.1.
- *Data views*: All data items in project trees can be viewed in individual windows, typically in grid views. Routed through data dictionaries, all data is displayed to the user in terms from the modelling domain. Together with the project trees, a collection of such views embody a comprehensive workspace. A snapshot of a sample workspace is provided in Figure 8.2.
- *Context-sensitive pop-up menus*: Most GUI objects, e.g., icons in project trees and columns or rows in data views, are right-clickable and provide their own local pop-up menus. This gives intuitive and easy access to the set of operations currently applicable to that particular object.

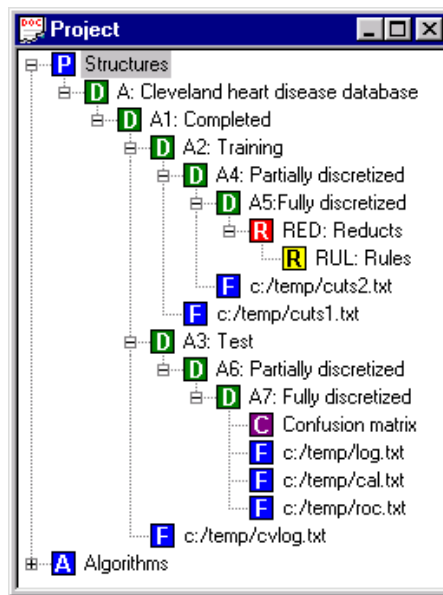


Figure 8.1: An example ROSETTA project tree. Each data item is represented as a separate icon, with the branches in the tree reflecting how the items relate to each other. Right-clicking an icon brings up a pop-up menu for the selected object.

- *Support for drag-and-drop:* As an alternative to pop-up menus, the project tree also has support for drag-and-drop. In the tree, not only are data items represented by icons, but possible operations are iconified, too. Hence, to initiate a computation, an algorithm icon may be dragged and dropped onto a data icon, or vice versa.
- *Comprehensive parameter dialogs:* Most algorithms need to be supplied with some parameters that determine details of their behavior. Often, default parameter settings are acceptable, but for flexibility and generality expert tuning must be possible. For some algorithms, combinations of parameters are only allowed according to certain rules. ROSETTA provides comprehensive support for this process by offering advanced parameter dialogs which may adapt according to the currently specified settings.
- *Annotations:* Data items can be annotated with user comments. Also, as new data items are created or transformed, they get automatically stamped with history details, revealing how they were created, which algorithms that were applied to them, which parameter settings that were used, etc. Hence, logs documenting the modelling sessions are automatically generated, thus promoting reproducibility of experiments between researchers when data is exchanged.

The screenshot displays the ROSETTA software interface with several panels:

- Project Tree (Top Left):** A hierarchical view of the workspace containing folders for 'Structures', 'Training set', 'Approximate rules', 'Testing set', 'Confusion matrix', and 'Algorithms'.
- Training Set View (Top):** A table with columns: cp, treatres, chol, rbs, restecg, thalach, esang, outpeak, slope, ca, thal, num. It lists 15 training set entries with their respective values.
- Confusion Matrix (Middle Left):** A 2x2 matrix showing predicted vs actual results. The diagonal elements are 74 (True Positives) and 53 (True Negatives). The off-diagonal elements are 17 (False Positives) and 53 (False Negatives). Metrics include Class Accuracy (0.8187), Area (0.9188), ROC Std error (0.02094), and Thr. (0.1) (0.428).
- Approximate Rules (Middle):** A list of 58 rules with columns for Rule, LHS Support, RHS Support, LHS Accuracy, RHS Accuracy, LHS Coverage, RHS Coverage, and RIS Substit. Rules range from 1 to 58.
- Completion Matrix (Bottom Left):** A small table with columns 'No' and 'Yes' and rows 'Actual' and 'Predicted'. Values are: Actual No (74), Actual Yes (17), Predicted No (53), Predicted Yes (53).
- Progress (Bottom Right):** A status window showing 'Ready' and 'Application took 00:00:01'.
- Bottom Panel:** A large area for displaying the full text of the selected rule, including its LHS and RHS conditions.

Figure 8.2: An example ROSETTA workspace, displaying several different types of data views. A project tree, showing how all the current data relate to each other, is drawn in the top left corner. All views display data in terms from the modelling domain.

The ROSETTA GUI offers several other features, including a system for displaying detailed progress messages and intermediate results, the ability to prematurely terminate lengthy computations, and an on-line help system.

8.4 Methodological Support

This section briefly describes some of the support offered by ROSETTA at each stage in the typical KDD process. In Section 2.2.2, the process of applying rough set methods for KDD was briefly outlined. The ROSETTA system was specifically designed to support the discernibility-based modelling methodology, both with respect to algorithmic features and the cognitive aspects of the GUI. At the same time, the system should be flexible enough to grant expert users elbowroom.

For convenience, this section assumes that a predictive model is being produced, i.e., we have a designated decision attribute in our table. However, the steps are easily generalizable to general information systems, i.e., unlabeled data. For a complete account of algorithms featured in ROSETTA, see Øhrn [139].

Typically, algorithms are invoked from the pop-up menus of the icons in the project tree, e.g., by right-clicking on a “D” icon. In the following, the symbol “▶” is used to denote menu navigation.

8.4.1 Selection

▶ODBC ▶Load ▶View

This phase bears to study design. The study population is defined, a set of features is defined and a modelling target is chosen. Together, this corresponds to assembling a decision system \mathcal{A} from a data source.

We here assume that the selection phase also handles representational issues. For instance, a categorical variable a with $|V_a| = k$ can equally well be represented in the less compact form of k binary attributes.²

ROSETTA offers support for almost all kinds of relevant data sources through the *Open Database Connectivity* (ODBC) interface.³ Effectively, this enables ROSETTA to import tabular data directly from a wide variety of sources, e.g., Excel spreadsheets, text files and databases from systems such as Oracle, dBase, Access or SAS.

²The categorical attribute a can be expanded into binary ones $\{a_{v_1}, \dots, a_{v_k}\}$, where $a_{v_i}(x) = 1$ if $a(x) = v_i$ and 0 otherwise. This is actually one way of letting negations enter the picture as discussed in Section 3.4.2, since the descriptor $(a_{v_i} = 0)$ essentially equals $(a \neq v_i)$. However, the cost of introducing such binary expansions is a higher number of variables, which in turn entails an increase in the computational workload.

³ODBC is an open, vendor-neutral interface for database connectivity that provides access to a wide variety of computer systems. The ODBC programming interface enables applications to access data in *Database Management Systems* (DBMSs) that use *Structured Query Language* (SQL) as a data access standard. This enables developers to not have to target a specific DBMS. Instead, users can add modules called *database drivers* that link the application to their choice of DBMSs.

$\langle \text{edge-specification}_i \rangle$	Semantics
<code>make-reflexive</code>	$E_a \leftarrow E_a \cup \{(v, v) \mid v \in V_a\}$
<code>make-symmetric</code>	$E_a \leftarrow E_a \cup \{(v_2, v_1) \mid v_1, v_2 \in V_a \text{ and } (v_1, v_2) \in E_a\}$
<code>make-transitive</code>	$E_a \leftarrow E_a^*$
<code>make-distance r_a</code>	$E_a \leftarrow E_a \cup \{(v_1, v_2) \mid v_1, v_2 \in V_a \text{ and } v_1 - v_2 \leq r_a\}$
<code>make-complement</code>	$E_a \leftarrow V_a^2 - E_a$
<code>$v_1 \rightarrow v_2$</code>	$E_a \leftarrow E_a \cup \{(v_1, v_2)\}$
<code>$v_1 \leftrightarrow v_2$</code>	$E_a \leftarrow E_a \cup \{(v_1, v_2), (v_2, v_1)\}$

Table 8.1: A ROSETTA IDG specification for an attribute a consists of a sequence of edge specifications. This table gives an overview of how various edge specification commands alter E_a . For details, see Øhrn [139]. Before executing $\langle \text{edge-specification}_1 \rangle$, the graph has no edges, i.e., $E_a = \emptyset$. Here, E_a^* denotes the transitive closure of E_a , computed by Warshall’s algorithm [6,36]. Also, when specifying individual edges, the symbol ‘*’ can be used as a wildcard.

During data import, *data dictionaries* are automatically constructed. Such dictionaries are meta-data containing information about attributes, e.g., names, types and units. All communication between the kernel and the front-end is routed through these dictionaries, so that information to the user can be displayed in terms from the modelling domain. Data dictionaries can also be imported and exported explicitly.

Also relevant to the selection phase is *attribute masking*, which can be done from within data views. By masking away one or more grid columns, selected attributes can be made “invisible” in any subsequent calculations. A typical example for the use of this feature is if we have columns which assign unique identifiers to each row, e.g., patient names or social security numbers.

8.4.2 Preprocessing

[D►Complete](#) [D►View](#)

This phase covers the issue of data cleansing, which is generally application specific. Removing obvious outliers is one common preprocessing task. Another common task is that of *completion*, i.e., the process converting a system \mathcal{A} with missing values into a new system \mathcal{A}' without any such “holes”.

For some completion strategies, both the set of objects and the set of attributes may get altered, while other alternatives modify only one of these. ROSETTA offers several completion functions, and options currently include variations of object removal, value substitution and value expansion [70, 111]. Value substitution and expansion is also known as *data imputation* [55].

Alternatively, rather than producing a completed information system to work with, the user may opt to overload the notion of discernibility to handle missing values directly, as discussed in Section 5.2.1. ROSETTA can employ general IDGs, and supports a small language in which IDGs can be conveniently expressed. See Table 8.1 for a small glimpse of this.

Editing of tables for preprocessing purposes can also be done manually from within data views.

8.4.3 Transformation

D►Discretize

Transformation of data may take place in a wide variety of ways, and which transformations that are suitable to apply is usually dictated by the application. One possibility is to perform a general transformation on the coordinate system spanned by the condition attributes, e.g., some kind of projection of the data onto a set of rotated or skewed axes [131, 132]. Such transformations, however, may destroy or severely distort the semantics of the attributes, thus taking away some of the appeal of employing rule-based models. Presently, therefore, we only consider semantics-preserving data transformations.

The most common transformation procedure in discernibility-based KDD is that of *discretization*, which basically corresponds to defining a coarser view of the world through making the attributes' value sets smaller. For numerical attributes, we can introduce intervals which in turn may be given linguistic labels and be treated as qualitative rather than quantitative entities. For symbolic attributes, we might choose to merge categories together.

Several alternative discretization functions are currently implemented within ROSETTA, including methods based on discernibility preservation [129–131, 133], entropy minimization [47], equal frequency binning [131] and various naive approaches. Attributes can also be discretized manually, if domain knowledge for this exists.

Note that discretization may or may not be needed to carry out. If IDGs are employed to realize Equation 5.7 or if we only have binary or symbolic attributes, then discretization may be skipped.

Discretization is further discussed in Appendix D.

8.4.4 Data Mining

D►Reduce {**R**, **R**}►Filter

The purpose of the data mining step is to produce a model from our preprocessed and transformed database. In our context we decompose the data mining step into a multistep process, which in practice is often interleaved for reasons of spatial and computational efficiency. First, reducts or approximations thereof are computed through a *reduction* process. The reduct set might then be filtered down according to some criterion, and then overlaid the transformed database in order to generate a set of decision rules. The rule set might in turn also be subjected to some filtering scheme.

Current ROSETTA options for reduction include genetic algorithms [233, 245], set covering heuristics [95], singleton approaches [82], brute force [217], dynamic reducts [12, 13] and approximate hitting set approaches [233]. Support for boundary region thinning [192, 250] and tolerance relations [106, 192] via IDGs is provided.

ROSETTA also offers several options for filtering away individual reducts or rules from collections of such. In addition to basic properties such as, e.g., coverage and accuracy, filtering criteria include attribute costs, advanced quality measures [22] and classificatory performance on external hold-out databases [3–5,231].

Several of the data mining algorithms in ROSETTA can employ meta-data. In addition to IDG specifications, some algorithms can make use of *attribute cost information*, i.e., information about costs associated with obtaining the values for each attribute. This kind of information can be used to steer algorithms towards solutions that would be “cheap” to realize in practice, e.g., by letting reduct computation algorithms have a propensity towards selecting attribute subsets with low costs rather than attribute subsets with low cardinalities.

8.4.5 Interpretation and Evaluation

 View  Classify

One approach to aiding in model inspection and interpretation is to produce small and compact models through applying some kind of filtering in the data mining step. A complementary approach is to accept large models and facilitate the inspection of individual “strong” rules or patterns. The ROSETTA GUI allows individual model components to be sorted according to various properties directly in the data views.

An ensemble of decision rules can be evaluated according to how well they classify unseen objects. Current classification procedures offered by ROSETTA include several different approaches based on voting, e.g., standard voting, object tracking and weighted voting [217]. Prioritization according to rule specificity [126] is also implemented. Classification using Naive Bayes [46] is available, too.

The result of classifying a batch of objects in ROSETTA is a confusion matrix. Additionally, the system can generate detailed logs and a range of performance measures. Current options for assessing classificatory performance include ROC analysis [75, 216], calibration curves [136], Brier scores [128] and its covariance decomposition [9], plus measures directly derivable from confusion matrices such as, e.g., classification accuracy.

To aid in statistical analysis of classificatory performance, the statistical tests of Hanley and McNeil [76] and McNemar [58] are available as stand-alone utilities that operate directly on ROSETTA output.

8.4.6 Automation

  Execute

ROSETTA offers support for partial automation of lengthy and repetitive command sequences. Through a simple scripting language, serial and parallel flows of data can be defined and executed. Dedicated scripting support for k -fold CV is provided.

8.4.7 Deployment



Classifiers can be exported from ROSETTA as source code, ready for subsequent deployment. Information systems can be exported as sets of Prolog facts, while decision rules can be exported as sets of Prolog rules. This establishes a link from ROSETTA to advanced inference engines, where the rules can be utilized together with any available domain theories as part of an expert system. A set of decision rules realizing a classifier can also be exported as fully working C++ code, suitable to be embedded in external applications.

8.4.8 Miscellaneous

As ROSETTA currently does not incorporate any kind of plotting or advanced data visualization, data such as, e.g., information systems or indiscernibility relations, have to be exported to other software packages to be visualized. Currently supported formats are Matlab [120] and GraphViz [68].

Automatically compiled summary reports of the actions performed during a ROSETTA session can be exported to HTML, complete with a hyperlink representation of the current project tree.

Other ROSETTA features include support for random sampling, partitioning, and computation of set approximations within the variable precision model [249, 250].

8.5 Miscellaneous

This chapter has provided an overview of the ROSETTA system and some of its features. However, for clarity, some points should also be made as to what ROSETTA is not:

- Even though medicine has been the primary application domain during development of the system, ROSETTA is in itself a general-purpose system that is not geared towards any particular application domain. By design, the system has support for several features that are relevant for analysis of medical data, but these features can, and probably should, be employed in other domains as well.
- ROSETTA command scripts are in no way intended as a general programming language. Although several operations not originally intended supported can be achieved, command scripts are very specialized towards expressing data transformation pipelines. For general programming tasks, the ROSETTA C++ class library should be used.
- ROSETTA is not intended as a DSS in the sense of being a tool for patient-specific consultation, although this is possible. Rather, the system works with collections

of data from sets of patients and produces output that can be put to use in programs specially tailored for decision-support of that kind.

As mentioned in Section 8.2, a portion of the computational kernel originates from the RSES library. How that external legacy code technically fits into the ROSETTA kernel is discussed in Section 10.3.3 and in Appendix B.

A website has been created where various electronic resources related to the ROSETTA system can be found [180]. In addition to providing a location from where to download the system, the site includes extensive documentation, utilities, comments and tips on how to use the program and its associated utilities.

As of December 16, 1999, 2970 downloads of ROSETTA from 1286 distinct users have taken place since the system was first released in the summer of 1997. ROSETTA has received very positive feedback from its users, and has been successfully applied to problems in a wide range of domains. In addition to being used for teaching and seminars, publications reported by ROSETTA users where the system has been employed include⁴ applications in power electronics [248], analysis of medical data [25,49,209,210,231,254], satellite control [161,166], software engineering [159,160,162,163], finance [73], public policy generation [118], medical ethics [11], history of science [238], real-time decision-making [164], anthropology [51], selection of controller gains [165], environmental modelling [65], and diagnosis of rotating machinery [103]. Less application-specific publications include papers on rule filtering [3,4], generation of default rules [92] and rough data modelling [116].

⁴Publications authored or co-authored by Aleksander Øhrn have been omitted from this list. The publications are listed in no particular order. The true list of publications is believed to be longer, due to lack of feedback from some users.

Chapter 9

A ROSETTA Case Study

9.1 Introduction

This chapter provides a small case study to exemplify how ROSETTA can be applied to analyze a medical database. The goal of this example is not to obtain optimal modelling results, but rather to show how ROSETTA can be used in a typical data mining and KDD scenario. As a case study, a publicly available database on coronary artery disease is employed.

For details concerning the methods employed in this case study and other ROSETTA features, see Øhm [139].

9.2 GUI Preliminaries

Chapter 8 gave a brief presentation of the ROSETTA GUI. For this case study, the following ROSETTA GUI details are worth noting:

- A decision system can be read into a new ROSETTA project by selecting **Open...** from the main **File** menu, and will be placed immediately below the root of the **Structures** node in the project tree.
- Branches in the project tree can be expanded or collapsed by left-clicking on the “田” or “日” symbols next to the icons.
- Right-clicking on an icon in the project tree brings up a pop-up menu for that object. In the following, the symbol “▶” will be used to denote menu navigation.
- Left-clicking twice on an icon in the project tree can be used as a shortcut for viewing that object in detail.
- Grayed columns in views of decision systems indicate that the corresponding attributes are “masked away” and subsequently ignored by the ROSETTA kernel

in any analysis steps. Missing values are indicated by the string “**Undefined**”.

- Rules can be sorted directly in their views by right-clicking the column to sort by.
- To rename an object, first left-click once on its icon to select it. Then left-click once more on the icon’s label. The icon’s label is edited directly in place.
- To view progress messages and warnings, select **Messages** from the main **View** menu.

9.3 Data Material

As a case study, the Cleveland Heart Disease Database from the UCI Repository of Machine Learning Databases [15] is examined. The database has recorded information about 303 patients over 13 condition attributes of various types. Each patient also has a binary decision attribute which reveals the presence or absence of coronary artery disease. From a machine learning point of view, the goal is to induce a classifier that predicts the value of the decision attribute.

The database, summarized in Table 9.1, stems from a clinical study by Detrano et al. [42] in which 352 different logistic regression models were created and subsequently tested on data collected at other geographical sites. For further details about the medical diagnostic problem, the data material and the precise semantics of each attribute, see Detrano et al. [42].

9.4 Experimental Setup

Figure 9.1 demonstrates a typical machine learning setup. The main input to the pipeline is the decision system \mathcal{A} from Table 9.1. This is subsequently split in two parts, and a classifier is induced from one part and applied to the other in order to harvest a performance estimate.

The individual steps in Figure 9.1 are explained below. Also shown are the corresponding ROSETTA menu choices for invoking the algorithms from Figure 9.1.

C \mathcal{A} ►**Complete**►**Remove incompletes**

Produces a complete table by removing all objects from U that have a missing value for any condition attribute $a \in A$.

$$U_1 = \{x \in U \mid \forall a \in A, a(x) \neq \top\} \quad (9.1)$$

S \mathcal{A}_1 ►**Other**►**Split in two...**

Splits U_1 into two disjoint universes U_2 and U_3 .

Attribute	Description	Statistics
a_1	AGE	Age (years). 29–77 (56)
a_2	SEX	Male sex? 0.680
a_3	CP	Chest pain type: <i>Typical angina</i> 0.076 <i>Atypical angina</i> 0.165 <i>Non-anginal pain</i> 0.284 <i>Asymptomatic</i> 0.475
a_4	TRESTBPS	Resting systolic blood pressure on admission to the hospital (mmHg). 94–200 (130)
a_5	CHOL	Serum cholesterol (mg/dl). 126–564 (241)
a_6	FBS	Fasting blood sugar over 120 mg/dl? 0.149
a_7	RESTECG	Resting electrocardiographic results: <i>Normal</i> 0.498 <i>ST-T abnormality</i> 0.013 <i>LV hypertrophy</i> 0.489
a_8	THALACH	Maximum heart rate achieved. 71–202 (153)
a_9	EXANG	Exercise induced angina? 0.327
a_{10}	OLDPEAK	ST depression induced by exercise relative to rest. 0.0–6.2 (0.8)
a_{11}	SLOPE	The slope of the peak exercise ST segment: <i>Upsloping</i> 0.469 <i>Flat</i> 0.462 <i>Downsloping</i> 0.069
a_{12}	CA	Number of major vessels colored by fluoroscopy: 0 0.581 1 0.215 2 0.125 3 0.066 T 0.013
a_{13}	THAL	Exercise thallium scintigraphic defects: <i>Normal</i> 0.548 <i>Fixed defect</i> 0.059 <i>Reversible defect</i> 0.386 T 0.007
d	DISEASE	Angiographic disease status: Over 50% diameter narrowing in any major vessel? 0.459

Table 9.1: Summary of attributes recorded for the 303 patients recorded in the Cleveland Heart Disease database. For binary attributes, the prevalence is given. For numerical attributes, the range and median are given. For other attributes, the distributions are given. *DISEASE* is the decision attribute, indicating absence or presence of coronary artery disease.

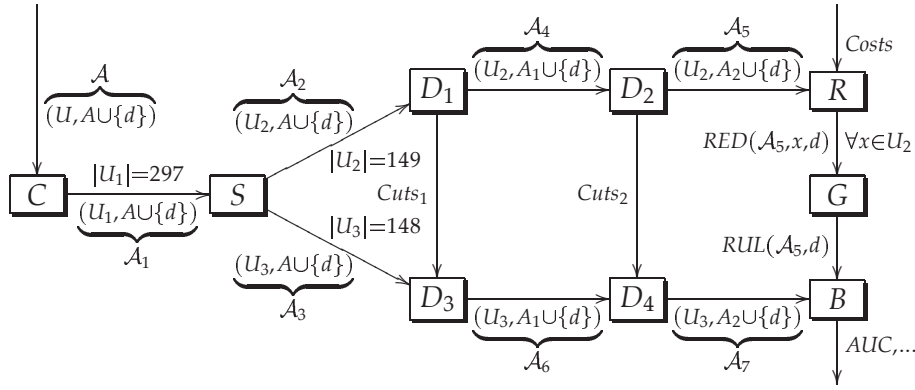


Figure 9.1: Experimental setup for the case study, outlining the flow of data. The content of the input decision system \mathcal{A} is summarized in Table 9.1, while the individual pipeline components are described in Section 9.4. \mathcal{A} is first cleansed of missing values to produce \mathcal{A}_1 , and then split into two disjoint subsystems, \mathcal{A}_2 and \mathcal{A}_3 . \mathcal{A}_2 is discretized in a two-stage procedure, and \mathcal{A}_3 is discretized using the cuts computed from \mathcal{A}_2 . Reducts and rules are then computed from the processed version of \mathcal{A}_2 , and the rules are used to classify the objects in the processed version of \mathcal{A}_3 . From this classification, performance estimates are harvested.

$$U_1 = U_2 \cup U_3 \quad U_2 \cap U_3 = \emptyset \quad (9.2)$$

D_1 \mathcal{A}_2 ►Discretize► Boolean reasoning algorithm...

Discretizes the numerical attributes in \mathcal{A} according to the discernibility-based multivariate procedure described in Appendix D. Produces a set of interval boundaries $Cuts_1$ as a side-effect.

$$Cuts_i = \{(a, c) \mid c \text{ is a cut for attribute } a \text{ computed by } D_i\} \quad (9.3)$$

D_2 \mathcal{A}_4 ►Discretize► Equal frequency binning...

A fallback routine for discretizing any numerical attributes that D_1 did not compute any cuts for. Employs an unsupervised univariate technique to do so, and produces a set of interval boundaries $Cuts_2$ as a side-effect.

D_3 \mathcal{A}_3 ►Discretize► From file with cuts...

Discretizes a table using the interval boundaries contained in $Cuts_1$.

D_4 \mathcal{A}_6 ►Discretize► From file with cuts...

Similar to D_3 , but employs $Cuts_2$ instead.

R \mathcal{A}_5 ►Reduce► Genetic algorithm...

```

begin ca
  nodes * Undefined
  make-reflexive
  Undefined -- *
end ca

begin thal
  nodes * Undefined
  make-reflexive
  Undefined -- *
end thal

```

Figure 9.2: A possible IDG file in ROSETTA format for dealing with missing values in the Cleveland database. Basically, the IDG states that all values are equal to themselves and that missing values match everything, effectively defining the *discerns/3* predicate for *CA* and *THAL* as in Equation 5.6. For attributes not listed, *discerns/3* reverts to Equation 5.4.

For each object $x \in U_2$, computes reducts relative to that object. Employs a genetic algorithm to do so, in which information about attribute costs can be used to steer the algorithm towards finding “cheap” solutions. Approximate solutions are found via minimal approximate hitting sets.

G Creates decision rules using the reducts computed by *R* as templates. However, for reasons of efficiency this step is done on-the-fly by *R*, and is transparent to the user.

B $\mathcal{A}_7 \blacktriangleright$ **Classify...**

Uses the rules output from *G* to classify the objects in \mathcal{A}_7 , and produces various performance measures.

9.4.1 Comments

A few comments on the experimental setup in Figure 9.1 are in order:

- We could have chosen several different ways of dealing with the missing values in \mathcal{A} in the *C* step other than removing incomplete objects. However, since there were so few objects with missing values, this option was selected. A different approach altogether would have been to use IDGs as described in Section 5.2.1. An example IDG file in ROSETTA format can be found in Figure 9.2.
- The attribute discretization is performed in two stages due to the presence of the discernibility-based D_1 procedure, since it effectively computes a reduct of the full decision system. This reflects itself in that D_1 might find that for some attributes, no cuts are needed as in fact the discernibility in the system is preserved even when those attributes are disregarded. A choice then has to be made as to how such redundant attributes should be dealt with. One option would be to discard them completely. Here, however, we choose to keep them and revert to discretizing them using algorithm D_2 .

- Discretization into intervals only makes sense for attributes whose domains can be totally ordered. “Grouping” together symbolic attribute values in sets as a symbolic counterpart to intervals on numerical domains is not currently implemented in ROSETTA. The question on how to handle the symbolic attributes in \mathcal{A} therefore has to be addressed.

The symbolic attributes $\{a_2, a_3, a_6, a_7, a_9, a_{11}, a_{13}\}$ will in this case study be kept as they are by employing “masking”, i.e., by making them invisible to the discretization routine before invoking it, and then reinstating the original masking states upon completion. In database terms, masking is equivalent to temporarily working on a projection of \mathcal{A} . For univariate discretization techniques, this works fine. For multivariate techniques like D_1 , however, this will have the effect that all discernibility considerations are done only on the basis of the visible numerical attributes $\{a_1, a_4, a_5, a_8, a_{10}, a_{12}\}$, and may thus result in more cuts than are strictly needed. To compensate for this, and to allow for noisy data, we may opt to preserve “most” and not all of the discernibility in the visible portion of the table by computing approximations of prime implicants of Equation D.6.

- In the fallback D_2 step, for the sake of simplicity, we will employ three intervals for all unmasked attributes. This intuitively corresponds to labeling the values as “low”, “medium” or “high” relative to the observations.

Note that if D_1 leaves no attributes undiscretized, then steps D_2 and D_4 can be skipped as they then collapse to simple pass-through identity operations.

- For reduction in the R step we make use of a list of attribute costs to guide the genetic algorithm towards finding low-cost solutions. Figure 9.3 lists a cost file in ROSETTA format. The cost of an attribute set is currently defined as the sum of the costs of the individual attributes. This definition does not take into account that when tests are performed in groups, there may be discounts due to shared common costs. Although not currently implemented, such a generalization is straightforward to add.
- In a more elaborate example, we could have included a filtering step in Figure 9.1 that would have filtered away “weak” or “expensive” rules. To simplify this case study, such a step is not included.
- A selection of performance measures will be harvested in the B step. In addition to performing an ROC analysis, we will also generate a calibration plot, and compute the Brier score and its covariance decomposition.

9.4.2 Parameters

Section 9.4 describes the steps in the pipeline, and from which pop-up menus the corresponding algorithms can be invoked. If an algorithm name has a trailing ellipsis, this means that the algorithm has a parameter dialog associated with it. In this case study, you can use the parameters listed below at each step.

```

age = 1.00
sex = 1.00
cp = 1.00
trestbps = 1.00
chol = 7.27
fbs = 5.20
restecg = 15.50
thalach = 102.90
exang = 87.30
oldpeak = 87.30
slope = 87.30
ca = 100.90
thal = 102.90

```

Figure 9.3: A file with cost information in ROSETTA format. The costs are in Canadian dollars, taken from the Ontario Health Insurance Program’s fee schedule. The listed costs are for individual tests, considered in isolation. When tests are performed in groups there may be discounts due to shared common costs.

Several algorithms take many more parameters than listed. If not explicitly specified, default values should suffice. Some of the parameters can be set via **Advanced parameters...** or other secondary parameter dialogs.

C No parameters required.

S Requires a parameter for determining $|U_2|$. $|U_3|$ is determined as $|U_1| - |U_2|$.

```

FACTOR = 0.5;
APPEND = True;

```

D₁ Requires a location for $Cuts_1$. Also requires parameters for controlling the approximation degree, and for selecting to exclude non-numerical attributes from discretization by “masking”.

```

FILENAME = cuts1.txt;          APPROXIMATE = True;
MASK = True;                  FRACTION = 0.97;

```

D₂ Requires a location for $Cuts_2$. Also requires parameters for controlling the number of intervals, and for “masking”. This step may or may not be needed to execute, depending on D_1 .

```

FILENAME = cuts2.txt;          INTERVALS = 3;
MASK = True;

```

D₃ Requires a location for $Cuts_1$, and a flag indicating the masking status at the time of creating $Cuts_1$.

```

FILENAME = cuts1.txt;
MASK = True;

```

D₄ Similar to *D₃*. This step may or may not be needed to execute, depending on *D₁*.

```
FILENAME = cuts2.txt;
MASK = True;
```

R Requires parameters for determining the discernibility type, and for defining the genetic algorithm. The latter includes specifying a location of the cost file, and parameters for controlling the approximation degree.

```
DISCERNIBILITY = Object;          COST = True;
MODULO.DECISION = True;          COST.FILENAME = costs.txt;
SELECTION = All;                 APPROXIMATE = True;
                                  FRACTION = 0.95;
                                  KEEP.LEVELS = 1;
```

G Transparent to the user, as it is executed on-the-fly by *R*. No parameters required.

B Requires parameters for determining how the rule-based classification should take place, and which performance estimates to harvest. In the parameter dialog for the **Standard voting** classifier, be sure to specify that the rules output from the *G* step should be employed. In addition to generating a confusion matrix, the parameters below generate a verbose log file, a calibration plot and an ROC curve as side-effects. The calibration file also includes information about the Brier score and its covariance decomposition.

```
CLASSIFIER = StandardVoter;      CALIBRATION = True;
FALLBACK = True;                 CALIBRATION.CLASS = Yes;
FALLBACK.CLASS = Yes;           CALIBRATION.FILENAME = cal.txt;
FALLBACK.CERTAINTY = 0.45;      CALIBRATION.GROUPS = 10;
MULTIPLE = Best;                ROC = True;
LOG = True;                     ROC.CLASS = Yes;
LOG.FILENAME = log.txt;         ROC.FILENAME = roc.txt;
LOG.VERBOSE = True;
```

9.4.3 Results

At this time, the reader should execute the steps in Figure 9.1 using the algorithms specified in Section 9.4 and the parameters from Section 9.4.2. A natural sequence to perform the operations in would be:

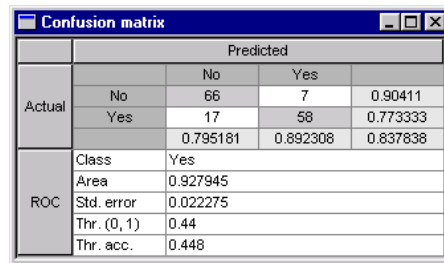
$$C, S, D_1, D_3, D_2, D_4, R, G, B$$

Note that, as discussed in Section 9.4.1, steps *D₂* and *D₄* may or may not be needed to execute, and that step *G* is performed automatically behind the scenes by step *R*.

After execution of all steps, the ROSETTA project tree might look similar to the one depicted in Figure 8.1.

Attribute	Intervals
a_1 AGE	$[-\infty, 52), [52, 60), [60, \infty)$
a_4 TRESTBPS	$[-\infty, 131), [131, \infty)$
a_5 CHOL	$[-\infty, 237), [237, \infty)$
a_8 THALACH	$[-\infty, 148), [148, \infty)$
a_{10} OLDPEAK	$[-\infty, 0.3), [0.3, \infty)$
a_{12} CA	$[-\infty, 1), [1, \infty)$

Table 9.2: Intervals computed from \mathcal{A}_2 in Figure 9.1 by means of D_1 and D_2 . Cuts for attributes $\{a_4, a_5, a_8, a_{10}, a_{12}\}$ were computed by D_1 , an approximate Boolean reasoning algorithm, and should be seen in conjunction to each other since this is inherently a multivariate technique. As it happens, D_1 found one cut for each of these attributes. Attribute a_1 was discretized by D_2 , a simple unsupervised univariate technique.



		Predicted		
		No	Yes	
Actual	No	66	7	0.90411
	Yes	17	58	0.773333
		0.795181	0.892308	0.837838
ROC	Class	Yes		
	Area	0.927945		
	Std. error	0.022275		
	Thr. (0,1)	0.44		
	Thr. acc.	0.448		

Figure 9.4: The confusion matrix output from step B in Figure 9.1, using the parameters suggested in Section 9.4.2.

With the suggested parameter settings, the set of cuts computed from \mathcal{A}_2 by D_1 and D_2 are listed in Table 9.2.

The set $RUL(\mathcal{A}_5, d)$ counts 2728 rules. For viewing purposes, it is a good idea to sort them according to their coverage. Table 9.3 lists a small handful of the highest-ranking rules after sorting.

Output from step B in Figure 9.1 are a confusion matrix and three ASCII files, `log.txt`, `roc.txt` and `cal.txt`. The confusion matrix is depicted in Figure 9.4, generated by always selecting the outcome class with the highest degree of certainty associated with it. For binary outcomes as discussed in Section 7.4, this corresponds to $\tau = 0.5$. Other thresholds could of course have been specified, resulting in different matrices. How sensitivity, specificity, classification accuracy, PPV and NPV vary as a function of τ are displayed in Figure 9.5.

Figure 9.4 also summarizes some results of the ROC analysis. The full ROC curve resides in the file `roc.txt`, along with some other information. Figure 9.5(d) displays the full ROC curve, showing how sensitivity and specificity are traded off against each other.

Rule $\alpha \rightarrow \beta$	Accuracy $\text{Pr}(\beta \mid \alpha)$	Coverage $\text{Pr}(\alpha \mid \beta)$
$(CP = \text{Asymptomatic}) \text{ AND } (THAL = \text{Reversible defect}) \rightarrow (DISEASE = \text{Yes})$	0.944	0.548
$(THALACH \geq 148) \text{ AND } (CA < 1) \text{ AND } (THAL = \text{Normal}) \rightarrow (DISEASE = \text{No})$	0.936	0.506
$(SEX = \text{Male}) \text{ AND } (FBS = \text{False}) \text{ AND } (CA \geq 1) \rightarrow (DISEASE = \text{Yes})$	0.886	0.500
$(CP = \text{Asymptomatic}) \text{ AND } (CA \geq 1) \rightarrow (DISEASE = \text{Yes})$	0.969	0.500
$(SLOPE = \text{Flat}) \text{ AND } (THAL = \text{Reversible defect}) \rightarrow (DISEASE = \text{Yes})$	0.879	0.468
$(CA \geq 1) \text{ AND } (THAL = \text{Reversible defect}) \rightarrow (DISEASE = \text{Yes})$	0.906	0.468
$(SEX = \text{Male}) \text{ AND } (CP = \text{Asymptomatic}) \text{ AND } (CA \geq 1) \rightarrow (DISEASE = \text{Yes})$	1.000	0.452
...

Table 9.3: A small handful of rules in $RLL(\mathcal{A}_5, d)$, sorted according to coverage. The rules were computed using a genetic algorithm for computing approximate prime implicants, as described by Vinterbo and Øhrn [233] and outlined in Section 5.2.6. All numerical measures are estimated from \mathcal{A}_5 . Intuitively, a “strong” rule is both accurate and has a high coverage. Rules with high coverage form succinct characterizations of the outcome classes, and are good for descriptive purposes. The accuracy of a rule reflects how “trustworthy” its consequent is.

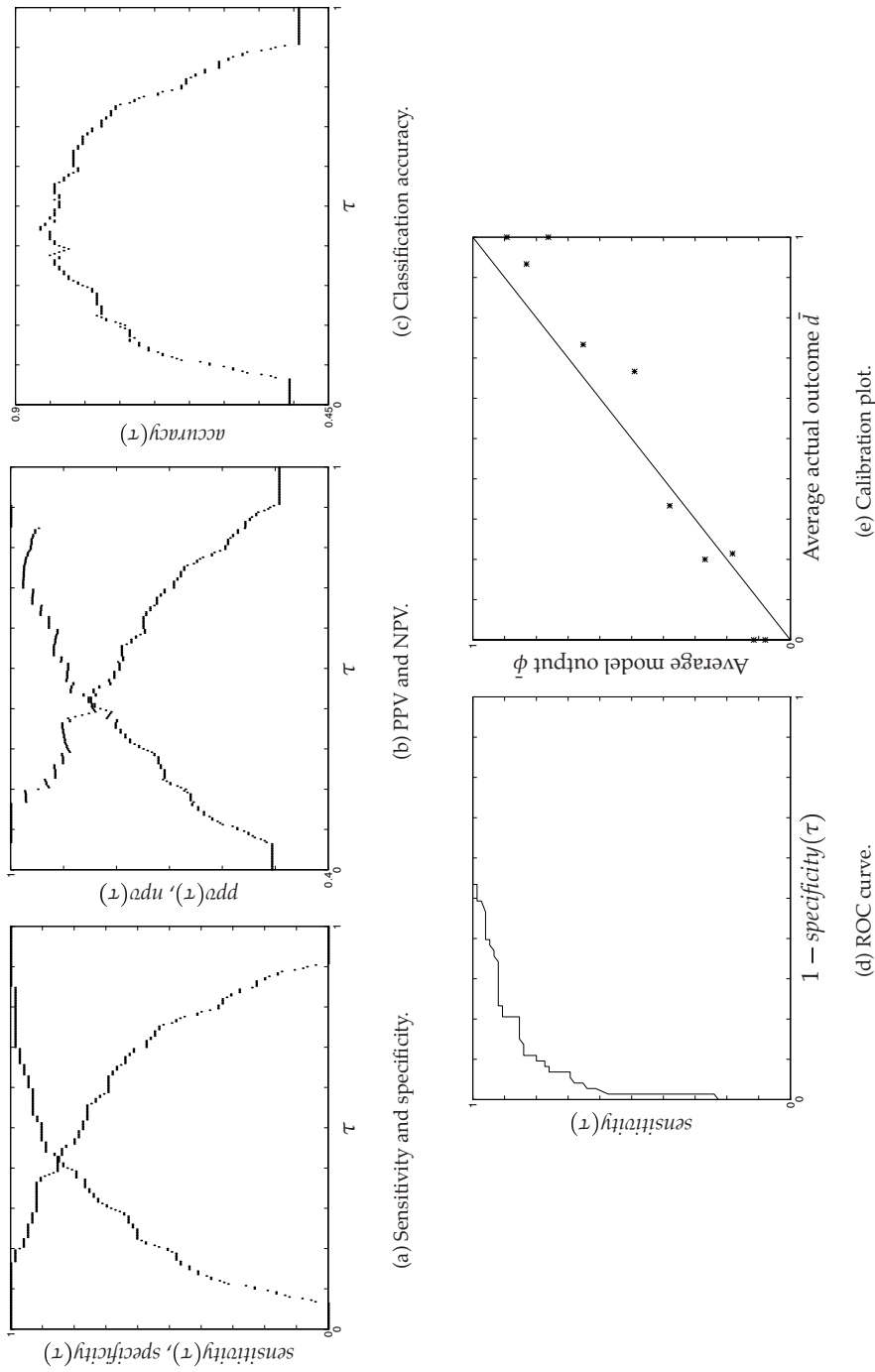


Figure 9.5: Performance plots for $RLL(\mathcal{A}_5, d)$ applied to \mathcal{A}_7 , generated from ROSETTA output. The ROC curve has an AUC of 0.928, with a standard error of 0.022. The calibration plot was generated using 10 groups.

Browsing the calibration plot file `cal.txt` shows that the Brier score of $RUL(\mathcal{A}_5, d)$ on \mathcal{A}_7 was 0.117. The covariance decomposition of this due to Arkes et al. [9] can also be found. Figure 9.5(e) displays the full calibration plot. The classifier seems fairly well calibrated.

Inspecting the verbose classification log file `log.txt` reveals that the fallback classification was never invoked, which is perhaps not too surprising considering the high number of rules in $RUL(\mathcal{A}_5, d)$.

9.5 Systematic Partitioning

The performance estimates obtained from executing the experimental setup in Section 9.4 only emanate from a single random two-way split. More reliable estimates are desired and can be obtained by systematically varying how the splitting step S is performed, as discussed in Section 7.2.1.

To illustrate how ROSETTA can be used for this purpose, the ROSETTA script language will be used to execute the pipelines in Figure 9.1 in a k -fold CV setting. To do this, we start from the completed decision system \mathcal{A}_1 , and execute the script in Figure 9.6. This can be done by invoking the following option:

`\mathcal{A}_1 ▶ Execute ▶ Pipeline script with CV...`

The script in Figure 9.6 consists of two parts: The first part describes the training sequence D_1, D_2, R and G , and the second part describes the test sequence D_3, D_4 and B . The rules generated in the first part are implicitly used in the second part. Also, some simplifications from Section 9.4 in the B step have been introduced. In particular, we only harvest ROC areas and classification accuracies. For k -fold CV, the pipelines are run through k times. The S step from Figure 9.1 is implicit, and is done according to the k -fold CV sampling scheme.

9.5.1 Parameters

The parameters for the individual components of the pipeline are given in Figure 9.6. However, the CV framework itself needs some parameters. Here, we will use $k = 10$. The following parameters are suggested, where `cvscript.txt` refers to Figure 9.6:

```
NUMBER = 10;                FILENAME.COMMANDS = cvscript.txt;
INVERT = False;            FILENAME.LOG = cvlog.txt;
LENGTH = 4;
```

```

BROrthogonalScaler
  {MODE = Save; MASK = True; FILENAME = c:/temp/cuts1#ITERATION#.txt;
   APPROXIMATE = True; FRACTION = 0.97}
EqualFrequencyScaler
  {MODE = Save; MASK = True; FILENAME = c:/temp/cuts2#ITERATION#.txt;
   INTERVALS = 3}
SAVGeneticReducer
  {DISCERNIBILITY = Object; MODULO.DECISION = T; SELECTION = All;
   COST = True; COST.FILENAME = c:/temp/costs.txt; APPROXIMATE = True;
   FRACTION = 0.95; KEEP.LEVELS = 1}
RuleGenerator
  {}

OrthogonalFileScaler
  {MODE = Load; MASK = True; FILENAME=c:/temp/cuts1#ITERATION#.txt}
OrthogonalFileScaler
  {MODE = Load; MASK = True; FILENAME=c:/temp/cuts2#ITERATION#.txt}
BatchClassifier
  {CLASSIFIER = StandardVoter; FALLBACK = True; FALLBACK.CLASS = Yes;
   FALLBACK.CERTAINTY = 0.45; MULTIPLE = Best; LOG = False;
   CALIBRATION = False; ROC = True; ROC.CLASS = Yes;
   ROC.FILENAME = c:/temp/roc#ITERATION#.txt}

```

Figure 9.6: A CV script in ROSETTA format, describing one pipeline for model induction and another for model assessment. Each parameter set should reside on a single line, but is here displayed on several lines. For parameters that are not listed, the default or currently set values are employed.

9.5.2 Results

After applying the algorithm from Section 9.5 to \mathcal{A}_1 using the parameters from Section 9.5.1, a CV log file `cvlog.txt` should have been produced. The CV log file contains a detailed breakdown of the data flows in Figure 9.1 for each iteration, with a performance summary at the end. Figure 9.7 displays an excerpt of the summary portion of `cvlog.txt`.

9.6 Comments

The goal of this example has not been to obtain optimal modelling results, but rather to demonstrate how ROSETTA can be used in a typical data mining and KDD scenario. Even better performances than the ones reported in Section 9.4.3 and Section 9.5.2 can almost surely be obtained through composing better pipelines and tweaking the algorithms' parameters. However, even the results obtained in this off-the-cuff case study compare very favorably with the results on the Cleveland database that are reported in the literature. Holte [82] has compiled a large survey of reported classification accuracies obtained from applying various machine learning methods to the Cleveland database. The accuracy results obtained in this case study surpass every method in-

```

Accuracy.Mean    = 0.821852
Accuracy.Median  = 0.825926
Accuracy.StdDev  = 0.082494
Accuracy.Minimum = 0.666667
Accuracy.Maximum = 0.9

ROC.AUC.Mean     = 0.914882
ROC.AUC.Median   = 0.915764
ROC.AUC.StdDev   = 0.053901
ROC.AUC.Minimum  = 0.825397
ROC.AUC.Maximum  = 0.982143

```

Figure 9.7: An excerpt of the summary portion of the CV log file resulting from having executed the script in Figure 9.6.

cluded in Holte’s survey.¹ A similar survey of *AUC* values does not exist, but the presently obtained median *AUC* value of 0.916 seems acceptable.

The inclusion of cost information in the *R* step was done to illustrate how ROSETTA can put such background knowledge to use. However, due to the limited size of the database, the genetic algorithm might find most, and perhaps all, prime implicants anyway, so the effect the costs have in the steering the search process might not be as large as it might have been if the number of condition attributes had been higher.

Although not demonstrated, another use of the costs in Figure 9.3 would be to filter and evaluate the rules according to the cost of checking their antecedents. This can be achieved in ROSETTA by invoking the following option:

RED►Filter►Cost filtering...

ROSETTA also offers the option to evaluate rules according to a large number of measures of rule quality. This can be done by invoking the following option:

RUL►Filter►Quality filtering loop...

Lastly, to test if the performance of two models differ significantly in a statistical sense as described in Section 7.4.6, this can be done with the HYPOCLASS program that accompanies ROSETTA. Currently, HYPOCLASS can perform McNemar’s and Hanley-McNeil’s tests, and operates directly on ROSETTA output.

¹As this survey was done in 1993, it probably does not fully reflect today’s situation. The simulation results are encouraging, nevertheless.

Chapter 10

Design and Implementation

Sections of this chapter have previously appeared as [140,145].

10.1 Introduction

The ROSETTA system was briefly introduced in Chapter 8 as a tool for supporting and executing the KDD process within a discernibility-based framework. This chapter delves beneath the surface of the ROSETTA system and presents the design and architecture of the software. Through examining some of the pragmatical issues arising from the KDD process, main design parameters are made clear and their proposed solutions outlined.

Section 10.2 outlines the system requirements and some of the rationale behind imposing these. The requirements for the computational kernel and the front-end are discussed in Section 10.2.1 and Section 10.2.2, respectively. A more thorough presentation and examination of the design and implementation of the computational kernel can be found in Section 10.3, while similar issues related to the front-end are discussed in Section 10.4.

This chapter is inherently software engineering oriented, and can be skipped without loss of continuity. For some sections, rudimentary knowledge of object-oriented software design and the C++ language is required. An outline of the ROSETTA source code library can be found in Appendix B.

10.2 System Requirements

In practice, a data analyst setting up a KDD pipeline is faced with a multitude of choices that have to be made, some of which are listed below. It is in the wake of these that some central system requirements of the ROSETTA system have surfaced:

- *Which of the steps in the pipeline are necessary for the application at hand?* The topology of the pipeline may differ with regards to both the purpose of the KDD task as well as the nature and state of the data material. Moreover, each of the phases in Figure 2.1 may themselves be composed of several substeps. A KDD toolkit should therefore offer a flexible means of constructing different pipeline topologies, preferably in a dynamic fashion so that the pipeline can have the ability to adapt to intermediate results during the course of processing, if such is requisite.
- *Which algorithms should be employed at the various stages?* There is almost always more than one way to accomplish a certain objective. Several algorithms may be candidates for the same task, each with different trade-offs with respect to computational complexity, accuracy and spatial requirements. In essence, interchangeability is paramount so that each functional building block in the pipeline can be easily substituted with other blocks that perform a similar task.
- *Which parameters should be used for a given algorithm?* Selecting the appropriate algorithms for each step in the KDD process is usually only part of the problem. Most often, the algorithms require a set of parameters that have to be manually specified and that may have considerable impact on the performance of the algorithm. Furthermore, in many situations it is desirable to repeat the same computational process with different sets of parameters. An easy way of passing parameters that is susceptible to automation is therefore needed.
- *Which hypotheses should be pursued in the mining process?* The KDD process is both interactive and iterative of nature, with a definite exploratory flavor. That is, even though the mining steps are typically as previously outlined at large, the person performing the data analysis makes several decisions along the way as intermediate results unfold, effectively deciding the specifics of the next few analysis steps. In many cases, at each level of analysis, several competing hypotheses may be candidates for further exploration, and several algorithms yielding different results may be equally well suited. To cope with such a high degree of fan-out, it is hence imperative to be able to operate in an environment which allows one to interactively manage and process data, while retaining data-navigational abilities.
- *How can I programmatically automate the process?* As part of the process of searching for a good pipeline composition, the data analyst will invariably want to chain candidate building blocks together to see how they interact and complement each other, and observe their overall output or performance. However, having to manually perform all the required steps would be extremely tedious work, especially if performance evaluation is done within a statistical resampling framework. An easy way to automate this process, at least partially, is therefore required.

Section 10.2.1 and Section 10.2.2 outline further requirements specific for the computational kernel and the front-end.

10.2.1 Kernel Requirements

The ROSETTA kernel is a general C++ class library offering all the most common rough set related structures and operations, and is intended for use by researchers as a toolkit for aiding rapid prototyping of new algorithms, as well as for automating sequences of steps in the overall KDD process.

A chief global requirement has been to keep a sharp distinction between the kernel of the system and the front-end. The kernel provides the relevant data structures and computations to be performed on these, while the front-end manages the manipulation and triggering of such. The kernel is fully independent of the front-end. This separation is important so that parts of the kernel can be embedded in other applications and employed on platforms to which the front-end cannot be ported.

Prior to development of ROSETTA, the RSES system [217] already contained some working C++ code for certain rough set calculations. Courtesy of Warsaw University, portions of this code was made available. An important design parameter for the ROSETTA kernel was therefore that it should encompass the RSES legacy code and employ it as an optional “plug-in” component. This is further explored in Section 10.3.3.

Several design parameters have been emphasized in the construction and design of the kernel library:

- *Maintainability*: The object-oriented features provided by the C++ language should be fully employed together with suitable advanced object-oriented design patterns. This promotes software component reuse and significantly contributes to the maintainability, flexibility and extensibility of the library.
- *Extensibility*: The library should cater for easy addition of new data structures and algorithms, as well as incorporation of legacy code. Notably, the system should engulf most of the core code of the RSES system, but without having to be dependent on its presence nor letting its state of design influence the overall design of ROSETTA.
- *Flexibility*: The design should allow for dynamically defined constructs and offer a versatile means of combining algorithms. Using existing library contents to try out new ideas should be easy and straightforward.
- *Modularity*: The kernel library should be composed of several individual sub-libraries with strict rules concerning their interdependencies. This minimizes recompilation and clarifies the overall library architecture.
- *Usability*: The operational behavior of the kernel should be consistent, and class interfaces should be uniform, complete and minimal. There should be a common way of doing things, a common idea permeating the library.
- *Efficiency*: The kernel should not be inefficiently implemented. It is a common misconception that the efficiency of numerical or scientific software written in

C++ must be sacrificed on the altar of object-orientation. This is not necessarily the case. The key to avoiding this is to be aware of what goes on behind the scenes, and to avoid writing code in critical inner loops that might incur a considerable overhead, e.g., implicit generation of temporary objects.

- *Portability*: The kernel should have no dependencies upon the front-end, and cutting-edge language constructs should, if used, be abstracted away by means of, e.g., macros in order to make the kernel as compiler-invariant as possible.

How these requirements have been met are discussed in Section 10.3.

10.2.2 Front-End Requirements

The observations on the nature of the KDD process as well as general requirements on modern GUIs have led to the formulation of the front-end requirements for ROSETTA. The resulting front-end is a GUI for interactive manipulation, construction and application of objects provided by the kernel. Windows NT was chosen as the primary target platform. The most important front-end design requirements have been:

- *User-friendliness*: The front-end should be intuitive, perspicuous and easy to use, and conform to the standard Windows GUI norms. Furthermore, the GUI should be object-oriented in the sense that the objects being manipulated should be represented as distinct and individual items in the interface, with the operations that are natural to perform on the objects available directly from the objects themselves.
- *Support the overall KDD process*: In order to function as an integrated system, the front-end should cater for all steps in the modelling process; from target data selection and preprocessing, via the actual mining procedures, to postprocessing, evaluation and presentation of the results.
- *Data-navigation*: During the whole modelling process, a vast multitude of different objects is likely to be generated. In order to retain navigational abilities and avoid drowning in data, it is therefore of vital importance how data is organized in the front-end. A front-end design goal has hence been to set the data in an environment where it is immediately clear which objects that exist and how they relate to each other.
- *Reflection of kernel contents*: The front-end should reflect the contents of the kernel. However, additional functionality will be added to the kernel incrementally, ultimately leading to unpleasant front-end maintainability problems to deal with if not properly handled from the beginning. A front-end design goal has therefore been that incremental additions made to the kernel should reflect themselves automatically in the front-end, with little or no additional front-end programming involved.

How these requirements have been met are discussed in Section 10.4.

10.3 The Computational Kernel

This section describes the design of the ROSETTA C++ rough set class library and how the kernel design goals have been achieved, and discusses the rationale behind some of the software engineering design decisions taken along the way. In addition, some examples of use are provided.

The ROSETTA kernel has been constructed with the design patterns and C++ language issues discussed by Meyers [122, 123] and Gamma et al. [61] as a guide. The techniques used comprise several of those described, both behavioral, creational and structural. Some software engineering issues concerning these are really only hinted at in this section.

10.3.1 Handles

Perhaps the most common source of C++ programming errors relates to the management of pointers. It is often the case that some objects are shared across an object ensemble, something that opens up a wealth of potential problems concerning ownership and administration. To alleviate this, the ROSETTA C++ class library uses “smart” pointers or handles.

A handle is a templated proxy around a traditional C-style pointer, with additional logic that ensures pointer validity and automatically takes care of memory management issues through reference counting. Handles have operators overloaded to ensure that they have the same semantics as traditional pointers, and automatically initialize to `NULL`. Moreover, a handle automatically deletes the object it refers to if it, when it leaves its scope or is reassigned, is the last handle referencing the object.

Although extremely simple, this garbage collection mechanism automatically and seamlessly minimizes or eliminates the problem of memory leaks, dangling pointers and multiple deletes, and significantly simplifies memory management and increases the safety of memory operations.

The described handle implementation only works as intended if the structures in memory form a directed acyclic graph. Pointers that introduce cycles must be ordinary “dumb” pointers given special attention. However, this shortcoming does currently not pose a problem, since a typical project defines a tree or a pipeline, a cycle-free structure.

10.3.2 Transitive Self-Identification

The use of run-time type identification (RTTI) in C++ is a controversial issue, since it to many degrees defeats the inherent polymorphic features of the language and opens up the door to code abuse and the writing of unmaintainable code. RTTI should therefore be used judiciously, and virtual functions employed instead if possible. Yet, there are some situations where RTTI is convenient. To this end, most commercial C++ code

libraries therefore supply some sorts of RTTI mechanism, and RTTI is even an integral part of the new ANSI/ISO C++ language standard.

Since, at the time of development, many existing compilers did not yet support this language feature, the ROSETTA library offers RTTI emulation. Since public inheritance models the is-a relation, the RTTI language feature is easily emulated via the use of virtual functions. Adding the property of transitive self-identification to a class in the ROSETTA library is effortlessly done through the calls to macros. In addition, various supplementary information such as, e.g., textual class descriptions may be stored along with the registering of each type identifier.

10.3.3 Legacy Code

Incorporation of external legacy code into the ROSETTA kernel has been done by means of a layering approach similar to the “Adapter” structural pattern [61]. By writing adaptors or wrappers that inherit from abstract classes, and embedding the appropriate objects from the external libraries in these, the external library is effectively hidden, yet made available to add value. The derived wrapper class then performs a conversion between the interface of the abstract object and the interface of the embedded object from the external library.

The wrapper classes are never referenced directly as this would render the system dependent on the presence of the wrapper classes and thus indirectly upon the design and incorporation of the external code. Instead the system always operates on the level of the abstract base classes, deferring the decision of which kind of leaf-node object in the class hierarchy that actually does the work to the overloading constructs of the C++ language. This makes the system independent of the presence of the external code, and enables substitution of the wrapper classes with other alternative implementations. Alternative implementations may also co-exist in parallel.

The wrapper approach works very well in practice. ROSETTA has been tested with various different configurations, and has been verified to function as a fully functioning system even when all external legacy code is omitted.

A short comment on efficiency might be in order here. Operating on the level of abstract base classes means that we are dealing with virtual functions. Since inlining of virtual functions does not make too much sense, this means that we have to incur the overhead associated with a function call. However, this slight penalty in performance is usually negligible, except perhaps for in extremely critical inner loops.

10.3.4 Creational Patterns

Since we as discussed in Section 10.3.3 wish to operate on the level of abstract objects and abstract objects cannot be instantiated, we need a creational pattern to overcome this in situations where objects need to be created. This has been done by applying a hybrid of the “Abstract Factory” and “Prototype” creational patterns [61]. Mak-

ing leaf-node objects available for use throughout the system is done by installing small, empty prototype objects to an `ObjectManager`, a static object managing a prototype pool. Furthermore, all classes have a virtual duplication method that returns a clone of themselves. By passing a type identifier to the object manager, the pool of installed prototypes is scanned by means of the transitive self-identification procedure described in Section 10.3.2. The installed prototype that matches the desired object type the most closely is then ultimately duplicated and returned, i.e., inexact matches are allowed. Although the `ObjectManager` can be operated on directly, a static intermediate `Creator` “factory” is also available.

Apart from installation of the prototype objects, the `new` and `delete` C++ keywords are thus never used directly to create or destroy objects in the ROSETTA library. Objects are instead dynamically allocated by the creational pattern outlined above, while the handle mechanism from Section 10.3.1 automatically takes care of the object destruction.

10.3.5 Main Object Dichotomy

It is worth noting that many algorithms are often implemented in terms of other algorithms. In essence, we have the notion of composite or compound algorithms. Consider for instance the conceptual procedure for computing dynamic reducts, outlined in Section 5.2.6. The basic formulation of the algorithm does not specify which sampling scheme to use, nor with which algorithm the subtable reduct calculation should be performed with. Coding various choices as sequences of conditional tests ultimately leads to unmaintainable code. Hence, the sub-algorithms themselves should be parameterized.

As a second motivating example, consider the case of the overall KDD process. As previously noted, the whole process is really a sequence of individually tunable steps that can be viewed as a pipeline of algorithms, with the output of one algorithm being the input to the next. In order to achieve partial automation of this process, it would be desirable to be able to chain algorithms together dynamically in a flexible fashion. Again, this leads to the design pattern of representing complex operations on objects as separate objects in their own right. This is highly reminiscent of the role functions have as so-called first-class elements¹ in functional programming languages such as Scheme [1].

The main dichotomy in the ROSETTA kernel is between structural and algorithmic objects. Structures are simple data containers and have only methods for simple administration and access, while complex computational procedures are represented as separate and interchangeable algorithm objects. An example of a structural object might be a decision table, while an example of an algorithmic object would be a reducer, i.e., an algorithm that computes a set of reducts. This separation, a variation of

¹Important “rights and privileges” of first-class elements are that they can be named by variables, passed as arguments to procedures, returned as the results of procedures, and included in data structures.

the “Visitor” and “Strategy” behavioral patterns [61], achieves among other:

- Enabling replacement or addition of new algorithms without having to modify existing structures, and vice versa. The structural classes rarely change, while defining new operations over the structures is a lot more common. When introducing a new algorithm to the system, the separation thus not only eliminates introducing bugs to the structural object the algorithm operates on, but also has a positive effect on library interdependencies and hence recompilation.
- Greater flexibility by enabling algorithms to be passed as parameters and dynamically chained together to form computational sequences.
- Simpler, smaller and more lucid class interfaces. The interfaces of the structural classes avoid getting polluted with an abundance of distinct and unrelated methods.
- Enables library clients to include into their projects only those parts of the library needed for their task at hand.

Note that any legacy code does not necessarily have to employ this architecture in order to be incorporated as previously described, although this would make an incorporation process more immediate.

10.3.6 Algorithm Application

In a rough set framework, the natural result of an algorithm if applied to a structure is often yet another structure. For instance, a set of reducts is created when applying a reduct computation algorithm to a decision table, and a set of rules is produced when applying a rule generation algorithm to a set of reducts. In other cases, the algorithm directly modifies the structure it is applied to. In the ROSETTA kernel setting, algorithms are applied to structures by passing them through a single virtual `Apply` method on the structure object. For composite structures, the `Apply` method may be overloaded to allow for individual processing of the substructures.

The return value semantics of the virtual `Apply` method are simple: If a new structural object is the natural endpoint of the algorithm, the new structure is returned. If the algorithm produces no new structural object but operates on and possibly modifies the input structure itself, the input structure is returned. Lastly, if an error occurs, nothing is returned. Optionally, if exception handling is desired used, an exception is raised.

The RTTI emulation from Section 10.3.2 can be used to define families of functionally similar algorithms, e.g., algorithms that carry out attribute discretization, or algorithms that compute reducts. Together with the outlined procedure, this implements the “Strategy” behavioral pattern [61]. This means that algorithms within the same family are fully interchangeable, letting them vary independently from clients that use them.

Each algorithm object keeps its own local set of parameters. Parameters are passed to algorithm objects through a standardized virtual `SetParameters` interface. The parameter string is parsed and keyword/value pairs automatically delegated to the right accessor methods. Library clients thus have to deal with a single method only, although on the cost of several keyword/value pairs. However, this approach is more susceptible to automation through script files, and smoothes out minor syntactical issues such as case sensitivity. The accessor methods can also be called directly.

10.3.7 Miscellaneous

In many cases it is desirable to know from which structure a structure has been derived, and in what way the structure was created. To this end, structures can be annotated in the way that they can automatically be “touched” with a description of which operation that was performed on them, which algorithm that was used and with which parameter set and so on. Done systematically, this offers a means of automatically generating documentation of the modelling session. The annotation is also reversible in the sense that an extract of the annotation can be passed as a parameter set to an algorithm, and hence be used as a means of automatically regenerating the computation.

10.3.8 A Small Example

To exemplify how the ROSETTA rough set C++ library can be programmatically applied, consider the following example of a complete albeit highly simplified KDD task: A decision table residing on an external storage medium is read and preprocessed by an attribute discretization algorithm. Reducts are subsequently computed, decision rules generated and exported to Prolog format. In the process, several intermediate structures are generated. The processing pipeline is displayed graphically in Figure 10.1. In a slightly extended example, the pipeline would be equipped with several intermediate postprocessing algorithms for weeding out “weak” reducts and rules, e.g., according to their support basis or cost.

The small C++ program in Figure 10.2 completely implements the pipeline in Figure 10.1, using a suitable set of algorithms. As can be seen, a very modest degree of programming is actually required.

10.4 The GUI Front-End

The ROSETTA front-end runs under Windows 95/98/NT and reflects the contents of the kernel C++ library. The GUI offers a user-friendly environment to interactively manage and process data, something that is invaluable when the model construction steps to take are not known beforehand but depend on decision-making as various intermediate structures unfold.

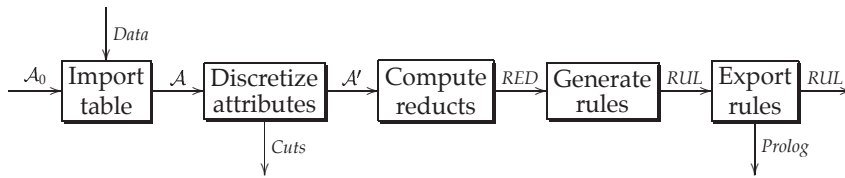


Figure 10.1: A small example pipeline. From tabular data residing on an external storage medium, decision rules in Prolog format are produced. Figure 10.2 displays how this pipeline can be completely implemented using the ROSETTA kernel C++ library.

```

Handle<Algorithm> algorithm[5];

// Set up the pipeline topology.
algorithm[0] = ObjectManager::GetIdentifiedAlgorithm(MYDECISIONTABLEIMPORTER);
algorithm[1] = ObjectManager::GetIdentifiedAlgorithm(RSESORTHOAGONALSCALER);
algorithm[2] = ObjectManager::GetIdentifiedAlgorithm(SAVGENETICREDUCER);
algorithm[3] = ObjectManager::GetIdentifiedAlgorithm(RULEGENERATOR);
algorithm[4] = ObjectManager::GetIdentifiedAlgorithm(PROLOGRULEEXPORTER);

// Set parameters.
algorithm[0]->SetParameters("FILENAME = table.txt");
algorithm[1]->SetParameters("MODE = Save; MASK = T; FILENAME = cuts.txt");
algorithm[2]->SetParameters("SEED = 1234; DISCERNIBILITY = Object");
algorithm[3]->SetParameters("");
algorithm[4]->SetParameters("FILENAME = rules.pl");

// Instantiate initial structure.
Handle<Structure> structure = Creator::DecisionTable();

// Add the structure to a project.
// ...

// Process the pipeline.
for (int i = 0; i < 5; i++) {
    structure = structure->Apply(*algorithm[i]);
    // ...
}

```

Figure 10.2: Complete implementation of the pipeline in Figure 10.1 using the ROSETTA kernel C++ library. Three lines of code have been omitted where indicated, as these are irrelevant for the current exposition.

10.4.1 Workspace

As previously argued, the generation of multiple intermediate structures requires that they get organized in some fashion in order for the user to retain data-navigational abilities. The ROSETTA GUI organizes its data in projects. A project is a collection of structures that all belong or are relevant to the same modelling task. More than one project may be open at the same time. The main window in the front-end of each project is therefore a tree view, where individual objects are represented as separate icons in the tree. How the various structures relate to each other is immediately apparent from the topology of the tree. As the modelling session unfolds, the tree is automatically updated to reflect the objects' interrelationships. The user may also manually edit the tree. A snapshot of a sample project tree is provided in Figure 8.1.

The project tree gives a bird's-eye view of the total state of the ongoing experiment. One can also zoom in and view the projects' individual member structures. Viewing of most structures is done in a matrix-like grid, in which the structures can also be manually edited or otherwise manipulated. By means of an optional dictionary, decision tables and all structural objects derived from such can be displayed to the user in terms from the modelling domain. In total, all the different views embody a comprehensive workspace. A snapshot of a sample workspace is provided in Figure 8.2.

10.4.2 Method Invocation

A central concept in the ROSETTA GUI is that there should be maximal proximity between the object being manipulated on and the triggering of the manipulation itself. The GUI therefore heavily makes use of context-sensitive pop-up menus, invocable by right-button mouse clicks. The operations that are natural to perform on an object are hence available directly from the object itself, with the list of potential actions being dependent on the object's type and state. For instance, a menu of actions that are natural to perform on a decision table is invocable from the decision table's icon, and a menu of actions that are natural to perform on a single column (attribute) in the table is invocable from the column header in the grid view of the table.

As a supplement and alternative to the context-sensitive pop-up menus, another GUI feature underscoring the proximity concept is the support for drag-and-drop. Using drag-and-drop, an algorithm may be applied to a structure by dragging the icon of the algorithm and dropping it onto the icon of the structure to apply it to, or vice versa. The dialog box for entering parameters associated with the algorithm will then automatically pop up. For instance, to compute the reducts in a decision table, the icon of an installed reducer algorithm may be dragged and dropped onto the icon of the decision table in the project tree.

10.4.3 Kernel Reflection

A goal in constructing the ROSETTA GUI was that incremental additions made to the kernel should reflect themselves automatically in the front-end with a minimum of programming effort. By scanning the prototype pool from Section 10.3.4 for installed objects, the front-end can automatically determine which structures that are available and which operations that are possible to perform on them. These can then automatically be made available to the user in the right places in the project tree and in the context-sensitive pop-up menus. Consequently, once a new structure or algorithm has been written, all that is needed to make them accessible in the front-end are the appropriate installation calls to the prototype pool.

Part IV

Applications

Chapter 11

Identifying Population Subgroups

The contents of this chapter have previously appeared as [104, 141, 151].

11.1 Introduction

This chapter explores how the semantics of the boundary region of a rough set can be exploited to identify subgroups of special interest in a larger population. The technique is an instance of feature extraction, where a new feature is constructed on the basis of existing ones in order to make subsequent modelling easier or more interpretable. Such new features might either serve as condition attributes or inputs to a modelling scheme, or, as in this case, as a decision attribute or output to model.

In particular, feature extraction for the following problem is investigated: With the increasingly rapid developments in medical technology, it is often the case that new medical tests are devised that are powerful with respect to diagnosis or prognosis. But even though a test is deemed as powerful, it may be the case that it is expensive to perform, invasive, cause patient discomfort or for other reasons be preferable to use sparingly. Furthermore, as argued by Harrell et al. [77], the yield of a new medical test should always be assessed in the context of other available information. It could very well be the case that combinations of more readily available parameters could enable one to arrive at the same diagnosis or prognosis in many cases, while the new test might only be really necessary to perform in very difficult cases. On the other hand, there might be cases that are so difficult to diagnose or prognosticate that the test would not be very helpful with respect to the decision problem at hand.

This chapter outlines a method that enables one to identify a population subgroup for which there exists no Boolean combinations of parameters free for a set of specified tests that would enable one to arrive at a diagnosis or prognosis with a high enough degree of certainty. In other words, this identified subgroup consists of the set of patients for which knowledge of the outcome of the specified tests are absolutely necessary for making a diagnosis or prognosis. Having identified and labeled this

subgroup, one can feed this extracted “identification feature” into a machine learning procedure for subsequent modelling of the identified subgroup. As a case study, a myocardial infarction problem is investigated.

Section 11.3 presents the identification methodology, while Section 11.3.1 defines suitable measures for evaluating rough set approximations. A description of an identification experiment can be found in Section 11.4, with the identification results in Section 11.5 and a discussion in Section 11.6.

11.2 Preliminaries

The mathematical foundations of this chapter are outlined in Chapter 5, in particular the sections on variable precision rough sets in Section 5.2.3. The same notation and terminology is employed in the following.

For the sake of simplifying the exposition, we will in the following assume that the decision problem at hand is binary, i.e., that $V_d = \{0, 1\}$. This is by no means an unreasonable assumption since in practice this situation is extremely frequent.

It is worth noting that this chapter does not make use of any computer-intensive constructs. Rough sets in themselves are simple to compute, the computationally demanding burden of calculating reducts does not come into play in the following.

11.3 Methodology

Let \mathcal{A} denote a decision system, and let $B \subseteq A$ denote a set of condition attributes that we are interested in avoiding to employ, if possible. The purpose of the identification process is to, in an oracle-like manner, identify patients $x \in U$ for whom it is possible to determine $d(x)$ without having knowledge of B . The output of this process is a new decision system that can be fed into a modelling procedure.

Rough set theory deals with the approximation of sets, e.g., the set of all patients that will either die or have a myocardial infarction within a certain follow-up period, or the set of all patients susceptible to a certain treatment. The set X , defined below, typically constitutes the set of patients that one would like to approximate and find minimal descriptions of during a modelling stage.

$$X = \{x \in U \mid d(x) = 1\} \quad (11.1)$$

The identification procedure consists of monitoring how the boundary region of X changes when attributes B are removed from A . Obviously, some patients may migrate into the boundary region while others will remain fixed. We are interested in identifying the former subgroup of the population.

The patients that migrate into the boundary region of X when attributes B are removed from A may do so from either the lower approximation or the outside region. This subgroup corresponds to the patients where knowledge of B is strictly required for identifying the approximation region to which they belong. Intuitively, these patients may be labeled as “possible but difficult” to diagnose or prognosticate, but where additional knowledge of B (or at least a subset of B) is crucially required to do so.

For all other patients, acquiring knowledge of B will have no effect with respect to which approximation region they end up in. This is due to one of two reasons:

- We already have enough information in $A - B$ to determine their membership status in X within a reasonable degree of certainty. This translates to the set of patients that remain fixed in the outside region or the lower approximation when we remove B from A , i.e., the patients for whom a sufficiently accurate assignment of membership or non-membership in X can be made without knowledge of B . Intuitively, these patients may be labeled as “easy” to diagnose or prognosticate.
- Their degree of membership or non-membership in X does not become sufficiently high or low when we consider A instead of only $A - B$. This translates to the set of patients that remain fixed in the boundary region when we remove B from A , i.e., the patients for whom a sufficiently accurate assignment of membership or non-membership in X cannot be made, even with knowledge of B . Intuitively, these patients may be labeled as “impossible” to diagnose or prognosticate.

The set of migrating patients can formally be defined as in Equation 11.3. Further subdividing the set of migrating patients into where they migrate from is of course possible. These ideas are displayed graphically in Figure 11.1.

$$\text{boundary}(A, \pi, X) = \overline{A}_\pi X - \underline{A}_\pi X \quad (11.2)$$

$$\begin{aligned} \text{migrate}(A, B, \pi, X) = \\ \text{boundary}(A - B, \pi, X) \cap (U - \text{boundary}(A, \pi, X)) \end{aligned} \quad (11.3)$$

Having defined the set of migrating patients, constructing a new decision system \mathcal{A}'_π from \mathcal{A} that can be used to model this patient group is straightforward, as shown in Equation 11.5. The identification process pinpoints the patients for which a Boolean combination of descriptors free for B exist for predicting the value of d , but does not in itself reveal any discerning characteristics of the identified patient group. For this, the decision system \mathcal{A}'_π can be passed on to a machine learning procedure for subsequent modelling of this patient group.

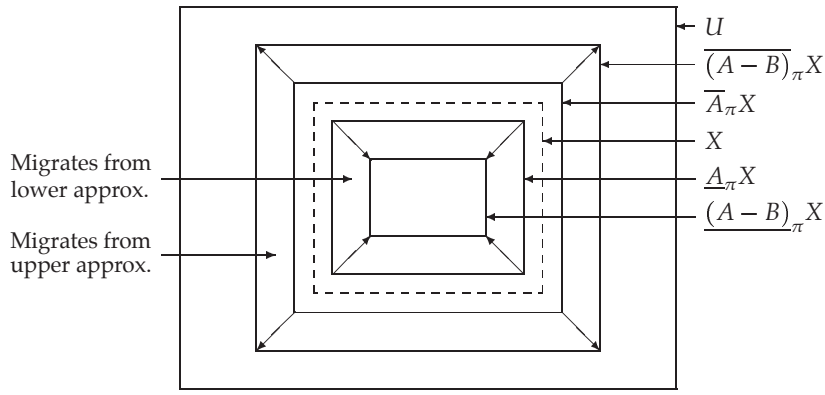


Figure 11.1: Monitoring the change in the boundary region of X when attributes B are removed from A . The set of patients that migrate into the boundary region are of particular interest.

$$d'_\pi(x) = \begin{cases} 1 & \text{if } x \in \text{migrate}(A, B, \pi, X) \\ 0 & \text{otherwise} \end{cases} \quad (11.4)$$

$$\mathcal{A}'_\pi = (U, (A - B) \cup \{d'_\pi\}) \quad (11.5)$$

11.3.1 Approximation Evaluation

The set approximations done in the identification step can be done with varying degrees of precision π . Of interest is then to construct approximations in the standard sense, as well as with more relaxed precision requirements. To evaluate the approximations, define *sensitivity* and *specificity* of the approximations as the following quantities:

$$\text{sensitivity}(A, \pi, X) = \frac{|\underline{A}_\pi X \cap X|}{|X|} \quad (11.6)$$

$$\text{specificity}(A, \pi, X) = \frac{|(U - \overline{A}_\pi X) \cap (U - X)|}{|U - X|} \quad (11.7)$$

Approximation sensitivity (respectively specificity) is thus to be interpreted as the number of objects that we correctly approximate as members (respectively non-members), divided by the actual number of members (respectively non-members).

Approximation accuracy is defined as the ratio between the total number of correctly approximated objects and the total number of objects, and is a weighting between the sensitivity and specificity.

$$accuracy(A, \pi, X) = \frac{|X|}{|U|}sensitivity(A, \pi, X) + \frac{|U - X|}{|U|}specificity(A, \pi, X) \quad (11.8)$$

11.4 Experiments

This section describes an identification experiment using a real-world dataset from a myocardial infarction prognostic problem. The data material has previously been analyzed by Geleijnse et al. [64].

Exercise testing provides important diagnostic and prognostic information in patients with known or suspected coronary artery disease. However, a large portion of patients with chest pain may not be able to exercise adequately, thus reducing the detection of coronary artery disease. For such patients, alternative stress modalities have to be used. In [64], a group of patients with chest pain underwent a dobutamine-atropine technetium-99m sestamibi single-photon emission computed tomography scintigraphic study. Using multivariate logistic regression, it was found that the single most important independent predictor for future hard cardiac events (cardiac death or non-fatal myocardial infarction) was an abnormal scan pattern. However, performing a scintigraphic scan is a relatively expensive procedure, and may for some patients not really be fully necessary as knowledge of the outcome of the scan may be redundant with respect to making a prognosis. If one through considering combinations of more readily available parameters could make the same decisions, one could thereby minimize the number of scans acquired and hence potentially cut both costs and use of resources.

The contents of the decision system \mathcal{A} are summarized in Table 11.1. There are 418 patients in the universe U , with the *DEATHMI* attribute as the decision attribute d and all other attributes defining the set of condition attributes A . All attributes are binary-valued, signifying the absence or presence of some feature. The cut-off values used for discretization of the inherently numerically valued attributes were decided upon externally by medical experts. There were no missing values in the data.

The 418 patients are all patients with chest pain, referred for the evaluation of suspected myocardial ischemia. The data in the decision system is largely the same as having been previously analyzed in [64], with some exceptions. 26 of the 418 patients with early elective coronary revascularization within 60 days after stress testing were excluded from the statistical analysis performed in [64]. None of these sustained a major cardiac event before coronary revascularization. The 26 patients could not be excluded from the present analysis, due to lack of knowledge of exactly which patients they were.

The attributes listed in Table 11.1 and used in the present analysis are a subset of those used in [64]. The endpoint or decision attribute is the same, reporting any subsequent hard cardiac events within a certain follow-up period. The *APSTRESS* and *STT*

Attribute	Description	Prevalence	
a_1	AGE	Over 70 years old?	0.191
a_2	OLDMI	Prior infarction?	0.490
a_3	HYPERT	Hypertension?	0.428
a_4	DM	Diabetes?	0.144
a_5	SMOK	Smoking?	0.280
a_6	CHOL	Hypercholesterolemia?	0.251
a_7	GENDER	Male?	0.569
a_8	HFMED	History of dec. cordis?	0.196
a_9	ANGP	History of angina?	0.249
a_{10}	APSTRESS	Angina during stress?	0.278
a_{11}	STT	ST-T changes?	0.311
a_{12}	SCANABN	Abnormal scan?	0.684
d	DEATHMI	Cardiac death or infarction?	0.112

Table 11.1: Summary of attributes recorded for the 418 patients used in the present identification experiment. All variables are binary-valued, with the prevalence as indicated. *DEATHMI* is the decision attribute, indicating whether a hard cardiac event occurred within the follow-up period. Condition attributes *APSTRESS*, *STT* and *SCANABN* are stress test related, while the other condition attributes are readily available clinical parameters.

attributes were acquired by means of the dobutamine-atropine stress test, while the *SCANABN* attribute originates from the scintigraphic scan described in Section 11.4 and found to be the single most important independent predictor¹ for future hard cardiac events [64]. All other attributes were selected primarily due to their simplicity and ease of acquisition.² For a more detailed description of the patient group and the precise semantics of each attribute, see [64].

Relating Table 11.1 to Equation 11.1, we are interested in forming approximations of the set of patients with *DEATHMI* entries of 1, i.e., the set of patients that either died or had a myocardial infarction within the follow-up period. Following the notation of Section 11.3, this was done when excluding the following attribute sets:

- $B = \{SCANABN\}$: *SCANABN* was singled out in [64] as the single most important independent predictor for *DEATHMI*, while at the same time being expensive to perform and thus desirable to employ sparingly.
- $B = \{APSTRESS, STT, SCANABN\}$: These attributes constitute the results from the stress test described in Section 11.4. Removing them leaves only more readily available clinical attributes for predicting *DEATHMI*.

¹Out of the 47 patients with a *DEATHMI* entry of 1, 44 also had a *SCANABN* entry of 1. Out of the 371 patients with a *DEATHMI* entry of 0, 242 had a *SCANABN* entry of 0.

²This was done while on a research stay in Trondheim by Piotr Szymański, MD, from the Postgraduate Medical School, Dept. of Cardiology, Warsaw, Poland.

π	Approx. region	B		
		\emptyset	$\{a_{12}\}$	$\{a_{10}, a_{11}, a_{12}\}$
0	Upper approx.	78	96	163
	Lower approx.	29	28	19
	Boundary region	49	68	144
	Outside region	340	322	255
	Sensitivity	0.617 (29/47)	0.596 (28/47)	0.404 (19/47)
	Specificity	0.916 (340/371)	0.868 (322/371)	0.687 (255/371)
	Accuracy	0.883 (369/418)	0.837 (350/418)	0.656 (274/418)
0.1	Upper approx.	78	79	92
	Lower approx.	29	28	19
	Boundary region	49	51	73
	Outside region	340	339	326
	Sensitivity	0.617 (29/47)	0.596 (28/47)	0.404 (19/47)
	Specificity	0.916 (340/371)	0.911 (338/371)	0.865 (321/371)
	Accuracy	0.883 (369/418)	0.876 (366/418)	0.813 (340/418)
0.2	Upper approx.	60	62	83
	Lower approx.	29	28	19
	Boundary region	31	34	64
	Outside region	358	356	335
	Sensitivity	0.617 (29/47)	0.596 (28/47)	0.404 (19/47)
	Specificity	0.957 (355/371)	0.949 (352/371)	0.887 (329/371)
	Accuracy	0.919 (384/418)	0.909 (380/418)	0.833 (348/418)
0.5	Upper approx.	52	47	32
	Lower approx.	52	47	32
	Boundary region	0	0	0
	Outside region	366	371	386
	Sensitivity	0.872 (41/47)	0.809 (38/47)	0.553 (26/47)
	Specificity	0.970 (360/371)	0.976 (362/371)	0.984 (365/371)
	Accuracy	0.959 (401/418)	0.957 (400/418)	0.935 (391/418)

Table 11.2: Approximating $X = \{x \in U \mid DEATHMI(x) = 1\}$ with different attribute sets $A - B$ and precision levels π .

11.5 Results

Table 11.2 lists the detailed results of identifying the set of migrating patients. The numbers indicate the cardinalities of the approximation regions in question. The drops in sensitivity and specificity can be attributed to exactly those patients that migrate into the boundary region when the indicated attributes are removed. Note that $\pi = 0.5$ amounts to always selecting the most probable category, and hence results in the empty boundary region.³

The main results from Table 11.2 can be summarized as follows:

- Using all attributes, the approximation sensitivity ranged between 61.7% and

³Technically, the variable precision rough set model is not defined for $\pi = 0.5$. Here, possible ties are resolved by assigning the objects in question to the interior of the set.

Approx. region	$\pi' \in \{0, 0.1, 0.2\}$	$\pi' = 0.5$
Upper approx.	35	25
Lower approx.	2	25
Boundary region	33	0
Outside region	383	393
Sensitivity	0.105 (2/19)	0.842 (16/19)
Specificity	0.960 (383/399)	0.977 (390/399)
Accuracy	0.921 (385/418)	0.971 (406/418)

Table 11.3: Approximating X'_0 from Equation 11.9 with $A - \{SCANABN\}$ and various secondary precision levels π' .

87.2%, depending on the chosen precision level π . Correspondingly, the approximation specificity ranged between 91.6% and 97.0%.

- Removing $\{SCANABN\}$, the size of the boundary region increased by 19 from 49 to 68 patients (for precision level $\pi = 0$), causing only a minor 2.1% drop in approximation sensitivity. The corresponding drop in approximation specificity was 4.8%.
- Removing $\{APSTRESS, STT, SCANABN\}$, the size of the boundary region increased by 95 from 49 to 144 (for precision level $\pi = 0$), causing a major 21.3% drop in approximation sensitivity. The corresponding drop in approximation specificity was 22.9%.

11.5.1 Meta Results

The small drop in approximation sensitivity and specificity in Table 11.2 when only *SCANABN* is removed seems to suggest that there is a substantial potential gain in considering combinations of less expensive and more easily available parameters in lieu of performing a scan.

These small drops are explained by the fact that relatively few patients migrate into the boundary set when only *SCANABN* is removed. But how accurately can we expect to be able to identify them without *SCANABN*? In order to answer this, we note that the set X'_π defined in Equation 11.9 is by definition rough without the *SCANABN* attribute and crisp with it. We can then proceed to perform a rough meta-analysis of X'_π for various secondary precision levels π' . Table 11.3 summarizes the results of a meta-analysis of X'_0 . Note that only 19 patients are members of X'_0 , as can be read from Table 11.2.

$$X'_\pi = \{x \in U \mid d'_\pi(x) = 1\} \quad (11.9)$$

The main results from Table 11.3 can be summarized as follows:

- The approximation sensitivity ranged between 10.5% and 84.2%, depending on the chosen secondary precision level π' . Correspondingly, the approximation specificity ranged between 96.0% and 97.7%.
- The upper approximation of X'_0 counted 35 patients (for secondary precision level $\pi' = 0$). The corresponding lower approximation counted 2 patients.

11.6 Analysis and Discussion

As can be read from Table 11.2, excluding only the *SCANABN* attribute results in only a minor drop in approximation sensitivity and specificity. Since so few patients migrate into the boundary region, this seems to suggest that there is a substantial potential gain in considering combinations of more easily available parameters in lieu of performing a scan. But even though the number of migrating patients is low, what matters in practice is our ability to identify them. If the upper approximation of the set of migrating patients had been extremely large, nothing much would have been gained. However, as can be read from Table 11.3, the upper approximation of the migrating set counts no more than 35 patients, even for the most conservative values of π and π' . This means that we cannot define X'_0 well internally, although we can circumscribe it fairly well. In practice, one would presumably send all patients in the upper approximation to acquire a scan. Assuming this to be true, the fact that the lower approximation counts only 2 patients and the approximation sensitivity is so low may not be important. And at any rate, 35 patients represents a relatively low percentage of the total number of patients.

Also evident from Table 11.2 is that the drop in both approximation sensitivity and specificity is substantial when the set $\{APSTRESS, STT, SCANABN\}$, i.e., all stress test information, is excluded. This seems to confirm the already known fact that stress testing yields valuable prognostic information.

Since the dataset is not exactly identical as the one used in the statistical analysis in [64], as explained in Section 11.4, results between the two are not directly comparable. But more importantly, the foci of the two analyses are different in nature. Whereas the statistical analysis sought to assess the prognostic power of the nuclear scan procedure, the rough set identification procedure sought to identify those patients for whom the procedure will presumably be of no help, and helps prepare for subsequent modelling of this subgroup.

It should be emphasized that the reported quantities in Section 11.5 are sensitivities and specificities of the set approximations as defined in Section 11.3.1, and are not to be interpreted as the performance of a classifier induced in the normal train / test fashion. If they were to be interpreted as such, the reported quantities would correspond to the performance of a classifier applied to the same data it was derived from, and thus gives an over-optimistic estimate of the performance an induced model may have on unseen data. So the reported quantities are thus relevant for the identification step and not a modelling step, but do provide a valuable estimate of the minimal loss of

performance a classifier induced without the excluded attributes might experience.

We note that unless the attribute (or set of attributes) we want to assess the prognostic power of (in our case, e.g., *SCANABN*) is a member of every reduct, then the migration set will necessarily be empty, since no degradation in the set approximation will take place. This is because there must then exist at least one reduct in which the attribute of interest is not a member, and these reducts, by definition of a reduct, preserve the indiscernibility relation.

Lastly, it should be noted that the issue of redundancy of a test addressed in this chapter is meant with respect to some particular decision making situation. A test may have multiple purposes and uses, and the presented identification methodology focuses on one such use, namely for classification into the decision classes defined through the decision attribute. For instance, a scintigraphic scan is useful for other things than predicting hard cardiac events, for instance to decide on future treatment.

Chapter 12

Anonymizing Sensitive Data

The contents of this chapter have previously appeared as [148].

12.1 Introduction

This chapter investigates how Boolean reasoning can be used to make the records in a database anonymous. In a medical setting, this is of particular interest due to privacy issues and to prevent the possible misuse of confidential information. As electronic medical records and medical data repositories get more common and widespread, the issue of making sensitive data anonymous becomes increasingly important.

Broadly speaking, the field of medical informatics can be said to concern itself with the capture of medically related data in a structured way and with ways to distribute and analyze these data to ultimately increase efficiency and the quality of healthcare. But as time progresses and more data becomes electronically available and shared, the challenge of protecting patient confidentiality becomes greater and more difficult. Woodward [243], Clayton et al. [33] and Sweeney [215] list several issues and challenges that this important problem complex raises, and provide several examples of how electronic medical records can be misused and have been misused in the past.

It is a common misconception that sensitive information is kept confidential simply because some directly identifying fields in the database, such as name or social security number, have been removed or obfuscated. Such databases may have been *de-identified*, but they are not likely to be *anonymous*. For example, a combination of attributes such as ethnicity, date of birth, gender and zip code may very well enable one to almost uniquely identify a patient by linking this information up to an external source of information, such as publicly available census data. This was demonstrated in practice by Sweeney [215]. Sweeney [213] defines the distinction between de-identification and anonymity as follows:

In de-identified data, all explicit identifiers, such as social security number,

name, address and phone number are removed, generalized or replaced with a made-up alternative; anonymous, however, implies that the data cannot be manipulated or linked to identify any individual.

Since any external information source may be used for linking, the only way to be certain that data is truly anonymous is therefore to make sure that the data is anonymous within the database itself. By that we mean that for each patient in the database there has to be at least one other patient from which the first patient cannot be discerned, at least with respect to the database fields that are most likely to be used for linking.

The fields of data mining and knowledge discovery are concerned with automating the detection and discovery of patterns in large volumes of data. Many such techniques are based on extracting identifying patterns that are then typically used for constructing association rules or for prediction or classification purposes. However, if identifying patterns can be found, they can also be employed in reverse to make the database they were extracted from anonymous. If we suppress some or all of the information described by the patterns, they become invalidated. This chapter investigates how an approach based on Boolean reasoning can be used to find and subsequently mask away combinations of field values that can be used to identify individual patients in a medical database.

Section 12.4 presents an anonymization methodology based on suppressing selected database entries, while Section 12.5 briefly reviews the features of two anonymization software systems. An example of the operation of the proposed cell suppression algorithm is presented in Section 12.6. Lastly, an analysis of different aspects of the algorithm and a discussion can be found in Section 12.7.

12.2 Preliminaries

The mathematical foundations of this chapter are outlined in Chapter 4 and Chapter 5, in particular the sections on discernibility and discernibility functions. The same notation and terminology is employed in the following. In particular, a database is in this context viewed as an information system \mathcal{A} , as defined in Section 5.2.

An overall assumption underlying cell suppression is that anonymization can be partially achieved through suppressing entries in the database, and that entries that are suppressed or “missing” cannot be used to discern between objects in the universe. Relating to Section 5.2.1, this means that the *discerns*/3 predicate is defined as in Equation 5.6, given below for reference. We can view a missing value as a variable that can be unified with everything.

$$\text{discerns}(a, x, y) \Leftrightarrow a(x) \neq a(y) \text{ and } a(x) \neq \top \text{ and } a(y) \neq \top \quad (12.1)$$

Depending on the semantics of the problem at hand, in some situations a missing value in a database might in reality signify a valid value, e.g., “not applicable”, rather

than “unknown”. It will in the following be assumed that a missing or suppressed value denotes the latter. This should not be a controversial assumption, as it can be guaranteed through some simple preprocessing, e.g., filling the appropriate empty cells with dummy values such as “Not applicable” or “-999999”.

The following exposition has deliberately been made with the goal of highlighting the “data mining in reverse” aspect of anonymization via cell suppression. Hence, the algorithms presented here should be taken as conceptual ideas rather than as absolute guides for practical implementation. This will be further discussed in Section 12.7.3.

12.3 Main Approaches

The presented approach to anonymization is based on mathematical logic and indiscernibility. Complementary techniques often have statistical roots [57, 87]. Denning [41] also outlines various approaches to database inference controls.

In *statistical disclosure control*, many techniques are used to anonymize data that are guaranteed to preserve some statistic such as the mean or variance of an attribute, however on the cost of “corrupting” the data. Examples of such techniques might be to add noise or perturbations in a controlled manner, or to simply randomly swap the attribute values around for a large number of the objects. Clearly, in many situations it is critical that the “truthfulness” of the data are preserved through the anonymization process. Obtaining anonymized data that preserves a higher-order statistic but that “lies” could, in a medical setting for instance, be potentially life-threatening if serving as a basis for medical care. For these reasons, the term *computational disclosure control* is sometimes employed instead, where truth-preserving anonymization is paramount.

Basically, there are three main approaches to making a database more anonymous while preserving the truthfulness of the data:

- *Outlier removal*. Certain rows or columns objects and/or attributes are removed altogether.
- *Generalization*. The value sets are made smaller.
- *Cell suppression*. Selected database entries are locally suppressed.

These approaches are by no means mutually exclusive, but complement each other. A full-fledged system for anonymization of sensitive data should probably apply a mixture of all three techniques to achieve optimal results.

An issue that should be considered when selecting methods for making a database anonymous is how the recipient of the processed data is going to make use of it. If the recipient is a researcher doing a data analysis project, complete but generalized data would probably be preferred to very specific data with missing values. This simply because most data analysis tools do not handle missing values in databases very well.

If, however, the processed data is only going to be used for browsing or for simple queries, difficulties associated with the handling of missing data might not matter.

12.3.1 Removing Outliers

A database may contain objects that in some sense significantly stray from the rest of the database contents. Such objects are referred to as *outliers*. If such an abnormal object can be identified, removing it from the database is a safe but perhaps overly radical way of maintaining anonymity. Conversely, if an outlier is labeled so because of a deviant value of a particular attribute, removing the attribute from the database would aid in maintaining the anonymity of the outlier. Attribute removal is equivalent to joining equivalence classes.

Removing or dropping outliers from a database is not entirely unproblematic, though. In the medical domain, the outliers often constitute the interesting cases. Moreover, if the dropped outliers belong to some subgroup of the population, removing them may skew the distribution of data as perceived by the recipient of the processed database.

12.3.2 Generalization

The value set of many attributes can often be ordered or organized in a hierarchy. For example, diagnose codes can be organized according to the ICD-9 classification system [59], and dates and numerical values can be sorted. The generalization process amounts to defining a coarser view of the world. This is done by reducing the set of possible values an attribute is allowed to take on, and may thus make two previously discernible objects indiscernible. If the attribute values can be semantically organized in a tree structure, this indiscernibility can be achieved by substituting the attribute value for an object with a more general attribute value somewhere above it in the tree. If the attribute values can be sorted or ordered along a line, generalization can be done by grouping the values into intervals. Otherwise, generalization can be done by forming subsets of the value set and assigning the same value to all elements in the subset.

In the machine learning literature, this is also referred to as *discretization* or *grouping*. In the statistical disclosure control community, generalization is also called *global recoding*.

12.3.3 Cell Suppression

Cell suppression consists of blanking out selected attribute values for outliers. Cell suppression thus preserves the number of rows and columns in the original database, and does not “blur” the perceived values of the non-suppressed entries. The algorithms we present in this chapter are examples of cell suppression.

```

procedure anonymize( $x$ ) {
   $P_A(x) \leftarrow \{p \mid p \text{ is a term of } BCF(f_A(x))\}$ 
  invalidate( $P_A(x), x$ )
}

procedure invalidate( $P, x$ ) {
  while ( $P \neq \emptyset$ ) {
     $a \leftarrow \text{select}(P)$ 
     $a(x) \leftarrow \top$ 
     $P \leftarrow P - \{p \in P \mid a \text{ occurs in } p\}$ 
  }
}

function select( $P$ ) {
  return  $\arg \max_a |\{p \in P \mid a \text{ occurs in } p\}|$ 
}

```

Figure 12.1: Pseudo-code for making an object $x \in U$ in a database \mathcal{A} anonymous. First, the set $P_A(x)$ of patterns that can be used to identify x are computed. Each pattern p is then subsequently invalidated by suppressing the value of an attribute a that occurs in p . This is done with a bias towards suppressing as few attribute values as possible.

12.4 Methodology

Based on the theoretical foundations in Chapter 4 and Chapter 5, this section presents the fundamental building blocks needed to assemble an algorithm for making a database anonymous via cell suppression.

12.4.1 Basic Algorithm

In order to make an object $x \in U$ anonymous, we can execute the *anonymize* procedure outlined in Figure 12.1. First, we compute the set $P_A(x)$ of prime implicants of the discernibility function $f_A(x)$. The set $P_A(x)$ thus summarizes all the ways in which x can be identified. Now, in order to make x anonymous we have to disable each pattern $p \in P_A(x)$ from being applicable. A pattern p can be rendered inapplicable if at least one of the attribute values it contains is suppressed. The *invalidate* procedure implements a simple greedy algorithm for suppressing attribute values for all $p \in P_A(x)$, with a bias towards suppressing as few attribute values for object x as possible. First, the attribute a that occurs the most often in the set $P_A(x)$ is determined by the *select* function. The value of attribute a is then suppressed, and the set of remaining valid patterns updated. This process is repeated until all the patterns are invalidated. Although not listed, a post-processing step might be desirable to add after the *invalidate* procedure to hamper the suppressed values from being reconstructed. This issue will be discussed in Section 12.4.4.

What the procedure *anonymize* actually does is to enlarge the indiscernibility set $R_A(x)$, or, equivalently, add edges to the indiscernibility graph G_A . This ensures us that, after execution of the procedure, object x is less identifiable than before, but still does not allow us to control and check if an object is “anonymous enough”. In fact, *anonymize* may not even be necessary to execute if this is the case to start with. A prerequisite for resolving this issue is to be able to numerically quantify the degree of anonymity of x in the context of the database \mathcal{A} .

Quantifying Anonymity

Letting $|f_A(x)|$ denote the number of factors in the unsimplified representation of the discernibility function $f_A(x)$, we can define the degree of anonymity for object x with respect to a database \mathcal{A} as shown below. Since no object is discernible from itself, $|f_A(x)|$ can be no larger than $|U| - 1$. Furthermore, $R_A(x)$ always includes x , so the total number of objects $|U|$ must equal the number of objects $|R_A(x)|$ that are indiscernible from x plus the number of objects $|f_A(x)|$ that are discernible from x .

$$\alpha_A(x) = \frac{|R_A(x)| - 1}{|U| - 1} = 1 - \frac{|f_A(x)|}{|U| - 1} \quad (12.2)$$

The quantity $\alpha_A(x)$ thus gives us a way to measure how anonymous x really is in the context of \mathcal{A} . If $\alpha_A(x) = 0$, then x is uniquely identifiable by attributes A and hence completely non-anonymous. Conversely, if $\alpha_A(x) = 1$, then x cannot be discerned from any other object and is hence completely anonymous.

Equation 12.2 is a very simple one, and an issue it does not take into account is the practical difficulty involved with identifying x . For instance, two objects that are uniquely identifiable would both have an anonymity measure of 0, regardless of how the complexity of the identifying patterns of the two objects differ. Intuitively, a uniquely identifiable object that can only be identified through a very complex pattern would be said to be more anonymous than a uniquely identifiable object that has several very simple identifying patterns. In order to cover this notion, a refinement of the formula for $\alpha_A(x)$ should also incorporate $|P_A(x)|$ as well as the lengths (and possibly nature) of the prime implicants $p \in P_A(x)$.

Having a way to quantify the degree of anonymity for a single object x enables us to express the overall level of anonymity $\alpha_A(U)$ of a full database \mathcal{A} in a variety of ways. For example, we can simply compute the average value of $\alpha_A(x)$ for all objects in the database.

$$\alpha_A(U) = \frac{1}{|U|} \sum_{x \in U} \alpha_A(x) \quad (12.3)$$

A chain is only as strong as its weakest link, so another possibility would be to define the overall anonymity level as the smallest $\alpha_A(x)$ for any object x .

```

procedure anonymize( $\mathcal{A}$ ,  $\tau$ ) {
  repeat {
     $done \leftarrow true$ 
    for each ( $x \in U$ ) {
      if ( $\alpha_A(x) < \tau$ ) {
        anonymize( $x$ )
         $done \leftarrow false$ 
      }
    }
  } until ( $done$ )
}

```

Figure 12.2: Pseudo-code for one possible way of making a database \mathcal{A} “anonymous enough”. The *anonymize* procedure in Figure 12.1 is called repeatedly for each object until all objects have a degree of anonymity no less than a specified threshold τ .

$$\alpha_A(U) = \min_{x \in U} \alpha_A(x) \quad (12.4)$$

The way anonymity is quantified here is rather traditional. A bin size, in the terminology of Sweeney [213], is mathematically the same as the cardinality of an equivalence class,¹ and hence, conceptually, overlaps with the definition of $\alpha_A(x)$. As currently formulated, however, the set $R_A(x)$ is a general cluster and not necessarily an equivalence class. Determining an optimal bin size is outside the scope of this chapter, but some heuristics and problems regarding this issue are described in [213].

12.4.2 Anonymizing a Database

Now that the fundamental building blocks have been outlined, they can be combined in a multitude of ways in order to assemble an algorithm where the degree of anonymity can be controlled. One possible example is shown in Figure 12.2. More elaborate implementations could also include individual anonymity levels τ_x . Additionally, if a more refined definition of $\alpha_A(x)$ was developed, a function that related this to τ_x could be passed to the *invalidate* procedure as an additional third argument, and be used to possibly abort the *while* loop before the set of identifying patterns was fully emptied.

It should be pointed out that making an object anonymous may alter the discernibility functions for other objects. Hence, different orderings in the *for* loop in Figure 12.2 may yield different anonymizations of \mathcal{A} . Alternative implementations could either fix a “good” ordering, or compute the identifying patterns for all objects and then invalidate them in two separate *for* loops that are executed sequentially. This would make the result order-independent, and is outlined in Figure 12.3. However, Figure 12.3 has the cost of possibly suppressing slightly more cells than strictly necessary,

¹If all equivalence classes have a cardinality higher than k , this is often referred to as *k-anonymity*.

```

procedure anonymize( $\mathcal{A}$ ,  $\tau$ ) {
  while ( $\alpha_A(U) < \tau$ ) {
    for each ( $x \in U$ )
       $P_A(x) \leftarrow \{p \mid p \text{ is a term of } BCF(f_A(x))\}$ 
    for each ( $x \in U$ )
      invalidate( $P_A(x)$ ,  $x$ )
  }
}

```

Figure 12.3: Pseudo-code for another possible way of making a database \mathcal{A} “anonymous enough”. In this case, contrary to Figure 12.2, the result is independent of the order the objects are processed in. Also, the anonymity threshold τ is relative to the whole database and not relative to each individual object. However, slightly more cells than necessary may be suppressed.

since there is no test for $\alpha_A(x)$ per object. Had we introduced such a test, ordering would once again matter.

12.4.3 Relative Anonymization

It might be the case that even though $\alpha_A(x)$ is substantially larger than zero, all members of $R_A(x)$ have the same value for some attribute d . Thus, we may still be able to infer the value of $d(x)$, even though we cannot ascertain exactly which record in \mathcal{A} that corresponds to object x . This leads us to consider the concept of d -relative anonymity, i.e. anonymity with respect to attribute d .

Relative anonymity has definite practical importance. For example, attribute d could denote the result of a HIV test, x could refer to a Hollywood celebrity, and we want to ensure that the celebrity’s HIV status is kept confidential even if a tabloid journalist gains access to the database. Obviously, our ability to block inferences about the value of $d(x)$ is determined by the spread of values d takes for all objects in $R_A(x)$. In the following, for sake of simplicity, d will be assumed to not be a member of A and to not have any missing values. Relating to Chapter 5, we now interpret a database as a decision system rather than an information system.

In order to ensure d -relative anonymity for object x , we must be able to answer the following: *Which patterns that match x enable us to infer the value of $d(x)$?* As it happens, this pertinent question can be answered within the present framework by a slight twist of the previously proposed anonymization procedures. Our focus hence turns from identifying patterns to patterns that constitute antecedents of propositional rules.

For rule generation, the key observation to make is that we do not need to discern x from objects that have the same value for attribute d as x has. Or, more precisely, we do not need to discern between objects that have the same value for a generalized version ∂_A of d , defined by Equation 5.36. The generalized attribute ∂_A when applied to x simply summarizes the set of values that the objects in the indiscernibility set $R_A(x)$ take on for attribute d .

```

procedure anonymize( $x, d$ ) {
   $P_A(x) \leftarrow \{p \mid p \text{ is a term of } BCF(f_A^d(x))\}$ 
  invalidate( $P_A(x), x$ )
}

```

Figure 12.4: Pseudo-code for making object $x \in U$ in a database \mathcal{A} anonymous relative to attribute d . The *invalidate* procedure is defined in Figure 12.1.

A variation of the basic *anonymize* algorithm with focus on d -relative anonymity is given in Figure 12.4. The only difference between Figure 12.1 and Figure 12.4 is that we compute the prime implicants from $f_A^d(x)$ instead of from $f_A(x)$, where ∂_A functions as a “filter”, as defined in Section 5.3.4. Each element of $P_A(x)$ determines a pattern p that defines a decision rule as defined in Section 6.3.1. The set of decision rules generated this way are both complete and minimal with respect to descriptors in \mathcal{A} [192]. The comment made in Section 12.4.1 about an additional anti-reconstruction step to be executed after the *invalidate* procedure, also applies here. This issue will be further discussed in Section 12.4.4.

After executing the *anonymize* procedure in Figure 12.4, $|\partial_A(x)| > 1$. However, we still lack a way of numerically quantifying how *well* an inference about the value of $d(x)$ is blocked. Having such a measure would enable the procedure in Figure 12.4 to be employed in a wider database setting, similar to Figure 12.2 or Figure 12.3.

It should be pointed out that using the generalized decision ∂_A as a “filter” as defined in Section 5.3.4 and employed in Figure 12.4 is typically suitable for one anonymization cycle only, if the algorithm in Figure 12.4 is to be used as a component of the procedures in Figure 12.2 or Figure 12.3. The reason for this is that after one cycle for each object, all objects may then have the same generalized decision value. If $|V_d| = 2$ this is definitely the case, since then the function $f_A^d(x)$ will have no factors. Hence, if d -relative anonymization is to be repeated until some anonymity criterion is met, then a slightly different version than Equation 5.37 should be used as a basis for Equation 5.42, for instance by using ∂_A^{π} from Equation 5.40 as a “filter” instead of ∂_A .

Quantifying Relative Anonymity

The relative degree of anonymity $\alpha_A(x, d)$ for an object x with respect to an attribute d gives a measure of how well the value of $d(x)$ is “disguised” in $R_A(x)$. As noted, our ability to block inferences about the value of $d(x)$ is determined by the spread of values d takes for all objects in $R_A(x)$.

$$\alpha_A(x, d) = \frac{|\partial_A(x)| - 1}{|V_d| - 1} \quad (12.5)$$

A naive way of defining $\alpha_A(x, d)$ is given by Equation 12.5. If $\alpha_A(x, d) = 0$, then all objects in $R_A(x)$ have the same value for attribute d . Conversely, if $\alpha_A(x, d) = 1$, then all possible values for d are represented in $R_A(x)$. However, the definition above says

nothing about the internal distribution of values within $R_A(x)$, and hence yields little information about the certainty with which one can make an inference about $d(x)$.

The difficulty of determining $d(x)$ increases with the degree of heterogeneity of $R_A(x)$ with respect to d . A natural way of incorporating this notion is by letting $\alpha_A(x, d)$ vary proportionally to the entropy² or disorder of $R_A(x)$ with respect to d , as defined by Equation 12.7.

$$\Pr(v \mid R_A(x)) = \frac{|\{y \in R_A(x) \mid d(y) = v\}|}{|R_A(x)|} \quad (12.6)$$

$$\alpha_A(x, d) \propto \sum_{v \in V_d} -\Pr(v \mid R_A(x)) \log_2 \Pr(v \mid R_A(x)) \quad (12.7)$$

An extended definition of $\alpha_A(x, d)$ should also take into account $|R_A(x)|$, similarly to how $\alpha_A(x)$ was defined in Section 12.4.1.

12.4.4 Hampering Reconstruction

The presented *anonymize* algorithm ensures that $|R_A(x)| > 1$, or, in the case of d -relative anonymization, that $|\partial_A(x)| > 1$. This alone, however, does not ensure that any query posed to the processed database will not return a single row, although any such returned single row will almost certainly contain suppressed values. In some special cases, suppressed values are reconstructible if outside knowledge about the value sets of the suppressed attributes are applied together with a priori knowledge about the database contents. To exemplify, consider querying the toy database in Table 12.1 for all black females. The query will return the singleton $\{x_3\}$, assuming that missing values match everything. Of course, a query for all female Eskimos would also yield the same query result, but if we know beforehand that the database definitely contains a black person, then the suppressed value for the *Ethnicity* field for object x_3 can effectively be reconstructed. In order to hamper a suppressed value from being deterministically reconstructible using knowledge external to the database, more information may need to be suppressed. If this is not an issue, an anti-reconstruction post-processing step may not be necessary.

Assume that $a(x)$ has been suppressed, and let $\hat{a}(x)$ denote $a(x)$ prior to suppression. Define the subset $N(a, x)$ as those objects in $R_A(x)$ for which attribute a has not been suppressed, and that have a different value for a than object x had originally.

$$N(a, x) = \{y \in R_A(x) \mid \hat{a}(x) \neq a(y) \neq \top\} \quad (12.8)$$

²On a historical note, the term *entropy* was introduced at the suggestion of John von Neumann to Claude Shannon. Allegedly, von Neumann said this use of the term was safe since few people could define entropy no matter how it was used.

	<i>Gender</i>	<i>Ethnicity</i>	...
U	a_1	a_2	...
x_1	F	Caucasian	...
x_2	F	Caucasian	...
x_3	F	T	...
x_4	M	Asian	...
x_5	M	Asian	...

Table 12.1: A small toy database exemplifying how suppressed values can be reconstructed if background knowledge of the database contents is available. A query for all black females would return the singleton $\{x_3\}$, assuming that missing values match everything.

If we suppress $a(y)$ in addition to $a(x)$ for some $y \in N(a, x)$, this would rectify our particular problem. In our example in Table 12.1, this corresponds to suppressing the *Ethnicity* field for either object x_1 or object x_2 as well. Doing this in a nested loop for all suppressed variables for all objects is one way of hampering reconstruction that can be invoked as a post-processing step to perform after the *invalidate* call in either of the outlined *anonymize* algorithms. Since such additional suppression for one object might alter the indiscernibility set for another object, the same comment made in Section 12.4.2 about order-dependence also applies here. Also, there is the question of selecting which object in $N(a, x)$ to additionally suppress. In case of d -relative anonymity, we would typically select an object y such that $d(y) \neq d(x)$.

Note that additional suppression using the $N(a, x)$ construction may or may not be necessary or helpful in hampering $a(x)$ from being reconstructed, depending on what external knowledge we expect a reconstructor to have. Suppose for instance that our query for black females returned two objects, both with the *Ethnicity* field suppressed. If we knew beforehand that the database must contain one black female, we still cannot be sure which of the two returned records that belong to that individual. But if we knew beforehand that the database must contain two black females, both suppressed *Ethnicity* fields are reconstructible. There is thus a problem- and recipient-specific limit on the effort one should put into hampering reconstruction. Consider for instance the extreme upper bound of an omniscient reconstructor. Then, he or she will be able to reconstruct the entire table, even if we suppress every single field in the database.

As noted and demonstrated, suppressed cells can often be reconstructed through cleverly applying background knowledge of some sort.³ Denning [41] describes several methods of attack on disclosure techniques, all of which involve using released statistics and supplementary knowledge to solve a system of equations for some unknown.

³In fact, to be absolutely sure that confidentiality is not breached, the data should perhaps not have been released in the first place. This illustrates that confidentiality is just as much a policy issue as it is a technological issue.

12.5 Software Systems

Software systems for making confidential databases more anonymous exist. A review of and comparison between the Datafly [214] and μ -ARGUS [87] systems is done by Sweeney [213]. Neither Datafly nor μ -ARGUS address relative anonymization. The Datafly system performs generalization and may remove outliers entirely, but does not incorporate cell suppression. The μ -ARGUS system also performs generalization, but performs cell suppression instead of removing the outliers.

This chapter has focused on presenting a formal mathematical approach for detecting identifying combinations of fields in a database. In the μ -ARGUS system, this problem is addressed in a very limited manner by only considering combinations of two or three fields among a certain subset specified by the user. Larger combinations or combinations that involve attributes outside the specified subset go undetected. The Datafly system combats the problem in a better way by requiring that the cardinality of the equivalence classes of objects with respect to a specified subset of attributes be no less than a certain threshold. If this requirement cannot be met, generalization of the value set of one of the attributes considered is attempted. Outliers may be dropped in the process, and identifying combinations of attributes outside the specified subset may still go undetected.

12.6 Examples

A prototype for performing the outlined anonymization procedures has been implemented within the framework of the ROSETTA library [145,147]. This section provides examples of how the outlined Boolean reasoning algorithms operate.

12.6.1 Anonymization

Consider the small example database \mathcal{A} in Table 12.2(a). All objects in the database are uniquely identifiable, and most of the records have identifying characteristics even when the SSN attribute is held aside. After one iteration of the anonymization algorithm outlined in Figure 12.3, the database has been transformed into the “more anonymous” database shown in Table 12.2(b). In this database, each object has at least one other object from which it cannot be discerned.

As a case study, object x_5 will be considered in detail. First, the discernibility function $f_{\mathcal{A}}(x_5)$ is constructed as shown below. Each conjunction in the POS formula stems from one of the other objects, and expresses how that particular object can be discerned from object x_5 .

U	SSN a_1	BirthYear a_2	Gender a_3	Ethnicity a_4	Zip a_5
x_1	0123456789	1964	M	Caucasian	02116
x_2	1234567890	1964	F	Caucasian	02138
x_3	2345678901	1970	M	Black	02144
x_4	3456789012	1968	F	Asian	02166
x_5	4567890123	1969	F	Black	02156
x_6	5678901234	1970	M	Black	02144
x_7	6789012345	1964	F	Caucasian	02138
x_8	7890123456	1969	F	Asian	02116
x_9	8901234567	1968	F	Asian	02166
x_{10}	9012345678	1964	M	Caucasian	02166

(a) Before

U	SSN a_1	BirthYear a_2	Gender a_3	Ethnicity a_4	Zip a_5
x_1	T	1964	M	Caucasian	T
x_2	T	1964	F	Caucasian	02138
x_3	T	1970	M	Black	02144
x_4	T	1968	F	Asian	02166
x_5	T	1969	F	T	T
x_6	T	1970	M	Black	02144
x_7	T	1964	F	Caucasian	02138
x_8	T	T	F	Asian	T
x_9	T	1968	F	Asian	02166
x_{10}	T	1964	M	Caucasian	T

(b) After

Table 12.2: Example database \mathcal{A} before and after one anonymization cycle.

$$\begin{aligned}
f_A(x_5) = & (a_1^* + a_2^* + a_3^* + a_4^* + a_5^*) \\
& \cdot (a_1^* + a_2^* + a_4^* + a_5^*) \\
& \cdot (a_1^* + a_2^* + a_3^* + a_5^*) \\
& \cdot (a_1^* + a_2^* + a_4^* + a_5^*) \\
& \cdot (a_1^* + a_2^* + a_3^* + a_5^*) \\
& \cdot (a_1^* + a_2^* + a_4^* + a_5^*) \\
& \cdot (a_1^* + a_4^* + a_5^*) \\
& \cdot (a_1^* + a_2^* + a_4^* + a_5^*) \\
& \cdot (a_1^* + a_2^* + a_3^* + a_4^* + a_5^*)
\end{aligned}$$

We then proceed to compute $BCF(f_A(x_5))$, which semantically represents the same function as $f_A(x_5)$ but rather expressed as a minimal SOP formula. Each term in $BCF(f_A(x_5))$ represents a minimal identifying pattern for object x_5 , and the set $P_A(x_5)$ comprises all of these.

$$BCF(f_A(x_5)) = (a_1^*) + (a_2^* \cdot a_4^*) + (a_3^* \cdot a_4^*) + (a_5^*)$$

The *invalidate* procedure then suppresses the cell entries that would otherwise identify object x_5 . This is done by initially noting that the *Ethnicity* attribute occurs the most frequently in $P_A(x_5)$. This field is therefore suppressed for object x_5 , and those elements in $P_A(x_5)$ that contain this attribute are removed from $P_A(x_5)$. This leaves only two disjoint elements in $P_A(x_5)$ (namely $\{SSN\}$ and $\{Zip\}$), so these are in turn also suppressed for object x_5 .

From Table 12.2(a) to Table 12.2(b), the indiscernibility set $R_A(x_5)$ has grown from the singleton $\{x_5\}$ to the set $\{x_5, x_8\}$, hence increasing our crude measure of anonymity $\alpha_A(x_5)$ from 0 to 1/9.

12.6.2 Relative Anonymization

If we augment the small example database in Table 12.2(a) with an extra attribute *HIV*, we obtain the database shown in Table 12.3(a). We want to anonymize the database relative to the *HIV* variable, blocking deterministic inferences about the *HIV* status of each patient. After one iteration of the anonymization algorithm outlined in Figure 12.3, but using the discernibility function computed modulo *HIV* as shown in Figure 12.4, the database in Table 12.3(a) has been transformed into the database shown in Table 12.3(b). In the processed database, each object will have at least one other object indiscernible from itself that has a different value for the *HIV* variable.

Again, object x_5 will be considered in detail. To construct the *HIV*-relative discernibility function $f_A^d(x_5)$, we can simply drop those conjunctions from $f_A(x_5)$ that discern object x_5 from objects with negative *HIV* values.

U	SSN a_1	BirthYear a_2	Gender a_3	Ethnicity a_4	Zip a_5	HIV d
x_1	0123456789	1964	M	Caucasian	02116	Positive
x_2	1234567890	1964	F	Caucasian	02138	Positive
x_3	2345678901	1970	M	Black	02144	Negative
x_4	3456789012	1968	F	Asian	02166	Negative
x_5	4567890123	1969	F	Black	02156	Negative
x_6	5678901234	1970	M	Black	02144	Negative
x_7	6789012345	1964	F	Caucasian	02138	Negative
x_8	7890123456	1969	F	Asian	02116	Positive
x_9	8901234567	1968	F	Asian	02166	Negative
x_{10}	9012345678	1964	M	Caucasian	02166	Negative

(a) Before

U	SSN a_1	BirthYear a_2	Gender a_3	Ethnicity a_4	Zip a_5	HIV d
x_1	⊥	1964	M	Caucasian	⊥	Positive
x_2	⊥	1964	F	Caucasian	02138	Positive
x_3	⊥	⊥	M	⊥	⊥	Negative
x_4	⊥	⊥	F	Asian	⊥	Negative
x_5	⊥	1969	F	⊥	⊥	Negative
x_6	⊥	⊥	M	⊥	⊥	Negative
x_7	⊥	1964	F	Caucasian	02138	Negative
x_8	⊥	⊥	F	Asian	⊥	Positive
x_9	⊥	⊥	F	Asian	⊥	Negative
x_{10}	⊥	1964	M	Caucasian	⊥	Negative

(b) After

Table 12.3: Example database \mathcal{A} before and after one anonymization cycle relative to HIV.

$$\begin{aligned}
f_A^d(x_5) &= (a_1^* + a_2^* + a_3^* + a_4^* + a_5^*) \\
&\cdot (a_1^* + a_2^* + a_4^* + a_5^*) \\
&\cdot (a_1^* + a_4^* + a_5^*)
\end{aligned}$$

The prime implicants of $f_A^d(x_5)$ are then computed and subsequently blocked by the *invalidate* procedure. Each term in $BCF(f_A^d(x_5))$ represents a minimal antecedent for a rule that could otherwise be used to deterministically predict the value of variable *HIV* for object x_5 .

$$BCF(f_A^d(x_5)) = (a_1^*) + (a_4^*) + (a_5^*)$$

From Table 12.3(a) to Table 12.3(b), the indiscernibility set $R_A(x_5)$ has grown from the singleton $\{x_5\}$ to the set $\{x_4, x_5, x_8\}$. As a result, the set of possible *HIV* values for the objects in $R_A(x_5)$ has grown from $\{\text{Negative}\}$ to $\{\text{Negative}, \text{Positive}\}$.

12.6.3 Remarks

Neither Table 12.2(b) or Table 12.3(b) have been exposed to any kind of anti-reconstruction scheme, as outlined in Section 12.4.4. As it happens, Table 12.3(b) is a good example of a situation where hampering reconstruction of attribute a_4 is a good idea if we expect a reconstructor to know beforehand that the database contains a black female. In this case, we would also suppress the value of attribute a_4 for object x_8 , since $x_8 \in N(a_4, x_5)$ and $d(x_8) \neq d(x_5)$.

Alternatives to performing such post-processing include more anonymization cycles, another definition of the *discerns/3* predicate in Section 5.2.1, or a less conservative implementation of the *invalidate* procedure in Figure 12.1.

12.7 Analysis and Discussion

This chapter has considered the problem of making databases with sensitive contents anonymous, and framed the problem in the formal mathematical setting of Boolean reasoning. Furthermore, an algorithm has been presented that can be used to mask away combinations of values that serve as identifying patterns for individual records in a database, or for a particular field in the database.

The presented algorithm has several desirable properties:

- It has a firm, mathematical foundation.
- The degree of anonymity can be tailored according to the specific needs of the recipient, and according to the amount of trust we place in the recipient. Furthermore, the required measure of anonymity can be specified as far down as to the individual objects in the database.

- Outliers are not removed, but are only partially masked in order to make them acceptably anonymous.
- The framework can be used to both preserve object anonymity as well as to block deterministic inferences about specified database fields.

However, the presented algorithm in its most basic form would probably still not be sufficient for most real-world anonymization applications as a stand-alone system. The most prominent reason for this is that the algorithm does not incorporate any kind of generalization feature. Initial generalization of the databases could rather quickly ensure an acceptably large bin size on a per attribute basis, and might even circumvent the problems described in Section 12.4.4 by making any equivalence class for any object with respect to the full attribute set A a non-singleton. Additionally, an anonymization approach based on discernibility is best suited for categorical variables. When dealing with variables over numerical domains, generalization or discretization thus becomes an important and necessary preprocessing step.

A suitable place for the presented algorithm would therefore be as a subsystem of a larger and more elaborate anonymization system such as Datafly, where it could be invoked after generalization has taken place. Using the algorithm we presented would allow us to verify that no further generalization is needed, or to locally suppress entries for outliers instead of removing them altogether from the database. Also, the procedure may be used to detect identifying combinations of fields not included in the subset of attributes that have to be specified by the user in both the Datafly and μ -ARGUS systems, and to perform relative anonymization.

12.7.1 Interpretation and Links

Many of the mathematical ideas employed in this chapter can be justified and explained in terms of rough set theory, as outlined in Chapter 5. Skowron [192] discusses tolerance information systems and applications of Boolean reasoning in conjunction with tolerance relations and indiscernibility phenomena. It is worth noting the close relationship between how the identifying patterns are detected and how decision rules are generated in the rough set approach of rule induction. In fact, an identifying pattern really defines a classification rule if we envision our database to be augmented with an extra one-to-one attribute d .

In the basic formulation of rough set theory, the indiscernibility relation R_A defined in Section 5.2.2 would have to be an equivalence relation, and the anonymity of an object x would consist of it being "hidden" in the upper approximation of the singleton $\{x\}$. The upper approximation, in turn, is a special case of the more general topological notion of closure.

There is an intimate link between prime implicants of discernibility functions and the notions of keys and functional dependencies in relational databases. If we consider the full database's discernibility function as defined in Equation 5.21, a prime implicant p

of this function actually defines a minimal functional dependency $P \rightarrow A - P$ in table \mathcal{A} , where P is the set of attributes that occur in p .

The cell values that are suppressed for object x in the *invalidate* procedure in Figure 12.1 actually form an implicant of the Boolean POS function $BCF(f_A(x))'$, where the implicant is composed of the complemented Boolean variables returned by the *select* function. We typically want the implicant to be a prime implicant, to avoid unnecessary suppressions from taking place. Hence, the suggested definition of the *select* function seems reasonable. In fact, the *invalidate* procedure together with the currently defined *select* function implements a variation of the set covering heuristic defined by Johnson [95].

12.7.2 Changing Biases

As noted, the *select* function in Figure 12.1 makes the *invalidate* procedure have a bias towards suppressing as few cells as possible, i.e., having a bias towards preferring short terms of $BCF(BCF(f_A(x))')$. However, it can easily be redefined to provide other behaviors. For example, one may instead choose to return the attribute that minimizes a user-defined cost. That way, user preferences can be incorporated.

Processing data to ensure anonymity necessarily causes a certain degree of loss of information. The challenge of any anonymization process is to preserve as much information as possible, while ensuring a preset level of anonymity. One way of measuring the data-quality or information content is to compute the entropy or disorder in the database, according to standard information-theoretic formulae. Therefore, instead of having the user define the attribute costs, an alternative would be to overload the *select* function to additionally take into account the information loss that results by suppressing $a(x)$.

12.7.3 Complexity Issues

The biggest drawback of the algorithm, as presented, is its potential complexity. In general, computing all prime implicants is an NP-hard problem [192]. However, this theoretical drawback does not necessarily render the approach useless in practice. An exhaustive computation is still feasible unless the number of attributes $|A|$ is very high. If only the attributes considered likely to be used for linking to external data sources are input to the anonymization process (demographic fields, typically), this may in itself bring the problem down to a feasible size. Additionally, simplification and absorption as discussed in Section 5.2.4 can often reduce the size of the problem significantly. And either way, computationally efficient heuristics exist that can be used to search for individual prime implicants. If only some prime implicants are computed, the heuristics can be equipped with a suitable bias so that the short (and thus presumably more “easily” detectable) patterns are given priority.

Computing the entries of the discernibility matrix is an $O(|U|^2|A|)$ process, and this will probably in practice be a limiting factor for very large databases. However, cer-


```

procedure anonymize( $x$ ) {
   $B \leftarrow \mathit{pick}(BCF(f_A(x)'))$ 
  for each ( $a \in B$ )
     $a(x) \leftarrow \top$ 
}

```

Figure 12.5: The algorithm from Figure 12.1 revisited. A far more efficient implementation. The function *pick* picks one of the prime implicants of $f_A(x)'$ according to some criterion. Such a prime implicant is easily computed.

tain simplifications can be made. First of all, the matrix is symmetric and has empty diagonal entries, so less than half the matrix actually needs to be computed. Moreover, since Boolean algebras have the property of multiplicative idempotence, we need only consider the number of distinct objects $|U_d|$, so the complexity can be brought down to $O(\frac{1}{2}|U_d|(|U_d| - 1)|A|)$. If the database is to be anonymized several times or with different recipient profiles, the matrix can be precomputed and stored so that it only needs to be computed once.

Lastly, it should be stressed that the exposition in this chapter has deliberately been made with the goal of highlighting the “data mining in reverse” aspect of anonymization via cell suppression. Hence, the algorithms presented here should be taken as conceptual ideas rather than as absolute guides for practical implementation. In Section 12.7.1 and Section 12.7.2, the observation was made that all the different minimal sets of attributes to suppress for an object x was given by $BCF(BCF(f_A(x)'))'$. However, since a Boolean function and its canonical form are semantically equivalent, there is no need to actually compute Blake’s canonical form twice. Once will suffice, since $BCF(BCF(f_A(x)'))'$ and $BCF(f_A(x)')$ are equivalent. Furthermore, since $f_A(x)$ is a POS formula, $f_A(x)'$ will be a SOP formula.⁴ The complexity of computing $BCF(f_A(x)')$ is therefore brought down to that of eliminating duplicates and carrying out absorption, as described in Section 5.2.4. But even this operation might not be needed to carry out in full, since we only need one prime implicant. A more efficient formulation of Figure 12.1 is given in Figure 12.5. For further efficiency, savings can likely be made if computation of a prime implicant is done on-the-fly while iterating over the discernibility matrix during function construction.

12.7.4 Record Order and Multiplicity

An issue that the Boolean reasoning algorithm does not address that could potentially be used to identify an object is the object’s order of occurrence in the database. Data is often entered sequentially into the database, and if the order of occurrence for an object is known, the record for that individual can always be located by a fixed look-up call. However, this situation can be easily remedied by simply permuting or scrambling the objects in the database, either before or after anonymization.

⁴If the function is implemented as a list of bit sets, then complementing the function involves no computation whatsoever, but is simply a matter of reinterpreting the data structure.

Another issue not explicitly addressed is that several records in the database may stem from the same patient, for instance if each record denotes a separate patient visit. If we know that a certain patient has extremely many visits, the database records for that patient may potentially be identified on the basis of such multiplicity information alone. This issue also has bearing to the validity of the assumption that the database is a single flat table and not multiple tables. Ways to overcome these problems are briefly discussed in [213].

12.7.5 Additional Suppression

Additional suppression in order to hamper suppressed database fields from being reconstructed was discussed in Section 12.4.4. Another issue to consider when taking a recipient's a priori knowledge into account is whether some fields that have not been suppressed perhaps should be suppressed. For instance, if a recipient somehow knows that the database in Table 12.3(b) contains exactly one person born in 1969, then it doesn't matter if a query for all people born in 1969 will return the set $\{x_3, x_4, x_5, x_6, x_8, x_9\}$. Since we have one match that is exact for the *BirthYear* field, the recipient has effectively located the row belonging to that particular person, and the set of possible values of the *HIV* field has been reduced to a singleton. If one has reason to believe that a non-trusted recipient of a processed database possesses background information of this type, a suitable post-processing scheme to counter-effect that information should be invoked.

Chapter 13

Diagnosing Acute Appendicitis

Portions of this chapter have previously been published as [142,143].

13.1 Introduction

Acute appendicitis is one of the most common problems in clinical surgery in the western world, and its diagnosis is sometimes difficult to make, even for experienced physicians. The costs of the two types of diagnostic errors in the binary decision-making process are also very different. Clearly, unnecessary operations are desirable to avoid. But failing to operate at an early enough stage may lead to perforation of the appendix. Perforation of the appendix is a serious condition, and leads to morbidity and occasionally death. Therefore, a high rate of unnecessary surgical interventions is usually accepted. Analysis of collected data with the objective of improving various aspects of diagnosis is therefore potentially valuable.

This chapter reports on an analysis of a database of patients thought to have acute appendicitis. The main objective of this study has been to address the following two questions:

1. Based only upon readily available clinical attributes, does a computer model perform better than a team of physicians at diagnosing acute appendicitis?
2. Does a computer model based upon both clinical attributes and biochemical attributes perform better than a model based only upon the clinical attributes?

These two issues have previously been addressed in the medical literature by Hallan et al. [71,72] using the same database of patients as presently considered. Multivariate logistic regression, the de facto standard method for analysis of binary data in the health sciences, was used in those studies. This paper addresses the same issues, but rather using one of the simplest approaches to rule-based classification imaginable,

namely a collection of univariate if-then rules. Univariate if-then rules are also referred to as 1R rules.

Section 13.3 briefly reviews the applied methodology, while Section 13.4 provides an overview of the data material and the experiments. The experimental results are given in Section 13.5 and are analyzed statistically in Section 13.6. A discussion and conclusions can be found in Section 13.7 and Section 13.8.

13.2 Preliminaries

The mathematical foundations of this chapter are outlined in Chapter 6 and Chapter 7. In particular, Section 6.4.1 about univariate rules and Section 7.4.3 and Section 7.4.6 about ROC curves and hypothesis testing are of relevance. Appendix C is also of relevance. The same notation and terminology is employed in the following.

A nice feature of 1R rules is that they are extremely fast and easy to construct, and do not make use of any computer-intensive constructs. As such, 1R rules can be seen as an extreme case of what Kowalczyk [108] calls *rough data modelling*. In rough data modelling collections of equivalence classes are used directly as classifiers. These equivalence classes are typically constructed by a bottom-up search procedure which can be quite fast if only attribute subsets of limited sizes are considered. 1R rules can also be viewed as a simple case of Mollestad's [126, 127] *default rules*, where the bottom-up lattice search is prematurely aborted. 1R rules have also previously been investigated by Holte [82].

13.3 Methodology

Let U denote the universe of patients, let A denote the set of classifier input attributes, and let d denote the outcome attribute. The set of 1R rules is defined as follows:

$$1R(A) = \bigcup_{x \in U} \bigcup_{a \in A} \{\text{if } (a = a(x)) \text{ then } (d = d(x))\} \quad (13.1)$$

If numerical attributes are to be properly incorporated into classification rules, they need to be discretized. Discretization amounts to searching for intervals or bins, where all cases that fall within the same interval are grouped together. This enables numerical attributes to be treated as categorical ones, and several algorithms for this purpose are available. In this study, for simplicity, all numerical attributes were discretized using a simple "equal frequency binning" technique. This fully automatic approach simply divides the attribute domain into a predetermined number of intervals such that each interval contains approximately the same number of cases.

To make the most out of scarce data, k -fold CV as described in Section 7.2.1 was employed. In the training stage of the CV pipeline, the union of the $k - 1$ blocks were first

discretized using an equal frequency binning technique with three bins. Intuitively, this corresponds to labeling the values “low”, “medium” or “high”. 1R rules were subsequently computed from the discretized union of blocks. In the testing stage, the hold-out block was first discretized using the same bins that were computed in the training stage, and the cases in the discretized hold-out block were then classified using standard voting among the previously computed 1R rules, as outlined in Section 6.4.1.

The results from the voting processes among the 1R rules were used to construct ROC curves, as described in Section 7.4.3. Performance measures for each iteration were harvested by computing the area under the ROC curves, computed using the trapezoidal rule for integration, as well as their associated standard errors as determined by Equation 7.23.

Two variations of k -fold CV were applied. First, a single 10-fold CV replication was performed, corresponding to how CV is traditionally employed. Additionally, five different replications of 2-fold CV was performed, as proposed by Dietterich [43, 44] and Alpaydin [7]. Running several replications with $k = 2$ was chosen to keep the test sets large enough so that the standard errors of the AUC estimates would be reasonable. With higher values for k and thus smaller test sets, computing the AUC based on only a few cases would mean that the resulting estimates would have a very high degree of variability,¹ something which in turn would mean that any significant differences that might be present would almost surely go undetected in a subsequent statistical analysis stage. Also, with $k = 2$, folds for each replication are completely independent since neither the training sets nor the test sets overlap.

The outlined procedure was done for the five different classifiers below, all with the goal of predicting the presence or absence of acute appendicitis on the basis of various clinical and biochemical attributes. By using the same seed to the random number generator used for sampling, we can reproduce all the random samples and thus ensure that the division into k blocks is identical across all classifiers.

- *Simple 1R*: A computer model realized through a collection of univariate if-then rules, based only upon readily available clinical attributes.
- *Extended 1R*: A computer model realized through a collection of univariate if-then rules, based upon the same attributes as the simple 1R model, but with additional access to the results of certain biochemical tests.
- *Physicians*: A classifier realized by probability estimates given by a team of physicians, based upon the same attributes as the simple 1R classifier.
- *Best B1R*: A 1R computer model realized through considering only the single best biochemical attribute.
- *Best C1R*: A 1R computer model realized through considering only the single best clinical attribute.

¹As an extreme case, consider leave-one-out CV. Then, the standard error would not even be defined.

Attribute	Description	Statistics
a_1	AGE	Age (years). 3–86 (22)
a_2	SEX	Male sex? 0.553
a_3	DURATION	Duration of pain (hours). 2–600 (22)
a_4	ANOREXIA	Anorexia? 0.693
a_5	NAUSEA	Nausea or vomiting? 0.708
a_6	PREVIOUS	Previous surgery? 0.093
a_7	MOVEMENT	Aggravation of pain by movement? 0.615
a_8	COUGHING	Aggravation of pain by coughing? 0.599
a_9	MICTUR	Normal micturation? 0.872
a_{10}	TENDRLQ	Tenderness in right lower quadrant? 0.860
a_{11}	REBTEND	Rebound tenderness in right lower quadrant? 0.553
a_{12}	GUARD	Guarding or rigidity? 0.307
a_{13}	CLASSIC	Classic migration of pain? 0.494
a_{14}	TEMP	Rectal temperature (°C). 36.4–40.3 (37.7)
a_{15}	ESR	Erythrocyte sedimentation rate (mm). 1–90 (10)
a_{16}	CRP	C-reactive protein concentration (mg/l). 0–260 (12)
a_{17}	WBC	White blood cell count ($\times 10^9$). 2.9–31 (12.1)
a_{18}	NEUTRO	Neutrophil count (%). 38–93 (80)
d	DIAGNOSIS	Acute appendicitis? 0.381

Table 13.1: Summary of attributes recorded for the 257 patients thought to have acute appendicitis. For binary attributes, the prevalence is given. For numerical attributes, the range and median are given. *DIAGNOSIS* is the decision attribute, indicating whether the patient really had acute appendicitis. Condition attributes *ESR*, *CRP*, *WBC* and *NEUTRO* are the results of biochemical tests, while the other condition attributes are readily available clinical parameters.

Lastly, a statistical analysis comparing their differences was performed using the methods of Hanley and McNeil [76] and Alpaydin [7].

13.4 Experiments

The methodology outlined in Section 13.3 has been applied to a medical database with 257 patients thought to have acute appendicitis, summarized in Table 13.1. The 257 patients were referred by general practitioners to the department of surgery at a district general hospital in Norway, and were all suspected to have acute appendicitis after an initial examination in the emergency room. Attributes $\{a_1, \dots, a_{14}\}$ are readily available clinical attributes, while attributes $\{a_{15}, \dots, a_{18}\}$ are the results of biochemical tests. The outcome attribute d is the final diagnosis of acute appendicitis, and was based on histological examination of the excised appendix.

After the clinical variables were recorded the physician also gave an estimate of the probability that the patient had acute appendicitis, based on these. The estimated

Attribute	Mean (SD)	Attribute	Mean (SD)
a_{17} WBC	0.783 (0.079)	a_7 MOVEMENT	0.603 (0.093)
a_{13} CLASSIC	0.725 (0.090)	a_{14} TEMP	0.598 (0.094)
a_{18} NEUTRO	0.718 (0.093)	a_3 DURATION	0.568 (0.042)
a_{11} REBTEND	0.703 (0.083)	a_4 ANOREXIA	0.568 (0.114)
a_{16} CRP	0.679 (0.126)	a_{15} ESR	0.556 (0.069)
a_2 SEX	0.655 (0.100)	a_5 NAUSEA	0.538 (0.063)
a_{12} GUARD	0.642 (0.102)	a_6 PREVIOUS	0.516 (0.062)
a_8 COUGHING	0.631 (0.081)	a_9 MICTUR	0.509 (0.057)
a_{10} TENDRLQ	0.613 (0.061)	a_1 AGE	0.504 (0.162)

Table 13.2: Results from a 10-fold CV run where each attribute is evaluated according to how well that attribute functions as a classifier on its own. The attributes are sorted according to their mean AUC values.

Classifier	Definition
Simple 1R	$1R(\{a_1, \dots, a_{14}\})$
Extended 1R	$1R(\{a_1, \dots, a_{18}\})$
Best B1R	$1R(\{a_{17}\})$
Best C1R	$1R(\{a_{13}\})$

Table 13.3: Defining the various 1R classifiers from Section 13.3 in terms of Equation 13.1. The best biochemical and clinical attributes were determined from the simulations in Table 13.2.

probabilities were given in increments of 10% from 0% to 100%. Nine residents with two to six years of surgical training participated in the study. These estimates directly define a realization of the certainty function ϕ defined in Section 7.4.

For a detailed description of the patient group and the attribute semantics, see [71, 72]. The same set of 257 patients was analyzed in [71], while a superset containing 305 patients was analyzed in [72]. Logistic regression was used in both studies.

To determine which attribute to employ for the single-attribute 1R classifiers, each attribute was initially evaluated according to how well each it functioned as a classifier on its own. This was done by carrying out 18 10-fold CV simulations, one for each attribute a_i , and in simulation i only employing rules involving attribute a_i . The results can be found in Table 13.2. As can be seen, most of the attributes obtain very low AUC scores when considered in isolation. One notable exception is biochemical attribute *WBC*, which obtains a mean AUC value of 0.783 with the employed discretization. The best clinical attribute is *CLASSIC*, with a mean AUC value of 0.725.

A summary of the various 1R classifiers from Section 13.3 defined in terms of Equation 13.1 can be found in Table 13.3.

13.5 Results

The results from the 10-fold CV simulation and the five different replications of 2-fold CV are given in Table 13.4 and Table 13.5, respectively. All simulations were carried out using the ROSETTA software system [145, 147]. The physicians and the simple 1R classifier both made use of the clinical variables only, while the extended 1R classifier had additional access to the results of the biochemical tests. On average, the extended 1R classifier seemed to perform somewhat better than both the simple 1R classifier and the team of physicians. The simple 1R classifier and the physicians seemingly perform approximately the same, with the former achieving a slightly better average score.

It is trivial to produce a classifier that classifies the training data perfectly. Although this would be a very optimistically biased estimate, 1R rules are so simple they do not possess enough degrees of freedom to overfit the data much. Reference ROC curves obtained when applying the classifiers to the full set of 257 patients from which they were constructed are displayed in Figure 13.1 and Figure 13.2. Table 13.6 provides details of the *WBC* and *CLASSIC* classifiers.

The exact same set of 257 patients has been previously analyzed by Hallan et al. [71] using multivariate logistic regression. In that study the set of cases was randomly split in two halves, and a regression model derived from one half was applied to the other half. This was done for 20 random splits, and the mean AUC and the standard deviation of the 20 samples was calculated. A regression model based upon only the clinical attributes had a mean AUC of 0.854 (0.028), while a regression model based on both the clinical attributes and the biochemical attributes had a mean AUC of 0.920 (0.024). Carlin et al. [27] have also analyzed the same set of patients, but used traditional rough set methods. This was also done with 20 random splits, and the mean AUC and the standard deviation of the 20 samples was 0.850 (0.024) for a model based on clinical variables, and 0.923 (0.023) for a model based on both the clinical attributes and the biochemical attributes. In all fairness it should be said that both the logistic regression and rough set studies only included clinical attributes $\{a_2, a_8, a_{10}, a_{11}, a_{12}, a_{13}\}$, and did not include biochemical attribute a_{15} . However, this was done because Hallan et al. found that adding other clinical attributes or attribute a_{15} did not improve the models further.

13.6 Analysis

In order to draw any trustworthy conclusions from the results in Section 13.5, a statistical analysis has been performed. The standard tool for comparing correlated AUC values is Hanley-McNeil's method, outlined in Section 7.4.6. However, this method is usually employed for a single two-way split only and not in a CV setting. In a CV setting one might very well ask what the models to assess really are. One could of course perform the Hanley-McNeil test for each fold, but it is somewhat unclear how to combine the collection of obtained p -values. Furthermore, one might question the

	Physicians	Simple 1R	Extended 1R	Best B1R	Best C1R
1	0.837 (0.081)	0.879 (0.071)	0.932 (0.053)	0.775 (0.093)	0.654 (0.109)
2	0.831 (0.089)	0.875 (0.078)	0.916 (0.065)	0.853 (0.084)	0.794 (0.097)
3	0.872 (0.074)	0.908 (0.064)	0.961 (0.042)	0.768 (0.096)	0.696 (0.106)
4	0.827 (0.104)	0.887 (0.087)	0.891 (0.086)	0.632 (0.129)	0.842 (0.100)
5	0.639 (0.113)	0.709 (0.106)	0.827 (0.087)	0.682 (0.109)	0.561 (0.116)
6	0.733 (0.103)	0.818 (0.089)	0.915 (0.063)	0.852 (0.082)	0.730 (0.104)
7	0.729 (0.102)	0.815 (0.088)	0.839 (0.082)	0.759 (0.098)	0.768 (0.096)
8	0.935 (0.061)	0.892 (0.077)	0.967 (0.043)	0.850 (0.089)	0.712 (0.112)
9	0.914 (0.077)	0.940 (0.065)	0.996 (0.017)	0.786 (0.112)	0.842 (0.100)
10	0.912 (0.084)	0.858 (0.104)	0.951 (0.064)	0.873 (0.099)	0.652 (0.138)
Mean	0.823 (0.089)	0.858 (0.083)	0.920 (0.060)	0.783 (0.099)	0.725 (0.108)
Median	0.834 (0.087)	0.877 (0.083)	0.924 (0.064)	0.780 (0.097)	0.721 (0.105)
SD	0.095 (0.016)	0.065 (0.015)	0.055 (0.022)	0.079 (0.014)	0.090 (0.013)

Table 13.4: Results from a single 10-fold CV run. AUC quantities are given for each iteration, with standard errors in parentheses.

		Physicians	Simple 1R	Extended 1R	Best B1R	Best C1R
1	1	0.795 (0.045)	0.844 (0.040)	0.911 (0.031)	0.802 (0.044)	0.722 (0.049)
	2	0.844 (0.037)	0.840 (0.037)	0.919 (0.027)	0.788 (0.042)	0.701 (0.048)
2	1	0.813 (0.040)	0.826 (0.039)	0.893 (0.032)	0.755 (0.045)	0.722 (0.047)
	2	0.819 (0.042)	0.838 (0.040)	0.925 (0.028)	0.848 (0.039)	0.699 (0.050)
3	1	0.755 (0.045)	0.823 (0.040)	0.898 (0.031)	0.828 (0.039)	0.703 (0.048)
	2	0.881 (0.035)	0.868 (0.037)	0.924 (0.028)	0.748 (0.047)	0.717 (0.049)
4	1	0.794 (0.042)	0.826 (0.039)	0.899 (0.031)	0.814 (0.040)	0.654 (0.050)
	2	0.839 (0.040)	0.865 (0.037)	0.930 (0.027)	0.766 (0.046)	0.768 (0.046)
5	1	0.791 (0.043)	0.833 (0.039)	0.899 (0.031)	0.786 (0.043)	0.714 (0.048)
	2	0.845 (0.038)	0.821 (0.041)	0.901 (0.031)	0.789 (0.044)	0.708 (0.049)
Mean		0.818 (0.041)	0.838 (0.039)	0.910 (0.030)	0.792 (0.043)	0.711 (0.048)
Median		0.816 (0.041)	0.835 (0.039)	0.906 (0.031)	0.788 (0.044)	0.711 (0.048)
SD		0.036 (0.003)	0.017 (0.001)	0.013 (0.002)	0.032 (0.003)	0.028 (0.001)

Table 13.5: Results from five different 2-fold CV runs, each replication with a different seed to the random number generator. AUC quantities are given for each fold and replication, with standard errors in parentheses.

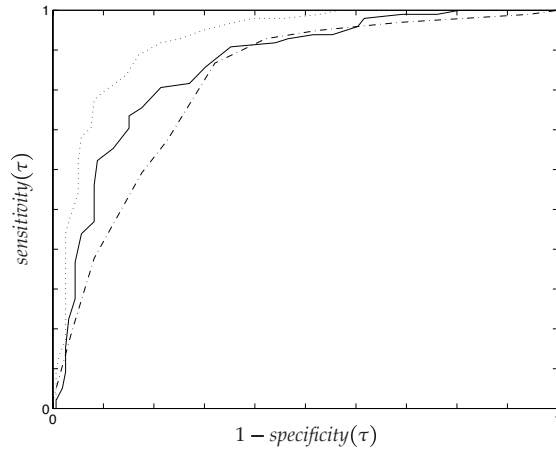


Figure 13.1: Reference ROC curves obtained when applying the classifiers to the full set of 257 patients from which they were constructed. The top dotted line represents the extended 1R classifier, while the middle solid line represents the simple 1R classifier. The physicians are represented by the bottom dashed line. The AUC values and their standard errors of the classifiers are 0.817 (0.029) for the physicians, 0.859 (0.026) for the simple 1R classifier, and 0.924 (0.019) for the extended 1R classifier.

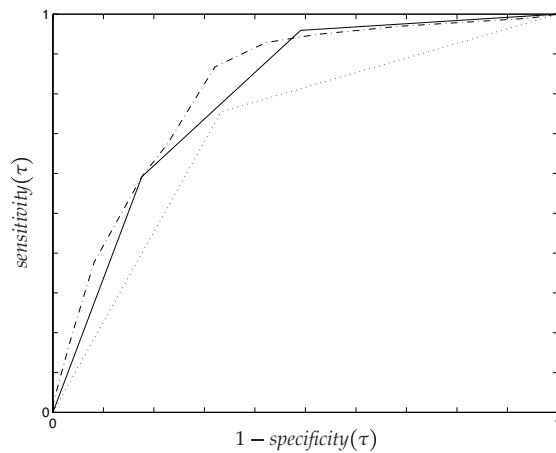


Figure 13.2: Reference ROC curves obtained when applying the classifiers to the full set of 257 patients from which they were constructed. The top dashed line represents the physicians, the middle solid line represents the *WBC* classifier, and the bottom dotted line denotes the *CLASSIC* classifier. The AUC values and their standard errors of the classifiers are 0.711 (0.034) for the *CLASSIC* classifier, 0.795 (0.030) for the *WBC* classifier, and 0.817 (0.029) for the physicians.

Classifier	Condition	$\phi(x)$
Best B1R	$WBC(x) < 9.7$	0.047
	$9.7 \leq WBC(x) < 14.0$	0.419
	$14.0 \leq WBC(x)$	0.674
Best C1R	$CLASSIC(x) = 0$	0.185
	$CLASSIC(x) = 1$	0.583

Table 13.6: Rules for predicting acute appendicitis using only attributes WBC or $CLASSIC$, the best biochemical and clinical attributes respectively. The listed cuts and $\phi(x)$ values were computed from the full set of 257 patients. For the results in Table 13.4 and Table 13.5, however, these are computed dynamically per fold and replication.

usefulness of this approach altogether if many folds are used due to the then small sizes of the test sets, as briefly discussed in Section 13.3. For this reason, the results from the 10-fold CV simulations in Table 13.4 are not statistically analyzed.² For 2-fold CV, however, the test sets may be large enough for such an analysis to be viable. The results in Table 13.5 have been analyzed using the method of Hanley and McNeil on a per-fold basis, and Table 13.7 contains the p -values per fold per replication.

Simply averaging the p -values in Table 13.7 to obtain a summary p -value would not capture any systematic variations in differences in performance across folds and replications, information which is obviously of importance. There are, however, statistical analysis methods that have been specifically designed for combining CV together with detection of differences in performance. One such method is the 5x2CV test, originally proposed by Dietterich [43,44] for comparing error rates, and subsequently improved by Alpaydin [7]. Applying the improved 5x2CV F -test to the results in Table 13.5 yields that there is no significant difference between the physicians and the simple 1R classifier ($p < 0.613$), but that the extended 1R classifier is significantly better than both the physicians ($p < 0.03$) and the simple 1R classifier ($p < 0.0003$). Furthermore, there is no significant difference between the physicians and the WBC classifier ($p < 0.766$). However, the physicians perform significantly better than the $CLASSIC$ classifier ($p < 0.048$).

²To get around the analysis problem of the small test sets and the subsequent high degree of uncertainty associated with the AUC estimates, one might envision that, for each classifier, all the pairs $(d(x), \phi(x))$ from all the test sets were “pooled” together. These pools would then form a larger collection of test classifications per classifier, and could be used for statistical comparisons. With AUC as the performance measure, however, pooling of the classifications of the test sets may result in mangling the results and introducing undesirable and unfair effects. For example, both the classification sets $\{(0,0.1), (0,0.5), (1,0.9)\}$ and $\{(0,0.05), (1,0.3), (1,0.7), (1,0.95)\}$ result in perfect AUC values of 1. Pooled together, however, the AUC value is 0.917. Since the individual classifications that contribute to the pool stem from training and testing with different data, it hardly seems fair to impose the same threshold across all pool contributors, as is done if the total pool is considered.

		Physicians Simple 1R	Simple 1R Extended 1R	Extended 1R Physicians
1	1	0.3043	0.0323	0.0127
	2	0.9171	0.0076	0.0559
2	1	0.7798	0.0289	0.0558
	2	0.6946	0.0057	0.0153
3	1	0.1634	0.0064	0.0016
	2	0.7558	0.0756	0.2573
4	1	0.4699	0.0084	0.0112
	2	0.5733	0.0305	0.0322
5	1	0.4098	0.0250	0.0192
	2	0.5963	0.0114	0.1746
Mean		0.5664	0.0232	0.0636
Median		0.5848	0.0182	0.0257
SD		0.2325	0.0214	0.0846

Table 13.7: Pairwise statistical analysis of the results in Table 13.5, done on a per fold per replication basis. All p -values are 2-sided and computed using the Hanley-McNeil method for comparing correlated AUC values, described in Section 7.4.6.

13.7 Discussion

If A is a singleton, then $1R(A)$ is a true univariate classifier. Otherwise, $1R(A)$ can be seen as a multivariate classifier composed of several univariate classifiers. The various 1R classifiers perform surprisingly well wrt. discrimination on the acute appendicitis database. A 1R approach will fail, however, in situations when certain combinations of parameters are needed to discern between objects. One such example situation is depicted in Figure 13.3. The present success of 1R rules suggests that such situations are not very common in the acute appendicitis database.

This study has focused on discrimination only, and has not touched upon the issue of calibration. Calibration is one of the issues discussed by Hallan et al. [72], and is an important feature if the classifier is to be used in an interactive decision-support setting. Preliminary investigations suggest that the simple and extended 1R classifiers with standard voting do not exhibit particularly good calibration, as one also from the smear effect in Figure 13.3 might be tempted to suggest holds for 1R models in general. However, as explored in Section 7.4.4, this might be rectified through a recalibration procedure. For the *WBC* and *CLASSIC* classifiers, the $\phi(x)$ values equal the probability of disease given x as estimated from the training data, and should hence be well calibrated.³

³This is because with these classifiers we can be sure that only one rule will fire. In fact, we can be sure that no more than one rule will fire if our rule base is constructed by overlaying an attribute set A , not necessarily a singleton, over the whole data table and reading off the descriptions of the equivalence classes and using these as rule antecedents. Such an approach resembles Kowalczyk's rough

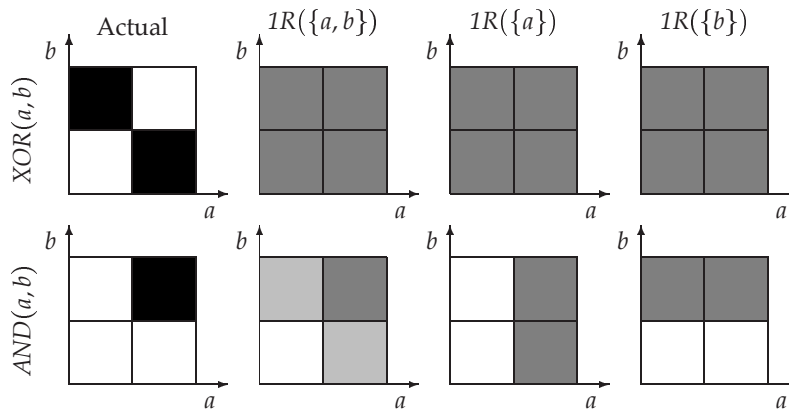


Figure 13.3: A 1R approach will fail in certain situations. This figure depicts two simple logical classification tasks, XOR and AND, and how various 1R classifiers will be able to model these situations by assigning certainty factors $\phi(x)$ to $x \in \{0, 1\}^2$ by voting. In the XOR situation, a 1R approach will fail completely. In the AND situation, however, $1R(\{a, b\})$ will be able to fully reproduce the true situation if we threshold $\phi(x)$ appropriately afterwards, even though the values around $(1, 1)$ get smeared. $1R(\{a\})$ and $1R(\{b\})$ will only be able to partially reproduce the true AND situation.

In Section 13.1, it was argued that performing a large number of unnecessary operations was preferable to missing any cases of acute appendicitis. This corresponds to prioritizing test sensitivity before test specificity. Selection of classifiers and thresholds under various cost scenarios is discussed in Section 7.4.3. As can be seen from Figure 13.1, the simple 1R classifier and the physicians display virtually identical performance in the area of ROC space of interest, while the extended 1R classifier outperforms them both everywhere.

A point that is often made in favor of inducing rule-based classifiers is the potential for knowledge discovery, since classification rules is a representation that can be inspected and interpreted by non-experts. 1R rules will in practice probably yield mostly known simple facts since they do not relate any attributes together in their if-part, but the computational effort to induce them is negligible and the resulting set of rules is often quite small and manageable. Moreover, simulations by Holte [82] showed that the best individual 1R rules were usually able to come within a few percentage points of the error rate that more complex models can achieve, on a spread of common benchmark domains. The present study suggests that this might be true for other performance measures, too.

The already small 1R models can probably be reduced even more without sacrificing the discriminatory performance, by filtering away those rules that deal with less important attributes. The results by Carlin et al. [27] and the attribute selection done by Hallan et al. [71] seem to support this conjecture.

data modelling [108].

The equal frequency binning technique used for discretizing the numerical attributes was chosen for the sake of simplicity. Other more advanced discretization techniques could potentially yield slightly better results. For example, preliminary experiments suggest that the AUC score for the extended 1R model can be increased slightly using the automatic algorithm outlined by Nguyen and Skowron [133].⁴ However, for the objectives in Section 13.1, optimality is not critical.

It may at first glance seem a bit odd that it is easier to detect a statistically significant difference between the two 1R classifiers than between the extended 1R classifier and the physicians, even though the average extended 1R performance is closer to the average simple 1R performance than to the average performance by the physicians. This can be attributed to the fact that the estimates made by the physicians display greater variance than the 1R estimates, which are thus more easily separable.

On a technical note, it should be stated that the 5x2CV F -test makes a simplifying assumption when it comes to applying the test to AUC performance measures. The term σ^2 in Equation C.3 should really be a function of i and j , i.e., not constant across all folds and replications. An estimate that takes this into account can be computed using Equation 7.25. However, the simplification that σ^2 is constant may not be all that bad, and hopefully the errors that this simplification introduces may even cancel each other out. Note that assuming a constant value σ^2 is not the same as setting $r = 0$ in Equation 7.25.

13.8 Conclusions

Based on the results in Section 13.5 and the analysis in Section 13.6, the answers to the two main questions raised in Section 13.1 are:

1. Based only upon readily available clinical attributes, does a computer model perform better than a team of physicians at diagnosing acute appendicitis? *Not significantly, at least not with a set of very simple 1R classification rules as the computer model.*
2. Does a computer model based upon both clinical attributes and biochemical attributes perform better than a model based only upon the clinical attributes? *Yes, even with a set of very simple 1R classification rules as the computer model there is a significant improvement when biochemical attributes are additionally taken into account. Furthermore, the best biochemical attribute alone performs on par with a team of physicians having knowledge of only clinical attributes.*

Of course, these conclusions are with respect to AUC as the performance measure and the equal frequency discretization scheme outlined in Section 13.3.

⁴The Boolean reasoning algorithm of Nguyen and Skowron is inherently multivariate. The utility of using multivariate discretization algorithms for 1R rules is debatable, since 1R rules do not make use of information about combinations.

Although not directly comparable, it hardly seems likely that the results reported in the literature and repeated in Section 13.5 based on logistic regression [71,72] or complex rough set models [27] are significantly different from the 1R results reported in this study.

Part V

Synopsis

Chapter 14

Summary and Results

14.1 Summary

Part I

In Chapter 2, the context was set by very briefly introducing the fields of data mining and KDD and medical informatics. Several general points were then made about the intersection of these fields, and arguments were made as to why many issues that are often ignored or glossed over in the machine learning literature ought to be considered when the medical domain is targeted. The items covered included comments on the availability, quality and management of medical data, and on some of the challenges that KDD practitioners face in this regard. The importance of attribute costs was also discussed, as was the implications this has for the indiscriminate use of Occam's razor as a guide for model selection. The validity of simplifying assumptions was then addressed. As for model assessment, it was argued that the widely popular performance measure of classification accuracy is by itself inadequate for the medical domain. More refined discriminatory measures are called for, and the need to consider the issue of calibration was also discussed.

Chapter 3 provided a thorough review of the literature on rough sets in medicine, and systematized the reported applications into main categories. Some of the appeal logical methods based on discernibility have as a foundation for medical machine learning was then outlined, and advantages and drawbacks of the approach were subsequently discussed in detail.

Part II

Chapter 4 gave a brief introduction to Boolean reasoning, the main tool for solving minimization problems in discernibility-based modelling. Through encoding a problem as a Boolean function, solutions to the problem could be obtained from prime implicants of this.

The main exposition of discernibility-based modelling was given in Chapter 5. Starting from the concept of a discernibility matrix, it was shown how a binary indiscernibility relation could be defined. Properties of this relation depended on how we chose to define the notion of discernibility, and the role of tolerance relations was demonstrated through an example from the medical domain. It was further shown how the indiscernibility relation could be employed as a basis for assembling rough set approximations, and how prime implicants of Boolean discernibility functions could be used to discard functionally superfluous information. By varying which objects we chose to discern between and how we defined discernibility to take place, a broad range of different functions with different interpretations and applications could be constructed.

In Chapter 6 it was shown how prime implicants of the Boolean discernibility functions could be employed as templates for minimal conjunctive patterns and decision rules. Important numerical quantities of such patterns and rules were then presented, and it was investigated how, via voting, ensembles of decision rules could be utilized to assign certainty measures to decision classes for new and unseen objects.

Chapter 7 discussed ways of assessing classifier performance of particular relevance for medical applications. The concepts of discrimination and calibration for evaluation of binary outcome classifiers were discussed, and ROC analysis was explored in detail. Some important statistical tests for comparing performance estimates were subsequently presented.

Part III

Chapter 8 gave an initial presentation of the ROSETTA software system for discernibility-based data analysis, developed to meet the requirements imposed by the medical domain as discussed in the previous chapters. An overview of the system's main features was provided. ROSETTA has already served as a simulation tool for researchers world-wide in a broad range of domains, also outside of medicine.

An example of how ROSETTA could be employed to analyze a medical database was given in Chapter 9. As a case study, a real-world database on coronary heart disease was selected. It was demonstrated how decision rules could be extracted and their classificatory performance evaluated, both in a simple two-way split experiment as well as in a computer-intensive 10-fold CV setting. How background cost information could be put to use was also demonstrated. Although optimal results was not the goal of the case study, the obtained results compare favorably with results reported in the literature.

Chapter 10 provided a software engineering view of the ROSETTA system, dissecting the system's architecture and explaining the rationale underlying some of the central design decisions by relating to issues arising from the KDD process. Examples of how the ROSETTA C++ library could be employed to quickly assemble relevant KDD pipelines were also given.

ROSETTA can be used to tackle a wide variety of tasks, not only traditional data mining and KDD applications. The suite of applications in Part IV serve as a demonstration of this.

Part IV

In Chapter 11, an application was given on how the semantics of the boundary region of a rough set could be exploited to identify interesting subgroups of a larger population. Through monitoring how the boundary region changed when some information was withheld, one could pinpoint exactly which patients that caused the subsequent drops in approximation sensitivity and specificity. As this patient group corresponded to those for whom knowledge of the withheld information was crucial, identifying this group could be relevant in the context of developing rules for screening. As a case study, a real-world database on forecasting hard cardiac events was analyzed. Not only did the simulations suggest that potential savings could be made with respect to carefully selecting which patients that should have a scintigraphic scan performed, but the identified population subgroup could also be circumscribed fairly well using other available information.

Chapter 12 reported on an application where discernibility considerations were used to decide on which information to suppress before disseminating sensitive medical data. Being an exercise in data mining in reverse, the proposed algorithm was based on the observation that if we knew which minimal combinations of information that could be used to identify individuals, then we also knew which information that ought to be suppressed to block such identification from taking place. Some thoughts on how the degree of anonymization could be controlled were also presented.

An application on using very simple classification rules to diagnose acute appendicitis was presented in Chapter 13. Through extensive simulations on a real-world database, the diagnostic performance of a small collection of very simple rules was compared to that of a group of experienced physicians. Based only on clinical attributes, the rules performed better on average, but not significantly so in a statistical sense. If biochemical attributes were added to the computer model, however, the performance improvement became significantly better. It was also demonstrated that a computer model based only on a single biochemical attribute performed no worse than a team of physicians working with clinical information.

14.2 Main Results and Contributions

For a data-driven modelling methodology to gain broad acceptance in a new domain and generate an impact, the methodology has to be prepared to undergo some adaptation and assimilation to the domain, at least superficially. This goes not only to how the actual inductive learning takes place and domain-specific factors that have to be considered, but also to how the resulting models are presented and evaluated. This

this thesis has shown how discernibility-based methods can be equipped to possess several qualities that are needed for analyzing tabular medical data, and how these models can be evaluated according to current standard measures used in the health sciences. To this end, tools have been developed that make this possible, and some novel medical applications have been devised in which the tools are put to use.

The perhaps most immediate contributions of this thesis are Parts III and IV, the ROSETTA system and applications comprising original contributions in their entirety. Parts I and II largely, but not exclusively, report and build on relevant work reported in the literature. In order to cross-fertilize the affected fields, scattered works have been brought together and assessed, rephrased and generalized to fit a certain context.

Through the work presented in this thesis, the versatility of discernibility-based methods for empirical modelling in general has been demonstrated. By examining aspects of the medical domain, several issues for data-driven modelling that become particularly important in the context of medicine have been illuminated. Some of the most important identified issues have subsequently been held up to the current state-of-the-art in modelling based on the relatively young fields of discernibility and rough sets, and it has been shown how the relevant methodology can be suitably adapted and employed.

- Evaluation measures central in the health sciences have been carried over to the field of rough sets. It has been demonstrated how rough set classifiers can be evaluated through ROC analysis, how calibration can be appraised, and how set approximations can be assessed in terms of sensitivity and specificity. This seems to be novel in the context of rough sets.
- It has been outlined how the use of attribute costs can be embedded in the model construction process. By employing cost information in the reduction process, low-cost rather than low-cardinality solutions can be obtained. Costs can also be made use of for model filtering and evaluation purposes.
- It has been made clear how one by overloading the notion of discernibility can cater for, e.g., hierarchically ordered attribute values. In the medical domain, such hierarchies can be encountered in some CMTs. Missing values can be perceived as a special case of this.

Starting from a theoretically well-founded approach, a modelling methodology has been established and an extensive toolkit for discernibility-based empirical modelling has been designed and implemented. The software system, ROSETTA, is a robust, user-friendly and powerful system for discernibility-based KDD, and has by design been accommodated with the necessary features for analyzing tabular data from the medical domain. A large number of researchers from all over the world have downloaded ROSETTA and employed the system in published works. As such, the ROSETTA system constitutes a contribution to the advancement of applied discernibility-based modelling, both within medicine as well as in general.

By and large, discernibility-based data analysis can be varied along two main axes: Which objects in the universe of discourse that we deem it necessary to discern between, and how we define that discernibility among these objects is allowed to take place. This thesis has explored various facets of this also in three novel and distinctly different medical applications:

- By observing the flux in approximation regions when our indiscernibility relation changes due to the removal of some information, an approach to identification of interesting population subgroups was presented. Geared towards pinpointing objects for whom a certain battery of expensive medical tests could be avoided, a prognostic problem in cardiology was used as a case study. For the purpose of predicting future hard cardiac events, a group of patients was identified for whom a scintigraphic scan could be avoided wrt. their discernibility status. Furthermore, meta-approximations showed that this group could be fairly well circumscribed. On a general level, the proposed method is a contribution towards lowering costs and increasing efficiency in healthcare.
- Discernibility considerations coupled with Boolean reasoning enable us to compute minimal, identifying patterns of various kinds. By noting that we can very well choose to obfuscate these patterns, an approach to anonymization of sensitive medical data via cell suppression was outlined. In an increasingly computerized world, the issue of preserving confidentiality is not likely to become any less important. By showing how discernibility can provide a foundation for cell suppression, a contribution has been made to the sound management of sensitive medical data.
- Through reducing the set of discerning attributes to singletons, it was investigated how extremely simple classification rules could be used to diagnose acute appendicitis. From a clinical viewpoint, it is of interest to determine how well computer models compare with medical doctors. Extensive simulations showed that a simple computer model was on par with a team of experienced surgeons with respect to diagnosing acute appendicitis, and the added value of certain biochemical attributes was also demonstrated.

Chapter 15

Further Work

15.1 Introduction

This chapter comments on issues that are candidates for future work. The items are categorized into the following sections:

- *General*: Comments on a few general issues related to discernibility-based modelling as discussed in Parts I and II.
- *Tools*: Discusses issues pertaining to software tools and the ROSETTA system in particular, as presented in Part III.
- *Applications*: Suggests some further directions to investigate for the applications presented in Part IV.

These items are discussed in Section 15.2, Section 15.3 and Section 15.4, respectively.

15.2 General

In Section 2.4.3 the distinction between discrimination and calibration was outlined. The popular approach of using standard voting to resolve conflicts in rule-based classifiers seems to perform adequately for discrimination, as witnessed by numerous applications in the literature. As for calibration, the status is more uncertain since this is seldom if ever reported on. Both the applications in Chapter 9 and in Chapter 13 discriminate well, while preliminary investigations suggest that the latter is not particularly well calibrated but exhibits a clear tendency to cluster the $\phi(x)$ values densely around a central value. It is clearly of interest to develop solid recalibration methods to remedy this and similar situations, as outlined in Section 7.4.4. Further research is needed in this regard. Additionally, more studies are needed to obtain a clearer picture of the relative merits wrt. calibration of the voting schemes described in Section 6.4.

How IDGs could be used to overload the notion of discernibility for each condition attribute was described in Section 5.2.1. A natural continuation of this line of thought is to further allow IDGs to be employed for decision attributes as well. A suitable place to introduce such a scheme would be in Equation 5.37, which would be generalized to take the IDG for the decision attribute into account.

Tolerance relations in rough set data analysis can in some sense be said to still be in their infancy. The theory is mature and well developed, but their use for practical applications in data mining and KDD is rarely reported on since, often, equivalence relations suffice or are simpler to work with. However, tolerance relations are attractive for several reasons. Not only can one use IDGs to take attribute value semantics into account, but their use can also minimize the need to have to discretize or otherwise preprocess the data. More experience is needed with hands-on uses of tolerance relations, as this might identify areas of pragmatical importance.

Approximate decision rules are usually produced in the induction step by means of various techniques, e.g., dynamic reducts or methods inspired by default reasoning [126, 127]. Both these techniques are fairly computationally intensive. In the case study in Chapter 9, an approximate solution to the discretization problem was employed. The relative merits remain to be determined of (i) letting a traditional algorithm compute proper reducts from data tables where inconsistencies have been introduced in the preprocessing stage versus (ii) letting a computer-intensive algorithm compute approximate reducts from data tables where discernibility has been fully preserved. Also, how the two approaches are best combined is of interest to ascertain.

A great deal of work can be done to improve upon some of the areas discussed in Section 3.4. E.g., the potential rule-based models have for offering traceable or explainable predictions is sometimes hampered by the size of the rule set. Schemes to make large rule-based models more manageable are clearly of interest to develop.

A more formal connection between dynamic reducts from Section 5.2.6 and bagging and boosting from Section 7.2.1 should be established. Conceptually, the techniques share a number of common traits that hint of making a formal unification possible. For example, a discernibility function constructed from a sampled subsystem is likely to be similar to one constructed from a bootstrapped information system. Ideas from boosting might also yield valuable clues on how to best sample subsystems in a dynamic reduct framework.

In discernibility-based data analysis, Boolean reasoning is traditionally used to compute reducts, i.e., minimal sets of attributes. But similar techniques can also be applied to compute subsets of objects that are minimal with respect to some criterion. Interesting and practical applications of this should be able to find.

Although, as demonstrated, some background knowledge can enter the discernibility-based modelling process, it would be of great interest to develop schemes for model induction in the presence of domain theories, e.g., in the situation where initial causal relationships between attributes can be established a priori by domain experts. The use of such background knowledge should be fully optional, though, since in general such relationships may be difficult to establish.

15.3 Tools

This section lists some possible future enhancements to the ROSETTA tool. Some of the suggestions are simple to implement and can easily be added, while others require substantial programming resources.

There are several features that can be added to ROSETTA without too much effort. All the groundwork has already been laid, and the required elements for implementing the features are already in place in the ROSETTA library. These features include:

- Viewing of the general conjunctive patterns from Section 6.2.3 as association rules in the GUI.
- The use of IDGs for preprocessing purposes, along the lines of Hu et al. [86] where they outline how concept hierarchies can be used to “group” symbolic attribute values together as a kind of discretization.
- Cost-assessment of a set of attributes where shared costs are handled, as discussed in Section 9.4.1. In order for a more realistic assessment of costs for use in guiding computations and model filtering and evaluation, this feature should be added.
- Augment ROSETTA with more formats for data export, allowing improved visualization of ROSETTA output.¹ Models could also be exported to the developing PMML² language [37] to ease the interchange of models between software.
- Add support for other resampling methods than CV. Also, augmenting ROSETTA with a bagging or boosting feature would be useful.
- Extend the ROSETTA GUI with a simple decision-support module designed to offer both case-based and model-based explanations.

Further developments that would require a medium amount of programming efforts include:

- Modularize the executable using dynamic link libraries (DLLs). Orthogonal components such as the GUI, the kernel and the RSES library could then reside in separate DLLs, allowing for individual updates. End-user written DLLs could also be detected and loaded into the system at run-time, making ROSETTA extensional in a dynamic sense.

¹There are several extensive data mining systems that tightly integrate visual and analytical techniques, with specialized visualization techniques for decision trees, rule sets and decision tables. See, e.g., MineSet [125] from Silicon Graphics, Inc. Output from ROSETTA could be piped into such systems.

²The Predictive Model Markup Language (PMML) is a simple markup language that uses XML as its meta-language in a manner similar to the way Hypertext Markup Language (HTML) uses SGML as its meta-language.

- Substitute existing GUI widgets with controls that offer alternative and improved ways of displaying data to the user.³ The incorporation of such into the ROSETTA GUI might further increase the usability of the system, and, as man-machine and user interface technologies progress, prolong the time until the current ROSETTA interface seems outdated.
- Optimize the memory allocation scheme to increase speed and avoid memory fragmentation.

Extensions that will probably require considerable programming and design work include:

- Parallelization of the computer-intensive procedures in ROSETTA amenable to such. For instance, entries in a discernibility matrix can be computed independently, and genetic algorithms are well suited for distributed computing. To increase efficiency, selected critical portions of ROSETTA could be rewritten to exploit this, e.g., using the Parallel Virtual Machine [153] software package.⁴
- Development of a general programming language and an interpreter for ad hoc programming, where the features of the ROSETTA kernel are offered as primitives, and that can be invoked from within the ROSETTA GUI. For expressive simplicity, the language should possess a modelling-oriented syntax. An initial investigation on how this might be done has already been carried out [2]. Figure 10.1 also hints in the direction of visual programming languages.
- Development of data structures and algorithms that can handle very large volumes of data, such as, e.g., molecular biological sequence databases. Data access patterns and spatial requirements for storing temporary working structures would have to be examined closely, and even quadratic algorithms may yield unacceptable running times for extremely large databases.

Several of the extensions proposed in this section do not necessarily have to be done within ROSETTA. As an alternative, several of the features can equally well be implemented as, e.g., Perl scripts that employ the command-line version of ROSETTA as a computational engine, or scripts that operate on and transform and process ROSETTA input and output.

³For example, in the case of very large project trees, so-called hyperbolic trees might provide less cluttered views than ordinary trees. Also, large data tables can be displayed to the user in more concise ways than as simple 2D spreadsheets. Several companies specialize in developing and selling such advanced GUI controls, see, e.g., the homepage of Inxight Software, Inc. [89]. Their Hyperbolic Tree and Table Lens controls are GUI widgets of the types described. A number of good, free GUI controls and programming ideas can be found at the CodeGuru website [34].

⁴Parallel Virtual Machine (PVM) is a freely available software package that permits a heterogeneous collection of UNIX and/or NT computers hooked together by a network to be used as a single large parallel computer. PVM has more or less become a world-wide de facto standard for heterogeneous distributed computing.

15.4 Applications

Identification of Population Subgroups

Chapter 11 proposed an application that exploited the semantics of rough set boundary regions. With respect to the particular database \mathcal{A} in Table 11.1, there are two obvious directions for further exploration:

- Induce decision rules from the original database \mathcal{A} . Examining the resulting rules might reveal medically interesting relationships between factors contributing to future hard cardiac events. Quantitative assessment of the prediction quality of the induced rules, e.g., in the form of an ROC analysis, would also be of interest.
- Induce decision rules from the derived database \mathcal{A}'_{π} . Examining the resulting rules might reveal concise descriptions of the patients that migrate into the boundary region, and hence help to define a screening protocol.

There are several reasons why these tasks were not carried out in Chapter 11. Firstly, they lie outside the scope of the presented identification process, although they do form a logical precursor and sequel to this. But moreover, the following factors were present:

- As argued in Section 3.4.2, very close cooperation between the data analyst and the domain experts responsible for collecting the data is necessary for any actual knowledge discovery to take place. Unfortunately wrt. to interpretation of rules for clinical relevance, our contacts at the Dutch hospital where the data material was collected were technical and not medical personnel. Also, geographical distance was a prohibitive factor.
- The previous study [64] of the database in Table 11.1 was of a pure KDD nature, in which the full data material was used to derive a logistic regression model that was subsequently interpreted to find factors contributing to future hard cardiac events. As such, no portions of the data material were held aside for testing and assessment of predictive performance. Thus, we had no baseline with which to compare the predictive performance of any decision rules derived from \mathcal{A} .

Lastly, an item that would be nice to do is to graphically depict the quality of the rough set approximations by plots akin to ROC curves. Fixing A and X , the precision parameter π can be varied and Equation 11.6 and Equation 11.7 be invoked to generate plot points. These plots may in turn be used to determine a suitable value for π in practice.

Anonymization of Sensitive Data

Chapter 12 presented how discernibility considerations could provide a foundation for cell suppression. Further research can be done along several axes:

- Develop alternative ways of quantifying anonymity, letting Equation 12.2 take into account some of the issues discussed in Section 12.4.1. Equation 12.5 could also be further developed along similar veins.
- Develop good heuristics for determining good object-orderings for anonymization on the database level, as discussed in Section 12.4.2.
- Develop good control strategies for guiding the anonymization process. Possible heuristics could, e.g., focus on how the indiscernibility sets become enlarged, i.e., which objects that get added at each iteration. Also, how should recipient profiles and preferences best be captured by the control heuristics?
- Investigate further how cell suppression, generalization and outlier removal complement each other. How can an “optimal” mixture of these techniques be determined for practical anonymization?

Diagnosing Acute Appendicitis

Chapter 13 explored how very simple classification rules could be used to diagnose acute appendicitis. Candidate items for further research include:

- Investigations of issues related to 1R models and calibration, since the experiments addressed discrimination only.
- Investigations of how various discretizations influence the result. Can a significant difference be detected between the simple 1R classifier and the team of physicians if another discretization method is used?
- How can the already small collections of 1R rules be filtered down to even smaller rule sets?
- Would the introduction of bagging or boosting be able to make the simple 1R model significantly better than the team of physicians?

Furthermore, it could be of interest to induce multivariate rules from the database in Table 13.1 for the purpose of medical interpretation and KDD.

Part VI

Appendices

Appendix A

Nomenclature

In addition to standard symbols from set theory and logic, the notation used in this work is summarized below. Pointers to the sections where the notation is first defined and introduced are also given. Lastly, a list of some of the abbreviations employed throughout the thesis is given for reference.

Notation

Symbol	Description	Section
\mathcal{A}	An information or decision system.	5.2, 5.3
A, B, \dots	Sets of attributes in an information system.	5.2
$\underline{A}_\pi X, \overline{A}_\pi X$	Lower and upper approximations of a set.	5.2.3
a_1, a_2, \dots	Individual attributes in an information system.	5.2
a_1^*, a_2^*, \dots	Boolean variables corresponding to attributes.	5.2.4
B	A carrier set in a Boolean algebra.	4.2
C	A confusion matrix.	7.3
d	A decision attribute in a decision system.	5.3
$\partial_{A'} \partial_A^\pi$	A generalized decision attribute.	5.3.2, 5.3.3
E_a	A set of edges in an IDG.	5.2.1
$f_{A'}, f_A^d$	A discernibility function of an object.	5.2.4
$F_A(x)$	A characteristic formula of an object.	6.2.3
$g_{A'}, g_A^d$	A discernibility function of an information system.	5.2.4
G_A	An indiscernibility graph.	5.2.2
$M_{A'}, M_A^d$	A discernibility matrix.	5.2.1, 5.3.3
PAT	A set of general patterns.	6.2.3
$R_{A'}, R_A^d$	An indiscernibility relation.	5.2.2
RED	A set of reducts.	5.2.5, 5.3.5
RUL	A set of decision rules.	6.3.1
U	The universe of objects in an information system.	5.2

Continued...

		... Continued
v	An attribute value.	5.2
V_a	The value set of an attribute.	5.2
X_1, X_2, \dots	Decision classes.	5.3.1
x, y, \dots	Individual objects in the universe of discourse.	5.2
ϕ	Estimates the certainty that an object has outcome 1.	7.4
κ	A classifier.	7.1
π	A precision level of a rough set approximation.	5.2.3
θ	Binarizes the output of ϕ wrt. a threshold.	7.4
τ	A threshold value employed by θ .	7.4
μ_A^X	The rough membership function.	5.2.3
\top	Denotes a missing value.	5.2
\preceq_a	Defines a partial order on an attribute's value set.	5.2.1

Abbreviations

Abbreviation	Description	Section
AUC	Area under the ROC curve.	7.4.3
CMT	Controlled medical terminology.	2.3.1
CV	Cross-validation.	7.2.1
DBMS	Database management system.	3
DLL	Dynamic link library.	15.3
DSS	Decision support system.	2.3.1
EMR	Electronic medical record.	2.3.1
GUI	Graphical user interface.	8.1
ICU	Intensive care unit.	2.4.3
IDG	Indiscernibility definition graph.	5.2.1
KDD	Knowledge discovery in databases.	2.2
HSV	Highly selective vagotomy.	3.2.1
MFC	Microsoft foundation classes.	B
NPV	Negative predictive value.	7.3
ODBC	Open database connectivity.	8.4.1
PMML	Predictive model markup language.	15.3
POS	Product of sums.	5.2.4
PPV	Positive predictive value.	7.3
PVM	Parallel virtual machine.	4
ROC	Receiver operating characteristic.	7.4.3
RSES	Rough Set Expert System.	8.2
RTTI	Run-time type identification.	10.3.2
SE	Standard error.	7.4.6
SOP	Sum of products.	4.3
SQL	Structured query language.	3
STL	Standard template library.	B

Appendix B

The ROSETTA C++ Library

Directories

A brief overview of the directory structure of the ROSETTA C++ library is found below. The library is distributed across several directories, each with strict rules governing their interdependencies.

kernel Contains the computational kernel of ROSETTA.

basic Contains elementary structures, e.g., smart pointers and classes for reference counting, basic structures such as strings and bitsets, macro definitions, management of identifiers, type definitions of vectors, maps, sets and other basic containers, etc.

system Contains wrappers for standard C system files. Functions as a portability layer.

sys Contains wrappers for certain standard C system files. Functions as a portability layer.

stl Contains the STL implementation in use. Implementations of basic containers such as vectors, maps and sets, etc.

structures Contains code for higher-level structures such as decision tables, collections of reducts and rules, discernibility matrices and functions, ROC curves, etc.

algorithms Contains code for higher-level algorithms such as algorithms for discretization, computation of reducts, filtering of reducts and rules, voting, import/export routines, etc.

utilities Contains code for various utilities, such as random number generators, tools for statistical hypothesis testing, computation of partitions, common mathematical operations, etc.

rses Contains code relevant to the RSES library.

library Contains legacy code from the RSES library.¹

structures Contains encapsulating wrappers for some of the structural objects from the RSES library.

algorithms Contains encapsulating wrappers for some of the algorithms from the RSES library.

sav Contains code relevant to the SAV library.

library Contains legacy code from a library for genetic computations.²

ea Contains general code for genetic algorithms.

hits Contains code specific for applying genetic algorithms to compute minimal hitting sets.

algorithms Contains encapsulating wrappers for some of the algorithms from the SAV library.

frontend Contains the GUI front-end of ROSETTA.

algorithms Contains algorithmic objects that depend on MFC and the Windows platform, specifically ODBC code.

dialogs Contains code for handling the logic behind the dialog boxes.

structuredialogs Contains code for handling the logic behind the dialog boxes related to structural objects such as statistics dialogs, etc.

algorithmdialogs Contains code for handling the logic behind the dialog boxes related to algorithmic objects such as dialogs for entering algorithm parameters, etc.

views Contains code for displaying structural objects in the front-end such as views for decision tables, project trees, etc.

common Contains potentially front-end dependent code called from the kernel, e.g., methods for giving error messages, installing prototype objects and their associated dialogs, etc.

Statistics

Table B.1 contains some library statistics.

Development Tools

The baseline platform for ROSETTA development has been Windows NT, and the primary development environment has been the Microsoft Visual C++ compiler and its

¹Developed at the Group of Logic, University of Warsaw, Poland [69].

²Developed by Staal Amund Vinterbo, Knowledge Systems Group, NTNU, Trondheim.

Directory	Files	Lines
kernel/basic	32	7386
kernel/system	16	35
kernel/system/sys	2	2
kernel/structures	88	28416
kernel/algorithms	168	28307
kernel/utilities	25	4455
kernel/rses	2	228
kernel/rses/library	97	17121
kernel/rses/structures	12	5478
kernel/rses/algorithms	26	3437
kernel/sav/library/ea	64	3012
kernel/sav/library/hits	16	1218
kernel/sav/algorithms	2	902
frontend	14	5153
frontend/algorithms	4	1081
frontend/dialogs	8	1083
frontend/dialogs/structuredialogs	18	2498
frontend/dialogs/algorithmdialogs	96	13883
frontend/views	24	9942
common	8	2118
	722	135755

Table B.1: ROSETTA C++ library statistics. The number of lines of C++ code is counted using a simple line count, and thus also includes blank lines and comments. Binary resources (e.g., bitmaps) are not included, neither are some external libraries (e.g., STL and Objective Grid Lite).

integrated development environment [234]. The GUI front-end was developed using Microsoft Foundation Classes (MFC), a collection of C++ classes that simplify Windows GUI programming, and is implemented as an MFC Multiple Document Interface application following the standard MFC document/view architecture. A commercial third-party MFC grid control package, Objective Grid Lite [135] from Stingray Software, was used to realize the matrix-like views of various structural objects in the GUI. For reasons of portability, MFC is not employed in the computational kernel. Instead, basic containers such as vectors and associative maps and basic algorithms such as sorting are culled from the Standard Template Library (STL). The STL variant used has been STLport [211], an adaptation of Silicon Graphics' STL implementation where great care has been taken to make the templates as portable and robust as possible against compiler bugs and platform quirks. The ROSETTA installation program was generated by GkSetup [66], and Microsoft Visual SourceSafe [235] was used for source code version control. Versions currently in use at the time of writing are Windows NT 4.0, Visual C++ 6.0, Objective Grid Lite 6.1, STLport 3.2, GkSetup 1.73 and Visual SourceSafe 6.0.

Two design requirements from Section 10.2.1 were that the kernel should be portable and independent of the front-end. Indeed, the kernel has been successfully compiled under version 1.1.2 of the GNU egcs compiler on a Sun SPARCStation running SunOS 5, and a command-line version of ROSETTA under UNIX is fully operative. Porting the GUI front-end to run on other platforms than Windows should also in principle be possible, but would, due to the use of MFC, require some relatively expensive commercial third-party libraries.

Appendix C

The 5x2CV Test

The 5x2CV F -test, proposed by Alpaydin [7] as a robust improvement to a test proposed by Dietterich [43], can be used to quantitatively compare the performance of two classifiers. As its name implies, the test is based on performing five replications of 2-fold CV.

Let Δ_{ij} denote the *difference* between the performance measures of the two classifiers on fold $j \in \{1, 2\}$ of replication $i \in \{1, \dots, 5\}$. The average difference in performance on replication i is $\bar{\Delta}_i$ and the estimated variance is s_i^2 .

$$\bar{\Delta}_i = \frac{(\Delta_{i1} + \Delta_{i2})}{2} \quad (\text{C.1})$$

$$s_i^2 = (\Delta_{i1} - \bar{\Delta}_i)^2 + (\Delta_{i2} - \bar{\Delta}_i)^2 \quad (\text{C.2})$$

Let H_0 denote the null hypothesis that the two classifiers perform equally well. Under H_0 , Δ_{ij} can be treated as being $N(0, \sigma^2)$ distributed, and we have:

$$A = \sum_{i=1}^5 \sum_{j=1}^2 \frac{\Delta_{ij}^2}{\sigma^2} \sim \chi_{10}^2 \quad (\text{C.3})$$

$$B = \sum_{i=1}^5 \frac{s_i^2}{\sigma^2} \sim \chi_5^2 \quad (\text{C.4})$$

$$f = \frac{A/10}{B/5} = \frac{\sum_{i=1}^5 \sum_{j=1}^2 \Delta_{ij}^2}{2 \sum_{i=1}^5 s_i^2} \sim F_{10,5} \quad (\text{C.5})$$

We then reject H_0 if the statistic f is sufficiently large. For 95% confidence, $f = 4.74$.

Appendix D

Discretization

Main Categories

Broadly speaking, we can view discretization as a function D that applied to a system \mathcal{A} yields a system \mathcal{A}' such that the attributes $\{a_1, \dots, a_m, d\}$ in \mathcal{A} are transformed into attributes $\{a'_1, \dots, a'_m, d'\}$ in \mathcal{A}' , where a'_i is the transformed version of a_i and $|V_{d'}| \leq |V_d|$ and $|V_{a'_i}| \leq |V_{a_i}|$ for all $i \in \{1, \dots, m\}$.

$$D(\underbrace{(U, A \cup \{d\})}_{\mathcal{A}}) = \underbrace{(U, A' \cup \{d'\})}_{\mathcal{A}'} \quad (\text{D.1})$$

In ROSETTA, algorithms for automatic discretization of numerical attributes generally fall into one of three categories:

- *Each condition attribute is considered in isolation, and no knowledge of any outcome or decision attribute is employed in the process:* An example of this is a simple equal frequency binning technique, which operates on the attribute histogram and creates a specified number of intervals in such a manner that the number of objects that fall into each interval is approximately the same.
- *Only one condition attribute is considered at a time, but is done so in conjunction with the decision attribute:* An example of this is the algorithm by Dougherty et al. [47], which recursively partitions the set of attribute values so that a local measure of entropy is optimized, until a stopping criterion based on the minimum description length principle is met. Naive approaches also sort into this category.
- *All condition attributes are considered simultaneously, and are done so in conjunction with the decision attribute:* An example of this is the algorithm by Nguyen and Skowron [133], in which combinations of all naively generated cuts of all attributes are considered together, and the discretization problem is reframed as a Boolean reasoning problem such that the discernibility in the original decision system is preserved using a minimum number of cuts.

Unsupervised multidimensional clustering algorithms would form a fourth category, but none such are currently implemented in ROSETTA.

A Boolean Reasoning Approach

Methods for discretization based on discernibility are intuitively appealing if used together with discernibility-based data mining methods. Nguyen and Skowron [133] propose such an algorithm.

Let \mathcal{A} denote a decision system. For the sake of simplifying the exposition, we will assume that all condition attributes A are numerical. For each attribute $a \in A$ we can sort its value set V_a to obtain the following ordering:

$$v_a^1 < \dots < v_a^i < \dots < v_a^{|V_a|} \quad (\text{D.2})$$

Let C_a denote the set of all naively generated cuts for attribute a , defined as shown below. The set C_a simply consists of all cuts midway between two observed attribute values, except for the cuts that are clearly not needed if we do not bother to discern between objects with the same decision values.

$$X_a^i = \{x \in U \mid a(x) = v_a^i\} \quad (\text{D.3})$$

$$\Delta_a^i = \{v \in V_a \mid \exists x \in X_a^i \text{ such that } d(x) = v\} \quad (\text{D.4})$$

$$C_a = \left\{ \frac{v_a^i + v_a^{i+1}}{2} \mid |\Delta_a^i| > 1 \text{ or } |\Delta_a^{i+1}| > 1 \text{ or } \Delta_a^i \neq \Delta_a^{i+1} \right\} \quad (\text{D.5})$$

If we employ all naively generated cuts, the original discernibility in \mathcal{A} with respect to the decision attribute is preserved. However, we can probably reduce the number of cuts drastically if we consider how they as an ensemble partition the condition space. To find such minimal subsets of cuts, we construct a Boolean POS function h as shown below, where each cut corresponds to a Boolean variable:

$$h = \prod_{(x,y)} \sum_a \left\{ \sum c^* \mid c \in C_a \text{ and } a(x) < c < a(y) \text{ and } \partial_{\mathcal{A}}(x) \neq \partial_{\mathcal{A}}(y) \right\} \quad (\text{D.6})$$

Each factor in h is a sum that stems from a pair of objects x and y that we want to discern between, and each if these are in turn composed of several sums of cuts from each attribute a . Only those cuts in C_a that separate x from y are considered in the sum.

The set of solutions to the problem of finding minimal subsets of cuts that preserve the original discernibility in \mathcal{A} with respect to the decision attribute, are defined through

the prime implicants of h . Since we presumably are interested in employing as few cuts as possible, the set covering heuristic of Johnson [95] is typically used to arrive at a single solution. Approximate solutions that introduce minor inconsistencies in \mathcal{A} can also be envisioned as a means of handling noisy data.

Note that a straightforward implementation of this algorithm will have a worst-case complexity of order $O(|A||U|^3)$, which may be prohibitively high for large decision systems. The dominating step of the outlined algorithm is the construction of h , and functions as the main bottleneck. However, as noted by Nguyen and Nguyen [130] and Nguyen [129], special techniques can be used to find a single prime implicant of h without having to actually construct h , resulting in an algorithm running in $O(|A||U|\log|U|)$ time.

In the literature, the outlined discernibility-based approach to discretization of a decision system \mathcal{A} is usually formulated in terms of computing reducts of a new and carefully constructed decision system \mathcal{A}' . As demonstrated in Section 5.2.4, such an exposition is equivalent to the present one.

Bibliography

- [1] Harold Abelson, Gerald Jay Sussman, and Julie Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge, MA, second edition, 1996.
- [2] Thomas Ågotnes. Extending ROSETTA: The embedding of a programming language. Project Report, Department of Computer and Information Science, Norwegian University of Science and Technology, 1998.
- [3] Thomas Ågotnes. Filtering large propositional rule sets while retaining classifier performance. MSc thesis, Norwegian University of Science and Technology, Department of Computer and Information Science, February 1999.
- [4] Thomas Ågotnes, Jan Komorowski, and Terje Løken. Taming large rule models in rough set approaches. In Żytkow and Rauch [255], pages 193–203.
- [5] Thomas Ågotnes, Jan Komorowski, and Aleksander Øhrn. Finding high performance subsets of induced rule sets: Extended summary. In Zimmermann and Lieven [253].
- [6] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1983.
- [7] Ethem Alpaydin. Combined 5x2CV F test for comparing supervised classification learning algorithms. Research Report 98-04, IDIAP, Martigny, Switzerland, May 1998. To appear in *Neural Computation*.
- [8] Douglas G. Altman. *Practical Statistics for Medical Research*. Chapman & Hall, London, UK, 1991.
- [9] Hal R. Arkes, Neal W. Dawson, Theodore Speroff, Frank E. Harrel, Jr., Carlos Alzola, Russell Phillips, Norman Desbiens, Robert K. Oye, William Knaus, and Alfred F. Connors, Jr. The covariance decomposition of the probability score and its use in evaluating prognostic estimates. *Medical Decision Making*, 15:120–131, 1995.
- [10] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36(1/2):105–139, July 1999.

- [11] Carol Bayley and Mark Waller. Does medical-ethical deontology exist? Let's look at the rules: An exercise in rough set data analysis using the Rosetta engine and various proposed rule-quality measures. *Journal of the Philosophical Society of America*, 1999. Submitted for publication.
- [12] Jan G. Bazan. A comparison of dynamic and non-dynamic rough set methods for extracting laws from decision tables. In Polkowski and Skowron [169], chapter 17, pages 321–365.
- [13] Jan G. Bazan, Andrzej Skowron, and Piotr Synak. Dynamic reducts as a tool for extracting laws from decision tables. In *Proc. International Symposium on Methodologies for Intelligent Systems*, volume 869 of *Lecture Notes in Artificial Intelligence*, pages 346–355. Springer-Verlag, 1994.
- [14] Eta S. Berner, George D. Webster, Alwyn S. Shugerman, James R. Jackson, James Algina, Alfred L. Baker, Eugene V. Ball, C. Glenn Cobbs, Vincent W. Dennis, Eugene P. Frenkel, Leonard D. Hudson, Elliott L. Mancall, Charles E. Rackley, and O. David Taunton. Performance of four computer-based diagnostic systems. *New England Journal of Medicine*, 330(25):1792–1796, June 1994.
- [15] C. Blake, E. Keogh, and C. J. Merz. UCI repository of machine learning databases. [<http://www.ics.uci.edu/~mlearn/MLRepository.html>], 1998. University of California, Irvine, Department of Information and Computer Sciences.
- [16] Daniel A. Bloch. Evaluating predictions of events with binary outcomes: An appraisal of the Brier score and some of its close relatives. Technical Report 135, Stanford University, Division of Biostatistics, Stanford University, CA, May 1990.
- [17] George Boole. *An Investigation of the Laws of Thought*. Dover Publications, New York, NY, 1854.
- [18] George E. P. Box, William G. Hunter, and J. Stuart Hunter. *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building*. Wiley, 1978.
- [19] Leo Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.
- [20] G. W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78:1–3, 1950.
- [21] Frank Markham Brown. *Boolean Reasoning: The Logic of Boolean Equations*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1990.
- [22] I. Bruha. Quality of decision rules: Definitions and classification schemes for multiple rules. In G. Nakhaeizadeh and C. C. Taylor, editors, *Machine Learning and Statistics: The Interface*, chapter 5, pages 107–131. John Wiley & Sons, 1997.
- [23] Stanley N. Burris. *Logic for Mathematics and Computer Science*. Prentice Hall, Upper Saddle River, NJ, 1998.

- [24] Mats Carlin, Tom Kavli, Bjørn Lillekjendlie, and Aleksander Øhrn. A comparison of four methods for nonlinear data modelling. In *Proc. Nordic Symposium on Neural Networks and Advanced Applications (NSANN'93)*, pages 26–40, Bergen, Norway, October 1993.
- [25] Ulf Carlin. Mining medical data with rough sets. MSc thesis, Norwegian University of Science and Technology, Department of Computer and Information Science, February 1998.
- [26] Ulf Carlin, Jan Komorowski, and Aleksander Øhrn. Rough set analysis of medical datasets in a case of patients with suspected acute appendicitis. In *Proc. ECAI'98 Workshop on Intelligent Data Analysis in Medicine and Pharmacology (IDAMAP'98)*, pages 18–28, Brighton, UK, 1998.
- [27] Ulf Carlin, Jan Komorowski, and Aleksander Øhrn. Rough set analysis of patients with suspected acute appendicitis. In *Proc. Seventh Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'98)*, pages 1528–1533, Paris, France, July 1998. EDK Éditions Médicales et Scientifiques.
- [28] Rich Caruana, Hooshang Kangarloo, John David N. Dionisio, Usha Sinha, and David Johnson. Case-based explanation of non-case-based learning methods. In Lorenzi [117], pages 212–215.
- [29] Pete Chapman, Julian Clinton, Thomas Khabaza, Thomas Reinartz, and Rüdiger Wirth. The CRISP-DM process model. [<http://www.crisp-dm.org>], March 1999.
- [30] Svetlana Cheshenchuk and Wojciech Ziarko. Mining patient data for predictive rules to determine maturity status of newborn children. *Bulletin of the International Rough Set Society*, 3(1/2):23–26, March 1999.
- [31] Christopher G. Chute, editor. *Proceedings AMIA 1998 Annual Symposium*, Orlando, FL, November 1998. Supplement to *Journal of the American Medical Informatics Association*, Hanley & Belfus, Inc.
- [32] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.
- [33] Paul D. Clayton, editor. *For the Record: Protecting Electronic Health Information*. National Research Council, National Academy Press, Washington DC, 1997.
- [34] The CodeGuru homepage. [<http://www.codeguru.com/>].
- [35] Enrico W. Coiera. Artificial intelligence in medicine: The challenges ahead. *Journal of the American Medical Informatics Association*, 3(6):363–366, November 1996.
- [36] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 1990.
- [37] The Data Mining Group homepage. [<http://www.dmg.org/>].

- [38] Thomas Dean, editor. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99)*, Stockholm, Sweden, July 1999. Morgan Kaufmann Publishers, Inc.
- [39] The Decision Systems Group homepage. [<http://dsg.harvard.edu/>]. Harvard Medical School.
- [40] Elizabeth R. DeLong, David M. DeLong, and Daniel L. Clarke-Pearson. Comparing the areas under two or more correlated receiver operating characteristic curves: A nonparametric approach. *Biometrics*, 44:837–845, September 1988.
- [41] Dorothy Elizabeth Robling Denning. *Cryptography and Data Security*. Addison-Wesley, 1982.
- [42] R. Detrano, A. Janosi, W. Steinbrunn, M. Pfisterer, J. Schmid, S. Sandhu, K. Guppy, S. Lee, and V. Froelicher. International application of a new probability algorithm for the diagnosis of coronary artery disease. *American Journal of Cardiology*, 64:304–310, 1989.
- [43] Thomas G. Dietterich. Statistical tests for comparing supervised classification learning algorithms. Technical report, Oregon State University, Department of Computer Science, October 1996.
- [44] Thomas G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, October 1998.
- [45] Pedro Domingos. MetaCost: A general method for making classifiers cost-sensitive. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining (KDD'99)*, San Diego, CA, 1999. ACM Press.
- [46] Pedro Domingos and Michael Pazzani. Beyond independence: Conditions for the optimality of the simple Bayesian classifier. In *Proc. Thirteenth International Conference on Machine Learning*, pages 105–112, Bari, Italy, 1996. Morgan Kaufmann.
- [47] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In A. Friedlitz and S. Russell, editors, *Proc. Twelfth International Conference on Machine Learning*, pages 194–202. Morgan Kaufmann, 1995.
- [48] Stephan Dreiseitl, Lucila Ohno-Machado, and Michael Binder. Comparing trichotomous tests by three-way ROC analysis. Submitted for publication, 1999.
- [49] Stephan Dreiseitl, Lucila Ohno-Machado, and Staal Vinterbo. Evaluating variable selection methods for diagnosis of myocardial infarction. In Lorenzi [117], pages 246–250.
- [50] Didier Dubois and Henri Prade. Putting rough sets and fuzzy sets together. In Słowiński [204], chapter II-1, pages 203–232.

- [51] Jamba Dunne and Mark Waller. What is a datum? Rough sets, maximum entropy, Bayesian propagation and conditional independence. University of California Press, 1999.
- [52] Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall, New York, NY, 1993.
- [53] U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996.
- [54] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. The KDD process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, November 1996.
- [55] Ad Feelders. Handling missing data in trees: Surrogate splits or statistical imputation? In Żytkow and Rauch [255], pages 329–334.
- [56] Jan Fibak, Zdzisław Pawlak, Krzysztof Słowiński, and Roman Słowiński. Rough sets based decision algorithm for treatment of duodenal ulcer by HSV. In *Bulletin of the Polish Academy of Sciences*, volume 34 of *Biological Sciences*, pages 227–249. Polish Academy of Sciences, 1986.
- [57] S. E. Fienberg. Confidentiality and disclosure limitation methodology: Challenges for national statistics and statistical research. Technical Report 668, Department of Statistics, Carnegie Mellon University, 1997.
- [58] J. L. Fleiss. *Statistical Methods for Rates and Proportions*. John Wiley & Sons, New York, second edition, 1981.
- [59] United States National Center for Health Statistics. International classification of diseases, 9th edition with clinical modifications, 1980.
- [60] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
- [61] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Addison-Wesley, Reading, MA, 1995.
- [62] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, NY, 1979.
- [63] Günther Gediga and Ivo Düntsch. Rough set dependency analysis in evaluation studies: An application in the study of repeated heart attacks. November 1994.
- [64] Marcel Geleijnse, Abdou Elhendy, Rob van Domburg, Jan Cornel, Ambroos Reijs, Jos Roelandt, Eric Krenning, and Paolo Fioretti. Prognostic value of dobutamine-atropine stress technetium-99m sestamibi perfusion scintigraphy in patients with chest pain. *Journal of the American College of Cardiologists*, 28(2):447–454, August 1996.

- [65] Roshbehari Ghosh and Mark Waller. Alternative techniques for the modeling of arsenical leachate fate and transport mechanisms on the Indian subcontinent. Technical Report 023-9853, WHO and the UN Environmental Programme, 1998.
- [66] The GkSetup homepage. [<http://www.gkware.com/gksetup/>].
- [67] G. A. Gorry. Computer-assisted clinical decision making. *Methods of Information in Medicine*, 12:45–51, 1973.
- [68] The GraphViz homepage. [<http://www.research.att.com/sw/tools/graphviz/>]. AT&T Research.
- [69] The Group of Logic homepage. [<http://alfa.mimuw.edu.pl/logic/>]. University of Warsaw, Poland.
- [70] Jerzy W. Grzymala-Busse and Linda K. Goodwin. Predicting preterm birth risk using machine learning from data with missing values. *Bulletin of the International Rough Set Society*, 1(1):17–21, 1997.
- [71] Stein Hallan, Arne Åsberg, and Tom-Harald Edna. Additional value of biochemical tests in suspected acute appendicitis. *European Journal of Surgery*, 163(7):533–538, July 1997.
- [72] Stein Hallan, Arne Åsberg, and Tom-Harald Edna. Estimating the probability of acute appendicitis using clinical criteria of a structured record sheet: The physician against the computer. *European Journal of Surgery*, 163(6):427–432, June 1997.
- [73] James Hampton. Product review: ROSETTA. *Journal of Computational Intelligence in Finance*, pages 45–46, January 1998.
- [74] David J. Hand. *Construction and Assessment of Classification Rules*. John Wiley & Sons, Chichester, UK, 1997.
- [75] James A. Hanley and Barbara J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143:29–36, April 1982.
- [76] James A. Hanley and Barbara J. McNeil. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148:839–843, September 1983.
- [77] Frank E. Harrel, Jr., Robert M. Califf, David B. Pryor, Kerry L. Lee, and Robert A. Rosati. Evaluating the yield of medical tests. *Journal of the American Medical Association*, 247(18):2543–2546, May 1982.
- [78] R. R. Hashemi, F. R. Jelovsek, and M. Razzaghi. Developmental toxicity risk assessment: A rough sets approach. *Methods of Information in Medicine*, 32(1):47–54, February 1993.
- [79] David Heckerman. A tutorial on learning with Bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, March 1995.

- [80] John Hertz, Anders Krogh, and Richard G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City, CA, 1991.
- [81] Jørgen Hilden. The area under the ROC curve and its competitors. *Medical Decision Making*, 11(2):95–101, April 1991.
- [82] Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, 11(1):63–91, April 1993.
- [83] D. W. Hosmer and S. Lemeshow. Goodness-of-fit tests for the multiple logistic regression model. *Comm. Statist.*, A9:1043–1069, 1980.
- [84] D. W. Hosmer, S. Lemeshow, and J. Klar. Goodness-of-fit testing for the logistic regression model when the estimated probabilities are small. *Biom. J.*, 33(8):911–924, 1988.
- [85] David W. Hosmer and Stanley Lemeshow. *Applied Logistic Regression*. John Wiley & Sons, 1989.
- [86] Xiaohua Hu, Nick Cercone, and Jiawei Han. An attribute-oriented rough set approach for knowledge discovery in databases. In Ziarko [251], pages 90–99. Proc. International Workshop on Rough Sets and Knowledge Discovery (RSKD'93), Banff, Alberta, Canada.
- [87] A. Hundepool and L. Willenborg. μ - and τ -ARGUS: Software for statistical disclosure control. In *Proc. Third International Seminar on Statistical Confidentiality*, Bled, Slovenia, 1996.
- [88] Torgeir Hvidsten, Tor-Kristian Jenssen, Astrid Lægreid, Aleksander Øhrn, Dyre Tjeldvoll, and Jan Komorowski. Boolean reasoning in the analysis of gene expression data. Data Mining for Bioinformatics: Towards In Silico Biology, November 1999. Cambridge, UK. Poster presentation.
- [89] The Inxight homepage. [<http://www.inxight.com/>]. Inxight Software, Inc.
- [90] Jacek Jelonek, Krzysztof Krawiec, Roman Słowiński, Jerzy Stefanowski, and Janusz Szymaś. Neural networks and rough sets: Comparison and combination for classification of histological pictures. In Ziarko [251], pages 426–433. Proc. International Workshop on Rough Sets and Knowledge Discovery (RSKD'93), Banff, Alberta, Canada.
- [91] Jacek Jelonek, Krzysztof Krawiec, Roman Słowiński, Jerzy Stefanowski, and Janusz Szymaś. Rough set reduction of features for picture-based reasoning. In Lin and Wildberger [115], pages 89–92.
- [92] Tor-Kristian Jenssen. Refinements to Mollestad's algorithm for synthesis of default rules. MSc thesis, Norwegian University of Science and Technology, Department of Computer and Information Science, March 1998.

- [93] Tor-Kristian Jenssen, Jan Komorowski, and Aleksander Øhrn. Some heuristics for default knowledge discovery. In Polkowski and Skowron [168], pages 373–380.
- [94] Tor-Kristian Jenssen, Jan Komorowski, and Aleksander Øhrn. Some heuristics for default knowledge discovery. In Sverre Storøy et al., editors, *Proc. Norsk Informatikkonferanse (NIK'98)*, pages 293–299, Kristiansand, Norway, November 1998. Tapir.
- [95] David S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.
- [96] Maciej Kandulski, Jacek Marciniak, and Konstanty Tukałło. Surgical wound infection: Conducive factors and their mutual dependencies. In Słowiński [204], chapter I-7, pages 95–110.
- [97] Jerome P. Kassirer. A report card on computer-assisted diagnosis – the grade: C. *New England Journal of Medicine*, 330(25):1824–1825, June 1994. Editorial.
- [98] Jack David Katzberg and Wojciech Ziarko. Variable precision extension of rough sets. *Fundamenta Informaticae*, 27(2, 3):155–168, August 1996.
- [99] R. L. Kennedy, A. M. Burton, H. S. Fraser, L. N. McStay, and R. F. Harrison. Early diagnosis of acute myocardial infarction using clinical and electrocardiographic data at presentation: Derivation and evaluation of logistic regression models. *European Heart Journal*, 17:1181–1191, August 1996.
- [100] Willy Klösgen and Jan Żytkow, editors. *Handbook of Data Mining and Knowledge Discovery*. Oxford University Press, 2000.
- [101] The Knowledge Systems Group homepage. [<http://www.idi.ntnu.no/IDT/grupper/KS-grp/>]. Norwegian University of Science and Technology.
- [102] W. W. Koczkodaj, M. Orłowski, and V. W. Marek. Myths about rough set theory. *Communications of the ACM*, 41(11):102–103, 1998.
- [103] Jan Komorowski, Terje Løken, Maurice White, and Mladen Jecmenica. Diagnosing rotating machinery using rough sets. In Mieczysław Kłopotek and Maciej Michalewicz, editors, *Proc. Eighth Workshop on Intelligent Information Systems (IIS'99)*, pages 137–145, Ustroń, Poland, June 1999.
- [104] Jan Komorowski and Aleksander Øhrn. Modelling prognostic power of cardiac tests using rough sets. *Artificial Intelligence in Medicine*, 15(2):167–191, January 1999.
- [105] Jan Komorowski, Aleksander Øhrn, and Andrzej Skowron. ROSETTA and other software systems for rough sets. In Klösgen and Żytkow [100].
- [106] Jan Komorowski, Zdzisław Pawlak, Lech Polkowski, and Andrzej Skowron. Rough sets: A tutorial. In Pal and Skowron [152], pages 3–98.

- [107] Jan Komorowski and Jan Żytkow, editors. *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'97)*, volume 1263 of *Lecture Notes in Artificial Intelligence*, Trondheim, Norway, June 1997. Springer-Verlag.
- [108] W. Kowalczyk. Rough data modelling: A new technique for analyzing data. In Polkowski and Skowron [169], chapter 20, pages 400–421.
- [109] Jerzy Krysiński. Analysis of structure-activity relationships of quaternary ammonium compounds. In Słowiński [204], chapter I-9, pages 119–136.
- [110] Jerzy Krysiński. Rough sets in the analysis of the structure-activity relationships of antifungal imidazolium compounds. *J Pharm Sci*, 84(2):243–248, February 1995.
- [111] Marzena Kryszkiewicz. Rules in incomplete information systems. In Wang [240], pages 73–76.
- [112] Marzena Kryszkiewicz. Properties of incomplete information systems in the framework of rough sets. In Polkowski and Skowron [169], chapter 21, pages 422–450.
- [113] Nada Lavrač. Selected techniques for data mining in medicine. *Artificial Intelligence in Medicine*, 16(2):3–23, June 1999.
- [114] Robert S. Ledley and Lee B. Lusted. Reasoning foundations of medical diagnosis. *Science*, 130(3366):9–21, July 1959.
- [115] T. Y. Lin and A. M. Wildberger, editors. *Soft Computing*. Society for Computer Simulation, San Diego, CA, 1995.
- [116] Terje Løken. Rough modeling: Extracting compact models from large databases. MSc thesis, Norwegian University of Science and Technology, Department of Computer and Information Science, March 1999.
- [117] Nancy M. Lorenzi, editor. *Proceedings AMIA 1999 Annual Symposium*, Washington, DC, November 1999. Supplement to *Journal of the American Medical Informatics Association*, Hanley & Belfus, Inc.
- [118] Pawel Lutomski and Mark Waller. Who has Kreitos? Does public policy reflect public judgment? An exercise in dispositional logic. In *Findings: The Marin County Survey on Attitudes toward Immigrant and Immigration Policy*. WWWC Press, 1999. The James Madison and Henry Cowell Foundations.
- [119] Daniel R. Masys, editor. *Proceedings 1997 AMIA Annual Fall Symposium (formerly SCAMC)*, Nashville, TN, October 1997. Supplement to *Journal of the American Medical Informatics Association*, Hanley & Belfus, Inc.
- [120] The MATLAB homepage. [<http://www.mathworks.com/products/matlab/>]. The MathWorks, Inc.

- [121] Charles E. Metz, Benjamin A. Herman, and Cheryl A. Roe. Statistical comparison of two ROC-curve estimates obtained from partially-paired datasets. *Medical Decision Making*, 18(1):110–121, 1998.
- [122] Scott Meyers. *Effective C++: 50 Specific Ways to Improve Your Programs and Designs*. Addison-Wesley Professional Computing Series. Addison-Wesley, Reading, MA, 1992.
- [123] Scott Meyers. *More Effective C++: 35 New Ways to Improve Your Programs and Designs*. Addison-Wesley Professional Computing Series. Addison-Wesley, Reading, MA, 1996.
- [124] R. S. Michalski. Pattern recognition as rule-guided inductive inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:349–361, 1980.
- [125] The MineSet homepage. [<http://www.sgi.com/software/mineset/>]. Silicon Graphics, Inc.
- [126] Torulf Mollestad. *A Rough Set Approach to Data Mining: Extracting a Logic of Default Rules from Data*. PhD thesis, Norwegian University of Science and Technology, Department of Computer and Information Science, February 1997. Report NTNU 1997:7.
- [127] Torulf Mollestad and Jan Komorowski. A rough set framework for mining propositional default rules. In Pal and Skowron [152], pages 233–262.
- [128] A. H. Murphy. A new vector partition of the probability score. *Journal of Applied Meteorology*, 12:595–600, 1973.
- [129] Hung Son Nguyen. Efficient SQL-querying method for data mining in large data bases. In Dean [38], pages 806–811.
- [130] Hung Son Nguyen and Sinh Hoa Nguyen. Some efficient algorithms for rough set methods. In *Proc. Fifth Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU'96)*, pages 1451–1456, Granada, Spain, July 1996.
- [131] Hung Son Nguyen and Sinh Hoa Nguyen. Discretization methods in data mining. In Polkowski and Skowron [169], chapter 22, pages 451–482.
- [132] Hung Son Nguyen, Sinh Hoa Nguyen, and Andrzej Skowron. Searching for features defined by hyperplanes. ICS Research Report 69/95, Warsaw University of Technology, Institute of Computer Science, 1995.
- [133] Hung Son Nguyen and Andrzej Skowron. Quantization of real-valued attributes. In *Proc. Second International Joint Conference on Information Sciences*, pages 34–37, Wrightsville Beach, NC, September 1995.
- [134] R. A. Nurdyke, C. A. Kulikowski, and C. W. Kulikowski. A comparison of methods for the automated diagnosis of thyroid dysfunction. *Computers and Biomedical Research*, 4:374–389, 1971.

- [135] The Objective Grid Lite homepage. [<http://www.roguewave.com/products/oglite/>]. Rogue Wave Software, Inc.
- [136] Lucila Ohno-Machado. *Medical Applications of Neural Networks: Connectionist Models of Survival*. PhD dissertation, Stanford University, Department of Computer Science, March 1996. Report STAN-CS-TR-96-1564.
- [137] Lucila Ohno-Machado, Hamish S. Fraser, and Aleksander Øhrn. Improving machine learning performance by removing redundant cases in medical data sets. In Chute [31], pages 523–527.
- [138] Lucila Ohno-Machado, Staal Vinterbo, Aleksander Øhrn, and Stephan Dreiseitl. Clinical data processing tools: A machine learning resource. In Lorenzi [117], page 1132.
- [139] Aleksander Øhrn. *ROSETTA Technical Reference Manual*. Knowledge Systems Group, Department of Computer and Information Science, NTNU, Trondheim, Norway, November 1999.
- [140] Aleksander Øhrn and Jan Komorowski. ROSETTA: A rough set toolkit for analysis of data. In Wang [240], pages 403–407.
- [141] Aleksander Øhrn and Jan Komorowski. Analyzing the prognostic power of cardiac tests using rough sets. In *Working Notes of the Invited Session on Intelligent Prognostic Methods in Medical Diagnosis and Treatment Planning, Second IMACS Multiconference on Computational Engineering in Systems Applications (CESA'98)*, pages 54–61, Nabeul-Hammamet, Tunisia, April 1998.
- [142] Aleksander Øhrn and Jan Komorowski. Diagnosing acute appendicitis with very simple classification rules. In Żytkow and Rauch [255], pages 462–467.
- [143] Aleksander Øhrn and Jan Komorowski. Diagnosing acute appendicitis with very simple classification rules. In Sverre Storøy et al., editors, *Proc. Norsk Informatikkonferanse (NIK'99)*, pages 105–116, Trondheim, Norway, November 1999. Tapir.
- [144] Aleksander Øhrn, Jan Komorowski, Andrzej Skowron, and Piotr Synak. A software system for rough data analysis. *Bulletin of the International Rough Set Society*, 1(2):58–59, 1997.
- [145] Aleksander Øhrn, Jan Komorowski, Andrzej Skowron, and Piotr Synak. The design and implementation of a knowledge discovery toolkit based on rough sets: The ROSETTA system. In Polkowski and Skowron [169], chapter 19, pages 376–399.
- [146] Aleksander Øhrn, Jan Komorowski, Andrzej Skowron, and Piotr Synak. The ROSETTA software system. *Bulletin of the International Rough Set Society*, 2(1):28–30, 1998.
- [147] Aleksander Øhrn, Jan Komorowski, Andrzej Skowron, and Piotr Synak. The ROSETTA software system. In Polkowski and Skowron [170], pages 572–576.

- [148] Aleksander Øhrn and Lucila Ohno-Machado. Using Boolean reasoning to anonymize databases. *Artificial Intelligence in Medicine*, 15(3):235–253, 1999.
- [149] Aleksander Øhrn, Lucila Ohno-Machado, and Todd Rowland. Building manageable rough set classifiers. In Chute [31], pages 543–547.
- [150] Aleksander Øhrn and Todd Rowland. Rough sets: A knowledge discovery technique for multifactorial medical outcomes. *American Journal of Physical Medicine & Rehabilitation*, 79(1), January 2000. 9 pages.
- [151] Aleksander Øhrn, Staal Vinterbo, Piotr Szymański, and Jan Komorowski. Modelling cardiac patient set residuals using rough sets. In Masys [119], pages 203–207.
- [152] Sankar K. Pal and Andrzej Skowron, editors. *Rough Fuzzy Hybridization: A New Trend in Decision-Making*. Springer, Singapore, 1999.
- [153] The Parallel Virtual Machine homepage. [<http://www.epm.ornl.gov/pvm/>]. Oak Ridge National Laboratory.
- [154] Zdzisław Pawlak. Rough sets. *International Journal of Information and Computer Science*, 11(5):341–356, 1982.
- [155] Zdzisław Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data*, volume 9 of *Series D: System Theory, Knowledge Engineering and Problem Solving*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1991.
- [156] Zdzisław Pawlak and Andrzej Skowron. Rough membership functions. In Yager et al. [246], pages 251–271.
- [157] Zdzisław Pawlak, Krzysztof Słowiński, and Roman Słowiński. Rough classification of patients after highly selective vagotomy for duodenal ulcer. *International Journal of Man-Machine Studies*, 24:413–433, 1986.
- [158] W. Pedrycz and J. F. Peters, editors. *Computational Intelligence in Software Engineering*, volume 16 of *Advances in Fuzzy Systems: Applications and Theory*. World Scientific, Singapore, December 1998.
- [159] J. F. Peters, L. Han, and S. Ramanna. Approximate time rough software cost decision system: Multicriteria decision-making approach. In Raś and Skowron [176], pages 556–564.
- [160] J. F. Peters and S. Ramanna. Application of the Choquet integral in a real-time software deployability control system. In *Proc. Fifth International Conference on Soft Computing and Information/Intelligent Systems (IIZUKA'98)*, pages 386–370, 1998.
- [161] J. F. Peters and S. Ramanna. Framework for approximate time rough control systems: An integrated fuzzy sets/rough sets approach. In *Proc. International Symposium on Artificial Intelligence in Real-Time Control (AIRTC'98)*, pages 1–8, Grand Canyon National Park, AZ, October 1998.

- [162] J. F. Peters and S. Ramanna. Time-constrained software cost model: Concepts and roughly fuzzy Petri net model. In Pedrycz and Peters [158], pages 339–370.
- [163] J. F. Peters and S. Ramanna. A rough sets approach to assessing software quality: Concepts and rough Petri net models. In Pal and Skowron [152], pages 349–380.
- [164] J. F. Peters, A. Skowron, Z. Suraj, S. Ramanna, and A. Paryzek. Modeling real-time decision-making systems with rough fuzzy Petri nets. In Zimmermann and Lieven [252], pages 985–989.
- [165] J. F. Peters and K. Ziaei. Generating rules in selecting controller gains: A combined rough sets/fuzzy sets approach. In *Proc. Canadian Conference on Electrical and Computer Engineering (CCECE'98)*, pages 233–236, Waterloo, Ontario, May 1998.
- [166] J. F. Peters, K. Ziaei, and S. Ramanna. Approximate time rough control: Concepts and application to satellite attitude control. In Polkowski and Skowron [168], pages 491–498.
- [167] G. Piatetsky-Shapiro and W. J. Frawley, editors. *Knowledge Discovery in Databases*. AAAI/MIT Press, 1991.
- [168] Lech Polkowski and Andrzej Skowron, editors. *Proceedings of the First International Conference on Rough Sets and Current Trends in Computing (RSCTC'98)*, volume 1424 of *Lecture Notes in Artificial Intelligence*, Warsaw, Poland, June 1998. Springer-Verlag.
- [169] Lech Polkowski and Andrzej Skowron, editors. *Rough Sets in Knowledge Discovery 1: Methodology and Applications*, volume 18 of *Studies in Fuzziness and Soft Computing*. Physica-Verlag, Heidelberg, Germany, 1998.
- [170] Lech Polkowski and Andrzej Skowron, editors. *Rough Sets in Knowledge Discovery 2: Applications, Case Studies and Software Systems*, volume 19 of *Studies in Fuzziness and Soft Computing*. Physica-Verlag, Heidelberg, Germany, 1998.
- [171] Paul Pritchard. A simple sub-quadratic algorithm for computing the subset partial order. *Information Processing Letters*, 56:337–341, 1995.
- [172] Paul Pritchard. A simple sub-quadratic algorithm for computing the subset partial order. Technical Report CIT-95-04, School of Computing and Information Technology, Griffith University, Queensland, Australia 4111, March 1995.
- [173] Foster Provost and Tow Fawcett. Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions. In *Proc. Third International Conference on Knowledge Discovery and Data Mining*, pages 43–48, Huntington Beach, CA, 1997. AAAI Press.
- [174] Foster Provost, Tow Fawcett, and Ron Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proc. Fifteenth International Conference on Machine Learning*, Madison, WI, 1998.

- [175] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.
- [176] Zbigniew W. Raś and Andrzej Skowron, editors. *Proceedings of the 11th International Symposium on Foundations of Intelligent Systems (ISMIS'99)*, volume 1609 of *Lecture Notes in Artificial Intelligence*, Warsaw, Poland, June 1999. Springer.
- [177] Eric Raymond. *The New Hacker's Dictionary*. MIT Press, Cambridge, MA, second edition, 1993.
- [178] Donald A. Redelmeier, Daniel A. Bloch, and David H. Hickam. Assessing predictive accuracy: How to compare Brier scores. *Journal of Clinical Epidemiology*, 44(11):1141–1146, 1991.
- [179] Brian D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.
- [180] The ROSETTA homepage. [<http://www.idi.ntnu.no/~aleks/rosetta/>]. Norwegian University of Science and Technology, Department of Computer and Information Science.
- [181] Todd Rowland, Lucila Ohno-Machado, and Aleksander Øhrn. Comparison of multiple prediction models for ambulation following spinal cord injury. In Chute [31], pages 528–532.
- [182] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
- [183] Steven L. Salzberg. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1(3):317–328, November 1997.
- [184] Helmut Schäfer. Efficient confidence bounds for ROC curves. *Statistics in Medicine*, 13:1551–1561, 1994.
- [185] Robert Schalkoff. *Pattern Recognition: Statistical, Structural and Neural Approaches*. John Wiley & Sons, 1992.
- [186] Robert E. Schapire. A brief introduction to boosting. In Dean [38], pages 1401–1406.
- [187] W. B. Schwartz. Medicine and the computer: The promise and problems of change. *New England Journal of Medicine*, 283:1257–64, 1970.
- [188] N. Shan and W. Ziarko. An incremental learning algorithm for constructing decision rules. In Ziarko [251], pages 335–346. Proc. International Workshop on Rough Sets and Knowledge Discovery (RSKD'93), Banff, Alberta, Canada.
- [189] E. H. Shortliffe, R. Davis, S. G. Axline, B. G. Buchanan, C. C. Green, and S. N. Cohen. Computer-based consultations in clinical therapeutics: Explanation and rule acquisition capabilities of the MYCIN system. *Computers and Biomedical Research*, 8(4):303–320, 1975.

- [190] Edward H. Shortliffe. Clinical decision-support systems. In *Medical Informatics: Computer Applications in Health Care* [191], chapter 15, pages 466–502.
- [191] Edward H. Shortliffe, Leslie E. Perreault, Gio Wiederhold, and Lawrence M. Fagan. *Medical Informatics: Computer Applications in Health Care*. Addison-Wesley, Reading, MA, 1990.
- [192] Andrzej Skowron. Synthesis of adaptive decision systems from experimental data. In Agnar Aamodt and Jan Komorowski, editors, *Proc. Fifth Scandinavian Conference on Artificial Intelligence*, number 28 in *Frontiers in Artificial Intelligence and Applications*, pages 220–238. IOS Press, May 1995.
- [193] Andrzej Skowron and Jerzy Grzymala-Busse. From rough set theory to evidence theory. In Yager et al. [246], chapter 10, pages 193–236.
- [194] Andrzej Skowron and Hung Son Nguyen. Boolean reasoning scheme with some applications in data mining. In Żytkow and Rauch [255], pages 107–115.
- [195] Andrzej Skowron and Lech Polkowski. Analytical morphology: Mathematical morphology of decision tables. *Fundamenta Informaticae*, 27(2, 3):255–271, August 1996.
- [196] Andrzej Skowron and Cecylia Rauszer. The discernibility matrices and functions in information systems. In Słowiński [204], chapter III-2, pages 331–362.
- [197] Krzysztof Słowiński. Rough classification of HSV patients. In Słowiński [204], chapter I-6, pages 77–93.
- [198] Krzysztof Słowiński, Jan Fibak, and Waldemar Jankowiak. Verification of conclusions from rough set analysis of highly selective vagotomy (HSV) during a follow-up program. Research Report 2/95, Warsaw University of Technology, Institute of Computer Science, 1995.
- [199] Krzysztof Słowiński and El Sanossy Sharif. Rough sets approach to analysis of data of diagnostic peritoneal lavage applied for multiple injuries patients. In Ziarko [251], pages 420–425. Proc. International Workshop on Rough Sets and Knowledge Discovery (RSKD'93), Banff, Alberta, Canada.
- [200] Krzysztof Słowiński and Roman Słowiński. Sensitivity analysis of rough classification. *International Journal of Man-Machine Studies*, 32:693–705, 1990.
- [201] Krzysztof Słowiński and Roman Słowiński. Sensitivity of rough classification to changes in norms of attributes. In Słowiński [204], chapter III-3, pages 363–372.
- [202] Krzysztof Słowiński, Roman Słowiński, and Jerzy Stefanowski. Rough sets approach to analysis of data from peritoneal lavage in acute pancreatitis. *International Journal of Medical Informatics*, 13(3):143–159, 1988.
- [203] Krzysztof Słowiński, Jerzy Stefanowski, Andrzej Antczak, and Zbigniew Kwias. Rough sets approach to the verification of indications for treatment of urinary stones by extracorporeal shock wave lithotripsy (ESWL). In Lin and Wildberger [115], pages 93–96.

- [204] Roman Słowiński, editor. *Intelligent Decision Support: Handbook of Applications and Advances in Rough Sets Theory*, volume 11 of *Series D: System Theory, Knowledge Engineering and Problem Solving*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992.
- [205] Roman Słowiński. Rough set learning of preferential attitude in multi-criteria decision making. In Jan Komorowski and Zbigniew Raś, editors, *Proc. International Symposium on Methodologies for Intelligent Systems (ISMIS'93)*, volume 689 of *Lecture Notes in Artificial Intelligence*, pages 642–651. Springer-Verlag, 1993.
- [206] Roman Słowiński and Jerzy Stefanowski. Rough classification in incomplete information systems. *Mathl. Comput. Modelling*, 12(10–11):1347–1357, 1989.
- [207] Roman Słowiński and Daniel Vanderpooten. Similarity relation as a basis for rough approximations. ICS Research Report 53/95, Warsaw University of Technology, Institute of Computer Science, 1995.
- [208] Jerzy Stefanowski, Krzysztof Słowiński, Andrzej Antczak, and Zbigniew Kwias. Verification of conclusions from the rough set analysis of the experience with extracorporeal shock wave lithotripsy (ESWL) for treatment of urinary stones. Research Report 45/95, Warsaw University of Technology, Institute of Computer Science, 1995.
- [209] Jaroslaw Stepaniuk. Rough set based data mining in diabetes mellitus data table. In Zimmermann and Lieven [252], pages 980–984.
- [210] Jaroslaw Stepaniuk. Rough set data mining of diabetes data. In Raś and Skowron [176], pages 457–464.
- [211] The STLport homepage. [<http://www.stlport.org/>].
- [212] M. H. Stone. The theory of representations for Boolean algebras. *Trans. Amer. Math. Soc.*, 40:37–111, 1936.
- [213] Latanya Sweeney. Guaranteeing anonymity when sharing medical data: The Datafly system. Working Paper AIWP-WP344, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1997.
- [214] Latanya Sweeney. Guaranteeing anonymity when sharing medical data: The Datafly system. In Masys [119], pages 51–55.
- [215] Latanya Sweeney. Weaving technology and policy together to maintain confidentiality. *Journal of Law, Medicine & Ethics*, 25:98–110, 1997.
- [216] J. A. Swets and R. M. Pickett. *Evaluation of Diagnostic Systems: Methods from Signal Detection Theory*. Academic Press, New York, NY, 1982.
- [217] Piotr Synak. *Rough Set Expert System User's Guide*. Institute of Mathematics, Warsaw University, Poland, 1995. Version 1.0.

- [218] Peter Szolovits, Ramesh S. Patil, and William B. Schwartz. Artificial intelligence in medical diagnosis. *Annals of Internal Medicine*, 108(1):80–87, January 1988.
- [219] Peter Szolovits and Stephen G. Pauker. Categorical and probabilistic reasoning in medical diagnosis. *Artificial Intelligence*, 11:115–144, 1978.
- [220] D. Tsaptsinos and M. G. Bell. Medical knowledge mining using rough set theory. In *Proc. International Conference on Neural Networks and Expert Systems in Medicine and Healthcare*, Plymouth, UK, August 1994.
- [221] Christine L. Tsien and Hamish S. F. Fraser. Optimizing diagnosis of myocardial infarction in the emergency room: A new flowchart decision aid. Presented at the American College of Cardiology Dearborn Summit, September 1997.
- [222] Shusaku Tsumoto. Automated induction of medical expert system rules from clinical databases based on rough set theory. *Information Sciences*, 112:67–84, 1998.
- [223] Shusaku Tsumoto. Automated knowledge acquisition from clinical databases based on rough sets and attribute-oriented generalization. In Chute [31], pages 548–552.
- [224] Shusaku Tsumoto. Extraction of experts' decision rules from clinical databases based on rough set theory. *Journal of Intelligent Data Analysis*, 2(3), 1998.
- [225] Shusaku Tsumoto. Modelling medical diagnostic rules based on rough sets. In Polkowski and Skowron [168], pages 475–482.
- [226] Shusaku Tsumoto. Representation of medical knowledge based on rough sets. In Zimmermann and Lieven [252], pages 970–974.
- [227] Shusaku Tsumoto and Hiroshi Tanaka. Incremental learning of probabilistic rules from clinical databases based on rough set theory. In Masys [119], pages 198–202.
- [228] J. H. van Bommel and M. A. Musen, editors. *Handbook of Medical Informatics*. Springer, 1997.
- [229] Staal Vinterbo. *Predictive Models in Medicine: Some Methods for Construction and Adaptation*. PhD thesis, Norwegian University of Science and Technology, Department of Computer and Information Science, December 1999. Report NTNU 1999:130.
- [230] Staal Vinterbo and Lucila Ohno-Machado. A recalibration method for predictive models with dichotomous outcomes. In Vinterbo [229]. Submitted for publication.
- [231] Staal Vinterbo, Lucila Ohno-Machado, and Hamish Fraser. A description of a strategy for building rough set classifiers using performance filtering of reducts. In Zimmermann and Lieven [252], pages 975–979.

- [232] Staal Vinterbo and Aleksander Øhrn. A rough set approach to clustering. In Wang [240], pages 383–386.
- [233] Staal Vinterbo and Aleksander Øhrn. Approximate minimal hitting sets and rule templates. In Vinterbo [229]. Submitted for publication.
- [234] The Visual C++ homepage. [<http://msdn.microsoft.com/visualc/>]. Microsoft Corporation.
- [235] The Visual SourceSafe homepage. [<http://msdn.microsoft.com/ssafe/>]. Microsoft Corporation.
- [236] Alicja Wakulicz-Deja, Mariusz Boryczka, and Piotr Paszek. Discretization of continuous attributes on decision system in mitochondrial encephalomyopathies. In Polkowski and Skowron [168], pages 483–490.
- [237] Alicja Wakulicz-Deja and Piotr Paszek. Diagnose progressive encephalopathy applying the rough set theory. *International Journal of Medical Informatics*, 46(2):119–127, September 1997.
- [238] Mark Waller. Connectionist tensor-product variable-binding and rough set epistemology: They logically coterminate to resolve the Wittgenstein-Turing debates over the principle of exclusion. *Journal of the History and Philosophy of Science*, 1999. Submitted for publication.
- [239] Ronald E. Walpole and Raymond H. Myers. *Probability and Statistics for Engineers and Scientists*. Prentice-Hall, London, UK, fifth edition, 1993.
- [240] Paul P. Wang, editor. *Proc. Third International Joint Conference on Information Sciences*, volume 3, Durham, NC, March 1997.
- [241] Barbara E. Wojcik and Wojciech Ziarko. Rough sets approach to analysis of databases of women with breast cancer treated in the U.S. military facilities. In *Proc. Symposium on Modelling, Analysis and Simulation, IMACS Multiconference on Computational Engineering in Systems Applications (CESA'96)*, volume 2, pages 748–752, Lille, France, July 1996.
- [242] Erling A. Woods. *The Hybrid Phenomena Theory*. PhD thesis, Norwegian Institute of Technology, Department of Technical Cybernetics, June 1993. Report ITK 1993:72-W.
- [243] Beverly Woodward. The computer-based patient record and confidentiality. *New England Journal of Medicine*, 333(21):1419–1422, November 1995.
- [244] L. K. Woolery and J. Grzymala-Busse. Machine learning for an expert system to predict preterm birth risk. *Journal of the American Medical Informatics Association*, 1(6):439–446, 1994.
- [245] Jakub Wroblewski. Finding minimal reducts using genetic algorithms (extended version). In *Proc. Second International Joint Conference on Information Sciences*, pages 186–189, September 1995.

- [246] R. Yager, M. Fedrizzi, and J. Kacprzyk, editors. *Advances in the Dempster-Shafer Theory of Evidence*. John Wiley & Sons, 1994.
- [247] Y. Y. Yao and T. Y. Lin. Generalization of rough sets using modal logics. *Intelligent Automation and Soft Computing*, 2(2):103–120, 1996.
- [248] Q. Zhang, Z. Han, and F. Wen. A new approach for fault diagnosis in power systems based on rough set theory. In *Proc. Fourth International Conference on Advances in Power System Control, Operation and Management (APSCOM-97)*, volume 2, pages 598–602, Hong Kong, China, November 1997.
- [249] Wojciech Ziarko. Analysis of uncertain information in the framework of variable precision rough sets. *Foundations of Computing and Decision Sciences*, 18(3–4):381–396, 1993.
- [250] Wojciech Ziarko. Variable precision rough set model. *Journal of Computer and System Sciences*, 46:39–59, 1993.
- [251] Wojciech P. Ziarko, editor. *Rough Sets, Fuzzy Sets and Knowledge Discovery*, Workshops in Computing. Springer-Verlag, 1994. Proc. International Workshop on Rough Sets and Knowledge Discovery (RSKD'93), Banff, Alberta, Canada.
- [252] Hans-Jürgen Zimmermann and Karl Lieven, editors. *Proc. Sixth European Congress on Intelligent Techniques and Soft Computing (EUFIT'98)*, Aachen, Germany, September 1998.
- [253] Hans-Jürgen Zimmermann and Karl Lieven, editors. *Proc. Seventh European Congress on Intelligent Techniques and Soft Computing (EUFIT'99)*, Aachen, Germany, September 1999.
- [254] David Zitner, Grace I. Paterson, and Donald F. Fay. Methods in health decision support systems: Methods for identifying pertinent and superfluous activity. In Joseph K. H. Tan and Samuel Sheps, editors, *Health Decision Support Systems*, chapter 8. Aspen Publishers, 1998.
- [255] Jan M. Żytkow and Jan Rauch, editors. *Proceedings of the Third European Symposium on Principles and Practice of Knowledge Discovery in Databases (PKDD'99)*, volume 1704 of *Lecture Notes in Artificial Intelligence*, Prague, Czech Republic, September 1999. Springer-Verlag.