DOCTORAATSPROEFSCHRIFT

2011 | Faculteit Wetenschappen

Class association rule mining using multidimensional numbered information spaces

Proefschrift voorgelegd tot het behalen van de graad van
Doctor in de Wetenschappen, te verdedigen door:

Iliya MITOV

Promotor: prof. dr. Koen VANHOOF (UHasselt)
Copromotor: prof. dr. Krassimir MARKOV
(Institute of Mathematics and Informatics, Bulgarije)

universiteit hasselt

DOCTORAATSPROEFSCHRIFT

2011 | Faculteit Wetenschappen

Class association rule mining using multidimensional numbered information spaces

Proefschrift voorgelegd tot het behalen van de graad van
Doctor in de Wetenschappen, te verdedigen door:

Iliya MITOV

Promotor: prof. dr. Koen VANHOOF (UHasselt)
Copromotor: prof. dr. Krassimir MARKOV
(Institute of Mathematics and Informatics, Bulgarije)

D/2011/2451/50

universiteit hasselt

Hasselt University

Faculty of Science

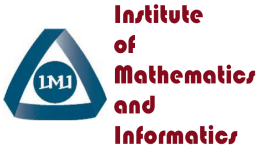
November 2011

Class Association Rule Mining Using Multi-Dimensional Numbered Information Spaces

A thesis submitted for the degree of Doctor of Science by:

Iliya Mitov

Promoter: Prof. Dr. Koen Vanhoof
Co-promoter: Assoc. Prof. Dr. Krassimir Markov



Abstract

Data mining is of great importance in the overall process of knowledge discovery. In this dissertation we focused our attention in the part of discovery-oriented methods and especially classification algorithms.

Class-Association Rules (CAR) algorithms have a special place within the family of classification algorithms. This type of classifiers offers a number of advantages: efficiency of the training regardless of the training set; easy handling with high dimensionality; very fast classification; high accuracy; classification model easily comprehensible for humans. The main classification workflow of CAR algorithms usually involves three phases: generating the rules, pruning, and recognition.

The mining of association rules is a typical data mining task that works in an unsupervised manner. A major advantage of association rules is that they are theoretically capable to reveal all interesting relationships in a database. But for practical applications the number of mined rules is usually too large to be exploited entirely. Hence, a pruning phase is applied in order to build accurate and compact classifiers. The pruning can be applied during preprocessing, simultaneously to the association rules mining, or during post-processing. Different rule quality measures and rule ordering schemes can be applied in the process of rule selection. There are also different options which can be considered for the recognition phase – e.g. to use a simple rule or to use a set of rules with different types of ordering schemas.

On the other hand, the process of creating classification models inevitably touches upon the use of appropriate access methods which facilitate access to different kinds of structures used in such algorithms.

Our effort had been focused on the memory organization called *Multi-dimensional numbered information spaces* which allows to operate with context-free multidimensional data structures. The program realization of such structures is named ArM 32. Multi-Domain Information Model (MDIM) and respectively Arm 32 are based on the process of replacing names by numbers which allows to use mathematical functions and addressing vectors for accessing the information.

Our approach is to use such structures and operations in the implementation of one class association rule classifier in order to provide evidence on the vitality

of the idea of using context-free multidimensional data structures and direct access as a powerful tool for knowledge discovery. We have proposed two classification algorithms – Pyramidal Growing Networks (PGN) and Multi-layer Pyramidal Growing Networks (MPGN).

PGN creates association rules, optimized for maximal accuracy of produced rules. One of the main characteristics of PGN is that it is a parameter-free classifier. The association rule mining is executed from the longest rules to the shorter ones until no intersections between patterns in the classes are possible. In the pruning phase the contradictions and inconsistencies of more general rules are cleared, after that the pattern set is compacted excluding all more concrete rules within the classes.

PGN is introduced as a useful tool for questioning the support-first principle used by many associative classifiers when mining for association rules. PGN reverses the common approach and focuses primarily on the confidence of the association rules and only in a later stage on the support of the rules. The main purpose is twofold: to provide a proof of concept for this new approach and to gather evidence on its potential.

MPGN is based on multilayer structure. It involves possibility to escape combinatorial explosion using smart disposing of the information in the multilayer structures called "pyramids". These structures can be easily implemented using ArM-structures.

These algorithms are implemented in the data mining environment PaGaNe, developed by the team from the Institute of Mathematics and Informatics – Bulgarian Academy of Sciences; Iliya Mitov and Krassimira Ivanova are the principal developers. PaGaNe incorporates different types of statistical analysis methods, discretization algorithms, association rule miner, as well as classification algorithms, which all are based on the use of multi-dimensional numbered information spaces.

The Lenses dataset is used as a test example to illustrate the specifics of the proposed algorithms, the process of creating classification models as well as the process of recognition. We demonstrate that PGN produces the pattern set that is both minimal and complete for covering the learning set, which is an indicator for expectation that PGN will produce tight model and good accuracy results. In the case of MPGN we have demonstrated the process of creating main construction elements. We also have illustrated the functionality which allows to visualize how the pyramids are being created and how the queries are being recognized.

We carried out experiments with 25 datasets from the UCI machine learning repository [Frank and Asuncion, 2010]. The experiments had been conducted using the data mining environment PaGaNe, the knowledge analysis system Weka, and LUCS-KDD Repository. A comparison between PGN, MPGN and some other CAR algorithms, as well as decision tree and decision rule classifiers which have similar behavior of creating the task model, had been done.

One series of experiments aimed to study what accuracy had been obtained while preprocessing real data with different discretizators realized in PaGaNe. We found that in general PGN-classifier trained on data preprocessed by Chi-merge with 95% significance level achieves lower classification error than those trained on data preprocessed by the other discretization methods. The main reason for this is that using Chi-square statistical measure as criterion for class dependency in adjacent intervals of a feature results in good separation between class labels.

A second set of experiments studied the process of growing the learning sets and how this reflects on the classification model and the accuracy of PGN and MPGN; more specifically, we studied the critical point of the amount of the learning set in which classification model is relatively compact and the received accuracy stabilizes. Of course this critical point highly depends on the choice of dataset.

A third set of experiments were focused on analyzing different exit points of MPGN. The received results showed that in a lot of cases the build constructs lead to excluding only one class as best competitor. Other cases usually fall into competition between classes, where different strategies for ordering the competitors can be applied. A very few cases fall into the way where MPGN-algorithm did not work and alternative choice is given.

A fourth set of experiments aimed to analyze the dependencies of classifiers' behaviors when the noise rush in the dataset attributes; for this set we used the Monks1 dataset. The experiments demonstrated that noising in the dataset worsens considerably the accuracy of PGN which had been designed to perform well in clear datasets. However, experiments with other existing classifiers showed that they also were not been able to resist noising attacks.

We made the comparison of overall accuracy between PGN, MPGN (with two recognition strategies – S1 and S2), CMAR, OneR, JRip, J48 and REPTree. The Friedman test showed statistical difference between tested classifiers. The post-hoc Nemenyi test showed that our PGN has best overall performance between examined classifiers and MPGN is competitive with CMAR, J48, JRip and REPTree.

The experimental results are very positive and show that PGN is competitive with classification methods that build similar classification behavior. At the same time, it has an essential advantage over the other classifiers being parameter free. Furthermore, the empirical results showed that PGN is slightly more sensitive to noise than techniques such as C4.5 and RIPPER. However, its overall accuracy was still very good compared to these classifiers. In general, the results provide evidence that the confidence-first approach yields interesting opportunities for knowledge discovery.

Acknowledgements:

This work was supported by Hasselt University under the Project R-1876 "Intelligent systems' memory structuring using multidimensional numbered information spaces" and by the Bulgarian National Science Fund under the Project D002-308 "Automated Metadata Generating for e-Documents Specifications and Standards".

I would like to express my gratitude to Hasselt University, Belgium and Institute of Mathematics and Informatics, Bulgaria for ensuring the right conditions for establishing this work.

I am grateful to my advisor Prof. Koen Vanhoof from Hasselt University, Belgium, for his guidance throughout my doctoral studies and all the time and effort he put in the development of me and my work.

I am also indebted to my co-promoter Assoc. Prof. Krassimir Markov from Institute of Mathematics and Informatics, Bulgaria. We have been working together for over twenty years on implementing the paradigm of multi-dimensional numbered information spaces in different practical domains.

I would send my special thanks to Benoit Depaire, who helped me in studying, establishing and implementing of data mining techniques.

This thesis benefitted further from the stimulating and enjoyable discussions with Prof. Levon Aslanyan and Prof. Hasmik Sahakyan from the Institute for Informatics and Automation Problems of NAS, Armenia and Prof. Victor Gladun and Prof. Vitalii Velychko from the Glushkov Institute of Cybernetics, Ukraine who encouraged and supported my research. Our collaboration in the field of attribute subset selection and pyramidal structures helped to contextualize this thesis better.

I also would like to thank Emilia Todorova from Glasgow Caledonian University for her assistance with language revision.

Finally, I am very much in debt for the unconditional support, endless patience and constant encouragement I have received from my companion in life, Valeria.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ACRI	Associative Classifier with Reoccurring Items
ADTree	Alternating Decision Tree
AIS	ARM algorithm abbreviated from the families of the creators: Agrawal, Imielinski, Swami
AM	Access Method
AODE	Averaged One-Dependence Estimators
ARC-AC	Association Rule-based Categorizer for All Categories
ARC-BC	Association Rule-based Categorizer by Category
ArM 32	FOI Archive Manager 32
ARUBAS	Association Rule Based Similarity
BIE	Basic Information Element
C4.5	Decision tree classifier, successor of ID3
CACA	Class-based Associative Classification Approach
CAR	Class Association Rules
CBA	Classification Based on Associations
CBR	Case-based reasoning
CHAID	CHi-squared Automatic Interaction Detector
CMAR	Classification Based on Multiple CARs
CODASYL	Conference on Data Systems Languages
CorClass	Correlated Association Rule Mining
CPAR	Classification based on Predictive Association Rules
CRAN	The Comprehensive R Archive Network
DBMS	Database Management System
ELKI	Environment for DeveLoping KDD-Applications Supported by Index-Structures
FDM	Fast Distributed Mining of association rules
FOIL	First Order Inductive Learner
FP-Tree	Frequent Pattern Tree
FWI	Fire Weather Index
GMES	Global Monitoring for Environment and Security
GNNAT	Geometric Near-Neighbor Access Tree
GNU	Recursive acronym "GNU's Not Unix"
GPS	Global Positioning System
GUI	Graphical User Interface

HNB	Hidden Naïve Bayes
IB1	Instance Based algorithm, used 1-nearest neighbors classification
IBk	Instance Based algorithm, used k-nearest neighbors classification
ID3	Iterative Dichotomiser
J48	Weka implementation of Quinlan's C4.5 algorithm
jHepWork	J – from Jython; Hep – from High-energy physics (HEP) examples
JRip	Weka implementation of RIPPER
Jython	an implementation of the Python programming language written in Java
KDD	Knowledge Discovery in Databases
KNIME	Konstanz Information Miner
k-NN	k-Nearest Neighbors
LADTree	multi-class Alternating Decision tree using the LogitBoost strategy
LibSVM	Library for Support Vector Machines
LUCS-KDD	Liverpool University of Computer Science – Knowledge Discovery in Data
MBR	Minimum Bounding Rectangle
MCAR	Multi-class Classification based on Association Rule
MDIM	Multi-Domain Information Model
MDL	Minimum Description Length
MPGN	Multi-Layer PGN
PGN	Pyramidal Growing Networks
PMML	Predictive Model Markup Language
PRM	Predictive Rule Mining
R language	Programming language and statistical software environment
Rattle	R Analytical Tool To Learn Easily
RIPPER	Repeated Incremental Pruning to Produce Error Reduction
S language	Statistical programming language
SIGKDD	Special Interest Group on Knowledge Discovery in Databases
SMO	Sequential Minimal Optimization
SQL	Structured Query Language
SVM	Support Vector Machines
TFP	Total From Partial
TFPC	Total From Partial Classification
UCI	University of Carolina Irvine
Weka	Waikato Environment for Knowledge Analysis
XML	eXtensible Markup Language
YALE	Yet Another Learning Environment

1 Introduction

1.1 Class Association Rules

Over the past few centuries, the quantity of accumulated information in analogue and now in digital form is constantly growing. Because of the rapid development in all areas of human activity in modern society, the production, economic and social processes have become more complex. Most organizations using information technology resources collect and store large amounts of data. The challenge that all those organizations face today is, not how to collect and store the data needed, but how to derive meaningful conclusions from this massive volume of information. The solution is in the technology of data mining and, in particular, in the use of association rules.

The main objective of association rules mining is to discover regularities in the incoming data. Arising from the field of market basket analysis to generate interesting rules from large collections of data [Agrawal et al, 1993], the association rule mining prove to be a feasible approach to model relationships between class labels and features from a training set [Bayardo, 1998]. Since then, many associative classifiers were proposed, mainly differing in the strategies used to select rules for classification and in the heuristics used for pruning rules.

Associative classification offers a new alternative to classification schemes by producing rules based on conjunctions of attribute-value pairs that occur frequently in datasets. Frequent patterns and their corresponding associations or correlation rules characterize interesting relationships between attribute conditions and class labels, and thus have been recently used for more effective classification. The main purpose is that we can search for strong associations between frequent patterns (conjunctions of attribute-value pairs) and class labels. The association rules explore highly confident associations among multiple attributes.

Association rules are mined in a two-step process consisting of frequent item-set mining, followed by rule generation. The first step searches for patterns of attribute-value pairs that occur repeatedly in a dataset, where each attribute-

value pair is considered an item. The resulting attribute value pairs form frequent item-sets. The second step analyzes the frequent item-sets in order to develop association rules using certain criteria for measuring the significance of the rule.

1.2 Multi-Dimensional Numbered Information Spaces as Memory Structures for Intelligent Data Processing

An overview of available algorithms and used information structures shows the variety of decisions in association rule mining. As we can see, graph structures, hash tables, different kind of trees, bit matrices, arrays, etc., are used for storing and retrieving the information. Each kind of data structure brings some benefits but also has disadvantages. [Liu et al, 2003] discuss such questions and provides a comparison between tree structures and arrays demonstrating that tree-based structures are capable of reducing traversal cost because duplicated transactions can be merged and different transactions can share the storage of their prefixes. However, they incur high construction cost especially when the dataset is sparse and large. On the other hand, array-based structures demand little construction cost but they need much more traversal cost because the traversal cost of different transactions cannot be shared.

Hence, the memory organization we decided to use in this research was based on numbering as a main approach. Replacing the names by numbers permits to use mathematical functions and address vectors for accessing the information instead of search engines.

In addition, numbering has the advantage of using the same addressing mechanism for the external memory as the one used for the main computer memory. Our approach allows one to build high dimensional information structures. Practically we can use a great number of dimensions as well as the number of elements on given dimension.

This type of memory organization is called "Multi-Dimensional Numbered Information Spaces". Its advantages have been demonstrated in multiple real-life implementations over twenty-five years [Markov, 1984], [Markov, 2004], [Markov, 2005]. In the same time, this kind of memory organization has not been implemented in the area of the Artificial Intelligence and especially for intelligent systems' memory structuring.

In summary, the advantages of numbered information spaces are:

- the possibility to build growing space hierarchies of information elements;
- the great power for building interconnections between information elements stored in the information base.

The main idea is to replace the (symbol or real; point or interval) values of the objects' attributes with integer numbers of the elements of corresponding

ordered sets. Thus each object is described by a vector of integer values, which may be used as co-ordinate address in the multi-dimensional information space.

1.3 Objectives of the Dissertation

The goals of this thesis are two-fold:

- to introduce a parameter-free class association rule algorithm, which focuses primarily on the confidence of the association rules and only in a later stage on the support of the rules. We expect that this approach will ensure implementing high-quality recognition especially within unbalanced and multi-class datasets. The nature of such a classifier is more oriented to having characteristic rules;
- to show the advantages of using multidimensional numbered information spaces for developing memory structuring in data mining processes on the example of implementation of the proposed class association rule algorithms.

To achieve these goals we develop a pyramidal multi-dimensional model for memory organization in classification systems. Further, we will implement the corresponding experimental classification system, and finally, we conduct experiments and evaluation of the results in order to test the hypothesis we have made.

1.4 Outline

The dissertation is structured in nine chapters and an Appendix as follows:

1. Introduction.
2. Data Mining and Knowledge Discovery.
3. CAR Algorithms.
4. Multi-Dimensional Numbered Information Spaces.
5. PGN and MPGN Algorithms.
6. Program Realization.
7. Example of Lenses Dataset.
8. Sensitivity Analysis.
9. Conclusions and Future Work.

A brief overview of the content is given below.

Chapter 2 introduces data mining and its importance in a global process of knowledge discovery. A taxonomy of data mining methods is provided with special focus on classification methods. A brief overview of main types of classifiers and ensemble methods is made followed by a succinct description of

the process of discretization which is an important part of the process of preparing data in the global frame of knowledge discovery. Furthermore, this chapter presents existing open source data mining software. Finally we discuss the growing importance of standardization and interoperability within the software development of data mining algorithms and environments, and the possibilities of built-in additional online analytical processing systems, decision-support systems, etc.

Chapter 3 provides an overview of the field of CAR-classifiers. Here we have presented all the steps, which are typical in the classification process of CAR algorithms: generating the rules, pruning and recognizing. Several techniques are suggested for the phase of generating the rules. Pruning, an important step in the learning process of CAR algorithms, is applied as a preprocessing step, in parallel with the association rule mining or after it. Further, we present several rule quality measures and rule ordering schemes, used in CAR algorithms. During the recognition phase we also need to make a final decision using simple rule or set of rules with different types of ordering schemas. Finally, using a proposed frame, typical for CAR algorithms, we analyze twelve representatives of CAR algorithms, showing a wide variety of proposed techniques.

Chapter 4 is focused on the different kinds of existing methods for data management in the field of data mining and knowledge discovery. In this chapter we present a particular type of memory organization, called "Multi-Dimensional Numbered Information Spaces", as well as its program realization ArM 32. Their main structures and functions are described.

Chapter 5 contains a description of the proposed classification algorithms – PGN and MPGN.

Chapter 6 considers the implementation of proposed algorithms. It provides a short description of an experimental data mining environment – PaGaNe, which is the result of collaborative work of researchers from Bulgaria, Belgium, Armenia and Ukraine.

Chapter 7 reveals the specific steps which pass PGN and MPGN algorithms on the example of the Lenses dataset.

Chapter 8 is focused on presenting several experiments made with the use of already developed tools. Special attention is paid to the sensitivity analysis of the results. Comparison between PGN, MPGN and some decision tree and decision rule classifiers, which have similar behavior of creating the task model, is made.

Finally, **chapter 9** provides conclusions and an overview of directions for future research.

In the Appendix additional experiments, showing the comparison of PGN and MPGN with a wide range of different kinds of classifiers, realized in Weka, are included.

The work contains 35 tables, 46 figures, and 106 references.

2 Data Mining and Knowledge Discovery

Abstract:

In this chapter we start with a brief overview of the field of data mining and its importance in the global process of knowledge discovery.

A taxonomy of data mining methods is shown with particular focus on the classification methods. Adding to that, we give a short explanation of the main types of classifiers.

The chapter also includes a brief overview of the process of discretization as an important preprocessing step for most of the classification algorithms.

Several existing open source data mining software systems are described.

Finally we have included a discussion about the increasing necessity of standardization and interoperability within the software implementation of data mining algorithms and environments, and the possibilities of additional built-in online analytical processing systems, decision-support systems, etc.

2.1 Knowledge Discovery

Data Mining is a part of the overall process of Knowledge Discovery in databases (KDD) [Fayyad et al, 1996]. While Knowledge Discovery is defined as the process of seeking new knowledge about an application domain [Klosgen and Zytkow, 1996], data mining is concerned with the application (by humans) of algorithms designed to analyze data or to extract pattern in specific categories of data. The knowledge discovery process consists of many steps, with data mining being one of them.

The Knowledge Discovery in Databases (KDD) process had been defined by many authors. For instance [Fayyad et al, 1996] define it as "the nontrivial process of identifying valid, novel, potentially useful, and ultimately

understandable patterns in data". [Friedman, 1997] considers the KDD process as an automatic exploratory data analysis of large databases.

The KDD process has been formed by different stages, which iteratively interact with each other. During the years, several models have been proposed (for instance in [Fayyad et al, 1996]). Generally, the process of knowledge discovery can be divided into following stages [Han and Kamber, 2006]:

1. Data cleaning (the removal of noise and inconsistent data).
2. Data integration (combining multiple data sources).
3. Data selection (retrieval of data relevant to the analysis task from the database).
4. Data transformation (transformation or consolidation of data suited for mining; this can be done, for example by performing summary or aggregation operations).
5. Data mining (an essential process where intelligent methods are applied in order to extract data patterns).
6. Pattern evaluation (used to identify the most interesting patterns representing knowledge based on some interestingness measures).
7. Knowledge presentation (use of visualization and knowledge representation techniques to present the mined knowledge to the user).

Data mining, which is discussed further over the next part of the chapter, is an essential part in the global process of knowledge discovery.

2.2 Data Mining

Data Mining is the process of analyzing a large set of raw data in order to extract hidden information which can be predicted. It is a discipline, which is at the confluence of artificial intelligence, data bases, statistics, and machine learning. The questions related to data mining present several aspects, the main being: classification, clustering, association and regularities. Technically, data mining is the process of analyzing data from many different dimensions or sides, and summarizing the relationships identified [Kouamou, 2011].

The data mining methods are divided essentially in two main types:

- verification-oriented (the system verifies the user's hypothesis);
- discovery-oriented (the system finds new rules and patterns autonomously) [Fayyad et al, 1996].

One taxonomy of data mining methods is given in [Maimon and Rokach, 2005]. In Figure 1, we give one widened variation focusing on the different classification schemes, used in data mining.

Verification methods deal with the evaluation of a hypothesis proposed by an external source. These methods include the most common approaches of traditional statistics, like *goodness-of-fit test*, *t-test of means*, and *analysis of*

variance. Such methods are not usually associated with data mining because most data mining problems are concerned with the establishment of a hypotheses rather than testing a known one.

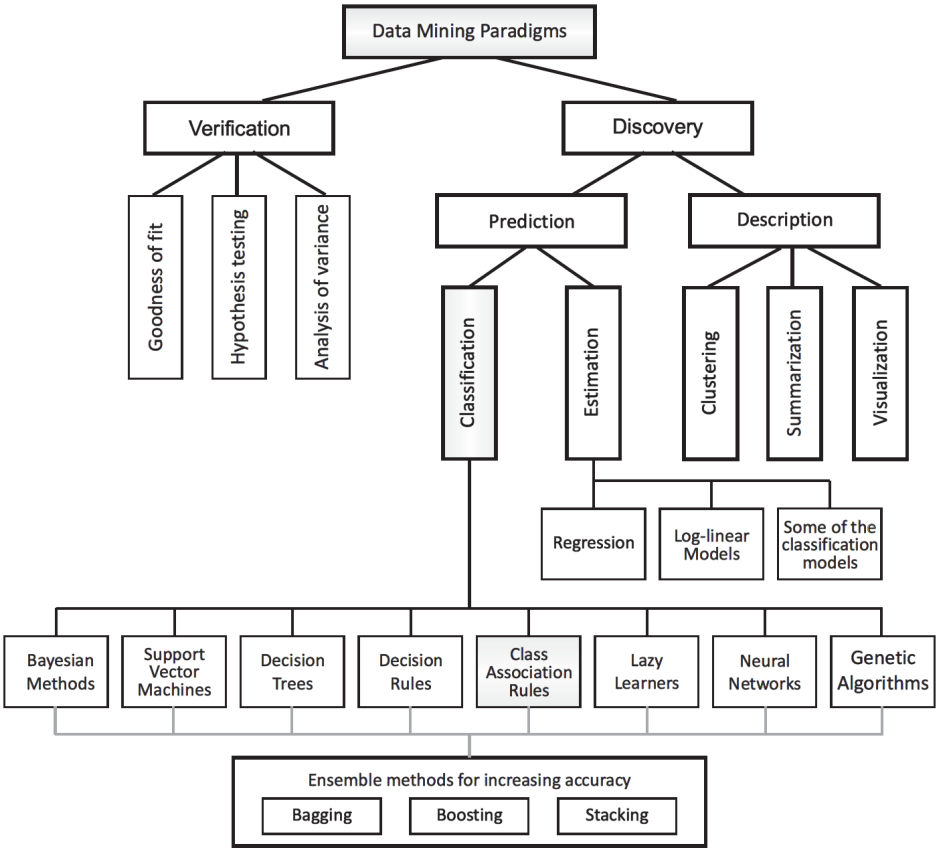


Figure 1. Detailed taxonomy of data mining methods, based on [Maimon and Rokach, 2005]

Discovery methods are methods that automatically identify patterns in the data. The discovery method branch consists of prediction approaches versus the description ones.

Description-oriented data mining methods focus on understanding how the underlying data operates. The main orientations of these methods are clustering, summarization and visualization.

Clustering is the process of grouping the data into classes or clusters, in a way that objects within a cluster have high similarity in comparison to one another but are very different to objects in other clusters. Dissimilarities are

assessed based on the attribute values describing the objects, using various approaches in distance measures.

Summarization is the process of reducing a text document or a larger corpus of multiple documents into a paragraph that conveys the main purpose of the text. There are two fundamental methods for this: extraction and abstraction. Extractive methods aim to select a subset of existing words, phrases, or sentences in the original text to form a summary. Unlike abstractive methods, where an internal semantic representation is built and then natural language generation techniques are used to create a summary that is more similar to one generated by a human. Such summary might contain words which are not explicitly present in the original text.

Visualization in data mining can be split into data visualization, mining result visualization, mining process visualization, and visual data mining. The variety, quality, and flexibility of visualization tools may strongly influence the usability, interpretability, and attractiveness of a data mining system.

Prediction-oriented methods aim to build a behavioral model that can create new and unobserved samples and is able to predict the values of one or more variables related to the sample.

Of course, the difference between description-oriented methods and prediction-oriented methods is very fuzzy.

Most of the discovery-oriented techniques are based on inductive learning [Mitchell, 1997], where a model is constructed explicitly or implicitly by generalizing from a sufficient number of training examples. The underlying assumption, derived from the inductive approach, is that the trained model is applicable to future examples, that have not yet been observed.

There are two main discovery-oriented techniques: classification and estimation. These two types of data analysis are used to extract models that describe important data classes or to predict future data trends. The main difference between classification and estimation is that classification maps the input space into predefined classes, while estimation models maps the input space into a real-valued domain.

Estimation models are used to construct a continuous-valued function, or ordered value, which are used as for estimation. The most commonly used techniques are different types of regression models (involving single predictor variable, or two or more predictor variables; linear or non-linear regression, etc.), while other models are also used (such as log-linear models that approximate discrete multidimensional probability distributions using logarithmic transformations). Some of the classifier models can also be tuned to be used for estimation (such as Decision Trees, Neural Networks, etc.) [Han and Kamber, 2006].

Classification models predict categorical (discrete, unordered) labels. Different kinds of classification models will be discussed later.

2.3 The "World" of Patterns

The word "instance" defined in a more general sense is that there can be a denotation of a real physical object, a process, a situation, etc.

The "attribute" describes a specific feature in an observed object or process, etc. Thus an instance is presented as a set of concrete values, which belong to a variety attributes. These attributes can be categorical or continuous. In our approach they must be discretized first.

Everything that characterizes instances and can be used in such logic operations as extraction, recognition, identification, etc. relates to the attributes. However, it should be noted that separation of attributes on "essential" and "unessential" is substantially conditional and depends on problems for which decision they are used.

In processes of recognition and production of models, the pattern is used as a Boolean function of the attributes, having the value "true" for instances from volume of pattern and "false" in other cases.

The pattern is usually defined in logic as a "concept", i.e. an idea that reflects essence of instances. Most of the used patterns are result of generalization of attributes that characterizes the instances of the class. The generalization is based on extraction of regularities from interconnected instances and/or patterns of the given class. The same idea may be extended for the regularities between classes.

From a philosophical point of view [Wagner, 1973] [Wille, 1982], the "pattern" ("concept") consists of two parts – extensional and intentional. The extensional part covers all instances belonging to this pattern, but the intentional part includes all the attributes that are representative for these instances. Relationships between instances and their attributes play an important role in determining the hierarchical relationship between patterns and attributes. The set of instances generalized in the pattern constitute its volume.

2.4 Pattern Recognition

The process of extracting patterns from datasets is called pattern recognition.

Pattern recognition algorithms generally aim to provide a reasonable answer for all possible inputs and to do "fuzzy" matching of inputs. This is opposed to pattern matching algorithms, which look for exact matches in the input with pre-existing patterns (typical example of which is regular expression matching).

Pattern recognition undergoes an important developing for many years. This is not a uni-modular research domain such as the classical mathematical sciences, it has a long history of establishment. The theoretical development in this domain include a number of sub disciplines such as feature selection, object

and feature ranking, analogy measuring, sequence labeling, parsing, clustering, supervised and unsupervised classification, etc. In the same time pattern recognition is indeed an integrated theory studying object descriptions and their classification models. This is a collection of mathematical, statistical, heuristic and inductive techniques of fundamental role in executing the intellectual tasks, typical for a human being – but on computers [Aslanyan and Sahakyan, 2010].

Classification is a typical example of pattern recognition; it aims to assign each input value to one from a given set of classes. Other examples are regression, which assigns a real-valued output to each input; sequence labeling, which assigns a class to each member of a sequence of values; and parsing, which assigns a parse tree to an input sentence, describing the syntactic structure of the sentence.

While the goals of data mining and pattern recognition appear to be similar, pattern recognition, which is split into the supervised learning (classification) and unsupervised learning (cluster analysis), can be interpreted in terms of rules like data mining. The main difference is in a scope of learning examples that are used within the process. Regular pattern recognition supposes satisfactory learning set able to determine the shapes of classes learned. There are many techniques focused on knowledge discovery based on a few examples, because the application area cannot provide enough learning examples [Arakelyan et al, 2009]. This is the case of High Dimensional Small Sample Size Data Analysis. Data Mining like the cluster analysis in pattern recognition can work without a given learning set. Instead, the rule template is given like the association rule template or a frequent fragment template. The commonsense reasoning is that Data Mining deals within large databases or on data flows.

2.5 Classification Algorithms

Classification is the task of identifying the sub-population to which new observations belong where the identity of the sub-population is unknown, on the basis of a training set of data containing observations with a known sub-population. The new individual items are placed into groups based on quantitative information on one or more measurements, traits or characteristics, etc.), and based on the training set in which previously decided groupings are already established.

In order to increase the obtained accuracy, ensemble methods, or so called meta-classifiers as upper stage, are used.

2.5.1 Classifiers

The variety of classification algorithms mainly can be grouped to: Bayesian Methods, Support Vector Machines, Decision Trees, Decision Rules, Class Association Rules, Lazy Learners, Neural Networks, and Genetic Algorithms.

- ✓ ***Bayesian Methods***

Bayesian classifiers are statistical classifiers which can predict class membership probabilities, such as the probability that a given instance belongs to a particular class. Bayesian classification is based on Bayes' theorem [Bayes, 1763] that shows the relation between two conditional probabilities which are the reverse of each other. Bayesian classifiers have exhibited high accuracy and speed when applied to large databases [Han and Kamber, 2006].

Naïve Bayesian classifiers assume that the effect of an attribute value on a given class is independent of the values of the other attributes. This assumption is called class conditional independence. Bayesian belief networks are graphical models that can also be used for classification, which allow the representation of dependencies among subsets of attributes.

- ✓ ***Support Vector Machines***

The Support Vector Machines (SVM) [Boser et al, 1992] use a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane. With an appropriate nonlinear mapping to a sufficiently high dimension, data from two classes can always be separated by a hyperplane. The SVM finds this hyperplane using support vectors ("essential" training instances) and margins (defined by the support vectors).

Although the training of even the fastest SVMs can be extremely time consuming, they are highly accurate, owing to their ability to model complex nonlinear decision boundaries. They are much less prone to over-fitting than other methods. The support vectors found also provide a compact description of the learned model.

- ✓ ***Decision Trees***

Decision tree induction is the learning of decision trees from class-labeled training instances. A decision tree is a flowchart-like tree structure, where each internal node (non-leaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label. The topmost node in a tree is the root node.

Given a question Q, for which the associated class label is unknown, the attribute values are tested against the decision tree. A path is traced from the

root to a leaf node, which holds the class prediction for Q. Decision trees can easily be converted to classification rules.

The construction of decision tree classifiers does not require any domain knowledge or parameter setting; they can handle high dimensional data; their representation in tree form is intuitive and generally is easy to understand by human users. The learning and classification steps of decision tree induction are simple and fast, and usually achieve good accuracy values.

One of the oldest tree classification methods is CHAID (acronym of Chi-squared Automatic Interaction Detector) [Kass, 1980]. CHAID builds non-binary trees making series of split operations, based on chi-square measure.

Other representative of this group was ID3 (Iterative Dichotomiser), developed by Ross Quinlan [Quinlan, 1986], after expanded to C4.5 [Quinlan, 1993]. Most algorithms for decision tree induction follow proposed ideas in ID3 and C4.5 for using a greedy approach in which decision trees are constructed in a top-down recursive divide-and-conquer manner. The top-down approach starts with a training set of instances and their associated class labels. The training set is recursively partitioned into smaller subsets as the tree is being built. J48 is a Weka implementation of C4.5 [Witten and Frank, 2005].

Representative Tree (shortly named REPTree) is an extension of C4.5 [Witten and Frank, 2005], which builds a decision tree using information gain reduction and prunes it using reduced-error pruning. Optimized for speed, it only sorts values for numeric attributes once. It deals with missing values by splitting instances into pieces, as C4.5 does. The algorithm has parameters – maximum tree depth and number of folds for pruning, which can be used when REPTree participates as classifier in ensemble schema.

- ✓ ***Decision Rules***

In the rule-based classifiers the learned model is represented as a set of IF-THEN rules. The "IF"-part of a rule is known as the rule antecedent. The "THEN"-part is the rule consequent. In the rule antecedent, the condition consists of one or more attribute tests (such as age = youth, student = yes) that are connected with logical function "AND". The rule's consequent contains a class label.

The rule induction is similar to tree induction but tree induction is breadth-first, as well as rule induction is depth-first (which means generating one rule at a time until all positive examples are covered) [Alpaydin, 2010].

One typical representative of a decision rules classifier is OneR [Holte, 1993]. OneR takes as input a set of examples, each with several attributes and a class. The aim is to infer a rule that predicts the class given the values of the attributes. The OneR algorithm chooses the most informative single attribute and bases the rule on this attribute alone. Shortly algorithm consists of creating the rules with antecedent each possible value of each attribute and consequent

corresponded class label, after that for each class label find the rule with maximal accuracy.

Another well-known classifier from this group is JRip. It is a Weka implementation of RIPPER (Repeated Incremental Pruning to Produce Error Reduction), proposed by William Cohen [Cohen, 1995]. RIPPER attempts to increase the accuracy of rules by replacing or revising individual rules. It uses reduced error pruning in order to decide when to stop adding more conditions to a rule; this reduces the amount of training data. RIPPER uses a heuristic based on the minimum description length principle as a stop-criterion. Rule induction is followed by a post-processing step that revises the rules in order to approximate what would have been obtained by a global pruning strategy.

- ✓ ***Class Association Rules***

Association rules show strong relations between attribute-value pairs (or items) that occur frequently in a given dataset. The general idea is to search for strong associations between frequent patterns (conjunctions of attribute-value pairs) and class labels. Association rules explore highly confident associations among multiple attributes. This approach helps to overcome some constraints introduced by decision-tree induction, which considers only one attribute at a time. Class Association Rules (CAR) algorithms will be discussed in more details in the next chapter.

- ✓ ***Lazy Learners***

All classifiers which had been already described belong to the so-called eager learners. Eager learners give a set of training instances and construct classification model before receiving query to classify.

Lazy classifiers are at the opposite side. They give training instances and only store them without any or with a. When a query is submitted, the classifier performs generalization in order to classify the query based on its similarity to the stored training instances.

Contrary to the eager learning methods, lazy learners do less work in the training phase and more work in the recognition phase.

The weak point of lazy classifiers is their computational expensiveness of the recognition process. On the other hand, they are well-suited to implementation on parallel hardware. They naturally support incremental learning.

There are two main groups of lazy learners: k-nearest-neighbor classifiers and case-based reasoning.

- ✓ *k-Nearest-Neighbor Classifiers*

Nearest-neighbor classifiers are based on learning by analogy, that is by comparing a given query with training instances similar to it. The training

instances are described by n attributes and are represented as points in a n-dimensional pattern space. Recognition consists of searching the pattern space for the k training instances ("k nearest neighbors") that are closest to the query. "Closeness" is defined in terms of a distance metric, such as Euclidean distance. Typical examples are IB1 and IBk [Aha and Kibler, 1991]. One interesting exception here is the KStar classifier which uses an entropy-based distance function [Cleary and Trigg, 1995].

- ✓ *Case-Based Reasoning*

Case-based reasoning classifiers use databases of problem solutions to solve new problems. Unlike nearest-neighbor classifiers, which store training instances as points in Euclidean space, a CBR would store instances as complex symbolic descriptions. When given a new case to classify, a case-based reasoner will first check if an identical training case exists. If one is found, then the accompanying solution to that case is returned. If no identical case is found, then the case-based reasoner will search for training cases having components that are similar to those of the new case. Conceptually, these training cases may be considered as neighbors of the new case. If cases are represented as graphs, this involves searching for subgraphs that are similar to subgraphs within the new case. The case-based reasoner tries to combine the solutions of the neighboring training cases in order to propose a solution for the new case. The case-based reasoner may employ background knowledge and problem-solving strategies in order to propose a feasible combined solution [Han and Kamber, 2006].

- ✓ ***Neural Networks***

The field of neural networks was originally conceived by psychologists and neurobiologists who sought to develop and test computational analogues of neurons. A neural network is a set of connected input/output units in which each connection has a weight associated with it. During the learning phase, the network learns by adjusting the weights so as to be able to predict the correct class label of the input instances.

Long training times; a great number of parameters that are typically best determined empirically; as well as poor interpretability are amongst the weaknesses of neural networks.

Advantages of neural networks, however, include their high tolerance to noisy data as well as their ability to classify patterns on which they have not been trained. They can be used when you may have little knowledge of the relationships between attributes and classes. They are well-suited for continuous-valued inputs and outputs, unlike most decision tree algorithms. Neural network algorithms are inherently parallel; parallelization techniques can be used to speed up the computation process.

In our experiments we have used Multi-Layer Perceptron, realized in Weka, for the representative of this class of algorithms.

✓ ***Genetic Algorithms***

Genetic algorithms attempt to incorporate in classification tasks the principles of natural evolution. An initial population is created consisting of randomly generated rules. Each rule can be represented by a string of bits. As a simple example, suppose that samples in a given training set are described by two Boolean attributes A_1 and A_2 and that there are two class labels coded by "0" and "1". The rule "if A_1 and not A_2 then 0" can be encoded as the bit string "100," where the two leftmost bits represent attributes A_1 and A_2 and the rightmost bit represents the class. For attributes/classes that have $k > 2$ values k bits are used to encode the attribute's values.

Based on the notion of survival of the fittest, a new population is formed to consist of the fittest rules in the current population, as well as offspring of these rules. Typically, the fitness of a rule is assessed by its classification accuracy on a set of training samples. Offsprings are created by applying genetic operators such as crossover and mutation. In crossover, substrings from pairs of rules are swapped to form new pairs of rules. In mutation, randomly selected bits in a rule's string are inverted. The process of generating new populations based on prior populations of rules continues until a population P evolves where each rule in P satisfies a the specified fitness threshold.

The weak point of genetic algorithms is their time consuming learning process. However, genetic algorithms are easily parallelizable and have been used for classification as well as other optimization problems. In data mining, they may be used to evaluate the fitness of other algorithms [Han and Kamber, 2006].

2.5.2 Ensemble Methods

Ensemble methods combine a series of k learned models, $M_1,...M_k$, with the aim of creating an improved composite model M^*. The main strategies here are bagging and boosting [Han and Kamber, 2006], as well as stacking [Witten and Frank, 2005].

✓ ***Bagging***

The term bagging denotes "bootstrap aggregation". Given a set D of d instances, bagging works as follows. For the iteration $i, i = 1,...,k$ a training set

D_i is sampled with replacement [StatTrek, 2011] from the original set D. Because sampling with replacement is used, some of the original tuples of D may not be included in D_i, where as others may occur more than once. A classifier model M_i is learned for each training set D_i. To classify a query Q each classifier M_i returns its class prediction which counts as one vote. The bagged classifier M^* counts the votes and assigns the class with the most votes to Q [Breiman, 1996].

The bagged classifier often has significantly greater accuracy than a single classifier derived from the original training data. It is also more robust to the effects of noisy data. The increased accuracy occurs because the composite model reduces the variance of the individual classifiers.

- ✓ ***Boosting***

In boosting, weights are assigned to each training instance. A series of k classifiers is iteratively learned. After a classifier M_i is learned, the weights are updated to allow the subsequent classifier M_{i+1} aggravating training instances that were misclassified by M_i. The final boosted classifier M^* combines the votes of each individual classifier, where the weight of each classifier's vote is a function of its accuracy.

- ✓ ***Stacking***

Stacked generalization or stacking, is an alternative method for combining multiple models. Unlike bagging and boosting, stacking is not used to combine models of the same type. Instead it is applied to models built by various learning algorithms (for example a decision tree inducer, a Naïve Bayes learner and an instance-based learning method). The usual procedure would be to estimate the expected error for each algorithm by cross-validating and then to choose the most appropriate one in order to form a model which can be used for prediction on future data combining outputs by voting. However voting criteria is not reliable enough. The problem is that it is not clear which classifier can be trusted (there are several types of classifiers which can be used). Stacking introduces the concept of the meta-learner which replaces the voting procedure. Stacking attempts to learn which classifiers are reliable using a different learning algorithm – the meta-learner – to discover what is the best way to combine the output from the base learners [Witten and Frank, 2005].

2.6 Discretization

The discretization process is known to be one of the most important data preprocessing tasks in data mining. Many machine learning techniques can only be applied to datasets which have been composed from the categorical attributes. However, in the real world, many attributes are naturally continuous, for example: height, weight, length, temperature, speed, etc. It is essential for a practical data mining system to be able to handle attributes of this sort. Although it would be possible to treat a continuous attribute as a categorical one using primary values, this is very unlikely to prove satisfactory. If the continuous attribute consists of a large number of different values in the training set, it is very likely that any particular value will only occur a small number of times, perhaps even only once, and rules that include tests for specific values will probably be of very little importance for a prediction [Bramer, 2007]. A solution to this problem would be to partition numeric variables into a number of sub-ranges and treat each sub-range as a category. This process of partitioning continuous variables into categories is usually termed as discretization.

There are several advantages of data discretization, which have been listed below:

- the experts usually describe the parameters using linguistic terms instead of exact values. In some ways, discretization can provide a better acknowledgement of attributes;
- it provides regularization because it is less prone to variation in the estimation of small fragmented data;
- the amount of data can be greatly reduced because of redundant data which can be identified and removed;
- it enables better performance for rule extraction.

Primary methods can be defined as:

- *Supervised* or *Unsupervised* [Dougherty et al, 1995]: If we look at unsupervised methods, continuous ranges are divided into sub-ranges by a user specified parameter – for instance, equal width (specifying range of values), equal frequency (number of instances in each interval), clustering algorithms – like k-means – (specifying a number of clusters). These methods may not be giving good results in cases where the distribution of the continuous values is not consistent and the outliers significantly affect the ranges. Of course, if there is not a class information available, unsupervised discretization is the only possible choice. In supervised discretization methods, class information is used to find the proper intervals which have been caused by cut-points. Different methods have been developed to use this class information for finding meaningful intervals in continuous attributes. Supervised discretization can be further characterized as *error-based*, *entropy-based* or *statistics-based* according to whether the intervals have been selected using

metrics based on error on the training data, entropy of the intervals, or a statistical measure;

- *Hierarchical* or *Non-hierarchical*: Hierarchical discretization selects cut points in an incremental process, forming an implicit hierarchy over the value range. The procedure can be *Split* or (and) *Merge* [Kerber, 1992]. There are methods which are non-hierarchical: for example: the methods used for scanning the ordered values only once and sequentially forming intervals;
- *Top-down* or *Bottom-up,* or in other words *Split* or *Merge* [Hussain et al, 1999]: Top-down methods start with one interval and split intervals in the process of discretization. Bottom-up methods start with the complete list of all the continuous values from the feature as cut-points and remove some of them by "merging" intervals as a discretization progresses. Different thresholds for stopping criteria are used;
- *Static* or *Dynamic*: In the static approach, discretization is done prior to the classification task (during the pre-processing phase). A dynamic method would discretize the continuous values while a classifier is being built, like illustrated in C4.5 [Quinlan, 1993]. Dynamic methods are mutually connected with a corresponding classification method, where the algorithm can work with real attributes;
- *Parametric* or *Non-parametric*: Parametric discretization requires input from the user, such as the maximum number of discretized intervals. Non-parametric discretization only uses information from data and does not need input from the user;
- *Global* or *Local* [Dougherty et al, 1995]: A local method would discretize in a localized region of the instance space (i.e. a subset of instances) while a global discretization method will use the entire instance space to discretize. Therefore a local method is usually associated with a dynamic discretization method where only a region of instance space is used for discretization;
- *Univariate* or *Multivariate* [Bay, 2000]: Univariate discretization quantifies one continuous feature at a time while multivariate discretization considers multiple features simultaneously.

In the experiments we conducted, the focus is on representatives of supervised methods. We have chosen two methods, which are different from the point of view of both the hierarchical direction and the forming of interval criteria. The first is Fayyad-Irani top-down method which is based on the optimization of the local measurement of the entropy and as stopping criterion – the Minimum Description Length (MDL) principle is used [Fayyad and Irani, 1993]. The second is Chi-merge – a bottom-up method based on the chi-square statistics measure.

2.7 Existing Data Mining Software Systems

There are several well-known data mining open-source systems which are aiming to support the study and research of the field of data mining, as well as the implication of some modules in real tasks.

✓ ***Weka***

Weka (http://www.cs.waikato.ac.nz/~ml/weka/) (Waikato Environment for Knowledge Analysis) is developed by the University of Waikato, New Zealand since 1993.

Weka is well-known suite for machine learning software that supports several typical data mining tasks, particularly data preprocessing, clustering, classification, regression, visualization, and feature selection.

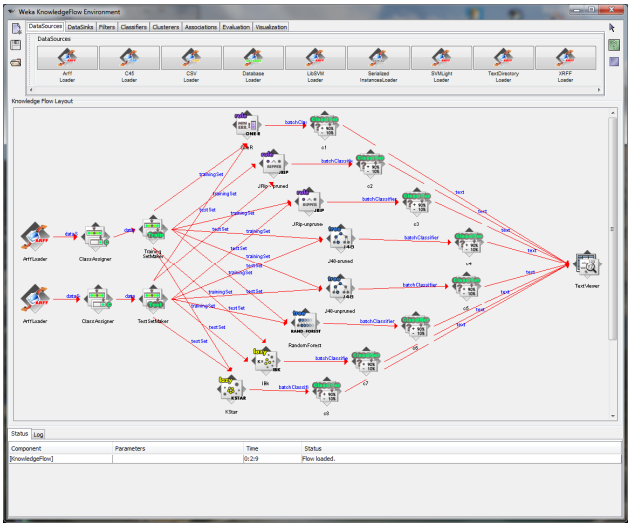


Figure 2. Weka knowledge flow interface

Weka is written in Java and it provides access to SQL databases utilizing Java Database Connectivity and it is also able to process the results returned by a database query. The main user interface is the Explorer, but the same functionality can be accessed from through the command line interface or the component-based Knowledge Flow interface (Figure 2).

In 2005, Weka received the SIGKDD Data Mining and Knowledge Discovery Service Award. In 2006, Pentaho Corporation acquired an exclusive license to

use Weka for business intelligence. It forms the data mining and predictive analytics component of the Pentaho business intelligence suite.

In the experimental part of this dissertation, we make comparison of other classifiers using the Weka environment. Figure 2 is a screenshot of the knowledge flow task, used in our experiments with various datasets.

✓ ***LUCS-KDD Repository***

The LUCS-KDD Repository (http://www.csc.liv.ac.uk/~frans/KDD/) (Liverpool University of Computer Science – Knowledge Discovery in Data) has been developed and maintained by the Department of Computer Science, University of Liverpool, UK since 1997. This repository provides a common environment for research tasks and comparison between different algorithms, some of which are a product of the group that supports repository. A number of algorithms were developed since the work on LUCS-KDD commenced; they are released as open access and serve different applications. The team adopted several algorithms: association rule mining (Apriori-T and TFP) and class association rules algorithms (TFPC) featuring preprocessing of the data and set-enumeration tree structures (the P-tree and the T-tree) to facilitate search.

The experiments with CMAR classifier are made using its program realization in LUCS-KDD Repository.

✓ ***Orange***

Orange (http://www.ailab.si/orange) is developed and maintained at the Faculty of Computer and Information Science, University of Ljubljana, Slovenia.

Orange is an open component-based data mining and machine learning software suite that features friendly yet powerful, fast and versatile visual programming front-end for data analysis and visualization, and Python bindings and libraries for scripting.

It is written in C++ and Python, and its graphical user interface is based on the cross-platform Qt framework.

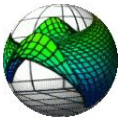
It includes a comprehensive set of components for data preprocessing, feature scoring and filtering, modeling, model evaluation and exploration techniques.

✓ ***RapidMiner***

RapidMiner (http://rapidminer.com/), formerly called YALE (Yet Another Learning Environment), is created and maintained by Rapid-I GMBH, Germany. It is a machine learning and data mining environment written in Java which is utilized for both research and real-world data mining tasks. It enables

experiments to be made up of a huge number of arbitrarily nestable operators, which are detailed in XML files and are made with RapidMiner's graphical user interface. RapidMiner provides more than 500 operators for all main machine learning procedures; it also combines learning schemes and attribute evaluators of the Weka learning environment. It is available as a stand-alone tool for data analysis and as a data-mining engine that can be integrated into other products.

✓ ***jHepWork***



Designed for scientists, engineers and students, jHepWork (http://jwork.org/jhepwork/) is a multiplatform free and open-source Java-based data analysis framework created as an attempt to develop a data analysis environment using open-source packages with a comprehensible user interface which would be competitive to commercial software. It is specifically made for interactive scientific plots in 2D and 3D and includes numerical scientific libraries implemented in Java for mathematical functions, random numbers, and other data mining algorithms. jHepWork is based on a high-level programming language Jython, but Java coding can also be used to call jHepWork numerical and graphical libraries.

The jHepWork is a collective effort of many people dedicated to open-source scientific software, coordinated by Sergei Chekanov since 2005.

✓ ***SIPINA***

SIPINA (http://eric.univ-lyon2.fr/~ricco/sipina.html), has been developed at the University of Lyon, France since 1995. It is an open data mining software which implements a number of supervised learning paradigms, but mainly classification tree software (it specializes on Classification Trees algorithms such as ID3, CHAID, and C4.5, but other supervised methods e.g. k-NN, Multilayer Perceptron, Naive Bayes, etc. are also available.

SIPINA can handle both continuous and discrete attributes. SIPINA theoretical limitations are 16,384 attributes and 500,000,000 examples. Because it loads the complete dataset in the memory before the learning process, the true limitation is the capacity of the computer memory available.

SIPINA allows feature transformations (discretizing an attribute, coding a set of attributes from a discrete attribute, etc.), feature selection using "filter methods" (selecting the best predictive attributes independently of the supervised algorithms used prior to induction) or "wrapper methods" (where a supervised algorithm selects the best attributes), error evaluation and classification.

✓ ***TANAGRA***

The TANAGRA project is a successor of SIPINA (http://eric.univ-lyon2.fr/~ricco/tanagra/en/tanagra.html) [Rakotomalala, 2005]. It combines several data mining methods from the domains of exploratory data analysis, statistical learning, machine learning and databases.

TANAGRA implements various supervised learning algorithms, more specifically an interactive and visual construction of decision trees. TANAGRA contains some supervised learning but also other paradigms such as clustering, factorial analysis, parametric and nonparametric statistics, association rule, etc.

The primary goal of the TANAGRA project is to make available to researchers and students an easy-to-use data mining software, conforming to the present norms of the software development in this domain, and allowing to analyze either real or synthetic data.

A further goal of TANAGRA is to offer researchers an architecture allowing them to easily add their own data mining methods, which would allow to compare performances and establish benchmarks.

The last goal targeting novice developers is to disseminate a possible methodology for the development of this kind of software. Developers can take advantage of free access to source code and can see how this sort of software is built and what problems to avoid; they also can observe what are the main stages of the implementation project, and which tools and code libraries to use. Thus TANAGRA can be considered as a pedagogical tool for learning programming techniques.

✓ ***KNIME***

KNIME (http://www.knime.org/) (Konstanz Information Miner), maintained by KNIME GMBH, Germany, is a user friendly, intelligible, and comprehensive open-source data integration, processing, analysis, and exploration platform. It gives users the ability to visually create data flows or pipelines, selectively execute some or all analysis steps, and later study the results, models, and interactive views. KNIME is written in Java, and it is based on Eclipse and makes use of its extension method to support plugins thus providing additional functionality. Through plugins, users can add modules for text, image, and time series processing and can integrate a range of open source projects, such as R programming language, Weka, the Chemistry Development Kit, and LibSVM.

KNIME has been selected by Gartner as Cool Vendor 2010 in the key technology areas Analytics, Business Intelligence, and Performance Management.

✓ ***AlphaMiner***

AlphaMiner (http://www.eti.hku.hk/alphaminer/index.html) is developed by the E-Business Technology Institute of the University of Hong Kong.

The technology of Business Intelligence (BI) helps companies to improve business decision making. Over the past decade, international companies in the banking, telecommunications, insurances, retails and e-business sectors have successfully used BI to solve numerous business problems in marketing, customer service, cross selling, customer retention, fraud detection and risk management. BI solutions are costly and only large enterprises can afford them. AlphaMiner data mining system provides affordable BI technologies by leveraging existing open source technologies and empowers small companies with the capability to make better decisions in the fast changing business environment. Plug-able component architecture provides extensibility for adding new BI capabilities in data import and export, data transformations, modeling algorithms, model assessment and deployment. Versatile data mining functions offer powerful analytics to conduct industry specific analysis including customer profiling and clustering, product association analysis, classification and prediction.

✓ ***ELKI***

ELKI (http://www.dbs.ifi.lmu.de/research/KDD/ELKI/) (Environment for DeveLoping KDD-Applications Supported by Index-Structures), developed by the Institute for Computer Science of University of Munich, Germany [Achtert et al, 2010], is a data mining software framework with a focus on clustering and outlier detection methods written in Java.

As discussed above, data mining research makes use of multiple algorithms for similar tasks. A fair and useful comparison of these algorithms is difficult due to several reasons:

- most of the software tools are commercial and their implementations are not easily available;
- even when different software implementations are available, an evaluation in terms of efficiency is biased to evaluate the efforts of different authors in efficient programming instead of evaluating algorithmic merits. Probably this is influenced by the fact that usability evaluations could be performed easier than an objective evaluation of the algorithms.

On the other hand, efficient data management tools like index-structures can show considerable impact on data mining tasks and are therefore useful for a broad variety of algorithms.

In ELKI, data mining algorithms and data management tasks are separated and allow for separate evaluation. This distinguishes ELKI among data mining frameworks like Weka framework for index structures like GiST. At the same time, ELKI is open to arbitrary data types, distance or similarity measures, or file formats. The fundamental approach applied in ELKI is the independence of file parsers or database connections, data types, distances, distance functions, and data mining algorithms. Helper classes, e.g. for algebraic or analytic computations, are available for all algorithms on equal terms.

- ✓ ***R***

R (http://www.r-project.org/) is a programming language and software environment for statistical computing and graphics. The R language is widely used for statistical software development and data analysis.

R is an implementation of the S programming language combined with lexical scoping semantics inspired by Scheme. R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is now developed by the R Development Core Team.

R is part of the GNU project. Its source code is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems. R uses a command line interface, however several graphical user interfaces are available for use with R.

The capabilities of R are extended through user-submitted packages, which allow specialized statistical techniques, graphical devices, as well as import/export capabilities to many external data formats. The "Task Views" page (http://cran.r-project.org/web/views/) on the CRAN website lists the wide range of applications (Finance, Genetics, Machine Learning, Medical Imaging, Social Sciences and Spatial statistics) to which R has been applied and for which packages are available. For instance, the "arules" package allows extracting association rules with R (http://cran.univ-lyon1.fr/web/packages/arules/index.html).

- ✓ ***Rattle***

Rattle (the R Analytical Tool To Learn Easily) (http://rattle.togaware.com/), created and supported by Togaware Pty Ltd., Australia, is a open-source data mining toolkit used to analyze very large collections of data [Williams, 2009].

Rattle presents statistical and visual summaries of data, transforms data into forms that can be readily modeled, builds both unsupervised and supervised models from the data, presents the performance of models graphically, and scores new datasets.

The scientific blog DecisionStats (29.04.2010) listed Rattle as one of the top 10 graphical user interfaces in statistical software.

Through a simple and logical graphical user interface based on Gnome, Rattle can be used by itself to deliver data mining projects. Rattle also provides an entry into sophisticated data mining using the open source and free statistical language R.

Rattle runs under GNU/Linux, Macintosh OS/X, and MS/Windows. The aim is to provide an intuitive interface that takes you through the basic steps of data mining, as well as illustrating the R code that is used to achieve this. Whilst the tool itself may be sufficient for all of a user's needs, it also provides a stepping stone to more sophisticated processing and modeling.

2.8 Standardization and Interoperability

With the advancement of modern information technologies and the boost in data mining, the use of knowledge discovery becomes an everyday practice. Business Intelligence, Web Mining, Medical Diagnostics, Drives and Controls, GPS Systems, Global Monitoring for Environment and Security, etc. are only few of the application areas where data mining is a core component.

Data mining gradually became an emergent technology across multiple industries and sectors. Such expanded and enlarged use means that it is necessary to design a data mining environment which meets the following requirements:

- data interoperability (currently each system uses its own notation for data entry, for instance C4.5-standard, arff-standard, etc.);
- openness for adding new algorithms to the environment;
- modularity in order to allow combining of different techniques that became a part of a global process;
- the modules must allow use by different systems, not only a closed use within their own environment;
- user flexibility and possibility to guide the entire data mining process.

Kouamou described in 2011 the logical structure of data mining environment [Kouamou, 2011]. The author suggests that integration and interoperability of modern data mining environments are achieved by application of modern industrial standards, such as XML-based languages.

Notably, such systems are able to import and export models in PMML (Predictive Model Markup Language), which provides a standard way to represent data mining models which allows sharing between different statistical applications. PMML is an XML-based language developed by the Data Mining Group, an independent group of numerous data mining companies.

The adoption of standards in this discipline already made it possible to develop procedures of data exchange between various platforms. At the same time there are reflections on the standardization of a data mining process model. The presentation of these efforts demonstrated that the challenge for the future is to develop and popularize widely accepted standards in data mining environment; if developed and adopted, such a standard will stimulate major industry growth and interest. It would also promote development and delivery of solutions that use business language, resulting in performing projects faster, cheaper, more manageably, and more reliably.

Conclusion

In this chapter we made an overview of data mining and knowledge discovery. Looking at the taxonomy of the main types of data mining methods we focus our attention on the segment of discovery-oriented methods and more specifically on classification algorithms.

We observed the wide family of classification algorithms, dividing into following main categories: Bayesian Methods, Support Vector Machines, Decision Trees, Decision Rules, Class Association Rules, Lazy Learners, Neural Networks and Genetic Algorithms.

We also presented the ensemble methods such as Bagging, Boosting and Stacking used to increase classification accuracy,. Such methods can be used as a more advanced stage on the primary classifiers.

Most of the classification algorithms deal with categorical attributes. Because of this, we made a brief overview of discretization methods as important preprocessing step for such algorithms.

Further we looked at several well-known open-source systems aimed to support research work, or having significant influence on real work in the field of data mining.. We provided brief descriptions of the following existing open-source data mining software systems: Weka, LUCS-KDD Repository, Orange, RapidMiner, JHepWork, SIPINA, TANAGRA, KNIME, AlphaMiner, ELKI, R, and Rattle.

In this context the issues of standardization and interoperability are becoming crucial. We elaborated on several aspects that gained importance lately: data interoperability; opening for adding new algorithms to existing environments; allowing combination of different techniques using module approach of separate elements; user flexibility and possibility to guide the entire data mining process.

3 CAR Algorithms

Abstract

In this chapter we discuss in detail CAR algorithms.

We present the main types of algorithms for association rule mining, pruning techniques, quality rule measures and rule ordering strategies.

We also describe a number of specific CAR algorithms.

3.1 Introduction

Association rule mining quickly became a popular instrument to model relationships between class labels and features from a training set [Bayardo, 1998]. It appeared initially within the field of market basket analysis for discovering interesting rules from large data collections [Agrawal et al, 1993]. Since then, many associative classifiers were proposed, mainly differing in the strategies used to select rules for classification and in the heuristics used for pruning rules. "Class association rules" (CAR) algorithms have its important place in the family of classification algorithms.

Zaïane and Antonie suggested that the five major advantages of associative classifiers are the following [Zaïane and Antonie, 2005]:

- the training is very efficient regardless of the size of the training set;
- training sets with high dimensionality can be handled with ease and no assumptions are made on dependence or independence of attributes;
- the classification is very fast;
- classification based on association methods presents higher accuracy than traditional classification methods [Liu et al, 1998] [Li et al, 2001] [Thabtah et al, 2005] [Yin and Han, 2003];
- the classification model is a set of rules easily interpreted by human beings and can be edited [Sarwar et al, 2001].

Within the data mining community, research on classification techniques has a long and fruitful history. However, classification techniques based on association rules, are relatively new. The first associative classifier CBA was introduced by [Liu et al, 1998]. During the last decade, various other associative classifiers were introduced, such as CMAR [Li et al, 2001], ARC-AC and ARC-BC [Zaïane and Antonie, 2002], CPAR [Yin and Han, 2003], CorClass [Zimmermann and De Raedt, 2004], ACRI [Rak et al, 2005], TFPC [Coenen and Leng, 2005], HARMONY [Wang and Karypis, 2005], MCAR [Thabtah et al, 2005], CACA [Tang and Liao, 2007], ARUBAS [Depaire et al, 2008], etc.

CAR-algorithms are based on a relatively simple idea. Given a training set with transactions where each transaction contains all features of an object in addition to the class label of the object, the association rules are constructed, which have as consequent a class label. Such association rules are named "class association rules" (CARs).

Generally the structure of CAR-algorithms consists of three major data mining steps:

1. Association rule mining.
2. Pruning (optional).
3. Recognition.

The mining of association rules is a typical data mining task that works in an unsupervised manner. A major advantage of association rules is that they are theoretically capable of revealing all interesting relationships in a database. But for practical applications the number of mined rules is usually too large to be exploited entirely. This is why the pruning phase is stringent in order to build accurate and compact classifiers. The smaller the number of rules a classifier needs to approximate the target concept satisfactorily, the more human-interpretable is the result.

3.2 Association Rule Mining

Association rule mining was first introduced in [Agrawal et al, 1993]. It aims to extract interesting correlations, frequent patterns, associations, or casual structures among sets of instances in the transaction databases or other data repositories.

Association rule mining itself has a wide range of application domains such as market basket analysis, medical diagnosis/research, Website navigation analysis, homeland security and so on. In parallel, it participates as a step in the training process of CAR classifiers.

The datasets can be represented in two forms:

- transactional datasets;
- rectangular datasets.

In transactional datasets each record (transaction) can contain different number of items and order of the items can be arbitrary.

In rectangular datasets each record has the same number of attributes and position of the attribute value is fixed and corresponds to the attribute.

These differences are not particularly difficult to address since there is an easy way of converting transactional to binary rectangular dataset by ordering all possible items and pointing the presence of a concrete item with 1 (true) and respectively the absence with 0 (false).

The rectangular dataset also become transactional representation using attribute-value pairs in description of each record.

The description of the problem of association rule mining is firstly presented in [Agrawal et al, 1993]. The description of the problem provided below follows the one given in [Goethals, 2002].

Let $\mathbf{D}$ be a set of items.

A set $X = \{i_1, ..., i_k\} \subseteq \mathbf{D}$ is called an itemset or a k-itemset if it contains k items.

A transaction over $\mathbf{D}$ is a couple $T = (tid, I)$ where tid is the transaction identifier and I is an itemset. A transaction $T = (tid, I)$ is said to support an itemset $X \subseteq \mathbf{D}$ if $X \subseteq I$.

A transaction database D over $\mathbf{D}$ is a set of transactions over $\mathbf{D}$.

The cover of an itemset X in D consists of the set of transaction identifiers of transactions in D that support X: $cover(X, D) := \{tid \mid (tid, I) \in D, X \subseteq I\}$.

The support of an itemset X in D is the number of transactions in the cover of X in D: $support(X, D) := |cover(X, D)|$. Note that $|D| = support(\{\}, D)$.

An itemset is called frequent if its support is no less than a given absolute minimal support threshold $MinSup$, with $0 \le MinSup \le |D|$.

Let D be a transaction database over a set of items $\mathbf{D}$, and $MinSup$ a minimal support threshold. The collection of frequent itemsets in D with respect to $MinSup$ is denoted by $F(D, MinSup) := \{X \subseteq \mathbf{D} \mid support(X, D) \ge MinSup\}$.

An association rule is an expression of the form $X \Rightarrow Y$, where X and Y are itemsets, and $X \cap Y = \{\}$. Such a rule expresses the association that if a transaction contains all items in X, then that transaction also contains all items in Y. X is called the body or antecedent, and Y is called the head or consequent of the rule.

The support of an association rule $X \Rightarrow Y$ in D, is the support of $X \cup Y$ in D. An association rule is called frequent if its support exceeds a given minimal support threshold $MinSup$.

The confidence or accuracy of an association rule $X \Rightarrow Y$ in D is the conditional probability of having Y contained in a transaction, given that X is contained in that transaction:

$$confidence(X \Rightarrow Y, D) := P(Y \mid X) = \frac{support(X \cup Y, D)}{support(X, D)}.$$

The rule is called confident if $P(Y \mid X)$ exceeds a given minimal confidence threshold $MinConf$, with $0 \leq MinConf \leq 1$.

Especially in the case of classification association rules the head consists of only one attribute-value pair. In the case of rectangular data one of the columns contains class labels that divide the dataset into separate extensional parts.

Generally, an association rules mining algorithm consists of the following steps:

1. The set of candidate k-item-sets is generated by 1-extensions of the large (k-1)-item-sets generated in the previous iteration.
2. Supports for the candidate k-item-sets are generated by a pass over the database.
3. Item-sets that do not have the minimum support are discarded and the remaining item-sets are called large (frequent) k-item-sets.
4. This process is repeated until no more large item-sets are found to generate association rules from those large item-sets with the constraints of minimal confidence.

In many cases, the algorithms generate an extremely large number of association rules, often in thousands or even millions; in addition to this the association rules are sometimes very large. It is nearly impossible for the end-users to comprehend or validate such large number of complex association rules, thereby limiting the usefulness of the data mining results. Several researchers suggested strategies aimed at reducing the number of association rules:

- extracting of rules based on user-defined templates or instance constraints [Baralis and Psaila, 1997] [Ashrafi et al, 2004];
- developing interestingness measures to select only interesting rules [Hilderman and Hamilton, 2002]. For instance [Jaroszewicz and Simovici, 2002] proposed a solution to the problem using the Maximum Entropy approach;
- proposing inference rules or inference systems to prune redundant rules and thus present smaller, and usually more understandable sets of association rules to the user [Cristofor and Simovici, 2002];
- creating new frameworks for mining association rule to find association rules with different formats or properties [Brin et al, 1997].

Depending of the specificity of the observed problem many additional question arise. For instance [Liu et al, 1999] present an approach to the rare instance problem. The dilemma that arises in the rare instance problem is that searching for rules that involve infrequent (i.e., rare) instances requires a low support but using a low support will typically generate many rules that are of no interest. Using a high support typically reduces the number of rules mined but will eliminate the rules with rare instances. The authors attack this problem by allowing users to specify different minimum supports for the various instances in their mining algorithm.

For filtering out the interesting rules also sometimes the lift measure is used [Brin et al, 1997], which shows how many times more often two items occur together than expected if they where statistically independent.

The computational cost of association rules mining can be reduced by sampling the database, by adding extra constraints on the structure of patterns, or through parallelization.

Techniques for association rule discovery have gradually been adapted to parallel systems in order to take advantage of the higher speed and greater storage capacity that they offer. The transition to a distributed memory system requires the partitioning of the database among the processors, a procedure that is generally carried out indiscriminately. [Parthasarathy et al, 2001] wrote an excellent survey on parallel association rule mining with shared memory architecture covering most trends, challenges, and approaches adopted for parallel data mining.

3.2.1 Creating Association Rules

During the first stage, several techniques for creating association rules are used, which mainly are based on:

- Apriori algorithm [Agrawal and Srikant, 1994] (CBA, ARC-AC, ARC-BC, ACRI, ARUBAS);
- FP-tree algorithm [Han and Pei, 2000] (CMAR);
- FOIL algorithm [Quinlan and Cameron-Jones, 1993] (CPAR);
- Morishita & Sese Framework [Morishita and Sese, 2000] (CorClass).

Generating association rules can be made from all training transactions together (such it is in ARC-AC, CMAR, CBA) or can be made for transactions grouped by class label (as it is in ARC-BC), which offers small classes a chance to have representative classification rules.

We provide a brief overview of some distinctive algorithms created during the recent years, which are used or can be implemented at the step of creating the pattern set of CAR algorithms.

✓ ***AIS***

The AIS algorithm [Agrawal et al, 1993] was the first algorithm proposed for mining association rule in the early 90s, when a task for emulating the biological immune system in the real world scenarios became actual. AIS algorithm uses candidate generation to detect the frequent item-sets. The candidates are generated on the fly and are compared with previously found frequent item-sets. In this algorithm only one instance of consequent association rules are generated, which means that the consequent of those rules only contain one instance, for example we only generate rules like $X \cap Y \Rightarrow Z$ but not those rules as $X \Rightarrow Y \cap Z$. The main drawbacks of the AIS algorithm are too many passes over the whole database and too many candidate item-sets that finally turned out to be small are generated, which requires considerable memory and involves significant effort that turned out to be useless.

✓ ***Apriori***

The Apriori [Agrawal and Srikant, 1994] is the most popular algorithm for producing association rules. It created new opportunities to mine the data. Since its inception, many scholars have improved and optimized the Apriori algorithm and have presented new Apriori-like algorithms. Apriori uses pruning techniques to avoid measuring certain item-sets, while guaranteeing completeness. These are the item-sets that the algorithm can prove will not turn out to be large.

However, there are two bottlenecks of the Apriori algorithm. One is the complex candidate generation process that uses most of the time and memory because of the multiple scans of the database. Based on the Apriori algorithm, many new algorithms were designed with some modifications or improvements.

The Apriori algorithm for finding frequent item-sets makes multiple passes over the data. In the k -th pass it finds all item-sets having k instances called the k-item-sets. Each pass consists of two phases. Let F_k represent the set of frequent k-item-sets, and C_k the set of candidate k-item-sets (potentially frequent item-sets). The candidate generation phase where the set of all frequent (k-1)-item-sets, F_{k-1}, found in the ($k-1$)-th pass is applied first and it is used to generate the candidate item-sets C_k. The candidate generation procedure ensures that C_k is a superset of the set of all frequent k-item-sets. A specialized hash-tree data structure is used to store C_k. Then, data is scanned in the support counting phase. For each transaction, the candidates in C_k contained in the transaction are determined using the hash-tree data structure and their support count is incremented. At the end of the pass, C_k is examined

to determine which of the candidates are frequent, yielding F_k. The algorithm terminates when F_k or C_{k+1} becomes empty.

Several optimizations of Apriori algorithm are available, such as:

- PASCAL [Bastide et al, 2000], which introduces the notions of key patterns and use inference of other frequent patterns from the key patterns without access to the database;
- Category-based Apriori algorithm [Do et al, 2003], which reduces the computational complexity of the mining process by bypassing most of the subsets of the final item-sets;
- Apriori-T [Coenen et al, 2004], which makes use of a "reverse" set enumeration tree where each level of the tree is defined in terms of an array (i.e. the T-tree data structure is a form of Trie);
- FDM [Cheung et al, 1996], which is a parallelization of Apriori for shared machines, each with its own partition of the database. At every level and on each machine, the database scan is performed independently on the local partition. Then a distributed pruning technique is employed.

✓ *FP-Tree*

FP-Tree [Han and Pei, 2000] is another milestone in the development of association rule mining, which breaks the main bottlenecks of Apriori [Kotsiantis and Kanellopoulos, 2006]. The frequent item-sets are generated with only two passes over the database and without any candidate generation process. FP-tree is an extended prefix-tree structure storing crucial, quantitative information about frequent patterns. Only frequent length-1 instances will have nodes in the tree, and the tree nodes are arranged in such a way that more frequently occurring nodes will have better chances of sharing nodes than less frequently occurring ones. FP-Tree scales much better than Apriori because as the support threshold goes down, the number as well as the length of frequent item-sets increase dramatically. The frequent patterns generation process includes two sub processes: constructing the FT-Tree, and generating frequent patterns from the FP-Tree. The mining result is the same with Apriori series algorithms.

To sum up, the efficiency of FP-Tree algorithm accounts for three reasons:

1. The FP-Tree is a compressed representation of the original database because only those frequent instances are used to construct the tree, other irrelevant information are pruned.
2. This algorithm only scans the database twice.
3. FP-Tree uses a divide and conquers method that considerably reduced the size of the subsequent conditional FP-Tree.

Every algorithm has his limitations, for FP-Tree it is difficult to be used in an interactive mining system. Another limitation is that FP-Tree is that it is not suitable for incremental mining.

✓ *TreeProjection*

The innovation brought by TreeProjection [Agarwal et al, 2000] is the use of a lexicographic tree, which requires substantially less memory than a hash tree. The number of nodes in its lexicographic tree is exactly that of the frequent item-sets. The support of the frequent item-sets is counted by projecting the transactions onto the nodes of this tree. This improves the performance of counting the number of transactions that have frequent item-sets. The lexicographic tree is traversed in a top-down fashion. The efficiency of TreeProjection can be explained by two main factors:

1. the transaction projection limits the support counting in a relatively small space.
2. the lexicographical tree facilitates the management and counting of candidates and provides the flexibility of picking efficient strategy during the tree generation and transaction projection phrases.

✓ *Matrix Algorithm*

The Matrix Algorithm [Yuan and Huang, 2005] generates a matrix, which entries 1 or 0 by passing over the database only once, and then the frequent candidate sets are obtained from the resulting matrix. Finally, association rules are mined from the frequent candidate sets. Experimental results confirm that the proposed algorithm is more effective than Apriori Algorithm.

✓ *Sampling Algorithms*

For obtaining associations, several algorithms use sampling. Some examples are provided below:

- Toivonen's sampling algorithm [Toivonen, 1996]. This approach is a combination of two phases. During phase 1 a sample of the database is obtained and all associations in the sample are found. These results are then validated against the entire database. To maximize the effectiveness of the overall approach, the author makes use of lowered minimum support on the sample. Since the approach is probabilistic (i.e. dependent on the sample containing all the relevant associations) not all the rules may be found in this first pass. Those associations that were deemed not frequent in the sample but were actually frequent in the entire dataset are used to construct the complete set of associations in phase 2;
- Progressive sampling [Parthasarathy, 2002] is yet another approach; it relies on a novel measure of model accuracy (self-similarity of associations across progressive samples), the identification of a representative class of frequent item-sets that mimic (extremely accurately) the self-similarity values across the entire set of

associations, and an efficient sampling methodology that hides the overhead of obtaining progressive samples by overlapping it with useful computation;

- Sampling Error Estimation algorithm [Chuang et al, 2005] aims to identify an appropriate sample size for mining association rules. It has two advantages. First, it is highly efficient because an appropriate sample size can be determined without the need of executing association rules. Second, the identified sample size is very accurate, meaning that association rules can be highly efficiently executed on a sample of this size to obtain a sufficiently accurate result;
- Sampling large datasets with replacement [Li and Gopalan, 2004] is used when data comes as a stream flowing at a faster rate than can be processed. Li and Gopalan derive the sufficient sample size based on central limit theorem for sampling large datasets with replacement.

✓ ***Partition***

Partition [Savasere et al, 1995] is fundamentally different from other algorithms because it reads the database at most two times to generate all significant association rules. In the first scan of the database, it generates a set of all potentially large item-sets by scanning the database once and dividing it in a number of non-overlapping partitions. This set is a superset of all frequent item-sets so it may contain item-sets that are not frequent. During the second scan, counters for each of these item-sets are set up and their actual support is measured.

✓ ***FOIL***

FOIL (First Order Inductive Learner) is an inductive learning algorithm for generating classification association rules (CARs) developed by Quinlan and Cameron-Jones in 1993 [Quinlan and Cameron-Jones, 1993] and further developed by Yin and Han to produce the PRM (Predictive Rule Mining) CAR generation algorithm [Yin and Han 2003]. PRM was then further developed, by Yin and Han, to produce CPAR (Classification based on Predictive Association Rules).

FOIL is a sequential covering algorithm that learns first-order logic rules. It learns new rules one at a time, removing the positive examples covered by the latest rule before attempting to learn the next rule.

The hypothesis space search performed by FOIL is best understood by viewing it hierarchically. Each iteration through FOIL'S outer loop adds a new rule to its disjunctive hypothesis. The effect of each new rule is to generalize the current disjunctive hypothesis (i.e., to increase the number of instances it classifies as positive), by adding a new disjunct. Viewed at this level, the search is a specific-to-general search through the space of hypotheses, beginning with

the most specific empty disjunction and terminating when the hypothesis is sufficiently general to cover all positive training examples. The inner loop of FOIL performs a finer-grained search to determine the exact definition of each new rule. This inner loop searches a second hypothesis space, consisting of conjunctions of literals, to find a conjunction that will form the preconditions for the new rule. FOIL employs a specific performance FOIL Gain that differs from the entropy measure. This difference follows from the need to distinguish between different bindings of the rule variables and from the fact that FOIL seeks only rules that cover positive examples [Mitchell, 1997].

✓ ***Morishita & Sese Framework***

This framework [Morishita and Sese, 2000] efficiently computes significant association rules according to common statistical measures such as a chi-squared value or correlation coefficient. Because of anti-monotonicity of these statistical metrics, Apriori algorithm is not suitable for association rule generation. Morishita and Sese present a method of estimating a tight upper bound on the statistical metric associated with any superset of an item-set, as well as the novel use of the resulting information of upper bounds to prune unproductive supersets while traversing item-set lattices.

3.2.2 Rule Quality Measures

The process of generating association rules usually creates an extremely big number of patterns. This bottleneck imposes the necessity of measuring the significance, respectively redundancy of the generated rules and ordering using different criteria.

Here we will mention some examples of used ranking of association rules.

For a rule P and a class-labeled dataset $D = \{R_i \mid i = 1,...,n\}$ several kinds of rule quality measures and combinations of them are used:

- The time of generation of the rule. This is a weak restriction used when all constrains before order two rules in equal places;
- $ncovers(P)$: the number of instances covered by P (i.e. $R_i : body(P) \subseteq body(R_i)$);
- $pos(P)$: the number of instances correctly classified by P (i.e. $R_i : body(P) \subseteq body(R_i)$ and $head(P) = head(R_i)$);
- $neg(P)$: the number of negative instances covered by P (i.e. $R_i : body(P) \subseteq body(R_i)$ and $head(P) \neq head(R_i)$);
- $|D|$: the number of instances in D ;

- Coverage: $coverage(P) = \frac{ncovers(P)}{|D|}$;
- Accuracy: $accuracy(P) = \frac{pos(P)}{ncovers}$;
- Cardinality: $card(P) = |body(P)|$;
- Pessimistic error rate: $PER(P) = \frac{neg(P)+1}{neg(P)+pos(P)+2}$
- p_i is the probability of class c_i in D;
- Expected information: $Info(D) = -\sum_{i=1}^{m} p_i * \log_2(p_i)$;
- Information gain: $InfoGain(D) = Info(D) - \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$;
- FOIL gain (it favors rules that have high accuracy and cover many positive instances):

$$FOILGain(P,P') = pos(P') \times \left(\log_2 \frac{pos(P')}{pos(P')+neg(P')} - \log_2 \frac{pos(P)}{pos(P)+neg(P)} \right);$$

Further measures can be defined but those listed above are the most basic ones.

3.2.3 Pruning

In order to reduce the produced association rules, pruning in parallel with (pre-pruning) or after (post-pruning) creating association rules is performed. Different heuristics for pruning during rule generation are used, mainly based on minimum support, minimum confidence and different kinds of error pruning [Kuncheva, 2004]. In post-pruning phase, criteria such as data coverage (ACRI) or correlation between consequent and antecedent (CMAR) are also used.

During the pruning phase or in classification stage, different ranking criteria for ordering the rules are used. The most common ranking mechanisms are based on the support, confidence and cardinality of the rules, but other techniques such as the cosine measure and coverage measure (ACRI) also exist; we can mention amongst them:

- Pruning by confidence: retain more general rules with higher accuracy: $|R_1| < |R_2|$ and $conf(R_1) < conf(R_2)$, than R_1 is pruned (used in ARC-AC, ARC-BC);

- Pruning by precedence: special kind of ordering using "precedence" (CBA and MCAR);
- Correlation pruning: statistical measuring of the rule significance using weighted χ^2 (CMAR).

3.3 Recognition

In the recognition stage, three different approaches can be discerned [Depaire et al, 2008]:

1. using a single rule.
2. using a subset of rules.
3. using all rules.

An example which uses a single rule is CBA. It classifies an instance by using the single best rule covering the instance.

CPAR uses a subset of rules. It first gathers all rules covering the new instance and selects the best n rules per class. Next, it calculates the average Laplace accuracy per class and predicts the class with the highest average accuracy.

Additionally to support, coverage and confidence, ACRI uses also the cosine measure.

CMAR uses all rules covering a class to calculate an average score per class.

CMAR selects the rule with the highest χ^2 measure from the candidate set.

ARC-AC and ARC-BC use the sum of confidences as score statistics.

A different approach is proposed in TFPC, which suggests to consider the size of the antecedent and to favor long rules before making an allowance for confidence and support.

When a subset or all rules is being used, several order-based combined measures can be applied:

- Select all matching rules;
- Group rules per class value;
- Order rules per class value according to criterion;
- Calculate combined measure for best Z rules;
- Laplace Accuracy (CPAR): if k is the number of class values then $LA = \frac{support(R)+1}{support(body(R))+k}$.

3.4 Some Representatives of CAR Algorithms

In this subsection we present briefly several representatives of the CAR Algorithm.

- ✓ ***CBA***

In CBA [Liu et al, 1998], Apriori is applied to create the association rules.

For measuring the significance of the rules a special "precedence" definition is given:

$P^1 \succ P^2$ (rule P^1 precedes rule P^2) if:

1. $confidence(P^1) > confidence(P^2)$;
2. $confidence(P^1) = confidence(P^2)$ but $support(P^1) > support(P^2)$;
3. $confidence(P^1) = confidence(P^2)$, $support(P^1) = support(P^2)$, but P^1 is generated earlier than P^2.

Pruning is based on the pessimistic error rate based pruning method in C4.5.

Condition 1. Each training case is covered by the rule with the highest precedence among the rules that can cover the case.

Condition 2. Every rule correctly classifies at least one remaining training case when it is chosen.

The key point is instead of making one pass over the remaining data for each rule, the algorithm to find the best rule to cover each case.

During the recognition CBA just searches in the pruned and ordered list for the first rule that covers the instance to be classified. The prediction is the class label of that classification rule. If no rule covers the instance, CBA uses the default class calculated during pruning. If the decision list is empty, the majority class of the training instance will be assigned to each test instance as default.

- ✓ ***CMAR***

CMAR [Li et al, 2001] employs a novel data structure, CR-tree, to compactly store and efficiently retrieve a large number of rules for classification. CR-tree is a prefix tree structure to explore the sharing among rules, which achieves substantial compactness.

In the phase of rule generation, CMAR computes the complete set of rules. CMAR prunes some rule and only selects a subset of high quality rules for classification. CMAR adopts a variant of FP-growth method, which is much faster than Apriori-like methods, especially in the situations where large datasets, low

support threshold, and long patterns exist. The specificity of CMAR is also that it finds frequent pattern and generates rules in one step.

For every pattern, CMAR maintains the distribution of various class labels among data objects matching the pattern. This is done without any overhead in the procedure of counting (conditional) databases. On the other hand, CMAR uses class label distribution to prune. Once a rule is generated, it is stored in a CR-tree.

The number of rules generated by class-association rule mining can be huge. To make the classification effective and also efficient, we need to prune rules to delete redundant and noisy information. According to the facility of rules on classification, a global order of rules is composed. CMAR employs the following methods for rule pruning.

1. Using general and high-confidence rule to prune more specific and lower confidence ones.
2. Selecting only positively correlated rules.
3. Pruning rules based on database cover.

In the phase of classification, for a given data object, CMAR selects a small set of high confidence matching the object, highly related rules and analyzes the correlation among those rules.

- ✓ ***ARC-AC and ARC-BC***

In 2002, Zaïane and Antonie offered new associative classifiers for text categorization – ARC-AC and ARC-BC [Zaïane and Antonie, 2002]. For building association rules they used Apriori-like algorithm. They have considered two different approaches for extracting term-category association rules and for combining those rules to generate a text classifier.

In the first approach ARC-BC (Association Rule-based Categorizer by Category), each category is considered as a separate collection and the association rule mining applied to it. Once the frequent item-sets are discovered, the rules are simply generated by making each frequent item-set the antecedent of the rule and the current category the consequent.

The ARC-AC (Association Rule-based Categorizer for All Categories) considers all categories at whole. In this case one antecedent can be found with different consequents. During the recognition they introduce "dominant factor", which is the proportion of rules of the most dominant category in the applicable rules to the query.

- ✓ ***CPAR***

A greedy associative classification algorithm called CPAR was proposed in [Yin and Han, 2003]. CPAR adopts FOIL [Quinlan and Cameron-Jones, 1993] strategy in generating the rules from datasets. It seeks for the best rule condition that

brings the most gain among the available ones in the dataset. Once the condition is identified, the weights of the positive examples associated with it will be deteriorated by a multiplying factor, and the process will be repeated until all positive examples in the training dataset are covered.

The searching process for the best rule condition is time consuming process for CPAR since the gain for every possible item needs to be calculated in order to determine the best item gain. Thus, CPAR uses an efficient data structure, i.e. PN Array, to store all the necessary information for calculation of the items gain. In the rules generation process, CPAR derives not only the best condition but all close similar ones since there are often more than one attribute items with similar gain.

- ✓ ***CorClass***

CorClass [Zimmermann and De Raedt, 2004] directly finds the best correlated associations rules for classification by employing a branch-and-bound algorithm, using so called Morishita & Sese Framework [Morishita and Sese, 2000]. It follows the strategy in which calculating the upper bounds on the values attainable by specializations of the rule currently considered. The upper bound finally allows dynamic rising of the pruning threshold, differing from the fixed minimal support used in existing techniques. This will result in earlier termination of the mining process. Since the quality criterion for rules is used directly for pruning, no post-processing of the discovered rule set is necessary.

The algorithm uses two strategies for classifying a new object

1. Decision List: Rank all the rules (rules are ranked by quality according to some criterion) and use the first rule satisfied by an example for classification.
2. Weighted Combination: The general way to do this is to collect all such rules, assign each one a specific weight and for each class predicted by at least one rule sum up the weights of corresponding rules. The class value having the highest value is returned.

- ✓ ***ACRI***

The task of ACRI (Associative Classifier with Reoccurring Items) [Rak et al, 2005] is to combine the associative classification with the problem of recurrent items.

A delicate issue with associative classifiers is the use of a subtle parameter: support. Support is a difficult threshold to set, inherited from association rule mining. It is known in the association rule mining field that the support threshold is not obvious to tune in practice. The accuracy of the classifier can be very sensitive to this parameter.

The algorithm for mining associations in ACRI is based on earlier work of the authors Apriori-based MaxOccur [Zaiane et al, 2000]. The building of the

classification model follows their previous ARC-BC approach. The rational is based on the efficiency of this method in the case of non-evenly distributed class labels. MaxOccur run on transactions from each known class separately makes the core of the rule generator module. It mines the set of rules with reoccurring items from the training set.

These rules associate a condition set with a class label such that the condition set may contain items preceded by a repetition counter. The classification process might be considered as plain matching of the rules in the model to the features of an object to classify. Different classification rules may match, thus the classifier module applies diverse strategies to select the appropriate rules to use.

In addition, simple matching is sometimes not possible because there is no rule that has the antecedent contained in the feature set extracted from the object to classify. With other associative classifiers, a default rule is applied, either the rule with the highest confidence in the model or simply assigning the label of the dominant class. The ACRI approach has a different strategy allowing partial matching or closest matching by modeling antecedents of rules and new objects in a vector space.

- ✓ ***TFPC***

TFPC (Total From Partial Classification) [Coenen and Leng, 2005] is a classification association rule mining algorithm founded on the TFP (Total From Partial) association rule mining algorithm; which, in turn, is an extension of the Apriori-T (Apriori Total).

TFP (Total From Partial) algorithm builds a set enumeration tree structure, the P-tree, that contains an incomplete summation of support-counts for relevant sets. Using the P-tree, the algorithm uses an Apriori-like procedure to build a second set enumeration tree, the T-tree, that finally contains all the frequent sets (i.e. those that meet the required threshold of support), with their support-counts. The T-tree is built level by level, the first level comprising all the single items (attribute-values) under consideration. In the first pass, the support of these items is counted, and any that fail to meet the required support threshold are removed from the tree. Candidate-pairs are then generated from remaining items, and appended as child nodes. The process continues, as with Apriori, until no more candidate sets can be generated.

The class-competition is solved by using support and confidence measures.

- ✓ ***HARMONY***

HARMONY [Wang and Karypis, 2005] directly mines for each training instance one of the highest confidence classification rules that it supports and satisfies a user specified minimum support constraint, and builds the classification model from the union of these rules over the entire set of instances. Thus HARMONY

employs an instance-centric rule generation framework and mines the covering rules with the highest confidence for each instance, which can achieve better accuracy. Moreover, since each training instance usually supports many of the discovered rules, the overall classifier can better generalize to new instances and thus achieve better classification performance.

To achieve high computational efficiency, HARMONY mines the classification rules for all the classes simultaneously and directly mines the final set of classification rules by pushing deeply some effective pruning methods into the projection-based frequent item-set mining framework. All these pruning methods preserve the completeness of the resulting rule-set in the sense that they only remove from consideration rules that are guaranteed not to be of high quality.

✓ ***MCAR***

MCAR (Multi-class Classification based on Association Rule) [Thabtah et al, 2005] uses an efficient technique for discovering frequent items and employs a rule ranking method which ensures detailed rules with high confidence.

During the rules generation MCAR scans the training dataset to discover frequent single items, and then recursively combines the items generated to produce items involving more attributes. After that the rules are used to generate a classifier by considering their effectiveness on the training dataset, using expanded definition of "precedence":

$P^1 \succ P^2$ (rule P^1 precedes rule P^2) if:

1. $confidence(P^1) > confidence(P^2)$;
2. $confidence(P^1) = confidence(P^2)$ but $support(P^1) > support(P^2)$;
3. $confidence(P^1) = confidence(P^2)$, $support(P^1) = support(P^2)$, but $ActAcc(P^1) = ActAcc(P^2)$;
4. All conditions before are the same, but $card(P^1) < card(P^2)$;
5. Last condition: P^1 is generated earlier than P^2.

✓ ***CACA***

The following innovations are integrated in CACA [Tang and Liao, 2007]:

- use the class-based strategy to cut down the searching space of frequent pattern;
- design a structure call Ordered Rule Tree (OR-Tree) to store the rules and their information which may also prepare for the synchronization of the two steps;

- redefine the compact set so that the compact classifier is unique and not sensitive to the rule reduction;
- synchronize the rule generation and building classifier phases.

Class-based strategy: Given a training dataset D with k classes, the principle idea of class based rule mining is to divide the single attribute value set C_{all} for all classes into k smaller ones for every class, that is, to limit the searching in k low dimensional spaces other than a high dimensional one.

OR-Tree: To facilitate they design a structure, called Ordered-Rule-Tree (OR-Tree), under the inspiration of CR-Tree used in CMAR to store and rank rules. It is composed with a tree structure and an ordered list. When a rule $P^i = (c^i \mid a_1^i,...,a_n^i)$ satisfying the support and confidence thresholds is generated, attribute values $a_1^i,...,a_n^i$ are stored as nodes in this tree according to their frequency in D in descending order. The last node points to an information node storing the rule's information such as class label, support and confidence. Each rule can and only can have one information node. The ordered list is designed to organize all rules in the tree. Each node in the chain points to a certain rule. Nodes pointing to the rules with higher priority are closer to the head node, while those pointing to the rules with lower priority are farther from the head node.

The ranking rule criteria are as follows:

$P^1 \succ P^2$ (P^1 precedes P^2) if:

1. $confidence(P^1) > confidence(P^2)$;
2. $confidence(P^1) = confidence(P^2)$ but $support(P^1) > support(P^2)$;
3. $confidence(P^1) = confidence(P^2)$ and $support(P^1) = support(P^2)$ but $card(P^1) < card(P^2)$ (P^1 is more general than P^2);
4. Equal previous conditions, but P^1 is generated earlier then P^2.

To ensure compact classifier to be unique and not sensitive to the rule reduction, the redundant rules are defined as follows:

Definition of redundant rule: Given P^1, P^2 and P^3, that belong to rule set R, P^2 is redundant if:

- $P^1 = (c^1 \mid a_1^1,...,a_{k1}^1)$, $P^2 = (c^2 \mid a_1^2,...,a_{k2}^2)$:
  $c^1 \neq c^2$, $(a_1^1,...,a_{k1}^1) \subseteq (a_1^2,...,a_{k2}^2)$, $P^1 \succ P^2$;
- $P^1 = (c^1 \mid a_1^1,...,a_{k1}^1)$, $P^2 = (c^2 \mid a_1^2,...,a_{k2}^2)$:
  $c^1 = c^2$, $(a_1^1,...,a_{k1}^1) \subset (a_1^2,...,a_{k2}^2)$, $P^1 \succ P^2$;

– $P^1 = (c^1 \mid a_1^1,...,a_{k1}^1)$, $P^2 = (c^2 \mid a_1^2,...,a_{k2}^2)$, $P^3 = (c^3 \mid a_1^3,...,a_{k3}^3)$:
$c^1 = c^2 \neq c^3$, $(a_1^1,...,a_{k1}^1) \subset (a_1^2,...,a_{k2}^2)$, $(a_1^1,...,a_{k1}^1) \subset (a_1^3,...,a_{k3}^3)$,
$P^1 \succ P^2 \succ P^3$.

Definition of compact rule set: For rule set R, if $\mathrm{R}' \subset \mathrm{R}$, any redundant rule $P \notin \mathrm{R}'$ and R' is unique, then R' is the compact set of R.

CACA technically combined the rule generation and the building classifier phases together. Once a new rule is generated, the algorithm visits the OR-Tree partially to recognize its redundancy, stores it in the OR-Tree and ranks it in the rule set. Not only can the synchronization simplify the procedure of associative classification but also apply the pruning skill to shrink the rule mining space and raise the efficiency.

✓ ***ARUBAS***

In contrast with many existing associative classifiers, ARUBAS [Depaire et al, 2008] uses class association rules to transform the feature space and uses instance-based reasoning to classify new instances. The framework allows the researcher to use any association rule mining algorithm to produce the class association rules. Five different fitness measures are used for classification purposes.

The main idea behind the ARUBAS framework, is transformation of the original feature space into a more powerful feature space. The original feature space is called the attribute space, where each record $R^i = (c^i \mid a_1^i,...,a_n^i)$ is coded as a set of attribute values and a class value.

In attribute space, each dimension consists of a single attribute. In the new feature space, which we will call pattern space, each dimension will consist of a combination of attributes, also called a pattern, which is denoted as $P_p = \langle (A_{i1}, a_{i1}),...,(A_{ik}, a_{ik}) \rangle$. For achieving more power for the feature space, only combinations of attributes (or patterns) which are strongly associated with a single class value is given.

The first step in the ARUBAS framework is to use any CAR mining technique to find a set of CARs, which is used to transform the feature space. The antecedent of each CAR, which represents an item-set, will become a pattern P_p and hence a dimension in the new feature space. The value of an instance R^i for a pattern dimension P_p is 1 (if the instance contains the pattern) or 0 (if it doesn't).

The instance similarity is used for classifying new instances. To measure the similarity between a new instance R^i and a known training instance R^t ARUBAS focuses on the patterns contained by both instances and how many patterns

both instances have in common, but on those patterns coming from the CARs which predicted the class value of the training instance R^t.

The main idea behind the association rule based similarity framework is that classification is based on similarity between a new instance and an entire class. This similarity is not measured in the original attribute space, but in the pattern space, which is constructed by means of CARs.

Conclusion

This chapter provided an overview of the area of CAR-classifiers. CAR algorithms have its important place in the family of classification algorithms with several advantages, such as: efficiency of the training regardless of the training set; easy handling with high dimensionality; very fast classification; high accuracy; human comprehensible classification model.

We observed all typical steps in the whole classification process of CAR algorithms: generating the rules, pruning, and recognition.

In the phase of generating the rules several techniques are observed: the pioneer AIS, most used Apriori, alternative FP-Tree, TreeProjection, Matrix Algorithm, Sampling Algorithms, Partition, FOIL and Morishita & Sese Framework.

The pruning is important step in the learning process of CAR algorithms, applied as preprocessing step, in parallel of association rule mining or after it. Here we made a brief observation of several rule quality measures and rule ordering schemes, used in CAR algorithms.

In the recognition phase we also observed different types of choosing final decision – using simple rule or set of rules with different types of ordering schemas.

Finally, using the proposed framework, typical for CAR algorithms, we analyze the some representatives of CAR algorithms: CBA, CMAR, ARC-AC and ARC-BC, CPAR, CorClass, ACRI, TFPC, HARMONY, MCAR, CACA, ARUBAS, showing wide variety of proposed techniques.

4 Multi-Dimensional Numbered Information Spaces

Abstract

This chapter presents the advance of different types of access methods developed in the last years and used in data mining processes in order to facilitate access to the different kinds of structures.

A special attention is paid to a memory organization, called "Multi-dimensional numbered information spaces" which allows to operate with context-free multidimensional data structures.

The software implementation of such structure is named Multi-Domain Access Method ArM 32. The implementation of the memory organization and available functional operations are presented.

The purpose is to use such structures and operations in the implementation of one class association rule classifier in order to show the vitality of the idea of using context-free multidimensional data structures and direct access as a powerful tool for knowledge discovery.

4.1 Memory Management

Memory management is a complex field of computer science. Over the years, many techniques have been developed to make it more efficient [Ravenbrook, 2010]. Memory management usually addresses three areas: hardware, operating system, and application, although the distinctions are a little fuzzy. In most computer systems, all three are present to some extent, forming layers between the user's program and the actual memory hardware:

- *memory management at the hardware level* is concerned with the electronic devices that actually store data. This includes the use of RAM and memory caches;

- *memory in the operating system* must be allocated to user programs, and reused by other programs when it is no longer required. The operating system can pretend that the computer has more memory than it actually does, and that each program has the machine's memory to itself. Both of these are features of virtual memory systems;
- *application memory management* involves supplying the memory needed for a program's objects and data structures from the limited resources available, and recycling that memory for reuse when it is no longer required. Because in general, application programs cannot predict in advance how much memory they are going to require, they need additional code to handle their changing memory requirements.

Application memory management combines two related tasks:

- *allocation:* when the program requests a block of memory, the memory manager must allocate that block out of the larger blocks it has received from the operating system. The part of the memory manager that does this is known as the allocator;
- *recycling:* when memory blocks have been allocated, but the data they contain is no longer required by the program, the blocks can be recycled for reuse. There are two approaches to recycling memory: either the programmer must decide when memory can be reused (known as manual memory management); or the memory manager must be able to work it out (known as automatic memory management).

The progress in memory management gives the possibility to allocate and recycle not directly blocks of the memory but structured regions or fields corresponding to some types of data. In such case we talk about corresponded "*access methods*".

4.2 Access Methods

Access Methods (AM) have been available from the beginning of the development of computer peripheral devices. There are multiple possibilities for developing different AM. In the beginning, the AM were functions of the Operational Systems Core or so called Supervisor, and were executed via corresponding macro-commands in the assembler languages [Stably, 1970] or via corresponding input/output operators in the high level programming languages like FORTRAN, COBOL, PL/I, etc.

The establishment of the first databases in the sixties of the previous century caused gradually accepting the concepts "physical" as well as "logical" organization of the data [CODASYL, 1971], [Martin, 1975]. In 1975, the concepts "access method", "physical organization" and "logical organization" became clearly separated.

In the same time, Christopher Date noted:

"The Database Management System (DBMS) does not know anything about:

- how physical records (blocks) are disposed;
- how the stored fields are integrated in the records (nevertheless that in many cases it is obvious because of their physical disposition);
- how the sorting is realized (for instance it may be realized on the base of physical sequence, using an index or by a chain of pointers);
- how the direct access is realized (i.e. by index, sequential scanning or hash addressing).

This information is a part of the structures for data storing but it is used by the access method but not by the DBMS" [Date, 1975].

Every access method presumes an exact organization of the file, which it is operating with and is not related to the interconnections between the files, respectively – between the records of one file and that in the others files. These interconnections are controlled by the physical organization of the DBMS.

Therefore, in the DBMS we may distinguish four levels:

1. Access methods at the core (supervisor) of the operation system.
2. Specialized access methods which upgrade these at the core of the operating system.
3. Physical organization of the DBMS.
4. Logical organization of the DBMS.

During the 80s, the overall progress in research and developments in the information technologies, and more specifically in image processing, data mining and mobile support boosted impetuous progress of designing convenient "spatial information structures" and "spatial-temporal information structures" and corresponding access methods. From different points of view, this period has been presented in [Ooi et al, 1993], [Gaede and Günther, 1998], [Arge, 2002], [Mokbel et al, 2003], [Moënne-Loccoz, 2005]. Usually the "one-dimensional" (linear) AM are used in the classical applications, based on the alphanumerical information, whereas the "multi-dimensional" (spatial) methods are aimed to serve the work with graphical, visual, multimedia information.

4.2.1 Interconnections between Raised Access Methods

One of the most popular analyses is given in [Gaede and Günther, 1998]. The authors presented a scheme of the genesis of the basic multi-dimensional AM and theirs modifications. This scheme firstly was proposed in [Ooi et al, 1993] and it was expanded in [Gaede and Günther, 1998]. An extension in direction to the multi-dimensional spatio-temporal access methods was given in [Mokbel et al, 2003].

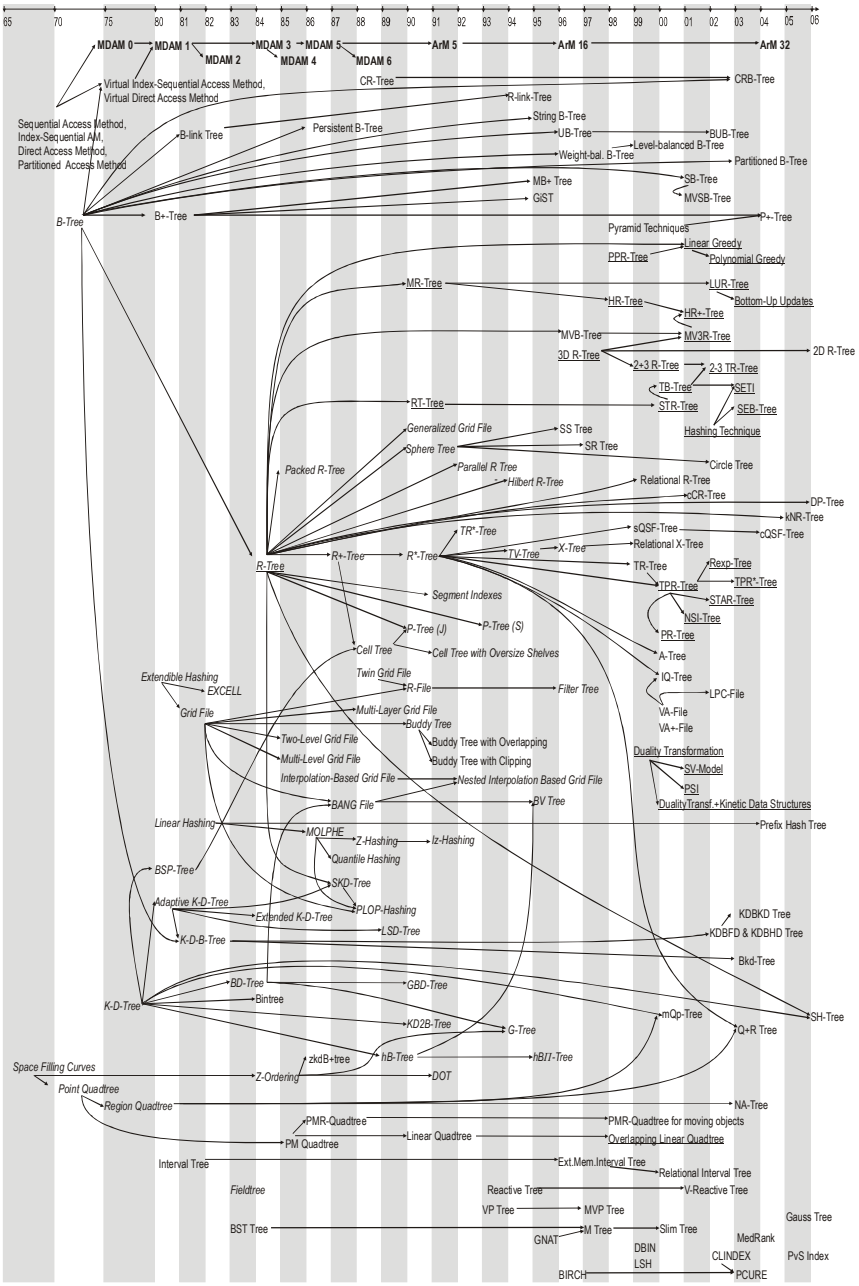


Figure 3. Genesis of the Access Methods and their modifications extended variant of [Gaede and Günther, 1998] and [Mokbel et al, 2003] presented in [Markov et al, 2008]

The survey [Markov et al, 2008] presents a new variant of this scheme (Figure 3), where the new access methods, created after 1998, are added. A comprehensive bibliography of corresponded articles, where the methods are firstly presented is given.

4.2.2 The Taxonomy of the Access Methods

From the point of view of the served area, the access methods, presented on Figure 3, can be classified as follows (Figure 4): One-dimensional AM; Multidimensional Spatial AM; Metric Access Methods; High Dimensional Access Methods; and Spatio-Temporal Access Methods.

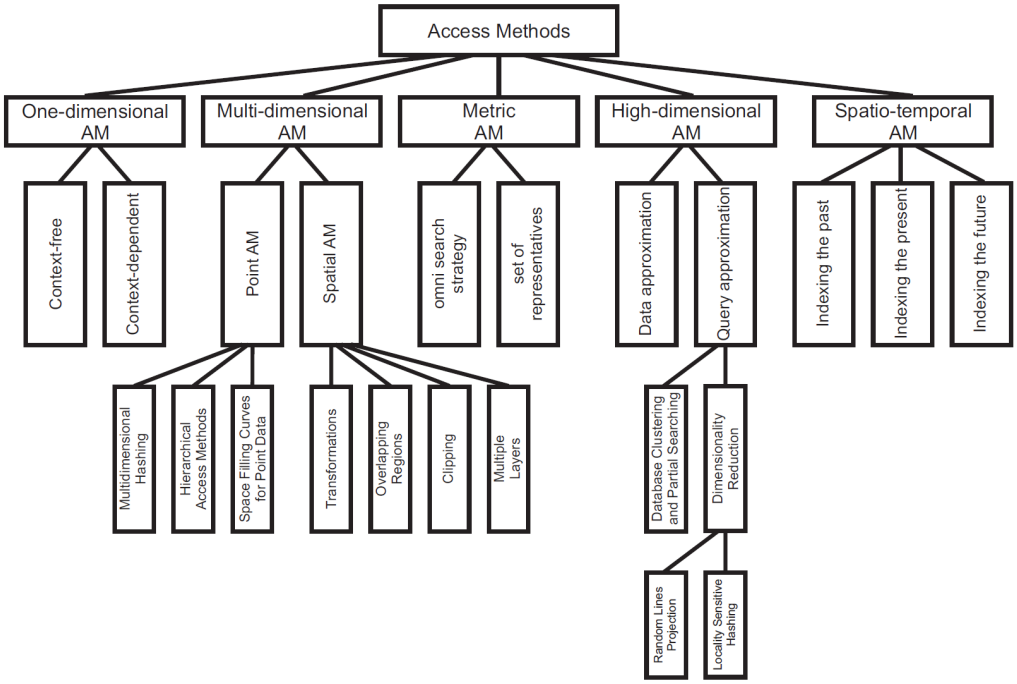


Figure 4. Taxonomy of the access methods

- ***One-Dimensional Access Methods***

One-dimensional AM are based on the concept "record". The "record" is a logical sequence of fields, which contain data eventually connected to unique identifier (a "key"). The identifier (key) is aimed to distinguish one sequence from another [Stably, 1970]. The records are united in the sets, called "files".

There exist three basic formats of the records – with fixed, variable, and undefined length.

In the *context-free AM,* the storing of the records is not connected to their content and depends only on external factors – the sequence, disk address, or position in the file. The necessity of stable file systems in the operating systems does not allow a great variety of the context-free AM. There are three main types well known from sixties and seventies: *Sequential Access Method; Direct Access Method and Partitioned Access Method* [IBM, 1965-68].

The main idea of the *context-depended AM* is that a part of the record is selected as a key, which is used for making decision where to store the record and how to search it. This way the content of the record influences the access to the record.

Historically, from the sixties of the previous century on, the majority of research and development is directed mainly to this type of AM. Modern DBMS are built using context-depended AM such as: unsorted sequential files with records with keys; sorted files with fixed record length; static or dynamic hash files; index file and files with data; clustered indexed tables [Connolly and Begg, 2002].

- ***Multidimensional Spatial Access Methods***

Multidimensional Spatial Access Methods are developed to serve information about spatial objects, approximated with points, segments, polygons, polyhedrons, etc. The implementations are numerous and include traditional multi-attributive indexing, geographical and/or GMES information systems, and spatial databases, content indexing in multimedia databases, etc.

From the point of view of the spatial databases, access methods can be split into two main classes of access methods – Point Access Methods and Spatial Access Methods [Gaede and Günther, 1998].

Point Access Methods are used for organizing multidimensional point objects. Typical instances are traditional records, where every attribute of the relation corresponds to one dimension. These methods can be separated in three basic groups:

- *Multidimensional Hashing* (for instance Grid File and its varieties, EXCELL, Twin Grid File, MOLPHE, Quantile Hashing, PLOP-Hashing, Z-Hashing, etc);
- *Hierarchical Access Methods* (includes such methods as KDB-Tree, LSD-Tree, Buddy Tree, BANG File, G-Tree, hB-Tree, BV-Tree, etc.);
- *Space Filling Curves for Point Data* (like Peano curve, N-trees, Z-Ordering, etc).

Spatial Access Methods are used for work with objects which have an arbitrary form. The main idea of the spatial indexing of non-point objects is to use an approximation of the geometry of the examined objects as more simple

forms. The most used approximation is Minimum Bounding Rectangle (MBR), i.e. minimal rectangle, which sides are parallel of the coordinate axes and completely include the object. There exist approaches for approximation with Minimum Bounding Spheres (SS Tree) or other polytopes (Cell Tree), as well as their combinations (SR-Tree).

The common problem in operating with spatial objects is their overlapping. There are different techniques to avoid this problem. From the point of view of the techniques for the organization of the spatial objects, Spatial Access Methods fall into four main groups:

- *Transformation:* this technique uses transformation of spatial objects to points in the space with more or less dimensions. Most of them spread out the space using space filling curves (Peano Curves, z-ordering, Hibert curves, Gray ordering, etc.) and then use some point access method upon the transformed dataset;
- *Overlapping Regions:* here the datasets are separated in groups; different groups can occupy the same part of the space, but every space object associates with only one of the groups. The access methods of this category operate with data in their primary space (without any transformations) eventually in overlapping segments. Methods which use this technique include R-Tree, R-link-Tree, Hilbert R-Tree, R*-Tree, Sphere Tree, SS-Tree, SR-Tree, TV-Tree, X-Tree, P-Tree of Schiwietz, SKD-Tree, GBD-Tree, Buddy Tree with overlapping, PLOP-Hashing, etc.;
- *Clipping:* this technique uses the clipping of one object to several sub-objects, which will be stored. The main goal is to escape overlapping regions. However, this advantage can lead to the tearing of the objects, extending the resource expenses, and decreasing the productivity of the method. Representatives of this technique are R+-Tree, Cell-Tree, Extended KD-Tree, Quad-Tree, etc.;
- *Multiple Layers:* this technique can be considered as a variant of the techniques of Overlapping Regions, because the regions from different layers can overlap. Nevertheless, there exist some important differences: first – the layers are organized hierarchically; second – every layer splits the primary space in a different way; third – the regions of one layer never overlaps; fourth – the data regions are separated from the space extensions of the objects. Instances for these methods are Multi-Layer Grid File, R-File, etc.

➢ *Metric Access Methods*

Metric Access Methods deal with relative distances of data points to chosen points, named anchor points, vantage points or pivots [Moënne-Loccoz, 2005]. These methods are designed to limit the number of distance computation, calculating first distances to anchors, and then finding the searched point in a narrowed region. These methods are preferred when the distance is highly

computational, as e.g. for the dynamic time warping distance between time series. Representatives of these methods are: Vantage Point Tree (VP Tree), Bisector Tree (BST-Tree), Geometric Near-Neighbor Access Tree (GNNAT), as well as the most effective from this group – Metric Tree (M-Tree) [Chavez et al, 2001].

- ***High Dimensional Access Methods***

Increasing the dimensionality strongly aggravates the qualities of the multidimensional access methods. Usually these methods exhaust their possibilities at dimensions around 15. Only X-Tree reaches the boundary of 25 dimensions, after which this method gives worse results then sequential scanning [Chakrabarti, 2001].

A possible solution is based on the data approximation and query approximation in sequential scan. These methods form a new group of access methods – High Dimensional Access Methods.

Data approximation is used in VA-File, VA+-File, LPC-File, IQ-Tree, A-Tree, P+-Tree, etc.

For query approximation, two strategies can be used:

- *examine only a part of the database*, which is more probably to contain the resulting set – as a rule these methods are based on the clustering of the database. Some of these methods are: DBIN, CLINDEX, PCURE;
- *splitting the database to several spaces* with fewer dimensions and searching in each of them. Here two main methods are used:
  1. *Random Lines Projection:* representatives of this approach are MedRank, which uses B+-Tree for indexing every arbitrary projection of the database, and PvS Index, which consist of combination of iterative projections and clustering.
  2. *Locality Sensitive Hashing:* based on the set of local-sensitive hashing functions [Moënne-Loccoz, 2005].

- ***Spatio-Temporal Access Methods***

The Spatio-Temporal Access Methods have additional defined time dimension [Mokbel et al, 2003]. They operate with objects which change their form and/or position across time. According to position of time interval in relation to present moment, the Spatio-Temporal Access Methods are divided to:

- *indexing the past:* these methods operate with historical spatio-temporal data. The problem here is the continuous increase of the information over time. To overcome the overflow of the data space two approaches are used – sampling the stream data at certain time position or updating the information only when data is changed. Representatives of this group are: RT-Tree, 3DR-Tree, STR-Tree, MR-Tree, HR-Tree, HR+-Tree,

MV3R-Tree, PPR-Tree, TB-Tree, SETI, SEB-Tree. Spatio-temporal indexing schemes for historical data can be split in three categories:

- the first category includes methods which are integrating both spatial and temporal aspects into already existing spatial methods;
- the second category can be described as using snapshots of the spatial information in each time instance;
- the third category focuses on trajectory-oriented queries, while spatial dimension remains in the background.

– *indexing the present:* in contrast to previous methods, where all movements are known, here the current positions are neither stored nor queried. Some of the methods, which answer the questions of the current position of the objects are 2+3R-Tree, 2-3TR-Tree, LUR-Tree, Bottom-Up Updates, etc.;

– *indexing the future:* these methods have to represent the current and predict the future position of a moving object – here are embraced the methods like PMR-Quadtree for moving objects, Duality Transformation, SV-Model, PSI, PR-Tree, TPR-Tree, TPR*-tree, NSI, VCIR-Tree, STAR-Tree, REXP-Tree.

The survey of the access methods suggests that the context-free multi-dimensional access methods practically are not available. A step in developing such methods is the Multi-domain Information Model and corresponding Multi-domain Access Method introduced in [Markov, 1984] [Markov, 2004]. It is presented further in this chapter.

4.3 Multi-Dimensional Numbered Information Spaces

The proposed external memory structure is based on the numbering as a main approach. The idea consists in replacing the (symbol or real; point or interval) values of the objects' attributes with integer numbers of the elements of corresponding ordered sets. This way, each object will be described by a vector of integer values, which may be used as the co-ordinate address in the multi-dimensional information space.

In other words, the process of replacing the names by numbers permits the use of mathematical functions and address vectors for accessing the information instead of search engines.

This type of memory organization is called "Multi-dimensional numbered information spaces". Its advantages have been demonstrated in multiple practical implementations during more than twenty-five years [Markov, 1984], [Markov, 2004], [Markov, 2005]. In the recent years, it had been implemented in the area of intelligent systems memory structuring for several data mining tasks and especially in the area of association rules mining.

4.3.1 Multi-Domain Information Model (MDIM)

The independence of dimensionality limitations is very important for developing new intelligent systems aimed to process high-dimensional data. To achieve this one needs information models and corresponding access method to cross the boundary of the dimensional limitations and to obtain the possibility to work with information spaces with practically unlimited number of dimensions. The first step is to establish context free multi-dimensional models and based on it to develop high-level context depended applications. Examining the state of the art in this area shows that the context-free multi-dimensional information models and access methods practically are not available. One attempt in this direction is establishing the *Multi-Domain Information Model (MDIM)* [Markov, 2004] and the corresponding Multi-domain Access Method. Their possibilities for operating with context-free multidimensional data structures will be presented below.

- ***Basic Structures of MDIM***

Basic structures of MDIM are basic information elements, information spaces, indexes and metaindexes, and aggregates. The definitions of these structures are given below:

- ✓ ***Basic Information Element***

The basic information element (BIE) of MDIM is an arbitrary long string of machine codes (bytes). When it is necessary, the string may be parceled out by lines. The length of the lines may be variable.

Let the universal set $UBIE$ be the set of all BIE .

Let E_1 be a set of basic information elements:

$$E_1 = \{e_i \mid e_i \in UBIE, i = 1,...,m_1\} .$$

Let μ_1 be a function, which defines a biunique correspondence between elements of the set E_1 and elements of the set C_1 of positive integer numbers:

$$C_1 = \{c_i \mid c_i \in \mathbf{N}, i = 1,...,m_1\} ,$$

$$\text{i.e. } \mu_1 : E_1 \leftrightarrow C_1 .$$

The elements of C_1 are said to be numbers (co-ordinates) of the elements of E_1 .

✓ ***Information Spaces***

The triple $S_1 = (E_1, \mu_1, C_1)$ is said to be a ***numbered information space of range 1*** *(one-dimensional or one-domain information space)****.***

The triple $S_2 = (E_2, \mu_2, C_2)$ is said to be a ***numbered information space of range 2*** iff E_2 is a set whose elements are numbered information spaces of range 1 and μ_2 is a function which defines a biunique correspondence between elements of E_2 and elements of the set C_2 of positive integer numbers:

$$C_2 = \{c_j \mid c_j \in \mathbf{N}, j = 1,...,m_2\},$$

$$\text{i.e. } \mu_2 : E_2 \leftrightarrow C_2 .$$

The triple $S_n = (E_n, \mu_n, C_n)$ is said to be a ***numbered information space of range n*** *(n-dimensional or multi-domain information space)* iff E_n is a set whose elements are information spaces of range $n-1$ and μ_n is a function, which defines a biunique correspondence between elements of E_n and elements of the set C_n of positive integer numbers:

$$C_n = \{c_k \mid c_k \in \mathbf{N}, k = 1,...,m_n\},$$

$$\text{i.e. } \mu_n : E_n \leftrightarrow C_n .$$

Every basic information element "e" is considered as an ***information space*** S_0 of range 0. It is clear that the information space $S_0 = (E_0, \mu_0, C_0)$, is constructed in the same manner as all others:

- the machine codes (bytes) $b_i, i = 1,...,m_0$ are considered as elements of E_0;
- the position $p_i, i \in \mathbf{N}$ of b_i in the string e is considered as co-ordinate of b_i, i.e. $C_0 = \{p_l \mid p_l \in \mathbf{N}, l = 1,...,m_0\}$;
- function μ_0 is defined by the physical order of bi in e and we have:

$$\mu_0 : E_0 \leftrightarrow C_0$$

In this way, the string S_0 may be considered as a set of ***sub-elements (sub-strings)***. The number and length of the sub-elements may be variable. This option is very helpful but it closely depends on the concrete realizations and it is not considered as a standard characteristic of MDIM.

The information space S_n, which contains all information spaces of a given application is called the ***information base*** of range n. Usually, the concept

information base without indication of the range is used as generalized concept to denote all available information spaces.

✓ *Indexes and Metaindexes*

The sequence $A=(c_n,c_{n-1},...,c_1)$ where $c_i \in C_i, i=1,...,n$ is called the ***multidimensional space address*** of range n of a basic information element. Every space address of range m, $m<n$, can be extended to space address of range n by adding leading $n-m$ zero codes. Every sequence of space addresses $A_1, A_2, ..., A_k$, where k is arbitrary positive number, is said to be a ***space index***.

A special kind of space index is the ***projection***, which is the analytically given space index. There are two types of projections:

- hierarchical projection – where the top part of coordinates is fixed and the low part vary for all possible values of coordinates, where non-empty elements exist;
- arbitrary projection – in this case, it is possible to fix coordinates in arbitrary positions and the other coordinates vary for all possible values of coordinates, where non-empty elements exist.

Every index may be considered as a basic information element, i.e. as a string, and may be stored in a point of any information space. In such case, it will have a multidimensional space address, which may be pointed in the other indexes, and, this way, we may build a hierarchy of indexes. Therefore, every index, which points only to indexes, is called ***meta-index***.

The approach of representing the interconnections between elements of the information spaces using (hierarchies) of meta-indexes is called ***polyindexation***.

✓ *Aggregates*

Let $G=\{S_i \mid i=1,...,m\}$ be a set of numbered information spaces.

Let $\tau=\{\nu_{ij}: S_i \to S_j \mid i=const, j=1,...,m\}$ be a set of mappings of one "main" numbered information space $S_i \subset G, i=const$, into the others $S_j \subset G, j=1,...,m$, and, in particular, into itself.

The couple: $D=(G,\tau)$ is said to be an "***aggregate***".

It is clear that we can build m aggregates using the set G because every information space $S_j \subset G, j=1,...,m$, may be chosen as a main information space.

➢ *Operations in the MDIM*

After defining the information structures, we need to present the operations, which are admissible in the model.

It is clear that the operations are closely connected to the defined structures.

In MDIM, we assume that **all** information elements of **all** information spaces **exist**. If for any $S_i : E_i = \varnothing \wedge C_i = \varnothing$, than it is called ***empty.*** Usually, most of the information elements and spaces are empty. This is very important for practical implementations.

✓ *Operation with Basic Information Elements*

Because of the rule that all the structures given above must exist, we only need two operations: (1) updating and (2) getting the value of BIE.

For both types of operations, we need two service operations: (1) getting the length and (2) the positioning in the BIE.

Updating, or simply – writing the element, has several modifications with obvious meaning: writing a BIE as a whole, appending/inserting in a BIE, cutting/replacing a part of a BIE and deleting a BIE.

There is only one operation for getting the value of a BIE, i.e. **Read** a portion from a BIE starting from given position. We may receive the whole BIE if the starting position is the beginning of BIE and the length of the portion is equal to the BIE length.

✓ *Operation with Spaces*

With a **single space,** we may do only one operation – clearing (deleting) the space, i.e. replacing all BIE of the space with empty BIE – ∅. After this operation, all BIE of the space will have zero length. Really, the space is cleared via replacing it with empty space.

With **two spaces**, we may provide two operations with two modifications both: (1) copying and (2) moving the first space in the second.

The modifications define how the BIE in the recipient space are processed. We may have: copy/move with clear and copy/move with merge.

The "clear" modifications first clear the recipient space and after that provide a copy or move operation.

The merge modifications offer two types of processing: destructive or constructive. The ***destructive merging*** may be "conservative" or "alternative". In the conservative approach, the recipient space BIE remains in the result if it is with none zero length. In the other approach the donor space BIE remains in the result. In the ***constructive merging*** the result is any composition of the corresponding BIE of the two spaces.

Of course, the move operation deletes the donor space after the operation.

✓ *Operation with Indexes and Metaindexes*

The indexes are the main approach for describing the interconnections between the structures.

We may receive the space address of the **next** or **previous, empty** or **non-empty** elements of the space starting from any given co-ordinate. This corresponds to the processing of given hierarchical projections.

By analogy, we may receive the space address of the **nextproj** or **previousproj non-empty** elements of the space for the current address in operation with a given arbitrary projection.

The possibility to count the number of non-empty elements of a given projection is useful for practical realizations.

The operations with indexes are based on usual logical operations between sets. The difference from usual sets is that the information spaces are built by the interconnection between two main sets: the set of co-ordinates and the set of information elements.

This way the operations with indexes may be classified in two main types: context-free and context-depended operations.

The context-free operations defined in the MDIM are based on the classical logical operations – intersection, union, and supplement, but these operations are not trivial. Because of the complexity of the structure of the information spaces, these operations have at least two principally different realizations based on:

- co-ordinates;
- information elements.

The operations based on co-ordinates are determined by the existence of the corresponding space information elements. Therefore, the values of the co-ordinates of the existing information elements determine the operations.

In the other case, the existing BIE values determine the logical operations.

In both cases, the result of the logical operations is an index.

The context-dependent operations need special implementations for concrete purposes.

The main information operation is creating the indexes and meta-indexes. The main purpose of the MDIM is to provide the possibility for access to the practically unlimited information space and easy approach for building interconnection between its elements. The goal of the concrete applications is to build tools for creating and operating with the indexes and meta-indexes and to implement these tools in the realization of user requested systems.

For instance, such tools may realize the transfer from one structure to another, information search, sorting, making reports, more complicated information processing, etc. The information operations can be grouped into four

sets according to the main information structures involved: basic information elements, information spaces and index or meta-index structures.

- ✓ ***Operations with Aggregates***

Theory of aggregates may be assumed as an extension of the Relation theory because the relation in the sense of the model of Codd [Codd, 1970] may be represented by the aggregate. It is easy to see that if the aggregation mappings are one-one mappings it will be relation in the sense of the model of Codd. So, we may say that the aggregate is a more universal structure than the relation and the operations with aggregates include those of relation theory. The relation algebra is a very good starting point to understand the algebra of aggregates. The new element is that the mappings of different aggregates may be not one-one mappings. This field is not investigated until now.

4.3.2 Multi-Domain Access Method ArM 32

The program realization of MDIM is called Multi-Domain Access Method. For a long period, it has been used as a basis for the organization of various information bases. There exist several realizations of MDIM for different hardware and/or software platforms. The most resent one is the FOI Archive Manager – ArM. One of the first goals of the development of ArM was to represent the digitalized military defense situation, which is characterized by a variety of complex objects and events, which occur in the space and time and have a long period of variable existence. The high number of layers, aspects, and interconnections of the real situation may be represented only by information space hierarchy. In addition, the different types of users with individual access rights and needs insist on the realization of a special tool for organizing such information base. Over the years, the efficiency of ArM is proved in wide areas of information service of enterprise managements and accounting. Organizing the datum in appropriate multi-dimensional information space model permits omitting the heavy work of creating of OLAP structures [Markov, 2005].

The newest ArM Version No.9, called ArM32, is developed for MS Windows and realizes the proposed algorithms.

The ArM32 elements are organized in numbered information spaces with variable ranges. There is no limit for the ranges of the spaces. Every element may be accessed by a corresponding multidimensional space address (coordinates) given via coordinate array of type cardinal. At the first place of this array, the space range needs to be given. Therefore, we have two main constructs of the physical organizations of ArM32 – numbered information spaces and elements.

In ArM32, the length of the string may vary from 0 up to 1G bytes. There is no limit for the number of strings in an archive but their total length plus internal indexes could not exceed 4G bytes in a single file.

The main ArM32 operations with basic information elements are:

- *ArmRead* (reading a part or a whole element);
- *ArmWrite* (writing a part or a whole element);
- *ArmAppend* (appending a string to an element);
- *ArmInsert* (inserting a string into an element);
- *ArmCut* (removing a part of an element);
- *ArmReplace* (replacing a part of an element);
- *ArmDelete* (deleting an element);
- *ArmLength* (returning the length of the element in bytes).

The operations over the spaces are:

- *DelSpace* (deleting the space);
- *CopySpace* and *MoveSpace* (copying/moving the first space in the second in the frame of one file);
- *ExportSpace* (copying one space from one file the other space, which is located in other file).

The operations, aimed to serve the hierarchical projections are:

- *ArmNextPresent* and *ArmPrevPresent* (traversing of existing elements);
- *ArmNextEmpty* and *ArmPrevEmpty* (finding neighbor empty element).

For arbitrary projections the operations are: *ArmNextProj* and *ArmPrevProj*.

The operations, which create indexes, are:

- *ArmSpaceIndex* (returns the space index of the non-empty structures in the given information space;
- *ArmProjIndex* (gives the space index of basic information elements of a given hierarchical or arbitrary projection).

The service operations for counting non-empty elements or subspaces are correspondingly:

- *ArmSpaceCount* (returning the number of the non-empty structures in given information space);
- *ArmProjCount* (calculating the number of elements of given (hierarchical or arbitrary) projection).

The ArM32 logical operations defined in the multi-domain information model are based on the classical logical operations – intersection, union, and supplement, but these operations are not so trivial. Because of complexity of the structure of the spaces these operations have at least two principally different realizations based on codes of the information spaces' elements and on contents of those elements.

The ArM32 information operations can be grouped into four sets corresponding to the main information structures: elements, spaces, aggregates, and indexes. Information operations are context depended and need special realizations for concrete purposes. Such well-known operations are for instance transferring from one structure to another, information search, sorting, making reports, etc.

Finally, several operations, which serve information exchange between ArM32 archives (files) such as copying and moving spaces from one to another archive exist.

4.3.3 Advantages of Multi-Dimensional Numbered Information Spaces

We need to discuss shortly the main concept we use – the information space. Its main structure is an ordered set of numbered information elements. These elements may be information spaces or terminal elements. Of course, the hierarchical structures are well-known. The new aspect of this model is the possibility to connect elements from different spaces and levels of the hierarchy using poly-indexation and in this way to create very large and complex networks with a co-ordinate hierarchical basis.

The variety of interconnections is the characteristic, which permits us to call the ordered set of numbered information elements "Information Space". In the information space, different information structures may exist at the same time in the same set of elements. In addition, the creation and destruction of the link's structures do not change the basic set of elements. The elements and spaces always exist but, in any cases, they may be "empty". At the end, the possibility to use coordinates is good for well-structured models where it is possible to replace search with addressing.

Hence, the advantages of the numbered information spaces are:

- the possibility to build growing space hierarchies of information elements;
- the great power for building interconnections between information elements stored in the information base;
- the practically unlimited number of dimensions (this is the main advantage of the numbered information spaces for well-structured tasks where it is possible "to address, not to search");
- the possibility to create effective and useful tools, in particular for association rule mining.

In the next chapter we demonstrate the advantages of using such memory structuring in the field of data mining on the example of realization of one class association rule classifier, called MPGN.

Conclusion

Here we made an overview of data structures used for presenting information in different fields of data mining and pattern recognition.

A special kind of data organization called "Multi-dimensional numbered information spaces", allowing context-free access to high dimension points, where different kind of data structures can be stored was discussed.

The idea to use the process of replacing the names by numbers, which permits using of mathematical functions and addressing vectors for accessing the information instead of search engines is established in Multi-Domain Information Model (MDIM) and the corresponding Multi-domain Access Method.

The program realization of this method, called ArM 32 was outlined. The main structures and operations with them are discussed.

5 PGN and MPGN Algorithms

Abstract

In this chapter the algorithms of two classifiers– PGN and MPGN – are described.

PGN creates association rules, striving for maximal accuracy of produced rules.

MPGN employs multilayer structure. It offers the possibility to escape combinatorial explosion using smart disposing of the information.

A coding convention is introduced first, followed by the description of the PGN and MPGN algorithms.

5.1 Coding Convention

Usually in classification tasks rectangular datasets are used. They are a set of instances $\mathbf{R} = \{R^i, i \in 1,...,r\}$. Each instance represent a set of attribute-value pairs $R = \{C = c, A_1 = a_1, ..., A_n = a_n\}$. Because in the rectangular datasets the positions of class and attributes are fixed, the instances are written as vectors, which contain only values of attributes. For increasing the readability class value (first position) is separated with "|": $R = (c \mid a_1, ..., a_n)$.

Every instance has the same quantity of attributes, but some of the values may be omitted. First attribute is the class variable denoted c; the input attributes are denoted a_k. In the remainder of the text, we will simply refer to them as "attributes".

Attribute positions of a given instance, which can take arbitrary values from the attribute domain, are denoted as "-".

Thus each instance (record) is presented as: $R=(c\,|\,a_1,...,a_n)$; where n is the number of attributes (feature space dimension), $c\in \mathrm{N}$; $a_k\in \mathrm{N}$ or $a_k=\text{"}-\text{"}$, $k\in[1,...,n]$.

More precisely, the class values and attribute values receive values, which are natural numbers from 1 to some maximal value (specific for each attribute), i.e. $c\in[1,...,M_c]$, $a_k\in[1,...,M_{a_k}]$.

Pattern is denoted P and has similar structure as instances. A pattern is a subset of an instance.

$$\left.\begin{array}{l} P=(c\,|\,a_1,...,a_n) \\ R=(c\,|\,b_1,...,b_n) \\ a_i=b_i \text{ or } a_i=\text{"}-\text{"} \end{array}\right\} P\subseteq R\,.$$

For example $P=(2\,|\,3,2,-,2)\subseteq R^1=(2\,|\,3,2,1,2)$.

In $P=(c\,|\,a_1,...,a_n)$ usually (c) is called head of a pattern and $(a_1,...,a_n)$ is called its body.

Each pattern defines a rule in a following manner: if some attributes have given specified values (looking non-arbitrary values of the body of the pattern), than (in some degree of accuracy) we can say that the observed object belongs to the class, pointed in the head of the pattern.

The cardinality of one pattern is defined as number of "non-arbitrary" attribute values:

$|P|=\text{number of } a_k\neq\text{"}-\text{"};\ |P|\le n$.

For the set of patterns $\mathbf{P}=\{P^i, i\in 1,...,m\}$ we can define maximal cardinality as maximum of cardinalities of patterns in the set.

$$MaxCard(\mathbf{P})=\max_{i\in 1,...,m}|P^i|\,.$$

The intersection between P^i and P^j is the result of matching of these patterns.

$$P^i\cap P^j=(c^l\,|\,a_1^l,...,a_n^l):\ c^l=\begin{cases} c^i : c^i=c^j \\ \text{"}-\text{"} : c^i\neq c^j \end{cases} \text{ and } a_k^l=\begin{cases} a_k^i : a_k^i=a_k^j \\ \text{"}-\text{"} : a_k^i\neq a_k^j \end{cases}\,.$$

If $|P^i\cap P^j|>0$ and $c^i=c^j$, then $P=P^i\cap P^j$ is a pattern. P is successor of P^i and P^j. And from other side, P^i and P^j is called predecessors of P.

For example, the successor of $R^1 = (2 \mid 3,2,1,2)$ and $R^3 = (2 \mid 3,2,2,2)$ is $P = (2 \mid 3,2,-,2)$.

Between patterns that belong to different classes different situations can exist. We are interested in two cases:

- exception pattern versus general pattern;
- contradictory patterns.

One pattern becomes an exception pattern for other ones if the body of second pattern (more general) is a subset of the body of first pattern (more concrete), but they belong to different classes.

$P^i = (c^i \mid a_1^i,...,a_n^i)$ is an exception pattern of $P^j = (c^j \mid a_1^j,...,a_n^j)$ if $(a_1^i,...,a_n^i) \supset (a_1^j,...,a_n^j)$ and $c^i \neq c^j$.

The contradictory situation means that the patterns have equal attributes (equal bodies) but belong to different classes (different heads).

P^i contradicts to P^j if $P^i = (c^i \mid a_1,...,a_n)$ and $P^j = (c^j \mid a_1,...,a_n)$, but $c^i \neq c^j$.

Hence the attributes at hand or the information available are not able to discriminate between the two classes.

Both situations can occur when:

- we have missing attributes;
- the class values are ill defined (exceptions);
- or the class/attributes values cannot be measured as required (noise, error).

The query instance Q (or only query) is similar to a pattern, but the class value is unknown. It is denoted $Q = (? \mid b_1,...,b_n)$.

The intersection size between pattern P and query Q is defined as $IntersectionSize(P,Q) = \frac{|P \cap Q|}{|P|}$. Let's mention that this operation is not symmetric, i.e. $IntersectionSize(P,Q) \neq IntersectionSize(Q,P)$.

The intersection percentage is calculated as $IntersectPerc(P,Q) = 100 * IntersectSize(P,Q)$.

The intersection percentage is 100% in the case when P became a subset of pattern Q', which has the head of the pattern P and the body of the query Q.

$$\left.\begin{array}{l} P = (c \mid a_1,...,a_n) \\ Q = (? \mid b_1,...,b_n) \end{array}\right\} \text{ if } P \subseteq Q' = (c \mid b_1,...,b_n) \text{ then } IntersectPerc(P,Q) = 100\%.$$

For the set of patterns $\mathbf{P} = \{P^i, i \in 1,...,m\}$ we can define maximal percentage as maximum of intersection percentages of patterns in the set.

$$MaxIPerc(\mathbf{P}, Q) = \max_{i \in 1,...,m} IntersectPerc(P^i, Q).$$

The support of a pattern P in a dataset $\mathbf{R} = \{R^i, i \in 1,...,r\}$ is the number of instances for which P became their subset.

$$Supp(P, \mathbf{R}) = \text{number of } R^i : P \subseteq R^i,\ R^i \in \mathbf{R}, i \in 1,...,r.$$

The confidence of a pattern $P = (c \mid a_1,...a_n)$ in a dataset $\mathbf{R} = \{R^i, i \in 1,...,r\}$ is equal to the ratio between support of the pattern and support of the body of the pattern in the dataset.

$$conf(P, \mathbf{R}) = \frac{Supp(P, \mathbf{R})}{Supp((- \mid a_1,...a_n), \mathbf{R})}.$$

5.2 PGN Classifier

Here we propose a CAR algorithm, named PGN. One of the main specifics of PGN is that it is a parameter free classifier. Let mention that in classical CAR algorithms users have to provide the support and confidence level.

The association rule mining goes from longest rules (instances) to the shorter ones until no intersections between patterns in the classes are possible. In the pruning phase the contradictions and inconsistencies of more general rules are cleared, after that the pattern set is compacted throwing all more concrete rules within the classes.

The remainder of the text contains the description of the algorithm of PGN classifier.

As example, a simple dataset is used, including the following instances:

```
R1:  (1| 1, 2, 4, 1)
R2:  (1| 1, 2, 3, 1)
R3:  (1| 3, 1, 3, 2)
R4:  (1| 3, 1, 4, 2)
R5:  (1| 1, 2, 4, 1)  Equal to R1
R6:  (1| 3, 1, 4, 2)  Equal to R4
R7:  (2| 3, 1, 1, 2)
R8:  (2| 2, 1, 1, 2)
R9:  (2| 3, 1, 2, 2)
```

5.2.1 Training Process

The training process consists of several steps:

- generalization – the process of association rule mining;
- pruning – the process of clearing exceptions between classes and lightening the pattern set;
- searching patterns with unique attributes. This step is optional as well as it not typical CAR strategy and from other side it created very powerful patterns, which is good for some dataset, but not for the others.

➢ ***Step 1: Generalization***

The step of creating the pattern set consists of two sub-steps:

1. Adding instances to the pattern set.
2. Creating all possible intersection patterns between patterns within the class.

✓ ***Sub-step 1.1: Adding Instances***

The instances of the learning set $LS = \left\{ R^i \right\}$, $i = 1,...,t$ are added to the pattern set as initial patterns. All patterns are separated in accordance of their classes (Figure 5).

Figure 5. Adding instances in the pattern set

✓ ***Sub-step 1.2: Adding Intersections***

For each class every combination of two patterns is intersected. If a new pattern exists, it would be added to the pattern set. If the patterns-candidates to be written into the pattern set (instances as well as patterns) are already in it, then they would not be duplicated; however the set of instances that are possible creators of the pattern would be expanded.

The process goes iteratively until no intersections are possible.

Figure 6 shows the process of creating the pattern set on the example dataset.

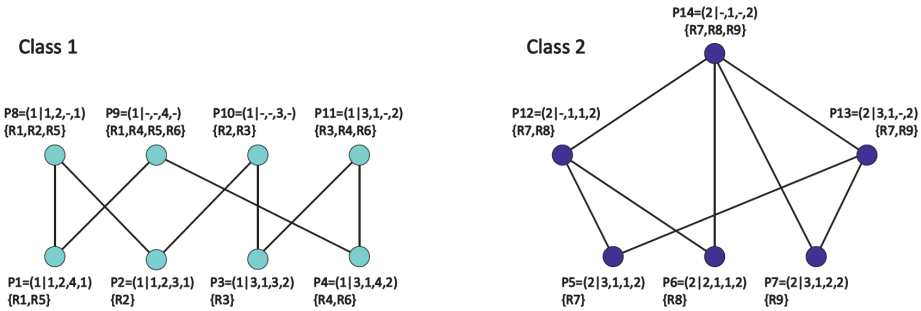


Figure 6. Adding intersections in the pattern set

Thus, after this step, the pattern set consists of these objects:

$$PS=\left\{P^l\right\},\ P^l:\begin{cases} R^l\in LS \\ P^l=P^i\cap P^j;\ \ P^i\in PS, P^j\in PS, c^i=c^j;\ \left|P^l\right|>0 \end{cases}.$$

- ***Step 2: Pruning***

In this step some patterns are removed from the pattern set using two sub-steps:

1. Deleting contradictory patterns as well as general patterns that have exception patterns in some other class.
2. Removing more concrete patterns within the classes. This step ensures compactness of the pattern set that can be used in the recognition stage.

- ✓ ***Sub-step 2.1: Clearing Contradictions and General-Exception Patterns between Classes***

In this sub-step the patterns, belonging to different classes are paired. If one pattern matches another pattern (but they have a different class value), then the more general is removed. If two patterns match each other then both of them are removed.

$$P^i,P^j\in PS,\ c^i\neq c^j:\ \begin{cases} \left|P^i\cap P^j\right|=\left|P^i\right|<\left|P^j\right|: \text{remove } P^i \\ \left|P^i\cap P^j\right|=\left|P^j\right|<\left|P^i\right|: \text{remove } P^j \\ \left|P^i\cap P^j\right|=\left|P^i\right|=\left|P^j\right|: \text{remove } P^i,P^j \end{cases}$$

If a dataset does not contain missing values, then all instances have equal $\left|R\right|=n$ and all other patterns will have smaller sizes. This means that checking

up for data consistency can be done with comparison of patterns only with the instances but not with all patterns from other classes.

This sub-step tries to supply the maximum confidence of the resulting rules.

This operation removes the patterns that do not formulate a representative for a given class combination, because in another class there exists pattern with an equal or more concrete combination of the same values of attributes, which can pretend to recognize the request.

Furthermore, by removing incorrect patterns (records with equal attributes, which belongs to different classes) this operation ignores the possible inconsistencies in the learning set.

It should be noted that the idea of supplying a confidence threshold of 100% can result in an empty pattern set for noisy datasets.

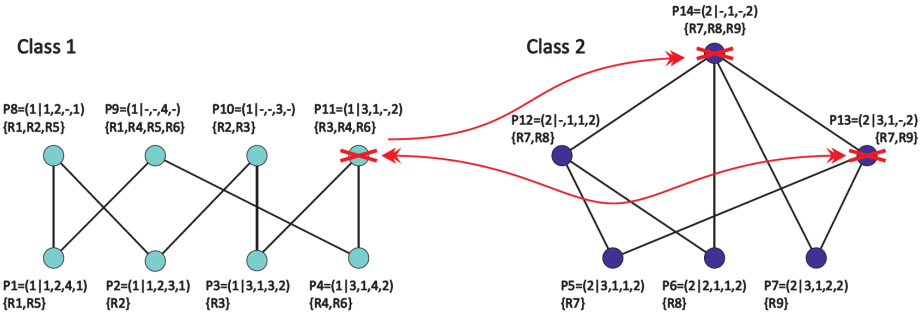


Figure 7. Supplying maximum confidence of the rules

Figure 7 illustrates the first pruning sub-step for example dataset.

This process passes in two steps: labeling followed by removing.

✓ *Sub-step 2.2: Retain Most General Rules in the Classes*

This sub-step is provided again within the classes. Patterns from equal classes are compared and, conversely to the previous step, more concrete patterns are deleted, i.e. the larger pattern is removed.

$$P^i, P^j \in PS,\ c^i = c^j : \begin{cases} \left|P^i \cap P^j\right| = \left|P^i\right| < \left|P^j\right| : \text{remove } P^j \\ \left|P^i \cap P^j\right| = \left|P^j\right| < \left|P^i\right| : \text{remove } P^i \end{cases}$$

The rationale behind is that after first sub-step in the pattern set remains only patterns that are not exceptions to the other class. Because of this, we can make lighter the pattern set by removing patterns for which other patterns are subsets.

Figure 8 illustrates lightening the pattern set for example dataset.

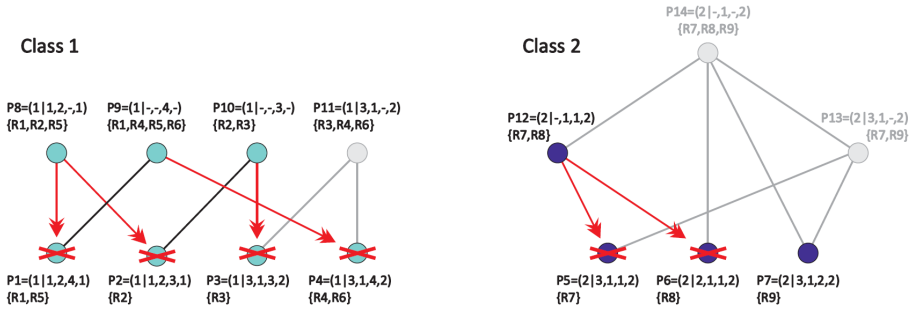


Figure 8. Retain most general rules

As a result outcome of this step in the pattern set remain only patterns that are general for the class that they belong and their bodies are not subsets of the bodies of patterns in other classes.

In the example dataset the pattern set contains following patterns with corresponded support:

	Pattern set	Support	Support set
	Class 1		
P8	(1| 1, 2, -, 1)	3	{R1,R2,R5}
P9	(1| -, -, 4, -)	4	{R1,R4,R5,R6}
P10	(1| -, -, 3, -)	2	{R2,R3}
	Class 2		
P7	(2| 3, 1, 2, 2)	1	{R9}
P12	(2| -, 1, 1, 2)	2	{R7,R8}

5.2.2 Recognition Process

The record to be recognized is given by the values of its attributes $Q = (?\,|\, a_1, a_2, ..., a_n)$. Some of the features may be omitted.

We try to find the best fit between the query and the patterns from the recognition model.

The idea is that in the recognition model at the same time there are patterns that are very global (only a few non-arbitrary attributes) and some patterns that are concrete (with more / a lot of non-arbitrary attributes). The global patterns have a short size, but they are very powerful for the class. They contain only a few attributes, but trustworthy for recognizing this class. In parallel, the concrete patterns remains because more global combinations were killed by other classes – so, it means that maybe there are no such kind of very specific attributes and only complex combination of them characterizes the class.

So, we affirm that intersection percentage can make some kind of alignment between short and long patterns. Of course, the intersection percentage suffers

from some kind of inequality when is not 100%; in this case short patterns became lower value then the longer ones. But, we argue that this preceding of the more concrete rules when there are not total matching with the query is better because more features are equal.

These assumptions we implemented in the algorithm of recognition as follows: During the recognition stage all patterns $P^l \in PS$, which have maximal intersection percentage (size) with request Q build a list of patterns of potential answers. The list consists of triplets containing the number of the class, the coverage and the position of the pattern. While traversing the patterns, dynamically the highest intersection percentage (size) is held. As potential answers only these patterns are retained that have such intersection percentage (size). Finally, the class, which has maximal sum of supports of patterns of this class, belonging to the list of potential answer, is given as answer.

Let's see two examples on the tested dataset.

For the query $Q = (?\,|\,2,1,-,2)$:

	Pattern set	$P \cap Q$	$IntSize(P,Q)$	Support	Support set
	Class 1				
P8	(1| 1, 2, -, 1)	(?| -, -, -, -)	0	3	{R1,R2,R5}
P9	(1| -, -, 4, -)	(?| -, -, -, -)	0	4	{R1,R4,R5,R6}
P10	(1| -, -, 3, -)	(?| -, -, -, -)	0	2	{R2,R3}
	Class 2				
P7	(2| 3, 1, 2, 2)	(?| -, 1, -, 2)	0.50	1	{R9}
P12	(2| -, 1, 1, 2)	(?| -, 1, -, 2)	0.667	2	{R7,R8}

The maximal intersection size between patterns and query is 0.667 and only one pattern from class 2 has such intersection size. Class 2 is given as answer.

For the query $Q = (?\,|\,1,2,1,2)$:

	Pattern set	$P \cap Q$	$IntSize(P,Q)$	Support	Support set
	Class 1				
P8	(1| 1, 2, -, 1)	(?| 1, 2, -, -)	0.667	3	{R1,R2,R5}
P9	(1| -, -, 4, -)	(?| -, -, -, -)	0	4	{R1,R4,R5,R6}
P10	(1| -, -, 3, -)	(?| -, -, -, -)	0	2	{R2,R3}
	Class 2				
P7	(2| 3, 1, 2, 2)	(?| -, -, -, 2)	0.250	1	{R9}
P12	(2| -, 1, 1, 2)	(?| -, -, 1, 2)	0.667	2	{R7,R8}

The maximal intersection size between patterns and query is 0.667.

There are two patterns from both classes for which $IntSize(P,Q) = 0.667$, i.e. there are two sets of potential answers: $class\ 1 : \{P8\}$ and $class\ 2 : \{P12\}$. In this case the set of patterns of class 1 has higher support 3. Because of this class 1 is given as answer.

5.3 MPGN Algorithm

The PGN classifier has several advantages and as we can see in the chapter with experiments shows very good benefits. It possesses all advantages of CAR classifiers, such as creating compact pattern set, used in the recognition stage, easy interpretation of the results, and very good accuracy for clear datasets.

In parallel, during the program realization one disadvantage is seen, connected with exponential growth of operations during the process of creating the pattern set.

In order to overcome this bottleneck, as well as to quickly find the potential answer in the recognition stage MPGN algorithm is created.

MPGN is abbreviation from "Multi-layer Pyramidal Growing Networks of information spaces", which is kind of CAR algorithm that uses advantages of numbered information spaces. The main goal is to extend the possibilities of network structures by using a special kind of multi-layer memory structures called "pyramids", which permits defining and realizing of new opportunities.

The basic ideas in PGN and MPGN are similar. The main differences are connected with:

- MPGN extends PGN structures for presenting patterns in the pattern set using multi-layer memory structures, called pyramids;
- the possibility to save all patterns in an efficient manner, using multi-dimensional numbered information spaces, allows to keep all patterns in the pattern set, because of this the pruning step is different – only contradictory patterns are removed;
- such pattern set contains possibilities to be implemented different kind of algorithm in the recognition stage, searching for maximal cardinality with 100% intersection percentage down to the constructed pyramids.

It should be noted that multi-layer memory structures can be easily implemented in PGN. In practice, the steps of creating the pattern set use theoretically the same algorithms in both classifiers and can be realized using common tools.

5.3.1 Training Process

The training process in MPGN consists of:

- preprocessing step;
- generalization step;
- pruning step.

➢ ***Preprocessing Step***

MPGN deals with instances and patterns separately for each class. This allows the MPGN algorithm to be implemented for use on parallel computers which could be particularly helpful within the current trend of using cloud services and grid infrastructures.

The preprocessing step is aimed to convert the learning set in a standard form for further steps. It consists of:

- discretization of numerical attributes [Mitov et al, 2009b];
- numbering the values of attributes.

The instances are converted to numerical vectors after discretization and the juxtaposing positive integers to nominal values had been made.

➢ ***Generalization Step***

The process of generalization is a chain of creating the patterns of upper layer as intersection between patterns from lower layer until new patterns are generated. For each class, the process starts from the layer 1 that contains the instances of the training set. Patterns, generated as intersections between instances of the training set are stored in layer 2. Layer N is formed by patterns generated as intersections between patterns of the layer N-1. This process continues until further intersections are not possible.

During generalization, for every class a separate pyramidal network structure is built. The process of generalization creates "vertical" interconnections between patterns from neighborhood layers. These interconnections for every pattern are represented by a set of "predecessors" and a set of "successors".

The predecessors' set of a concrete pattern contains all patterns from the lower layer which were used in the process of its generalization. Thus in cases of different intersections generating the same pattern in the final outcome all patterns appearing in the intersection would be united as predecessors of the resulting pattern.

The predecessors sets for instances of layer one are empty.

The successors' set of a concrete pattern contains the patterns from upper layer, which are created from it.

The successors' sets of patterns on the top of the pyramid are empty. These patterns are called "vertexes" of the corresponded pyramids.

One pattern may be included in more than one pyramid, but the vertex pattern belongs only to one pyramid.

It is possible for any pyramid to contain only one instance.

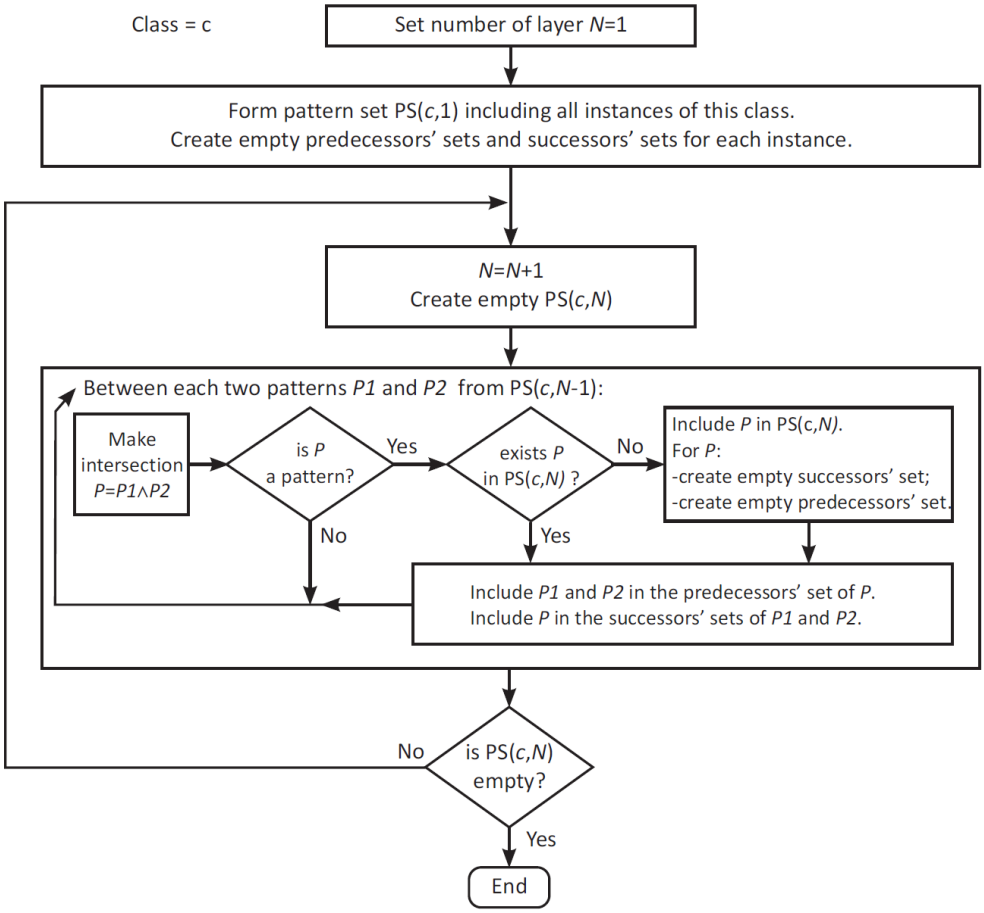


Figure 9. MPGN – the process of generalization of one class

Figure 9 presents the block scheme of the process of generalization for one class.

Here, as example we will use following dataset:

Class 1	
R1:	(1|5,5,5,5)
R2:	(1|5,3,5,4)
R3:	(1|5,4,5,3)
R4:	(1|1,1,1,1)
R5:	(1|4,1,3,1)
R6:	(1|1,2,1,1)
R7:	(1|1,2,2,2)
R8:	(1|4,2,4,1)

Class 2	
R9:	(2|4,2,3,1)
R10:	(2|3,2,3,1)
R11:	(2|2,1,2,1)
R12:	(2|4,1,2,1)
R13:	(2|2,2,2,1)

Separating of two classes are made only for increasing the readability.

We will use this dataset for showing different steps of MPGN algorithm.

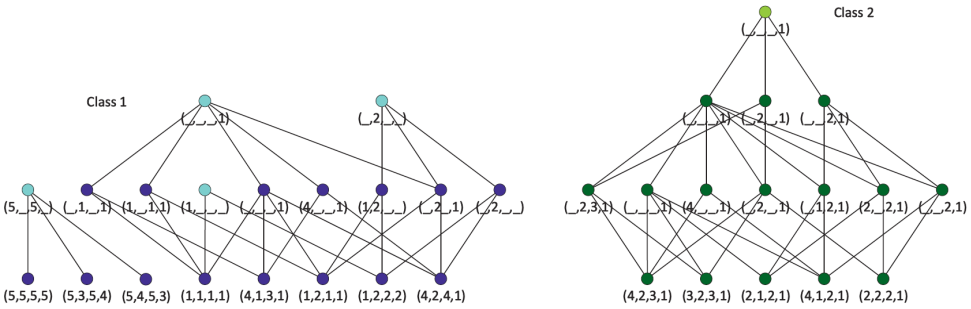


Figure 10. MPGN – Result of generalization step on the example dataset

Figure 10 illustrates the result of the generalization step of MPGN for example dataset. For simplifying the texts in the figures here and later patterns are presented only with value attributes. The class label is omitted as it is known from the pyramids in which pattern belongs. Light points denote vertexes of the created pyramids.

➢ *Pruning Steps*

The pruning steps combines two different processes:

- pre-pruning – in parallel with the generalization;
- post-pruning – pruning the contradictions.

✓ *Pre-pruning*

During the generalization a huge amount of combinations arises in big datasets. To restrict the combinatorial explosure different techniques can be applied. We use three different mechanisms for solving which of created patterns to be included in the process of generalization.

The first mechanism allows to be excluded the patterns that are generated by little number of patterns from the lower layer. This is similar to support but here is taken into account not the primary instances while the direct predecessors.

The other one tries to exclude the very general patterns from the beginning layers, using the presumption that these patterns will be arisen again in the upper layers. For this purpose, the ratio between the cardinality of the generated patterns and the cardinality of the predecessor patterns can be used as restriction.

✓ ***Post-pruning***

Post-pruning is the process of iterative analysis of vertex patterns of all pyramids from different classes and removing all contradictory vertex patterns. The algorithm is presented on Figure 11.

As a result, some of the most general patterns are deleted, because the vertexes with the same bodies were available in other classes (and they also are deleted). The primary pyramids are decomposed to several sub-pyramids with lower number of layers.

The vertexes of such pyramids do not contradict with vertexes of pyramids of other classes.

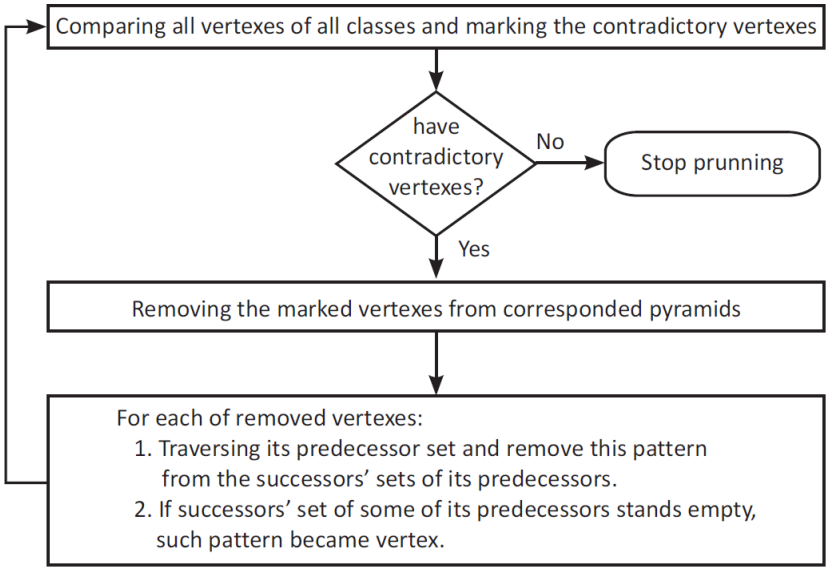


Figure 11. MPGN – post-pruning

Here we will give the visual explanation of the pruning the contradictions in already made pyramids of the example dataset.

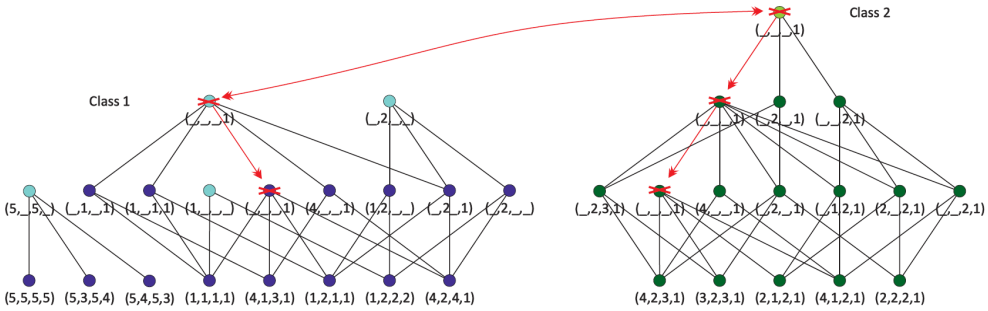


Figure 12. Post-pruning – starting process

Figure 12 shows the start of the process, where vertexes of pyramids of class 1 and class 2 are compared and contradictory vertexes as well as all successor equal patterns are destroyed.

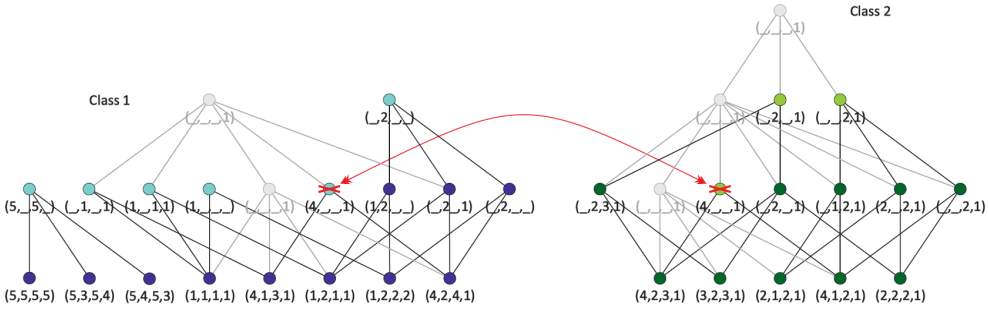


Figure 13. Post-pruning – continuing the process

The process of destroying of contradictory vertexes cause the arising of new vertexes from the patterns of corresponding pyramids. For new vertexes the search and destroying of contradictory patterns are applied again.

Figure 13 shows this next step on the example dataset. The process continues iteratively since no contradictions between vertexes of pyramids are found. In our case after second traversing no new contradictions were found and process of destroying pyramids stops.

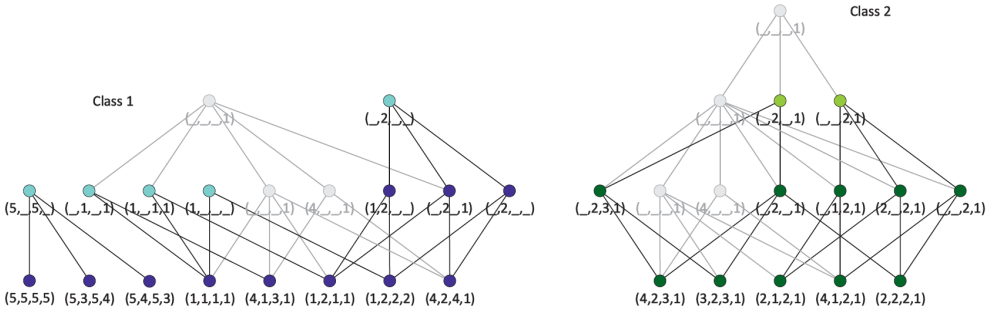


Figure 14. Final result of post-pruning

Figure 14 shows the final result of the post-pruning. In grey we show the destroyed parts of pyramids.

5.3.2 Recognition Process

The instance to be recognized is given by the values of its attributes $Q = (?\,|\,b_1,...,b_n)$. Some of the values may be omitted. If some attributes are numerical, the values of these attributes are replaced with the number of corresponded discretized interval, where the value belongs. The categorical attributes also is recoded with the corresponded number values.

Initially the set of classes, which represent potential answers CS includes all classes: $\mathrm{CS} = \{c \,|\, c = 1,...,Mc\}$.

The recognition process consists of two main steps:

- creating recognition set for every class separately;
- analyzing resulting recognition sets from all classes and making decision which class to be given as answer.

➢ ***Creating Recognition Set for Each Class***

At this stage each class is processed separately.

The goal is to create for each class the recognition set, which contains all patterns with maximal cardinality that have 100% intersection percentage with the query.

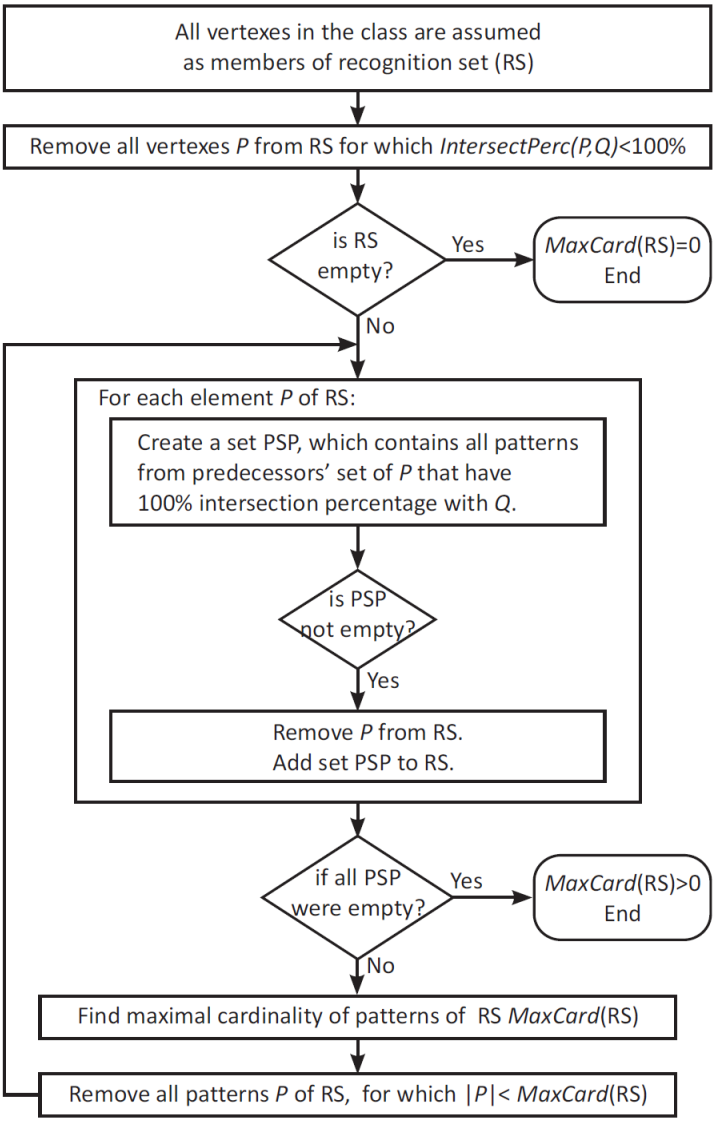


Figure 15. MPGN – creating recognition set for one class

The process starts from the vertexes of all pyramids that belong to examined class. Using the predecessor sets of the patterns in the recognition set each pattern is replaced with the set of their predecessor that have 100% intersection percentage with the query, if this set is not empty. After lighting the recognition set keeping only patterns with maximal coverage the process is iteratively repeated down to the layers until no new patterns became in the recognition set.

Figure 15 shows the block-scheme of this process.

The process of creation of recognition set for each class also can be implemented on parallel computers.

- ***Analyzing Results and Make Final Decision***

Figure 16 shows the general schema of this step.

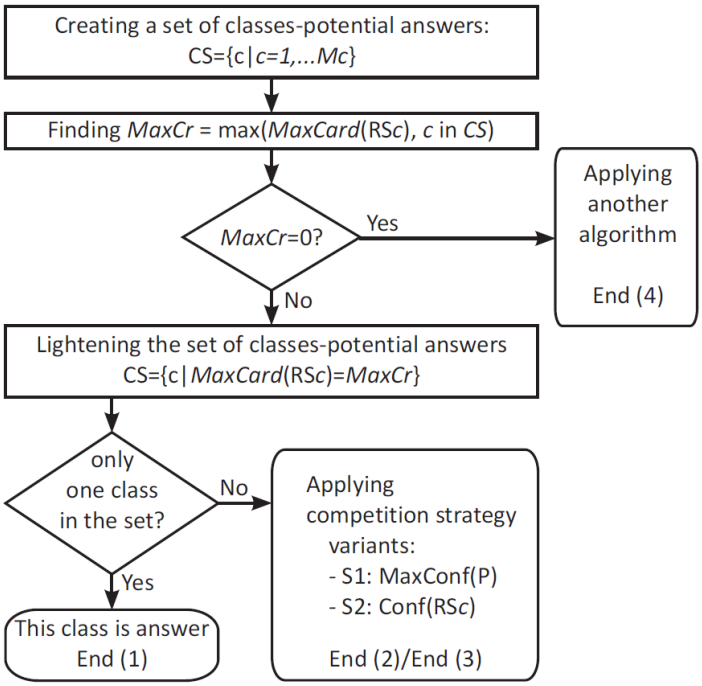


Figure 16. MPGN – comparative analysis between classes

The result of the first step processing are the recognition sets for all classes $\mathrm{RS}c$, $c \in [1,...,M_c]$, which contain the patterns with maximal cardinality for this class $MaxCard(\mathrm{RS}c)$, $c \in [1,...,M_c]$ that have 100% intersection percentage with Q.

The goal is to find the class, which contains the patterns with highest cardinality in its recognition set.

For this purpose, first the maximum of all maximal cardinalities of the recognition sets of classes from CS is discovered.

$$MaxCr = \max_{c \in \mathrm{CS}} MaxCard(\mathrm{RS}c)$$

All classes that have not such maximal cardinality are excluded from the set CS.

$$\mathrm{CS} = \{c : MaxCard(\mathrm{RS}c) = MaxCr\}$$

After this step, if only one class remains in CS, then this class is given as an answer (End 1).

Let us see the recognition process over the example dataset for query $Q=(?|5,2,3,1)$, showed in Figure 17.

Each class is examined separately.

Only vertex $(1|\_,2,\_,\_)$ of class 1 matches the query. From its predecessors the pattern $(1|\_,2,\_,1)$ match the query and has bigger cardinality equal to 2. No matching is found in its predecessors and process for class 1 stops.

In the case of class 2 the vertex $(2|\_,2,\_,1)$ matches the query and its predecessor $(2|\_,2,3,1)$ also matches and has bigger cardinality equal to 3.

As result of comparison of maximal cardinalities between classes class 2 is given as answer.

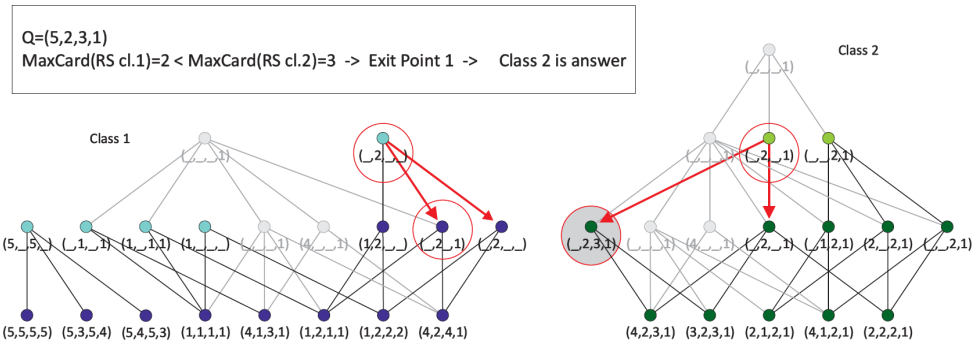


Figure 17. Example of recognition in MPGN – Exit Point 1

In the case when several classes exist with maximal cardinality $MaxCr$ (i.e. $Card(\mathrm{CS})>1$), different strategies can be used to choose the best competitor. Here we will discuss two basic options:

- S1: choose from each class a single rule with maximal confidence within the class and compare with others;
- S2: find "confidence of recognition set", i.e. the number of instances that are covered of patterns from recognition set of this class over the number of all instances of this class and compare results.

✓ *Variant Multiple Classes-Candidates: S1*

The algorithm for the S1-option can be summarized as follows (Figure 18):

Find the pattern with maximal confidence of each of recognition sets of the classes in CS:

$$MaxConf(P \in \mathrm{RS}c) = \max_{P \in \mathrm{RS}c}(Conf(P)).$$

We find the maximum of received numbers from all classes:

$$MaxCn = \max_{c \in \mathrm{CS}} MaxConf(P \in \mathrm{RS}c)$$

Again we make lightening of CS retaining only classes that have such maximum:

$$\mathrm{CS} = \{c : MaxCard(\mathrm{RS}c) = MaxCr \text{ and } MaxConf(P \in \mathrm{RS}c) = MaxCn\}$$

If only one class has such a maximum, this class is given as an answer. In the other case, the class with maximal support from CS is given as answer.

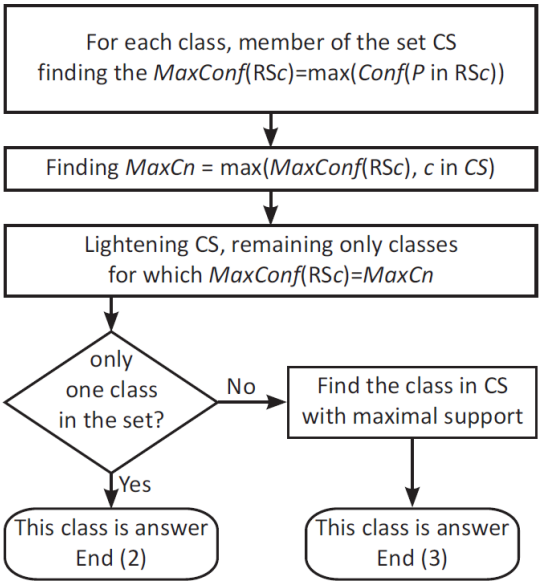


Figure 18. Recognition strategy S1: using 1 rule with maximal confidence

Let us see the behavior of MPGN recognition: S1 strategy (Figure 19) on the case of query $Q=(?|5,2,5,1)$.

For class 1 patterns that matched the query with maximal cardinality 2 are $(1|5,\_,5,\_)$ and $(1|\_,2,\_,1)$. For class 2 pattern that matched the query with maximal cardinality again 2 is $(2|\_,2,\_,1)$.

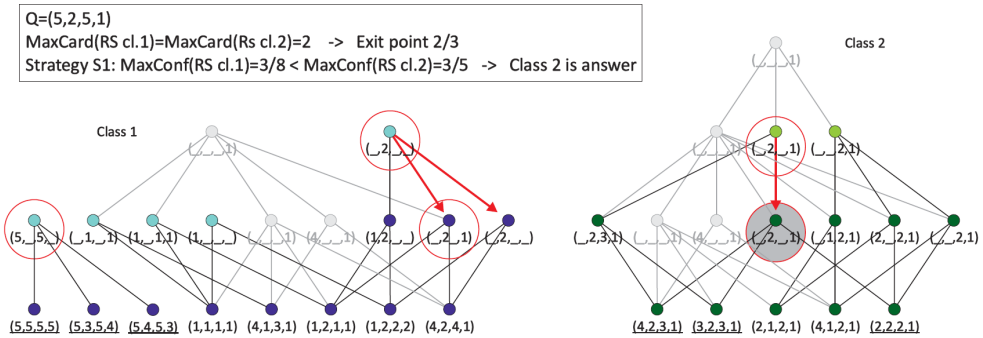


Figure 19. Example of recognition in MPGN – Exit Point 2: Strategy S1

Because of the equality of the maximal cardinality we continue with finding the rule with maximal confidence within each class.

For class 1 the confidence of the pattern $(1|5,\_,5,\_)$ is 3/8 (maximal for this class). For class 2 $(2|\_,2,\_,1)$ has confidence 3/5. Following strategy S1 class 2 is given as answer.

✓ *Variant Multiple Classes-Candidates: S2*

This algorithm is similar to the previous (Figure 20).

The main difference is that instead of finding the pattern with maximal confidence of each of recognition sets, here the "confidence of recognition set" is evaluated, i.e. the number of instances that are covered by patterns from the recognition set of this class over the number of all instances of this class and then results are compared.

The rationale behind this is to take into account how many instances had been covered by all patterns from the recognition set.

In practice this is the disjunction of the predecessors' sets of Layer 1 (not direct predecessors) of the patterns that belong to $\mathrm{RS}c$.

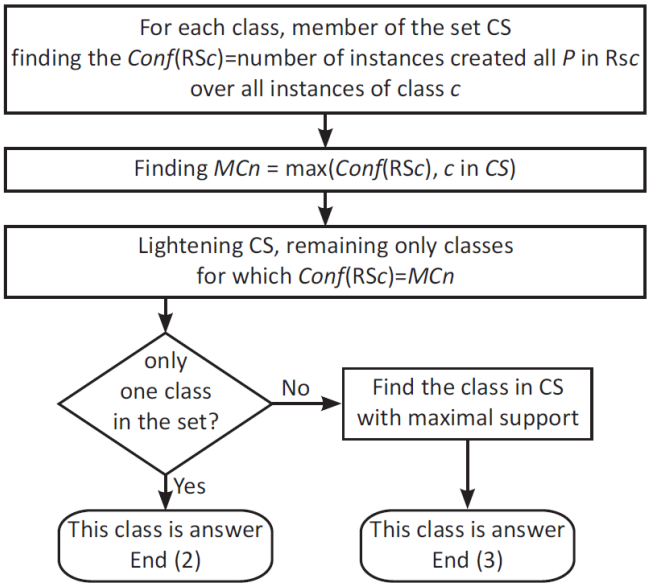


Figure 20. Recognition strategy S2: using confidences of the recognition sets

Let us see the behavior of MPGN recognition: S2 strategy (Figure 21) on the case of the same query $Q=(?|5,2,5,1)$.

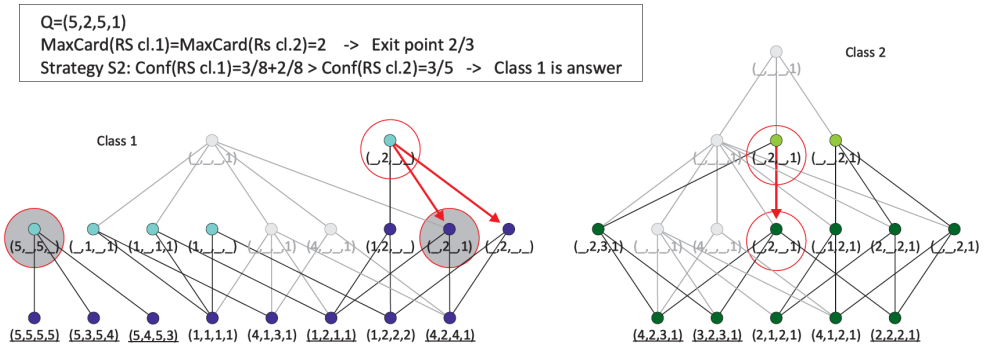


Figure 21. Example of recognition in MPGN – Exit Point 2: Strategy S2

Following strategy S2 we found the confidences of the set of patterns that matches query with maximal cardinality. They are correspondingly: 3/8+2/8=0.625 for class 1 and 3/5=0.600 for class 2. As a result class 1 is given as answer.

✓ *Variant: Empty Recognition Sets*

The worst case is when all recognition sets were empty.

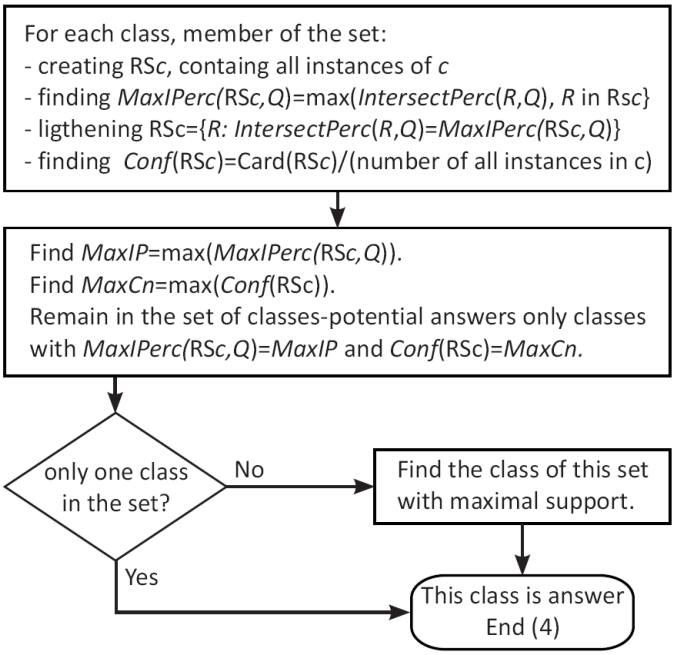


Figure 22. Variant of recognition when 100% intersection percentage gives not result

Here we create new recognition sets, including instances with maximal intersection percentage with the query.

$$MaxIPerc(\mathrm{RS}c,Q) = \max_{R\in \mathrm{RS}c}(IntersectPerc(R,Q));$$

$$\mathrm{RS}c = \{R : IntersectPerc(R,Q)) = MaxIPerc(\mathrm{RS}c)\}.$$

Find $$Conf(\mathrm{RS}c) = \frac{|\mathrm{RS}c|}{|\{\text{instances of class } c\}|}$$

The class that contains the set with maximal confidence is given as an answer. The reason is that a higher confidence is received because the rules are more inherent to this class. If two or more classes have equal confidence, than the class with maximal support is given as answer (End 4).

This process is presented on Figure 22.

An analysis of the results from the previous step is made at this step; it is performed simultaneously over all classes simultaneously. The parallelization of this process is not explicitly shown (it is not inherent to algorithm). The parallelization can be implemented during the process of cross-comparison of the results from all the classes.

Conclusion

In this chapter we provided a description of PGN and MPGN algorithms.

PGN creates association rules, striving for maximal accuracy of produced rules. One of the main specifics of PGN is that it is a parameter free classifier. The association rule mining goes from longest rules (instances) to the shorter ones until no intersections between patterns in the classes are possible. In the pruning phase the contradictions and inconsistencies of more general rules are cleared, after that the pattern set is compacted throwing all more concrete rules within the classes.

MPGN employs multilayer structure. It offers a possibility to escape combinatorial explosion using smart disposing of the information in so called multilayer structures "pyramids". Later these structures easy can be realized using ArM-structures.

In the case of MPGN the process of association rule mining and part of the pruning step are made in parallel, which allows to overcome the bottleneck of exponential growth of created rules. The other pruning step is connected with the process of clearing the contradictions between vertex patterns of all pyramids from different classes.

The recognition process in MPGN first creates the recognition sets for each class, after that analyzes the results in order to make a final decision. In this step different options can be observed: only one class is class-candidate; multiple classes are classes-candidates (in this case two different strategies are proposed: S1 – to use the most powerful rule, or S2 – to analyze the normalized support of the whole recognition set of each class), or worst case scenario when recognition sets are empty. The possible actions in the case of each of those different options had been presented and discussed.

6 Program Realization

Abstract

The realization of the proposed algorithms PGN and MPGN were made in a common data mining analysis environment PaGaNe. It contains a variety of statistical and data mining tools. PaGaNe uses ArM 32 as a basic access method and all algorithms are fine-tuned to use the advantages given by such memory structure environment.

For these purposes preprocessing steps, creating bijective functions between attribute values and natural numbers, as well as converting input information into numerical vectors is implemented.

Taking into account the fact that most of CAR algorithms work with categorical data, several known discretization methods for partitioning the attribute space had been implemented and presented.

The basic structure used for keeping created patterns in the pattern sets of proposed algorithms are the so called pyramids – multi-layer structures, containing vertical connections between patterns.

The algorithm for realizing smart storing and extracting of patterns from these structures, using advantages of context-free access method, realized in ArM 32, is proposed.

6.1 Common Environment

An international joint research group has been working on the design of a data mining analysis environment called "PaGaNe". It integrates a number of data mining algorithms, such as association rule miners, class association rule (CAR) algorithms, etc. [Mitov et al, 2009a/b].

A distinguished feature of PaGaNe is that it uses the advantages of multi-dimensional numbered information spaces [Markov, 2004], provided by the access method ArM 32, such as:

- the possibility to build growing space hierarchies of information elements;
- the great power for building interconnections between information elements stored in the information base;
- the possibility to change searching with direct addressing in well-structured tasks.

An important feature of the approaches used in PaGaNe, is the replacement of the symbolic values of the objects' features with integer numbers of the elements of corresponding ordered sets. Thus all instances or patterns can be represented by a vector of integer values, which may be used as co-ordinate address in the corresponding multi-dimensional information space.

The program realization of PGN and MPGN are implemented within this environment.

6.2 Preprocessing Step

The preprocessing step is aimed to convert the learning set in a standard form for further steps. It consists of:

- discretization of numerical attributes;
- numbering the values of attributes;
- attribute subset selection.

6.2.1 Input Data

The data can be entered directly, but usually files, containing the datasets are used. Standard ".csv"-files, that contain rectangular datasets can be used as input files. For assuring compatibility with WEKA ".arff"-files also can be used as an input format. The user can use different files for a learning set and examining set, or splitting incoming file into learning set and examining set in a particular proportion. Cross-validation also can be applied.

6.2.2 Discretization

Discretization methods from different classes had been selected in order to examine which of them supplies more convenient discretization for PGN Classification Method.

- ***Fayyad-Irani Discretization***

Fayyad-Irani Discretization method [Fayyad and Irani, 1993] is supervised hierarchical split method, which uses the class information entropy of candidate

partitions to select boundaries for discretization. Class information entropy is a measure of purity and it measures the amount of information which would be needed to specify to which class an instance belongs. It considers one big interval containing all known values of a feature and then recursively partitions this interval into smaller subintervals until MDL criterion or an optimal number of intervals is achieved.

The MDL Principle states that the best hypothesis is the one with minimal description length. As partitioning always decreases the value of the entropy function, considering the description lengths of the hypotheses allows balancing the information gain and eventually accepting the null hypothesis. Performing recursive bipartitions with this criterion leads to a discretization of the continuous explanatory attribute at hand. Fayyad-Irani Discretizator evaluates as a candidate cut point the midpoint between each successive pair of the sorted values. For each evaluation of a candidate cut point, the data are discretized into two intervals and the resulting class information entropy is calculated. A binary discretization is determined by selecting the cut point for which the entropy is minimal amongst all candidate cut points. This binary discretization is applied recursively, always selecting the best cut point. A MDL criterion is applied to decide when to stop discretization. It has been shown that optimal cut points for entropy minimization must lie between examples of different classes.

This method does not need additional parameters to be chosen by the user.

- ***Chi-Merge Discretization***

Chi-merge [Kerber, 1992] is a supervised hierarchical bottom-up (merge) method that locally exploits the chi-square criterion to decide whether two adjacent intervals are similar enough to be merged;

Chi-square (χ^2) is a statistical measure that conducts a significance test on the relationship between the values of a feature and the class. Kerber argues that in an accurate discretization, the relative class frequencies should be fairly consistent within an interval but two adjacent intervals should not have similar relative class frequency. The χ^2 statistic determines the similarity of adjacent intervals based on some significance level. It tests the hypothesis that two adjacent intervals of a feature are independent of the class. If they are independent, they should be merged; otherwise they should remain separate.

The bottom-up method based on chi-square is ChiMerge. It searches for the best merge of adjacent intervals by minimizing the chi-square criterion applied locally to two adjacent intervals: they are merged if they are statistically similar. The stopping rule is based on a user-defined Chi-square threshold to reject the merge if the two adjacent intervals are insufficiently similar. No definite rule is given to choose this threshold.

The stopping rule is based on a Chi-square threshold, which depends of degrees of freedom (in our case – the number of possible values of class minus one) and the significance level (commonly used significance levels are 90%, 95%, 99%). The chi-square threshold table in the system is adopted from [Bramer, 2007].

At the pre-processing step, the system builds a mapping function for the real values of each attribute to a number that correspond to the interval in which the value belongs to; this is a result of implementing a discretization method..

Figure 23 presents a screenshot from the experimental system "PaGaNe", which visualizes the results of discretization process using "Chi-merge" with parameter 90% significance level for attribute "sepal length" for "Iris" dataset from UCI repository [Frank and Asuncion, 2010]. Five intervals which had been formed as well as the distribution of different class values in the intervals can be seen. The right part of the screen is used to list the cut-points from each interval, the number of instances of the learning set and correspondences belonging to the class values of these instances.

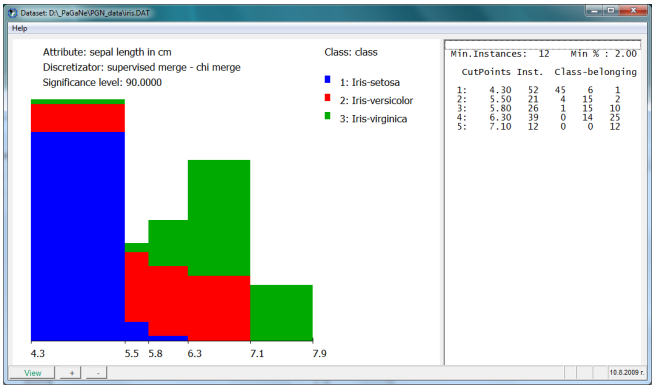


Figure 23. A Screenshot of visualizing discretization of attribute "sepal length in cm" of Iris database using Chi-merge discretizator

The system uses these intervals to find the corresponding nominal values for real attributes in learning and examining sets. This conversion of real data to categorical values gives the opportunity of PGN-classifier to be implemented on databases with the real values of attributes.

6.2.3 Converting Primary Instances into Numerical Vectors

At this stage, the bijective functions between primary values and positive integers are generated.

After that, the input instances are coded into numerical vectors, replacing attribute values with their corresponded numbers.

For example, during the input of instances of "Lenses" database the following mapping functions (numbering) is created:

class	
hard	1
none	2
soft	3

Age	
pre-presbyopic	1
presbyopic	2
young	3

prescription	
hypermetrope	1
myope	2

astigmatic	
no	1
yes	2

tears	
normal	1
reduced	2

and here is given the numerical representation of some instances:

Object;class;age;prescription;astigmatic;tears		***Numerical representations of instances***
1;none;young;myope;no;reduced	R_1=	(2|3,2,1,2)
2;soft;young;myope;no;normal	R_2=	(3|3,2,1,1)
3;none;young;myope;yes;reduced	R_3=	(2|3,2,2,2)
4;hard;young;myope;yes;normal	R_4=	(1|3,2,2,1)
5;none;young;hypermetrope;no;reduced	R_5=	(2|3,1,1,2)

The idea is to prepare data for direct use as addresses in the multi-dimensional numbered information spaces.

6.2.4 Attribute Subset Selection

The statistical observations on PaGaNe performance over the dataset can show that some attributes give no important information and the environment allows to point such attributes not to participate in further processing.

The automatic subset selection is an open part of the PaGaNe realization and it is in the front of current and near-future investigation [Aslanyan and Sahakyan, 2010].

6.3 PGN Program Realization

The first realization of the PGN algorithm did not use the added capacity of multi-layer structures. The patterns, created during the first phase, were kept sequentially. In this implementation the combinatorial explosion for big datasets was limited with examining only intersections between primary instances.

The good results, received by the experiments enforced further research in two directions:

- to find a way to overcome the bottleneck of exponential combinatorial growth of intersections;

- to investigate other possibilities within CAR algorithms in different steps of the process.

Here we will not present in detail the implementation of the PGN classifier. We will mention only that, using appropriate parameters for changing pruning step and recognition criterion, the realization of MPGN covers the PGN algorithm.

6.4 MPGN Program Realization

The main focus here is to show the advantages of multi-dimensional numbered information spaces in the process of realization of multi-layer structure of MPGN.

6.4.1 Training Process

Very important aspect is that for each class there exists separate class space, which has multilayer structure called "pyramids". All layers have equal structure and consist of "pattern-set" and "link-space". For each class space a "vertex-set" also is kept, which is used in the recognition stage.

- ***Construction Elements***

  - ✓ ***The Pattern-Set***

Each pattern belongs to a definite class c and layer l. The full denotation of pattern should be $P(c,l)$ in order to be clear in which class this pattern belongs to (note that c is class value of the pattern). We omit c whenever it is clear from the context and will denote $P(l)$. When l is also clear from the context we will denote only P.

All patterns of the class c, which belongs to layer l form theirs pattern-set: $PS(c,l) = \{P^i(c,l) \mid i = 1,...,n_{c,l}\}$. Each pattern $P(c,l)$ from $PS(c,l)$ have identifiers $pid(P,c,l)$ or shortly $pid(P)$ where it is clear, which are natural numbers. The identifiers are created in increasing order of incoming the patterns into pattern-set.

The process of generalization creates "vertical" interconnections between patterns from different (neighborhood) layers. These interconnections for every pattern are represented by two sets of "predecessors" and "successors".

The predecessors' set $PredS(P^i)$ contains the identifiers of patterns from the lower layer, which were used in the process of receiving this pattern. The predecessors sets for instances of layer one are empty.

The successors' set $SuccS(P^i)$ contains the identifiers of patterns from the upper layer, which are created by this pattern. The successors' sets of patterns on the top of the pyramid are empty. These patterns are called "vertexes" of the corresponded pyramids.

One pattern may be included in more than one pyramid. The vertex pattern belongs only to one pyramid (they became top of the pyramids).

✓ ***The Link-Space***

The goal of the Link-space is to describe all regularities between attributes which are available in the classes. Links to the patterns, which contain it, are created for every value of each attribute thus allowing to create a hierarchy of sets. The structure of this hierarchy is as follows:

- attribute value set: a set of class sets for given value of given attribute;
- attribute set: a set of attribute value sets for given attribute;
- link-space (one): a set of all possible attribute sets.

Creation of link-space uses the advantages of multi-dimensional numbered information spaces, especially the possibility to overcome searching by using direct pointing via coordinate addresses.

Below we will focus our attention on the link-space, which becomes a key element of accelerating the creation of new patterns as well as searching patterns satisfying the queries.

Let c be a number of examined class and l be the number of given layer of c:

- attribute value set $VS(c,l,t,v)$, $v=1,...,n_t$ is a set of all identifiers of instances/patterns for class c, layer l, which have value v for the attribute t: $VS(c,l,t,v)=\{pid(P^i,c,l),\ i=0,..,x \mid a_t^i=v\}$;
- attribute set $AS(c,l,a)$ for concrete attribute $a=1,...,n$ is a set of attribute value sets for class c, layer l and attribute t: $AS(c,l,t)=\{VS(c,l,t,1),...,VS(c,l,t,n_t)\}$, where n_t is the number of values of attribute t;
- link-space $LS(c,l)$ is a set of all possible attribute sets for class c and layer l: $LS(c,l)=\{AS(c,l,1),...,AS(c,l,n)\}$.

In Figure 24, the visualization of class 3 of the "Lenses" dataset during creation of the patterns is shown.

Such information is stored in ArM-structures by a very simple convention – the attribute value sets $VS(c,l,t,v)$ is stored in the points of ArM-archive using the corresponding address $(4,c,l,t,v)$, where 4 is the dimension of ArM space, c is the number of the class, l is the number of the layer, t is the number of the attribute and v is the number of the corresponding values of the given attribute. The disposition of link-spaces in ArM-structures allows very fast extraction of available patterns in the corresponding layer and class.

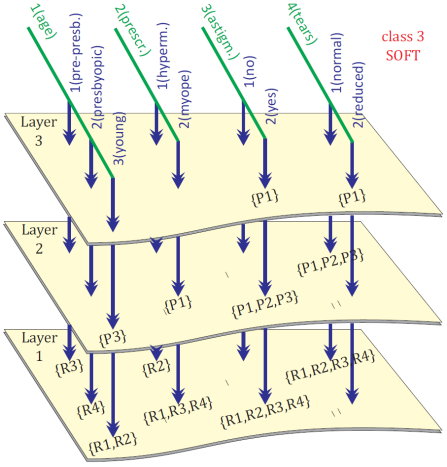


Figure 24. Visualization of link-spaces

✓ ***Vertex-Set***

The vertex-set contains information about the patterns that have not successors in the pyramids of the corresponded class.

$$VrS(c)=\{pid(P^i,c,l)\,|\,i=1,...,n_{c,l};l=1,...,l_{\max}:SuccS(P^i)=\varnothing\}.$$

➢ ***Generation of the Rules***

The process of rules generation is a chain of creating the patterns of the upper layer as an intersection between patterns from lower layer until new patterns are generated.

Each instance $R=(c\,|\,a_1,...,a_n)$ from the learning set is included into the pattern-set of the first layer of its class c: $R\in PS(c,1)$.

Starting from layer $l=2$ the following steps are made:

1. Creating the link-space of the lower layer $l-1$ of class c with adding the identifiers of patterns $P^i\in PS(c,l-1)$, $i=1,...,n_{c,l-1}$ in the attribute value sets of the values: $pid(P^i)\in VS(c,l-1,k^i,a_k^i),\ k=1,...,n$. Let remark that this sets became ordered during creation;

2. The set of patterns, which are intersections of the patterns of the lower layer $P^i \cap P^j$, $P^i, P^j \in PS(c,l-1)$, $i,j=1,...,k_{c,l-1}$, $i \neq j$ is created. The algorithm of creating of this set is given below;
3. If patterns are not generated (i.e. $PS(c,l)=\{\}$), then the process of generation the rules for this class stops;
4. On the basis of receiving a set of patterns, the pattern-set $PS(c,l)$ of layer l is created.
   - Each pattern from the receiving set of patterns is checked for existence in the $PS(c,l)$;
   - If this pattern does not exist in $PS(c,l)$, it receives an identifier which is equal to the next number of identifiers of the patterns in the pattern-set; the pattern is added at the end of the pattern-set; and its predecessor-set is created with two pairs $\{(pid(P^i),l-1)$, $(pid(P^j),l-1)\}$;
   - If this pattern already exists in $PS(c,l)$, its predecessor-set is formed as union of existent predecessor-set and $\{(pid(P^i),l-1)$, $(pid(P^j),l-1)\}$.
5. For each pattern from layer l check for existence the same pattern in lower layers (from layer $l-1$ to layer 2). If such pattern exists, then it is removed from the pattern-set and corresponded link-space of lower layer and the predecessor-set of current pattern is enriched with predecessor-set of removed pattern (by union).
6. Incrementing layer l and repeating the process.

As a result, for every class a separate pyramidal network structure is built. Each pyramid is described by a predecessor-set and a successor-set of patterns in neighbor layers.

- ***Generating the Set of Patterns, which are Intersections of Patterns from the Lower Level***

For restriction the exponential growth of intersections in program realization there are included two parameters:

- L1 (from 0 to 100) – percentage of reduced patterns per layer;
- L2 (from 0 to 100) – minimal ratio in percent between cardinality of the generated pattern toward maximal cardinality of patterns in lower layer.

The process of generation the intersections of the patterns from given pattern-set $PS(c,l)$ loops each pattern $P^i \in PS(c,l)$.

For this pattern $P^i = (c \mid a_1^i,...,a_n^i)$ the generation of possible patterns is:

1. An empty set of resulting patterns is created;
2. For all attribute values $a_1^i,...,a_n^i$ different from "-" of P^i we take corresponding attribute-value-sets $VS(c,l,k^i,a_k^i),\ i=1,...,n$. The numbers of identifiers of the patterns in these sets are ordered.
3. All extracted attribute-value-sets are activated.
4. From each of them the first identifier is given.
5. While at least one attribute-value-set is active, the following steps are made:
   - Assign the initial values of the resulting pattern: $V = (c \mid -,-,...,-)$;
   - Locate minimal identifier $pid(P^j)$ from all active attribute-value-sets;
   - If $pid(P^j) = pid(P^i)$, then this attribute-value-set is deactivated;
   - All active attribute-value-sets $VS(c,l,k^j,a_k^j),\ j=1,...,k_{c,l}$, for which $pid(P^j)$ is current identifier, cause filling of corresponded attribute value a_k^i of k^{th} attribute in V. For these sets the next identifier is given;
   - If $|V|>0$ and $\min\left(\frac{|V|}{|P^i|},\frac{|V|}{|P^j|}\right) > L2$ this pattern is included into the set of resulting patterns with additional information, containing $pid(P^i)$ and $pid(P^j)$.

This process is illustrated in Figure 25.

At the end the patterns in the created pattern set are sorted by the number of their predecessors and L1% of them with lower number of predecessors are removed.

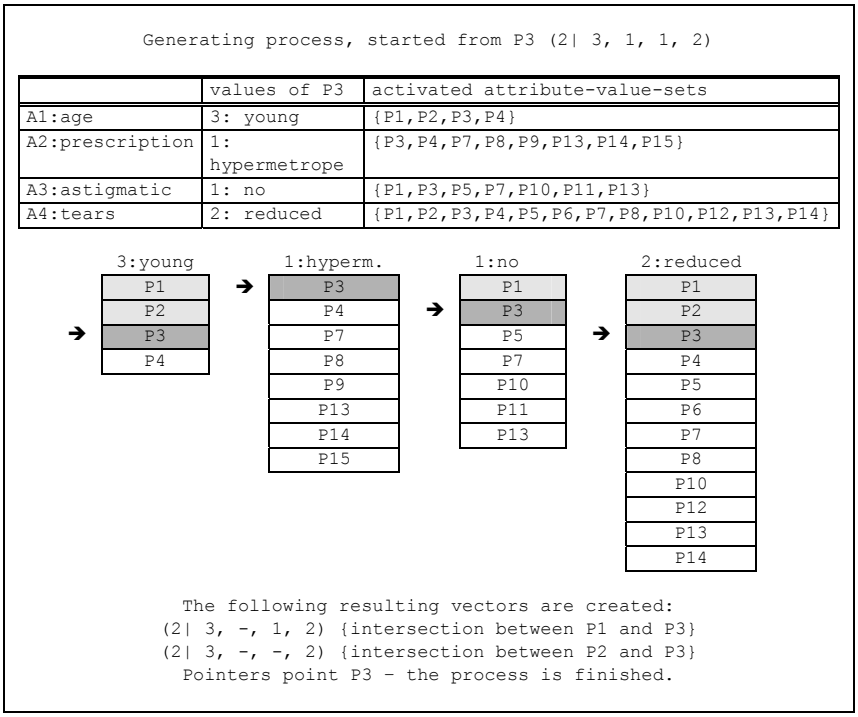


Figure 25. Visualization of process of generating a set of patterns

6.4.2 Recognition Process

The record to be recognized is represented by the values of its attributes $Q = (?\,|\, b_1,\ldots,b_n)$. Some of the features may be omitted. The classification stage consists of several steps:

1. Using the service attribute-values-space, the system takes corresponded attribute value sets for all attributes $b_1,\ldots,b_n$ as well as the attribute value sets for "-" as a value of each attribute.
2. The union of these sets gives a set of possible classes $\{c_1,\ldots,c_y\}$, which the record may belong to. This approach decreases the amount of the information, needed for pattern recognition.
3. All classes, which are presented in this union $\{c_1,\ldots,c_y\}$ are scanned in parallel. For each class c_x and for each layer of class space of c_x the following steps are done:

- For all attribute values $b_1,...,b_n$ different from "-" of Q we obtain the corresponding attribute value sets from link-space of class c_x of current layer;
- The intersection between all these sets is made. As a result a recognition set of candidate patterns is created. If this set is empty, the target class is class with maximal support;
- For each pattern P, which is a member of the recognition set, calculate $IntersectPerc(P,Q)$.

4. From all recognition sets of the classes and layers the patterns with maximal cardinality are found.
5. These recognition sets are lightened with excluding the patterns, which cardinality are less than maximal cardinality. The new set of classes-potential answers $\{c_1,...,c_{y'}\}$ contains only classes, which recognition sets are not empty.
6. If only one class is in the set of classes-potential answers, then this is the target class and the process stops.
7. Otherwise, if this set is empty, we give again primary set of classes-potential answers $\{c_1,...,c_{y'}\} = \{c_1,...,c_y\}$ and the process continues with examining this set.
8. Examine $\{c_1,...,c_{y'}\}$:
   - for each class which is a member of this set, the number of instances with maximal intersection percentage with the query is found and the ratio between these number and all instances in the class is calculated;
   - the maximum of intersection percentages from all classes is determinate and in the set $\{c_1,...,c_{y'}\}$ only classes with this maximal percentage and maximal ratio is remained;
   - if $\{c_1,...,c_{y'}\}$ contains only one class – the class is given as answer. Otherwise the class of $\{c_1,...,c_{y'}\}$ with maximal instances is given as an answer. And the process stops.

We can assume that the possibilities to keep in a manageable way numerous created patterns allows to use this environment to test different kinds of recognition models.

Here the algorithm of recognition strategy S1 was described. The algorithm for strategy S2 differs only in points 4 and 5, where the criteria for selection of classes-candidates are so called "confidence of recognition set".

Conclusion

We presented the structure and functionality of the software realization of MPGN algorithm.

The realization of the proposed algorithms PGN and MPGN were made in a common data mining analysis environment PaGaNe.

Because PGN and MPGN deal with categorical attributes, different discretization methods are implemented as a preprocessing step.

The basic construction elements, used in the realization of MPGN, which keep vertical connections between patterns: the pattern-set, the vertex-set and the link-space have been presented.

In addition, the algorithm which realizes smart storing and extracting of patterns from these structures, using advantages of context-free access method, realized in ArM 32 have been discussed.

7 Example on Lenses Dataset

Abstract

This chapter provides examples of the use of PGN and MPGN algorithms on the "Lenses" dataset.

7.1 Lenses Dataset

Lenses dataset is provided from the UCI machine learning repository as the simplest example which can be used to illustrate the steps in the data mining algorithms.

Object	class	age	prescription	astigmatic	tears
1	none	young	myope	no	reduced
2	soft	young	myope	no	normal
3	none	young	myope	yes	reduced
4	hard	young	myope	yes	normal
5	none	young	hypermetrope	no	reduced
6	soft	young	hypermetrope	no	normal
7	none	young	hypermetrope	yes	reduced
8	hard	young	hypermetrope	yes	normal
9	none	pre-presbyopic	myope	no	reduced
10	soft	pre-presbyopic	myope	no	normal
11	none	pre-presbyopic	myope	yes	reduced
12	hard	pre-presbyopic	myope	yes	normal
13	none	pre-presbyopic	hypermetrope	no	reduced
14	soft	pre-presbyopic	hypermetrope	no	normal
15	none	pre-presbyopic	hypermetrope	yes	reduced
16	none	pre-presbyopic	hypermetrope	yes	normal
17	none	presbyopic	myope	no	reduced
18	none	presbyopic	myope	no	normal
19	none	presbyopic	myope	yes	reduced
20	hard	presbyopic	myope	yes	normal
21	none	presbyopic	hypermetrope	no	reduced
22	soft	presbyopic	hypermetrope	no	normal
23	none	presbyopic	hypermetrope	yes	reduced
24	none	presbyopic	hypermetrope	yes	normal

During the input of instances of "Lenses" database, the following numbering is created:

class	
hard	1
none	2
soft	3

age	
pre-presbyopic	1
presbyopic	2
young	3

prescription	
hypermetrope	1
myope	2

astigmatic	
no	1
yes	2

tears	
normal	1
reduced	2

As a result, instances are presented as numerical vectors juxtaposing each attribute value with the corresponding number.

For example, the instance (none|young,myope,no,reduced) is converted to (2|3,2,1,2). For more readability we use "|" for separating class label from other attributes.

7.2 PGN

Here we will present the behavior of training and recognition processes in PGN on the example of Lenses dataset.

The description of Lenses dataset in UCI repository [Frank and Asuncion, 2010] argues that the dataset is complete (all possible combinations of attribute-value pairs are represented), each instance is complete and correct and 9 rules cover the training set.

Our goal is to show that PGN extracts these 9 rules.

7.2.1 Training Process in PGN

The training process in PGN consists of two steps that are usual for CAR classifiers.

- generalization;
- pruning.

In PGN pruning is post-processing phase after generalization.

➢ ***Step 1: Generalization***

The step of generalization tries to extract each possible intersections between instances and patterns within the class.

✓ ***Sub-step 1.1: Adding Instances***

In this sub-step the instances are added incrementally into pattern set as primary patterns.

$pid(R)$	R
[1]	(2| 3, 2, 1, 2)
[2]	(3| 3, 2, 1, 1)
[3]	(2| 3, 2, 2, 2)
[4]	(1| 3, 2, 2, 1)
[5]	(2| 3, 1, 1, 2)
[6]	(3| 3, 1, 1, 1)
[7]	(2| 3, 1, 2, 2)
[8]	(1| 3, 1, 2, 1)
[9]	(2| 1, 2, 1, 2)
[10]	(3| 1, 2, 1, 1)
[11]	(2| 1, 2, 2, 2)
[12]	(1| 1, 2, 2, 1)
[13]	(2| 1, 1, 1, 2)
[14]	(3| 1, 1, 1, 1)
[15]	(2| 1, 1, 2, 2)
[16]	(2| 1, 1, 2, 1)
[17]	(2| 2, 2, 1, 2)
[18]	(2| 2, 2, 1, 1)
[19]	(2| 2, 2, 2, 2)
[20]	(1| 2, 2, 2, 1)
[21]	(2| 2, 1, 1, 2)
[22]	(3| 2, 1, 1, 1)
[23]	(2| 2, 1, 2, 2)
[24]	(2| 2, 1, 2, 1)

✓ *Sub-step 1.2: Intersections within the Classes*

In this sub-step the intersections between each two patterns that belong to the same class are made.

If a new pattern is created it is added into the pattern set. For example, the intersection between $R_1 = (2 \mid 3,2,1,2)$ and $R_3 = (2 \mid 3,2,2,2)$ creates a new pattern $R_{25} = (2 \mid 3,2,\_,2)$.

If the pattern already exists, only the set of instances that are possible creators of the pattern is expanded. For example, $R_1 = (2 \mid 3,2,1,2) \cap R_{17} = (2 \mid 2,2,1,2)$ creates a pattern $R_{28} = (2 \mid \_,2,1,2)$ already created by $R_1 = (2 \mid 3,2,1,2) \cap R_9 = (2 \mid 1,2,1,2)$.

It is possible the intersection does not to produce a pattern when all attribute values in two patterns differs. It is seen in the case of $R_1 = (2 \mid 3,2,1,2) \cap R_{16} = (2 \mid 1,1,2,1) = \varnothing$.

$pid(P_1)$	P_1	$pid(P_2)$	P_2	$pid(P_1 \cap P_2)$	$P_1 \cap P_2$
[1]	(2| 3, 2, 1, 2)	[3]	(2| 3, 2, 2, 2)	[25]	(2| 3, 2, _, 2)
[1]	(2| 3, 2, 1, 2)	[5]	(2| 3, 1, 1, 2)	[26]	(2| 3, _, 1, 2)
[1]	(2| 3, 2, 1, 2)	[7]	(2| 3, 1, 2, 2)	[27]	(2| 3, _, _, 2)
[1]	(2| 3, 2, 1, 2)	[9]	(2| 1, 2, 1, 2)	[28]	(2| _, 2, 1, 2)
[1]	(2| 3, 2, 1, 2)	[11]	(2| 1, 2, 2, 2)	[29]	(2| _, 2, _, 2)
[1]	(2| 3, 2, 1, 2)	[13]	(2| 1, 1, 1, 2)	[30]	(2| _, _, 1, 2)

```
[  1]  (2|  3,  2,  1,  2)  [ 15]  (2|  1,  1,  2,  2)  [ 31]       (2|  _,  _,  _,  2)
[  1]  (2|  3,  2,  1,  2)  [ 16]  (2|  1,  1,  2,  1)  [   ]       0
[  1]  (2|  3,  2,  1,  2)  [ 17]  (2|  2,  2,  1,  2)  [ 28]       (2|  _,  2,  1,  2)
[  1]  (2|  3,  2,  1,  2)  [ 18]  (2|  2,  2,  1,  1)  [ 32]       (2|  _,  2,  1,  _)
[  1]  (2|  3,  2,  1,  2)  [ 19]  (2|  2,  2,  2,  2)  [ 29]       (2|  _,  2,  _,  2)
[  1]  (2|  3,  2,  1,  2)  [ 21]  (2|  2,  1,  1,  2)  [ 30]       (2|  _,  _,  1,  2)
[  1]  (2|  3,  2,  1,  2)  [ 23]  (2|  2,  1,  2,  2)  [ 31]       (2|  _,  _,  _,  2)
[  1]  (2|  3,  2,  1,  2)  [ 24]  (2|  2,  1,  2,  1)  [   ]       0
[  2]  (3|  3,  2,  1,  1)  [  6]  (3|  3,  1,  1,  1)  [ 33]       (3|  3,  _,  1,  1)
[  2]  (3|  3,  2,  1,  1)  [ 10]  (3|  1,  2,  1,  1)  [ 34]       (3|  _,  2,  1,  1)
[  2]  (3|  3,  2,  1,  1)  [ 14]  (3|  1,  1,  1,  1)  [ 35]       (3|  _,  _,  1,  1)
[  2]  (3|  3,  2,  1,  1)  [ 22]  (3|  2,  1,  1,  1)  [ 35]       (3|  _,  _,  1,  1)
[  3]  (2|  3,  2,  2,  2)  [  5]  (2|  3,  1,  1,  2)  [ 27]       (2|  3,  _,  _,  2)
[  3]  (2|  3,  2,  2,  2)  [  7]  (2|  3,  1,  2,  2)  [ 36]       (2|  3,  _,  2,  2)
[  3]  (2|  3,  2,  2,  2)  [  9]  (2|  1,  2,  1,  2)  [ 29]       (2|  _,  2,  _,  2)
[  3]  (2|  3,  2,  2,  2)  [ 11]  (2|  1,  2,  2,  2)  [ 37]       (2|  _,  2,  2,  2)
[  3]  (2|  3,  2,  2,  2)  [ 13]  (2|  1,  1,  1,  2)  [ 31]       (2|  _,  _,  _,  2)
[  3]  (2|  3,  2,  2,  2)  [ 15]  (2|  1,  1,  2,  2)  [ 38]       (2|  _,  _,  2,  2)
[  3]  (2|  3,  2,  2,  2)  [ 16]  (2|  1,  1,  2,  1)  [ 39]       (2|  _,  _,  2,  _)
[  3]  (2|  3,  2,  2,  2)  [ 17]  (2|  2,  2,  1,  2)  [ 29]       (2|  _,  2,  _,  2)
[  3]  (2|  3,  2,  2,  2)  [ 18]  (2|  2,  2,  1,  1)  [ 40]       (2|  _,  2,  _,  _)
[  3]  (2|  3,  2,  2,  2)  [ 19]  (2|  2,  2,  2,  2)  [ 37]       (2|  _,  2,  2,  2)
[  3]  (2|  3,  2,  2,  2)  [ 21]  (2|  2,  1,  1,  2)  [ 31]       (2|  _,  _,  _,  2)
[  3]  (2|  3,  2,  2,  2)  [ 23]  (2|  2,  1,  2,  2)  [ 38]       (2|  _,  _,  2,  2)
[  3]  (2|  3,  2,  2,  2)  [ 24]  (2|  2,  1,  2,  1)  [ 39]       (2|  _,  _,  2,  _)
[  4]  (1|  3,  2,  2,  1)  [  8]  (1|  3,  1,  2,  1)  [ 41]       (1|  3,  _,  2,  1)
[  4]  (1|  3,  2,  2,  1)  [ 12]  (1|  1,  2,  2,  1)  [ 42]       (1|  _,  2,  2,  1)
[  4]  (1|  3,  2,  2,  1)  [ 20]  (1|  2,  2,  2,  1)  [ 42]       (1|  _,  2,  2,  1)
[  5]  (2|  3,  1,  1,  2)  [  7]  (2|  3,  1,  2,  2)  [ 43]       (2|  3,  1,  _,  2)
[  5]  (2|  3,  1,  1,  2)  [  9]  (2|  1,  2,  1,  2)  [ 30]       (2|  _,  _,  1,  2)
[  5]  (2|  3,  1,  1,  2)  [ 11]  (2|  1,  2,  2,  2)  [ 31]       (2|  _,  _,  _,  2)
[  5]  (2|  3,  1,  1,  2)  [ 13]  (2|  1,  1,  1,  2)  [ 44]       (2|  _,  1,  1,  2)
[  5]  (2|  3,  1,  1,  2)  [ 15]  (2|  1,  1,  2,  2)  [ 45]       (2|  _,  1,  _,  2)
[  5]  (2|  3,  1,  1,  2)  [ 16]  (2|  1,  1,  2,  1)  [ 46]       (2|  _,  1,  _,  _)
[  5]  (2|  3,  1,  1,  2)  [ 17]  (2|  2,  2,  1,  2)  [ 30]       (2|  _,  _,  1,  2)
[  5]  (2|  3,  1,  1,  2)  [ 18]  (2|  2,  2,  1,  1)  [ 47]       (2|  _,  _,  1,  _)
[  5]  (2|  3,  1,  1,  2)  [ 19]  (2|  2,  2,  2,  2)  [ 31]       (2|  _,  _,  _,  2)
[  5]  (2|  3,  1,  1,  2)  [ 21]  (2|  2,  1,  1,  2)  [ 44]       (2|  _,  1,  1,  2)
[  5]  (2|  3,  1,  1,  2)  [ 23]  (2|  2,  1,  2,  2)  [ 45]       (2|  _,  1,  _,  2)
[  5]  (2|  3,  1,  1,  2)  [ 24]  (2|  2,  1,  2,  1)  [ 46]       (2|  _,  1,  _,  _)
[  6]  (3|  3,  1,  1,  1)  [ 10]  (3|  1,  2,  1,  1)  [ 35]       (3|  _,  _,  1,  1)
[  6]  (3|  3,  1,  1,  1)  [ 14]  (3|  1,  1,  1,  1)  [ 48]       (3|  _,  1,  1,  1)
[  6]  (3|  3,  1,  1,  1)  [ 22]  (3|  2,  1,  1,  1)  [ 48]       (3|  _,  1,  1,  1)
[  7]  (2|  3,  1,  2,  2)  [  9]  (2|  1,  2,  1,  2)  [ 31]       (2|  _,  _,  _,  2)
[  7]  (2|  3,  1,  2,  2)  [ 11]  (2|  1,  2,  2,  2)  [ 38]       (2|  _,  _,  2,  2)
[  7]  (2|  3,  1,  2,  2)  [ 13]  (2|  1,  1,  1,  2)  [ 45]       (2|  _,  1,  _,  2)
[  7]  (2|  3,  1,  2,  2)  [ 15]  (2|  1,  1,  2,  2)  [ 49]       (2|  _,  1,  2,  2)
[  7]  (2|  3,  1,  2,  2)  [ 16]  (2|  1,  1,  2,  1)  [ 50]       (2|  _,  1,  2,  _)
[  7]  (2|  3,  1,  2,  2)  [ 17]  (2|  2,  2,  1,  2)  [ 31]       (2|  _,  _,  _,  2)
[  7]  (2|  3,  1,  2,  2)  [ 18]  (2|  2,  2,  1,  1)  [   ]       0
[  7]  (2|  3,  1,  2,  2)  [ 19]  (2|  2,  2,  2,  2)  [ 38]       (2|  _,  _,  2,  2)
[  7]  (2|  3,  1,  2,  2)  [ 21]  (2|  2,  1,  1,  2)  [ 45]       (2|  _,  1,  _,  2)
[  7]  (2|  3,  1,  2,  2)  [ 23]  (2|  2,  1,  2,  2)  [ 49]       (2|  _,  1,  2,  2)
[  7]  (2|  3,  1,  2,  2)  [ 24]  (2|  2,  1,  2,  1)  [ 50]       (2|  _,  1,  2,  _)
[  8]  (1|  3,  1,  2,  1)  [ 12]  (1|  1,  2,  2,  1)  [ 51]       (1|  _,  _,  2,  1)
[  8]  (1|  3,  1,  2,  1)  [ 20]  (1|  2,  2,  2,  1)  [ 51]       (1|  _,  _,  2,  1)
[  9]  (2|  1,  2,  1,  2)  [ 11]  (2|  1,  2,  2,  2)  [ 52]       (2|  1,  2,  _,  2)
[  9]  (2|  1,  2,  1,  2)  [ 13]  (2|  1,  1,  1,  2)  [ 53]       (2|  1,  _,  1,  2)
[  9]  (2|  1,  2,  1,  2)  [ 15]  (2|  1,  1,  2,  2)  [ 54]       (2|  1,  _,  _,  2)
[  9]  (2|  1,  2,  1,  2)  [ 16]  (2|  1,  1,  2,  1)  [ 55]       (2|  1,  _,  _,  _)
[  9]  (2|  1,  2,  1,  2)  [ 17]  (2|  2,  2,  1,  2)  [ 28]       (2|  _,  2,  1,  2)
```

```
[  9]  (2| 1, 2, 1, 2) [ 18]  (2| 2, 2, 1, 1) [ 32]      (2| _, 2, 1, _)
[  9]  (2| 1, 2, 1, 2) [ 19]  (2| 2, 2, 2, 2) [ 29]      (2| _, 2, _, 2)
[  9]  (2| 1, 2, 1, 2) [ 21]  (2| 2, 1, 1, 2) [ 30]      (2| _, _, 1, 2)
[  9]  (2| 1, 2, 1, 2) [ 23]  (2| 2, 1, 2, 2) [ 31]      (2| _, _, _, 2)
[  9]  (2| 1, 2, 1, 2) [ 24]  (2| 2, 1, 2, 1) [   ]      0
[ 10]  (3| 1, 2, 1, 1) [ 14]  (3| 1, 1, 1, 1) [ 56]      (3| 1, _, 1, 1)
[ 10]  (3| 1, 2, 1, 1) [ 22]  (3| 2, 1, 1, 1) [ 35]      (3| _, _, 1, 1)
[ 11]  (2| 1, 2, 2, 2) [ 13]  (2| 1, 1, 1, 2) [ 54]      (2| 1, _, _, 2)
[ 11]  (2| 1, 2, 2, 2) [ 15]  (2| 1, 1, 2, 2) [ 57]      (2| 1, _, 2, 2)
[ 11]  (2| 1, 2, 2, 2) [ 16]  (2| 1, 1, 2, 1) [ 58]      (2| 1, _, 2, _)
[ 11]  (2| 1, 2, 2, 2) [ 17]  (2| 2, 2, 1, 2) [ 29]      (2| _, 2, _, 2)
[ 11]  (2| 1, 2, 2, 2) [ 18]  (2| 2, 2, 1, 1) [ 40]      (2| _, 2, _, _)
[ 11]  (2| 1, 2, 2, 2) [ 19]  (2| 2, 2, 2, 2) [ 37]      (2| _, 2, 2, 2)
[ 11]  (2| 1, 2, 2, 2) [ 21]  (2| 2, 1, 1, 2) [ 31]      (2| _, _, _, 2)
[ 11]  (2| 1, 2, 2, 2) [ 23]  (2| 2, 1, 2, 2) [ 38]      (2| _, _, 2, 2)
[ 11]  (2| 1, 2, 2, 2) [ 24]  (2| 2, 1, 2, 1) [ 39]      (2| _, _, 2, _)
[ 12]  (1| 1, 2, 2, 1) [ 20]  (1| 2, 2, 2, 1) [ 42]      (1| _, 2, 2, 1)
[ 13]  (2| 1, 1, 1, 2) [ 15]  (2| 1, 1, 2, 2) [ 59]      (2| 1, 1, _, 2)
[ 13]  (2| 1, 1, 1, 2) [ 16]  (2| 1, 1, 2, 1) [ 60]      (2| 1, 1, _, _)
[ 13]  (2| 1, 1, 1, 2) [ 17]  (2| 2, 2, 1, 2) [ 30]      (2| _, _, 1, 2)
[ 13]  (2| 1, 1, 1, 2) [ 18]  (2| 2, 2, 1, 1) [ 47]      (2| _, _, 1, _)
[ 13]  (2| 1, 1, 1, 2) [ 19]  (2| 2, 2, 2, 2) [ 31]      (2| _, _, _, 2)
[ 13]  (2| 1, 1, 1, 2) [ 21]  (2| 2, 1, 1, 2) [ 44]      (2| _, 1, 1, 2)
[ 13]  (2| 1, 1, 1, 2) [ 23]  (2| 2, 1, 2, 2) [ 45]      (2| _, 1, _, 2)
[ 13]  (2| 1, 1, 1, 2) [ 24]  (2| 2, 1, 2, 1) [ 46]      (2| _, 1, _, _)
[ 14]  (3| 1, 1, 1, 1) [ 22]  (3| 2, 1, 1, 1) [ 48]      (3| _, 1, 1, 1)
[ 15]  (2| 1, 1, 2, 2) [ 16]  (2| 1, 1, 2, 1) [ 61]      (2| 1, 1, 2, _)
[ 15]  (2| 1, 1, 2, 2) [ 17]  (2| 2, 2, 1, 2) [ 31]      (2| _, _, _, 2)
[ 15]  (2| 1, 1, 2, 2) [ 18]  (2| 2, 2, 1, 1) [   ]      0
[ 15]  (2| 1, 1, 2, 2) [ 19]  (2| 2, 2, 2, 2) [ 38]      (2| _, _, 2, 2)
[ 15]  (2| 1, 1, 2, 2) [ 21]  (2| 2, 1, 1, 2) [ 45]      (2| _, 1, _, 2)
[ 15]  (2| 1, 1, 2, 2) [ 23]  (2| 2, 1, 2, 2) [ 49]      (2| _, 1, 2, 2)
[ 15]  (2| 1, 1, 2, 2) [ 24]  (2| 2, 1, 2, 1) [ 50]      (2| _, 1, 2, _)
[ 16]  (2| 1, 1, 2, 1) [ 17]  (2| 2, 2, 1, 2) [   ]      0
[ 16]  (2| 1, 1, 2, 1) [ 18]  (2| 2, 2, 1, 1) [ 62]      (2| _, _, _, 1)
[ 16]  (2| 1, 1, 2, 1) [ 19]  (2| 2, 2, 2, 2) [ 39]      (2| _, _, 2, _)
[ 16]  (2| 1, 1, 2, 1) [ 21]  (2| 2, 1, 1, 2) [ 46]      (2| _, 1, _, _)
[ 16]  (2| 1, 1, 2, 1) [ 23]  (2| 2, 1, 2, 2) [ 50]      (2| _, 1, 2, _)
[ 16]  (2| 1, 1, 2, 1) [ 24]  (2| 2, 1, 2, 1) [ 63]      (2| _, 1, 2, 1)
[ 17]  (2| 2, 2, 1, 2) [ 18]  (2| 2, 2, 1, 1) [ 64]      (2| 2, 2, 1, _)
[ 17]  (2| 2, 2, 1, 2) [ 19]  (2| 2, 2, 2, 2) [ 65]      (2| 2, 2, _, 2)
[ 17]  (2| 2, 2, 1, 2) [ 21]  (2| 2, 1, 1, 2) [ 66]      (2| 2, _, 1, 2)
[ 17]  (2| 2, 2, 1, 2) [ 23]  (2| 2, 1, 2, 2) [ 67]      (2| 2, _, _, 2)
[ 17]  (2| 2, 2, 1, 2) [ 24]  (2| 2, 1, 2, 1) [ 68]      (2| 2, _, _, _)
[ 18]  (2| 2, 2, 1, 1) [ 19]  (2| 2, 2, 2, 2) [ 69]      (2| 2, 2, _, _)
[ 18]  (2| 2, 2, 1, 1) [ 21]  (2| 2, 1, 1, 2) [ 70]      (2| 2, _, 1, _)
[ 18]  (2| 2, 2, 1, 1) [ 23]  (2| 2, 1, 2, 2) [ 68]      (2| 2, _, _, _)
[ 18]  (2| 2, 2, 1, 1) [ 24]  (2| 2, 1, 2, 1) [ 71]      (2| 2, _, _, 1)
[ 19]  (2| 2, 2, 2, 2) [ 21]  (2| 2, 1, 1, 2) [ 67]      (2| 2, _, _, 2)
[ 19]  (2| 2, 2, 2, 2) [ 23]  (2| 2, 1, 2, 2) [ 72]      (2| 2, _, 2, 2)
[ 19]  (2| 2, 2, 2, 2) [ 24]  (2| 2, 1, 2, 1) [ 73]      (2| 2, _, 2, _)
[ 21]  (2| 2, 1, 1, 2) [ 23]  (2| 2, 1, 2, 2) [ 74]      (2| 2, 1, _, 2)
[ 21]  (2| 2, 1, 1, 2) [ 24]  (2| 2, 1, 2, 1) [ 75]      (2| 2, 1, _, _)
[ 23]  (2| 2, 1, 2, 2) [ 24]  (2| 2, 1, 2, 1) [ 76]      (2| 2, 1, 2, _)
```

As a result of step 1, in the pattern set the following patterns are created and corresponded sets of instances, creators of these patterns, are gathered:

$pid(P)$	P	set of instances-creators of the pattern $\{pid(R_i)\}$
[1]	(2| 3, 2, 1, 2)	{1}
[2]	(3| 3, 2, 1, 1)	{2}
[3]	(2| 3, 2, 2, 2)	{3}
[4]	(1| 3, 2, 2, 1)	{4}
[5]	(2| 3, 1, 1, 2)	{5}
[6]	(3| 3, 1, 1, 1)	{6}
[7]	(2| 3, 1, 2, 2)	{7}
[8]	(1| 3, 1, 2, 1)	{8}
[9]	(2| 1, 2, 1, 2)	{9}
[10]	(3| 1, 2, 1, 1)	{10}
[11]	(2| 1, 2, 2, 2)	{11}
[12]	(1| 1, 2, 2, 1)	{12}
[13]	(2| 1, 1, 1, 2)	{13}
[14]	(3| 1, 1, 1, 1)	{14}
[15]	(2| 1, 1, 2, 2)	{15}
[16]	(2| 1, 1, 2, 1)	{16}
[17]	(2| 2, 2, 1, 2)	{17}
[18]	(2| 2, 2, 1, 1)	{18}
[19]	(2| 2, 2, 2, 2)	{19}
[20]	(1| 2, 2, 2, 1)	{20}
[21]	(2| 2, 1, 1, 2)	{21}
[22]	(3| 2, 1, 1, 1)	{22}
[23]	(2| 2, 1, 2, 2)	{23}
[24]	(2| 2, 1, 2, 1)	{24}
[25]	(2| 3, 2, _, 2)	{1, 3}
[26]	(2| 3, _, 1, 2)	{1, 5}
[27]	(2| 3, _, _, 2)	{1, 3, 5, 7}
[28]	(2| _, 2, 1, 2)	{1, 9, 17}
[29]	(2| _, 2, _, 2)	{1, 3, 9, 11, 17, 19}
[30]	(2| _, _, 1, 2)	{1, 5, 9, 13, 17, 21}
[31]	(2| _, _, _, 2)	{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23}
[32]	(2| _, 2, 1, _)	{1, 9, 18}
[33]	(3| 3, _, 1, 1)	{2, 6}
[34]	(3| _, 2, 1, 1)	{2, 10}
[35]	(3| _, _, 1, 1)	{2, 6, 10, 14, 22}
[36]	(2| 3, _, 2, 2)	{3, 7}
[37]	(2| _, 2, 2, 2)	{3, 11, 19}
[38]	(2| _, _, 2, 2)	{3, 7, 11, 15, 19, 23}
[39]	(2| _, _, 2, _)	{3, 11, 16, 19, 24}
[40]	(2| _, 2, _, _)	{3, 11, 18}
[41]	(1| 3, _, 2, 1)	{4, 8}
[42]	(1| _, 2, 2, 1)	{4, 12, 20}
[43]	(2| 3, 1, _, 2)	{5, 7}
[44]	(2| _, 1, 1, 2)	{5, 13, 21}
[45]	(2| _, 1, _, 2)	{5, 7, 13, 15, 21, 23}
[46]	(2| _, 1, _, _)	{5, 13, 16, 21, 24}
[47]	(2| _, _, 1, _)	{5, 13, 18}
[48]	(3| _, 1, 1, 1)	{6, 14, 22}
[49]	(2| _, 1, 2, 2)	{7, 15, 23}
[50]	(2| _, 1, 2, _)	{7, 15, 16, 23, 24}
[51]	(1| _, _, 2, 1)	{8, 12, 20}
[52]	(2| 1, 2, _, 2)	{9, 11}
[53]	(2| 1, _, 1, 2)	{9, 13}
[54]	(2| 1, _, _, 2)	{9, 11, 13, 15}
[55]	(2| 1, _, _, _)	{9, 16}

```
[ 56]   (3|  1,  _,  1,  1)  {10, 14}
[ 57]   (2|  1,  _,  2,  2)  {11, 15}
[ 58]   (2|  1,  _,  2,  _)  {11, 16}
[ 59]   (2|  1,  1,  _,  2)  {13, 15}
[ 60]   (2|  1,  1,  _,  _)  {13, 16}
[ 61]   (2|  1,  1,  2,  _)  {15, 16}
[ 62]   (2|  _,  _,  _,  1)  {16, 18}
[ 63]   (2|  _,  1,  2,  1)  {16, 24}
[ 64]   (2|  2,  2,  1,  _)  {17, 18}
[ 65]   (2|  2,  2,  _,  2)  {17, 19}
[ 66]   (2|  2,  _,  1,  2)  {17, 21}
[ 67]   (2|  2,  _,  _,  2)  {17, 19, 21, 23}
[ 68]   (2|  2,  _,  _,  _)  {17, 18, 23, 24}
[ 69]   (2|  2,  2,  _,  _)  {18, 19}
[ 70]   (2|  2,  _,  1,  _)  {18, 21}
[ 71]   (2|  2,  _,  _,  1)  {18, 24}
[ 72]   (2|  2,  _,  2,  2)  {19, 23}
[ 73]   (2|  2,  _,  2,  _)  {19, 24}
[ 74]   (2|  2,  1,  _,  2)  {21, 23}
[ 75]   (2|  2,  1,  _,  _)  {21, 24}
[ 76]   (2|  2,  1,  2,  _)  {23, 24}
```

- ***Step 2: Pruning***

Here the process of clearing exceptions between classes and lightening the pattern set is made.

- ✓ ***Sub-step 2.1: Check up for Data Consistency***

In this sub-step the patterns, belonging to different classes are paired. If one pattern matches another pattern (but they have a different class value), then the more general one is removed. If two patterns match each other then both of them are removed.

$pid(P_1)$	P_1	$pid(P_2)$	P_2	Pattern to be removed
[2]	(3| 3, 2, 1, 1)	[32]	(2| _, 2, 1, _)	[32] (2| _, 2, 1, _)
[2]	(3| 3, 2, 1, 1)	[40]	(2| _, 2, _, _)	[40] (2| _, 2, _, _)
[2]	(3| 3, 2, 1, 1)	[47]	(2| _, _, 1, _)	[47] (2| _, _, 1, _)
[2]	(3| 3, 2, 1, 1)	[62]	(2| _, _, _, 1)	[62] (2| _, _, _, 1)
[4]	(1| 3, 2, 2, 1)	[39]	(2| _, _, 2, _)	[39] (2| _, _, 2, _)
[6]	(3| 3, 1, 1, 1)	[46]	(2| _, 1, _, _)	[46] (2| _, 1, _, _)
[8]	(1| 3, 1, 2, 1)	[50]	(2| _, 1, 2, _)	[50] (2| _, 1, 2, _)
[8]	(1| 3, 1, 2, 1)	[63]	(2| _, 1, 2, 1)	[63] (2| _, 1, 2, 1)
[10]	(3| 1, 2, 1, 1)	[55]	(2| 1, _, _, _)	[55] (2| 1, _, _, _)
[12]	(1| 1, 2, 2, 1)	[58]	(2| 1, _, 2, _)	[58] (2| 1, _, 2, _)
[14]	(3| 1, 1, 1, 1)	[60]	(2| 1, 1, _, _)	[60] (2| 1, 1, _, _)
[16]	(2| 1, 1, 2, 1)	[51]	(1| _, _, 2, 1)	[51] (1| _, _, 2, 1)
[18]	(2| 2, 2, 1, 1)	[34]	(3| _, 2, 1, 1)	[34] (3| _, 2, 1, 1)
[18]	(2| 2, 2, 1, 1)	[35]	(3| _, _, 1, 1)	[35] (3| _, _, 1, 1)
[20]	(1| 2, 2, 2, 1)	[68]	(2| 2, _, _, _)	[68] (2| 2, _, _, _)
[20]	(1| 2, 2, 2, 1)	[69]	(2| 2, 2, _, _)	[69] (2| 2, 2, _, _)
[20]	(1| 2, 2, 2, 1)	[71]	(2| 2, _, _, 1)	[71] (2| 2, _, _, 1)
[20]	(1| 2, 2, 2, 1)	[73]	(2| 2, _, 2, _)	[73] (2| 2, _, 2, _)
[22]	(3| 2, 1, 1, 1)	[70]	(2| 2, _, 1, _)	[70] (2| 2, _, 1, _)
[22]	(3| 2, 1, 1, 1)	[75]	(2| 2, 1, _, _)	[75] (2| 2, 1, _, _)

✓ *Sub-step 2.2: Retain Most General Rules*

This sub-step is provided again within the classes. Patterns from equal classes are compared and, conversely to the previous step, more concrete patterns are deleted, i.e. the larger pattern is removed.

$pid(P_1)$	P_1	$pid(P_2)$	P_2	Pattern to be removed
[1]	(2| 3, 2, 1, 2)	[25]	(2| 3, 2, _, 2)	[1] (2| 3, 2, 1, 2)
[2]	(3| 3, 2, 1, 1)	[33]	(3| 3, _, 1, 1)	[2] (3| 3, 2, 1, 1)
[3]	(2| 3, 2, 2, 2)	[25]	(2| 3, 2, _, 2)	[3] (2| 3, 2, 2, 2)
[4]	(1| 3, 2, 2, 1)	[41]	(1| 3, _, 2, 1)	[4] (1| 3, 2, 2, 1)
[5]	(2| 3, 1, 1, 2)	[26]	(2| 3, _, 1, 2)	[5] (2| 3, 1, 1, 2)
[6]	(3| 3, 1, 1, 1)	[33]	(3| 3, _, 1, 1)	[6] (3| 3, 1, 1, 1)
[7]	(2| 3, 1, 2, 2)	[27]	(2| 3, _, _, 2)	[7] (2| 3, 1, 2, 2)
[8]	(1| 3, 1, 2, 1)	[41]	(1| 3, _, 2, 1)	[8] (1| 3, 1, 2, 1)
[9]	(2| 1, 2, 1, 2)	[28]	(2| _, 2, 1, 2)	[9] (2| 1, 2, 1, 2)
[10]	(3| 1, 2, 1, 1)	[56]	(3| 1, _, 1, 1)	[10] (3| 1, 2, 1, 1)
[11]	(2| 1, 2, 2, 2)	[29]	(2| _, 2, _, 2)	[11] (2| 1, 2, 2, 2)
[12]	(1| 1, 2, 2, 1)	[42]	(1| _, 2, 2, 1)	[12] (1| 1, 2, 2, 1)
[13]	(2| 1, 1, 1, 2)	[30]	(2| _, _, 1, 2)	[13] (2| 1, 1, 1, 2)
[14]	(3| 1, 1, 1, 1)	[48]	(3| _, 1, 1, 1)	[14] (3| 1, 1, 1, 1)
[15]	(2| 1, 1, 2, 2)	[31]	(2| _, _, _, 2)	[15] (2| 1, 1, 2, 2)
[16]	(2| 1, 1, 2, 1)	[61]	(2| 1, 1, 2, _)	[16] (2| 1, 1, 2, 1)
[17]	(2| 2, 2, 1, 2)	[28]	(2| _, 2, 1, 2)	[17] (2| 2, 2, 1, 2)
[18]	(2| 2, 2, 1, 1)	[64]	(2| 2, 2, 1, _)	[18] (2| 2, 2, 1, 1)
[19]	(2| 2, 2, 2, 2)	[29]	(2| _, 2, _, 2)	[19] (2| 2, 2, 2, 2)
[20]	(1| 2, 2, 2, 1)	[42]	(1| _, 2, 2, 1)	[20] (1| 2, 2, 2, 1)
[21]	(2| 2, 1, 1, 2)	[30]	(2| _, _, 1, 2)	[21] (2| 2, 1, 1, 2)
[22]	(3| 2, 1, 1, 1)	[48]	(3| _, 1, 1, 1)	[22] (3| 2, 1, 1, 1)
[23]	(2| 2, 1, 2, 2)	[31]	(2| _, _, _, 2)	[23] (2| 2, 1, 2, 2)
[24]	(2| 2, 1, 2, 1)	[76]	(2| 2, 1, 2, _)	[24] (2| 2, 1, 2, 1)
[25]	(2| 3, 2, _, 2)	[27]	(2| 3, _, _, 2)	[25] (2| 3, 2, _, 2)
[26]	(2| 3, _, 1, 2)	[27]	(2| 3, _, _, 2)	[26] (2| 3, _, 1, 2)
[27]	(2| 3, _, _, 2)	[31]	(2| _, _, _, 2)	[27] (2| 3, _, _, 2)
[27]	(2| 3, _, _, 2)	[36]	(2| 3, _, 2, 2)	[36] (2| 3, _, 2, 2)
[27]	(2| 3, _, _, 2)	[43]	(2| 3, 1, _, 2)	[43] (2| 3, 1, _, 2)
[28]	(2| _, 2, 1, 2)	[29]	(2| _, 2, _, 2)	[28] (2| _, 2, 1, 2)
[29]	(2| _, 2, _, 2)	[31]	(2| _, _, _, 2)	[29] (2| _, 2, _, 2)
[29]	(2| _, 2, _, 2)	[37]	(2| _, 2, 2, 2)	[37] (2| _, 2, 2, 2)
[29]	(2| _, 2, _, 2)	[52]	(2| 1, 2, _, 2)	[52] (2| 1, 2, _, 2)
[29]	(2| _, 2, _, 2)	[65]	(2| 2, 2, _, 2)	[65] (2| 2, 2, _, 2)
[30]	(2| _, _, 1, 2)	[31]	(2| _, _, _, 2)	[30] (2| _, _, 1, 2)
[30]	(2| _, _, 1, 2)	[44]	(2| _, 1, 1, 2)	[44] (2| _, 1, 1, 2)
[30]	(2| _, _, 1, 2)	[53]	(2| 1, _, 1, 2)	[53] (2| 1, _, 1, 2)
[30]	(2| _, _, 1, 2)	[66]	(2| 2, _, 1, 2)	[66] (2| 2, _, 1, 2)
[31]	(2| _, _, _, 2)	[38]	(2| _, _, 2, 2)	[38] (2| _, _, 2, 2)
[31]	(2| _, _, _, 2)	[45]	(2| _, 1, _, 2)	[45] (2| _, 1, _, 2)
[31]	(2| _, _, _, 2)	[49]	(2| _, 1, 2, 2)	[49] (2| _, 1, 2, 2)
[31]	(2| _, _, _, 2)	[54]	(2| 1, _, _, 2)	[54] (2| 1, _, _, 2)
[31]	(2| _, _, _, 2)	[57]	(2| 1, _, 2, 2)	[57] (2| 1, _, 2, 2)
[31]	(2| _, _, _, 2)	[59]	(2| 1, 1, _, 2)	[59] (2| 1, 1, _, 2)
[31]	(2| _, _, _, 2)	[67]	(2| 2, _, _, 2)	[67] (2| 2, _, _, 2)
[31]	(2| _, _, _, 2)	[72]	(2| 2, _, 2, 2)	[72] (2| 2, _, 2, 2)
[31]	(2| _, _, _, 2)	[74]	(2| 2, 1, _, 2)	[74] (2| 2, 1, _, 2)

The resulting set after this step is:

$pid(P)$	P	set of instances-creators of the pattern $\{pid(R_i)\}$
[31]	(2| _, _, _, 2)	{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23}
[33]	(3| 3, _, 1, 1)	{2, 6}
[41]	(1| 3, _, 2, 1)	{4, 8}
[42]	(1| _, 2, 2, 1)	{4, 12, 20}
[48]	(3| _, 1, 1, 1)	{6, 14, 22}
[56]	(3| 1, _, 1, 1)	{10, 14}
[61]	(2| 1, 1, 2, _)	{15, 16}
[64]	(2| 2, 2, 1, _)	{17, 18}
[76]	(2| 2, 1, 2, _)	{23, 24}

These patterns correspond to following rules:

P	rule
(2| _, _, _, 2)	Class=none,Tears=reduced
(3| 3, _, 1, 1)	Class=soft,Age=young,Astigmatic=no,Tears=normal
(1| 3, _, 2, 1)	Class=hard,Age=young,Astigmatic=yes,Tears=normal
(1| _, 2, 2, 1)	Class=hard,Prescription=myope,Astigmatic=yes,Tears=normal
(3| _, 1, 1, 1)	Class=soft,Prescription=hypermetrope,Astigmatic=no,Tears=normal
(3| 1, _, 1, 1)	Class=soft,Age=pre-presbyopic,Astigmatic=no,Tears=normal
(2| 1, 1, 2, _)	Class=none,Age=pre-presbyopic,Prescription=hypermetrope,Astigmatic=yes
(2| 2, 2, 1, _)	Class=none,Age=presbyopic,Prescription=myope,Astigmatic=no
(2| 2, 1, 2, _)	Class=none,Age=presbyopic,Prescription=hypermetrope,Astigmatic=yes

Thus, we have achieved 9 rules that are equal to the sufficient set of rules for total description of the Lenses dataset given in [Cendrowska, 1987].

7.2.2 Recognition Process in PGN

We take the instance (age=young, prescription=myope, astigmatic=no, tears=reduced) as a query. In practice this instance belongs to the class "none".

The corresponded numerical vector of the query is $Q = (?|3,2,1,2)$.

P	$IntersectionSize(P,Q)$
(2| _, _, _, 2)	1
(3| 3, _, 1, 1)	2/3
(1| 3, _, 2, 1)	1/3
(1| _, 2, 2, 1)	1/3
(3| _, 1, 1, 1)	1/3
(3| 1, _, 1, 1)	1/3
(2| 1, 1, 2, _)	0
(2| 2, 2, 1, _)	2/3
(2| 2, 1, 2, _)	0

The list with highest intersection size (1) contains only the first pattern, which belongs to class 2. Class "2: none" is given as answer.

7.3 MPGN

The focus in the construction and realization of MPGN is to show the advantages of multi-layer structures that can be easily stored in ArM archives.

Because of this, here we will point our attention in this direction using as example Lenses dataset.

7.3.1 Training Process of MPGN

The training process of MPGN consists of generalization and pruning.

The step of generalization is similar to the generalization of PGN, but uses multi-layer disposing of patterns, which decrease the number of intersections and allows to operate with patterns in more structured manner.

The main focus of MPGN is just on this step, because on the base of already created pattern set, different kinds of consequent steps can be examined.

The pruning step in MPGN differs from pruning of PGN, deleting vertexes of pyramids that contradict each other.

- ***Generalization***

After generalization of each class the multi-layer structures, containing patterns with corresponded predecessor sets and successor sets are created.

Each pattern P is named $pid(P)$ in the following manner: $[nclass/nlayer/number]$, where $nclass$ and $nlayer$ are the class and the layer, in which the pattern belongs and $number$ is unique number of the pattern within chosen class and layer.

For alleviating the writing in the predecessors' and successors' sets only unique number of the pattern is written. The class of these patterns is the same. The layer in the predecessors' set is $nlayer-1$ and correspondingly the layer in the successors' set is $nlayer+1$.

We should remember that predecessors' sets of the instances (patterns in layer 1) are empty and successors' sets of vertexes are also empty. The vertexes can belongs to different layers.

Using the predecessors' sets and successors' sets, graphical representations of created pyramids for each class are made.

✓ ***Class 1: "hard"***

The generalization of class "hard" created 3 layers, containing 4 instances in layer 1, 3 intermediate patterns in layer 2 and one vertex in the upper layer.

$pid(P)$	P	Predecessor set	Successor set
	Layer = 1		
[1/1/1]	(1| 3, 2, 2, 1)	{}	{1,2,3,1,2,3}
[1/1/2]	(1| 3, 1, 2, 1)	{}	{2,3,2,3}
[1/1/3]	(1| 1, 2, 2, 1)	{}	{1,2,1,2}
[1/1/4]	(1| 2, 2, 2, 1)	{}	{1,2,1,2}
	Layer = 2		
[1/2/1]	(1| _, 2, 2, 1)	{1,3,4}	{1}
[1/2/2]	(1| _, _, 2, 1)	{1,2,3,4}	{1}
[1/2/3]	(1| 3, _, 2, 1)	{1,2}	{1}
	Layer = 3		
[1/3/1]	(1| _, _, 2, 1)	{1,2,3}	{}

Corresponding link-spaces of class 1 are:

Layer No:	***Attribute***	***Attribute value***	***pid set***
Layer 1	1 age	1 pre-presbyopic	: {P3}
		2 presbyopic	: {P4}
		3 young	: {P1,P2}
	2 prescription	1 hypermetrope	: {P2}
		2 myope	: {P1,P3,P4}
	3 astigmatic	1 no	: {}
		2 yes	: {P1,P2,P3,P4}
	4 tears	1 normal	: {P1,P2,P3,P4}
		2 reduced	: {}
Layer 2	1 age	1 pre-presbyopic	: {}
		2 presbyopic	: {}
		3 young	: {P3}
	2 prescription	1 hypermetrope	: {}
		2 myope	: {P1}
	3 astigmatic	1 no	: {}
		2 yes	: {P1,P2,P3}
	4 tears	1 normal	: {P1,P2,P3}
		2 reduced	: {}
Layer 3	1 age	1 pre-presbyopic	: {}
		2 presbyopic	: {}
		3 young	: {}
	2 prescription	1 hypermetrope	: {}
		2 myope	: {}
	3 astigmatic	1 no	: {}
		2 yes	: {P1}
	4 tears	1 normal	: {P1}
		2 reduced	: {}

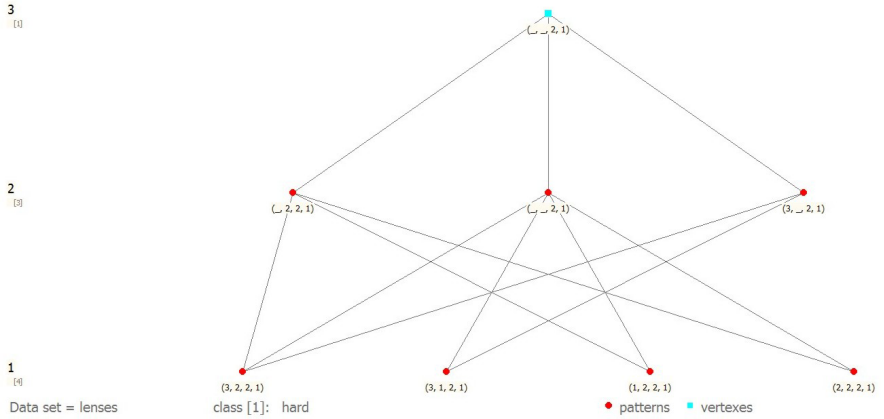


Figure 26. MPGN pyramid for class "hard" of Lenses dataset

Figure 26 shows the pyramid for class "hard".

- ✓ ***Class 2: "none"***

The class "none" starts with more instances (15) and combinations between them create more patterns. During the generalization four layers are created.

$pid(P)$	P	Predecessor set	Successor set
	Layer = 1		
[2/1/1]	(2|3,2,1,2)	{}	{1,3,4,11,13,17,18,19,38,42,1,3,4,11,13,17,18,19,38,42}
[2/1/2]	(2|3,2,2,2)	{}	{1,2,4,8,12,13,17,37,42,1,2,4,8,12,13,17,37,42}
[2/1/3]	(2|3,1,1,2)	{}	{1,3,5,7,10,13,18,35,38,1,3,5,7,10,13,18,35,38}
[2/1/4]	(2|3,1,2,2)	{}	{1,2,5,6,7,8,9,13,35,37,1,2,5,6,7,8,9,13,35,37}
[2/1/5]	(2|1,2,1,2)	{}	{1,3,4,11,16,17,18,19,32,36,43,1,3,4,11,16,17,18,19,32,36,43}
[2/1/6]	(2|1,2,2,2)	{}	{1,2,4,8,12,16,17,33,36,43,44,1,2,4,8,12,16,17,33,36,43,44}
[2/1/7]	(2|1,1,1,2)	{}	{1,3,5,7,10,16,18,32,34,39,43,1,3,5,7,10,16,18,32,34,39,43}
[2/1/8]	(2|1,1,2,2)	{}	{1,2,5,6,7,8,9,16,33,34,39,40,43,44,1,2,5,6,7,8,9,16,33,34,39,40,43,44}
[2/1/9]	(2|1,1,2,1)	{}	{6,7,8,30,39,40,41,43,44,6,7,8,30,39,40,41,43,44}
[2/1/10]	(2|2,2,1,2)	{}	{1,3,4,11,14,15,17,18,19,20,21,22,23,31,1,3,4,11,14,15,17,18,19,20,21,22,23,31}
[2/1/11]	(2|2,2,1,1)	{}	{15,17,18,19,21,23,27,31,41,15,17,18,19,21,23,27,31,41}
[2/1/12]	(2|2,2,2,2)	{}	{1,2,4,8,12,14,15,17,20,23,24,28,1,2,4,8,12,14,15,17,20,23,24,28}
[2/1/13]	(2|2,1,1,2)	{}	{1,3,5,7,10,14,15,18,21,22,25,29,1,3,5,7,10,14,15,18,21,22,25,29}
[2/1/14]	(2|2,1,2,2)	{}	{1,2,5,6,7,8,9,14,15,24,25,26,28,29,1,2,5,6,7,8,9,14,15,24,25,26,28,29

			}
[2/1/15]	(2|2,1,2,1)	{}	{6,7,8,15,24,25,26,27,30,41,6,7,8,15,24,25,26,27,30,41}
	Layer = 2		
[2/2/1]	(2|_,_,_,2)	{1,2,3,4,5,6,7,8,10,12,13,14}	{1}
[2/2/2]	(2|_,_,2,2)	{2,4,6,8,12,14}	{1,3}
[2/2/3]	(2|_,_,1,2)	{1,3,5,7,10,13}	{1,7}
[2/2/4]	(2|_,2,_,2)	{1,2,5,6,10,12}	{1,6,12}
[2/2/5]	(2|_,1,_,2)	{3,4,7,8,13,14}	{1,2}
[2/2/6]	(2|_,1,2,_)	{4,8,9,14,15}	{2,3,5}
[2/2/7]	(2|_,1,_,_)	{3,4,7,8,9,13,14,15}	{2}
[2/2/8]	(2|_,_,2,_)	{2,4,6,8,9,12,14,15}	{3}
[2/2/9]	(2|_,1,2,2)	{4,8,14}	{1,2,3,5}
[2/2/10]	(2|_,1,1,2)	{3,7,13}	{1,2,7}
[2/2/11]	(2|_,2,1,2)	{1,5,10}	{1,6,7,11,12}
[2/2/12]	(2|_,2,2,2)	{2,6,12}	{1,3,6,12}
[2/2/13]	(2|3,_,_,2)	{1,2,3,4}	{1}
[2/2/14]	(2|2,_,_,2)	{10,12,13,14}	{1,4}
[2/2/15]	(2|2,_,_,_)	{10,11,12,13,14,15}	{4}
[2/2/16]	(2|1,_,_,2)	{5,6,7,8}	{1}
[2/2/17]	(2|_,2,_,_)	{1,2,5,6,10,11,12}	{6}
[2/2/18]	(2|_,_,1,_)	{1,3,5,7,10,11,13}	{7}
[2/2/19]	(2|_,2,1,_)	{1,5,10,11}	{6,7,11}
[2/2/20]	(2|2,2,_,2)	{10,12}	{1,4,6,9,12}
[2/2/21]	(2|2,_,1,_)	{10,11,13}	{4,7,10}
[2/2/22]	(2|2,_,1,2)	{10,13}	{1,4,7,10}
[2/2/23]	(2|2,2,_,_)	{10,11,12}	{4,6,9}
[2/2/24]	(2|2,_,2,_)	{12,14,15}	{3,4}
[2/2/25]	(2|2,1,_,_)	{13,14,15}	{2,4}
[2/2/26]	(2|2,1,2,_)	{14,15}	{2,3,4,5}
[2/2/27]	(2|2,_,_,1)	{11,15}	{4,8}
[2/2/28]	(2|2,_,2,2)	{12,14}	{1,3,4}
[2/2/29]	(2|2,1,_,2)	{13,14}	{1,2,4}
[2/2/30]	(2|_,1,2,1)	{9,15}	{2,3,5,8}
[2/2/31]	(2|2,2,1,_)	{10,11}	{4,6,7,9,10,11}
[2/2/32]	(2|1,_,1,2)	{5,7}	{1,7}
[2/2/33]	(2|1,_,2,2)	{6,8}	{1,3}
[2/2/34]	(2|1,1,_,2)	{7,8}	{1,2}
[2/2/35]	(2|3,1,_,2)	{3,4}	{1,2}
[2/2/36]	(2|1,2,_,2)	{5,6}	{1,6,12}
[2/2/37]	(2|3,_,2,2)	{2,4}	{1,3}
[2/2/38]	(2|3,_,1,2)	{1,3}	{1,7}
[2/2/39]	(2|1,1,_,_)	{7,8,9}	{2}
[2/2/40]	(2|1,1,2,_)	{8,9}	{2,3,5}
[2/2/41]	(2|_,_,_,1)	{9,11,15}	{8}
[2/2/42]	(2|3,2,_,2)	{1,2}	{1,6,12}
[2/2/43]	(2|1,_,_,_)	{5,6,7,8,9}	{}
[2/2/44]	(2|1,_,2,_)	{6,8,9}	{3}
	Layer = 3		
[2/3/1]	(2|_,_,_,2)	{1,2,3,4,5,9,10,11,12,13,14,16,20,22,28,29,32,33,34,35,36,37,38,42}	{}
[2/3/2]	(2|_,1,_,_)	{5,6,7,9,10,25,26,29,30,34,35,39,40}	{3}
[2/3/3]	(2|_,_,2,_)	{2,6,8,9,12,24,26,28,30,33,37,40,44}	{4}
[2/3/4]	(2|2,_,_,_)	{14,15,20,21,22,23,24,25,26,27,28,29,31}	{2}
[2/3/5]	(2|_,1,2,_)	{6,9,26,30,40}	{3,4}
[2/3/6]	(2|_,2,_,_)	{4,11,12,17,19,20,23,31,36,42}	{1}
[2/3/7]	(2|_,_,1,_)	{3,10,11,18,19,21,22,31,32,38}	{}
[2/3/8]	(2|_,_,_,1)	{27,30,41}	{}
[2/3/9]	(2|2,2,_,_)	{20,23,31}	{1,2}
[2/3/10]	(2|2,_,1,_)	{21,22,31}	{2}
[2/3/11]	(2|_,2,1,_)	{11,19,31}	{1}
[2/3/12]	(2|_,2,_,2)	{4,11,12,20,36,42}	{1}
	Layer = 4		
[2/4/1]	(2|_,2,_,_)	{6,9,11,12}	{}
[2/4/2]	(2|2,_,_,_)	{4,9,10}	{}
[2/4/3]	(2|_,1,_,_)	{2,5}	{}
[2/4/4]	(2|_,_,2,_)	{3,5}	{}

Corresponded link-spaces of class 2 are:

Layer:	*attribute*	*Attribute value* *pid set*
Layer 1	1 age	1 pre-presbyopic: {P5,P6,P7,P8,P9} 2 presbyopic : {P10,P11,P12,P13,P14,P15} 3 young : {P1,P2,P3,P4}
	2 prescription	1 hypermetrope : {P3,P4,P7,P8,P9,P13,P14,P15} 2 myope : {P1,P2,P5,P6,P10,P11,P12}
	3 astigmatic	1 no : {P1,P3,P5,P7,P10,P11,P13} 2 yes : {P2,P4,P6,P8,P9,P12,P14,P15}
	4 tears	1 normal : {P9,P11,P15} 2 reduced : {P1,P2,P3,P4,P5,P6,P7,P8,P10,P12,P13,P14}
Layer 2	1 age	1 pre-presbyopic: {P16,P32,P33,P34,P36,P39,P40,P43,P44} 2 presbyopic : {P14,P15,P20,P21,P22,P23,P24,P25,P26,P27,P28,P29,P31} 3 young : {P13,P35,P37,P38,P42}
	2 prescription	1 hypermetrope : {P5,P6,P7,P9,P10,P25,P26,P29,P30,P34,P35,P39,P40} 2 myope : {P4,P11,P12,P17,P19,P20,P23,P31,P36,P42}
	3 astigmatic	1 no : {P3,P10,P11,P18,P19,P21,P22,P31,P32,P38} 2 yes : {P2,P6,P8,P9,P12,P24,P26,P28,P30,P33,P37,P40,P44}
	4 tears	1 normal : {P27,P30,P41} 2 reduced : {P1,P2,P3,P4,P5,P9,P10,P11,P12,P13,P14,P16, P20,P22,P28,P29,P32,P33,P34,P35,P36,P37,P38,P42}
Layer 3	1 age	1 pre-presbyopic: {} 2 presbyopic : {P4,P9,P10} 3 young : {}
	2 prescription	1 hypermetrope : {P2,P5} 2 myope : {P6,P9,P11,P12}
	3 astigmatic	1 no : {P7,P10,P11} 2 yes : {P3,P5}
	4 tears	1 normal : {P8} 2 reduced : {P1,P12}
Layer 4	1 age	1 pre-presbyopic: {} 2 presbyopic : {P2} 3 young : {}
	2 prescription	1 hypermetrope : {P3} 2 myope : {P1}
	3 astigmatic	1 no : {} 2 yes : {P4}
	4 tears	1 normal : {} 2 reduced : {}

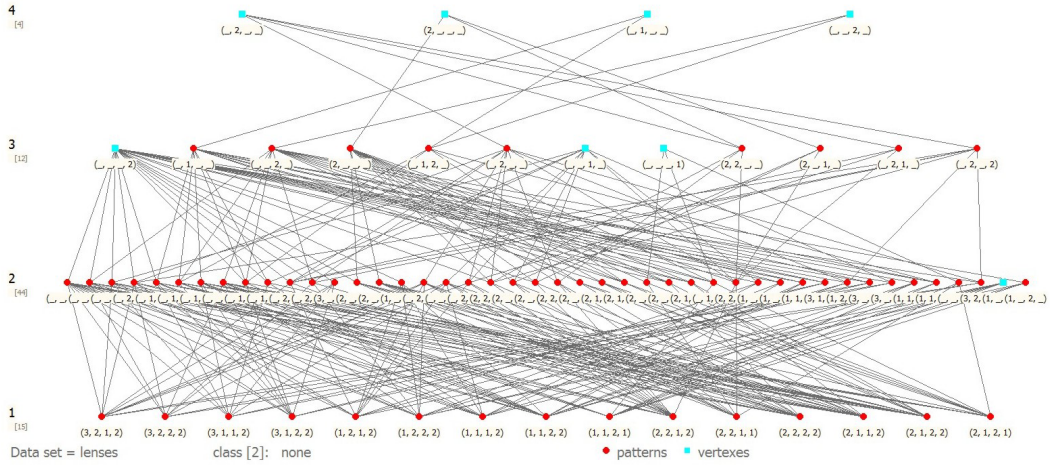


Figure 27. MPGN pyramid for class "none" of Lenses dataset

Figure 27 shows the graphical representation of the class "none". The possibility for one class to have several vertexes can be observed here. It is also notable that vertexes can belongs to different layers. Here we have one vertex in layer 2, three vertexes in layer 3 and four vertexes in the upper layer 4.

- ✓ ***Class 3: "soft"***

The generalization of class 3 "soft" creates also 3 layer pyramid with one vertex.

$pid(P)$	P	Predecessor set	Successor set
	Layer = 1		
[3/1/1]	(3| 3, 2, 1, 1)	{}	{1,4,5}
[3/1/2]	(3| 3, 1, 1, 1)	{}	{1,2,5}
[3/1/3]	(3| 1, 2, 1, 1)	{}	{1,3,4}
[3/1/4]	(3| 1, 1, 1, 1)	{}	{1,2,3}
[3/1/5]	(3| 2, 1, 1, 1)	{}	{1,2}
	Layer = 2		
[3/2/1]	(3| _, _, 1, 1)	{1,2,3,4,5}	{1}
[3/2/2]	(3| _, 1, 1, 1)	{2,4,5}	{1}
[3/2/3]	(3| 1, _, 1, 1)	{3,4}	{1}
[3/2/4]	(3| _, 2, 1, 1)	{1,3}	{1}
[3/2/5]	(3| 3, _, 1, 1)	{1,2}	{1}
	Layer = 3		
[3/3/1]	(3| _, _, 1, 1)	{1,2,3,4,5}	{}

Corresponding link-spaces of class 3 are:

Layer:	attribute	Attribute value	pid set
Layer 1	1 age	1 pre-presbyopic	: {P3,P4}
		2 presbyopic	: {P5}
		3 young	: {P1,P2}
	2 prescription	1 hypermetrope	: {P2,P4,P5}
		2 myope	: {P1,P3}
	3 astigmatic	1 no	: {P1,P2,P3,P4,P5}
		2 yes	: {}
	4 tears	1 normal	: {P1,P2,P3,P4,P5}
		2 reduced	: {}
Layer 2	1 age	1 pre-presbyopic	: {P3}
		2 presbyopic	: {}
		3 young	: {P5}
	2 prescription	1 hypermetrope	: {P2}
		2 myope	: {P4}
	3 astigmatic	1 no	: {P1,P2,P3,P4,P5}
		2 yes	: {}
	4 tears	1 normal	: {P1,P2,P3,P4,P5}
		2 reduced	: {}
Layer 3	1 age	1 pre-presbyopic	: {}
		2 presbyopic	: {}
		3 young	: {}
	2 prescription	1 hypermetrope	: {}
		2 myope	: {}
	3 astigmatic	1 no	: {P1}
		2 yes	: {}
	4 tears	1 normal	: {P1}
		2 reduced	: {}

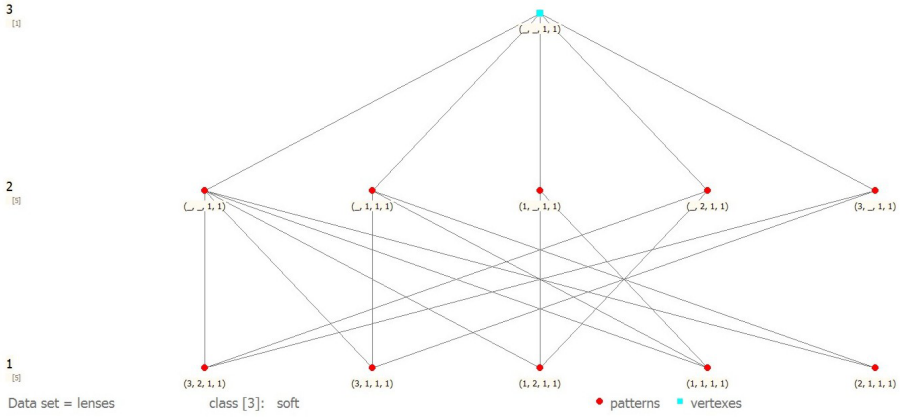


Figure 28. MPGN pyramid for class "soft" of Lenses dataset

Figure 28 shows the pyramid for the class "soft".

- ***Pruning***

The vertexes of pyramids of the three classes do not contradict. Because of this the pyramids remains unchanged.

7.3.2 Recognition Process in MPGN

We take again the same instance (age=young, prescription=myope, astigmatic=no, tears=reduced) as a query. The numerical vector of the query is $Q=(?|3,2,1,2)$.

The first phase of recognition traverses each class individually.

The vertex of the class "hard" is $P_{1/3/1}=(1|\_,\_,2,1)$. The intersection percentage is 0%. The recognition set of the first class is empty.

The vertex of the class "soft" is $P_{3/3/1}=(3|\_,\_,1,1)$. The intersection percentage is 50%. The recognition set of the first class is empty, because only patterns with 100% intersection percentage are included in the set.

For the class "none" the vertex $P_{2/4/1}=(2|\_,2,\_,\_)$ has 100% intersection percentage and initial recognition set $\{P_{2/4/1}\}$ is created.

The predecessors' set of $P_{2/4/1}$ is $\{P_{2/3/6}, P_{2/3/9}, P_{2/3/11}, P_{2/3/12}\}$.

$pid(P)$	P	$\|P\|$	$IntersectionPercentage(P,Q)$	Predecessors' set
[2/3/6]	(2| _, 2, _, _,	1	100%	{4,11,12,17,19,20,23,31,36,42}
[2/3/9]	(2| 2, 2, _, _,	2	50%	{20,23,31}
[2/3/11]	(2| _, 2, 1, _,	2	100%	{11,19,31}
[2/3/12]	(2| _, 2, _, 2)	2	100%	{4,11,12,20,36,42}

Maximal cardinality is 2 and the set of patterns that have such cardinality and 100% intersection percentage is $\{P_{2/3/11}, P_{2/3/12}\}$. This set replace pattern $P_{2/4/1}$ in the recognition set.

The analysis of the predecessors' set of $P_{2/3/11}$ and $P_{2/3/12}$ shows the following results.

For $P_{2/3/11}$:

$pid(P)$	P	$\|P\|$	$IntersectionPercentage(P,Q)$	Predecessors' set
[2/2/11]	(2| _, 2, 1, 2)	3	100%	{1,5,10}
[2/2/19]	(2| _, 2, 1, _,	2	100%	{1,5,10,11}
[2/2/31]	(2| 2, 2, 1, _,	3	66.66%	{10,11}

Maximal cardinality is 3 and $P_{2/2/11}$ replaces $P_{2/3/11}$ in the recognition set.

For $P_{2/3/12}$:

$pid(P)$	P	$\lvert P \rvert$	$IntersectionPercentage(P,Q)$	Predecessors' set
[2/2/4]	(2| _, 2, _, 2)	2	100%	{1,2,5,6,10,12}
[2/2/11]	(2| _, 2, 1, 2)	3	100%	{1,5,10}
[2/2/12]	(2| _, 2, 2, 2)	3	66.66%	{2,6,12}
[2/2/20]	(2| 2, 2, _, 2)	3	66.66%	{10,12}
[2/2/36]	(2| 1, 2, _, 2)	3	66.66%	{5,6}
[2/2/42]	(2| 3, 2, _, 2)	3	66.66%	{1,2}

Again the maximal cardinality is 3 and $P_{2/2/11}$, which fulfill both conditions is a candidate to replace $P_{2/3/12}$ in the recognition set. $P_{2/3/12}$ is removed, because the new set is not empty. The new recognition set is $\{P_{2/2/11}\}$, which had already been included in the previous step.

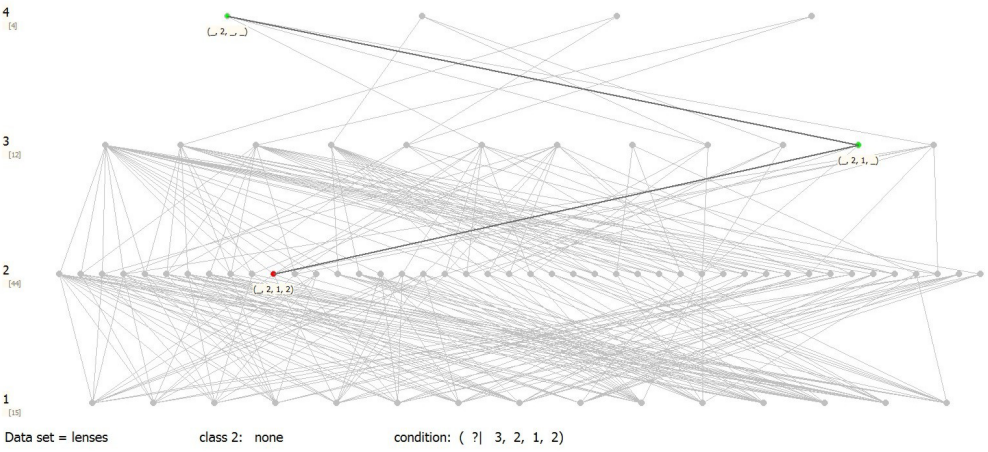


Figure 29. The process of recognition in MPGN

From the instances $P_{2/1/1} = (2 \mid 3,2,1,2)$, $P_{2/1/5} = (2 \mid 1,2,1,2)$, $P_{2/1/10} = (2 \mid 2,2,1,2)$, which are in the predecessors' set of $P_{2/2/11}$.the first fully covers the query. There is a parameter which can mark the first layer not to be given in the recognition process. Figure 29 shows the recognition process when this first layer is excluded.

In the second phase, the recognition sets from all classes are compared and maximal cardinality is estimated. Because two of the classes have empty recognition sets and only one class gives some hypothesis, the class "none" is given as an answer.

Conclusion

Here we have shown the behavior of PGN and MPGN and results from its processing on the example of Lenses dataset.

We traversed all stages of PGN-classifier and we have demonstrated that PGN produces the pattern set that is minimal and complete for covering the learning set.

In the case of MPGN we showed the process of creating the main construction elements on the example of the Lenses dataset.

Also we showed the possibilities for visualizing the processes of creating the pyramids and the recognizing the queries.

8 Sensitivity Analysis

Abstract

We made different experiments for studying the specific behavior of the proposed algorithms and for comparing our results with results from other classifiers.

Because PGN and MPGN as well as most of other classifiers deal with categorical attributes we studied different discretizators in order to choose the more convenient for our classifiers.

Further experiments follow the process of growing the learning sets and how this reflects to the classification model and the accuracy of PGN and MPGN.

One particular study addressed the analysis of exit points of MPGN in order to examine the significance of different branches of the recognition phase.

Other experiments analyzed the classifiers' behaviors when there is a noise rush in the dataset attributes.

The overall accuracy and the F-measures in particular obtained from different classifiers are compared and analyzed.

8.1 Global Frame of the Experiments

We will first discuss the main components in our experiments: chosen datasets; the processes (such as cross-validation, noising, etc.); analyzed constructs (classification models, accuracies, confusion matrices, etc.).

8.1.1 The Experimental Datasets

We have provided experiments with datasets from UCI Machine Learning Repository [Frank and Asuncion, 2010].

In these experiments the following datasets were used – Audiology, Balance scale, Blood transfusion, Breast cancer wo, Car, CMC, Credit, Ecoli, Forestfires, Glass, Haberman, Hayes-roth, Hepatitis, Iris, Lenses, Monks1, Monks2, Monks3, Post operative, Soybean, TAE, Tic tac toe, Wine, Winequality-red, and Zoo. The description of the used datasets is provided in Table 1.

Table 1. Datasets' Description

Dataset	Number of attributes	Number of classes	Number of instances	Type of attributes
audiology	69	24	200	Categorical
balance_scale	4	3	624	Categorical
blood_transfusion	3	2	748	Real
breast_cancer_wo	9	2	699	Categorical
car	6	4	1728	Categorical
cmc	9	3	1473	Categorical, Integer
credit	15	2	690	Categorical, Integer, Real
ecoli	7	8	336	Real
forestfires	12	2	517	Real
glass	9	6	214	Real
haberman	3	2	306	Integer
hayes-roth	4	3	132	Categorical
hepatitis	19	2	155	Categorical, Integer, Real
iris	4	3	150	Real
lenses	4	3	24	Categorical
monks1	6	2	432	Categorical
monks2	6	2	601	Categorical
monks3	6	2	554	Categorical
post-operative	8	3	90	Categorical, Integer
soybean	35	19	307	Categorical
tae	5	3	151	Categorical, Integer
tic_tac_toe	9	2	958	Categorical
wine	13	3	178	Integer, Real
winequality-red	11	6	1599	Real
zoo	16	7	101	Categorical, Integer

Some of the datasets contain numerical values of attributes, which cause additional questions of choosing appropriate discretization algorithm. As we have already mentioned, discretization as pre-processing step is realized in PaGaNe. We used Fayyad-Irani and Chi-square methods.

8.1.2 The Experiments

In order to receive more stable results we applied k -fold cross validation.

The process of cross validation in PaGaNe randomizes the input dataset, after that creates k folds in which sequentially puts the instances from the dataset until all instances are included in one of the folds. After that, k variants of the learning set and examining set are created, each time using succession fold as examining set and the set of other folds as learning set.

PaGaNe has a functionality to export the learning sets and examining sets as "arff"-files, which is an appropriate format for the knowledge environment of Weka. We used the exported learning set and examining set by PaGaNe as input files in Weka in order to achieve equality of the data in learning and recognition processes for all classifiers that are compared.

In the part of analysis of preprocessing discretizing step we make $k = 3$ fold cross validation. In this way the proportion between learning and examining sub-sets were respectively 2:1 (66.67%). For these experiments we use primary variants of the datasets, choosing only the datasets with real parameters (Blood transfusion, Ecoli, Forest fires, Glass, and Iris).

PGN and MPGN deal with nominal attributes. Consequently, in the experiments we first discretize the numerical attributes using Chi-merge with 95% significance level. Here, again, in order to achieve equal condition for the experiments with different classifiers, we used already discretized learning set and examining set as input files in Weka.

In the part where we study the appropriate size of the learning set we made experiments with $k = 2,3,4,5$ fold cross validation in order to receive different kinds of splitting between learning and examining set.

All other experiments are made using $k = 5$ fold cross validation (the proportion between learning and examining set – 4:1, i.e. 80%).

We made comparison with **CMAR** [Li et al, 2001] as representative of other CAR-classifiers. We used the program realization of CMAR in the LUCS-KDD Repository. CMAR is used with support threshold 1% and confidence threshold to 50%. The parameters are used as they are proposed by the experimental part of the paper that firstly present CMAR [Li et al, 2001].

Also, the following classifiers, implemented in Weka, representatives of most similar recognition models to CAR algorithms are used for comparison:

- Rules:
  - **OneR**: one-level decision tree expressed in the form of a set of rules that all test one particular attribute [Holte, 1993];
  - **JRip**: implementation a propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [Cohen, 1995];
- Trees:
  - **J48** – a Weka implementation of C4.5 [Quinlan, 1993] that produces a decision tree;
  - **REPTree** – an extension of C4.5 [Witten and Frank, 2005], which builds a decision tree using information gain reduction and prunes it using reduced-error pruning.

The ratio of this choice is that CAR-classifiers, Rules and Trees have a similar model representation language.

8.1.3 The Analyzed Constructs

The most popular metric for comparing models created as a result of the learning procedures in such types of classifiers as class association rules, decision trees and decision rules is the number of the rules.

Especially for the MPGN algorithm there are four different exit points of the recognition stage, each of them connected with a different part of the algorithm. Gathering such statistics is realized in PaGaNe in order to do sensitivity analysis and to study the behavior of the algorithm MPGN.

The confusion matrix is usually applied as a basis for analyzing the results of the classifiers. The confusion matrix is $m \times m$ matrix (Table 2), where m is the number of class labels. The rows indicate the class where the test query actually belongs to. The columns show the class label assigned to the query by the classifier. The numbers of correctly recognized instances are represented on the diagonal.

Table 2. The structure of confusion matrix

	Cl.1	Cl.2	…	Cl.m	
Cl.1	Correctly recognized queries of Cl.1	Number of queries, which are actually of Cl.1, but were predicted as Cl.2	…	Number of queries, which are actually of Cl.1, but were predicted as Cl.m	Actual number of queries of Cl.1
Cl.2	Number of queries, which are actually of Cl.2, but were predicted as Cl.1	Correctly recognized queries of Cl.2		Number of queries, which are actually of Cl.2, but were predicted as Cl.m	Actual number of queries of Cl.2
…	…		…	…	…
Cl.m	Number of queries, which are actually of Cl.m, but were predicted as Cl.1	Number of queries, which are actually of Cl.m, but were predicted as Cl.2	…	Correctly recognized queries of Cl.m	Actual number of queries of Cl.m
	Predicted number of queries of Cl.1	Predicted number of queries of Cl.2		Predicted number of queries of Cl.m	Total number of queries

Mainly classifiers usually are compared on the base of received accuracy. The accuracy is the number of correct answers over the total number of the test instances (queries).

More detailed analysis is made for each class label separately, using Recall, Precision and F-Measure.

Recall for a given class label is the number of correct answers over the actual number of the test instances (queries), or if we use the terms of confusion matrix it is the diagonal value over the sum by row.

Precision for a given class label is the number of correct answers over the predicted number of the test instances, i.e. the diagonal value over the sum by column.

F-measure is a parameter, which aims to accumulate information both for precision and recall. There are different formulas for calculating F-measure. Here F-measure is calculated as a harmonic mean of precision and recall:

$$F = \frac{2 * precision * recall}{precision + recall}.$$

For more global analysis we use average values of these measures, which characterize the dataset and classifier as a whole (not each class separately).

We use the Friedman test to detect statistically significant differences between the classifiers in terms of average accuracy [Friedman, 1940]. The Friedman test is a non-parametric test, based on the ranking of the algorithms on each dataset instead of the true accuracy estimates. We use Average Ranks ranking method, which is a simple ranking method, inspired by Friedman's statistic [Neave and Worthington, 1992]. For each dataset the algorithms are ordered according to the corresponded measure (accuracy, precision, etc.) and are assigned ranks accordingly. The best algorithm receives rank 1, the second – 2, etc. If two or more algorithms have equal value, they receive equal rank which is mean of the virtual positions that had to receive such number of algorithms if they were ordered consecutively each by other.

Let n is the number of observed datasets, k is the number of algorithms. Let r_j^i be the rank of algorithm j on dataset i. The average rank for each algorithm is calculated as $R_j = \frac{1}{n}\sum_{i=1}^{k} r_j^i$. Under the null-hypothesis, which states that all the algorithms are equivalent and so their ranks R_j should be equal, the Friedman statistic

$$\chi_F^2 = \frac{12n}{k(k+1)}\left[\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4}\right]$$

is distributed according to χ_F^2 with $k-1$ degrees of freedom.

The quantile values for $k-1$ degrees of freedom and probability α is give on Table 3 [Korn and Korn, 1961].

Table 3. The quantile values of χ^2 distribution for $k-1$ degrees of freedom and probability α

Number of classifiers k	2	3	4	5	6	7	8	9	10
$\alpha = 0.05$	3.841	5.991	7.815	9.488	11.070	12.592	14.067	15.507	16.919
$\alpha = 0.10$	2.706	4.605	6.251	7.779	9.236	10.645	12.017	13.362	14.684

When null-hypothesis is rejected, we can proceed with the Nemenyi test [Nemenyi, 1963] which is used when all classifiers are compared to each other. The performance of two classifiers is significantly different if the corresponding average ranks differ by at least the critical difference

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6n}}$$

where critical values q_α are based on the Studentized range statistic divided by $\sqrt{2}$. Some of the values of q_α is given in Table 4 [Demsar, 2006].

Table 4. Critical values for the two tailed Nemenyi test

Number of classifiers	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.960	2.343	2.569	2.728	2.850	2.949	3.031	3.102	3.164
$q_{0.10}$	1.645	2.052	2.291	2.459	2.589	2.693	2.780	2.855	2.920

The results of the Nemenyi test are shown by means of critical difference diagrams.

Our comparisons are based on the work of [Demsar, 2006]; he made a study of different kinds of used techniques for comparisons between classifiers over multiple datasets and recommended a set of simple, yet safe and robust non-parametric tests for statistical comparisons of classifiers.

8.2 Choosing an Appropriate Discretizator

In our evaluation we made experiments with the following datasets which contain real attributes – Blood transfusion, Ecoli, Forest fires, Glass, and Iris, using three fold cross-validation. In these experiments primary variants of the datasets (as they are in UCI repository) are used.

Chi-merge was examined with 90%, 95% and 99% significance level.

Fayyad-Irani is a non-parametric method.

Table 5 and Table 6 summarize the obtained overall accuracy in percentages (Table 5), and in ranking (Table 6).

Table 5. PGN accuracy (in percentage) for different discretization methods

Accuracy	Chi-merge: 90.00	Chi-merge: 95.00	Chi-merge: 99.00	Fayyad-Irani
blood_transfusion	66.44	60.55	72.86	76.21
ecoli	76.49	78.87	76.49	77.08
forestfires	56.47	57.06	53.77	54.54
glass	69.14	69.15	64.96	61.72
iris	96.00	96.00	94.67	94.67

Table 6. Ranking of PGN accuracy for different discretization methods

Accuracy	Chi-merge: 90.00	Chi-merge: 95.00	Chi-merge: 99.00	Fayyad-Irani
blood_transfusion	3	4	2	1
ecoli	3.5	1	3.5	2
forestfires	2	1	4	3
glass	2	1	3	4
iris	1.5	1.5	3.5	3.5
average	2.4	1.7	3.2	2.7

The Friedman test in this cases shows $\chi_F^2 = 3.54$, $\alpha_{0.05} = 6.251$, which means that the difference is not statistically distinctive.

Accuracy does not provide sufficient information to predict the separate class labels. Because of this we continue the analysis using average recall, which reflects more qualitative information for the received accuracy for each class label. Table 7 (in percentage) and Table 8 (ranking results) show the obtained average recalls for the examined datasets.

Table 7. PGN average recall (in percentage) for different discretization methods

aver. Recall	Chi-merge: 90	Chi-merge: 95	Chi-merge: 99	Fayyad-Irani
blood_transf.	57.233	57.167	58.933	50.000
ecoli	48.033	53.100	52.267	52.067
forestfires	56.467	57.167	53.633	54.633
glass	56.600	59.333	52.567	54.733
iris	96.500	96.567	93.967	95.267

Table 8. Ranking of PGN average recall for different discretization methods

aver. Recall	Chi-merge: 90	Chi-merge: 95	Chi-merge: 99	Fayyad-Irani
blood_transf.	2	3	1	4
ecoli	4	1	2	3
forestfires	2	1	4	3
glass	2	1	4	3
iris	2	1	4	3
average	2.4	1.4	3	3.2

On the other hand, the precision gives more concrete information as so called "measure of exactness". Table 9 (in percentage) and Table 10 (ranking results) show the obtained average precision values for the examined datasets.

Table 9. PGN average precision (in percentages) for different discretization methods

aver. Precision	Chi-merge: 90	Chi-merge: 95	Chi-merge: 99	Fayyad-Irani
blood_transf.	56.367	56.067	46.567	38.100
ecoli	44.067	51.200	50.433	52.233
forestfires	56.533	57.200	53.667	54.667
glass	68.633	63.667	55.200	62.367
iris	95.767	95.667	95.233	94.900

Table 10. Ranking of PGN average precision for different discretization methods

aver. Precision	Chi-merge:90	Chi-merge:95	Chi-merge:99	Fayyad-Irani
blood_transf.	1	2	3	4
ecoli	4	2	3	1
forestfires	2	1	4	3
glass	1	2	4	3
iris	1	2	3	4
average	1.8	1.8	3.4	3

In a classification task, a precision value of 100% for a class label C means that every item labeled as belonging to C does indeed belong to C, but says nothing about the number of items from C that were not labeled correctly. Contrary to that, a recall of 100% means that every item from class C was labeled as belonging to class C, but says nothing about how many other items were incorrectly also labeled as belonging to class C. In order to receive the complex measure that reflects both aspects, we also examine F-measure, which is harmonic mean of two measures above. The received F-measure is presented respectively in Table 11 (percentages) and Table 12 (ranking values).

Table 11. PGN average F-measure (in percentages) for different discretization methods

aver. F-measure	Chi-merge: 90	Chi-merge: 95	Chi-merge: 99	Fayyad-Irani
blood_transf.	53.833	52.733	50.233	43.200
ecoli	44.033	50.367	49.067	50.133
forestfires	56.200	56.933	53.500	54.333
glass	58.300	58.400	52.267	54.033
iris	95.900	95.900	94.000	94.633

Table 12. Ranking of PGN average F-measure for different discretization methods

aver. F-measure	Chi-merge: 90	Chi-merge: 95	Chi-merge: 99	Fayyad-Irani
blood_transf.	1	2	3	4
ecoli	4	1	3	2
forestfires	2	1	4	3
glass	2	1	4	3
iris	1.5	1.5	4	3
average	2.1	1.3	3.6	3

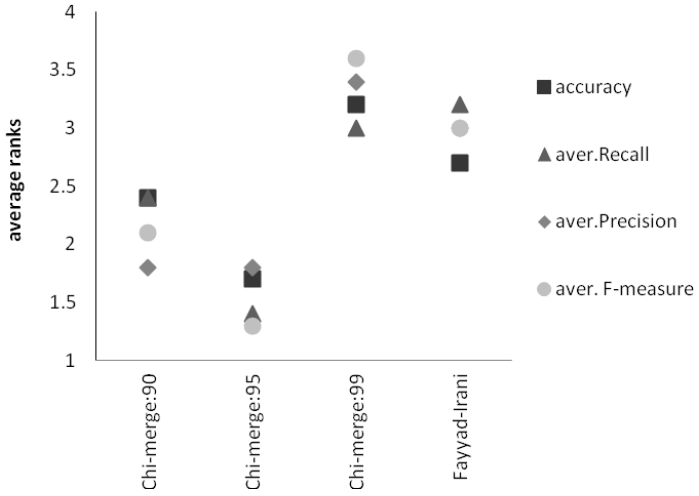


Figure 30. Comparison of different discretization methods

The analysis of the received results shows that Chi-merge discretization method with 95% significance level gives the best results for all examined measures. Close to it are the results from Chi-merge with 90% significance level. Chi-merge with 99% significance level gave worse results because of the significant fragmentation of the intervals. Fayyad-Irani method gives in some cases very good results, but fails in other databases.

The overall experiments help to make the conclusion that it is best to use Chi-merge discretization method with 95% significance level as the appropriate discretizator for the next experiments.

8.3 Studying the Size of the Learning Set

The aim of this part is to study the dependence of recognition accuracy from the size of the learning set.

For instance, one of the experiments was made over the Iris dataset (Table 13 and Figure 31). As we can see half of the instances are enough to receive stable good recognition for Iris dataset. The created model consists of about eight patterns.

Table 13. The number of patterns and accuracy from PGN-classifier for different split between learning set and examining set – Iris dataset

Split LS:ES	LS	ES	Patterns (av.number)	Accuracy (%)
1:4	30	120	6.0	90.50
1:3	37.5	112.5	6.0	92.46
1:2	50	100	6.3	93.33
1:1	**75**	**75**	**7.5**	**94.67**
2:1	100	50	8.0	94.67
3:1	112.5	37.5	8.3	94.65
4:1	120	30	8.4	94.67

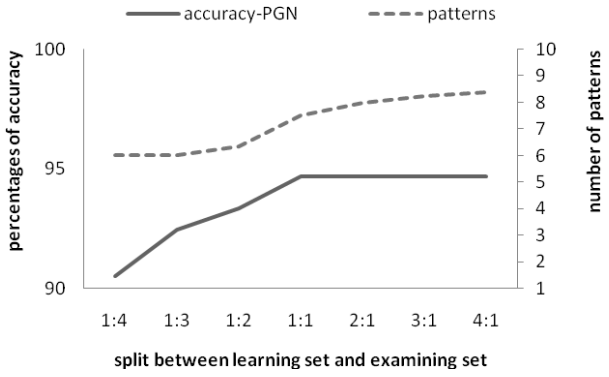


Figure 31. The number of patterns and accuracy from PGN-classifier for different split between learning set and examining set – Iris dataset

We conducted another experiment over the Glass dataset (Table 14 and Figure 32). For this dataset, good recognition with relatively small number of patterns is achieved in the case of about 140 instances. Increasing the number of learning instances did not receive better accuracy and in parallel superfluously expanded the pattern set.

The number of instances in the learning set that is enough to achieve good accuracy and tight pattern set highly would depend on the specific dataset.

Table 14. The number of patterns and accuracy from PGN-classifier for different split between learning set and examining set – Glass dataset

Split LS:ES	LS	ES	Patterns (av.number)	Accuracy (%)
1:4	42.8	171.2	29.6	67.17
1:3	53.5	160.5	35.5	63.08
1:2	71.3	142.7	44.7	73.13
1:1	107	107	68.0	74.30
2:1	**142.7**	**71.3**	**84.0**	**77.12**
3:1	160.5	53.5	88.3	76.21
4:1	171.2	42.8	95.2	77.10

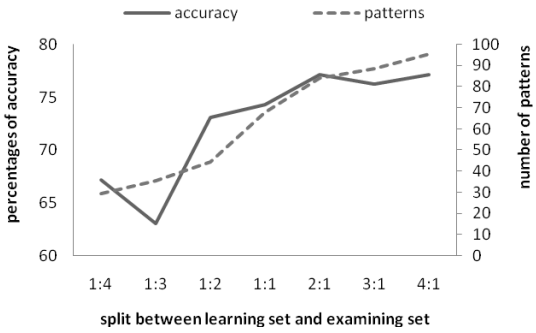


Figure 32. The number of patterns and accuracy from PGN-classifier for different split between learning set and examining set – Glass dataset

PGN is a parameter free method, but it is advisable for the user to run it with different training-learning splits in order to view the trade-off with size (simplicity) and accuracy of the model. For future research we could develop a wrapper procedure that solves this trade-off problem by using the minimum description length principle. We observed that accuracy stabilizes for some datasets but that the number of patterns increases. This is an indication that in future research we could improve the pruning part of PGN.

8.4 Examining the Exit Points of MPGN

During the design of MPGN we believe that the pyramidal network contains enough information to classify. So we expect that a majority of unseen cases can be classified by using the structure. In other words we expect that in the recognition phase only few cases will have an empty recognition set. We are also interested in how many cases have a recognition set with one class and how many cases have multiple conflicting classes. The latter cases can be classified on the basis of confidence or support. At the second step of the recognition phase MPGN can fail in different situations:

- only one class is class-candidate – we sign this case as Exit point 1;
- several classes are class-candidates. In this case two strategies are suggested in order to choose the best competitor: S1: from each class choose single rule with maximal confidence within the class and compare with others; and S2: find "confidence of recognition set", i.e. the number of instances that are covered of patterns from recognition set of this class over the number of all instances of this class and compare results. In both strategies cases are classified based on maximal confidence (Exit point 2) or maximal support (Exit point 3);
- empty recognition sets – in this case another algorithm is used – the Exit point 4.

Firstly we will analyze the three groups: one class (Exit point 1), multiple classes (Exit points 2 and 3) and no classes (Exit point 4). Secondly we will examine more into detail the two strategies for multiple classes (exit points 2 and 3).

In Table 15 and Table 16 the obtained results – number of cases and number of correct answers in cases, are presented respectively for the S1 and S2 recognition strategy.

Figure 33 and Figure 34 illustrate the percentage of different kinds of exits for S1, respectively S2 recognition strategy. The unbroken line signs percentages of different kinds of exits, the dashed line signs percentage of correct ones (the number of correct exits divided by total number of queries).

As we can see the difference between two variants are not significant because of the common constructions in the previous stages. In most cases the recognition leads to exit 1, which means that applying of the MPGN is worthwhile.

Table 15. The exit points – total and correct answers for MPGN – S1 recognition strategy

	One class	Multiple classes		No classes	One class	Multiple classes		No classes
dataset	Exit 1	Exit 2	Exit 3	Exit 4	Correct 1	Correct 2	Correct 3	Correct 4
audiology	74.50	0.00	0.00	25.50	65.00	0.00	0.00	4.00
balance_scale	48.72	23.08	28.04	0.16	48.24	15.38	17.63	0.16
blood_transfusion	5.75	0.00	0.00	94.25	3.74	0.00	0.00	71.93
breast_cancer_wo	93.99	0.72	5.29	0.00	90.84	0.57	1.43	0.00
car	73.32	6.71	19.97	0.00	70.60	3.30	8.97	0.00
cmc	49.22	23.35	27.29	0.14	25.39	8.62	11.88	0.14
credit	90.14	5.22	4.64	0.00	79.86	3.48	2.32	0.00
ecoli	76.38	11.49	12.13	0.00	66.38	4.04	6.38	0.00
forestfires	66.34	14.89	18.76	0.00	38.10	7.74	10.06	0.00
glass	85.98	8.41	5.14	0.47	72.90	3.27	3.74	0.47
haberman	38.89	0.98	1.63	58.50	29.08	0.33	1.31	42.48
hayes-roth	74.24	4.55	21.21	0.00	57.58	3.03	3.79	0.00
hepatitis	96.13	1.94	1.94	0.00	78.71	1.94	1.29	0.00
iris	85.33	4.67	9.33	0.67	82.67	3.33	8.00	0.00
lenses	87.50	4.17	8.33	0.00	79.17	4.17	0.00	0.00
monks1	63.89	20.83	15.28	0.00	62.04	20.37	0.46	0.00
monks2	76.71	20.80	2.50	0.00	70.22	8.15	2.16	0.00
monks3	77.80	5.96	16.25	0.00	75.45	0.36	14.62	0.00
post-operative	72.22	23.33	4.44	0.00	45.56	4.44	2.22	0.00
soybean	92.12	2.17	2.45	3.26	81.25	1.09	0.54	0.00
tae	72.19	8.61	17.22	1.99	42.38	5.96	4.64	0.00
tic_tac_toe	84.34	0.73	14.93	0.00	82.78	0.63	12.73	0.00
wine	92.70	3.37	3.93	0.00	91.01	0.56	0.56	0.00
winequality-red	74.11	13.70	8.19	4.00	48.78	5.07	3.81	1.69
zoo	86.14	0.00	0.00	13.86	86.14	0.00	0.00	2.97

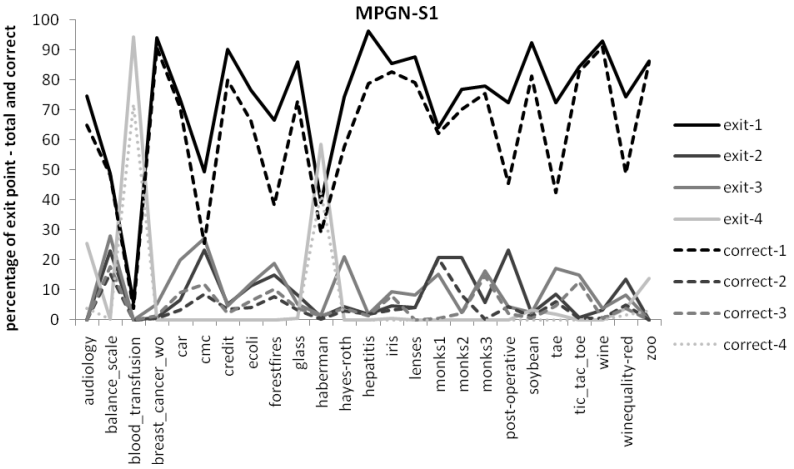


Figure 33. The exit points for MPGN – S1 recognition strategy

Table 16. The exit points – total and correct answers for MPGN – S2 recognition strategy

	One class	Multiple classes		No classes	One class	Multiple classes		No classes
dataset	Exit 1	Exit 2	Exit 3	Exit 4	Correct 1	Correct 2	Correct 3	Correct 4
audiology	74.50	0.00	0.00	25.50	65.00	0.00	0.00	4.00
balance_scale	48.72	22.44	28.69	0.16	48.24	16.19	18.91	0.16
blood_transfusion	5.75	0.00	0.00	94.25	3.74	0.00	0.00	71.93
breast_cancer_wo	93.99	2.00	4.01	0.00	90.84	1.57	1.14	0.00
car	73.32	9.38	17.30	0.00	70.60	6.02	9.09	0.00
cmc	49.22	22.40	28.24	0.14	25.39	8.55	12.56	0.14
credit	90.14	4.78	5.07	0.00	79.86	3.48	2.75	0.00
ecoli	76.38	11.70	11.91	0.00	66.38	4.26	6.38	0.00
forestfires	66.34	15.09	18.57	0.00	38.10	7.93	10.06	0.00
glass	85.98	8.41	5.14	0.47	72.90	3.27	3.74	0.47
haberman	38.89	0.98	1.63	58.50	29.08	0.33	1.31	42.48
hayes-roth	71.21	9.85	18.94	0.00	55.30	7.58	4.55	0.00
hepatitis	96.13	1.94	1.94	0.00	78.71	1.94	1.29	0.00
iris	85.33	4.67	9.33	0.67	82.67	3.33	8.00	0.00
lenses	87.50	4.17	8.33	0.00	79.17	4.17	0.00	0.00
monks1	63.89	22.92	13.19	0.00	62.04	22.92	0.93	0.00
monks2	76.71	18.64	4.66	0.00	70.22	7.32	3.49	0.00
monks3	77.80	3.61	18.59	0.00	75.45	0.72	17.33	0.00
post-operative	72.22	20.00	7.78	0.00	45.56	2.22	4.44	0.00
soybean	92.12	2.17	2.45	3.26	81.25	1.09	0.54	0.00
tae	72.19	9.27	16.56	1.99	42.38	5.96	4.64	0.00
tic_tac_toe	84.34	3.34	12.32	0.00	82.78	1.67	11.17	0.00
wine	92.70	3.93	3.37	0.00	91.01	1.69	1.12	0.00
winequality-red	74.11	13.20	8.69	4.00	48.78	4.88	4.25	1.69
zoo	86.14	0.00	0.00	13.86	86.14	0.00	0.00	2.97

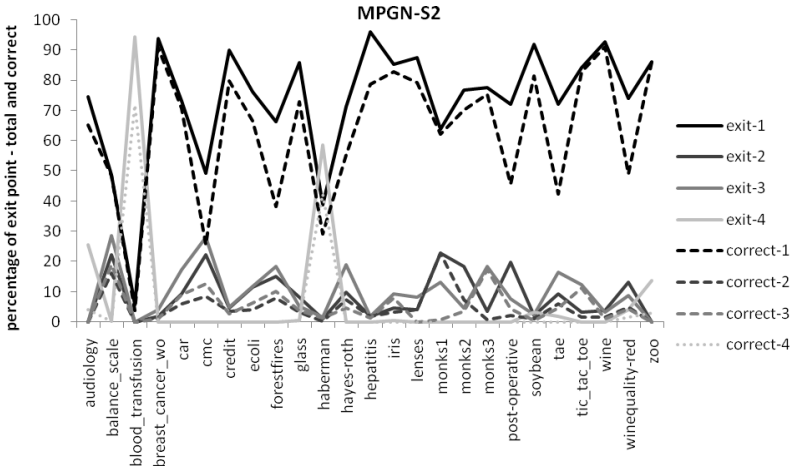


Figure 34. The exit points for MPGN – S2 recognition strategy

Table 17 and Table 18 present the results of coverage and accuracy by each case, respectively for S1 and S2 recognition strategy.

Table 17. The coverage and accuracy by exit points MPGN-S1 recognition strategy

dataset	Coverage			Accuracy		
	Exit 1	Exit 2 or 3	Exit 4	Exit 1	Exit 2 or 3	Exit 4
audiology	74.50	0.00	25.50	87.25	0.00	15.69
balance_scale	48.72	51.12	0.16	99.01	64.58	100.00
blood_transfusion	5.75	0.00	94.25	65.12	0.00	76.31
breast_cancer_wo	93.99	6.01	0.00	96.65	33.33	0.00
car	73.32	26.68	0.00	96.29	45.99	0.00
cmc	49.22	50.64	0.14	51.59	40.48	100.00
credit	90.14	9.86	0.00	88.59	58.82	0.00
ecoli	76.38	23.62	0.00	86.91	44.14	0.00
forestfires	66.34	33.66	0.00	57.43	52.87	0.00
glass	85.98	13.55	0.47	84.78	51.72	100.00
haberman	38.89	2.61	58.50	74.79	62.50	72.63
hayes-roth	74.24	25.76	0.00	77.55	26.47	0.00
hepatitis	96.13	3.87	0.00	81.88	83.33	0.00
iris	85.33	14.00	0.67	96.88	80.95	0.00
lenses	87.50	12.50	0.00	90.48	33.33	0.00
monks1	63.89	36.11	0.00	97.10	57.69	0.00
monks2	76.71	23.29	0.00	91.54	44.29	0.00
monks3	77.80	22.20	0.00	96.98	67.48	0.00
post-operative	72.22	27.78	0.00	63.08	24.00	0.00
soybean	92.12	4.62	3.26	88.20	35.29	0.00
tae	72.19	25.83	1.99	58.72	41.03	0.00
tic_tac_toe	84.34	15.66	0.00	98.14	85.33	0.00
wine	92.70	7.30	0.00	98.18	15.38	0.00
winequality-red	74.11	21.89	4.00	65.82	40.57	42.19
zoo	86.14	0.00	13.86	100.00	0.00	21.43

Table 18. The coverage and accuracy by exit points MPGN-S2 recognition strategy

dataset	Coverage			Accuracy		
	Exit 1	Exit 2 or 3	Exit 4	Exit 1	Exit 2 or 3	Exit 4
audiology	74.50	0.00	25.50	87.25	0.00	15.69
balance_scale	48.72	51.12	0.16	99.01	68.65	100.00
blood_transfusion	5.75	0.00	94.25	65.12	0.00	76.31
breast_cancer_wo	93.99	6.01	0.00	96.65	45.24	0.00
car	73.32	26.68	0.00	96.29	56.62	0.00
cmc	49.22	50.64	0.14	51.59	41.69	100.00
credit	90.14	9.86	0.00	88.59	63.24	0.00
ecoli	76.38	23.62	0.00	86.91	45.05	0.00
forestfires	66.34	33.66	0.00	57.43	53.45	0.00
glass	85.98	13.55	0.47	84.78	51.72	100.00
haberman	38.89	2.61	58.50	74.79	62.50	72.63
hayes-roth	71.21	28.79	0.00	77.66	42.11	0.00
hepatitis	96.13	3.87	0.00	81.88	83.33	0.00
iris	85.33	14.00	0.67	96.88	80.95	0.00

lenses	87.50	12.50	0.00	90.48	33.33	0.00
monks1	63.89	36.11	0.00	97.10	66.03	0.00
monks2	76.71	23.29	0.00	91.54	46.43	0.00
monks3	77.80	22.20	0.00	96.98	81.30	0.00
post-operative	72.22	27.78	0.00	63.08	24.00	0.00
soybean	92.12	4.62	3.26	88.20	35.29	0.00
tae	72.19	25.83	1.99	58.72	41.03	0.00
tic_tac_toe	84.34	15.66	0.00	98.14	82.00	0.00
wine	92.70	7.30	0.00	98.18	38.46	0.00
winequality-red	74.11	21.89	4.00	65.82	41.71	42.19
zoo	86.14	0.00	13.86	100.00	0.00	21.43

From the coverage percentages we can seen that in the majority of cases the recognition set contains one or multiple classes. Figure 35 gives the scatter plot of the coverage for one class (X axis) and the coverage of multiple classes (Y axis).

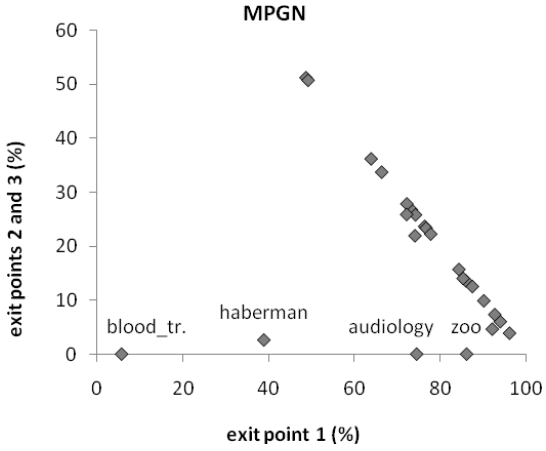


Figure 35. The scatterplot of the coverages for one class and multiple classes

There are four outliers: the datasets Audiology, Blood transfusion, Haberman and Zoo. The analysis of the Blood transfusion dataset shows that from one side there are contradictions between classes (6.4%) and from other side the attribute values are very sparse and during the pruning phase almost all patterns are pruned. Because of this the algorithm fall into Exit point 4. Similar situation is for part of Haberman dataset. The distribution of the coverages of datasets Audiology and Zoo have a different pattern. There are no cases with multiple classes in the recognition sets. The cases with an empty recognition set are representative and the accuracy of Exit point 4 is low.

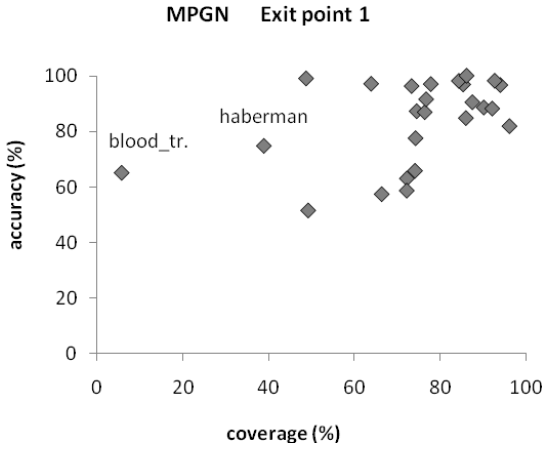


Figure 36. Scatter plot of Coverages and Accuracies for Exit point 1

The initial analysis is to check how good in terms of accuracy the different exits are performing. Figure 36 gives the scatter plot with coverage on the X axis and accuracy on the Y axis for Exit point 1. A general observation which can be made is that the accuracies are high.

Figure 37 presents the scatter plot of coverages and accuracies for the cases with multiple classes in the recognition set (Exit points 2 and 3). The accuracies are as expected lower than for the cases with one class in the recognition set. However in the corner "low accuracy, high coverage" there are no points (datasets).

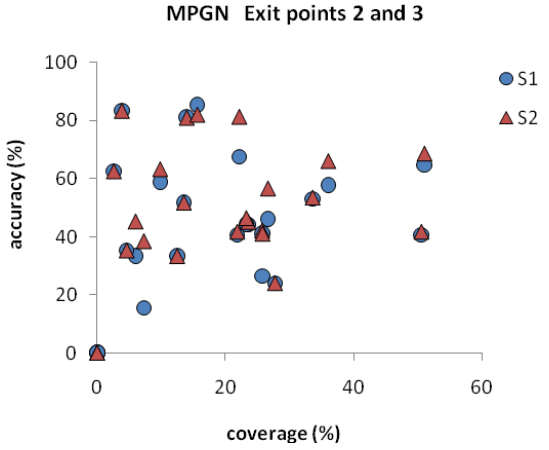


Figure 37. Scatter plot of Coverages and Accuracies for Exit points 2 and 3

In this part of the recognition phase we are using support and confidence. In future development we can extend our algorithm to become cost sensitive by manipulating confidence or support.

In section 8.6 we will compare accuracies with other classifiers; here we will consider the performance of the different exit points or recognition parts and especially the difference between one class and multiple classes. Figure 38 illustrates the relative performance: the accuracy of the recognition part divided by the mean accuracy of all classifiers.

Except for the post-operative dataset, the recognition based on one class recognition set does a high quality job. The relative performance is higher than one and as noted before the coverage is high or most datasets. Two lessons can be learned from this. First, it is worthwhile to examine whether MPGN can be used for ranking problems and campaign applications. Here we could only use cases classified by the one class recognition set. Second, to improve accuracy we should focus the recognition part with multiple classes. This influenced our decision to try two different strategies (see section 5.3.2).

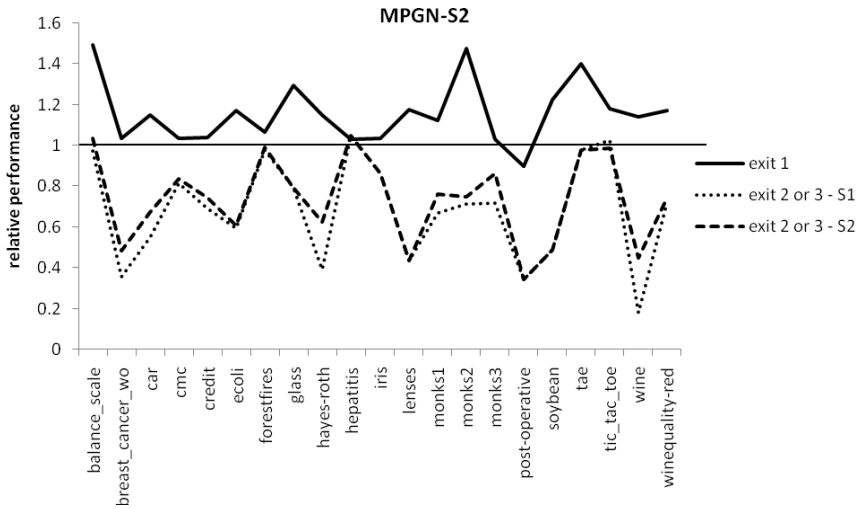


Figure 38. Relative performance of the recognition parts over the mean accuracy of all classifiers

Figure 39 presents the scatter plot of the obtained accuracies for both strategies.

We can conclude that Strategy 2 has a mean accuracy of 49% and Strategy 1 has a mean accuracy of 46%. In further research we could examine whether other methods/classifiers outperform these two strategies. If so we could adapt our recognition method here.

The Friedman test shows $\chi_F^2 = 2.56$, $\alpha_{0.05} = 1.960$, which means that the MPGN-S2 statistically outperforms MPGN-S1.

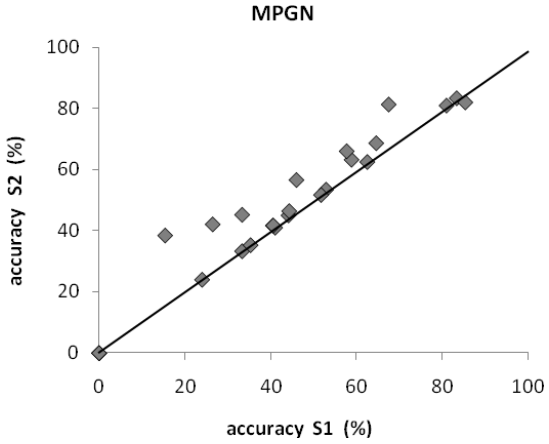


Figure 39. Scatter plot of the obtained accuracies for MPGN-S1 and MPGN-S2

In conclusion, the analysis of different recognition parts learned us that the initial idea of the PaGaNe algorithms works well and that for many cases the recognition set contains only one and the correct class. Some datasets have a specific distribution.

8.5 Noise in the Datasets

In the pruning phase we deleted patterns when there are contradictory cases without looking at noise, outliers or confidence. This can be too rudimentary when there is noise in the dataset. Therefore it is interesting to analyze the performance of algorithms against noise. The received accuracy is determined by two important factors:

- the inductive bias of the learning algorithm;
- the quality of the training data.

Given a learning algorithm, it is obvious that its classification accuracy depends on the quality of the training data. Generally, there are two types of noise sources [Wu, 1995]:

- attribute noise (the errors that are introduced in the attribute values of the instances);

- class noise (contradictory examples, i.e., the same examples with different class labels; or misclassifications, i.e. instances labeled with wrong classes).

Here we make experiments with artificial noising of datasets in order to study the robustness of PGN and MPGN classifier.

The noising of the datasets has been introduced by choosing random instance and attribute and replacing the value with arbitrary chosen possible for this attribute value. The system keeps the information for the instances and position when such changes are already made and does not make repetitive changing of the same positions. Such replacing are made until a desired percentage of noising is achieved.

We selected Monks1 dataset, which is a clear dataset with uniform class distribution and made 5, 10, 15 and 20 % noising of the attributes.

Noising within attributes reflects to noising of class labels because of the appearance of contradictory instances. Table 19 shows the resulting noise in class labels (appearing contradictory instances).

Table 19. Resulting noise in class labels after noising the attributes in Monks1 dataset

Percentage of noising in attributes	Resulting noise between class labels
0%	0.00 %
5%	6.00 %
10%	12.50 %
15%	17.25 %
20%	22.45 %

➢ ***PGN Behavior***

We processed the clear dataset and noisy datasets over the 5-fold cross-validation of PGN and checked the amount of the pattern sets and the obtained accuracy, see Table 20.

Table 20. The number of patterns and accuracy from PGN-classifier for noising datasets based on Monks1 dataset

Percentage of noising in attributes	Patterns (av.number)	Accuracy (%)
0%	59.4	100.00
5%	108.8	92.36
10%	145.2	79.62
15%	153.8	71.77
20%	145.0	66.67

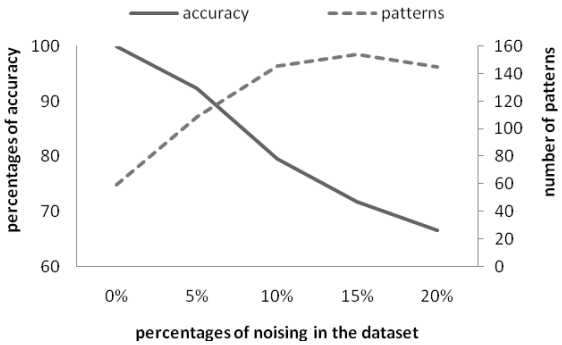


Figure 40. The number of patterns and accuracy from PGN-classifier for noising datasets based on Monks1 dataset

Table 20 and Figure 40 show the behavior of the amount of classification model (number of patterns in the pattern set) and the accuracy of the PGN classifier as a result of noising of Monsk1 dataset. The graphic affirms the expectation that when the noise in the dataset has been increased the number of rules would grow while the accuracy would decrease. The architecture of the PGN classifier makes it sensitive of the available noise – the patterns become more detailed and their number increases. The decrease in the number of patterns in the case of 20% noising can be explained with the fact that the dataset is quite different from the original and class labels have to conform to quite different rules.

- ***MPGN Behavior***

We made similar experiments with MPGN structure.

Table 21 and Figure 41 show the results for the MPGN classifier. The expectation were that noising the datasets will cause deeper distortion of the pyramids as a result of appearing more often of contradictory vertexes between class labels. The graphic shows that for 5% and 10% the number of pruned vertexes increase. The decrease after that maybe is by the same reason as in PGN – the class labels change their profiles as a result of changing the dataset.

Table 21. The number of pruned vertexes and accuracy from MPGN-classifier for noising datasets based on Monks1 dataset

Percentage of noising in attributes	Vertexes (av.number)	Accuracy MPGN-S1 (%)	Accuracy MPGN-S2 (%)
0%	782	82.900	85.916
5%	859	78.948	81.736
10%	1020	72.934	74.078
15%	937	64.374	66.690
20%	879	65.288	65.286

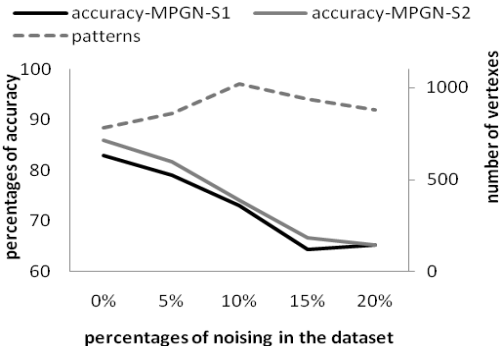


Figure 41. The number of pruned vertexes and accuracy from MPGN-classifiers for noising datasets based on Monks1 dataset

- ***Accuracy of Different Tested Classifiers for Noisy Datasets***

The same dataset has been tested with other classifiers in order to study their accuracy.

Table 22. The accuracy from different classifiers for noising datasets based on Monks1 dataset

	PGN	MPGN-S1	MPGN-S2	CMAR	OneR	JRip	J48	REPTree
0%	100.00	82.90	85.92	100.00	74.98	87.53	94.68	88.91
5%	92.36	78.95	81.74	95.14	72.67	83.37	86.79	81.96
10%	79.62	72.93	74.08	87.27	68.73	75.73	81.92	77.06
15%	71.77	64.37	66.69	80.79	68.05	71.80	77.33	74.06
20%	66.67	65.29	65.29	74.31	65.96	66.02	71.30	65.72

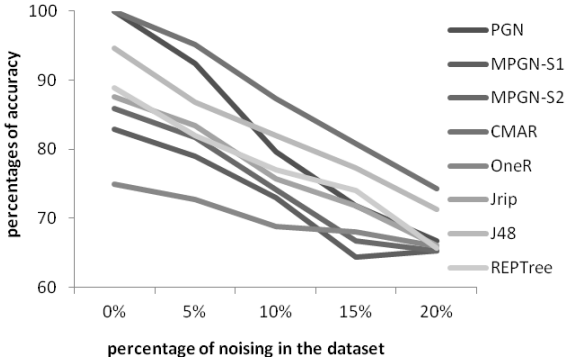


Figure 42. The accuracy for different classifiers for noising datasets based on Monks1

Table 22 and Figure 42 show that all classifiers have relatively similar decreasing of accuracy when noise in the datasets arises. The best performing method in this experiment is CMAR. Also very stable is J48. The PGN and MPGN are most sensitive to noise, which confirms our hypothesis that confidence-prioritising approach has its disadvantages in noising datasets.

8.6 Comparison with Other Classifiers

We compared the proposed classifiers with some of those implemented in Weka. We conducted the experiments with PGN, MPGN (with two recognition strategies S1 and S2), CMAR as representative of CAR-classifiers, OneR and JRip as representatives of decision rules classifiers, and J48 and REPTree as representatives of decision trees.

The comparisons are two-fold measuring:

- overall accuracy;
- F-measure results.

8.6.1 Comparison with Overall Accuracy

In Table 23 the obtained results for the overall accuracy are shown.

Table 23. Percentage of overall accuracy of examined datasets for PGN, MPGN-S1, MPGN-S2, CMAR, OneR, JRip, J48, and REPTree

Datasets	PGN	MPGN-S1	MPGN-S2	CMAR	OneR	JRip	J48	REPTree
audiology	75.50	69.00	69.00	59.18	47.00	69.50	72.00	62.50
balance_scale	77.89	81.41	83.49	86.70	60.10	71.95	66.18	67.15
breast_cancer_wo	96.43	92.85	93.56	93.85	91.85	93.28	94.28	93.99
car	92.59	82.87	85.71	81.77	70.03	86.75	90.80	88.20
cmc	49.90	46.03	46.64	53.16	47.25	50.38	51.60	50.17
credit	87.54	85.65	86.09	87.10	85.51	85.07	85.36	85.07
haberman	55.27	73.21	73.21	71.90	72.88	73.21	73.21	74.20
hayes-roth	81.94	65.22	67.49	83.42	50.77	78.12	68.23	73.53
hepatitis	80.65	81.94	81.94	84.52	81.94	77.42	79.36	79.36
lenses	74.00	83.00	83.00	88.00	62.00	83.00	83.00	80.00
monks1	100.00	82.9	85.92	100.00	74.98	87.53	94.68	88.91
monks2	73.06	80.52	81.02	59.74	65.73	58.73	59.90	63.90
monks3	98.56	90.43	93.50	98.92	79.97	98.92	98.92	98.92
post-operative	66.67	52.22	52.22	51.11	68.89	70.00	71.11	71.11
soybean	93.15	84.00	84.00	78.48	37.44	85.35	87.64	78.18
tae	52.94	52.88	52.88	35.74	45.76	34.43	46.97	40.43
tic_tac_toe	88.93	96.13	95.62	98.75	69.93	98.02	84.23	80.37
wine	96.09	92.19	93.87	91.70	78.63	90.45	87.03	88.16
winequality-red	64.98	59.35	59.60	56.29	55.54	53.72	58.22	57.03
zoo	98.10	89.24	89.24	94.19	73.29	88.19	94.14	82.19

In order to apply Friedman's test to measure statistical dissimilarity of different classifiers we ranked the results (Table 24).

Table 24. Ranking by accuracy of PGN, MPGN-S1, MPGN-S2, CMAR, OneR, JRip, J48, and REPTree

Datasets	PGN	MPGN-S1	MPGN-S2	CMAR	OneR	JRip	J48	REPTree
audiology	1	4.5	4.5	7	8	3	2	6
balance_scale	4	3	2	1	8	5	7	6
breast_cancer_wo	1	7	5	4	8	6	2	3
car	1	6	5	7	8	4	2	3
cmc	5	8	7	1	6	3	2	4
credit	1	4	3	2	5	7.5	6	7.5
haberman	8	3.5	3.5	7	6	3.5	3.5	1
hayes-roth	2	7	6	1	8	3	5	4
hepatitis	5	3	3	1	3	8	6.5	6.5
lenses	7	3.5	3.5	1	8	3.5	3.5	6
monks1	1.5	7	6	1.5	8	5	3	4
monks2	3	2	1	7	4	8	6	5
monks3	5	7	6	2.5	8	2.5	2.5	2.5
post-operative	5	6.5	6.5	8	4	3	1.5	1.5
soybean	1	4.5	4.5	6	8	3	2	7
tae	1	2.5	2.5	7	5	8	4	6
tic_tac_toe	5	3	4	1	8	2	6	7
wine	1	3	2	4	8	5	7	6
winequality-red	1	3	2	6	7	8	4	5
zoo	1	4.5	4.5	2	8	6	3	7
average	**2.975**	**4.625**	**4.075**	**3.85**	**6.8**	**4.85**	**3.925**	**4.9**

The Friedman test shows as follows:

- the number of the datasets are $n = 20$;
- the number of classifiers are $k = 8$;
- the degree of freedom is $k - 1 = 7$;
- for this degree of freedom the null hypothesis critical values are respectively – $\alpha_{0.05} = 14.067$ $\alpha_{0.10} = 12.017$;
- in our case $\chi_F^2 = 29.492$ which means that the null-hypothesis is rejected, i.e. the classifiers are statistically different.

This indicates that there are statistically significant differences in accuracy among these classifiers. The rejecting of null-hypothesis of Friedman test gives the assurance to make post-hoc Nemenyi test. In our case $CD = q_\alpha * 0.611$, $q_{0.10} = 2.780$, $q_{0.05} = 3.031$, i.e. $CD_{0.10} = 2.153$, $CD_{0.05} = 2.348$.

In Table 25 the average ranks of the classifiers are shown. The classifiers are ordered by average ranks. It should be noticed that PGN has best performance from examined classifiers.

Table 25. Average ranks of the classifiers and distance to the average rank of the first one

classifier	Average rank	Distance between average rank of the classifier and average rank of the first one
PGN	2.975	0
CMAR	3.850	0.875
J48	3.925	0.950
MPGN-S2	4.075	1.100
MPGN-S1	4.625	1.650
Jrip	4.850	1.875
REPTree	4.900	1.925
OneR	6.800	3.825

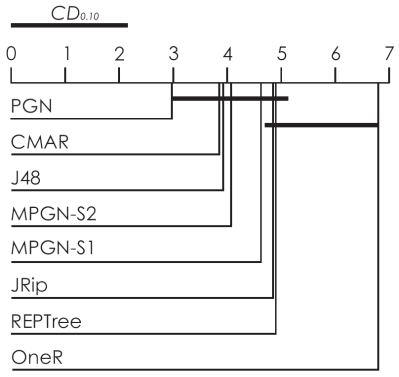


Figure 43. Visualisation of Nemenyi test results – 20 datasets

Figure 43 visualizes the results of the Nemenyi test ($CD_{0.10} = 2.153$). All groups of classifiers that are not significantly different are connected. From these results we see that PGN has best overall performance between examined classifiers and MPGN is competitive with J48, JRip and REPTree. The first four classifiers (PGN, CMAR, J48, and MPGN-S2) significantly outperform OneR.

8.6.2 Analyzing F-measures on Some Multi-class Datasets

The overall accuracy of a particular accuracy may be good in some cases, and yet if might not be able to recognize some of the class labels for different reasons – small percentage of presence of given class label, mixing with other one, etc.

We present below the obtained results for F-measure in order to see more detailed performance of different classifiers.

We make the experiments over datasets with two particular characteristics – too many class labels, and unbalanced support of different class labels.

We make the analysis over Glass, Winequality-red, and Soybean datasets from UCI-repository.

We choose F-measure as harmonic mean of recall (measure of completeness) and precision (measure of exactness).

- ***Detailed Performance for Glass Dataset***

Glass dataset has 6 class labels with very uneven distribution between them (Table 26).

Table 26. Percentage of instances belonging to corresponded class labels in Glass dataset

Class label in Glass dataset	Percentage of presence
2#	35.51
1#	32.71
7#	13.55
3#	7.94
5#	6.07
6#	4.21

Table 27 and Figure 44 present the F-measure for each class label from different classifiers.

Table 27. Percentage of F-measure from tested classifiers for Glass dataset

Class labels	PGN	MPGN-S1	MPGN-S2	OneR	CMAR	Jrip	J48	REPTree
2# (35.5%)	80.3	81.6	82.2	57.5	79.7	67.9	78.1	75.2
1# (32.7%)	80.3	81.1	81.1	64.4	80.5	69.0	76.5	70.1
7# (13.6%)	89.7	89.7	89.7	53.8	90.3	85.7	84.2	67.8
3# (7.9%)	51.9	59.5	57.9	0.0	38.5	16.0	29.6	34.5
5# (6.1%)	58.1	69.2	69.2	0.0	64.0	53.8	64.5	21.1
6# (4.2%)	88.9	94.1	94.1	0.0	77.8	60.0	55.6	80.0

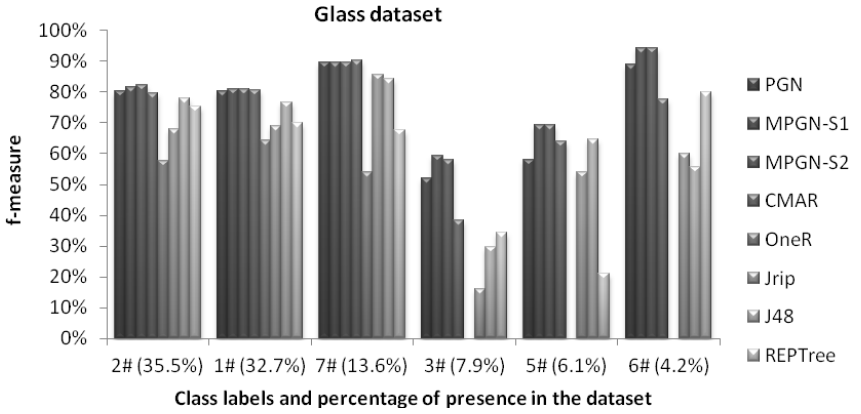


Figure 44. F-measure for examined classifiers for class-labels of Glass dataset

As we can see PGN and MPGN have good performance for each of the class labels. For instance low-presented class label "3#" is not good performed by CMAR, OneR, JRip, J48 and REPTree; "5#" – from OneR and REPTree; "6#" – from OneR (F-measures are less than 50%).

- ***Detailed Performance for Winequality-red Dataset***

Winequality-red dataset has 6 class labels with a variety of distribution (Table 28).

Table 28. Percentage of instances belonging to corresponded class labels in Winequality-red dataset

Class label in Winequality-red dataset	Percentage of presence
5#	42.59
6#	39.90
7#	12.45
4#	3.32
8#	1.13
3#	0.63

Table 29 and Figure 45 show F-measure for each class label from different classifiers.

Table 29. Percentage of F-measure from tested classifiers for Winequality-red dataset

Class labels	PGN	MPGN-S1	MPGN-S2	CMAR	OneR	Jrip	J48	REPTree
5#	74.7	26.7	66.7	70.2	65.7	65.7	68.9	65.8
6#	61.9	24.7	51.4	44.0	56.5	47.4	56.7	57.3
7#	48.6	46.3	44.9	40.6	0.0	28.9	36.1	35.9
4#	0.0	3.2	0.0	0.0	0.0	0.0	0.0	2.9
8#	20.0	12.5	14.3	0.0	0.0	0.0	0.0	0.0
3#	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

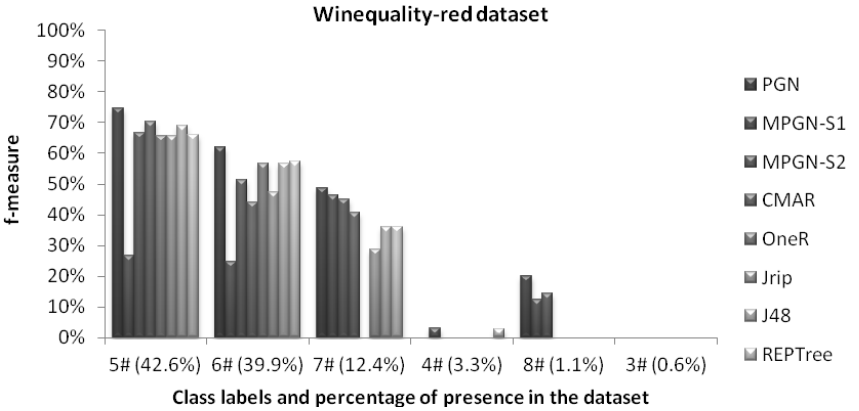


Figure 45. F-measure for examined classifiers for Winequality-red dataset

Practically, the class labels with few training instances are recognized only by PGN and MPGN; class "4#" is recognized by REPTree also. The least presented class cannot be detected by any algorithm.

- ***Detailed Performance for Soybean Dataset***

Soybean dataset has 19 class labels with different groups of distribution (Table 30).

Table 30. Percentage of instances belonging to corresponded class labels in Soybean dataset

Class label in Soybean dataset	Percentage of presence
alternarialeaf-spot	13.029
brown-spot	13.029
frog-eye-leaf-spot	13.029
phytophthora-rot	13.029
anthracnose	6.515

brown-stem-rot	6.515
bacterial-blight	3.257
bacterial-pustule	3.257
charcoal-rot	3.257
diaporthe-stem-canker	3.257
downy-mildew	3.257
phyllosticta-leaf-spot	3.257
powdery-mildew	3.257
purple-seed-stain	3.257
rhizoctonia-root-rot	3.257
cyst-nematode	1.954
diaporthe-pod-&-stem-blight	1.954
herbicide-injury	1.303
2-4-d-injury	0.326

Table 31 and Figure 46 show F-measure for each class label from different classifiers.

Here the class with lowest support "2-4-d-injury" is not recognized by any classifier because of the very low presence (1 instance) – it falls or in the learning set either in the examining set.

As we can see PGN recognizes successfully all other class labels instead of differences of their support. J48 also recognizes well but fails in low presented classes. MPGN has relatively well behavior.

Table 31. Percentage of F-measure from tested classifiers for Soybean dataset

Class labels	PGN	MPGN-S1	MPGN-S2	CMAR	OneR	Jrip	J48	REPTree
alternarialeaf-spot	86.7	77.4	77.4	93.0	35.8	62.3	89.2	50.7
brown-spot	91.8	74.7	74.7	73.4	38.2	71.9	87.5	58.2
frog-eye-leaf-spot	83.8	75.7	75.7	51.6	69.6	61.7	87.8	52.3
phytophthora-rot	100.0	100.0	100.0	91.9	71.2	76.8	95.2	73.3
anthracnose	100.0	93.0	93.0	94.7	76.9	81.1	89.5	75.0
brown-stem-rot	100.0	97.4	97.4	70.3	0.0	100.0	88.4	54.9
bacterial-blight	90.0	75.0	75.0	73.3	0.0	0.0	90.0	30.8
bacterial-pustule	90.0	85.7	85.7	80.0	0.0	75.0	85.7	42.9
charcoal-rot	100.0	100.0	100.0	100.0	28.6	94.7	100.0	57.1
diaporthe-stem-canker	100.0	88.9	88.9	100.0	41.7	84.2	88.9	42.9
downy-mildew	100.0	100.0	100.0	95.2	0.0	82.4	100.0	0.0
phyllosticta-leaf-spot	70.6	42.9	42.9	100.0	0.0	33.3	82.4	0.0
powdery-mildew	100.0	100.0	100.0	16.7	0.0	75.0	90.0	0.0
purple-seed-stain	95.2	100.0	100.0	100.0	0.0	88.9	82.4	46.2
rhizoctonia-root-rot	100.0	75.0	75.0	100.0	0.0	75.0	100.0	46.2
cyst-nematode	100.0	100.0	100.0	100.0	0.0	50.0	92.3	0.0
diaporthe-pod-&-stem-blight	100.0	90.9	90.9	57.1	50.0	66.7	66.7	54.5
herbicide-injury	88.9	40.0	40.0	82.4	0.0	57.1	75.0	40.0
2-4-d-injury	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

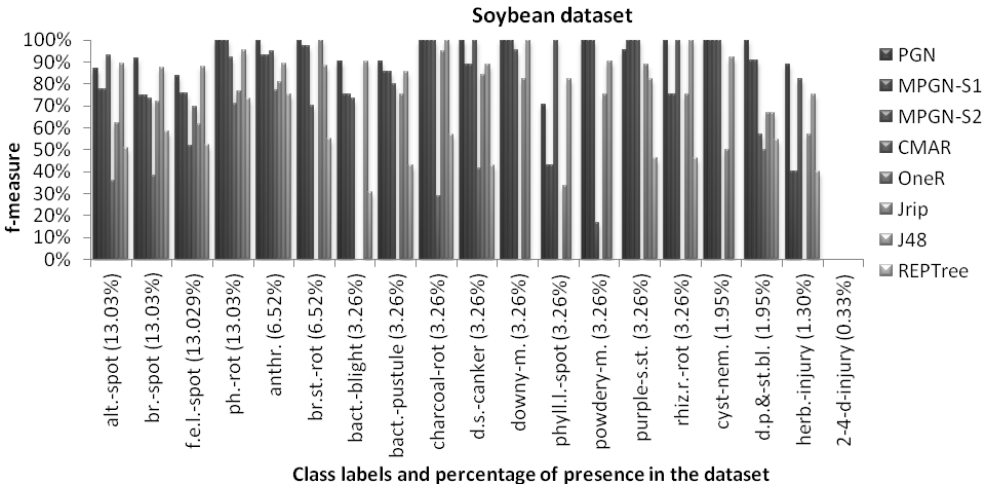


Figure 46. F-measure for examined classifiers for Soybean dataset

Looking at the results of the overall accuracy as well as refined analysis of the behavior of classifiers over multi-classes non-uniform distributed datasets we can conclude that our expectations that PGN-strategy for focusing more to the confidence than to support has a potential to create vivid classification algorithms.

Conclusion

We made experiments with 25 datasets from UCI machine learning repository. The experiments were performed using data mining environment PaGaNe and the knowledge analysis system Weka and LUCS-KDD Repository.

One series of experiments focused on the obtained accuracy when preprocessing real data with different discretizators, realized in PaGaNe. We made experiments with Blood transfusion, Ecoli, Forest fires, Glass, and Iris datasets. We found that in general PGN-classifier trained on data preprocessed by Chi-merge achieves lower classification.

Other experiments studied the process of growing the learning sets and how this reflects to the classification model and the accuracy of PGN and MPGN. In the case of Iris and Glass dataset we studied the critical point of the amount of the learning set, in which classification model is relatively compact and the received accuracy stands relatively equal with the accuracy, received from bigger learning set. Of course this critical point highly depends on the dataset and is different for different ones.

The analysis of exit points of MPGN showed that in most cases the build constructs lead to excluding only one class as best competitor.

Another experiment was aimed to analyze the depending of classifiers' behaviors when the noise rush in the dataset attributes on the case of Monks1 dataset. We showed that noising in the dataset significantly worsens the accuracy of PGN, which by its construction keeps well in clear datasets. However, experiments with other classifiers show that they did not resist noising attacks also.

We made the comparison of overall accuracy between PGN, MPGN (two recognition strategies – S1 and S2), CMAR, OneR, JRip, J48 and REPTree. The Friedman test showed statistical difference between tested classifiers. The post-hoc Nemenyi test showed that our PGN has best overall performance amongst examined classifiers and MPGN is competitive with J48, JRip and REPTree. The first four classifiers (PGN, CMAR, J48, and MPGN-S2) significantly outperform OneR.

The analysis of F-measures for different datasets with multiple classes and non-uniform distribution show that PGN and MPGN have not only good recognition accuracy for the chosen dataset, but also they recognize small classes controversy to the other classifiers (for instance OneR), which cannot construct recognition model for small class labels.

9 Conclusions and Future Work

9.1 Conclusions

The goals of this thesis were two-fold:

- to introduce a parameter-free class association rule algorithm, which focuses primarily on the confidence of the association rules and only in a later stage on the support of the rules. The main purpose is to provide a proof of concept for this new approach and collect evidence about its potential;
- to show the advantages of using multidimensional numbered information spaces for memory structuring in data mining processes on the example of realization of proposed class association rule algorithms.

In order to achieve these goals, several tasks were solved:

1. An introduction and survey of the scientific area of Data Mining and Knowledge Discovery and especially of the CAR Algorithms is made. Because most of examined classification models focus on categorical attributes, a short examination of discretization methods is made. A brief overview of already existing open source data mining environments aimed to support research work, as well as implications on real work, is presented.
2. An overview of different kinds of access methods is made. A taxonomy of access methods defining clearly the place of Multi-Dimensional Numbered Information Spaces in the frame of this taxonomy is shown. A comprehensive analysis of the possibilities of Multi-Dimensional Numbered Information Spaces, their main elements and functions is presented.
3. Two specialized algorithms, PGN and MPGN, are developed and a detailed theoretical description is provided.

4. A software of the proposed algorithms within the frame of the global data mining environment PaGaNe has been developed.
5. A comprehensive example, based on the Lenses Dataset, which illustrates the work of the algorithms, was shown.
6. A sensitivity analysis of the PGN and MPGN algorithms was made. We have carried out experiments with 25 datasets from UCI machine learning repository. The experiments were made using data mining environment PaGaNe, knowledge analysis system Weka, and LUCS-KDD Repository.
7. The experiments that study received accuracy when preprocess real data with different discretizators show Chi-merge with 95% significance level as best appropriate.
8. The analysis of exit points of MPGN showed that in most cases the build constructs lead to excluding only one class as best competitor. Comparing the results of strategies S1 and S2 shows the preference of choosing rule-set criterion against one rule in the competition.
9. The analysis of dependency of classifiers' behaviors when the noise rush in the dataset attributes shows that noising in the dataset significantly worsens the accuracy of PGN, which by its construction performs well in clear datasets. But experiments with the other classifiers show that they also did not behave well under noising attacks.
10. The comparison of overall accuracy between PGN, MPGN (two recognition strategies – S1 and S2), CMAR, OneR, JRip, J48 and REPTree using Friedman test showed statistical difference between tested classifiers. The post-hoc Nemenyi test showed that PGN has best overall performance between examined classifiers and MPGN is competitive with J48, JRip and REPTree. The first four classifiers (PGN, CMAR, J48, and MPGN-S2) significantly outperform OneR.
11. The analysis of F-measures for multi-classes datasets showed that PGN and MPGN have not only good recognition accuracy, but also they recognize small classes controversy better in comparison to other classifiers (for instance OneR) which fail to construct recognition model for small class labels.
12. Additional comparisons of PGN and MPGN with already examined as well as other types of classifiers, such as CAR-classifiers (CMAR), Rules (OneR, JRip), Trees (J48; REPTree), Lazy (Ibk, KStar), Bayes (BayesNet, NaiveBayes), Ensemble – Bagging (RandomForest), Support Vector Machines (SMO), Neural Networks (MultilayerPreceptron) is conducted and given in the Appendix.

Main contributions can be summarized as:

- a new CAR-classifier PGN that questions the common approach to prioritize the support over the confidence and focuses on confidence first by retaining only 100% confidence rules has been elaborated;
- a method for effective building and storing of pattern set in multi-layer structure MPGN during the process of associative rule mining using the possibilities of multidimensional numbered information spaces has been developed;
- software of proposed algorithms and structures has been implemented in the frame of data mining environment system PaGaNe;
- the conducted experiments prove the vividness of proposed approaches showing the good performance of PGN and MPGN in comparison with other classifiers from CAR, rules and trees, and especially in the case of multi-class datasets with uneven distribution of the class labels.

9.2 Directions for Future Research

This work highlighted some possible directions for further research which could tackle areas such as:

- implementing PGN pruning and recognition ideas over pyramidal structures of MPGN;
- improving the pruning part of PGN in order to accommodate phenomena that for some datasets increasing the amount of learning set leads to increasing of the number of pattern set but without increasing of accuracy;
- analyzing different variants of pruning and recognition algorithms based on statistical evidence, structured over already created pyramidal structures of patterns in order to achieve better recognition results;
- proposing different techniques for rule quality measure taking into account confidence of the rule in order to overcome the process of rejecting one rule preferring other one, rarely observed in the dataset;
- expanding the functionalities of the data mining environment PaGaNe for automatic subset selection;
- testing the possibilities of MPGN using exit-1 recognition in the field of campaign management;
- applying the established algorithms PGN and MPGN in different application areas such as business intelligence or global monitoring.

10 Appendix

10.1 Results of 5-fold Cross Validation for Different Classifiers

The results of 5-fold cross-validation for following classifiers:

- PGN group – PGN, MPGN-S1, MPGN-S2;
- CARs – CMAR;
- Rules – OneR, JRip;
- Trees – J48; REPTree;
- Lazy – Ibk, KStar;
- Bayes – BayesNet, NaiveBayes;
- Others – RandomForest (Ensemble – Bagging), SMO (Support Vector Machines), MultilayerPreceptron (Neural Networks)

Table 32. The accuracy of 5-fold cross-validation for classifiers, representatives of PGN-group, CARs, Rules, Trees, Lazy, Bayes, SVM, and Neural Networks

	PGN	MPGN-S1	MPGN-S2	CMAR	One R	Jrip	J48	REP Tree	IB k	K Star	Bayes Net	Naïve Bayes	Random Forest	SMO	Multil. Perc.
audiology01	77.50	67.5	67.5	35.9	45.00	67.50	70.00	57.50	80.00	72.50	67.50	60.00	77.50	72.50	77.50
audiology02	82.50	72.5	72.5	72.5	50.00	80.00	85.00	65.00	85.00	85.00	82.50	67.50	67.50	82.50	85.00
audiology03	65.00	57.5	57.5	50	50.00	60.00	55.00	62.50	65.00	70.00	55.00	57.50	70.00	70.00	62.50
audiology04	75.00	72.5	72.5	70	47.50	67.50	77.50	65.00	77.50	77.50	75.00	70.00	80.00	77.50	85.00
audiology05	77.50	75	75	67.5	42.50	72.50	72.50	62.50	75.00	75.00	75.00	67.50	72.50	80.00	82.50
audiology	**75.50**	**69.00**	**69.00**	**59.18**	**47.00**	**69.50**	**72.00**	**62.50**	**76.50**	**76.00**	**71.00**	**64.50**	**73.50**	**76.50**	**78.50**
balance_scale01	80.65	81.45	83.06	87.10	59.68	70.16	65.32	66.13	86.29	85.48	87.10	87.10	75.81	87.10	100.00
balance_scale02	72.00	74.40	76.80	82.40	56.00	69.60	67.20	67.20	79.20	81.60	87.20	87.20	74.40	88.00	100.00
balance_scale03	80.00	84.80	87.20	87.20	61.60	72.80	68.80	71.20	88.00	89.60	92.80	92.80	73.60	90.40	98.40
balance_scale04	81.60	84.80	88.00	90.40	64.00	72.80	64.00	68.00	91.20	92.00	94.40	94.40	82.40	90.40	99.20
balance_scale05	75.20	81.60	82.40	86.40	59.20	74.40	65.60	63.20	81.60	84.80	91.20	91.20	71.20	91.20	96.00
balance_scale	**77.89**	**81.41**	**83.49**	**86.70**	**60.10**	**71.95**	**66.18**	**67.15**	**85.26**	**86.70**	**90.54**	**90.54**	**75.48**	**89.42**	**98.72**
blood_transfusion01	57.05	73.15	73.15	74.50	77.18	74.50	75.17	77.18	75.84	76.51	75.17	75.84	75.17	76.51	75.17
blood_transfusion02	63.76	78.52	78.52	79.20	78.52	79.19	79.19	79.19	75.17	78.52	76.51	76.51	77.18	78.52	76.51
blood_transfusion03	70.67	78.67	78.67	78.67	79.33	78.67	78.67	79.33	77.33	79.33	70.67	72.67	77.33	79.33	71.33
blood_transfusion04	70.00	66.67	66.67	66.67	66.00	66.67	66.67	66.00	67.33	66.67	65.33	65.33	66.67	66.67	64.67
blood_transfusion05	66.67	81.33	81.33	81.33	82.00	81.33	81.33	82.00	82.00	82.00	78.00	78.00	82.00	82.00	82.00
blood_transfusion	**65.63**	**75.67**	**75.67**	**76.07**	**76.61**	**76.07**	**76.21**	**76.74**	**75.53**	**76.61**	**73.14**	**73.67**	**75.67**	**76.61**	**73.94**
breast_cancer_wo01	97.12	94.24	94.96	94.24	93.53	92.81	92.81	94.24	96.40	96.40	97.12	97.12	96.40	93.53	98.56
breast_cancer_wo02	99.29	95.71	96.43	96.43	95.00	95.71	97.86	97.86	97.14	97.14	99.29	99.29	96.43	99.29	99.29

breast_cancer_wo03	96.43	92.14	92.86	93.57	90.71	92.86	92.86	92.86	94.29	93.57	95.71	95.71	94.29	95.00	93.57
breast_cancer_wo04	94.29	88.57	90	90.71	87.86	90.71	92.14	90.71	92.86	92.14	95.71	95.71	94.29	95.00	93.57
breast_cancer_wo05	95.00	93.57	93.57	94.29	92.14	94.29	95.71	94.29	98.57	97.14	97.86	97.86	95.71	97.14	96.43
breast_cancer_wo	**96.43**	**92.85**	**93.56**	**93.85**	**91.85**	**93.28**	**94.28**	**93.99**	**95.85**	**95.28**	**97.14**	**97.14**	**95.42**	**95.99**	**96.28**
car01	93.04	83.19	87.25	85.8	73.33	89.57	92.75	89.86	95.36	89.86	87.54	87.54	95.65	92.75	100.00
car02	94.20	82.9	86.96	80.87	71.88	82.90	89.86	87.83	93.04	88.70	86.38	86.38	91.30	95.07	100.00
car03	92.77	84.68	86.42	82.37	68.79	89.02	91.62	89.02	91.33	85.55	84.39	84.10	93.35	91.62	100.00
car04	90.75	81.21	83.82	78.9	65.90	85.55	89.88	86.71	92.20	84.68	82.95	82.66	94.51	91.62	99.42
car05	92.20	82.37	84.1	80.92	70.23	86.71	89.88	87.57	92.77	85.26	85.26	85.26	92.77	91.91	99.71
car	**92.59**	**82.87**	**85.71**	**81.77**	**70.03**	**86.75**	**90.80**	**88.20**	**92.94**	**86.81**	**85.30**	**85.19**	**93.52**	**92.59**	**99.83**
cmc01	52.38	50	50	61.22	50.00	56.12	59.18	57.82	53.06	55.44	55.10	55.44	53.40	60.88	49.32
cmc02	48.64	41.5	42.52	48.98	47.28	48.64	50.00	46.94	42.18	46.26	53.40	53.40	47.96	51.36	47.96
cmc03	44.41	41.02	43.39	48.81	44.07	49.83	47.46	46.78	44.41	45.08	44.07	44.41	45.76	48.47	42.37
cmc04	49.83	47.12	47.46	51.19	45.08	46.44	50.17	46.78	48.14	53.22	49.83	49.83	49.15	51.53	47.12
cmc05	54.24	50.51	49.83	55.59	49.83	50.85	51.19	52.54	47.80	51.53	49.15	49.15	47.12	55.25	51.86
cmc	**49.90**	**46.03**	**46.64**	**53.16**	**47.25**	**50.38**	**51.60**	**50.17**	**47.12**	**50.31**	**50.31**	**50.45**	**48.68**	**53.50**	**47.73**
credit01	87.68	85.51	86.23	87.68	86.23	84.78	84.78	86.23	82.61	85.51	85.51	84.78	85.51	89.86	89.13
credit02	89.86	85.51	86.23	86.23	82.61	82.61	82.61	82.61	78.99	81.16	89.13	89.13	85.51	88.41	85.51
credit03	86.96	86.96	86.96	86.23	86.96	87.68	88.41	86.96	84.78	84.78	86.23	85.51	84.78	84.06	84.06
credit04	87.68	84.06	84.78	87.68	84.06	84.78	83.33	81.88	81.16	84.78	84.78	86.23	86.23	83.33	88.41
credit05	85.51	86.23	86.23	87.68	87.68	85.51	87.68	87.68	86.96	87.68	86.23	86.23	85.51	84.06	83.33
credit	**87.54**	**85.65**	**86.09**	**87.10**	**85.51**	**85.07**	**85.36**	**85.07**	**82.90**	**84.78**	**86.38**	**86.38**	**85.51**	**85.94**	**86.09**
ecoli01	76.12	76.12	77.61	74.63	55.22	76.12	71.64	77.61	76.12	76.12	79.10	77.61	76.12	80.60	76.12
ecoli02	83.58	79.10	79.10	85.07	65.67	80.59	77.61	76.12	80.59	79.11	83.58	88.06	80.59	88.06	77.61
ecoli03	77.61	80.60	80.60	83.58	58.21	83.58	82.09	83.58	80.59	85.07	91.04	91.04	80.59	86.57	86.57
ecoli04	79.11	76.12	76.12	85.07	64.18	85.08	77.62	82.09	83.58	83.58	89.55	88.06	86.57	86.57	86.57
ecoli05	82.36	66.18	66.18	77.94	58.82	75.00	76.47	76.47	77.94	77.94	79.41	79.41	77.94	79.41	76.47
ecoli	**79.76**	**75.62**	**75.92**	**81.26**	**60.42**	**80.07**	**77.09**	**79.17**	**79.76**	**80.36**	**84.54**	**84.84**	**80.36**	**84.24**	**80.67**
forestfires01	57.28	59.22	57.28	66.02	53.39	59.22	51.46	46.60	62.14	61.17	56.31	56.31	64.08	65.05	54.37
forestfires02	59.22	56.31	59.22	59.22	55.34	58.25	51.46	52.43	54.37	53.40	56.31	56.31	61.17	56.31	55.34
forestfires03	53.40	56.31	52.43	53.4	50.49	57.28	56.31	53.40	61.17	56.31	54.37	55.34	54.37	53.40	54.37
forestfires04	56.73	61.54	62.5	64.42	55.76	50.00	58.65	61.54	56.73	59.62	60.58	59.62	55.77	63.46	63.46
forestfires05	61.54	46.15	49.04	50.96	51.92	49.04	51.92	55.77	49.04	52.88	63.46	62.50	56.73	67.31	62.50
forestfires	**57.63**	**55.91**	**56.09**	**58.80**	**53.38**	**54.76**	**53.96**	**53.95**	**56.69**	**56.68**	**58.21**	**58.02**	**58.42**	**61.11**	**58.01**
glass01	80.95	83.33	83.33	78.57	54.76	76.19	76.19	66.67	80.95	83.33	83.33	80.95	80.95	80.95	78.57
glass02	76.74	79.07	79.07	79.07	53.49	58.14	74.42	69.77	79.07	79.07	74.42	74.42	76.74	79.07	79.07
glass03	81.40	79.07	76.74	79.07	53.49	69.77	79.07	65.12	79.07	76.74	72.09	72.09	72.09	76.74	69.77
glass04	76.74	81.4	81.4	74.42	46.51	62.79	65.12	62.79	74.42	76.74	67.44	67.44	74.42	72.09	67.44
glass05	76.74	79.07	81.4	79.07	65.12	65.12	72.09	72.09	81.40	79.07	74.42	76.74	76.74	76.74	76.74
glass	**78.51**	**80.39**	**80.39**	**78.04**	**54.67**	**66.40**	**73.38**	**67.29**	**78.98**	**78.99**	**74.34**	**74.33**	**76.19**	**77.12**	**74.32**
haberman01	49.18	72.13	72.13	72.13	72.13	72.13	72.13	72.13	73.77	72.13	80.32	80.33	73.77	72.13	80.33
haberman02	47.54	68.85	68.85	68.85	70.49	70.49	70.49	72.13	72.13	73.77	72.13	72.13	72.13	70.49	72.13
haberman03	78.69	78.69	78.69	75.41	78.69	81.97	78.69	81.97	80.33	80.33	81.97	80.33	78.69	81.97	80.33
haberman04	57.38	75.41	75.41	72.13	72.13	72.13	75.41	75.41	80.33	78.69	80.33	80.33	78.69	72.13	80.33
haberman05	43.55	70.97	70.97	70.97	70.97	69.35	69.35	69.35	70.97	72.58	70.97	70.97	74.19	69.35	74.19
haberman	**55.27**	**73.21**	**73.21**	**71.90**	**72.88**	**73.21**	**73.21**	**74.20**	**75.51**	**75.50**	**77.14**	**76.82**	**75.49**	**73.21**	**77.46**
hayes-roth01	96.15	76.92	76.92	92.31	50.00	92.31	76.92	88.46	69.23	65.38	88.46	88.46	76.92	88.46	92.31
hayes-roth02	84.62	73.08	76.92	88.46	53.85	84.62	69.23	76.92	76.92	69.23	92.31	92.31	84.62	88.46	88.46
hayes-roth03	80.77	53.85	57.69	80.77	50.00	69.23	65.38	61.54	50.00	53.85	84.62	84.62	80.77	80.77	84.62
hayes-roth04	74.07	55.56	55.56	77.78	48.15	70.37	59.26	59.26	48.15	48.15	81.48	81.48	66.67	81.48	74.07
hayes-roth05	74.07	66.67	70.37	77.78	51.85	74.07	70.37	81.48	74.07	70.37	81.48	81.48	74.07	77.78	77.78
hayes-roth	**81.94**	**65.22**	**67.49**	**83.42**	**50.77**	**78.12**	**68.23**	**73.53**	**63.67**	**61.40**	**85.67**	**85.67**	**76.61**	**83.39**	**83.45**
hepatitis01	70.97	74.19	74.19	77.42	80.65	74.19	74.19	77.42	83.87	77.42	80.65	80.65	87.10	67.74	70.97
hepatitis02	77.42	83.87	83.87	83.87	80.65	77.42	80.65	70.97	80.65	80.65	87.10	87.10	77.42	80.65	87.10
hepatitis03	83.87	80.65	80.65	87.1	83.87	83.87	90.32	83.87	83.87	80.65	83.87	87.10	83.87	80.65	80.65
hepatitis04	83.87	90.32	90.32	90.32	80.65	77.42	80.65	83.87	80.65	83.87	93.55	96.77	83.87	83.87	87.10
hepatitis05	87.10	80.65	80.65	83.87	83.87	74.19	70.97	80.65	77.42	80.65	80.65	80.65	83.87	74.19	80.65
hepatitis	**80.65**	**81.94**	**81.94**	**84.52**	**81.94**	**77.42**	**79.36**	**79.36**	**81.29**	**80.65**	**85.16**	**86.45**	**83.23**	**77.42**	**81.29**
iris01	96.67	96.67	96.67	90.00	96.67	90.00	96.67	90.00	90.00	90.00	90.00	90.00	96.67	90.00	96.67
iris02	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33	93.33
iris03	93.33	93.33	93.33	90.00	93.33	90.00	93.33	93.33	93.33	93.33	90.00	90.00	93.33	93.33	93.33
iris04	80.00	86.67	86.67	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00	90.00
iris05	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
iris	**92.67**	**94.00**	**94.00**	**92.67**	**94.67**	**92.67**	**94.67**	**93.33**	**93.33**	**93.33**	**92.67**	**92.67**	**94.67**	**93.33**	**94.67**
lenses01	50.00	75.00	75.00	100.00	50.00	75.00	75.00	100.00	50.00	50.00	50.00	50.00	50.00	50.00	50.00
lenses02	80.00	80.00	80.00	80.00	60.00	80.00	80.00	80.00	100.00	100.00	80.00	80.00	80.00	80.00	80.00
lenses03	60.00	60.00	60.00	60.00	20.00	60.00	60.00	60.00	80.00	80.00	60.00	60.00	60.00	60.00	60.00
lenses04	80.00	100.00	100.00	100.00	100.00	100.00	100.00	80.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
lenses05	100.00	100.00	100.00	100.00	80.00	100.00	100.00	80.00	60.00	60.00	60.00	60.00	80.00	60.00	80.00
lenses	**74.00**	**83.00**	**83.00**	**88.00**	**62.00**	**83.00**	**83.00**	**80.00**	**78.00**	**78.00**	**70.00**	**70.00**	**74.00**	**70.00**	**74.00**
monks101	100.00	94.19	93.02	100.00	67.44	100.00	90.70	94.19	98.84	98.84	67.44	67.44	98.84	67.44	100.00
monks102	100.00	77.91	83.72	100.00	75.58	75.58	100.00	79.07	97.67	97.67	75.58	75.58	91.86	75.58	100.00
monks103	100.00	89.53	97.67	100.00	74.42	100.00	93.02	100.00	100.00	100.00	74.42	74.42	100.00	74.42	100.00
monks104	100.00	75.86	78.16	100.00	82.76	87.36	100.00	85.06	96.55	96.55	82.76	82.76	94.25	82.76	100.00
monks105	100.00	77.01	77.01	100.00	74.71	74.71	89.66	86.21	96.55	96.55	74.72	74.71	96.55	74.71	100.00
monks1	**100.00**	**82.90**	**85.92**	**100.00**	**74.98**	**87.53**	**94.68**	**88.91**	**97.92**	**97.92**	**74.98**	**74.98**	**96.30**	**74.98**	**100.00**
monks201	77.50	77.50	80.00	62.50	66.67	62.50	66.67	67.50	72.50	80.00	63.33	63.33	68.33	66.67	100.00
monks202	71.67	80.83	79.17	63.33	64.17	54.17	50.00	65.83	73.33	77.50	62.50	63.33	68.33	64.17	100.00
monks203	75.00	81.67	83.33	65.00	68.33	59.17	65.00	66.67	71.67	78.33	61.67	61.67	62.50	68.33	100.00

monks204	75.00	77.50	77.50	54.17	65.83	54.17	54.17	56.67	71.67	74.17	62.50	62.50	61.67	65.83	100.00
monks205	66.12	85.12	85.12	53.72	63.64	63.64	63.64	62.81	68.60	74.38	56.20	56.20	66.12	63.64	100.00
monks2	**73.06**	**80.52**	**81.02**	**59.74**	**65.73**	**58.73**	**59.90**	**63.90**	**71.55**	**76.88**	**61.24**	**61.41**	**65.39**	**65.73**	**100.00**
monks301	99.09	90.00	94.55	99.09	83.64	99.09	99.09	99.09	98.18	98.18	96.36	96.36	98.18	96.36	99.09
monks302	100.00	94.59	97.30	100.00	81.08	100.00	100.00	100.00	100.00	100.00	98.20	98.20	99.10	98.20	100.00
monks303	96.40	90.09	90.09	98.20	81.08	98.20	98.20	98.20	96.40	96.40	96.40	96.40	96.40	98.20	98.20
monks304	99.10	90.99	93.69	99.10	79.28	99.10	99.10	99.10	98.20	98.20	97.30	97.30	98.20	97.30	99.10
monks305	98.20	86.49	91.89	98.20	74.77	98.20	98.20	98.20	95.50	96.40	93.69	93.69	98.20	93.69	98.20
monks3	**98.56**	**90.43**	**93.50**	**98.92**	**79.97**	**98.92**	**98.92**	**98.92**	**97.66**	**97.84**	**96.39**	**96.39**	**98.02**	**96.75**	**98.92**
post-operative01	61.11	55.56	50.00	38.89	77.78	77.78	77.78	77.78	66.67	72.22	72.22	72.22	61.11	77.78	66.67
post-operative02	55.56	38.89	44.44	50.00	61.11	61.11	61.11	61.11	61.11	61.11	61.11	61.11	55.56	61.11	55.56
post-operative03	77.78	66.67	66.67	72.22	77.78	77.78	77.78	77.78	77.78	77.78	83.33	83.33	83.33	77.78	61.11
post-operative04	66.67	50.00	50.00	38.89	61.11	66.67	66.67	66.67	66.67	66.67	66.67	66.67	66.67	66.67	61.11
post-operative05	72.22	50.00	50.00	55.56	66.67	66.67	72.22	72.22	66.67	66.67	66.67	66.67	55.56	66.67	55.56
post-operative	**66.67**	**52.22**	**52.22**	**51.11**	**68.89**	**70.00**	**71.11**	**71.11**	**67.78**	**68.89**	**70.00**	**70.00**	**64.45**	**70.00**	**60.00**
soybean01	95.08	88.52	88.52	83.61	37.70	90.16	90.16	81.97	93.44	93.44	88.52	81.97	93.44	91.80	91.80
soybean02	91.80	77.05	77.05	78.69	36.07	85.25	90.16	80.33	91.80	93.44	85.25	86.89	90.16	90.16	91.80
soybean03	90.16	75.41	75.41	67.21	32.79	81.97	86.89	73.77	85.25	88.52	88.52	85.25	86.89	86.89	91.80
soybean04	93.55	91.94	91.94	87.10	41.94	88.71	91.94	77.42	91.94	93.55	83.87	80.65	91.94	96.77	93.55
soybean05	95.16	87.10	87.10	75.81	38.71	80.65	79.03	77.42	91.94	90.32	85.48	79.03	87.10	88.71	91.94
soybean	**93.15**	**84.00**	**84.00**	**78.48**	**37.44**	**85.35**	**87.64**	**78.18**	**90.87**	**91.85**	**86.33**	**82.76**	**89.91**	**90.87**	**92.18**
tae01	50.00	46.67	46.67	40.00	43.33	43.33	43.33	40.00	56.67	53.33	46.67	46.67	43.33	53.33	50.00
tae02	53.33	50.00	50.00	30.00	50.00	30.00	40.00	30.00	53.33	46.67	43.33	46.67	36.67	46.67	53.33
tae03	50.00	40.00	40.00	26.67	43.33	26.67	46.67	40.00	50.00	50.00	40.00	36.67	40.00	43.33	50.00
tae04	53.33	60.00	60.00	43.33	56.67	36.67	50.00	56.67	56.67	63.33	53.33	53.33	63.33	56.67	60.00
tae05	58.06	67.74	67.74	38.71	35.48	35.48	54.84	35.48	70.97	64.52	48.39	51.61	61.29	58.06	61.29
tae	**52.94**	**52.88**	**52.88**	**35.74**	**45.76**	**34.43**	**46.97**	**40.43**	**57.53**	**55.57**	**46.34**	**46.99**	**48.92**	**51.61**	**54.92**
tic_tac_toe01	85.86	94.76	94.24	99.48	65.45	97.38	85.34	83.77	98.43	95.29	69.11	68.59	90.58	98.43	96.34
tic_tac_toe02	89.53	95.81	96.86	100	68.59	99.48	81.15	70.68	97.38	95.81	68.59	68.59	92.67	100.00	98.96
tic_tac_toe03	86.98	95.31	92.71	98.44	68.23	98.96	82.29	79.69	96.88	94.27	68.23	68.23	92.19	98.96	98.96
tic_tac_toe04	89.58	98.96	98.44	97.4	71.88	96.35	82.81	82.81	95.83	95.31	72.92	72.92	89.58	96.35	96.35
tic_tac_toe05	92.71	95.83	95.83	98.44	75.52	97.92	89.58	84.90	98.44	95.83	78.13	78.13	90.63	97.92	98.44
tic_tac_toe	**88.93**	**96.13**	**95.62**	**98.75**	**69.93**	**98.02**	**84.23**	**80.37**	**97.39**	**95.30**	**71.40**	**71.29**	**91.13**	**98.33**	**97.81**
wine01	97.14	100.00	100.00	97.14	68.57	97.14	82.86	85.71	100.00	100.00	100.00	100.00	94.29	100.00	100.00
wine02	100.00	94.29	97.14	97.14	85.71	82.86	82.86	82.86	100.00	100.00	100.00	100.00	97.14	100.00	97.14
wine03	94.44	86.11	86.11	83.33	86.11	91.67	91.67	86.11	88.89	88.89	97.22	94.44	91.67	97.22	97.22
wine04	88.89	83.33	88.89	80.89	66.67	88.89	88.89	91.67	94.44	94.44	100.00	100.00	91.67	94.44	97.22
wine05	100.00	97.22	97.22	100.00	86.11	91.67	88.89	94.44	97.22	97.22	100.00	100.00	97.22	100.00	97.22
wine	**96.09**	**92.19**	**93.87**	**91.70**	**78.63**	**90.45**	**87.03**	**88.16**	**96.11**	**96.11**	**99.44**	**98.89**	**94.40**	**98.33**	**97.76**
winequality-red01	63.32	56.74	57.37	57.68	57.37	52.35	58.62	55.17	64.26	63.95	57.99	57.99	63.95	59.25	67.40
winequality-red02	65.63	60.31	60.94	53.75	56.88	54.38	56.25	54.69	68.75	66.88	58.44	58.75	68.13	60.31	63.13
winequality-red03	66.25	62.19	60.31	53.44	51.25	50.63	55.31	57.81	63.44	62.50	56.88	58.13	65.00	56.56	67.50
winequality-red04	66.56	60.63	61.56	60.31	57.81	58.13	60.94	62.81	65.31	67.50	59.06	58.13	62.81	59.69	60.63
winequality-red05	63.13	56.88	57.81	56.25	54.38	53.13	60.00	54.69	59.69	62.50	60.00	60.00	61.88	59.38	61.56
winequality-red	**64.98**	**59.35**	**59.60**	**56.29**	**55.54**	**53.72**	**58.22**	**57.03**	**64.29**	**64.67**	**58.47**	**58.60**	**64.35**	**59.04**	**64.04**
zoo01	100.00	90.00	90.00	95.00	70.00	85.00	95.00	80.00	100.00	100.00	95.00	95.00	95.00	100.00	100.00
zoo02	100.00	95.00	95.00	95.00	70.00	75.00	95.00	75.00	95.00	95.00	95.00	95.00	95.00	95.00	95.00
zoo03	100.00	90.00	90.00	100.00	80.00	100.00	95.00	85.00	100.00	100.00	100.00	95.00	100.00	100.00	100.00
zoo04	100.00	95.00	95.00	100.00	75.00	100.00	100.00	90.00	100.00	100.00	100.00	95.00	100.00	100.00	100.00
zoo05	90.48	76.19	76.19	80.95	71.43	80.95	85.71	80.95	85.71	85.71	90.48	90.48	90.48	95.24	85.71
zoo	**98.10**	**89.24**	**89.24**	**94.19**	**73.29**	**88.19**	**94.14**	**82.19**	**96.14**	**96.14**	**96.10**	**94.10**	**96.10**	**98.05**	**96.14**

10.2 Confusion Matrices, Recall, Precision and F-measure for Some Multi-class Datasets

Here is given the precise information for the results of examined classifiers for following datasets:

- Glass;
- Soybean;
- Winequality-red.

The class labels are ordered by decreasing of the support and alphabetically for class labels with equal support. The class labels and their supports are given respectively in Table 26, Table 28, and Table 30.

Table 33. Confusion matrices, recalls, precisions and F-measures for Glass dataset

PGN	2#	1#	7#	3#	5#	6#	actual
2#	61	7	0	1	7	0	76
1#	10	57	1	2	0	0	70
7#	0	1	26	0	1	1	29
3#	3	7	0	7	0	0	17
5#	2	0	2	0	9	0	13
6#	0	0	0	0	1	8	9
predicted	76	72	29	10	18	9	214

recall	precision	f-measure
0.803	0.803	0.803
0.814	0.792	0.803
0.897	0.897	0.897
0.412	0.700	0.519
0.692	0.500	0.581
0.889	0.889	0.889

MPGN-S1	2#	1#	7#	3#	5#	6#	actual
2#	60	8	1	3	4	0	76
1#	6	58	1	5	0	0	70
7#	1	1	26	1	0	0	29
3#	1	5	0	11	0	0	17
5#	3	0	1	0	9	0	13
6#	0	1	0	0	0	8	9
predicted	71	73	29	20	13	8	214

recall	precision	f-measure
0.789	0.845	0.816
0.829	0.795	0.811
0.897	0.897	0.897
0.647	0.550	0.595
0.692	0.692	0.692
0.889	1.000	0.941

MPGN-S2	2#	1#	7#	3#	5#	6#	actual
2#	60	8	1	3	4	0	76
1#	5	58	1	6	0	0	70
7#	1	1	26	1	0	0	29
3#	1	5	0	11	0	0	17
5#	3	0	1	0	9	0	13
6#	0	1	0	0	0	8	9
predicted	70	73	29	21	13	8	214

recall	precision	f-measure
0.789	0.857	0.822
0.829	0.795	0.811
0.897	0.897	0.897
0.647	0.524	0.579
0.692	0.692	0.692
0.889	1.000	0.941

CMAR	2#	1#	7#	3#	5#	6#	actual
2#	59	8	2	2	3	2	76
1#	8	60	0	2	0	0	70
7#	0	1	28	0	0	0	29
3#	2	10	0	5	0	0	17
5#	3	0	2	0	8	0	13
6#	0	0	1	0	1	7	9
predicted	72	79	33	9	12	9	214

recall	precision	f-measure
0.776	0.819	0.797
0.857	0.759	0.805
0.966	0.848	0.903
0.294	0.556	0.385
0.615	0.667	0.640
0.778	0.778	0.778

OneR	2#	1#	7#	3#	5#	6#	actual
2#	46	26	4	0	0	0	76
1#	11	57	2	0	0	0	70
7#	12	3	14	0	0	0	29

recall	precision	f-measure
0.605	0.548	0.575
0.814	0.533	0.644
0.483	0.609	0.538

3#	7	10	0	0	0	0	17
5#	4	6	3	0	0	0	13
6#	4	5	0	0	0	0	9
predicted	84	107	23	0	0	0	214

0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000

JRip	2#	1#	7#	3#	5#	6#	actual
2#	54	13	1	2	3	3	76
1#	15	49	1	4	0	1	70
7#	3	0	24	0	2	0	29
3#	7	7	0	2	0	1	17
5#	4	1	1	0	7	0	13
6#	0	2	0	0	1	6	9
predicted	83	72	27	8	13	11	214

recall	precision	f-measure
0.711	0.651	0.679
0.700	0.681	0.690
0.828	0.889	0.857
0.118	0.250	0.160
0.538	0.538	0.538
0.667	0.545	0.600

J48	2#	1#	7#	3#	5#	6#	actual
2#	57	9	3	2	3	2	76
1#	9	57	0	4	0	0	70
7#	1	1	24	0	2	1	29
3#	1	12	0	4	0	0	17
5#	1	0	1	0	10	1	13
6#	1	0	0	0	3	5	9
predicted	70	79	28	10	18	9	214

recall	precision	f-measure
0.750	0.814	0.781
0.814	0.722	0.765
0.828	0.857	0.842
0.235	0.400	0.296
0.769	0.556	0.645
0.556	0.556	0.556

REPTree	2#	1#	7#	3#	5#	6#	actual
2#	56	12	3	3	2	0	76
1#	8	55	3	4	0	0	70
7#	4	3	20	0	2	0	29
3#	2	10	0	5	0	0	17
5#	2	7	2	0	2	0	13
6#	1	0	2	0	0	6	9
predicted	73	87	30	12	6	6	214

recall	precision	f-measure
0.737	0.767	0.752
0.786	0.632	0.701
0.690	0.667	0.678
0.294	0.417	0.345
0.154	0.333	0.211
0.667	1.000	0.800

Table 34. Confusion matrices, recalls, precisions and F-measures for Winequality-red dataset

PGN	5#	6#	7#	4#	8#	3#	actual
5#	559	119	2	1	0	0	681
6#	198	398	40	2	0	0	638
7#	17	102	80	0	0	0	199
4#	36	16	1	0	0	0	53
8#	0	9	7	0	2	0	18
3#	6	3	0	1	0	0	10
predicted	816	647	130	4	2	0	1599

recall	precision	f-measure
0.821	0.685	0.747
0.624	0.615	0.619
0.402	0.615	0.486
0.000	0.000	0.000
0.111	1.000	0.200
0.000	0.000	0.000

MPGN-S1	5#	6#	7#	4#	8#	3#	actual
6#	182	329	115	5	5	2	638
5#	493	156	23	3	1	5	681
7#	14	71	107	1	6	0	199
4#	28	18	5	1	0	1	53
8#	1	6	9	0	2	0	18
3#	5	1	4	0	0	0	10
predicted	723	581	263	10	14	8	1599

recall	precision	f-measure
0.285	0.252	0.267
0.229	0.269	0.247
0.538	0.407	0.463
0.019	0.100	0.032
0.111	0.143	0.125
0.000	0.000	0.000

MPGN-S2	5#	6#	7#	4#	8#	3#	actual
5#	505	157	14	1	0	4	681
6#	241	301	91	2	2	1	638
7#	46	54	92	1	6	0	199
4#	31	16	5	0	0	1	53
8#	4	5	7	0	2	0	18
3#	7	1	2	0	0	0	10
predicted	834	534	211	4	10	6	1599

recall	precision	f-measure
0.742	0.606	0.667
0.472	0.564	0.514
0.462	0.436	0.449
0.000	0.000	0.000
0.111	0.200	0.143
0.000	0.000	0.000

CMAR	5#	6#	7#	4#	8#	3#	actual

recall	precision	f-measure

5#	591	80	10	0	0	0	681
6#	323	232	83	0	0	0	638
7#	38	84	77	0	0	0	199
4#	38	14	1	0	0	0	53
8#	4	6	8	0	0	0	18
3#	8	1	1	0	0	0	10
predicted	1002	417	180	0	0	0	1599

0.868	0.590	0.702
0.364	0.556	0.440
0.387	0.428	0.406
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000

OneR	**5#**	**6#**	**7#**	**4#**	**8#**	**3#**	**actual**
5#	456	225	0	0	0	0	681
6#	206	432	0	0	0	0	638
7#	18	181	0	0	0	0	199
4#	23	29	0	0	1	0	53
8#	1	16	0	1	0	0	18
3#	3	7	0	0	0	0	10
predicted	707	890	0	1	1	0	1599

recall	**precision**	**f-measure**
0.670	0.645	0.657
0.677	0.485	0.565
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000

JRip	**5#**	**6#**	**7#**	**4#**	**8#**	**3#**	**actual**
5#	531	143	7	0	0	0	681
6#	312	286	39	1	0	0	638
7#	45	111	42	0	1	0	199
4#	34	18	0	0	0	1	53
8#	7	7	4	0	0	0	18
3#	6	3	0	1	0	0	10
predicted	935	568	92	2	1	1	1599

recall	**precision**	**f-measure**
0.780	0.568	0.657
0.448	0.504	0.474
0.211	0.457	0.289
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000

J48	**5#**	**6#**	**7#**	**4#**	**8#**	**3#**	**actual**
5#	488	179	11	3	0	0	681
6#	194	383	53	4	3	1	638
7#	17	119	60	1	2	0	199
4#	32	19	1	0	1	0	53
8#	0	9	8	1	0	0	18
3#	5	5	0	0	0	0	10
predicted	736	714	133	9	6	1	1599

recall	**precision**	**f-measure**
0.717	0.663	0.689
0.600	0.536	0.567
0.302	0.451	0.361
0.000	0.000	0.000
0.000	0.000	0.000
0.000	0.000	0.000

REPTree	**5#**	**6#**	**7#**	**4#**	**8#**	**3#**	**actual**
5#	442	223	5	11	0	0	681
6#	175	413	46	3	1	0	638
7#	15	126	56	0	2	0	199
4#	24	27	0	1	1	0	53
8#	1	10	6	1	0	0	18
3#	5	5	0	0	0	0	10
predicted	662	804	113	16	4	0	1599

recall	**precision**	**f-measure**
0.649	0.668	0.658
0.647	0.514	0.573
0.281	0.496	0.359
0.019	0.063	0.029
0.000	0.000	0.000
0.000	0.000	0.000

Table 35. Confusion matrices, recalls, precisions and F-measures for Soybean dataset

PGN	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**actual**	**Rec.**	**Prec.**	**F-meas.**
a)alt.-leaf-spot	36	1	2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	40	0.900	0.837	0.867
b)brown-spot	0	39	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.975	0.867	0.918
c)frog-eye-leaf-spot	7	1	31	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	40	0.775	0.912	0.838
d)phytophthora-rot	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	1.000	1.000	1.000
e)anthracnose	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	1.000	1.000	1.000
f)brown-stem-rot	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	20	1.000	1.000	1.000
g)bacterial-blight	0	0	0	0	0	0	9	1	0	0	0	0	0	0	0	0	0	0	0	10	0.900	0.900	0.900
h)bacterial-pustule	0	0	0	0	0	0	1	9	0	0	0	0	0	0	0	0	0	0	0	10	0.900	0.900	0.900
i)charcoal-rot	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	10	1.000	1.000	1.000
j)diap.-stem-canker	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	10	1.000	1.000	1.000
k)downy-mildew	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	10	1.000	1.000	1.000
l)phyll.-leaf-spot	0	4	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	10	0.600	0.857	0.706
m)powdery-mildew	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	10	1.000	1.000	1.000
n)purple-seed-stain	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	10	1.000	0.909	0.952
o)rhizoctonia-root-rot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	10	1.000	1.000	1.000

p)cyst-nematode	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	6	1.000	1.000	1.000
q)diap.pod-&-st.blight	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	6	1.000	1.000	1.000
r)herbicide-injury	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	4	1.000	0.800	0.889
s)2-4-d-injury	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0.000	0.000	0.000
predicted	43	45	34	40	20	20	10	10	10	10	10	7	10	11	10	6	6	5	0	307		

MPGN-S1	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**actual**	**Rec.**	**Prec.**	**F-meas.**
a)alt.-leaf-spot	36	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.900	0.679	0.774
b)brown-spot	3	34	2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	40	0.850	0.667	0.747
c)frog-eye-leaf-spot	8	4	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.700	0.824	0.757
d)phytophthora-rot	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	1.000	1.000	1.000
e)anthracnose	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	1.000	0.870	0.930
f)brown-stem-rot	0	1	0	0	0	19	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0.950	1.000	0.974
g)bacterial-blight	0	1	1	0	0	0	6	2	0	0	0	0	0	0	0	0	0	0	0	10	0.600	1.000	0.750
h)bacterial-pustule	0	1	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	10	0.900	0.818	0.857
i)charcoal-rot	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	10	1.000	1.000	1.000
j)diap.-stem-canker	0	2	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	10	0.800	1.000	0.889
k)downy-mildew	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	10	1.000	1.000	1.000
l)phyll.-leaf-spot	4	2	1	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	10	0.300	0.750	0.429
m)powdery-mildew	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	10	1.000	1.000	1.000
n)purple-seed-stain	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	10	1.000	1.000	1.000
o)rhizoctonia-root-rot	1	0	0	0	3	0	0	0	0	0	0	0	0	0	6	0	0	0	0	10	0.600	1.000	0.750
p)cyst-nematode	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	6	1.000	1.000	1.000
q)diap.pod-&-st.blight	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	6	0.833	1.000	0.909
r)herbicide-injury	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	4	0.250	1.000	0.400
s)2-4-d-injury	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.000	0.000	0.000
predicted	53	51	34	40	23	19	6	11	10	8	10	4	10	10	6	6	5	1	0	307			

MPGN-S2	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**actual**	**Rec.**	**Prec.**	**F-meas.**
a)alt.-leaf-spot	36	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.900	0.679	0.774
b)brown-spot	3	34	2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	40	0.850	0.667	0.747
c)frog-eye-leaf-spot	8	4	28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.700	0.824	0.757
d)phytophthora-rot	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	1.000	1.000	1.000
e)anthracnose	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	1.000	0.870	0.930
f)brown-stem-rot	0	1	0	0	0	19	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0.950	1.000	0.974
g)bacterial-blight	0	1	1	0	0	0	6	2	0	0	0	0	0	0	0	0	0	0	0	10	0.600	1.000	0.750
h)bacterial-pustule	0	1	0	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	10	0.900	0.818	0.857
i)charcoal-rot	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	10	1.000	1.000	1.000
j)diap.-stem-canker	0	2	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	10	0.800	1.000	0.889
k)downy-mildew	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	10	1.000	1.000	1.000
l)phyll.-leaf-spot	4	2	1	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	10	0.300	0.750	0.429
m)powdery-mildew	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	10	1.000	1.000	1.000
n)purple-seed-stain	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	10	1.000	1.000	1.000
o)rhizoctonia-root-rot	1	0	0	0	3	0	0	0	0	0	0	0	0	0	6	0	0	0	0	10	0.600	1.000	0.750
p)cyst-nematode	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	6	1.000	1.000	1.000
q)diap.pod-&-st.blight	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	6	0.833	1.000	0.909
r)herbicide-injury	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	4	0.250	1.000	0.400
s)2-4-d-injury	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.000	0.000	0.000
predicted	53	51	34	40	23	19	6	11	10	8	10	4	10	10	6	6	5	1	0	307			

CMAR	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**actual**	**Rec.**	**Prec.**	**F-meas.**
a)alt.-leaf-spot	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	1.000	0.870	0.930
b)brown-spot	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	1.000	0.580	0.734
c)frog-eye-leaf-spot	0	11	16	0	0	2	10	0	0	0	0	0	1	0	0	0	0	0	0	40	0.400	0.727	0.516
d)phytophthora-rot	0	3	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0.850	1.000	0.919
e)anthracnose	0	0	0	0	9	0	0	0	0	0	1	0	0	0	0	0	0	0	0	10	0.900	1.000	0.947
f)brown-stem-rot	0	5	5	0	0	26	4	0	0	0	0	0	0	0	0	0	0	0	0	40	0.650	0.765	0.703
g)bacterial-blight	0	3	1	0	0	3	33	0	0	0	0	0	0	0	0	0	0	0	0	40	0.825	0.660	0.733
h)bacterial-pustule	0	2	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	6	0.667	1.000	0.800
i)charcoal-rot	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	10	1.000	1.000	1.000
j)diap.-stem-canker	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	10	1.000	1.000	1.000
k)downy-mildew	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	10	1.000	0.909	0.952
l)phyll.-leaf-spot	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	10	1.000	1.000	1.000
m)powdery-mildew	3	2	0	0	0	0	3	0	0	0	0	0	1	0	0	0	1	0	0	10	0.100	0.500	0.167
n)purple-seed-stain	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	10	1.000	1.000	1.000
o)rhizoctonia-root-rot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	6	1.000	1.000	1.000
p)cyst-nematode	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	10	1.000	1.000	1.000

q)diap.pod-&-st.blight	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	4	0.500	0.667	0.571
r)herbicide-injury	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	7	0	10	0.700	1.000	0.824
s)2-4-d-injury	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.000	0.000	0.000
predicted	23	69	22	17	9	34	50	4	10	10	11	10	2	10	6	10	3	7	0	307			

OneR	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	actual	Rec.	Prec.	F-meas.
a)alt.-leaf-spot	24	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.600	0.255	0.358
b)brown-spot	18	21	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.525	0.300	0.382
c)frog-eye-leaf-spot	12	2	24	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	40	0.600	0.828	0.696
d)phytophthora-rot	0	0	0	37	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	40	0.925	0.578	0.712
e)anthracnose	2	1	0	0	15	0	0	0	0	0	0	0	0	0	0	0	2	0	0	20	0.750	0.789	0.769
f)brown-stem-rot	7	5	0	5	0	0	0	0	2	1	0	0	0	0	0	0	0	0	0	20	0.000	0.000	0.000
g)bacterial-blight	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0.000	0.000	0.000
h)bacterial-pustule	6	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0.000	0.000	0.000
i)charcoal-rot	0	0	0	4	0	0	0	0	3	3	0	0	0	0	0	0	0	0	0	10	0.300	0.273	0.286
j)diap.-stem-canker	0	0	0	2	0	0	0	0	3	5	0	0	0	0	0	0	0	0	0	10	0.500	0.357	0.417
k)downy-mildew	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0.000	0.000	0.000
l)phyll.-leaf-spot	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0.000	0.000	0.000
m)powdery-mildew	5	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0.000	0.000	0.000
n)purple-seed-stain	5	1	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0.000	0.000	0.000
o)rhizoctonia-root-rot	0	0	0	5	0	0	0	0	3	2	0	0	0	0	0	0	0	0	0	10	0.000	0.000	0.000
p)cyst-nematode	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0.000	0.000	0.000
q)diap.pod-&-st.blight	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	3	0	0	6	0.500	0.500	0.500
r)herbicide-injury	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0.000	0.000	0.000
s)2-4-d-injury	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.000	0.000	0.000
predicted	94	70	29	64	19	0	0	0	11	14	0	0	0	0	0	0	6	0	0	307			

JRip	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	actual	Rec.	Prec.	F-meas.
a)alt.-leaf-spot	19	5	15	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.475	0.905	0.623
b)brown-spot	0	32	6	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	40	0.800	0.653	0.719
c)frog-eye-leaf-spot	0	2	37	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.925	0.463	0.617
d)phytophthora-rot	0	1	1	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.950	0.644	0.768
e)anthracnose	0	0	1	4	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0.750	0.882	0.811
f)brown-stem-rot	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	20	1.000	1.000	1.000
g)bacterial-blight	1	3	3	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0.000	0.000	0.000
h)bacterial-pustule	0	0	2	2	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	10	0.600	1.000	0.750
i)charcoal-rot	0	0	0	1	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	10	0.900	1.000	0.947
j)diap.-stem-canker	1	0	1	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	10	0.800	0.889	0.842
k)downy-mildew	0	0	3	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	0	10	0.700	1.000	0.824
l)phyll.-leaf-spot	0	2	3	3	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	10	0.200	1.000	0.333
m)powdery-mildew	0	1	3	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	10	0.600	1.000	0.750
n)purple-seed-stain	0	1	1	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	10	0.800	1.000	0.889
o)rhizoctonia-root-rot	0	0	2	2	0	0	0	0	0	0	0	0	0	0	6	0	0	0	0	10	0.600	1.000	0.750
p)cyst-nematode	0	2	1	1	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	6	0.333	1.000	0.500
q)diap.pod-&-st.blight	0	0	1	0	2	0	0	0	0	0	0	0	0	0	0	0	3	0	0	6	0.500	1.000	0.667
r)herbicide-injury	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	4	0.500	0.667	0.571
s)2-4-d-injury	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0.000	0.000	0.000
predicted	21	49	80	59	17	20	0	6	9	9	7	2	6	8	6	2	3	3	0	307			

J48	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	actual	Rec.	Prec.	F-meas.
a)alt.-leaf-spot	37	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.925	0.860	0.892
b)brown-spot	3	35	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.875	0.875	0.875
c)frog-eye-leaf-spot	3	1	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.900	0.857	0.878
d)phytophthora-rot	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	1.000	0.909	0.952
e)anthracnose	0	0	0	2	17	1	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0.850	0.944	0.895
f)brown-stem-rot	0	0	1	0	0	19	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0.950	0.826	0.884
g)bacterial-blight	0	0	1	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	10	0.900	0.900	0.900
h)bacterial-pustule	0	0	0	0	0	0	1	9	0	0	0	0	0	0	0	0	0	0	0	10	0.900	0.818	0.857
i)charcoal-rot	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	10	1.000	1.000	1.000
j)diap.-stem-canker	0	0	0	0	1	1	0	0	0	8	0	0	0	0	0	0	0	0	0	10	0.800	1.000	0.889
k)downy-mildew	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	10	1.000	1.000	1.000
l)phyll.-leaf-spot	0	3	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	10	0.700	1.000	0.824
m)powdery-mildew	0	0	0	0	0	1	0	0	0	0	0	0	9	0	0	0	0	0	0	10	0.900	0.900	0.900
n)purple-seed-stain	0	0	0	0	0	1	0	1	0	0	0	0	1	7	0	0	0	0	0	10	0.700	1.000	0.824
o)rhizoctonia-root-rot	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	10	1.000	1.000	1.000
p)cyst-nematode	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0	0	0	6	1.000	0.857	0.923
q)diap.pod-&-st.blight	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	1	3	0	0	6	0.500	1.000	0.667

r)herbicide-injury	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	3	0	4	0.750	0.750	0.750
s)2-4-d-injury	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0.000	0.000	0.000
predicted	43	40	42	44	18	23	10	11	10	8	10	7	10	7	10	7	3	4	0	307			

REPTree	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**actual**	**Rec.**	**Prec.**	**F-meas.**
a)alt.-leaf-spot	17	8	11	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.425	0.630	0.507
b)brown-spot	0	32	5	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.800	0.457	0.582
c)frog-eye-leaf-spot	1	1	34	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.850	0.378	0.523
d)phytophthora-rot	0	1	6	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0.825	0.660	0.733
e)anthracnose	0	1	3	1	12	1	0	0	0	0	0	0	0	0	0	0	2	0	0	20	0.600	1.000	0.750
f)brown-stem-rot	0	2	3	0	0	14	0	0	0	1	0	0	0	0	0	0	0	0	0	20	0.700	0.452	0.549
g)bacterial-blight	1	4	2	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	10	0.200	0.667	0.308
h)bacterial-pustule	0	4	2	1	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	10	0.300	0.750	0.429
i)charcoal-rot	1	0	3	1	0	1	0	0	4	0	0	0	0	0	0	0	0	0	0	10	0.400	1.000	0.571
j)diap.-stem-canker	0	1	5	1	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	10	0.300	0.750	0.429
k)downy-mildew	1	4	3	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0.000	0.000	0.000
l)phyll.-leaf-spot	0	5	2	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0.000	0.000	0.000
m)powdery-mildew	5	2	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0.000	0.000	0.000
n)purple-seed-stain	0	1	1	2	0	1	1	1	0	0	0	0	0	3	0	0	0	0	0	10	0.300	1.000	0.462
o)rhizoctonia-root-rot	0	4	2	1	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	10	0.300	1.000	0.462
p)cyst-nematode	0	0	2	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	0.000	0.000	0.000
q)diap.pod-&-st.blight	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	6	0.500	0.600	0.545
r)herbicide-injury	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	4	0.250	1.000	0.400
s)2-4-d-injury	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.000	0.000	0.000
predicted	27	70	90	50	12	31	3	4	4	4	0	0	0	3	3	0	5	1	0	307			

References

[Achtert et al, 2010] Achtert, E., Kriegel, H.-P., Reichert, L., Schubert, E., Wojdanowski, R., Zimek, A.: Visual evaluation of outlier detection models. In 15th Int. Conf. on Database Systems for Advanced Applications (DASFAA 2010), Tsukuba, Japan, 2010, LNCS, Vol. 5982, 2010, pp.396-399.

[Agarwal et al, 2000] Agarwal, R. Aggarwal, C., Prasad V.: A tree projection algorithm for generation of frequent item-sets. In J. of Parallel and Distributed Computing, 61/3, 2001, pp.350-371.

[Agrawal and Srikant, 1994] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In Proc. 20th Int. Conf. Very Large Data Bases, 1994, pp.487-499.

[Agrawal et al, 1993] Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. Proc. of the ACM SIGMOD Int. Conf. on Management of Data, Washington, DC, 1993, pp. 207-216.

[Aha and Kibler, 1991] Aha, D., Kibler, D.: Instance-based learning algorithms. Machine Learning. No.6, 1991, pp.37-66.

[Alpaydin, 2010] Alpaydin, E.: Introduction to Machine Learning. The MIT Press, Second Ed., 2010.

[Arakelyan et al, 2009] Arakelyan, A., Boyajyan, A., Aslanyan, L., Muradyan, D., Sahakyan, H.: Algorithmic analysis of functional pathways affected by typical and atypical antipsychotics. In Proc. of 7th CSIT, Armenia, 2009, pp.361-363.

[Arge, 2002] Arge, L.: External memory data structures. Part 4, ch. 9 in Handbook of Massive Datasets, Kluwer Academic Publishers, 2002, pp.313-357.

[Ashrafi et al, 2004] Ashrafi, M., Taniar, D. Smith, K.: A new approach of eliminating redundant association rules, LNCS, Vol.3180, 2004, pp.465-474.

[Aslanyan and Sahakyan, 2010] Aslanyan, L., Sahakyan, H.:On structural recognition with logic and discrete analysis. Int. J. Information Theories and Applications, 17/1, 2010, pp.3-9.

[Baralis and Psaila, 1997] Baralis, E., Psaila, G.: Designing templates for mining association rules. Journal of Intelligent Information Systems, 9/1, 1997, pp.7-32.

[Bastide et al, 2000] Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., Lakhal, L.: Mining frequent patterns with counting inference. ACM SIGKDD Explorations Newsletter, 2000

[Bay, 2000] Bay, S.: Multivariate discretization of continuous variables for set mining. In Proc. of the 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2000, pp.315-319.

[Bayardo, 1998] Bayardo, R.: Efficiently mining long patterns from databases. In Proc. of the ACM SIGMOD Int. Conf. on Management of Data, Seattle, Washington, United States, 1998, pp.85-93.

[Bayes, 1763] Bayes, T.: Essay Towards Solving a Problem in the Doctrine of Chances. Encyclopædia Britannica Online. <http://www.britannica.com/EBchecked/topic/678260/Essay-Towards-Solving-a-Problem-in-the-Doctrine-of-Chances>, accessed at 01.03.2011.

[Boser et al, 1992] Boser, B., Guyon, I., Vapnik, V.: A training algorithm for optimal margin classifiers. In Proc. of the Annual Conference on Computational Learning Theory, Pittsburgh, 1992, pp.144-152.

[Bramer, 2007] M. Bramer. Principles of Data Mining. Springer Verlag London, 2007.

[Breiman, 1996] Breiman, L. Bagging predictors. Machine Learning 24/2, 1996, pp.123-140.

[Brin et al, 1997] Brin, S., Motwani, R., Ullman, J., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997, pp.255-264.

[Brin et al, 1997] Brin, S., Motwani, R., Ullman, J., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997, pp.255-264.

[Cendrowska, 1987] Cendrowska, J.: PRISM: An algorithm for inducing modular rules. International Journal of Man-Machine Studies, 1987, 27, pp.349-370.

[Chakrabarti, 2001] Chakrabarti, K.: Managing Large Multidimensional Datasets Inside a Database System. Phd Thesis, University of Illinois at Urbana-Champaign. Urbana, Illinois, 2001.

[Chavez et al, 2001] Chavez, E., Navarro, G., Baeza-Yates, R., Marroquin, J.: Searching in metric spaces. ACM Computing Surveys, 33/3, 2001, pp.273-321.

[Cheung et al, 1996] Cheung, D., Han, J., Ng, V., Fu, A., Fu, Y.: A fast distributed algorithm for mining association rules. In Proc. of 1996 Int. Conf. on Parallel and Distributed Information Systems, Miami Beach, Florida, 1996, pp.31-44.

[Chuang et al, 2005] Chuang, K., Chen, M., Yang, W.: Progressive sampling for association rules based on sampling error estimation, LNCS, Vol.3518, 2005, pp.505-515.

[Cleary and Trigg, 1995] Cleary, J., Trigg, L.: K*: An instance-based learner using an entropic distance measure. In: 12th International Conference on Machine Learning, 1995, pp.108 114.

[CODASYL, 1971] Codasyl Systems Committee. Feature Analysis of Generalized Data Base Management Systems. Technical Report, May, 1971.

[Codd, 1970] Codd, E.: A relation model of data for large shared data banks. Magazine Communications of the ACM, 13/6, 1970, pp.377-387.

[Coenen and Leng, 2005] Coenen, F., Leng, P.: Obtaining Best Parameter Values for Accurate Classification. Proc. ICDM'2005, IEEE, pp.597-600.

[Coenen et al, 2004] Coenen, F., Goulbourne, G., Leng, P.: Tree structures for mining association rules. Data Mining and Knowledge Discovery, Kluwer Academic Publishers, 8, 2004, pp.25-51.

[Cohen, 1995] Cohen, W.: Fast effective rule induction. In Proc. of the 12th Int. Conf. on Machine Learning, Lake Taho, California, Morgan Kauffman, 1995.

[Connolly and Begg, 2002] Connolly, T., Begg, C.: Database Systems. A Practical Approach to Design, Implementation, and Management. Third Edition. Addison-Wesley Longman, 2002.

[Cristofor and Simovici, 2002] Cristofor, L., Simovici, D.: Generating an informative cover for association rules. In Proc. of the IEEE Int. Conf. on Data Mining, 2002.

[Date, 1975] Date, C.: An Introduction to Database Systems. Addison-Wesley Inc. 1975.

[Demsar, 2006] Demsar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res., 7, 2006, pp.1-30.

[Depaire et al, 2008] Depaire, B., Vanhoof, K., Wets, G.: ARUBAS: an association rule based similarity framework for associative classifiers. IEEE Int. Conf. on Data Mining Workshops, 2008, pp.692-699.

[Do et al, 2003] Do, T.D., Hui, S.-C., Fong, A.: Mining frequent item-sets with category-based constraints, LNCS, Vol. 2843, 2003, pp.76-86.

[Dougherty et al, 1995] Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In Proc. of the 12th Int. Conf. on Machine Learning 1995, pp. 194-202.

[Fayyad and Irani, 1993] Fayyad, U., Irani, K.: Multi-interval discretization of continuous-valued attributes for classification learning. Proc. of the 13th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, San Mateo, CA, 1993, pp.1022-1027.

[Fayyad et al, 1996] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From data mining to knowledge discovery: an overview. In Advances in Knowledge Discovery and Data Mining. American Association for AI, Menlo Park, CA, USA, 1996, pp.1-34.

[Frank and Asuncion, 2010] Frank, A. Asuncion, A.: UCI Machine Learning Repository http://archive.ics.uci.edu/ml. Irvine, CA: University of California, School of Information and Computer Science, 2010.

[Friedman, 1940] Friedman, M.: A comparison of alternative tests of significance for the problem of m rankings. Annals of Mathematical Statistics, Vol. 11, 1940, pp.86-92.

[Friedman, 1997] Friedman, J.: Data mining and statistics: what is the connection? Keynote Address, 29th Symposium on the Interface: Computing Science and Statistics, 1997.

[Gaede and Günther, 1998] Gaede, V., Günther, O.: Multidimensional Access Methods. ACM Computing Surveys, 30/2, 1998.

[Goethals, 2002] Goethals, B.: Efficient Frequent Pattern Mining. PhD Thesis, Transnationale Univeriteit Limburg, 2002.

[Han and Kamber, 2006] Han, J., Kamber, M.: Data Mining: Concepts and Techniques. Morgan Kaufman Publ., Elsevier, 2006.

[Han and Pei, 2000] Han, J., Pei, J.: Mining frequent patterns by pattern-growth: methodology and implications. ACM SIGKDD Explorations Newsletter 2/2, 2000, pp.14-20.

[Hilderman and Hamilton, 2002] Hilderman, R., Hamilton, H.: Knowledge Discovery and Interest Measures. Kluwer Academic, Boston, 2002.

[Holte, 1993] Holte, R.: Very simple classification rules perform well on most commonly used datasets. Machine Learning, Vol. 11, 1993, pp.63-91.

[Hussain et al, 1999] Hussain, F., Liu, H., Tan, C., Dash, M.: Discretization: An Enabling Technique. Technical Report – School of Computing, Singapore, 1999.

[IBM, 1965-68] IBM System/360: Disk Operating System, Data Management Concepts. IBM System Reference Library, IBM Corp. 1965, Major Revision, Feb.1968.

[Jaroszewicz and Simovici, 2002] Jaroszewicz, S., Simovici, D.: Pruning redundant association rules using maximum entropy principle, LNCS, Vol. 2336, 2002, pp.135-142.

[Kass, 1980] Kass, G.: An exploratory technique for investigating Large quantities of categorical data. Journal of Applied Statistics 29/2, 1980, pp. 119-127.

[Kerber, 1992] Kerber R.: Discretization of numeric attributes. Proc. of the 10th National Conf. on Artificial Intelligence, MIT Press, Cambridge, MA, 1992, pp.123-128.

[Klosgen and Zytkow, 1996] Klosgen, W, Zytkow, J.: Knowledge discovery in databases terminology. In Advances in Knowledge Discovery and Data Mining. AAAI Press, 1996, pp.573-592.

[Korn and Korn, 1961] Korn, G., Korn, T.: Mathematical Handbook for Scientists and Engineers. McGraw-Hill, 1961.

[Kotsiantis and Kanellopoulos, 2006] Kotsiantis, S., Kanellopoulos, D.: Association rules mining: a recent overview. GESTS International Transactions on Computer Science and Engineering, 32/1, 2006, pp. 71-82.

[Kouamou, 2011] Kouamou, G.: A software architecture for data mining environment. Ch.13 in New Fundamental Technologies in Data Mining, InTech Publ., 2011, pp.241-258.

[Kuncheva, 2004] Kuncheva, L.: Combining Pattern Classifiers: Methods and Algorithms. Willey, 2004.

[Li and Gopalan, 2004] Li, Y., Gopalan, R.: Effective sampling for mining association rules. LNCS, Vol. 3339, 2004, pp.391-401.

[Li et al, 2001] Li, W., Han, J., Pei, J.: CMAR: Accurate and efficient classification based on multiple class-association rules. In: Proc. of the IEEE Int. Conf. on Data Mining ICDM, 2001, pp.369-376.

[Liu et al, 1998] Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In Knowledge Discovery and Data Mining, 1998, pp.80-86.

[Liu et al, 1999] Liu, B. Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. Proc. Knowledge Discovery and Data Mining Conf., 1999, pp.337-341.

[Liu et al, 2003] Liu, G., Lu, H., Lou, W., Yu, J.: On computing, storing and querying frequent patterns. Proc. of the 2003 ACM SIGKDD international conference on knowledge discovery and data mining (KDD'03), Washington, DC, 2003, pp.607-612

[Maimon and Rokach, 2005] Maimon, O., Rokach, L.: Decomposition Methodology for Knowledge Discovery and Data Mining. Vol. 61 of Series in Machine Perception and Artificial Intelligence. World Scientific Press, 2005.

[Markov et al, 2008] Markov K, Ivanova, K., Mitov, I., Karastanev, S.: Advance of the access methods. Int. J. Information Technologies and Knowledge, 2/2, 2008, pp.123-135.

[Markov, 1984] Markov K.: A multi-domain access method. Proc. of the Int. Conf. on Computer Based Scientific Research. Plovdiv, 1984, pp.558-563.

[Markov, 2004] Markov, K.: Multi-domain information model. Int. J. Information Theories and Applications, 11/4, 2004, pp.303-308.

[Markov, 2005] Markov, K.: Building data warehouses using numbered multidimensional information spaces. Int. J. Information Theories and Applications, 2005, 12/2, pp.193-199.

[Martin, 1975] Martin, J.: Computer Data-Base Organization. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[Mitchell, 1997] Mitchell, T.: Machine Learning, McGraw-Hill, 1997.

[Mitov et al, 2009a] Mitov, I., Ivanova, K., Markov, K., Velychko, V., Vanhoof. K., Stanchev, P.: "PaGaNe" – A classification machine learning system based on the multidimensional numbered information spaces. In World Scientific Proc. Series on Computer Engineering and Information Science, No.2, pp.279-286.

[Mitov et al, 2009b] Mitov, I., Ivanova, K., Markov, K., Velychko, V., Stanchev, P., Vanhoof, K.: Comparison of discretization methods for preprocessing data for pyramidal growing network classification method. In Int. Book Series Information Science & Computing – Book No: 14. New Trends in Intelligent Technologies, 2009, pp. 31-39.

[Moënne-Loccoz, 2005] Moënne-Loccoz N.: High-Dimensional Access Methods for Efficient Similarity Queries. Technical Report N:0505, University of Geneva, Computer Vision and Multimedia Laboratory, 2005.

[Mokbel et al, 2003] Mokbel, M., Ghanem, T., Aref, W.: Spatio-temporal access methods. IEEE Data Engineering Bulletin, 26/2, 2003, pp.40-49.

[Morishita and Sese, 2000] Morishita, S., Sese, J.: Transversing itemset lattices with statistical metric pruning. In Proc. of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 2000, pp.226-236.

[Neave and Worthington, 1992] Neave, H., Worthington, P.: Distribution Free Tests. Routledge, 1992.

[Nemenyi, 1963] Nemenyi, P.: Distribution-free multiple comparisons. PhD thesis, Princeton University, 1963

[Ooi et al, 1993] Ooi, B., Sacks-Davis, R., Han, J.: Indexing in Spatial Databases. Technical Report, 1993.

[Parthasarathy et al, 2001] Parthasarathy, S., Zaki, M., Ogihara, M., Li, W.: Parallel data mining for association rules on shared memory systems. Journal Knowledge and Information Systems, 3/1, 2001, pp. 1-29.

[Parthasarathy, 2002] Parthasarathy, S.: Efficient progressive sampling for association rules. Proc. of Int. Conf. on Data Mining, 2002, pp.354-361.

[Quinlan and Cameron-Jones, 1993] Quinlan, J., Cameron-Jones, R.: FOIL: A midterm report. In Proc. of European Conf. On Machine Learning, Vienna, Austria, 1993, pp.3-20.

[Quinlan, 1986] Quinlan, R.: Induction of decision trees. Machine Learning 1/1, 1986, pp.81-106.

[Quinlan, 1993] Quinlan, R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.

[Rak et al, 2005] Rak, R., Stach, W., Zaiane, O., Antonie M.-L.: Considering re-occurring features in associative classifiers. In Advances in Knowledge Discovery and Data Mining, LNCS, Vol. 3518, 2005, pp.240-248.

[Rakotomalala, 2005] Rakotomalala, R.: TANAGRA: a free software for research and academic purposes. In Proc. of EGC'2005, RNTI-E-3, Vol. 2, 2005, pp.697-702 (in French)

[Ravenbrook, 2010] Ravenbrook – software engineering consultancy, 2010, http://www.ravenbrook.com/

[Sarwar et al, 2001] Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: World Wide Web, 2001, pp.285-295.

[Savasere et al, 1995] Savasere, A., Omiecinski, E., Navathe, S.: An efficient algorithm for mining association rules in large databases. The 21st VLDB Conference, 1995, pp.

[Stably, 1970] Stably D.: Logical Programming with System/360. New York, 1970.

[StatTrek, 2011] http://stattrek.com/Lesson3/SamplingTheory.aspx

[Tang and Liao, 2007] Tang, Z., Liao, Q.: A new class based associative classification algorithm. IAENG International Journal of Applied Mathematics, 2007, 36/2, pp.15-19.

[Thabtah et al, 2005] Thabtah, F., Cowling, P., Peng, Y.: MCAR: multi-class classification based on association rule. In Proc. of the ACS/IEEE 2005 Int. Conf. on Computer Systems and Applications, Washington, DC, 2005, p.33.

[Toivonen, 1996] Toivonen, H.: Sampling large databases for association rules. In The VLDB Journal, 1996, pp.134-145.

[Wagner, 1973] Wagner, H.: Begriff. In: Handbuch philosophischer Grundbegriffe, München, Kösel, 1973, pp.191-209.

[Wang and Karypis, 2005] Wang, J., Karypis, G.: HARMONY: Eficiently Mining the Best Rules for Classification. In Proc. of SDM, 2005, pp.205-216.

[Wille, 1982] Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: Ordered Sets, Dordrecht-Boston, Reidel, 1982, pp.445-470.

[Williams, 2009] Williams, G.: Rattle: A data mining GUI for R. The R Journal, 1:2, 2009, pp.45-55. http://rattle.togaware.com/

[Witten and Frank, 2005] Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

[Wu, 1995] Wu, X.: Knowledge Acquisition from Database. Ablex Pulishing Corp., USA, 1995.

[Yin and Han, 2003] Yin, X., Han, J.: CPAR: Classification based on predictive association rules. In SIAM Int. Conf. on Data Mining (SDM'03), 2003, pp.331-335.

[Yuan and Huang, 2005] Yuan, Y., Huang, T.: A matrix algorithm for mining association rules. LNCS, Vol. 3644, 2005, pp.370-379.

[Zaiane and Antonie, 2002] Zaiane, O., Antonie, M.-L.: Classifying text documents by associating terms with text categories. J. Australian Computer Science Communications, 24/2, 2002, pp.215-222.

[Zaiane and Antonie, 2005] Zaiane, O., Antonie, M.-L.: On pruning and tuning rules for associative classifiers. In Proc. of Int. Conf. on Knowledge-Based Intelligence Information & Engineering Systems, LNCS, Vol. 3683, 2005, pp.966-973.

[Zaiane et al, 2000] Zaiane, O., Han, J., Zhu, H.: Mining recurrent items in multimedia with progressive resolution refinement. In Int. Conf. on Data Engineering, 2000, pp.461-470.

[Zimmermann and De Raedt, 2004] Zimmermann, A., De Raedt, L.: CorClass: Correlated association rule mining for classification. In Discovery Science, LNCS, Vol. 3245, 2004, pp.60-72.

Curriculum Vitae

Iliya Georgiev Mitov was born on May 26, 1963 in Sandansky, Bulgaria. After graduating from the secondary school in his home town, from 1983 till 1988 he studied Mathematics at Sofia University "St. Kliment Ohridsky", where he obtained his Master of Science degree in Computer Science.

He has been started as software engineer in the field of business informatics. Later, since 2003 he has been develop own IT company for automation of company management and accountancy. He has participated in several projects in the field of advance of non-governmental organizations in Bulgaria and Balkan region.

The main research interests of Iliya Mitov are: Data Mining, Knowledge Retrieval, Data Bases, Information systems, ERP-systems applied in areas such as: Analysis and Management of Economical and Natural Processes.

He has more than 60 scientific publications in journals and peer-reviewed conferences in the areas of his research interests.

Since 1990 he has been a principal co-developer of the Complex FOI, applied in numerous Bulgarian companies, which provided an excellent framework to investigate workflow, software, and management aspects in the field of business informatics. He also served as a principal co-developer of the data mining environment system PaGaNe, which provides a convenient workplace for examining different tools supporting the process of knowledge discovery.

Since 2009 he is carrying out a doctoral research project at the Hasselt University, which outcomes are presented in this dissertation.

www.uhasselt.be

Universiteit Hasselt | Campus Diepenbeek
Agoralaan | Gebouw D | BE-3590 Diepenbeek | België
Tel.: +32(0)11 26 81 11