Methodological aspects of designing induction-based applications

Verdenius, F.

Publication date
2005
Document Version
Final published version

METHODOLOGICAL ASPECTS OF DESIGNING INDUCTION-BASED APPLICATIONS

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. mr. P.F. van der Heijden
ten overstaan van een door het college voor promoties ingestelde
commissie, in het openbaar te verdedigen in de Aula der Universiteit
op vrijdag 28 januari 2005, te 12.00 uur door

Floor Verdenius
geboren te Heerenveen

Promotiecommissie:

Promotor: prof. dr. B.J. Wielinga
Co-promotor: dr. M.W. van Someren

Overige leden:
Prof. dr. ir. J.L. Top
Prof. dr. A.P.J.M. Siebes
Prof. dr. R. de Hoog
Prof. dr. P.W. Adriaans
Dr. ir. B.J.A. Kröse

Faculteit der Natuurwetenschappen, Wiskunde en Informatica
Universiteit van Amsterdam

SIKS Dissertation Series No. 2005–1

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

Methodological Aspects of Designing Induction-based Applications
ISBN 90-6754-825-1

Voorwoord

Vijf, vier en drie jaar geleden schatte ik de benodigde tijd voor afronding van dit proefschrift telkens op “nog maar één jaar”. Maar wie mij twee jaar geleden had voorspeld dat ik nu, in november 2004, het dankwoord zou schrijven, had ik voor gek verklaard. Ik begon in het najaar van 1994 met enkele ideeën voor een rationelere toepassing van machine learning technieken. Deze ideeën sloten aan bij die van Maarten van Someren. Maarten voorzag dat er ongeveer 10 jaar werk in zou zitten, zelf hoopte ik op een jaar of zes. Hij had duidelijk meer ervaring. Die lange duur kwam niet alleen doordat ik bij tijd en wijle mijn aandacht elders had, maar zeker ook door de complexiteit van het onderwerp. Elke keer als ik de draad weer oppakte, kon ik op Maarten zijn steun rekenen, met ideeën, opmerkingen en met enthousiasme om samen te schrijven. Maarten, er liggen nog wat open eindjes van hoofdstuk 6 en 7, en ik kijk er naar uit die samen op te pakken. Bob Wielinga zorgde er voor dat ik onderweg niet in al te veel details verdronk, en dat ik de hoofdlijn van het werk vasthield. Minder direct, maar niet minder belangrijk om tot een afronding te komen. Maarten en Bob, bedankt.

Daarnaast genoot ik de steun van diverse collega’s, die vaak vrienden werden. Robert Aarts liet mij op een andere manier kijken naar het toepassen van machine learning. Van jouw aanpak om machine learning en domein expertise te combineren heb ik veel geleerd. Robert Engels stond vervolgens mede aan de basis van de methodologische concepten uit de hoofdstukken 1, 5 en 6. Onze avonturen in onder meer Karlsruhe, Nashville, Drenthe en Wageningen zijn dierbare herinneringen. En je aanstekelijke enthousiasme werd node gemist, de laatste tijd. Gelukkig was er toen Hans Schepers. Meer dan een schrijfmaatje: altijd bereid mee te denken over welk onderwerp dan ook. In discussies met jou vond ik een doorbraak voor hoofdstuk 7: prototype matching. Vele anderen droegen steentjes bij: Hans van den Berg, Eric Boer, Thomas de Boer, Rob Broekhof, Jan Broeze (co-auteur van hoofdstuk 4), Jacques Dunselman, Felix Herrmann, Arnold Kraakman, Patrick Oonincx, Jan Paredis, Anneke Polderdijk, Maarten Schipper, Rob Schouten, Mark Sloof, Rob van der Spek, Toine Timmermans, Zbigniew Struzik, Henry van de Valk en Clare Wilkinson. Bedankt, allemaal! Reena Bakker-Dhaliwal ben ik dankbaar voor de correctie van het Engels.

Gedurende de afgelopen 10 jaar werkte ik voor Agrotechnology and Food Innovations b.v. (het voormalige ATO). Hoewel het grootste deel van dit werk in eigen beheer is uitgevoerd, hebben zowel individuele collega’s als het instituut mij gesteund. Op een moment dat ik zelf twijfelde aan voltooiing waren het Jan Top en Hans Maas die me stimuleerden en ruimte gaven nog één keer aan te zetten. En vanaf 2001 hebben Jan Top en Arjen Simons bij herhaling gezorgd voor de financiële ruimte om ook in kantoortijd aan dit onderzoek te werken. Mijn nogal eens schuivende planningshorizon had geen invloed op jullie steun en vertrouwen. Mede door jullie inspanning is hoofdstuk 7 deels gefinancierd uit DLO programma 391 van het ministerie van Landbouw, Natuurbeheer en Voedselkwaliteit.

De benodigde tijd voor dit proefschrift ging zonder enige twijfel ten koste van mijn aandacht voor familie en vrienden. Met de afronding ervan ontstaat er meer ruimte voor jullie.

Tot slot nog enige woorden voor mijn allerliefsten. Marijn, dank je wel dat je me hielp eindelijk eens te leren typen. De laatste hoofdstukken gingen een stuk sneller! Koen en Hanna, de grote mensenverhalen zijn af. De kindercomputer is nu echt voor jullie drieën (als ik hem weer aan de praat krijg). En meer tijd voor sprookjeskastelen, zwemmen, de moestuin, fietsen, molentjes, pannekoeken, de Uiver of paardrijden. Lieve Ina, ik ben nu eindelijk “afgestudeerd” (maar hopelijk niet uitgeleerd). Dank voor je geduld en je steun. Nu jij weer!

Wageningen, november 2004

Contents

1. Introduction

This chapter is based on: F. Verdenius, A.J.M. Timmermans & R.E. Schouten, Process Models for Neural Network Applications in Agriculture, *AI Applications*, 11 (3), 1997, pp. 31-46, and on F. Verdenius & R. Engels, A Process Model for Developing Inductive Applications, in: W. Daelemans (ed.), *Proceedings of Benelearn 1997*, Tilburg, pp. 119-128

1.1. Motivation of this work

This thesis considers methodological support for practical application of machine learning techniques. The methodological support focuses on how to use these techniques in solving real-world problems. As such, the focus lies on how to best design solutions and on technique selection in order to make optimal use of available techniques. It is explicitly not the intention to support technique tuning towards optimal performance on a specific data set, nor on designing new technique variants to solve a specific new problem type.

According to many publications in the last decade, machine learning (ML) is ready for real world application. "*A sign of a science maturing is when its applications start to evolve in number and quality. Machine learning (...) has now reached such a level of maturity. Especially since the beginning of the nineties, ML algorithms, tools and techniques are being used more and more extensively for solving real-world problems.*" (Rudström, 1995). "*Applying machine learning techniques to solve real world problems has gained more and more interest over the last decades.*" (Engels et al, 1997b). "*The ultimate test of machine learning is its ability to produce systems that are used regularly in industry, education, and elsewhere*" (Langley & Simon, 1994). "*In the last decade, we have seen an explosive growth in our capabilities to both generate and collect data.*" (Fayyad et al, 1996). But many agree that some obstacles still have to be overcome. "*Because the objects of study are intended to have practical utility, it is essential for research activities to be focussed (in part) on the elimination of obstacles that impede their practical application*" (Provost & Kohavi, 1998). Moreover, "*it is difficult for new technologies to be accepted, particularly in fields where well-assessed, although limited, classical software development methods exist*" (Saitta & Neri, 1998).

The above citations, by various authors who have worked on the application of applying ML techniques in practice, characterize the starting position of the research in this thesis. In the mid-nineties, a vast amount of ML techniques was available (see Mitchell, 1997; Berthold & Hand, 1999; and Hand et al., 2001 for overviews). The industrial application of ML techniques has become a self-standing research subject. It was the subject of a number of workshops, special events and publications, not only in the ML community (Aha & Riddle, 1995; Kodratoff & Moustakis, 1994; Engels et al., 1997a, 1998) but also in specific domains[1].

[1] Agriculture: IFAC workshops on AI in Agriculture; Lokhorst et al., 1996 ; Verdenius & Hunter, 2001; Holmes & Smith, 2001. Medicine: special sessions on Machine Learning in Medicine and Biology in the *International Conference on Machine Learning and Applications* series. Engineering: Minton, 1993 ; Chang, 2002. Bioinformatics: Hunter, 1993; Baldi & Brunak, 2001, conference series Intelligent Systems for Molecular Biology (ISMB); special session Machine Learning in Bioinformatics on International Conference on Knowledge-Based Intelligent Information & Engineering Systems 2002; Machine Learning in Bioinformatics conferences; Computational Systems Bioinformatics Conference; IEEE Computer Society Bioinformatics Conferences.

Machine learning researchers and technology developers use applications to test new techniques, and to understand the technical opportunities and limitations in practice. The emphasis of their work is not on industrial but on technical aspects of the application. Industrialists, on the other hand, see in ML tools the opportunity to exploit the hidden potential of the ever-expanding databases.

In applied research the main goal is to bring academic concepts towards industrial application. Much of the above mentioned projects have applied ML techniques in a technology-push setting: the developers have a technology and find a way to test the practical applicability of the technology. From an industrial point of view however, the gap between a problem and its solution is not so much the technology as it is the realization of specific application and business goals. Whether it is the implementation of traceability in production and distribution chains, the application of information and communication technology in business processes, the adaptation of mobile communication by the public, or the (non-)use of the abundantly available functions on consumer electronics, the critical success factor is the fit between specific goals in the real world and the functionality of a system.

If machine learning bears the promise of solving industrial problems, there must be instruments that translate the requirements from the real world into requirements of ML techniques. During a number of ML application projects it was apparent that this translation is not always straightforward. And although sometimes adaptation of techniques is required (e.g. Verdenius, 1991; van Someren et al, 1997), technological capabilities are often not by themselves the restrictive constraints. Therefore, this work originates from the need to translate real world requirements into working solutions, where ML techniques play the role of useful resources.

1.2. Machine Learning Application

With the term *machine learning* we refer to computer algorithms that formulate models on the basis of experience (Mitchell, 1997). In their simplest form these programs take a data set, and formulate some model of the data content. Figure 1.1a depicts an explorative setting that is often encountered in *data mining* (DM) or *knowledge discovery in databases* (KDD). For a single data set, a number of models is generated and each model provides new insights in the domain. The interpretation of the model is left to human analysts. Figure 1.1b presents a setting where a model is derived from data and deployed by an interpreter to regularly process new, previously unseen, data. The goal of the learning step is to acquire a static model. Figure 1.1c presents the setup of an *adaptive system*. To improve predictive quality of the learned model, the discrepancy between the factual output and the predicted output on the basis of the learned model is used to re-learn. In this way, a learning system can continuously improve its performance and adapt to a shifting concept. Often these application types are considered to require different forms of methodological support. As a result of the popularity of DM and KDD, various approaches for methodological support have been proposed (Weiss & Kulikowski, 1991; Garner et al., 1995; Adriaans & Zantinge, 1996; Fayyad et al, 1996; Chapman et al., 2000). Methodological support for DM considers the use of machine learning techniques as a stand-alone process. Consequently the mapping between input and output remains simple and singular. Some authors have proposed general approaches without specifying the type of application (DTI, 1994; Brodley & Smyth, 1997). For the adaptive system approaches dedicated methodologies have not been identified. In those cases practitioners are left to combine available general-purpose ML approaches with general software engineering (Sommerville, 1996) and knowledge engineering (Chang, 2002) approaches. In the latter approaches some attention is paid to the definition of the embedding system.

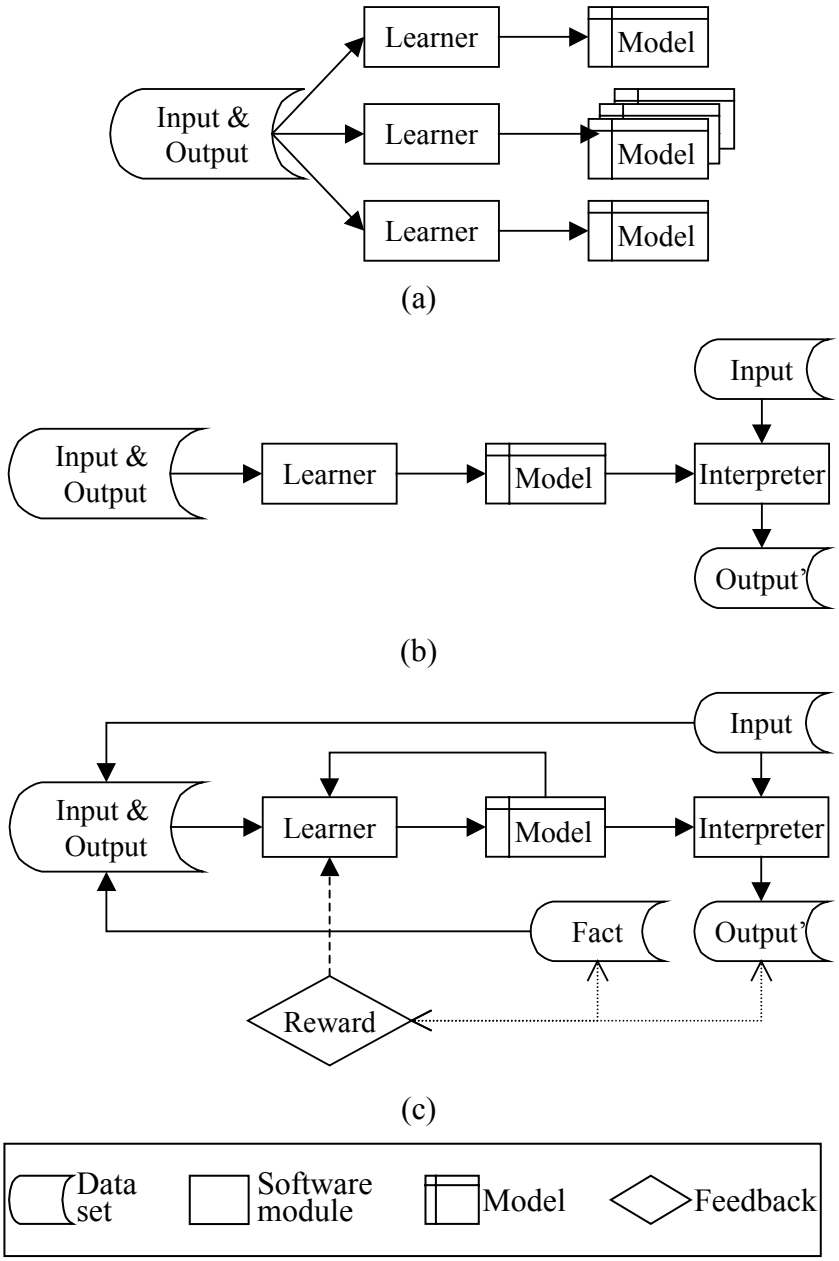


Figure 1.1 Configurations of machine learning techniques for (a) explorative learning, (b) one time knowledge acquisition and (c) adaptive system.

The assumption that underlies this thesis is that one approach can cover all three types of applications. In summary, the result of a ML application project can be a set of models for human interpretation, a static model with an accompanying interpreter, or an adaptive system. In general, we refer to all three types of result with the term *system*.

Typical tasks that ML techniques support include:

- *Classification:* for an instance I, find a class $c \in \{c_1, c_2, \ldots c_n\}$ with finite n,
- *Numerical prediction:* for an instance I, find a real-valued attribute $i \in \Re$ and

Learning is the activity of providing, from data, models to support classification and numerical prediction. Many of the considerations that are discussed in a classification context, e.g. methodological support and technique selection, can be directly translated to techniques for numerical prediction.

Apart from the basic task, classification or numerical prediction, the functionality of a ML application is also defined by the pragmatic deployment. Kodratoff et al. (1994) for instance distinguish between *pattern detection* and *identification of hidden patterns* as separate tasks. In terms of Figure 1.1, the difference between classification and pattern detection is whether the result used is the output of the interpreter (a prediction of the class of a new, unseen case) or a conceptual understanding of the learned model. In other words, the application of the ML technique is a combination of the technical functionality of the technique and the pragmatic use of the outcome in the real world.

The area of machine learning is a sub-field of Artificial Intelligence with close relations to inductive statistics and neural networks (see Figure 1.2). In this thesis we refer to *machine learning techniques* as being techniques that learn classification models and numerical prediction models in a supervised context. Machine learning techniques play a role in data mining, knowledge system construction, adaptive control and robotics.

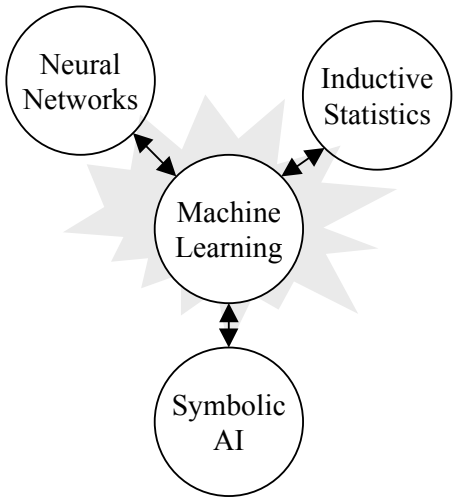


Figure 1.2 Machine learning is closely related to a few other disciplines.

Data mining (DM) is the application of (machine learning) algorithms for extracting patterns from data. One of the often-encountered applications of DM is knowledge discovery in databases (KDD), which is the process of *"... identifying valid, novel, potentially useful, and ultimately understandable patterns in data."* (Fayyad et al., 1996). Apart from mining the data in search for meaningful patterns, KDD includes additional steps, such as incorporating prior knowledge, data collection and interpretation of the results in the application context. Often, this process is "… *driven by goals that are defined according to a problem description representing the needs of an (external) agent."* (Engels & Studer, 1996). Over the last decade, DM and KDD have been very successful areas of application for ML techniques (Adriaans & Zantinge, 1996; Fayyad et al., 1996; Hand et al., 2001).). In DM, the analyses of data that results in a model is essentially the goal of the project as the resulting model supports (managerial or other) decisions. Whether it is used to improve the hit rate of a commercial mailing campaign or an identification model for craters on a photo of an extraterrestrial surface, the generated result is handed over to a (mostly human) actor for further processing. Knowledge system construction on the other hand results in an operational software system. The machine learning component is deployed to acquire a knowledge model that is included in the system. In addition the ML component may also be used to maintain and update the knowledge model. From now on, when talking about ML application this refers to the use of ML techniques for data mining and knowledge system construction.

1.3. Research Questions

The central goal of this work is to develop methodological support for industrial application of machine learning. For such support to be useful, it must correspond with current practices as well as fix some of its shortcomings. Although a number of projects generated valuable experience with the application of ML techniques in an industrial context (e.g. Verdenius, 1991; Verdenius, 1996; Verdenius & Broeze, 1999), these were insufficient as a basis for concrete methodological development. Nevertheless, the project experience was useful in providing the motivation to further develop the requisite methodology. Firstly, the desire to

ground this research with an understanding of the problems of applying machine learning techniques for industrial application leads us to investigate the question:

> *1. What are the experiences and the limitations in industrial application of machine learning techniques?*

Secondly, in order to offer the methodological support to a ML application there has to be a methodological framework. Many of the previously defined process models focused on technical machine learning aspects, thus ignoring the broader connection between a machine learning module and the embedding system. Thus this leads us to investigate further the following question:

> *2. Can a model be formulated that both considers the design issues for machine learning modules as well as the integration of the machine learning module in the embedding system?*

Finally, two methodological questions persistently reappear independent of the application domain and independent of the question whether it is a data mining or knowledge system project as follows:

> *3. Given the required overall functionality of the end result, can we learn or elicit one model to supports the whole functionality? If not, how can we identify a good decomposition of the overall functionality in sub-tasks, and how can we identify sub-tasks that can be realized by means of a machine learning implementation?*

and,

> *4. Given the input data and output data for a specific sub-task, what machine learning technique is most suitable for realizing the defined sub-task?*

1.4. Methodological support

What form does methodological support take? In software and knowledge engineering a long tradition in development methodologies exist (e.g. Sommerville, 1996; Chang, 2002). The development of machine learning applications is in this thesis seen as a sub-discipline of software and knowledge engineering. The support software engineering methods offer concern two major aspects of system development (e.g. Van den Broek, 1981):

- **Activity model**: An activity model defines a series of development activities the outcome of these activities being a working application. In some activity models there is a very explicit phasing. In such a case, the transition between phases is strictly regulated, e.g. by the delivery of specific end products (e.g. System Development Methodology; Turner et al., 1990). In other models, the phasing is less explicit. In these cases, activities concern aspects of the end-product that need to be considered, without prescribing a fixed order (e.g. CommonKADS, Schreiber et al., 2000). The main approaches are the waterfall model, the prototyping model, and the spiral model. In the *waterfall model*, the process of system development is linearly organized in a series of steps that are followed one-by-one. Typically, the consecutive steps include analysis, requirement definition, system design, construction, implementation, evaluation and maintenance. Every phase is concluded with the delivery of a product. In the *prototyping model*, the system development is cyclically organized. Every cycle realizes a next version of a prototype. Based on feedback from experts, the development goals are updated and a new cycle is initiated. The promise of prototyping is to deliver applicable results as early as possible. In the *spiral model*, the short, cyclic planning and control process of the prototyping approach is used. In each cycle one can choose a more traditional analysis-design-construct approach or a

prototyping approach. The activity model to be developed for machine learning application is intended to cover those machine learning aspects that are complementary to existing approaches. As such, it will define andproducts per activity, without offering a strict phasing.

- **Development tools**[2]: A coherent set of technical concepts and tools that support the actual system design, construction and implementation. For different phases of the development process, for different types of software, a large number of tools and formalisms have been developed, examples being ISAC (Lundeberg et al., 1985), NIAM (Nijssen & Halpin, 1989), OMT (Rumbaugh et al., 1991) and UML (Fowler & Scott, 2000).

In knowledge engineering most methods link to one of the existing process approaches in software engineering and focus on providing the technical concepts and tools that are needed to develop knowledge systems (e.g. CommonKADS, Schreiber et al, 2000). In CommonKADS, the spiral approach is deployed. The focus of the approach is on specifying a set of models for all relevant aspects of the knowledge system and on providing tools to support model development and system implementation.

For machine learning applications, the methodological support has focussed on data mining and knowledge discovery in databases. The activity models that have been developed to support DM and KDD (Adriaans & Zantinge, 1996; Fayyad et al., 1996) aim at the analyses of data and the delivery of a model. The activity models are exploration oriented and typically describe the goals and products per phase. Technical toolkits (e.g. Clementine[3]; WEKA, Witten & Frank, 2000) support the activities in the model. However, the support is less for the application of machine learning in knowledge systems. In this scenario, the ML component operates in a complex environment that defines the development goals. In addition the resulting product must operate in collaboration with other components.

1.4.1. Existing Activity Models

Many of the models that have been provided for DM and KDD fall into the category of activity models (Adriaans & Zantinge, 1996; Fayyad et al, 1996; Brodley & Smyth, 1997; Chapman et al., 2000). The models provide a subdivision of an application project together with a definition of the results that have to be delivered. Brodley & Smyth (1997) restrict themselves to the main steps of the process, whereas Chapman et al. (2000) provide a detailed subdivision of the development process, with guidelines for each step guidelines of the results. Most of the approaches however do not provide actual tools or knowledge models in order to derive that result. Adriaans & Zantinge, for instance, provide a comprehensive set of activities of the KDD process (1996, pg. 38). In this list of activities they distinguish a data-mining step that results in a learned model and subsequently they provide an overview of the data mining algorithms. However, support in how to select the algorithms, how to select the models, and how to identify the task to mine for is not provided. Apart from the general-purpose activity models there are also activity models that focus on one type of technology, for example the neural network approach of DTI (1994).

1.4.2. Existing Tools for Designing an ML Application

Activity models support the ML user in *what to do* without necessarily providing support on *how to do it*. There are also tools that support system design. Such approaches give guidelines

---

[2] We consider a tool to be a set of detailed guidelines on how to carry out design, analysis or implementation activities. These guidelines may or may not be implemented in software.

[3] www.spss.com/clementine

and tools for defining system functionality. OMT (Rumbaugh et al, 1991), CommonKADS (Schreiber et al., 2000), MLT (Kodratoff et al, 1994) and UGM (Engels, 1999) are examples of design methods, the latter two in the area of machine learning. And although many of these approaches consider control of the design process, their main orientation is the technical support of the design process (Kroese et al, 1994).

1.4.3. Existing Tools for Technique Selection

In contrast with the design tools, technique selection tools focus on the actual operation of techniques within a given definition of the functionality. Straightforward approaches include the use of (a few) preferred techniques and brute force comparison. In the former case, a small set of techniques is used where new problems are reformulated and transformed in order to achieve the best possible result on the data set at hand. In the latter case, a problem is processed by a large number of techniques. The quality of the results is assessed according to some measure, and consequently the best technique is selected. Both approaches use ML workbenches such as Clementine[4] or WEKA (Witten & Frank, 2000). These tools facilitate the exploration of various data configurations, techniques and technique settings. Performance comparison on the basis of cross validation (Schaffer, 1993) allows for the identification of the best-operating set-ups.

In the literature, the subject of technique selection has generated much debate (e.g. Salzberg, 1995), because how we decided what constitute *best* is relative. For example, what techniques perform 'best' is contextual dependant on whether the focus is on predictive accuracy, the cost-effective classification accuracy, the number of attributes to achieve classification, or the most comprehensive model. An obvious and often applied choice is to focus on predictive classification accuracy. Abundant comparative studies are available, but the results do not give a clear picture on the relation between problem definition and technique (Lim et al, 2000; Kiang, 2002; Michie et al., 1994, MetaL 2004). It is generally accepted in the ML community that cross validation (Schaffer, 1993) gives a reliable impression of the predictive accuracy of a technique on a data set.

Many projects, for instance the MLT (Kodratoff et al. 1994), Statlog (Michie et al., 1994) and MetaL (MetaL, 2004) projects, have tried more principal approaches to support technique selection. MLT heuristically relates a number of factors to preferred techniques. StatLog and MetaL try to generalize from the performance that a group of techniques scores on a number of data sets. One of the problems with these approaches is that the techniques and data sets that are used in these approaches do not give a total overview over the space of possible techniques and data sets (e.g. Kanters & Wets, 1997). Consequently it is unclear how the results of these projects can be generalized to other data sets.

1.5. Thesis Overview

The aim of this work is to support system developers in using learning techniques in their daily practice by offering principled tools that find their motivation in practical problems in the application process. Therefore, the starting point of this study is a survey of application projects using ML techniques (Chapter 2). What problems do the workers experience when applying ML techniques in their daily practice? What are the strengths and weaknesses in such conditions? Chapter 2 identifies two areas where further support is needed:

- Identification of potential tasks for using machine learning during decomposition of complex tasks; and

[4] http://www.spss.com/clementine

- Selection of the most appropriate machine learning technique for a specific problem.

This corresponds to the research questions 3 and 4 of section 1.3.

Chapter 3 and 4 provide practical experience in applying both learning and knowledge based components. Chapter 3 describes the design and implementation of a system for the planning of fruit treatment. The system combines machine learning and knowledge based components. Chapter 4 discusses the design and deployment of modeling approaches in wastewater. In modeling microbiological processes, a knowledge-based approach for compositional modeling is followed. In a control application with little human expertise, an adaptive learning solution is deployed. Both chapters discuss the resulting systems, and point to some of the design problems.

Chapter 5 presents a hierarchical activity model, the MEDIA model. The model distinguishes three levels of design activities, focussing on the application, the acquisition method and the acquisition technique respectively. For each level, design tools are required. At the lowest level the development of this tools is identified as the task for the technique developers. In the next two chapters, approaches and accompanying tools for the two remaining problems are developed.

Chapter 6 integrates the inductive approach in task decomposition. Given the available knowledge and data sources, the best mix to realize the required functionality is designed. In this case, 'best' is defined as the most optimal balance between costs and the quality of the end-result. Chapter 7 deals with a new approach to technique selection. Most existing approaches use a comparative analysis to find the best technique. By selecting techniques on the basis properties of data, a better understanding of the results is achieved. Chapter 8 discusses the contribution of this research.

2. Applications of Inductive Learning Techniques: A Survey in the Netherlands

This chapter has been published as F. Verdenius & M.W. van Someren, Applications of Inductive Learning Techniques: A Survey in the Netherlands, *AI Communications 10, nr. 1*, pp. 3-20 (1997). It is a revised and extended combination of Verdenius (1995) and Verdenius & van Someren (1995).

Abstract

Interest for machine learning techniques has increased over the last decade. In spite of this the application practice of these techniques has never been systematically analysed. This paper analyses the practical application of Inductive Learning Techniques in the Netherlands by means of a survey. Results of this survey are assessed in terms of introduction of (information) technological innovation. The application practice for these techniques finds itself in an initial stage. Current practice is dominated by technical issues and there is little attention for methodological issues associated with analysis of the problem and data collection. In the paper we propose a four-level model for describing the methodological aspects of ILT application. The current practice of application concentrates on the two lowermost layers. Tools for developing applications concentrate mainly on the lowest, technical level.

2.1. Introduction

The proof of a technology is in the usage. Machine Learning (ML) research has delivered a large set of interesting learning techniques. *Inductive Learning Techniques*[5] (ILTs) are a subset that since long is considered to be promising for practical applications. Recently this promise is reinforced by applications of machine learning techniques under the heading of Knowledge Discovery in Databases (KDD) or Data Mining (DM). Applying ILTs for solving practical problems however is not a trivial task. This chapter describes a survey of the current practices of learning technique applications in the Netherlands. The goal of this study is to obtain an overview of practical applications of ILTs, to assess the stage of development of this technology, to assess the success of these applications, and to identify major bottlenecks in ILT application.

This chapter is organized as follows. Section 2.2 presents the field of interest, Inductive Learning Techniques. Section 2.3 presents a model of the ILT application process. Section 2.4 briefly reviews the literature on ILT applications. Section 2.5 presents and discusses the results of the survey, focusing on a description of the state of the art, the degree of success and problems of the application projects, and limitations in ILT application. Section 2.6 concludes. In an addendum in Section 2.7, an update of the survey with data from the online KDDNuggets forum is given.

2.2. Inductive Learning Techniques

What are *Inductive Learning Techniques?* The notion of Inductive Learning is not uniquely defined in literature (a good overview from an application viewpoint is given by Weiss &

[5] The term *inductive learning techniques* is a heritage from early work. It is used as a synonym for the term *machine learning techniques* in the other chapters.

Kulikowski (1991)). In our survey we use the term Inductive Learning Techniques for those techniques that extract models as generalisations from input data. Some important classes of ILTs are:

1) **Induction of Decision Trees or Rules** (RI)

   An overview of the relevant techniques can be found in Bratko et al (1996). Frequently used examples are CN2 (Clark & Niblett, 1989) for generating rules from data, and C4.5 (Quinlan, 1993) for generating decision trees and extracting decision rules from data.

2) **Neural Network learning algorithms** (NN)

   An overview of neural network techniques is given in Simpson (1990). Examples are error back propagation for multi-layered perceptrons (Rumelhart et al., 1986), and self-organising maps (Kohonen, 1984). These techniques represent models in the form of networks of processing units, interconnected by weighed links. These models can be executed directly.

3) **Case Based Reasoning** (CBR),

   An overview is given by Aamodt & Plaza (1994). By CBR we mean both instance and exemplar based reasoning (Aha et al., 1991; Kibler & Aha, 1987) and case based reasoning (Kolodner, 1993). Note that for these techniques the model consists in part of the data that is the input for the learning process.

4) **Genetic and Evolutionary Algorithms** (GA)

   A review of GAs can be found in Giordana & Neri (1996), and an introduction is given in Goldberg (1989). GAs are general purpose optimisation algorithms that can be used to construct models according to criteria like simplicity and explanatory power. Models may use any representation.

5) **Inductive Statistics** (Stat)

   There are many handbooks for statistical techniques. Examples of such techniques are bayesian classifiers, statistical regression, and nearest neighbour techniques. A brief introduction is found in Weiss & Kulikowski (1991).

Although these techniques all perform a kind of inductive inference on data, they may serve several different purposes. An important distinction is that between *data analysis* and *inductive programming*[6]. In the case of data analysis, the primary goal is to construct a model that describes and explains a set of data (usually observations). ILTs open up the possibility to generate knowledge in domains where detailed domain knowledge is lacking but where data can easily be collected. The goal of inductive programming applications is the design and construction of computer systems that can automatically apply generated knowledge. In other words, if used for inductive programming, the ILT is used as a knowledge acquisition tool. The induction result takes the form of a *runnable* model (e.g. decision trees/rules, multi-layer perceptron and others). These runnable models can be constructed in one pass through a data set or induction can continue during application of the model in an *adaptive computer system*. In *inductive programming* applications, ILTs reduce the laborious and expensive phases of knowledge elicitation and knowledge maintenance for tasks for which real-world data become available. In CBR, knowledge is not extracted from a set of examples

[6] This concept was refered to as *knowledge acquisition* in Verdenius (1995) and Verdenius & van Someren (1995)

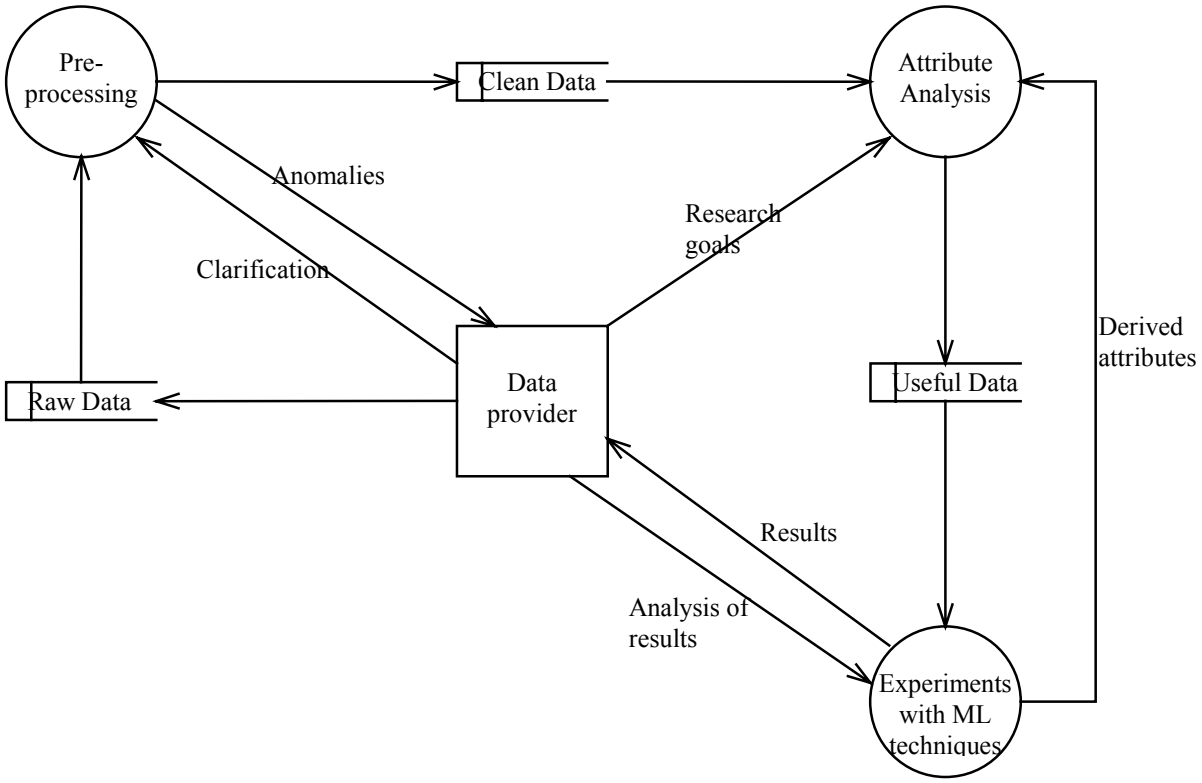


Figure 2.1. WEKA Process Model for Machine Learning Application, according to Garner et al. (1995).

to be stored as an explicitly learned model. Instead knowledge is stored implicitly in the form of a set of (selected, indexed) cases. Only at performance time, when a previously unseen case has to be processed, similar cases are examined, and a classification or prediction is extracted on the basis of similarity and possibly background knowledge. This inference is (partially) inductive (Michalski, 1994). As a result it is commonly considered to fall outside the group of ILTs. At performance time however, the functionality is not different from that of, for instance, a decision tree classifier: in both cases knowledge structures extracted from input data are used to infer an unknown feature of a new entity.

Traditionally within Machine Learning inductive learning comprises techniques that generate symbolic models from data (Holland et al., 1986). Our definition is substantially broader. Moreover, it is more application oriented, and emphasises the strong functional equivalence that exists between these techniques in their practical application: different ILTs according to this definition play similar roles in applications. In a survey it is obviously not possible to include all applications of inductive statistics. Therefore we only incorporate projects that include at least one of the ML based ILTs and report on the use of inductive statistics only in that context.

2.3. ILT Applications

Having defined the techniques of interest, the next step is to define when a project is considered to be an ILT application. To understand when and how ILTs are applied we must take a broader view than strict application of an ILT to a data set. Applying an ILT is done to solve some problem. We define an ILT application as follows:

> *An **ILT application** is a project where an Inductive Learning Technique is used to solve a problem, either owned by one involved party or being a*

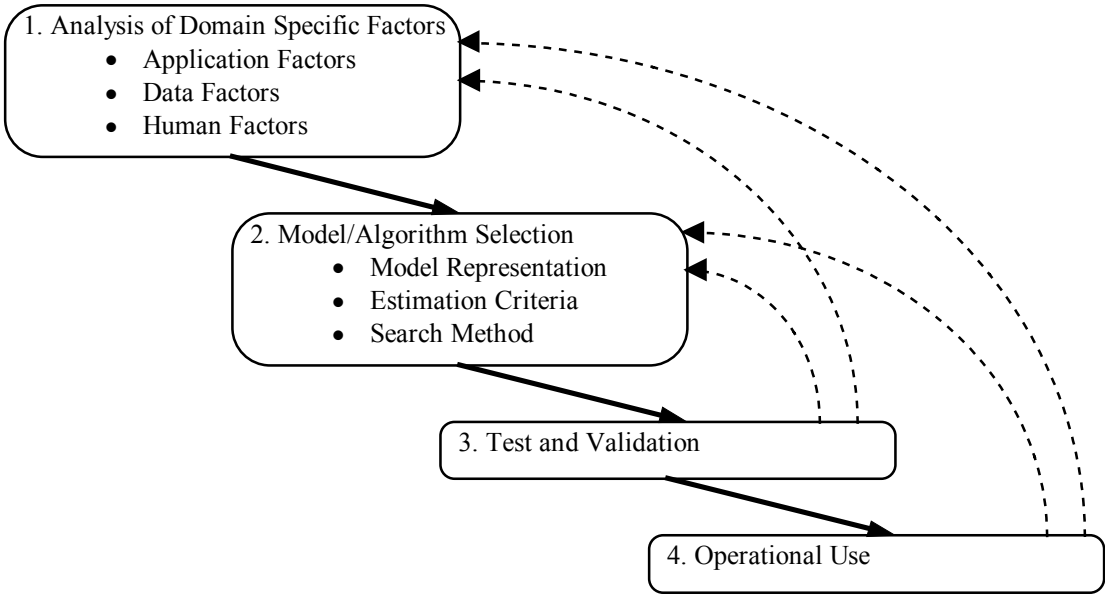


Figure 2.2 The application development process for machine learning techniques according to Brodley & Smyth (1995).

> *common, well known problem. The result of the project is either a system that (in principal) can be used to support the problem owner in solving the problem, or a body of knowledge that enables the problem owner himself to solve the problem.*

The first important aspect of ILT application is the decision/development path that has been followed to realise the application. Several methods exist that prescribe the development of machine learning applications in process models with intermediate decisions and milestones. We discuss three different process models and extract general method features to evaluate our survey results.

The first process model has been formulated in the framework of WEKA (Garner et al., 1995), a workbench that is intended to acquire knowledge for classification tasks from large databases. The intended application domain is agriculture; the intended project purpose is data analysis. The WEKA approach assumes that experts in this domain know very little about data analysis or machine learning. Consequently, ILT experts play an important role in applying WEKA. Domain experts are intensively consulted, but WEKA is not primarily an end-user tool. Garner et al. (1995) describe a process model for applying the system (Figure 2.1). This process model concentrates on *pre-processing* of data, *attribute analysis* and *experiments with ML techniques*, all being tasks where ILT experts heavily interact with domain experts. Data pre-processing guarantees the presence of a complete and usable data set. This includes data extraction from structured databases into flat table format and anomaly repair, as well as transformation of data to suit technical requirements. Attribute analysis aims at minimising the size of the attribute set, under conservation of the interesting information. Experiments with ML techniques consist of processing the data set with available techniques (by means of brute force experimentation), experiment evaluation and post-processing of results.

The WEKA process focuses on those process steps that directly relate to the operation of the ILT techniques. WEKA directs towards data-analysis and does not address possible fielding of ILT application results. Consequently, the WEKA process model does not contain ILT application definition and design or data definition activities that analyze user requirements.

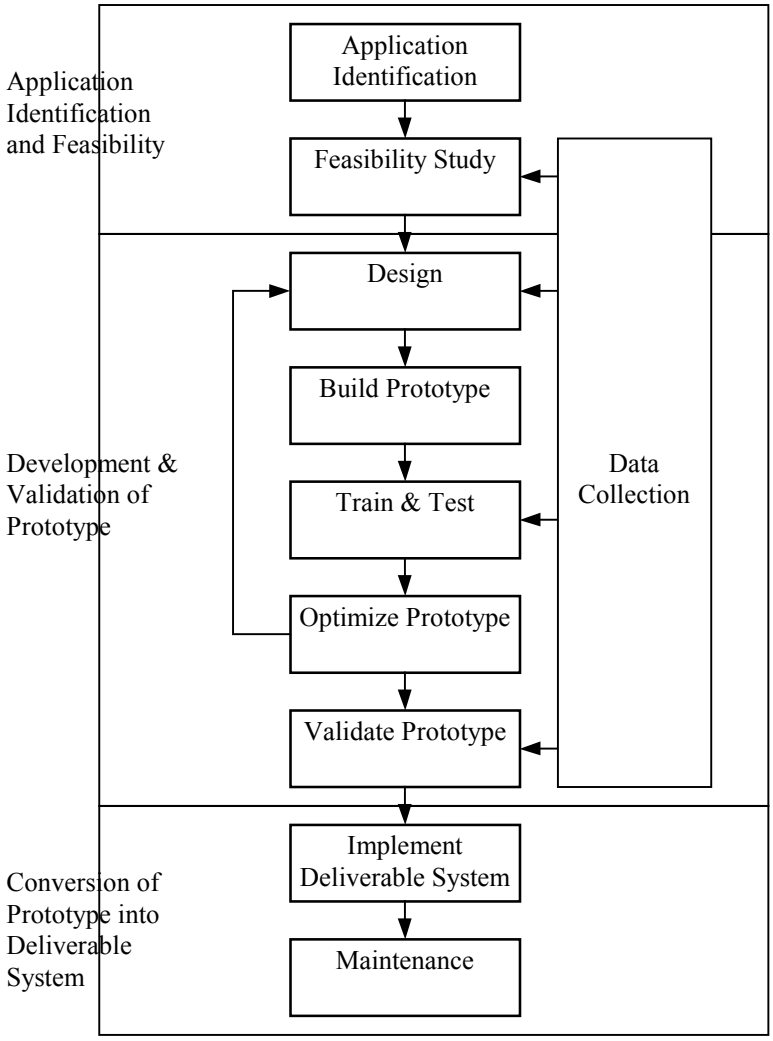


Figure 2.3 Process Model for NN application development according to DTI (1994).

Furthermore the model does not concern any aspect relating to the management of the project. Finally it is striking that WEKA relies on brute force experimentation for identifying the optimal ILT (configuration).

The other process models aim for inductive programming projects. Brodley and Smyth (1995) present a four-step approach that aims at developing a fielded ILT application for classification tasks (Figure 2.2). The approach they present consists of four phases: (1) analysis of domain specific factors, (2) model/algorithm selection, (3) test and validation, and (4) operational use. The first phase analyses factors arising from the environment of the application. The second phase uses the outcome of this analysis to select a learning algorithm. The third phase focuses on the development of the ILT model. When completed, the model can be used.

The main emphasis of this approach lies in the first two phases. The domain factors to be analyzed are extensively described, and the aspects of models/algorithms to consider are moulded in three components: (1) model representation, (2) estimation criteria, and (3) search method. How the domain factor analysis is used in algorithm selection is partially described, but not fully formalized. Under test and validation of the model reside the processes of deriving the classification model and of refining it for improvement on the basis of the

quality of the model. Moreover, iteration based on feedback from operational use is also included here. This phase is only briefly described.

The approach primarily aims at stand-alone usage of the resulting system/model. Integration with other system components is not foreseen. For optimal results, model selection requires detailed expertise of the available techniques. Detailed guidance in taking design and development decisions is not presented. Summarized, Brodley & Smyth present an approach that elaborates the process of requirement definition and model selection. They pay less attention to data preparation, the actual usage of the algorithm, and the structure that is necessary to manage learning projects.

This last point is one of the main focuses in yet another approach as presented by the UK Department of Trade and Industry (DTI, 1994; see Figure 2.3. for the proposed project life cycle). DTI presents a detailed approach for developing NN applications, which covers all relevant levels of project realization. On the project management level a combination of the waterfall approach and the Boehms spiral model for system development is used (Sommerville, 1996). At this level it is tentatively described which (type of) decisions have to be derived in each phase and which type of documents/products should be delivered. A very important aspect at this level is to gain and keep the confidence of clients that the use of neural technology is economically justified. On application design level, the approach specifies the type of analysis to perform and design decisions to take, e.g. discrimination of functionalities that can be encountered. On the technique level, the method is focused towards neural networks, providing heuristics on technique capabilities, and selection criteria. In general, the formal aspect of DTI is not very well developed. The main focus is to provide an overview of the development process in addition to monitoring and control possibilities for the management of the application development.

Therefore we can conclude that there is no uniform project approach. These three approaches do however implicitly define the process of ILT application on three levels and one control component:

1) *Application level*: This consists of analyzing a problem encountered in the real world, defining the scope of the solution, identifying resources (such as data, or sources of data, human experts, existing knowledge systems) for solving the problem, and constructing a conceptual model of the problem, often in the form of a decomposition of a problem into sub-problems. ILTs are a class of techniques that can be used to solve an entire problem or a sub-problem. Other possibilities are knowledge elicitation from human experts, formalized knowledge from textbooks, etc. A plan is made for solving the problem. This can include also final stages in the project such as fielding the result of ILT application. The results of this level are requirements of the target knowledge and descriptions of the resources that can be used for acquiring it. An ILT application can be a data analysis or an inductive programming problem. This difference will lead to different requirements, resources, and plans.

2) *Analysis level*: Once the requirements are defined, existing knowledge and data are acquired and an appropriate ILT must be selected. If a data set was given at the start of the project, data must be selected and transformed to allow the ILT to be applied. Additionally, data pre-processing is an important issue. Variables must be re-coded or deleted in order to reduce the size of the data space. Errors in the data must be detected and repaired. Finally, this is the level where the learning techniques must be selected. The bias of the selected techniques need to fit the problem and the characteristics of the data set. After all, a technique that prefers simple generalizations to complex generalizations

may miss complex hypotheses. And some techniques favor linear relations when non-linear relations may be preferable for certain variables.

3) *Technique level*: This concerns the operation of a specific ILT. The operation of an ILT concerns all aspects that are related to a specific technique. Once one or more techniques have been selected, there are additional choices about parameters of the systems and their application (e.g. cross validation methods).

4) *Project Management*: This concerns the control of the other three levels in relation to financial and operational constraints that are imposed on the project. Given requirements produce a sequence of actions, decisions, resource allocations on all three levels that ensure realization of project goals for maximum user/client satisfaction.

This structure can describe any ILT application project and is further elaborated in chapter 5. A particular ILT project may not have a pure data analysis or inductive programming goal. Some other goals with different project structures are:

- *Proof of concept*: Primary goal is to prove the practical feasibility of an ILT application without primary worries about the practical feasibility of the tool. All three development levels may be included, but focus is on all technical activities within the levels.
- *Technique comparison*: primary goal is to compare performances of several ILTs on one or several tasks in order to solve a specific problem. Main focus is on technique operation and the technique related activities within the ILT development level.

Earlier we indicated our research interest for this survey. The question that we want to answer with this survey concerns ILT application projects. Specifically our research questions on practical ILT applications are as follows::

1) What are the characteristics of ILT projects such as duration and number of people? Who applies which technique and for what kind of application? Although ILT research has a long history, ILT is new technology from the application perspective. Therefore, we consider it from the perspective of information technology applications.

2) Which ILT techniques are used in applications? How successful are the applications and can we identify key factors for success?

3) Can we identify problems and limitations that hinder applying ILTs? Can we relate them to application aspects? Can we identify research topics to facilitate future ILT applications?

2.4. ILT Applications in the Literature

In the literature, many reports on ILT applications can be found. Reports on ILT applications in literature fall in several categories: description of individual projects (e.g. Evans & Fischer, 1994; Simoudis et al, 1995; Timmermans & Hulzebosch, 1996; Verdenius, 1991; Giordana et al., 1993), overviews of projects (Kodratoff & Moustakis 1994; Langley, 1993; Langley & Simon, 1996; Rudström, 1995) and application tool descriptions (MLT: Kodratoff et al., 1994; Sleeman, 1994; WEKA: Garner et al., 1995; Clementine: Shearer & Khabaza, 1995). Kodratoff et al (1994) give a classification of ILT techniques by their function along various dimensions of learning problems, such as the skills of the ML user, the application goals and the learning task.

What can we learn from these reports on the above posed questions? Most publications are triggered by positive project results thereby indicating that it is possible to use ILTs to solve

complex problems. Many of the reports in the literature however stem from research oriented projects. This results in a specific bias on an overview extracted only from literature references. Consequently the state of the art in ILT application (i.e. an answer to question 1) cannot be obtained from literature. Moreover, data that can provide the answer to the second question will also be interpreted with a positive bias. Thus, the existing literature seems a poor basis for answering the questions. A survey among real world practitioners seems the only way to obtain more detailed insight. Such a survey has, until now, neer been initiated.

Technique selection is handled in different ways in ILT application projects. Mostly, the whole problem of technique selection is ignored, especially in case descriptions. Apparently, in many projects an a priori selection of techniques has been made. And although the motivations for these projects are hardly ever explicit, plausible motivations include: technique driven projects, (lack of) availability of knowledge on alternative techniques and (lack of) availability of tools. Technique selection is a major issue for the workbenches. In the MLT project a special tool is constructed to guide the user towards a set of techniques that is most suited for solving his problem. The knowledge used in this system is concentrated around technical considerations. However, an outcome of the StatLog project (Michie et al., 1994) is that it is difficult to foresee, purely on technical grounds, which technique is most preferable for a specific problem. WEKA comes with a fundamentally different approach to technique selection in the form of brute force induction. In the approach presented by Brodley & Smyth (1995) pragmatic arguments are presented, such as availability of tools, availability of knowledge, and availability of application requirements. We use the survey to get a better understanding of how practitioners select techniques and how they use the various inconsistent guidelines and approaches that are found in the literature.

2.5. A Survey of ILT Applications

Until now, no large-scale study has assessed the questions of Section 2.3 for real-world application projects. Therefore we have initiated a survey for applications of ILTs. For reasons of practical feasibility, we have restricted ourselves to assessing the situation in the Netherlands. The set of addressees for our survey include commercial companies, research institutes and universities in the Netherlands. The initial address base was collected on the basis of foreknowledge on the current practice of ILT application. Additional addresses were gathered on the basis of received responses. To further increase the number of surveys, we placed a call for ILT applications in the magazine of the Dutch association for AI (NVKI). Lastly a user association for NN, the VANN, have made their database of members available for sending the survey.

The questionnaire we have applied consists of three sections. Section 1 focuses on general company data. Section 2 collects superficial data for specific projects. Section 3 collects detailed project information. Each respondent could provide information on a maximum of 3 different projects. Section 3 is not completed for several reported projects by their respondents. Between end of May and beginning of September 1994, 177 forms were sent off. Out of 59 returned forms, 56 contained useful data. After data entry and initial pre-processing of the data, analysis took place on the basis of cross tabulation. Statistical significance is measured on the basis of Fisher's exact test, Likelihood ratio and Mantel Hearzel. In this text totaled results are presented.

2.5.1. Nature of ILT applications

All non-technical characteristics of the projects were considered within the classification *nature of application*. The results of this area are subdivided into the following sub-fields:

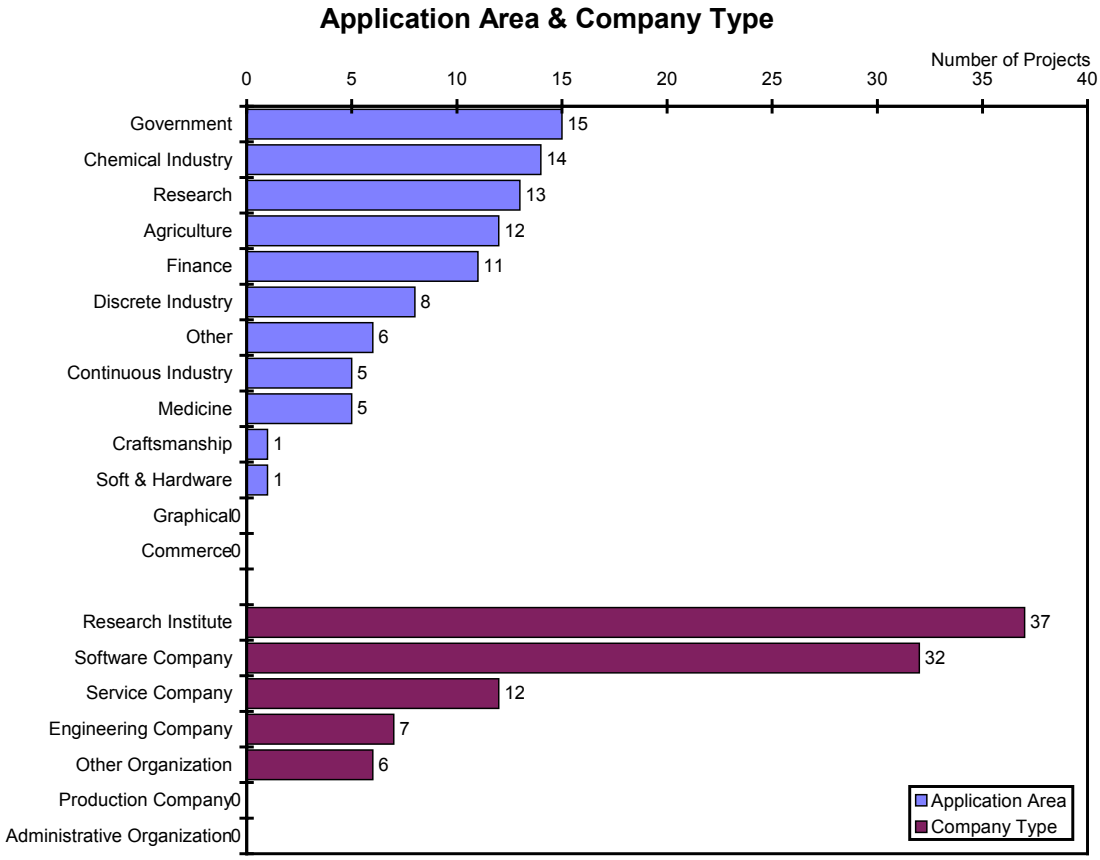


Figure 2.4 Application area and company type applying ILT (per project)

application area, application domain, *company type*, *project size in throughput* time, *project size in labour time*, *project focus*, *importance of ILT* and miscellaneous application aspects.

Figure 2.4 shows an overview of ILT application areas and company types. The major application areas include *government, industry, research, agriculture*, *finance, medicine* plus *hardware and software companies* among others. The types of companiers that constitute the primary users of OLT applications classify themselves as (hard- &) software company or as research organization. It is notable that several specialized companies in the field of ILTs consider themselves to be research organizations despite their independent commercial status.

Figure 2.5 presents an overview of the project size. The vast majority of the projects are scheduled for a period shorter than 4 years. In fact, almost half of the projects are scheduled for less than one year. This is further evident when focusing on projects that had already been realized. In that case the ratio between projects < 1 year and > 1 year is even more skewed. These results do not correlate with the technique that is used. The fact that the majority of the ILT projects had a research focus is clearly illustrated by Figure 2.6. The importance of the ILT-part to the project is expressed in Figure 2.7. In ILT application projects most time is spent on the ILT system application. Slight differences occur per technique; Genetic algorithms (GA) show high percentages over all, while neural networks for financial applications showed a relatively small share of the ILT share.

We have extracted from the textual responses several application aspects, i.e. *task*, *function* and *domain* of the ILTs (Figure 2.8). *Classification* is the dominant task where ILTs are used. However, the most important functional domain is application of ILTs in industrial process

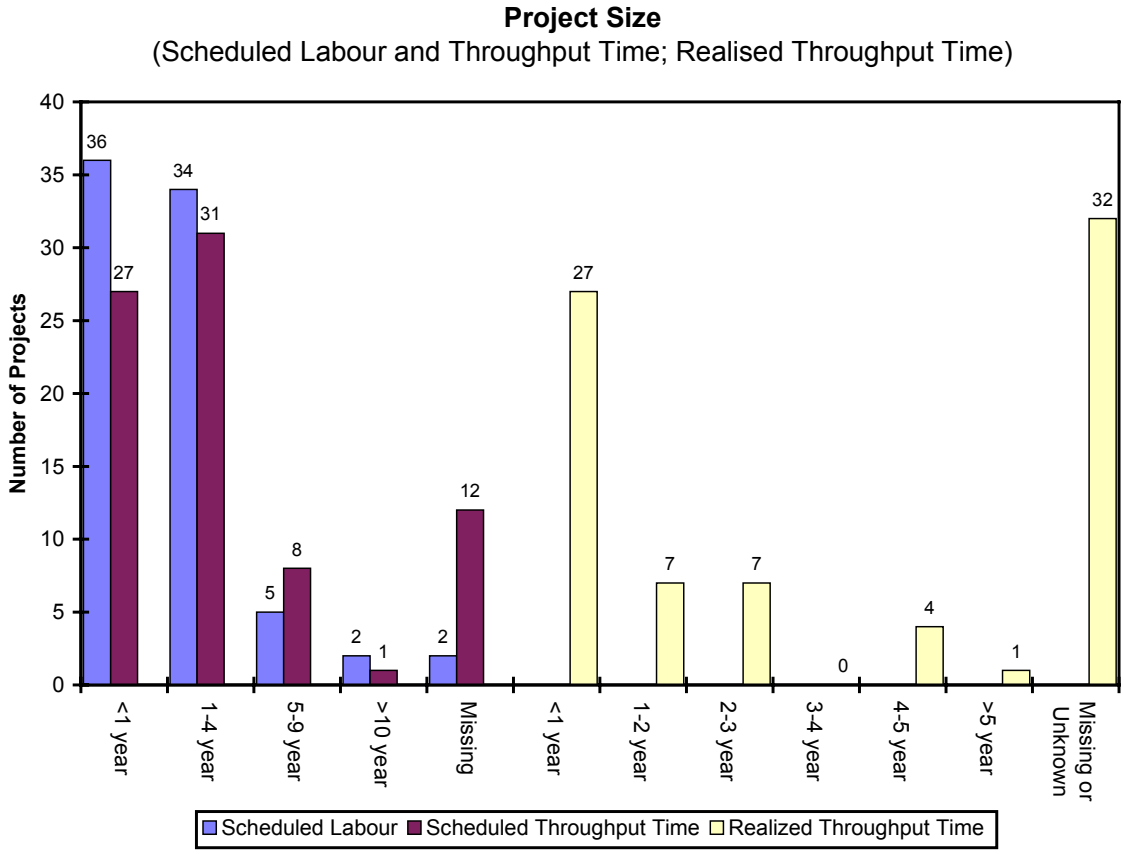


Figure 2.5 Project size. The left side depicts the scheduled size (labor + throughput time). The right part shows the throughput time for projects that have already been realized.

environments. For these applications, process modeling is of major importance. Another important functional domain is found in database related applications, where in marketing databases particularly play a significant role.

A major differentiation (for the functional domain) is between *product oriented* and *scientific/methodological* projects. In addition, some applications are not exclusively directed with learning as a goal or an end, but involve a comparison between different techniques or tools for other purposes than mere application oriented technique selection. Within these particular applications, there appears to be a category of *proof-of-principle* projects where a technique is experimentally evaluated. From the survey a typology of projects emerges:

Data analysis (DA): In these projects, given a specific analysis question, a pre-defined data set is given and a technique and tool is selected. The technique and tool is further fine-tuned to derive the best data analyses. We call such a project *product oriented*.

Inductive programming (IP): In these projects the goal is to construct a software system to perform a certain task. Domain analysis, data collection and technique and tool selection and technique tuning are all part of the project. Such projects are also considered to be *product oriented.*

Technique oriented (TO): Projects in this category concentrate on exploring one or more techniques. Data-set and task may also be given. Such projects were classified as *scientific/methodological*, where

Usage Type	Freq.
Data Modelling	25
Inductive programming	41
Unknown	13

Table 2.1. Frequencies of purposes for applying ILTs.

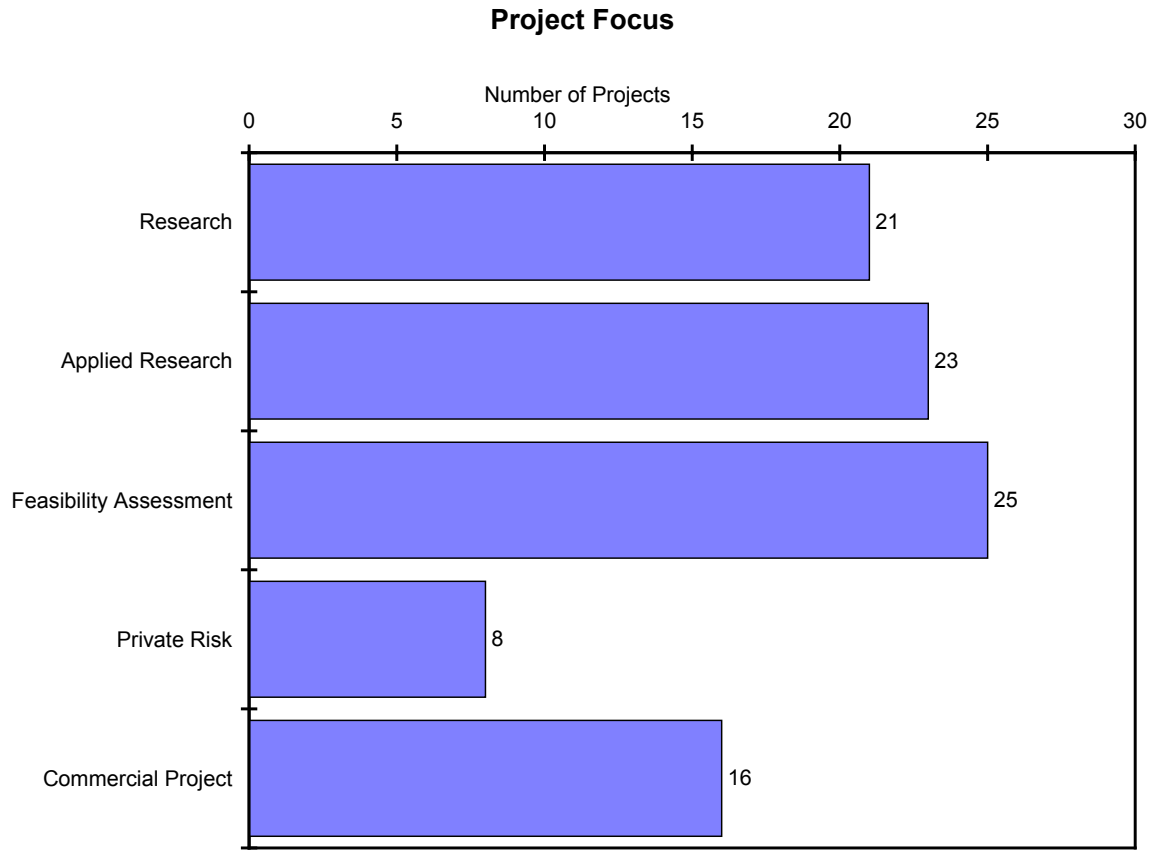


Figure 2.6 Application focus of the project.

exploration, development, and build-up of experience are important motivators.

Not all responses can be classified in one of these three groups due to incomprehensible information. Moreover a few respondents give information on the domain functions realized by applying ILTs, without describing how they are realized.

Another important aspect of ILT applications is the purpose of application, application area and analytical task. The differentiation between data analysis and inductive programming as the purpose, as reported by respondents, is presented in Table 2.1[7].

Finally, it is interesting to see what roles the respondents play in the project. Especially the ratio between internal roles (developer for own risk or internal client) versus external roles (client, developer for external client) is informative, because it gives an impression of how far the application of ML techniques have penetrated into industrial practice. From Table 2.2 it can be concluded that purely commercial projects (i.e. projects where external clients are

[7] The high number for Inductive programming (IP) might be inaccurate due to definition problems. Within TP, we have further distinguished four IP subtypes: one-time TP, initially and manual update, initial with automatic update, permanent. For many respondents that report application of ILT for one-time IP or IP with manual update, it seems the case that they really apply Data analysis, in the sense that the output of the ILT is not used for performing specific reasoning tasks in a (human or computer) KBS. The frequencies for one-time, initial manual, initial + automatic and permanent IP was: 8, 9, 9, 15.

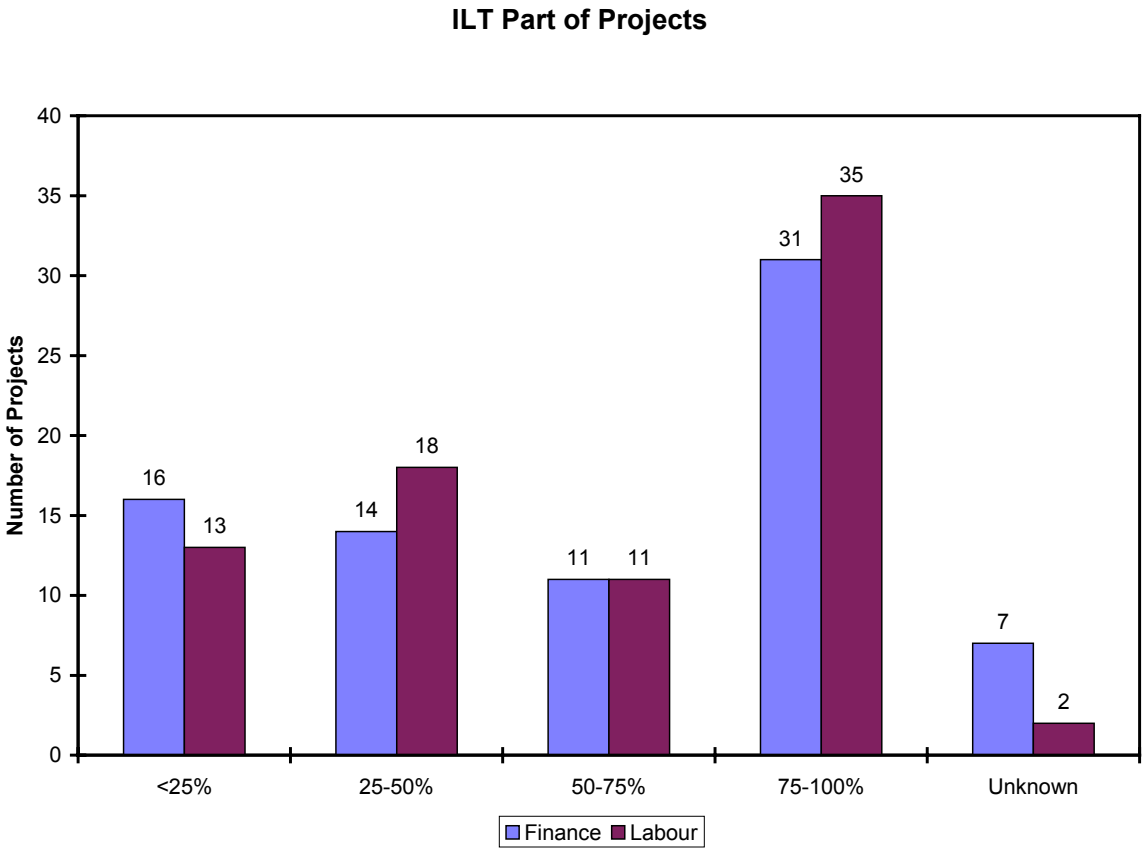


Figure 2.7 ILT part of the project (Finance and Labor).

involved) play an important role as more than 50% of the projects concern developments for commercial clients.

2.5.2. Which techniques were used on what data?

If we focus on the applied ILTs (Figure 2.9), it is evident that NN are the most frequently utilized. RI and GA are equally popular and are the second most employed ILTs. It was noticed both for NN and for RI that one specific technique variant dominates (NN: error back propagation: Rumelhart et al., 1986; RI: C4.5/ID3: Quinlan, 1986 & 1993 respectively). Users of inductive statistics on the other hand indicate names of various specific techniques that are used in their projects. For GA and CBR, respondents hardly give any further specification of the applied variant of the technique, though some tools are mentioned in the case of GA. Probably, the older tradition of inductive statistics, with a longer history of development and application, plays a role. All reported applications were purely inductive "batch mode" applications and did not involve interaction with domain experts or the use of expert knowledge in addition to the data.

Role	Freq.
Client	3
Developer for own risk	19
Developer for internal client	7
Developer for external client	41
Other	16

Table 2.2. Roles in projects (more than one answer per project).

Table 2.3 shows the relationship between technique and characteristics of the data set. Neural nets and statistical techniques are used for larger data sets than symbolic techniques. It is

Technique	#Records		#Attributes		#Binary Attributes	
	Used	Not Used	Used	Not Used	Used	Not Used
NN	9645	2371	605	23	202	47
RI	5207	7396	173	483	23	131
GA	7798	6715	839	303	1	130
Stat	21866	3486	275	431	406	35
	#Categorial Attributes		%Unknown		#Classes	
	Used	Not Used	Used	Not Used	Used	Not Used
NN	14	17	10	8	7	7
RI	24	12	10	7	8	6
GA	25	19	0	10	2	8
Stat	8	17	10	9	7	7

Table 2.3. Relation between technique and several aspects of the applied data set

difficult to explain this choice since respondents rarely clarify their motivation for the chosen technique.

2.5.3. Success of ILT applications

A first indication for success is the number of (pre-)operational systems. Of the 79 projects, 25 projects delivered results that became operational, either as an independent system, or in the form of a knowledge model, and 11 projects have their results in a pre-operational phase. When systems become operational, the user/client satisfaction with the practical performance of the system is a major indicator for success. The vast majority of respondents indicate to be satisfaction with the accomplished ILT result. Very few projects (5) report unsatisfactory results with an ILT and no project reports to have a very unsatisfying result. On the other hand 16 project report very satisfying results and 22 projects report the results as satisfying (see Figure 2.10). However, for 36 projects this question is not answered because the ILT application is not yet in operation. Mostly this is due to ongoing work at the time of the survey. Of this last category 6 projects are cancelled. Main reasons for canceling projects reside outside the ILTs (mainly difficulties in getting the proper data) or in the ILT results (insufficient prediction/classification results). This illustrates the importance of data pre-processing, such as attribute selection, as well as getting good clues for project risks as a result of technique capabilities. This can be found as separate issues in the process models discussed in Section 2.3.

We hoped to gain insights into the successfulness of the projects, and to be able to relate this to critical factors for project application. The profile of responses we have received makes it impossible to obtain clear insight as there is little variation in answers. There may be several causes for this lack of variation in responses. First, there may be a *commercial* or *public relations* bias within companies. Such a bias might, in spite of the earlier signaled openness, prevent to admission of disappointing experiences. A second reason may be that many applications are not completed so that successfulness is not yet assessable. Projects that are likely not to succeed are not completed (e.g. if performance is low). By applying other techniques to resolve a problem, the resulting application may not be considered an ILT application any more. Therefore, we are not able to analyze success factors directly due to a deficiency in response variation.

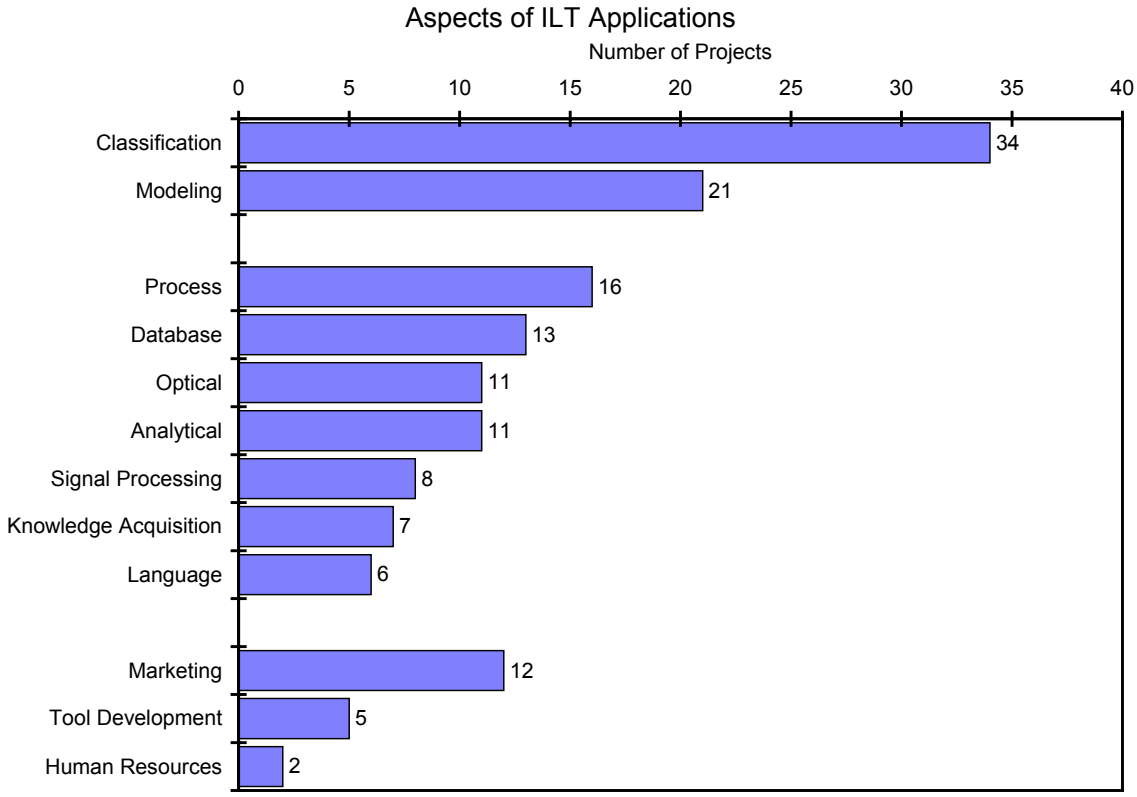


Figure 2.8 Aspects of ILT applications. The first group of items refers to tasks as performed by the ILT, the second group describes functional aspects and the last group details domain areas indicated in Figure 2.4

2.5.4. Problems and Limitations of ILT Application

To assess the state of application of ILTs, we consider the introduction of ILTs in practical applications to be a process of technical innovation. Models for assessing the state of innovation have been proposed, both for information technology (Nolan, 1979) and for new technologies in general (Bemelmans, 1994). Both authors assume that the introduction of a technology develops phase-wise, where phases can be recognized on typical application characteristics. Nolan decomposes companies into 4 aspects (*Systems, Resources, Management* and *Users*). The characteristics of these aspects develop according to 6 phases (*Initiation, Contagion, Control, Integration, Data Administration* and *Maturity*). The model is descriptive; it describes often-encountered practice. The model is also normative: it states that companies ought to experience all phases to optimally cope with information technology. Bemelmans' model is more general and also less normative. It describes 4 phases (*Initiation, Diffusion, Consolidation* and *Integration*) that a company cyclically goes through in coping with new technology. Both models give lists of characteristics per state that can be verified in practice, thus enabling the assessment of the companies' phases.

If we look at the technical-, project- and organizational characteristics of the applications we see that projects are relatively small, in many cases stand-alone data analysis applications. Several respondents indicate that *getting acquainted with learning technology* is a major reason for performing projects. Moreover, organization of the projects is signaled to be difficult. Guidelines on how to efficiently organize ILT projects are hardly encountered in the

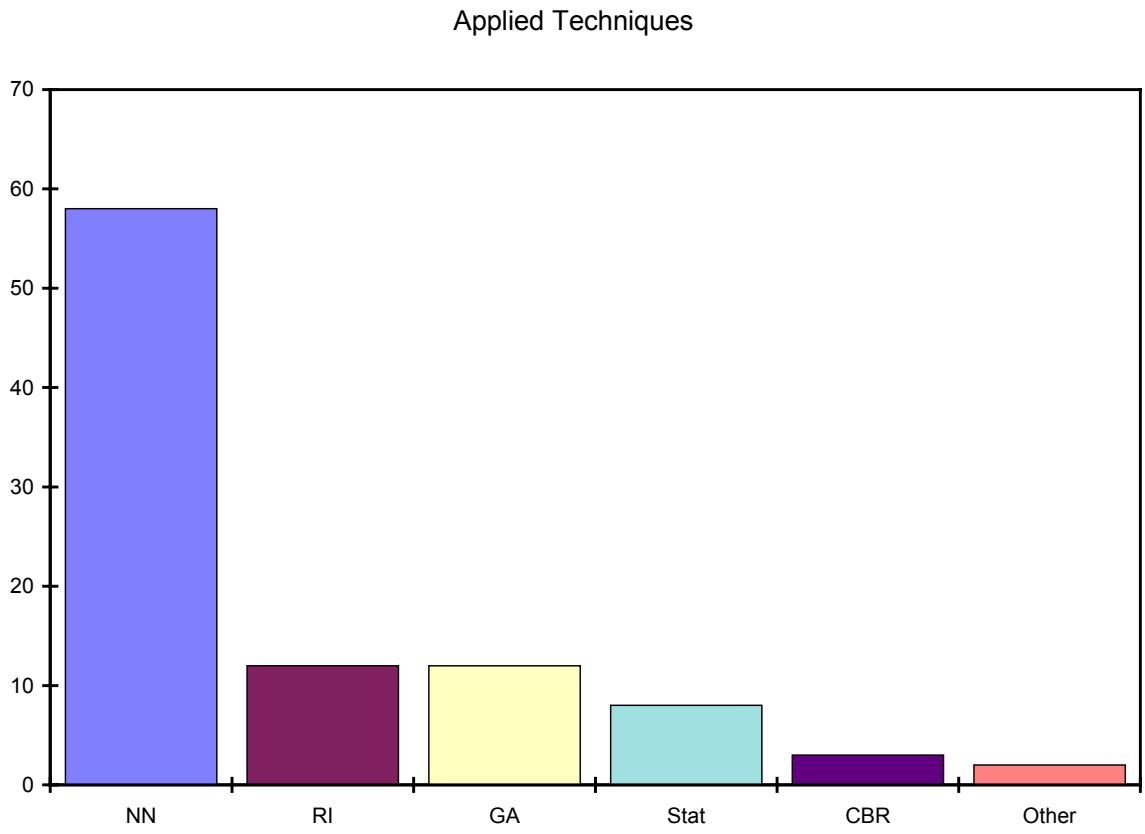


Figure 2.9 Overview of the applied ILT (NN = Neural Networks, RI = Symbolic Rule Induction, GA = Genetic Algorithm, CBR = Case Based Reasoning, Stat = Inductive statistics, Other = Other Techniques)

professional literature. All these points coincide with characteristics that both Nolan and Bemelmans attach to the Initiation phase. Also, most activities concentrate on the Technique operation level and the ILT development level.

The proliferation of ILTs to several areas within companies (administrative, process, marketing) on first glance seems a sign of diffusion (Bemelmans) and contagion (Nolan), but in this case appearances may be deceptive. Application in these areas is a result of the non-administrative origin of the technique, and not of proliferation across domains.

Positioning ILT practice in the initial phase is consistent with the fact that respondents report a lack of a clear method for ILT application, i.e. a set of clearly defined activities to design and implement a working ILT application for a given problem. Together with the usage of ILT tools (software tools that support the design and development of applications for a specific ILTs), we call this the formalization of ILT application. Figure 2.11 presents an over-view of method and tool usage. It shows that of all projects using ILTs, 32% used no specific method at all, while 56% applied a company specific method. Moreover, the majority of these company specific methods is evolutionary in nature. Combined with information based on the development steps there is reason to assume that the evolutionary exploration mostly boils down to a train-and-test approach (e.g. Michie et al., 1994, pg. 108).

The other aspect of formalization is tool usage, indicated in the same figure. ILT tools are applied in 36% of the projects. The other 64% use tools for the learning part that were constructed as part of the project. Moreover, usage of other than ILT tools can also be viewed

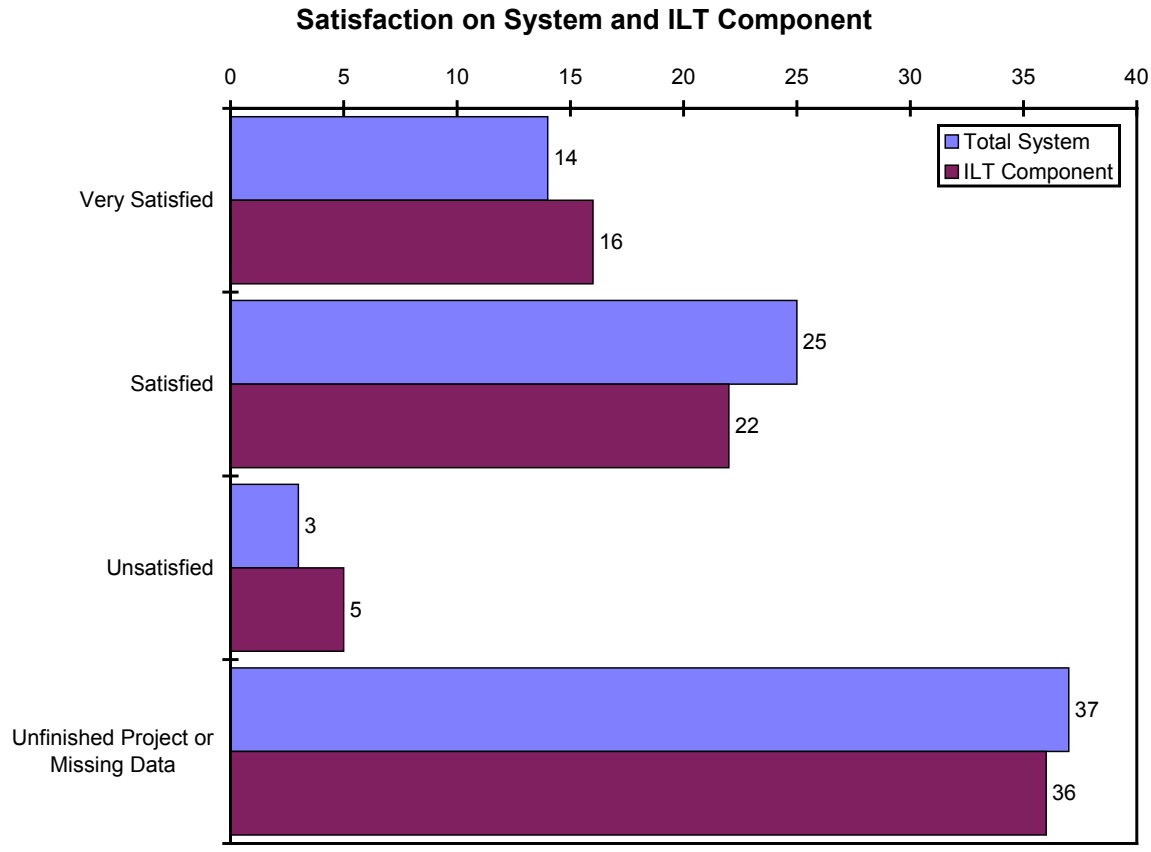


Figure 2.10 Satisfaction on system and ILT component as perceived by respondent.

in Figure 2.11. A related point is the low usage of tools, especially ILT tools[8]. It is remarkable to notice that almost half of the respondents use no ILT tools. As many applications utilize standard techniques that are available *off the shelf* (both on commercial and non-commercial basis), such as C4.5 (Quinlan, 1993) and back propagation neural networks (Rumelhart et al., 1986), the usage of standard tools seems obvious. For these techniques there are a large range of tools available for reasonable prices. In spite thereof, people prefer to develop the code in the project. A remarkable pattern here is that neural network users apply more tools than normally expected, while a majority of GA users apply no tools at all. For techniques such as GAs or CBR there may be less tools available to a broad audience. Tools that are available through the Internet are oriented more to the research community and apparently not to practitioners.

A reason for the *self-coding* behaviour may originate in a *technical* orientation of projects and their performers. People may be fascinated by a particular technique and probably unaware of available tools. The same factor may also explain the large number of respondents implying *technique fixed in advance* or *technique is suitable* justifications for the applied ILTs. Moreover, self-coding may serve an educational purpose (learning by doing).

[8] As we have seen in earlier sections, the projects are mostly limited to ILT implementation, which indicates that the tools to support design and implementation of large systems are not likely to be of use.

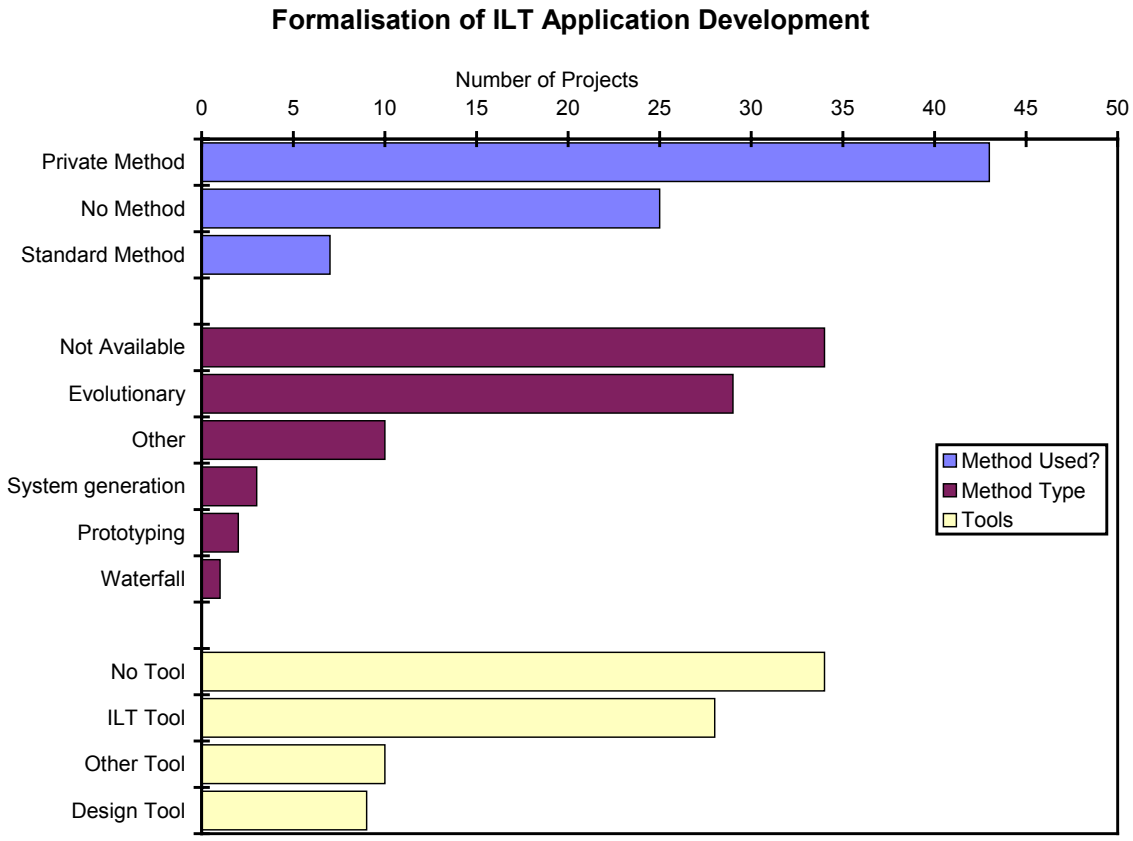


Figure 2.11 The degree of formalization during ILT application development.

Two specific subparts of the ILT application method are the justification arguments of ILT selection/rejection and the consideration of alternative techniques in selecting the ILTs. The first item, the justification arguments of ILT selection or rejection, is considered by asking for the reasons why the applied techniques are chosen. An overview of the grouped answers (grouping performed by the authors) can be found in Figure 2.12. The justification that was used most often is *suitability* of a technique for the task. This "justification" is found significantly more frequently for neural network projects. For GAs the reasons *fixed in advance* and *provides insight in results* are often mentioned. Apart from projects applying GA, several other projects had also fixed the technique to be applied. *(Low) Costs* and *learning speed* are arguments mentioned for applying inductive statistics.

From additional textual responses it can be concluded that most projects focus on activities in the ILT development and ILT operation layers of the model in Section 2.3. Moreover these responses show the need for a comprehensive methodological approach of ILT application. The presented framework may be seen as a first step towards such an approach.

The second item, the consideration of alternative techniques in selecting the ILTs, has also been included. Of all respondents 60% indicate that alternative techniques were considered. The distribution of considered alternative techniques can be seen in Figure 2.13.

Respondents indicate several types of problems and limitations. Two questions in the inquiry explicitly address the problems that may arise in applying ILTs. Respondents were asked to indicate whether, in general, the applied approach of development and the applied ILTs fit, and to indicate shortcomings in the approach. Of the respondents 90% confirms that the approach and the method fit. Several respondents indicated that applying the ILTs was the entire project; one respondent indicated application of a project approach that was specifi-

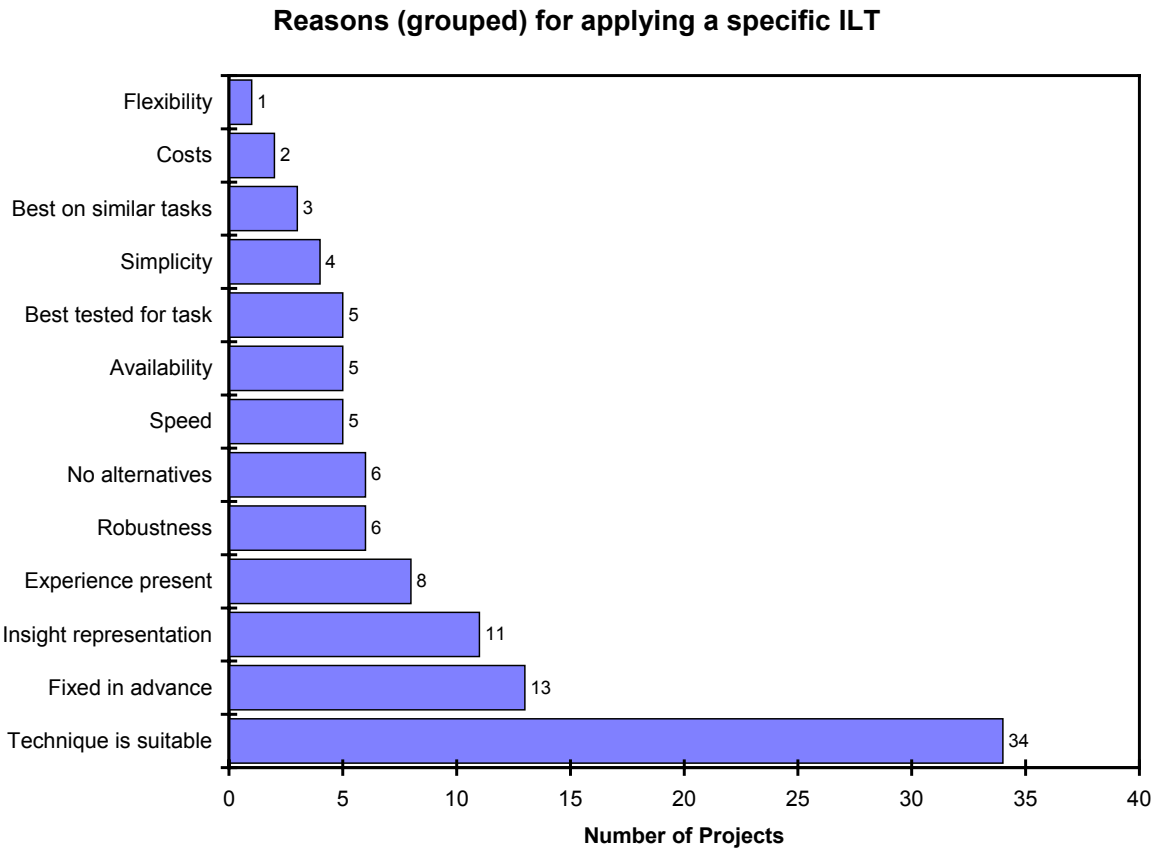


Figure 2.12 Reasons (grouped) for applying a specific ILT

cally designed for ILT application. Moreover, it is signaled that ILTs fit well in a data analysis project. Problems that are mentioned relate to the misfit between the approach for applying a specific ILT and the waterfall approach; the latter is indicated to not support the empirical search conducted with the ILT. Furthermore, some projects were cancelled due to practical problems (lack of data, lack of reproducibility of results, problems making an ILT application operational, and integration between ILTs and embedding system). Finally, problems are reported on ILT issues such as lack of training and operation guidelines, and lack of insight in technique and performance characteristic.

One general problem mentioned is the difficulty with predicting the outcome of ILT applications early in the project. Ideally one would like to decide early in a project if an ILT approach is likely to be successful and what it will cost to implement it as a working solution. However, With these issues being unknown, this uncertainty is an obstacle for industrial applications. A related type of problem occurs mainly in data analysis projects. In this case key tasks are selecting a technique and subsequently finding the best parameter settings for the data set. From the survey we get the impression that selection of an ILT and a tool that implements that ILT is either not done at all, because the technique or tool is fixed from the start, or that it is done empirically. In the latter case available tools are tested to find the optimal parameter settings. Moreover, in these testing, train-and-test approaches dominate. Little use is made of theoretical understanding of these techniques or experience by others (e.g. Weiss & Kulikowski, 1991; Kodratoff et al., 1994; Michie et al., 1994). Because ILTs often have several parameters with a range of possible values and because these parameters interact, many experiments are necessary. Therefore acquiring and applying knowledge,

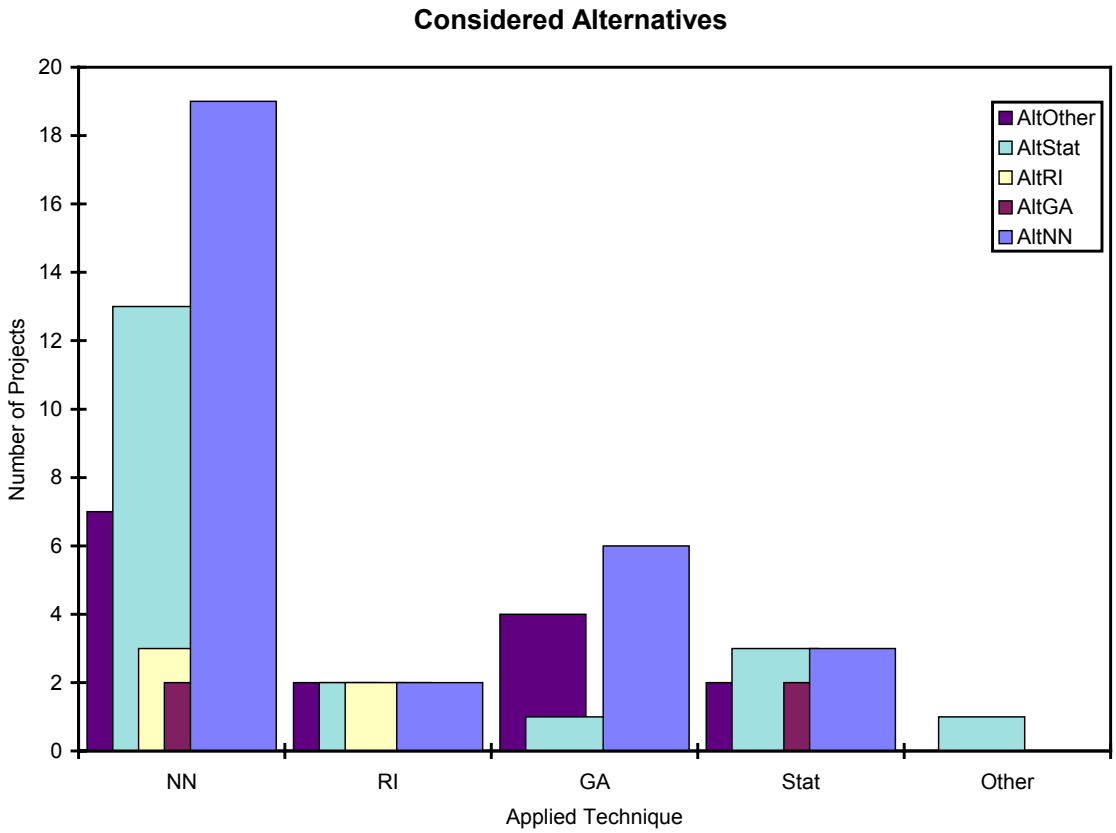


Figure 2.13 Alternatives that have been considered for applied techniques.

procedures, and tools to support the search for the best technique, tool, and parameter settings is a significant problem.

2.6. Conclusions and Further work

Our results show that companies are actually applying ILTs in practice. Concern about modest numbers of practical applications appears to be unjustified by this data. In our limited attempt we identified 79 practical applications, a substantial part of which has lead to operational applications. Moreover, more than half of the projects involved commercial clients.

The character of the applications however is often exploratory and technology-driven. We cannot speak of established application practice and routine use of ILT tools. This is confirmed when looking at the technical-, project- and organizational characteristics of the applications, from which we conclude that ILT application is in its initial phase. This is illustrated by comparing project characteristics with models for (information) technique proliferation in organizations as provided by Nolan and Bemelmans. The proliferation of ILTs to several areas on first glance seems a sign of diffusion (Bemelmans) and popularization (Nolan), but in this case appearances may be deceptive. Application in these areas may very well be the result of the non-administrative origin of the technique, and not of proliferation across domains. The user/client satisfaction for the reported projects is promising, and initial successes have been reported. Technical and methodological problems however still block a wide breakthrough. The solution to these problems is not yet within sight. Therefore transition to a next phase on short notice seems unlikely.

The techniques that are actually used in practice only cover a small part of the possibilities that are offered by ML research. Most applications concern NN techniques and tools. Apparently those who apply ILTs are not acquainted with the state of the art of ILT technology. The result is that a large number of available techniques are not found in practical applications. The practical experiences as reported in our survey however are in general positive.

The developments in research do not completely correspond with the applications as currently realized. The emphasis in research is on *refinement of algorithms*, *multi-strategy approaches* and the role of ML in general AI themes, such as *knowledge level learning*. In applying ILTs important issues are *technique selection*, *pragmatism* and *efficient production of a working application*.

With reference to the application project model, most projects seem to apply an approach focusing on the last 2 levels of the model. Techniques are selected by experimentation from a small pool of techniques. An explication of the steps that are followed in the process is completely absent, so is a full consideration of all relevant factors. An approach like the one developed by DTI, which tries to cover for such shortcomings, is only referred to once as developed by a company for their own research and development activities.

Several methodological problems are signaled that need to be solved. Most projects seem to lack planning and control of the ILT application process. Data selection and transformation, technique selection, and parameter setting are either not done at all, done experimentally, or are done in an unsystematic way. This is probably due to a lack of knowledge about these aspects of ILT techniques and also to a lack of comprehensive tools that make a wide range of techniques practically available.

Our inquiry gives clues as to where to look for solutions to some of these problems. We are now studying several projects that involve the full process of identifying the problem, finding the right approach, selecting a technique and a tool, collecting the data and applying a tool in more detail to better understand the problems and limitations that hinder ILT applications. In addition, we will further detail the scenario-based process model for applying ILTs in practice, based on the four level structure. The final goal of this work is to provide a general framework for ILT application.

2.7. Addendum: Follow up of survey

How has the situation evolved since 1994? Systematic follow up of this study has not taken place. But since early 2000, KDNuggets[9] organizes user polls on that may give a glimpse into the current state. In these user polls site visitors enter their opinions and experiences on a specific subject. The poll results are published on the KDNuggets web site. It is very hard to draw firm conclusions about these figures, as data collection circumstances differ fundamentally[10]. In our survey we individually addressed respondents, after a focused search for potential users, whereas the KDNuggets polls are open to any site visitor. However, some trends may be inferred. The Figures 2.14-2.16 provide material for comparison between the situation in the Netherlands in 1994 and the KDD Nugget respondents in the past years. Concerning technique usage over the last years, Figure 2.14 shows the application of a large number of techniques. This contrasts with the results in 1994 (Figure 2.9 of this chapter), where a strong dominance of neural networks appeared. First, neural networks were very fashionable in the early nineties, especially in the research environments. Second, the number of toolkits, allowing the experimental evaluation of many approaches, was limited in 1994.

Figure 2.15 shows a substantial penetration of standard methods in the application of data mining techniques. Especially CRISP-DM and SEMMA[11] are mentioned. Compared to Figure 2.11 of this chapter, the changes in method application are obvious. CRISP-DM and SEMMA were non-existent in 1994, and alternative approaches were not available to the common users. However, as signaled earlier, many of the non-standard, company specific methods may boil down to basic technical issues as deploying a train-and-test approach and technique selection by cross validation. Nevertheless, approximately half of the respondents use a non-standard method or no method at all.

Figure 2.16 shows the increasing response to the KDD Nugget survey, as well as the growth in (KDD) tools. This illustrates that at least in the area of data mining and KDD, machine learning kept the promise of being close to a breakthrough that it had 10 years ago. Unfortunately, figures on the use of ML in knowledge based systems are not available.

From these results it appears that the conclusions of 1994 still hold. Methodological support through CRISP and SEMMA focuses on structuring the explorative search for applicable techniques. They provide no support in the combining of ML solutions with other techniques. Problems are considered to be pure ML or DM problems per se. Moreover, over time the technique selection problem has become even more urgent, as the number of available techniques has expanded over time, and several toolkits are available that collect numerous techniques. Straightforward technique selection approaches, such as train-and-test and cross validation have not been improved, leaving them with the same drawbacks as before.

---

[9] www.kdnuggets.com

[10] KDNuggets provides a footnote on their poll webpage, stating: While this is not a scientific poll, and some vendors have been quite active in asking their staff or clients to vote, there are several safeguards in the poll mechanism to prevent visitors from voting twice.

[11] SEMMA is indicated in the KDNuggets poll as a method. The developers of SEMMA (www.sas.com) however indicate that SEMMA is not so much a method as well as a logical organization of the functional toolset they provide. SEMMA stands for *Sample, Explore, Modify, Model and Assess*. In content, it is close to the steps defined by Adriaans & Zantinge (1996), Fayyad et al (1996), Engels (1999) and CRISP-DM (Chapman et al., 2000).

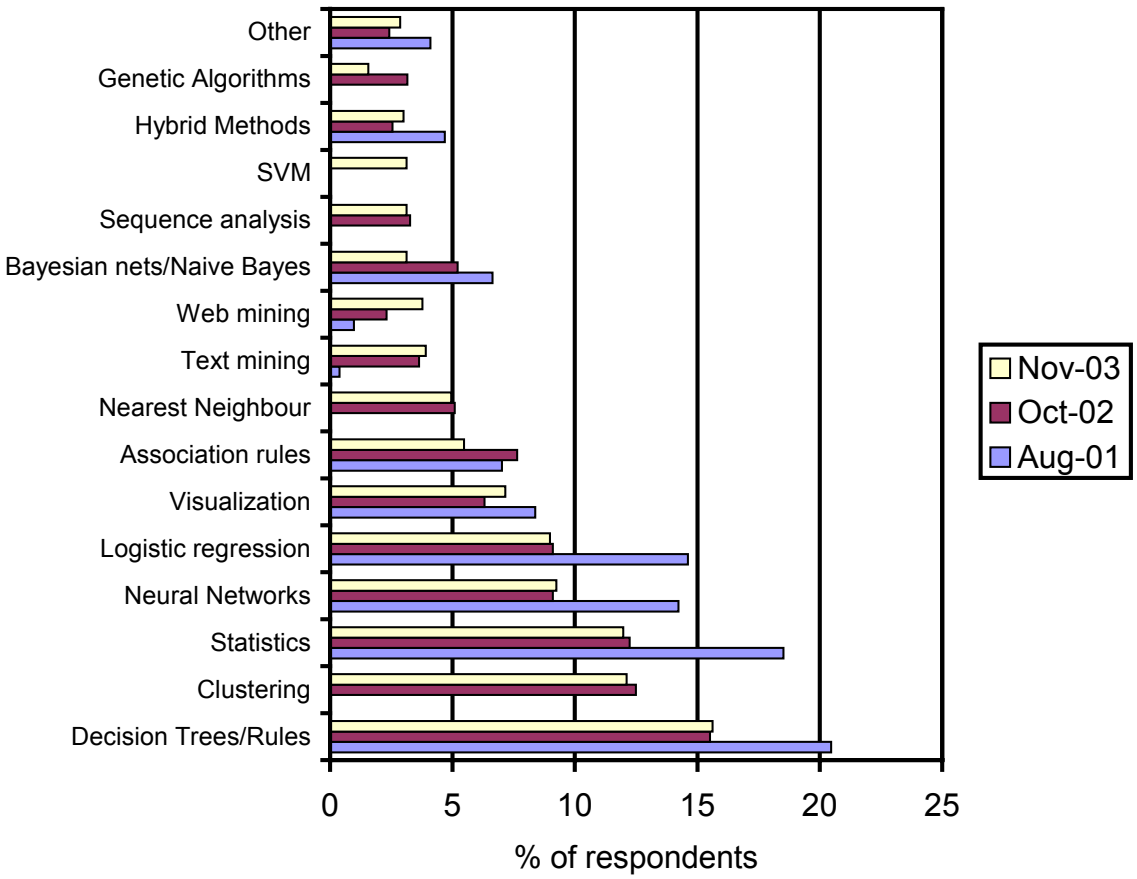


Figure 2.14. The evolution of technique usage according to KDD Nuggets in the period August 2001-November 2003.

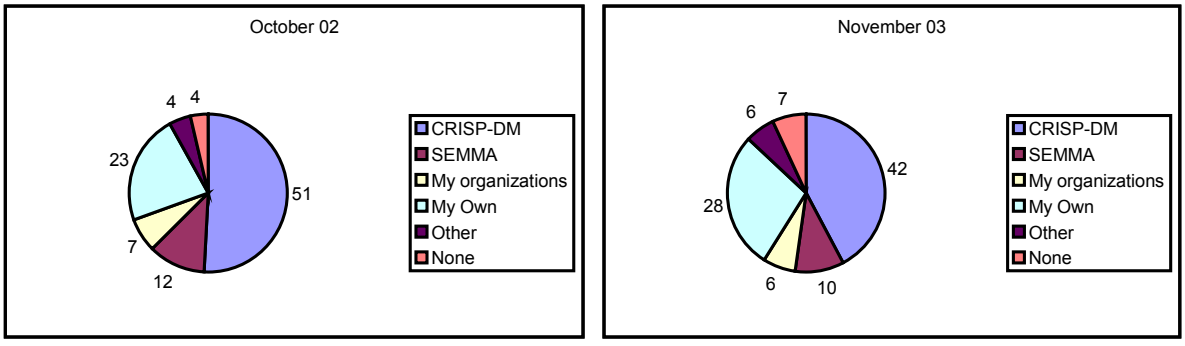


Figure 2.15. Method usage in October 2002 and November 2003.

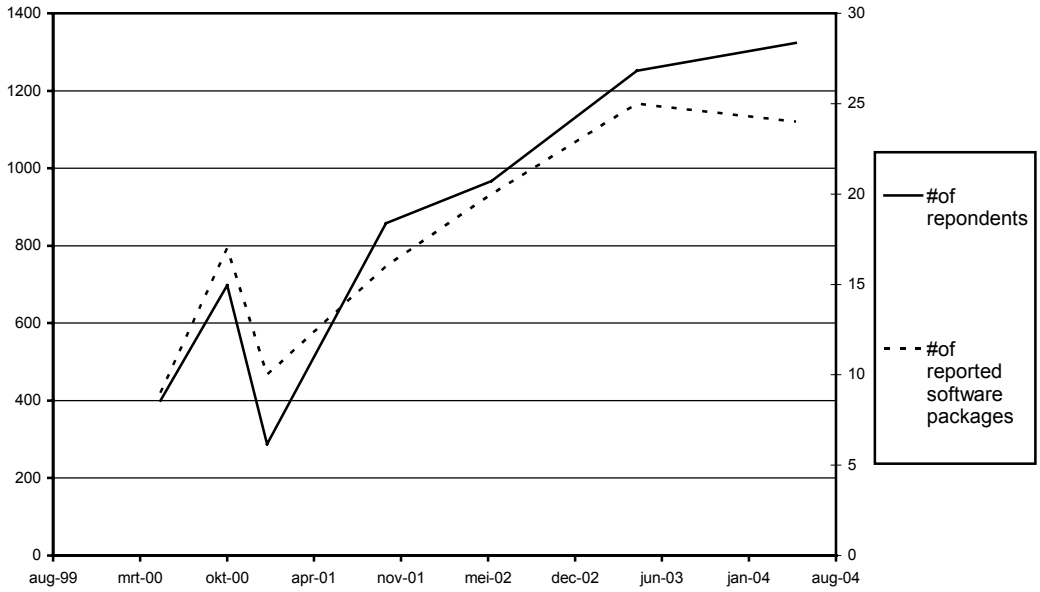


Figure 2.16. The evolution of the number of respondents and the number of different software packages that is reported for data mining and KDD in the period of 2000-2004.

3. Managing Product Inherent Variance During Treatment

This chapter has been published as Verdenius, F., (1996), Managing Product Inherent Variance During Treatment, *Computers and Electronics in Agriculture 15*, pp. 245-265. It is a revised and extended version of the paper that was presented on the IFAC/IFIP/EurAgEng workshop AI in Agriculture, Wageningen, May 29-31, 1995

Abstract

The natural variance of agricultural product parameters complicates recipe planning for product treatment, i.e. the process of transforming a product batch from its initial state to a pre-specified final state. For a specific product P, recipes are currently composed by human experts on the basis of heuristic matches between product state and recipe features. This approach makes use of standard recipes, that do not sufficiently reflect inherent differences between batches. Improvement of the recipe design process requires three problems to be solved: (1) assessment of the initial product state, (2) fixation of the recipe requirements and (3) design of a treatment recipe. To objectively assess the initial product state, additional measurement of a specific parameter is required. This parameter varies substantially between batches, requiring large measurement samples. Without objective assessment however, automated determination of the recipe requirements and recipe design is not possible. This paper describes a procedure to get an objective initial state assessment, and presents a Product Treatment Support System that takes an initial state assessment, and performs the process of recipe design. AI techniques are applied at three points in the process. Induction of decision trees is used to determine rules that are understandable to experts and that select products that are most suitable for state assessment. Neural networks are applied to transform the assessment of the initial state into the overall requirements of the recipe. Finally, the actual recipe is derived by means of constraint satisfaction.

3.1. Introduction

An important focus of agricultural research is the handling of quality variances in raw produce. The characteristics of agricultural products are represented by intrinsic product properties (Sloof et al., 1996). The set of values for all relevant intrinsic properties for a product at one time instant is called the *product state*, the average product state for a batch is called the *batch state*. In general there is a large inherent variance for these intrinsic properties within one batch of agricultural products, as well as between several batches. The impact of these variances depends on the destination of the product (batches). For some processing industries this impact may be minimized (e.g. by mixing or pre-processing). After possible leveling, these industries apply normal industrial quality standards. Products for direct consumption are sorted in quality classes. Each class definition encompasses boundary values for the product state. The class definition may vary over time, depending on season, technical improvements etc. Consumers select products from certain classes, meeting their quality and economic preference.

This contrasts with standard industrial quality systems, where end product quality is defined in terms of product specifications. For relevant quality parameters, these specifications define target values. From these target values, in turn, specifications for raw materials are derived. This standard industrial approach to quality is connected to the application of established procedures and recipes in production processes.

The state variances of agricultural products make it difficult for human experts to assess the actual state of a batch, even when detailed information on batch origin and history is available. This is the case for many agricultural products, both at harvest time and in the post-harvest life, e.g. in the case of melon (problem: assessing the best harvest time concerning internal fruit development; Larrigaudiere et al., 1995), pears (problem: assessing internal deterioration; Wang & Worthington, 1979) and apples (problem: assessing the ripeness stage at harvest time and in the post-harvest life; Streif, 1989). In this chapter the *Product Treatment Support System* (PTSS) is described, a system that provides support to experts in treating the agricultural product P[12]. The batch state variances hinder experts in designing flexible treatment recipes that take the actual batch state into account. The PTSS is developed in an applied research project for a client and delivered as a prototype. It supports the treatment expert in designing recipes that account for the variances in the initial batch state and for different final product specifications. The system is based on existing product treatment methods, but relies on a new measurement technique derived from product research and on techniques from Artificial Intelligence (AI).

The structure of the chapter is as follows. Section 3.2 presents the context of product treatment. Section 3.3 sketches the expert task, the PTSS and its role in the process. Section 3.4 discusses the AI elements in the PTSS: two learning modules, one for batch state assessment and one for recipe requirement prediction, and a constraint satisfaction module for recipe design. Section 3.5 presents the results. Section 3.6 presents the short and long term plans for further research. Section 3.7 concludes the chapter.

3.2. Product Treatment

Product treatment is the active process of directing the internal physiology of a product towards a pre-specified product state in a pre-specified time span (Verdenius et al., 1994). Product treatment is for example applied to products that are harvested in an immature state, and that are either transported over long distances or stored for some time (tropical fruits, oranges, tomatoes). Moreover, some products need different treatments for different target applications. When forcing tulip bulbs for example, the targeted flowering date may vary from early December until early May (and in case of freezing techniques even year-round). The exact flowering period is controlled by applying different treatment recipes (DeHertogh et al., 1983). Treatment recipes mainly consist of temperature regimes.

Product storage, on the other hand, is a conservative process. During product storage the aim is to control the product conditions in such a way that the natural development of a product is impeded. A typical example is the long-term storage of hard fruits such as apples and pears. Harvested in a short period in the autumn, fruits may be stored under Ultra Low Oxygen conditions for up to 8-10 months with minimal changes of product quality.

[12] We product details of product P for commercial reasons. We use example products to illustrate the main concepts. Main examples are drawn from tulip bulb forcing practice.

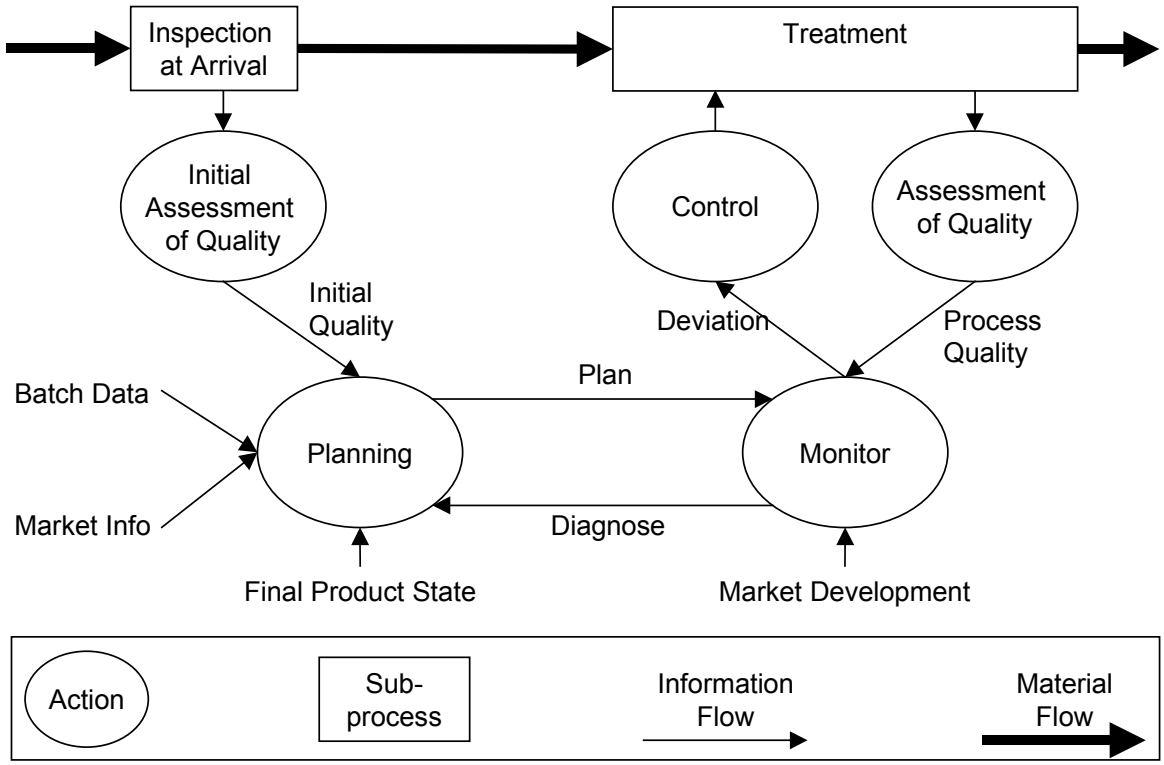


Figure 3.1. Schematic set-up of a treatment and storage processes.

Product treatment and product storage share properties in process set-up (Verdenius et al., 1994). The general structure of treatment and storage processes (TS-processes) is shown in Figure 3.1. The product arrives in a TS-facility, where the initial batch state is assessed. The assessment normally includes both subjective assessments such as a color class and objective measurements such as temperature or biochemical analysis. The information from the state assessment may be completed with information on batch characteristics (origin, transport details, growing details etc.). Based on the initial state assessment, market information (sales volume per period, foreseen destination like client or usage) and a required final product state, a recipe is composed. In the case of standard recipes this final state is a fixed quality that does not depend on batch state or the destination of the batch. The actual TS-process starts, and is monitored on the basis of a state development and possibly other factors such as market development. At finalization of the process the product state is assessed, and the product is delivered.

3.3. Supporting Recipe Design and Process Monitoring

3.3.1. Current Situation

Human expertise and craftsmanship play a central role in the main activities for the treatment of P. The expertise is empirical in nature, and treatment companies all apply their own heuristics. In spite of the differences in detailed content of the expertise however, the structure of both the task and the expertise are similar. At reception of the batches, experts have little information about batch characteristics (exact origin, cultivar, etc.) and history (age at harvest, transport phases, transport durations, transport conditions, etc.). They get a provisional impression of the product quality during the initial state assessment, based merely on outer appearance. This assessment is, with the exception of product temperature, subjective.

Experts use this assessment in combination with a marketing prognosis to design an initial recipe. The basis for the recipe is one out of a set of standard recipes. Recipe design starts with the formulation of deviating characteristics. These characteristics are transformed into adaptations of the standard recipe. This adaptation process is not subject to an explicit reasoning process, but a sort of 'direct mapping', based on experience. Acquiring explicit recipe design knowledge from experts therefore requires substantial effort.

This situation for P is comparable with the situation for tulip bulb forcing. A treatment in this case consists of a series of temperature and residence time combinations. Dependent on the desired flowering date, a standardized treatment recipe is selected that optimizes the bulb quality with respect to that date (DeHertogh et al., 1983; pp. 58-59). The recipes however contain variable boundaries for both residence time and temperature values for each step. These ranges constitute the freedom to calibrate recipes for observed or suspected differences in batch states.

In the course of the P-treatment process intermediate inspections gradually reveal more about the real product quality. The recipe is revised iteratively based on this additional information. During the second half of the process the development of the outer appearance accelerates, leading to an increased number of control actions. The physiology of P, and thus the development of the quality of P during the treatment process, is very much dependent on its history. Differences in treatment recipes explain differences in shelf life duration of up to 20% to 25%. As a result, control actions often come too late to prevent substantial quality losses.

An additional complication in the case of P is the complexity of the product chain. Suppliers deliver various product qualities to treatment companies. Differences between the treatment companies further enlarge quality variation. The product chain is very diverse as well in the after-treatment phase, leading to further divergent product quality.

3.3.2. Problem Identification

During the analysis of the treatment system and of the expert activities, two main problems were identified in the treatment process: the initial state assessment and recipe design.

3.3.2.1. Initial State Assessment

At the start of the project there were no objective measurement techniques available for the assessment of the product state of P. Experts had to rely on outer appearance to assess the product state. There was little difference between experts with respect to this assessment, but the applied assessment scale was not accurate enough to distinguish products and batches with substantially different states. The inner state of the product can be objectively assessed with various direct (and destructive) and indirect measurement techniques. Examples of indirect techniques are respiration characteristics and temperature balance. These techniques however require high quality facilities and accurate measurement equipment. It is questionable whether it is economically justifiable to include such equipment in treatment facilities. An alternative option was to search for more feasible means for identifying the product state.

We found a significant correlation between specific texture characteristics in the initial treatment phase and a single parameter that we will refer to as the A-value, measured by means of excitational vibration (Peleg, 1989). A measurement method was developed that combines this new parameter with one of the more traditional outer parameters that together cover product quality during the entire treatment process. The combination of these two parameters facilitates an objective assessment of the product state, and allows monitoring of the development of product quality.

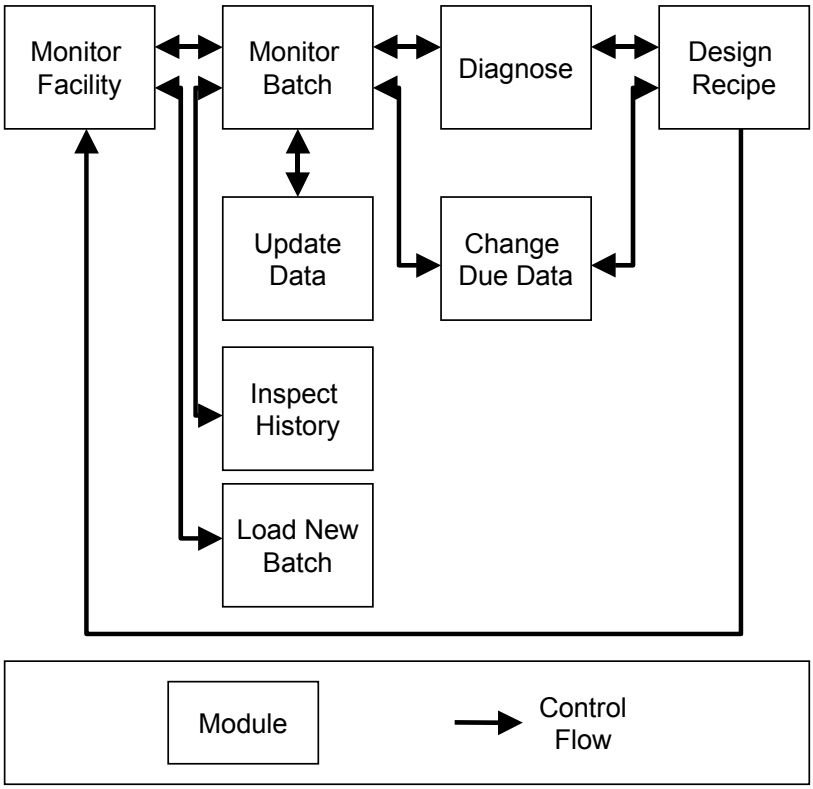


Figure 3.2. The functionality of PTSS in terms of modules and control flow

The problematic aspect of initial state assessment now lies in the large variance that dominates (almost all) product parameters, including the two parameters that are needed for product state measurement. For the A-value variance exists on different levels. Primarily it appears to be correlated with the variance of product dimensions (thickness, weight, etc.). Secondary sources of variance are the product origin, seasonal and yearly differences, variances in harvest-to-treatment time, variances in transport conditions etc. All but the product dimension related sources of variance can be considered constant for an entire batch of products. For the measurement method to be reliable, a selection procedure had to be designed that ensures the selection of a sample that is representative for the batch.

3.3.2.2. Recipe Design

Recipes are composed of time slices of a standardized duration (typically 0.5 or 1 day). A recipe prescribes the values for all relevant conditions. Normally, these conditions are temperature and gas composition. Some treatment facilities also have the opportunity to measure or control other parameters, such as the relative humidity, but this is not standard. The general format of treatment recipes is as follows:

A recipe unit U_t for product P for time-slice t is $\{T\ G\ [RH]\}$, where T is temperature, G is gas composition and (optionally) RH is the relative humidity. A complete recipe consists of a series of units U_t for each time slice of the recipe duration.

Detailed knowledge of the relation between treatment conditions and product development is required for designing a treatment recipe. Expertise is foremost empirical, and not based on theoretical insight into product behavior. Moreover, a complete theoretical understanding of product behavior does not yet exist. Our initial approach was therefore to mimic expert behavior in this respect. The resulting recipe generator applied a static set of base recipes, and generated deviating factors that were transformed into adaptations of the recipe. Though this

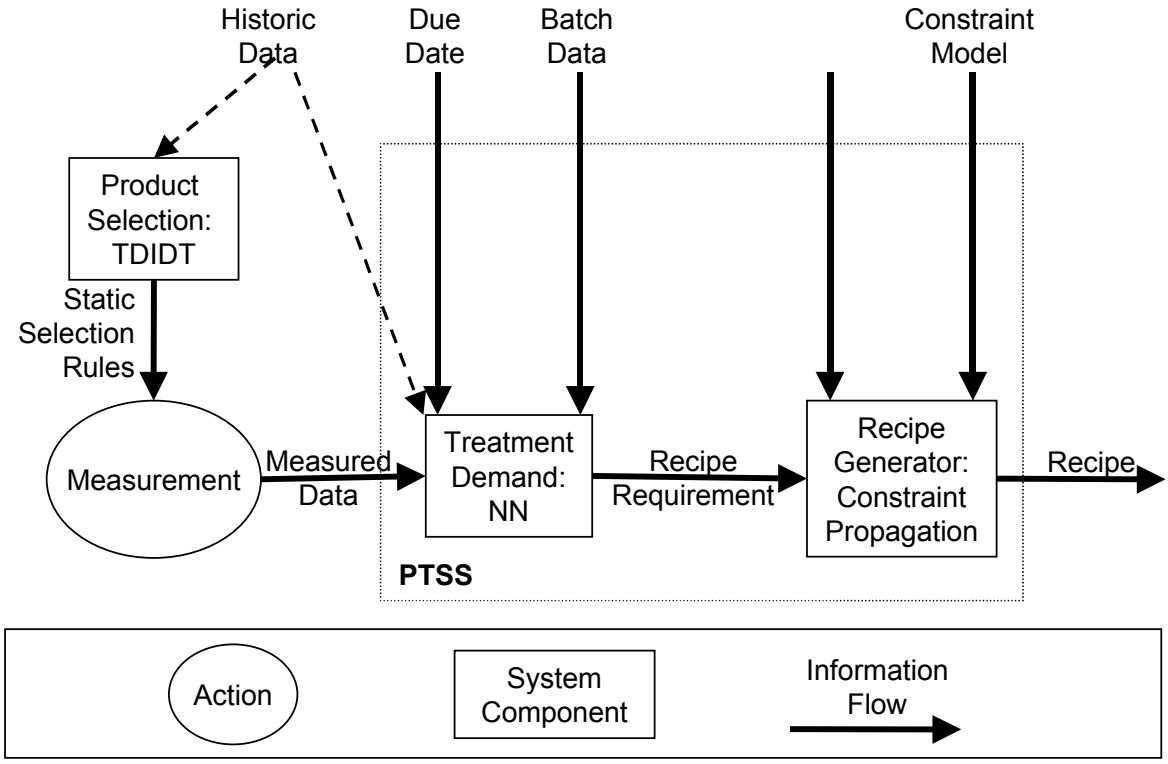


Figure 3.3. Connectivity of AI modules in PTSS

approach seemed to work for one expert, problems arose when taking knowledge from different experts into account. Experts from different treatment companies created different and incompatible models. We had to solve this problem by conducting experiments revealing actual product behavior to generate our own models.

The solution was found in developing a new approach for recipe design. Recipe design now consists of two steps. The first step is to assign recipe requirements to a batch on the basis of initial state measurements, batch data, the foreseen due date and final product specifications. A recipe requirement is a global specification of a recipe that quantifies relevant condition values of that recipe, without specifying how this requirement is realized in U_t. An example of such a specification is the temperature sum factor that is used in growth models. It is calculated by adding the daily temperature difference where the temperature remains below or exceeds a specific temperature threshold. Recipe requirements can be composed of several values for e.g. temperature, relative humidity, ethylene concentration, O_2/CO_2 ratio etc.

Because a detailed mechanistic understanding of the behavior of P during treatment is not available, a model of product behavior is derived from experiments in our pilot facilities.

The second step derives the actual recipe from the recipe requirements. This derivation applies empirical knowledge acquired from experts for those aspects where inter-expert agreement exists, and solves inter-expert disagreements by applying knowledge of the product and the treatment process as acquired from literature and from experiments.

3.3.3. Functionality and Implementation of PTSS

An important effect of the PTSS is that it structures human expertise and solves inter-expert inconsistencies. Moreover, it provides experts with tools and models to assess product state and represent product behavior. Dedicated physiological research is combined with system analysis

and system development. This research resulted in additional knowledge and a new method for state assessment as indicated previously.

The general functionality of PTSS is shown in Figure 3.2. A treatment facility is equipped with several treatment rooms where the actual treatment of the batches takes place. The basic function is the *Monitor Facility*. This provides the expert with state information of all batches under treatment. Based on regular (possibly on-line) measurement and comparison of actual and foreseen development, the expert receives early warnings in case of possible problems. From the *Monitor Facility*, the expert may decide to *Load a New Batch*, or to *Monitor a Batch* already under treatment. The latter enables the analysis of the actual batch development as compared to the expected development. Once in this module, the expert can *Update* the inspection data (if not automated), and *Inspect the Batch History*. Deviation of the actual from the expected development may lead to execution of the *Diagnose* module, supporting the expert in problem solving. Diagnosis uses predictive models, allowing deviations in product development to be corrected in an early stage. If required, a treatment *Recipe* can be *Designed*.

The PTSS-prototype has been implemented as a stand-alone system in KAPPA-PC™ version 2.3.2, by IntelliCorp. Linking to treatment facilities is foreseen on the design level, but currently the system functions off-line. The prototype has been developed on a 486 66 MHz PC under MS-Windows 3.1™.

3.4. AI Components

Three specific AI techniques have been applied in developing the new treatment approach. A technique for Top Down Induction of Decision Trees (TDIDT) is used to handle the variance during the initial state assessment. Neural Networks (NN) transform the initial state assessment into the recipe requirements, and constraint satisfaction is applied to derive the treatment recipe. Figure 3.3 shows the set-up and connections between these AI components.

3.4.1. Handling Variance with Inductive Learning

3.4.1.1. Function

PTSS requires the measurement by means of vibrational excitation (Peleg, 1989) of the A-value to reveal the inner state of product P for a batch. Measurement across multiple individual products shows product inherent variance. If a reliable estimate for the A-value of a batch has to be obtained, a large sample size is needed (depending on the accepted chance of calculating a false estimate and the threshold for relevant differences, the sample size is about 60 to 200 samples). Measurement of A in practice is only feasible if the sample size is reasonably small. Currently, experts take about 3 measurements.

We design a selection procedure that enables sampling with reduced variance. The procedure consists of two steps, the first step being a selection on externally observable characteristics such as size and shape. The second step requires measurement of a few, easy to obtain, attributes related to product appearance and dimensions such as objective product color and exact product dimensions. The measured values are used to decide whether or not the product is suitable for A-measurement. As experts do not typically handle complex directives, comprehensibility of the criteria is a major concern. The delivered sampling directives ensure reliable A-value estimation for the entire batch on the basis of reasonably small samples.

3.4.1.2. Method

The necessary decision rules are derived by means of an inductive learning technique. To achieve this, we use the distribution of the A-values, which is a normal distribution. Products

are classified on the basis of their deviation from the mean A value for a batch. Classes were established for 105 products from 8 batches from 3 growing locations, transported under two different conditions. For all products, product related attributes were measured. Batch characteristics were not included in the data set.

Our aim is to find a classification function that is able to identify *near_batch_mean* classes for unseen cases. Typical intra-batch standard deviation σ_{batch} for A is in the range of $\sigma_{batch} \leq 3$ on a scale of 0 to 30. Products are classified as *near_batch_mean* if their A-value falls in the range

$$A_{product} \in [\mu_{batch} - t_{near\_batch}\, \sigma_{batch},\ \mu_{batch} + t_{near\_batch}\, \sigma_{batch}],$$

where μ_{batch} is the batch mean value for A, and t_{near_batch} is a threshold parameter that defines the width of the interval in terms of σ_{batch} . Good results were obtained for $t_{near\_batch} = 0.75$.

The classification rules were derived using a TDIDT technique. TDIDT is a group of Machine Learning techniques that extract a tree-shaped knowledge structure for classification from pre-classified data sets (see Figure 3.4). Each splitting branch in the tree represents a test on an attribute value. Leaf nodes indicate the class that is expected for a product if the consecutive tests have been successfully passed. TDIDT is known for the good comprehensibility of the output for domain experts, either in tree-form or in the form of IF .. THEN .. rules.

We use *C4.5* (Quinlan, 1993), a well-known TDIDT technique, to construct 15 decision trees by means of a windowing technique. This is a technique where a decision tree is iteratively trained with more and more examples until a satisfying classification result is obtained. From the derived decision trees, rules were derived by application of *c45rules*, a rule derivation program accompanying *C4.5*. Rules that identify *near_batch_mean* classes, were tested on their quality. From the initial 41 rules selecting the *near_batch_mean* class, a set of 7 rules remained that were applicable in practice, an example being:

```
IF      Colour <= 10 AND
        Shape_Class = C AND
        Weight <= 175
THEN    Class = Near_Batch_Mean
```

All these rules require 3 or 4 attributes to be checked. The standard deviation of measured products on A is now reduced to about 0.3 σ_{batch}. Applying these rules, a sample size of 5 products suffices to obtain an estimate for the A value. This sample size is sufficiently small to be convenient for the normal process monitoring cycle.

3.4.2. Prediction of Recipe Requirement by Means of Neural Networks

3.4.2.1. Function

A recipe is a prescription of attribute values for the relevant process conditions for all time-slices in a treatment. In the current practice for P-treatment, standard recipes are available for (fixed) treatment durations. Such recipes are adapted by the expert on the basis of a perceived product state. If the foreseen treatment duration changes as a result of rescheduling, or if the product development lags behind the desired development, recipes are changed according to the insight of the expert.

In the PTSS recipe, design starts with establishing the treatment requirements for a batch. Treatment requirements are expressed as the thresholded temperature sum, eventually supplemented by other conditions, such as gas conditions or humidity. For this a regression model is required that predicts the treatment requirements on the basis of available batch data and required treatment result. Relevant batch data include categorial information (e.g. cultivar, origin, shape) as well as numerical information (e.g. product dimensions, A-value). Actually, humans are very good at solving prediction problems based on such heterogeneous data

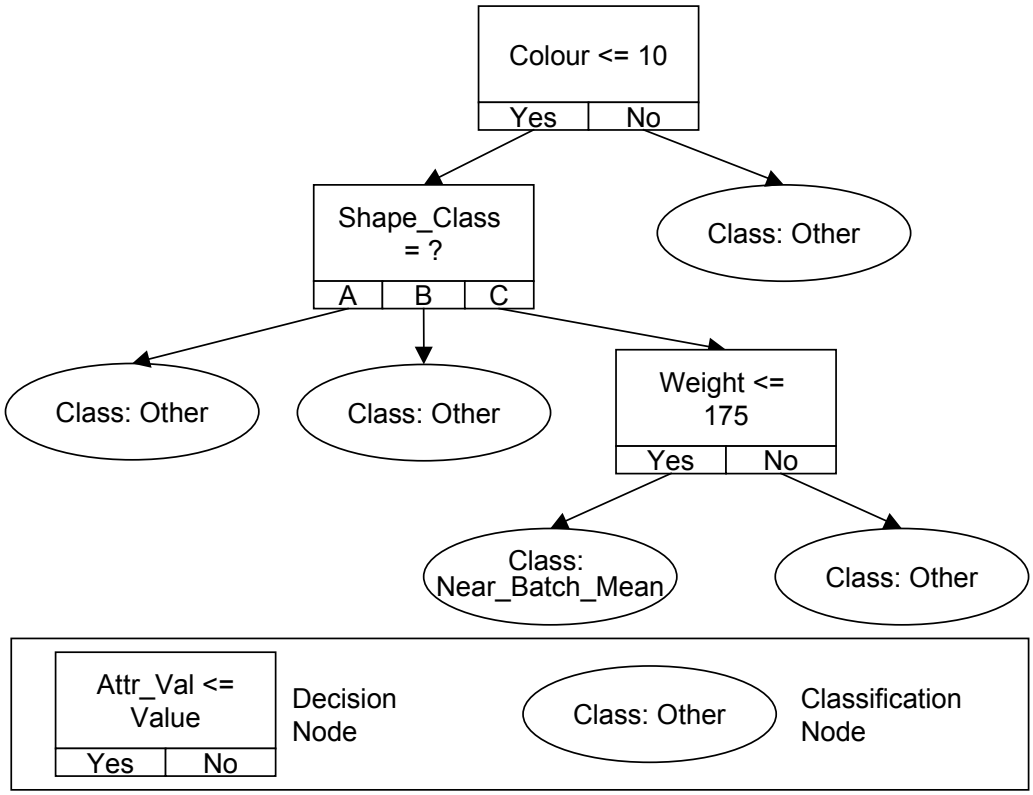


Figure 3.4. A decision tree for product selection. Decision nodes represent tests on attribute values (two numerical and one categorial test are depicted in this example tree). Nodes may contain additional data on classification quality, significance etc. (Feigned example, real trees are more complex)

attributes. Good mathematical models to derive the requirements from batch data for P are not available.

Neural networks (NN), a family of self-learning techniques, function analogously to the human nervous system. They process information in a network of interconnected layers of neuron-like elements (see Figure 3.5). One specific NN technique, error back propagation (Rumelhart et al., 1986) is very well suited for our kind of problem, being able to process numerical and categorial data simultaneously.

After training, the neural network contains a model of the experimental data. The network takes batch characteristics and initial state information as input and delivers a prediction of the recipe requirements. Ongoing experience in designing recipes and the accompanying treatment results, can be used to refine this model.

3.4.2.2. Method

Data was gathered by treating the batches as mentioned in Section 3.4.1.2 with two recipes under laboratory conditions. We use a data set of 105 products complying with the selection criterion as developed under Section 3.4.1.2 for deriving the regression model. The data include 21 relevant attributes, including several measurements. From these attributes, we derive a linear regression model as a starting point for further work. The linear regression result in a model incorporating 4 variables, obtaining a fit R^2 of 0.57. To model the treatment behavior, one can now choose between two viewpoints: 1) fitting a number of (non-linear) regression functions, hoping to obtain a good fit (black box modeling), or 2) deriving from a thorough understanding of fundamental processes a regression function, that can be fitted to

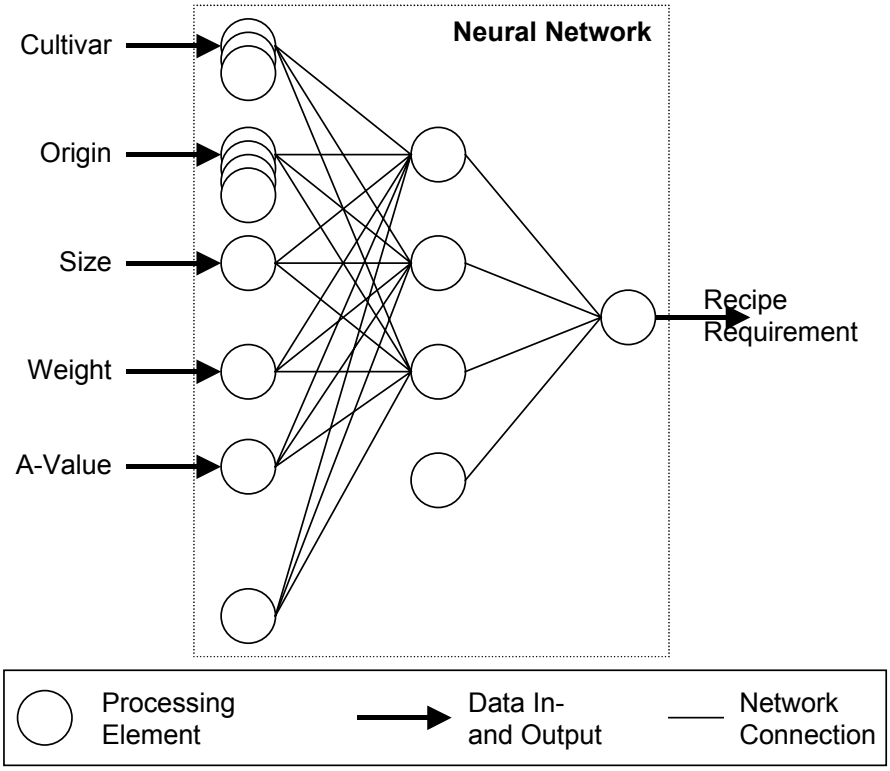


Figure 3.5. A 3-layer feed-forward neural network. The network mimics a regression function transforming data on the input layer (left) into output data (right). The two nodes with no inputs are bias nodes.

the experimental data (domain modeling). As we do not have a complete understanding of the problem domain, we are bound to follow the first option. Neural networks are well-suited to searching for a model in this manner.

Data from experiments are used to train an error back propagation neural network (Rumelhart et al., 1986). We experiment with several data set-ups. First, we vary the number of records offered to the network. This is done by artificially extracting a multitude of *mini-treatments* from one treatment record. A mini-treatment is generated by taking any combination of two time-slices in the data set to be a new treatment. In this way, we are able to generate data sets with about 7000 descriptions of *mini-treatments*. Due to the non-linear process, information on treatment history has to be included in mini-treatment descriptions. Second, we vary the number of attributes to be included in the data set. Attributes are added in order of the correlation to the treatment requirement. Finally, we experiment with the network topology, the number of nodes used in the hidden layer. The learning rate is set to 1.0, and the momentum is set to 0.9. The training tolerance (the deviation of a real valued prediction from the real value under which a record is classified as correct) is initially set to 0.1, and is step-wise reduced down to 0.01 if more than 80% of the records are classified correctly. Testing tolerance is set to 0.05.

The network that performed best is a small network, with the fewest number of attributes and trains on the smallest data set (not containing the mini-treatments). We explain the small network size by assuming that generalization by a small network filters the remaining noise in input characteristics. The fact that mini-treatments hinder performance may be understood as follows. The behavior of P under treatment is suspected to have a logistic trend. Little noise in the initial product condition is amplified during the treatment, and extinguishes again when the treatment comes to an end. When creating mini-treatments, data is generated about the section in the treatment that is very noise sensitive. Where the initial data set suffers from

	3 hidden nodes		6 hidden nodes	
	train	test	train	test
Average R^2	.9877	.9400	.9968	.9861
Best R^2		.9949		.9971
Best validated prediction		.075671		.107319

Table 3.1 Performance of neural network in four settings. The settings differ in the number and kind of attributes that are included (see text).

relatively little noise, generating mini-treatments emphasizes the importance of noise in the data set.

We set up an intensified training procedure to assess the actual value with the two most promising network configurations, using 3 and 6 hidden nodes. Via random sub-sampling the data set is divided into 5 train-test set combinations, which, for both 3 and 6 hidden nodes, are presented to networks with different initial settings. The networks are trained for 5000 cycles. The network that tests with the highest R^2 on the test-set is designated as the result of that run.

The final selection for a network is made on basis of the generalization performance on a validation data set (see Section 3.5), resulting in the network with only 3 hidden nodes being used in the PTSS. Execution of the neural network is controlled by the PTSS. Some typical results for the 3 and 6 hidden node network are presented in Table 3.1.

3.4.3. Constraint Satisfaction for Recipe Design

3.4.3.1. Function

The next (and last) step in recipe design is the transformation of the recipe requirement into a recipe. Not all recipes that fulfil the requirement are allowed. To be of a good quality, the recipe has to fulfil a number of constraints. Constraints either originate from domain experts, from their heuristics, or result from research (both literature and dedicated research). Constraints may limit minimum and maximum values (e.g. temperature, humidity, gas concentrations), consecutive differences (e.g. temperature for 2 successive time slots may not differ more than x degrees) or batch parameters (e.g. a product from origin X needs a lower temperature during the first half of the treatment).

Constraint satisfaction is used for this transformation. Constraint satisfaction has been applied elsewhere for recipe design (e.g. Aarts 1992). In PTSS all aspects of a recipe are formulated as constraints, being conditions between attributes that have to be satisfied in a good recipe.

3.4.3.2. Method

Recipe design by means of constraint satisfaction is a search process. The search problem is formulated as a tuple $<V, C, H>$, where V is a set of variables, C is the set of constraints and H is the set of search heuristics that may be applied in the search process. The value of each variable v_x in V is limited to a finite domain. Typical members of V are temperatures T_i, relative humidities RH_i, and concentration of gas component j G_{ji}, all to the i^{th} time interval of the recipe. A good recipe consists of an assignment of values to all members of V, such that all constraints in C are satisfied.

A constraint set C for the recipe of P-treatment contains unary, binary and n-ary constraints. Unary constraints are immediately translated into domain restrictions on variables. Binary and

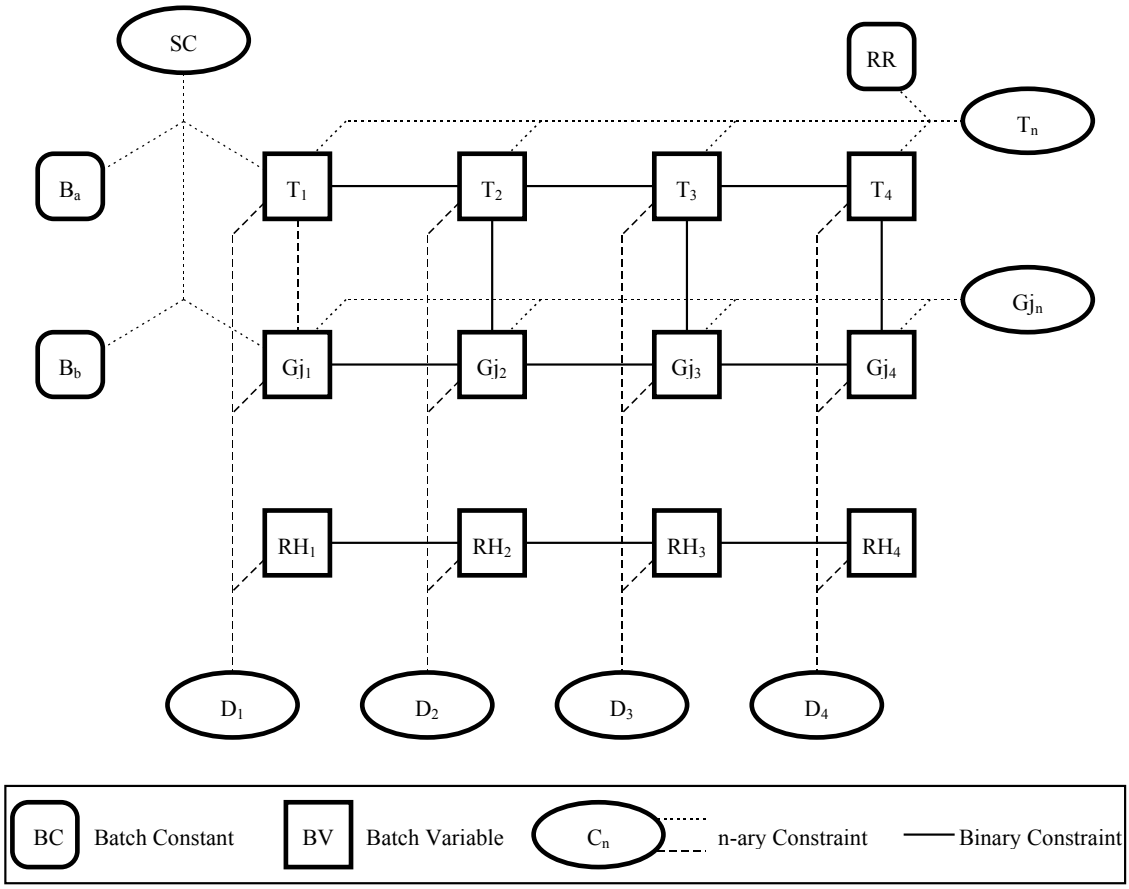


Figure 3.6. A (simplified) example of the constraint set that is used to generate the recipe. The network consists of both binary constraints (ellipse + solid lines) and n-ary constraints (dashed lines). Batch Constants are attributes that are constant during a treatment. Batch variables represent the recipe components to be planned. T = temperature, G = gas concentration, RH = relative humidity, SC = start constraint on initial conditions, RR = recipe requirement and D = Condition constraint for one day

n-ary constraints can link variables of the same dimension (e.g. two consecutive temperatures) or variables of different dimensions (e.g. a temperature and a gas condition).

After obtaining the recipe requirements from the neural network, a constraint satisfaction formulation for the recipe is derived automatically. Constraint propagation, to reduce the search space, and back-tracked search (Meseguer 1989; Kumar, 1992) are iterated until a recipe is derived. A module for constraint satisfaction, including these two steps, has been implemented in KAPPA-PC™.

As recipe design in PTSS is normally an under-constrained problem, different search heuristics may result in good, but different recipes. Set *H* contains search heuristics that are applied in practice. Each search heuristic concentrates on specific operational aspects of recipes. Typical members of *H* are *start as high as possible* (reason: to optimize utilization of resources the recipe is as short as possible) and *start as low as possible* (reason: optimal control over batch, and retains opportunity to speed up later). The choice between heuristics depends on external (e.g. market) circumstances as well as personal preferences of the expert. By using heuristics to guide the search, a find-first strategy can be applied: the first recipe that is generated according to the heuristic and that satisfies all constraints is proposed to the expert.

The recipe design process for product P as such is not very complex (see Figure 3.6). Typically, a problem counts 15 to 48 variables and 22 to 72 constraints, resulting in a moderate sized search space. An important reason to apply constraint satisfaction together with a heuristic search strategy is that the expertise is presented by experts in terms of constraints ("the treatment temperature must be in the range of *m*°C to *M*°C", or "if origin is *C* then decrease the temperature"). Moreover, research results can also easily be expressed as constraints. Therefore, a constraint model is a representation close to the current practice, and also easy to maintain. Alternative representations include instances or augmented cases. Both representations facilitate a case-based approach (Aamodt & Plaza, 1994): knowledge-poor (instances) and knowledge-intensive (augmented cases). The latter has been used in recent systems by Aarts (1995). Instances are not suitable as a representation in the PTSS as they constitute a non-inspectable knowledge base (due to the lazy generalizations used in instance based techniques). Augmented cases are only suitable if a model exists to augment actual cases. Moreover, they share the problem of inaccessibility of the knowledge. This is very crucial in the PTSS as the knowledge base is compiled from knowledge elements that are obtained from different sources (expert heuristics, literature, experimental results).

3.4.4. Example: PTSS Application Demonstrated for Tulip Bulb Forcing

How can the PTSS be used in practice? The following example is situated in the domain of tulip bulb forcing, covering all aspects of the real PTSS domain. We start with a very short introduction in tulip bulb forcing.

3.4.4.1. Tulip Bulb Forcing

Bulb forcing is by definition, "*... the flowering of a bulb using other than naturally occurring conditions*" (DeHertogh et al., 1983) and is used for the production of cut-flowers. The forcing of tulip bulbs (Visser, 1993) consists of consecutive (1) *preparation* of the bulbs in special facilities and (2) flowering in a greenhouse. During the preparation phase bulbs are exposed to conditions (mainly temperature) which program the time of flowering and flower quality. This phase is divided into three steps:

1. *G-initiation* (treatment until bulb development stage *G*, i.e. flower initiation completed): flower development in 20°C conditions, preceded by a 34° 'initiation period' in case of early harvest,
2. *intermediate treatment* (from stage G to the cold period): 17-20°C storage to minimize flower abortion,
3. *cold period* (low temperature mobilization): to ensure sufficient scape length and growth rate.

Commercial bulb forcing needs optimal control of the flowering period to schedule activities and optimize resource allocation. The recipe for these steps should ensure a predictable flowering period. Moreover, the number of days the bulbs need to be located in the greenhouse before cutting the flowers must be minimized. In other words: the bulbs require a pre-specified change of the product state in a pre-specified period of time. In the current situation standard preparation recipes are used. These recipes are based on the *average* behavior of products and desired flowering date (Boerma & VanRijbroek 1992). They neglect environmental differences (DeHertogh et al., 1983; pg. 82) between batches, such as differences in soil or weather conditions during growth. Differences between expected and observed product state development, resulting from inter-batch differences and inaccurate state assessment, are compensated by small adjustments of preparation recipes. The content of these adjustments depends on the expertise of individual forcing experts. This expertise is a non-documented part of forcing craft.

3.4.4.2. PTSS for Tulip Bulb Forcing

The PTSS for tulip bulb forcing follows the process flow as depicted in Figure 3.3:

1. Product selection[13]
2. State Measurement
3. Treatment Requirement
4. Cold Storage Recipe Design

Below we detail each phase. The content of the example is fictitious and does not reflect any relation with the current practice in tulip bulb forcing.

Step 1: Bulb Selection

At the start of a cold treatment the initial quality of a batch has to be assessed by inspection and measurement. The PTSS requires a (qualitative or quantitative) development label for the bulbs that are contained in the batch. In the current practice, this label is assigned after measuring a random sample. Large sample sizes may be required to obtain a reliable result. The PTSS uses a non-random sample of reduced size, that ensures that the batch state assessment be as close to the real batch state as possible. The new sampling procedure is implemented in the form of easy to apply selection rules as illustrated below. Measurements from practice can be used to generate these rules. Product experts are equipped with charts containing the selection rules.

Batch:
    Growing soil type = Dune Sand and
    Bulb size >= 12 and
Bulbs:
    beyond stage *G* and
    tunic colour RGB of [128 64 0] and
    Root length > 0.5 cm
will have:
    State measurement values close to the batch average.

Step 2: State Measurement

The second step is not typical to the PTSS: batch state measurement. Depending on the parameters to measure, this measurement may be off-line or on-line. Such measurements could include outer dimensions, internal development stages and concentrations of certain compounds (e.g. starch, sugar, biochemicals). The current PTSS prototype supports off line measurement, which enables the entry of measurement data into the PTSS either by hand or by an electronic link from measurement equipment.

Step 3: Treatment Requirement

As the third step in applying the PTSS the recipe requirements are established. After entering the measurement data the operator is presented with the treatment requirement as derived by the system. This requirement is used to generate a recipe. For this example it is assumed that cold treatments are specified by a coldsum C that is supplied to the bulbs. C is defined as:

$$C = \sum_i |T_i - \tau|$$

[13] This may be a very theoretical step for bulb forcing, and may not be relevant in practice. Such a step is necessary for many products however.

with T_i being the realized temperature for day i, and τ being a known temperature (probably cultivar dependent). Treatment requirement prediction requires a regression function

$$C = f(\text{OriginData}, \text{Preparation Data}, \text{BatchData}, \text{CultivarData}, \text{OtherData}).$$

OriginData contains specifications such as growing soil and climate. *PreparationData* contains information such as process conditions during G-initiation and intermediate treatment. *BatchData* contains information on batch measurements (e.g. state assessment) and assortment. *CultivarData* contains information on the cultivar or mutant represented in the batch. And *OtherData* contains relevant miscellaneous information.

This function has been implemented as the neural network in the PTSS for product P. The model is trained on the basis of examples, obtained from observed batches. Alternative model formulations (expertise model, (non-)linear regression model) can be considered if sufficiently detailed knowledge about relevant processes is available to allow hypothesis specification.

Step 4: Cold Storage Recipe Design

As the final step, the detailed recipe has to be designed on the basis of the global recipe specification. A recipe is a prescription of all relevant conditions over time. When the relevant time slice is a week, and temperature is the relevant condition, the following holds for recipe design.

$$R = \{T_1, T_2, \ldots T_i\}$$

R is a recipe for i weeks. For now, we assume that the following assumptions hold:

- Optimal temperature for stalk elongation effects: 9°C (Visser, 1993),
- Bulb physiological processes cease at temperatures below 2°C. In this case, no additional cold sum points are counted,
- Required C translates into a (small) interval of calculable size, with a tolerance δ%. Minimum C values are required to guarantee sufficient cold treatment for acceptable flower quality and a minimal number of greenhouse days, while a very large C is too costly: $C_{planned} \in [(1-\delta)C_{required}, (1+\delta)C_{required}]$,
- During cold treatments, temperatures may not increase.

This leads to the following constraint model:

1. $T_i \geq T_i + 1$
2. $C_{planned} \in [(1-\delta)C, (1+\delta)C]$
3. $C = \sum_i |\max(2, T_i) - t|$
4. $t_1 = 9°C$

Suppose that for a batch of bulbs the predicted $C_{required}$ value is 86.5, and tolerance d = 1% (i.e. the $C_{planned} \in [85.64, 87.37]$, that the recipe duration is 20 weeks, and that the τ has a value of 10°C, then a recipe of 8 weeks 9°C , 6 weeks 5°C and 6 days 2°C satisfies the listed constraint set.

Some additional remarks

The above example is made as simple as possible without losing validity. First of all, cold treatment is not a stand-alone activity. Instead, all bulb treatment phases may be planned into one *preparation recipe*. In this case, separate phase specifications may be required for each sub-phase. Also, the constraint model for recipe design should either be partitioned in several

sections, or a constraint model should be formulated to cover all three phases. It is obvious that such a model would be far more complex, covering both interactions between phases and fine tuning of the conditions in each phase. But even the cold treatment process is more complex than indicated. Especially the phase of recipe design may profit from insights into the physiological processes in tulip bulbs. A more realistic model would detail the effects of endogenous growth regulators. In our example, we assume a black box design for stalk elongation. The cold sum serves as an input to that model. In reality more complex processes responsible for the stalk elongation could be specified in the constraint model. This could include the activity of gibberelin (Rees, 1969), accumulated carbohydrate metabolism (Moe and Wickstrom, 1973) and others (DeHertogh et al., 1983).

3.5. Initial Results

We tested recipe planning for product P on 10 batches with known treatment history, using the *start as low as possible* heuristic (which mimics the strategy used for planning the recipes that were applied to the batches). The test results are shown in Table 3.2. The quality of PTSS generated recipes is evaluated in two ways. First, the quality of the NN prediction of the recipe requirements is assessed. This is done on the basis of the relative performance of the prediction as a deviation of the real value. A deviation of 10% is considered acceptable. The results of the NN requirement model are depicted under the column *Requirement Model Error* (RME) in Table 3.2.

$$RME = |Prediction - Realized|/Realized$$

Second, the quality of the derived recipe has to be established. This is done by comparing the derived and realized recipe on the basis of the recipe specification. The specification of the planned recipe never exactly matches the specification of the real recipe, due to rounding errors. The total deviation, Relative Deviation Requirement vs. Recipe (RDRR), should stay within acceptable limits.

$$RDRR = |requirement - recipe\ specification|/requirement$$

It is further noted that the specifications of the realized recipes differ substantially from the (now applied) standard recipes. The global specifications of standard recipes have values that are 13% to 35% smaller than the global specifications of the realized recipes.

How should we interpret these results? First, one has to be careful to interpret the results of the PTSS on the basis of so few samples. A thorough assessment of its performance is only possible after extensive testing at real sites. Second, it can be concluded from Table 3.2 that predictions of the recipe requirements are quite good in most cases, though substantial errors may still occur. However, compared with the current situation, it is an improvement to offer experts objective tools with this accuracy, instead of leaving them to adapt standard recipes *which are themselves inaccurate*. Moreover, the PTSS supports an iterative process, where experts inspect and measure the products regularly, facilitating identification of errors that may be corrected earlier than before. Third, it is shown that the deviation of the realized and planned recipe is limited for most batches, i.e. the planned process complies with the real requirements.

These results are encouraging enough to apply PTSS in real treatment facilities. Parallel to these validations, improvement and refinements have to be developed for all system components, at least for the regression model. As both the selection criterion and the recipe requirement module are implemented as learning techniques, it can be expected that ongoing experience with the system will cause the performance of these modules to be improved.

3.6. Further Research

The PTSS can now be used to plan recipes for P treatment in our pilot treatment facilities. Currently, the project is awaiting approval for the next stage of validation, which is the experimental deployment of the PTSS in one or more commercial treatment facilities. Initial tests will focus on describing the differences between the PTSS approach and the expert. After fine tuning the PTSS, its ability to support expert tasks can be tested.

Several extensions of the PTSS can be considered. First, it is noticed that the PTSS in its current form hardly uses fundamental understanding of physiological processes in product P. Instead, associative models have been used. Product research may reveal a more fundamental understanding of the low level processes occurring in P. This understanding may enable detailed reasoning steps on treatment aspects to be included in the composition of the recipes.

Second, process monitoring and diagnosis of unwanted treatment developments is now supported in a primitive way. This deserves further attention. Process monitoring can be supported by applying a model similar to the current model for determining treatment requirements, but in reverse order. With such a model, a prediction of the product state can be derived for all time slices of the recipe, using the initial state of the product and the treatment recipe. If, during the treatment, deviations from the predictions appear, re-planning of the recipe can be considered. The PTSS will generate alarms to alert the treatment expert. With this system the expert should be able to observe problems in the treatment process, enabling timely correction. The main obstacle is related to the problem of noisy mini-treatments as discussed in Section 3.4. Further analysis of our experimental data, and possibly additional experiments, are required to improve the neural net models for monitoring.

Batch	RME	Recipe Planner
		Relative Deviation Requirement vs. Recipe
1	.0061	.0330
2	.2997	.2633
3	.0746	.0533
4	.0370	.0667
5	.0174	.0067
6	.0072	.0214
7	.0783	.0842
8	.0463	.0397
9	.0039	.0024
10	.1006	.1204

Table 3.2 Relative performance of PTSS recipe planning.

Another interesting extension concerns the scope of the PTSS. In its current design, product treatment is considered as an isolated process for one specific product. Converting the system

to other products is relatively easy. Product and process knowledge is stored in a few product specific sections of the program: the neural network for initial state assessment, the module to derive the constraint satisfaction formulation for the recipe planning, and the characteristics of the treatment facilities are represented in the user interface section. Including information of product history and product destination may be more complicated. It may however be worthwhile to consider detailed information about the product history when designing a recipe. Similarly, information on the destination of the product, in terms of conditions and required quality at certain time points, may be used to design the recipe. We are currently involved in a project that for a large set of products studies the possibilities to include such additional information about logistic history and destination in the planning of the recipes. This project concerns a large number of products. As in the PTSS, the knowledge base will be a combination of knowledge from literature, knowledge obtained from experiments and human expertise.

The PTSS does contain generic properties, as can be concluded after comparison with other systems. In particular Aarts (1992) describes an approach of planning mashing profiles which supports a similar process structure. The main rational for his approach, as in the case of the PTSS, is to solve problems caused by variable quality of raw materials, in an environment where standard recipes are normally applied. His system, however, profits from the engineered production system in that domain. As a result, his planner has a detailed analysis of raw materials available at planning time, and profits from deep process knowledge in composing the recipe. This has led to the use of case-based techniques in later work (Aarts 1995).

3.7. Conclusion

This chapter presents the PTSS, a system that supports experts in planning treatment processes. The PTSS introduces opportunistic recipe planning in a domain that previously had to rely on standard recipes. Moreover, it comes with a method to objectively assess the initial state of the products. This helps experts to cope with inherent (intra- and inter-batch) variances in product attributes, thereby obtaining a predictable and more constant quality. The system is based on task analyses of experts, and uses learning and constraint satisfaction modules for handling the variances. The applied C4.5 learning module provides experts with a comprehensible procedure for selecting representative products. The neural network module is used to transform batch characteristics and state measurements into a recipe specification. The constraint satisfaction module contains a combination of current expertise and results form physiological research.

The first results with the PTSS are encouraging. Moreover, several options for improvement and extension of the system will be pursued in the near future. The most interesting option is to formulate a generic framework for recipe planning in domains with product-inherent quality variances.

4. Generalized and Instance-Specific Modeling For Biological Systems

This chapter has been published as F. Verdenius & J. Broeze, Generalized And Instance Specific Modeling For Biological Systems, *Environmental Modeling & Software, 14*, 1999, 339-348

Abstract

Biological, ecological and environmental systems are difficult to model. Due to the complexity of the relevant entities their behavior is hard to capture. Moreover, due to the evolutionary behavior of the biological entities and assumption-dependent non-linearities, models are valid in only a limited condition range and time frame. This chapter introduces the concepts of generalized and instance-specific models, and their relevance for biological, ecological and environmental systems. For each concept, a modeling approach from the field of AI is introduced. The concepts are illustrated in two modeling applications in the wastewater domain, one for opportunistic modeling and one for resource allocation.

4.1. Introduction

Biological, ecological and environmental (BEE) processes are becoming more and more important in policy– and decision making. The Kyoto summit has defined strategic targets to be realized by countries world-wide. On a smaller scale, regional officials have to predict, monitor and control environmental phenomena such as smog, manure, eutrophication and acid rain on local ecosystems, while on micro-scale, such as waste- and surface water systems, operators have to maintain operational set-points by controlling complex processes. Moreover, biological processes are more and more applied on an industrial scale.

Decision- and policy quality strongly depend on model validity. The complexity of BEE systems complicates the development of reliable models for use in the real world. This often leads to the deployment of models that are simplified and stripped versions of the models that would be required. Artificial Intelligence techniques can be used to improve the modeling of BEE processes.

In section 4.2 we explore problems in traditional modeling approaches for the complex systems that are found in natural and artificial biological systems. Then, in Section 4.3, we discuss differences between generalized and instance-specific models, and present two Artificial Intelligence approaches for this kind of problems. Section 4.4 presents for both types of approaches an example, as implemented in an activated sludge wastewater system. Section 4.5 concludes by discussing these approaches, and generalizing the concepts to biological, ecological and environmental systems in general.

4.2. Context

Modeling BEE systems introduces a number of specific problems. A model is a representation of a (real-world) entity that mimics some aspects of the behavior of the referent entity. Realistic systems consist of many individual entities of different kinds, each with their own behavior. Growth, decline and extinction behavior of organisms, as well as the behavior of large numbers of biological processes, depend on many variables that show autonomous dynamics. Even for describing simple behavioral aspects of such systems,

complex (sets of) differential equations are required. An illustration of this can be found in the modeling of the keeping quality of agricultural produce (Tijskens & Polderdijk, 1996).

The community of entities in compound systems shows substantial interaction and interdependence. The compound behavior of such systems is more than the sum of the behaviors of the contributing sub-systems. A simple example of such a system is the daisy world (Lovelock, 1988). In the realm of agriculture, the problems of controlling the keeping quality of mixed loads of produce are the result of this phenomenon (e.g. Paull, 1993). An additional problem is that many systems are not constant over time.

The standard approach to modeling of systems is to map system behavior on (partial) theories that are available for the domain. The first step is to decompose the complex systems into a number of relevant processes in the system, where relevance is expressed in terms of the modeling goals. For the identified processes, sub-models should be available. For describing behavioral aspects of biological systems, generalized theories that give a global description of the dynamical behavior may often be available. Alternatively, sub-models should be developed. In the second step, these sub-models are grouped into a model that represents the entire system. Finally, these sub-models are calibrated, to fit observed behavior of the system.

In practice, modelers often make use of a library of parameterized sub-models that, under the assumption that the pre-conditions of these sub-models are satisfied, represent the behavior of sub-systems. The modeling process is divided in two steps: (1) designing a configuration of sub-models that corresponds as best as is possible with the actual configuration of the system of concern and (2) fitting this model on observed (or assumed) system behavior by means of parameter calibration. Formulating and maintaining such models is both time- and labor consuming. As a result, a model is generated once and applied over a substantial period of time, and for a number of different aspects.

This modeling approach aims to deliver valid models to be delivered. However, the modeling approach suffers from at least four problems:

1. Expertise requirement: deployment of the modeling approach does not only assume profound expertise in the biological domain, but it also requires a thorough understanding of modeling tools and approaches. Often model builders are experts in the referent domain, but the modeling expertise is lacking.
2. Model focus: different application goals may require different models. Application goals may include yield optimization, diagnosis, etc.
3. Assumption violation: the sub-models in the libraries are valid under the assumption that pre-conditions hold. Mostly, these sub-models are only valid under *normal operational conditions*. Models for extreme conditions are sparsely available. When pre-conditions are violated, a model looses validity.
4. System drift: due to autonomous drift of the processes, calibrated model parameters lose validity over time.

Problem number 2 indicates that *different models would be needed* for studying different aspects of the same process. The last two problems make clear that models should be *updated regularly* to keep them up-to-date with the actual system. However, if recognized at all, the four points above are considered impossible to address in practice. The result is that many models that are used for decision and policy making are not accurate representations of the systems they are supposed to model.

Consequently, we signal two problems in the use of BEE models. First, for many problems, models are applied that are not well suited for that particular problem. Second, due to the inflexibility of modeling approaches, a number of problems that could be modeled on

theoretical grounds, are not modeled because of practical limitations. This situation immediately undermines the quality of decision making on both macro and micro level.

4.3. AI Modeling Techniques

Models help to optimize the decision quality in policy and decision making. The models are typically used for simulation and optimization. Models may play different roles, depending on the way they are modeled. For this chapter, the difference between a *generalized model* and an *instance-specific model* is important. The former employs techniques known as *compositional modeling* (Section 4.3.1), the latter employs *case-based reasoning* (Section 4.3.2).

4.3.1. Compositional Modeling

The goal of the compositional modeling approach (Falkenhainer & Forbus, 1991; Sloof & Simons, 1994) is to compose a model for a given dynamic system using a library of pre-defined model fragments, which is based on a generalized theory. Such a model is dedicated to user-defined queries about that system. Model libraries with pre-defined model-fragments are quite common, usually dedicated to one application domain. An example of such a model library, in the area of wastewater treatment, is the SIMBA[14] system. In this library, model fragments represent the behavior of specific plant components.

Classical versions of such model libraries are static; they do not support user-modification of the prototype sub-models or definition of new sub-models. Hence, in each model generated from the library, the user should define appropriate parameter settings. More recent model libraries allow for a more flexible management of reusable models. Breunese et al. (1998) present a concept in which the user can define new model fragments and make them available for reuse in other applications. Through the introduction of these tools, the model generation task has become more flexible: model fragments can simply be replaced by other (more or less detailed) models in accordance with functional requirements.

Compositional modeling makes use of this flexible approach in the form of a dynamic modeling approach. No unique model for the considered system is developed. A query represents a specific viewpoint on a system. The generated model satisfies the viewpoint issued by the user. As such, the method is very suitable for complex systems that cannot be properly represented with one unique and complete model. Thus, compositional modeling uses knowledge from theory as well as information on the process status, as represented in Figure 4.1. The following four stages are distinguished in compositional modeling:

1. *Query analysis* - The query about the system is analyzed to find a set of relevant objects and attributes as found in the physical system that have to be included in the model.
2. *Object expansion* - The smallest physical system is identified that contains all objects found in the query.

[14] SIMBA is a trademark of the Institut für Automation und Kommunikation Magdeburg, Barleben, Germany.

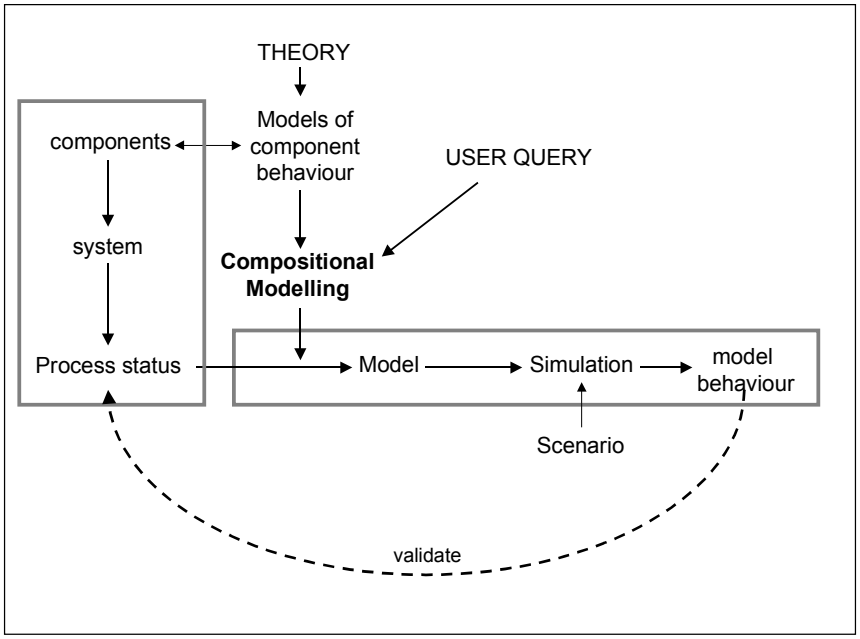


Figure 4.1. The role of compositional modeling in generalized modeling

3. *Candidate completion* - Here, it is decided how to model each object in the system identified during object expansion. In this stage, one or more candidate models may be found for the given query.
4. *Candidate evaluation and selection* - The simplest model is selected from the set of candidate models.

The compositional modeling task is a generic task and has been implemented for amongst others quality change models of agricultural produce (Sloof & Simons, 1994).

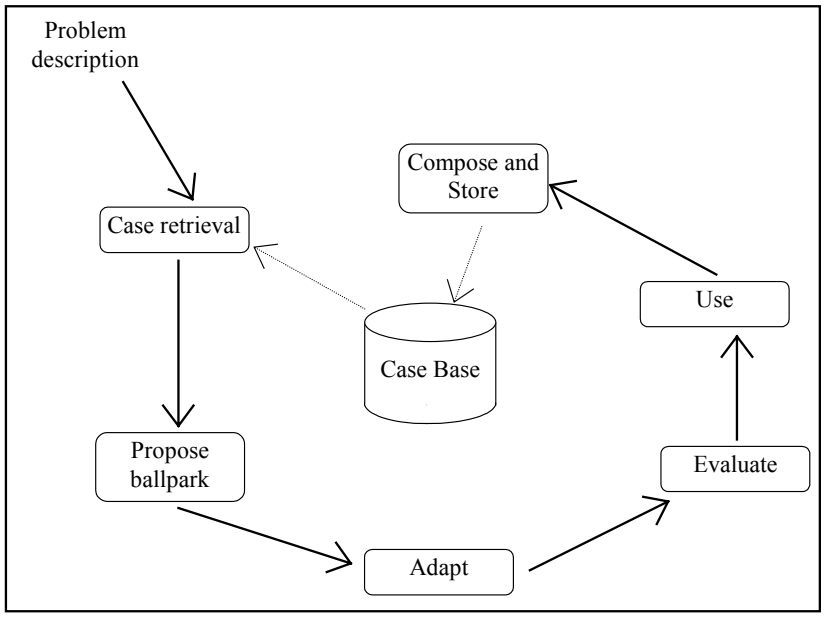


Figure 4.2. Overview of case-based reasoning process

4.3.2. Case-Based Reasoning

Case-Based reasoning (Kolodner, 1993) is a problem solving strategy originally stemming from psychology. The original observation is that humans solve new problems by recalling one or more similar situations from the past. From this historic situation, the applied solutions are transferred to the new problem, and modified to compensate for the differences between the historic and the current situation.

In case-based reasoning, this psychological strategy is translated into a problem solving strategy for use in computer software. Problems to solve are described as *cases* in a standardized format. The system memory is implemented in the *case base*, which is a database containing historic problems, solutions and their realized performance.

The problem solving process (Figure 4.2) starts with a well-defined problem description. This leads to a search in the case base for a historic case that resembles as much as possible the current problem. From this historic case the solution is retrieved, and copied as a ballpark solution into the new case. Then, a detailed comparison between the historic case and the problem description is made, leading to (potential) changes in the proposed solution. Potentially, the new solution is evaluated, and re-adapted on the basis of the evaluation results. Eventually, the adapted case is used as problem solution. This leads to new information on system behavior, which is added to the case base as a new case, incorporating both the proposed solution and the achieved result.

When realizing a case-based reasoning system, a number of implementation decisions have to be made. Kolodner (1993) and Aamodt & Plaza (1994) identify a number of issues to consider:

1. *Case representation and case indexing* - what information is needed in a case, and how are cases organized in such a way that effective and meaningful retrieval is achieved?
2. *Case adaptation* - what knowledge is used to adapt the ballpark solution?

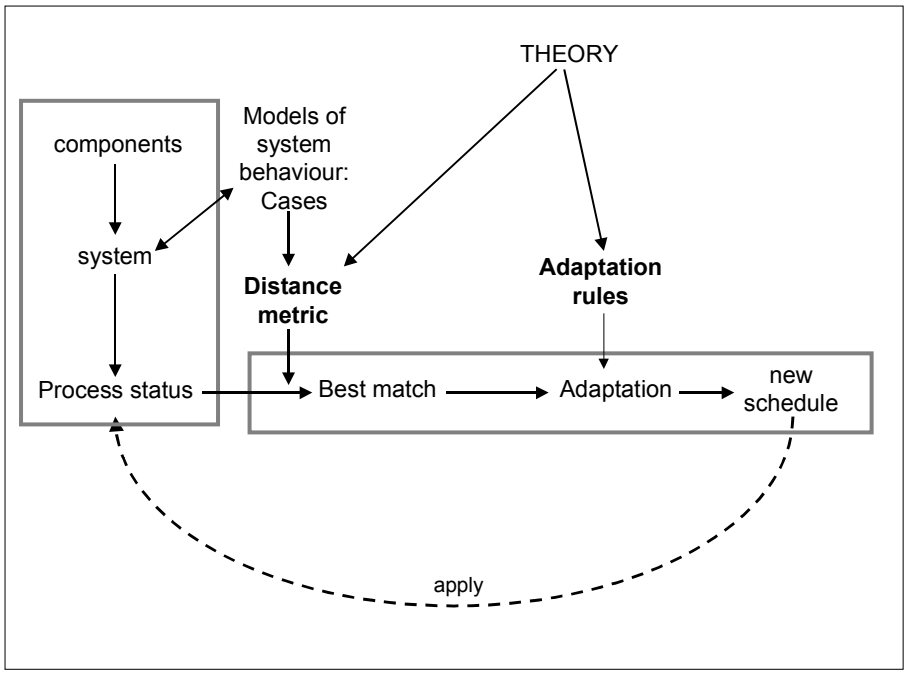


Figure 4.3. The role of case-based reasoning in instance specific modeling

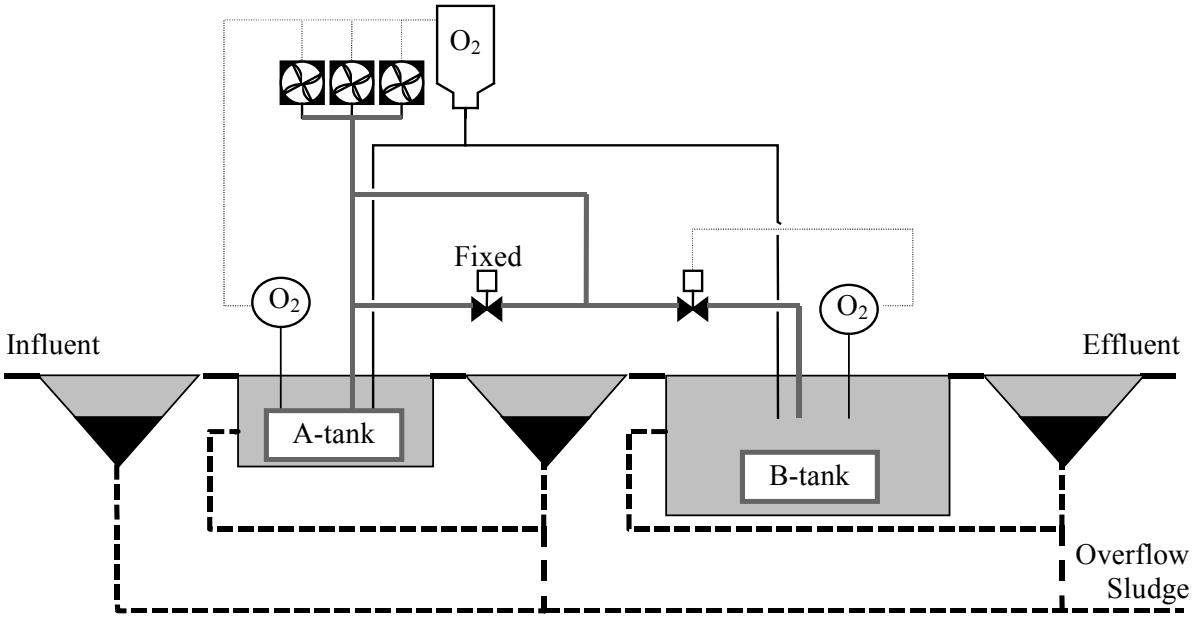


Figure 4.4. Part of the configuration of the BEB Seehausen plant

3. *Case evaluation and critique* - what evaluation criteria can be used, and how to process the evaluation result?
4. *Retain methods* - how to store cases, and which cases are stored?

As illustrated with Figure 4.3, the role of case-based reasoning shows some similarities with compositional modeling (Figure 4.1). Differences are the role of the domain theory and the level of abstraction of the delivered solution: case-based reasoning delivers a system specific solution.

4.4. Case Studies

This section presents two case studies in the domain of wastewater treatment. In Section 4.4.1 we give a short introduction of this domain. Section 4.4.2 presents an overview of the WaterCIME project. In this project, the modeling concepts presented in this chapter have been developed and field-tested in two applications. These applications are presented in Sections 4.4.3 and 4.4.4.

4.4.1. Wastewater Treatment

Wastewater treatment, the purification of industrial and communal wastewater has become a major industrial activity. Large wastewater producing industries operate dedicated plants that implement processes dedicated to the wastewater delivered in that industry. Communal systems aim at processing the mix of wastewater produced by households and industries.

Biological treatment is commonly used for wastewater plants, mostly complemented with physical and chemical purifying components. Biological systems deploy micro-organisms that adsorb and convert complex biological and chemical waste components such as phosphates, nitrates and carbon-hydrates into environmentally less harmful substances, e.g. CO_2, H_2O and bio-mass. A major technique for biological wastewater treatment is the activated sludge process. In this process the biomass consists of oxygen consuming (aerobic) micro-organisms. The required oxygenous conditions are realized by injecting large amounts of air into the wastewater. A typical example of a biological treatment system is displayed in

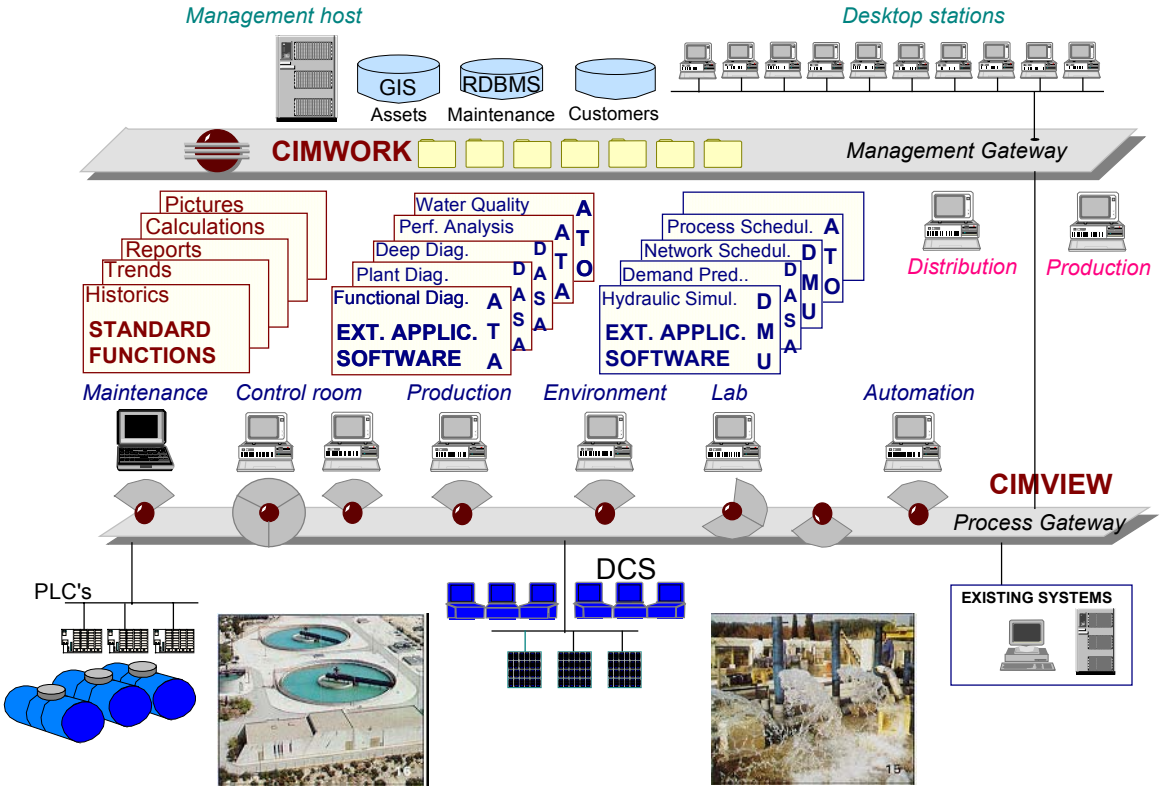


Figure 4.5. General Overview of the WaterCIME architecture. Data from the primary processes are linked to the process gateway. Aggregated data is managed at the Management gateway. Both process data and aggregated data can be accesses by all applications, on the basis of transparent data definition. Both layers are open for other modules to exchange data.

Figure 4.4. This figure represents an A-B reactor type, where A stands for Adsorption and B for Belebung (= aeration). In this system, commonly encountered in Germany, net-shaped organisms filter the coarse waste particles out of the water in the A-phase. Then dissolved waste is transformed by smaller micro-organisms in the B-phase.

Activated sludge plants are operated semi-automatically. Controlling the plant under normal conditions may apply three controls: aeration capacity, return sludge, and sludge removal. Standard configurations in current situations include SCADA systems for real-time monitoring of process conditions. However, these data prove insufficient for optimal control. Real-time measurement of oxygen demand is not possible. Reliable prediction of oxygen demand is required to optimize aeration. Aeration management therefore relies on the availability of reliable prediction of oxygen demand.

Under abnormal conditions, the major problems are to correctly diagnose the problems, and to identify the proper repair strategy for restoring the normal operating conditions. Faulty process states can originate from many different reasons: toxification of biomass, insufficient aeration capacity, malfunctioning of sensors, mechanical problems in the plant and many more. Fast and correct diagnosis is essential to ensure optimal process operation. However, process control systems only give partial information on the process state. Characterizing influent, even at a simple level, is a time-consuming process, let alone a full chemical and biological analysis. Advanced tools are being developed to support both operational control and process diagnosis.

Mathematical models of the wastewater processes have gained popularity. Over several years, a number of separate model libraries have been developed. Under co-ordination of the International Association on Water Quality (IAWQ) a unified library of mathematical models has been developed. This library is now widely accepted, and contains sub-models for many components that can occur in activated sludge plants. It is also the basis of the SIMBA model library.

A major goal of application of mathematical models is to support system design. Proper plant dimensions can be determined by performing simulations based on influent requirements (volume and characterization of waste components). Moreover, models are used for educational and research purposes, and for studying control strategies (e.g. PID controllers). As yet, operational exploitation of these models remains limited. Main reasons for this are long response times for real-time usage, model calibration and the required effort to keep models up-to-date.

4.4.2. The WaterCIME project

The case studies in this chapter are drawn from the WaterCIME project (ESPRIT project no. 8399). WaterCIME aims to improve water management information systems by developing an open software framework, covering both wastewater and clean water treatment, transport and storage. It adapts Computer Integrated Manufacturing and Engineering (CIME) concepts, which were originally developed for manufacturing industries. They provide techniques for looking at a company's information, decision-making and engineering systems, and planning their close relation.

An overview of the WaterCIME system is given in Figure 4.5. The resulting architecture can be subdivided in four sub-systems:

- The process sub-system consists of all sensors, actuators, PLC's and process computers that together provide operational control functionality (depicted below the process gateway in Figure 4.5).
- The CIME layer consists of a process gateway, collecting data from and to the process, and a management gateway, providing aggregated data several levels of management applications. The layer provides open and transparent access to all data available in the system. The concept eliminates multiple versions of databases, reducing the maintenance effort and improving data quality.
- Nine WaterCIME modules, which are modules that realize operational, tactical and strategic functions (depicted between process gateway and management gateway in Figure 4.5).
- Modules provided by other suppliers, e.g. GIS, simulation software, Maintenance modules, financial and customer administration (depicted above the management gateway in Figure 4.5).

Several advanced modules are developed that make use of the open structure of the framework to illustrate the added value of such a platform. The modules presented here were both installed and tested at the Seehausen wastewater plant of the *Bremer Entsörgungs Betriebe*, in Bremen, Germany. The systems proved their functionality during the tests.

4.4.3. Generalized modeling with WQSM

The Water Quality Simulation Module (WQSM) serves as a prototype system for problem-specific model generation. It creates generalized models for the wastewater treatment plant under consideration. In the current setting, the generalized theory defined by IAWQ is deployed (Henze et al., 1995). Models are generated in a format suitable for existing

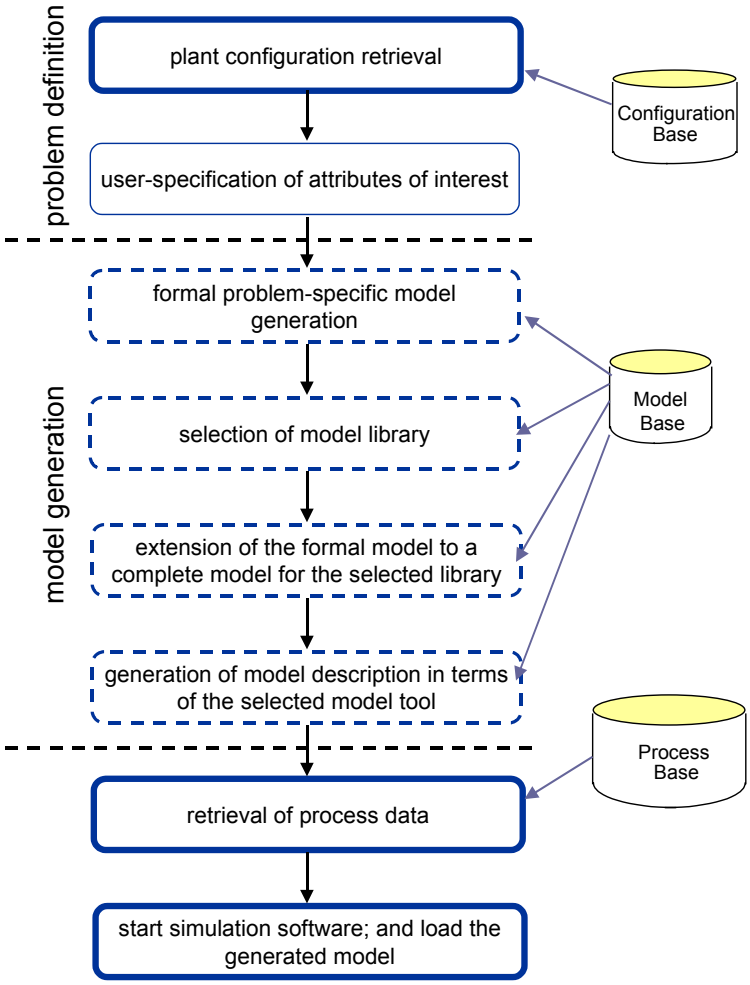


Figure 4.6 Subtasks in WQSM

simulation software (such as SIMBA or EnviroPro[15]). The modeling technique implemented in WQSM (Broeze et al., 1997) is based on the compositional modeling approach (Section 4.3.2).

WQSM largely utilizes data already available in the WaterCIME environment. One of the databases in the WaterCIME environment is the configuration database, describing the up-to-date plant and process configuration. In case of a reconfiguration or maintenance, the configuration database is updated by a maintenance manager. By using these data, WQSM automatically creates a backbone structure for the model to be created. This largely facilitates the modeling task.

Furthermore, historic and recent process data are available. WQSM has direct access to these data, thus largely facilitating practical simulation.

The modeling task in WQSM consists of the following sub-tasks (see also Figure 4.3):

[15] EnviroPro is a trademark of Intelligen.

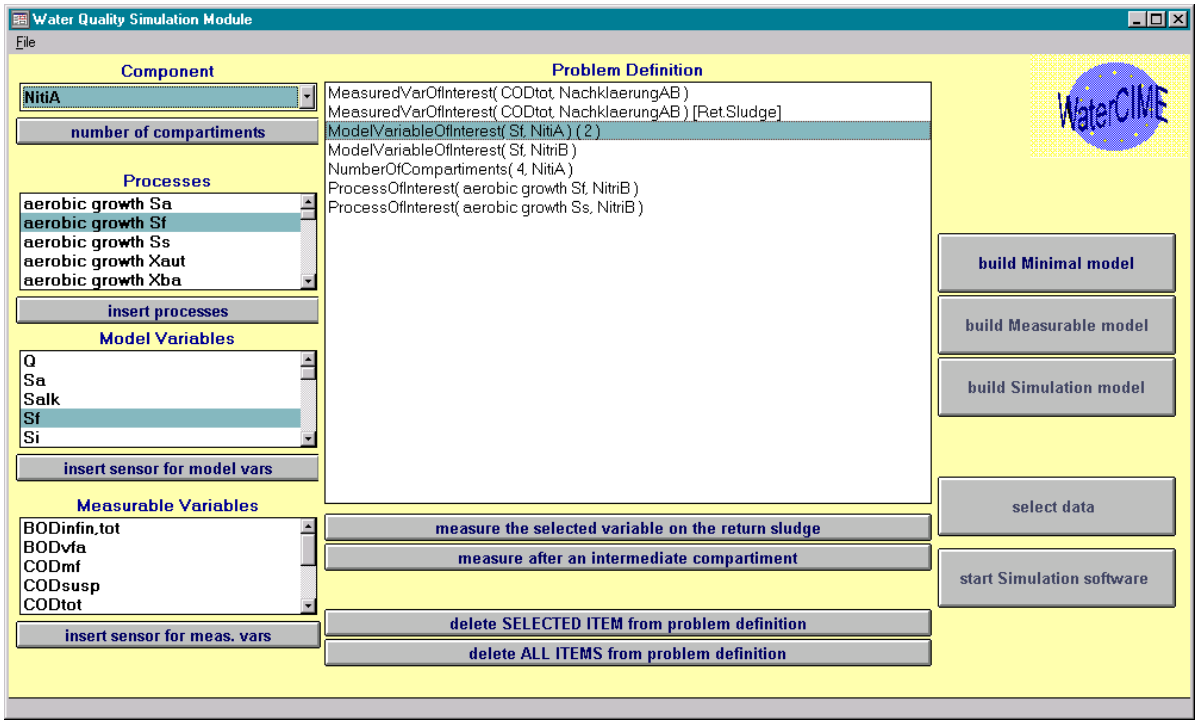


Figure 4.7. User Interface of WQSM

1. *Plant configuration retrieval*. WQSM uses a pre-defined plant configuration, here available in the configuration database within the WaterCIME environment. The plant layout, as described in the configuration database, serves as a backbone for the considered system during the modeling task.
2. *Specification of a problem of interest*. Queries about the system include a user-selection of processes and variables in components of interest.
3. *Formal model generation*. WQSM defines a formal model that includes all attributes of interest and the processes that influence those processes and variables.
4. *Selection of model library*. WQSM selects the simplest model library that can represent

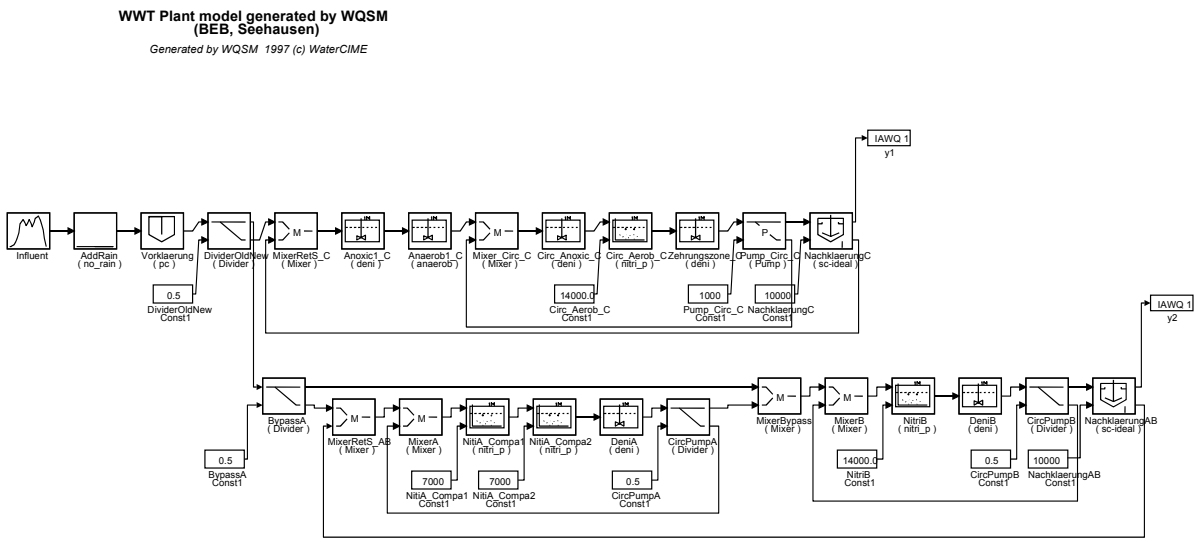


Figure 4.8. SIMBA model generated by WQSM for the Seehausen wastewater plant.

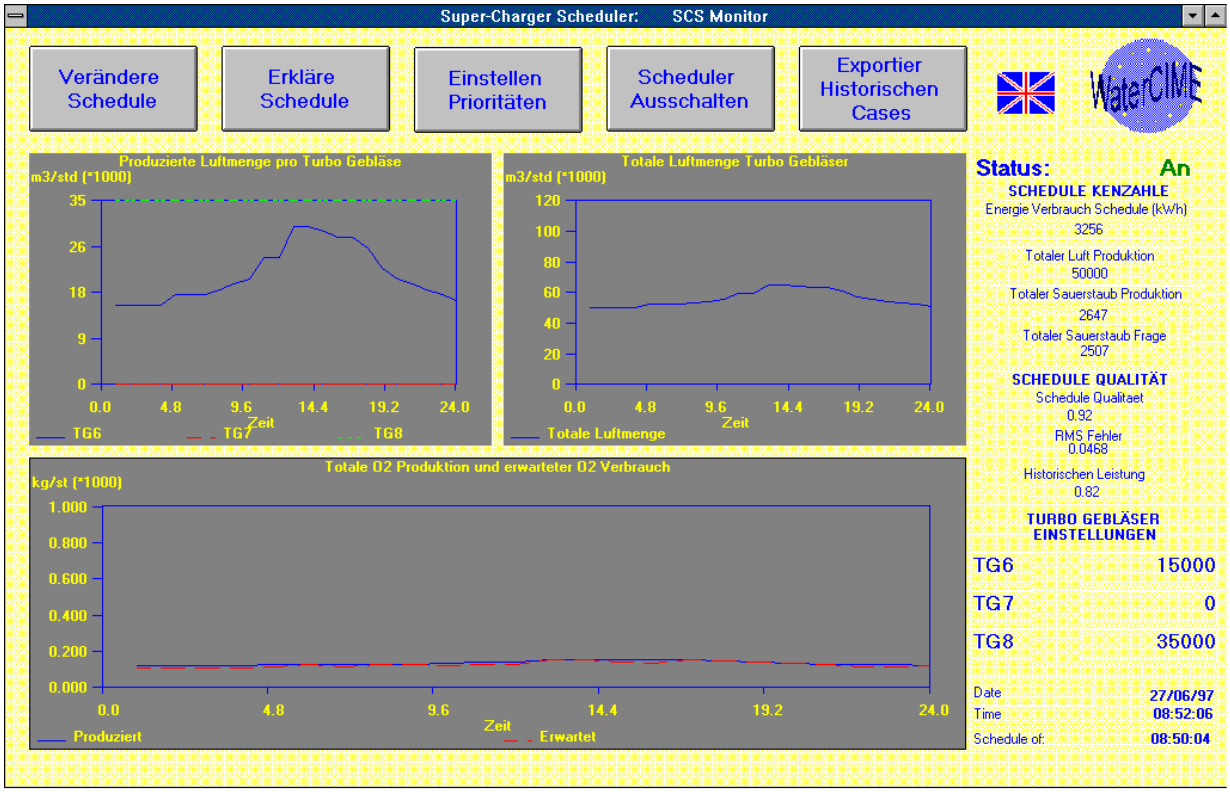


Figure 4.9. User-interface of the SCS.

all attributes gathered this way (in the present version of WQSM the SIMBA libraries ASM-1 and ASM-2 can be addressed).

5. *Extension of the model to a complete model for the selected model library*.
6. *Translation of the model to a format suitable for the selected model library.* The final modeling step is the representation of each component by one or more sub-models from the model library. WQSM chooses the simplest sub-model (the sub-model that is least expensive from a computational point of view) that features the attributes.
7. *Retrieval of process data*. WQSM has access to process data through the WaterCIME environment.. These data are used for calibration and actual influent (load) descriptions. Recent developments in the applied simulation environment allow for automatic calibration of the model with these data.
8. *Start the simulation software.*

Sub-tasks 1 and 2 represent stage 1 in the compositional modeling process (Section 4.3.1). Sub-task 3 stands for stages 2 and 3, whereas sub-task 4 reflects stage 4 of the compositional modeling process. In sub-tasks 5, 6 and 7 stages 3 and 4 are reapplied to a larger system.

The sub-tasks in WQSM are accessed through the prototype user-interface (Figure 4.7). The buttons in the left and middle column are used for user-specification of attributes of interest. The buttons in the right column start the model composition, retrieval of process data from the WaterCIME process database, and starting the simulation tool SIMBA.

In comparison to classical modeling approaches, WQSM largely simplifies modeling of wastewater treatment systems. The task of manually creating a model structure and appropriate flowsheets, with components that correctly represent processes and variables of interest, is taken over from the user. Furthermore, realistic process data can be easily

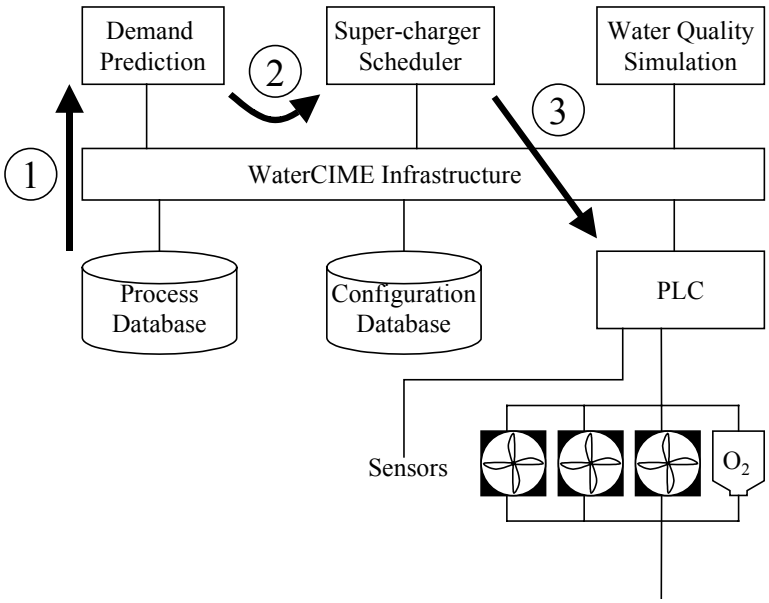


Figure 4.10. Overview of the WaterCIME subsystem. A demand prediction module delivers a short-term load prediction. The SCS module converts this to a 2-hour schedule for the aeration system.

accessed. As a result, creating problem-specific simulation models is largely simplified; modeling becomes available for more purposes and for a larger group of users.

Tests have shown that with WQSM adequate simulation models can be generated within a few minutes. This is in large contrast with the current practice; usually a number of months are required for creating an appropriate model structure, selecting suitable sub-models from a model library, data acquisition and calibration. During the tests, we have generated models with WQSM in SIMBA that are functionally equivalent to expert engineered SIMBA models. An example plant model is shown in Figure 4.8.

WQSM can typically be used for analyzing

- unexpected plant behavior, such as a change in process conditions in a reactor tank (aerobic, anoxic or anaerobic), and
- a varying degree of refinement of representation of reactor tanks with a prop flow (choosing a varying number of compartments in the model).

In future, links to model-based diagnosis tools are foreseen.

4.4.4. Instance-specific modeling with SCS

The Super-Charger Scheduler (SCS; Verdenius & Broeze, 1997) is a prototype system for process scheduling in the water domain. The system has been tested in a section of the wastewater plant Seehausen of the Bremer Entsörgungs Betriebe (BEB). The wastewater process is of the activated sludge type. It is aerated by three super-chargers (15,000 to 35,000 m^3), that operate in a closed control loop on the basis of measured oxygen level in the tanks.

In controlling the aeration system, two typical problems occur:

1. Small fluctuations in the Chemical Oxygen Demand (COD) may result in an oscillatory on-and-off switching of superchargers. This is energy inefficient, and may generate additional maintenance needs.

2. COD peaks or falls may result in a lack or overflow of aeration capacity.

Traditionally, the system is controlled in a closed control setting, where the resources are automatically allocated on the basis of oxygen measurements in the process. The introduction of the WaterCIME environment, with a demand prediction module, enables active scheduling of the resources in an open control system (Figure 4.10). Existing mathematical models are inadequate to solve this type of control problem. Moreover, as controlling the aeration is a new feature, experts have no knowledge on how to perform this task. Case-based reasoning is introduced as a solution to this allocation problem. A schematic overview of the scheduling system is provided in Figure 4.11.

Locating cases in a case base that stores more than 100,000 cases a year can only work when cases are properly indexed. Good index features can not be identified in advance, due to the novelty of the scheduling task. Moreover, over time the set of index features may change due to system drift. This introduces the need for an indexing technique that extracts without supervision adequate indexing features. A self organising map (SOM; Kohonen, 1995) is used as indexing mechanism. Regularly, the case base is re-processed by the SOM training algorithm (the *calibrate*-arrow in Figure 4.11). All cases are projected on a 2 dimensional grid. Case distance is measured with a weighted Euclidean measure. The SOM index delivers a number of potential matches. A detailed comparison on key features, indicated by experts, determines the ultimate best match.

Cases are represented as a grouped list of features. Feature groups concern

- *Case identification* – containing time, date, weather, day status (weekend, holiday, normal working day).
- *Plant status* - containing the actual status of the plant, including oxygen level, pH, influent flow, etc.
- *COD demand* – containing the expected (new case to schedule) or encountered (historic case) COD demand in the time frame that the schedule covers.
- *Schedule*- containing the super-charger setting over time.
- *Schedule result* – containing the development over time of the schedule performance features (historic schedule) or the required performance (new case to schedule).

This best matching case serves as the ballpark solution. It is adapted on the basis of differences between the historic problem and the newly encountered problem. The adaptations are inspired by the IAWQ model for activated sludge processes (Henze et al., 1995). In the current system cases are not subject to a critique during the planning phase. The inclusion of the solution results in the case base serves as a means to learn the behavior of the system.

4.5. Conclusions

Ecological systems require flexible approaches for modeling and control to cope with typical aspects of biological processes:

- complexity,
- process dynamics,
- process evolution, and
- the possibility to have different views on one process.

Existing modeling and control technology is not optimally suited to cope with the typical problems in this domain. Two typical problems where AI technology can improve the functionality of existing software are discussed in this chapter: modeling of biological

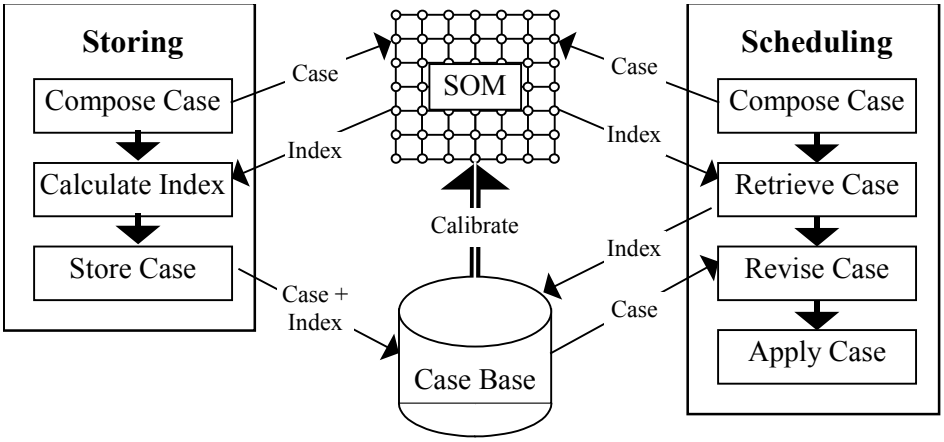


Figure 4.11. Role of Case-based Reasoning and Self Organising Maps in the Super-charger Scheduler.

processes by means of compositional modeling and on line scheduling for resource allocation with case-based reasoning. The result of the compositional modeling approach is a model that describes the behavior of a system as represented in process data and that is based on theoretical domain models. Case-based reasoning employs extensive descriptions of previous situations, in terms of process data, in order to justify operational decisions. This approach does not rely on theoretical domain knowledge. Instead, the system improves its behavior over time by acquiring new data on process behavior. Moreover, case-based reasoning can learn from its own past behavior in order to fine-tune its case adaptation knowledge. Two applications in wastewater management form an illustration of the approaches. These approaches are applicable in many related domains. The principles that play a role in wastewater treatment also occur in many other systems.

A remaining drawback with both approaches is that they pose some constraints on the application. In order to successfully compose a model, a reliable model library covering the relevant system components and behavior types is required. In many biological domains such a model base is not, or only partially, available. On the other hand, in case-based reasoning the available domain knowledge is only implicitly reflected in the resulting solution. An option is to combine these two approaches. Verdenius & Top (1998) suggest to match, in a case-based approach, system behavior onto the behavior of known models. Based on theoretical insight and system behavior, the best matching model structure can be adapted to optimally fit the perceived behavior, probably resulting in a new model. In this matching, cross-domain borders (e.g. mechanical, biochemical, electromechanical) can be overcome by representing models on the level of their structure (i.e. bond graph), abstracting from the actual domain.

We aim to further refine and apply the cases presented in this chapter. The development of a tool combining compositional modeling with automatic diagnosis is foreseen. The results of the examples are encouraging for these new developments.

5. The MEDIA Model

This chapter is based on: F. Verdenius & R. Engels, A Process Model for Developing Inductive Applications, in: W. Daelemans (ed.), *Proceedings of Benelearn 1997*, Tilburg, pp. 119-128

Abstract

A growing interest in real-world applications of ML techniques brings the need for methodological support for applying them. A number of methodologies for applying machine learning techniques that are available in the literature have been discussed in previous chapters. After shortly summarizing these approaches, we propose a model for structuring projects that use ML techniques as an add-on to existing approaches. The model is part of a comprehensive approach to support the application of machine learning techniques in various settings, and helps to plan projects where such techniques are involved. Chapter 2 divided the ML application process in three levels: designing the functionality of the total application, selection of a machine learning technique for indicated sub-tasks, and optimal configuration of the selected technique. In this chapter, the three levels of the ML application process are organized in three separate activities, and the interaction between the three levels is structured.

5.1. Introduction

Machine Learning techniques have become popular tools for solving real world problems. In Chapter 2, a number of references to studies were made that surveyed real world applications. These studies report successful applications, mainly in an applied research context. Some studies, including our own survey, also reveal the degree of use of machine learning techniques to solve industrial problems. With the transition of these techniques from a research environment to industry, the research interest shifts from developing technical improvements towards the structure of the design and the implementation process. An important outcome of the Chapter 2 survey is that practitioners are asking for methodological support. In the literature, this development is reflected in the growing number of reports on process models and methods for Machine Learning (ML) application. Chapter 3 and 4 discuss some problems in industrial problem solving. In both chapters, complex tasks are decomposed into sub-tasks, some of which are implemented in ML components, while others are implemented using knowledge system components that integrate background knowledge. Existing process models do not support these activities.

This chapter develops a model for applying ML applications that is complementary to existing approaches for system development. Section 5.2 briefly summarizes the process of ML development and the process models as proposed in the literature. Much of the existing work in this direction focuses on the technical details of applying the technique, thereby ignoring the design aspects of an ML application. As a result, ML users are mainly supported in solving their problems when they have already completed the major problem-solving step of designing the functionality of the solution. Section 5.3 discusses major problems in the process of ML application and evaluates some of its solutions.

Section 5.4 introduces the *Me*thod for *D*esigning *I*nductive *A*pplications (MEDIA), a model that structures the process of ML application. The application process according to MEDIA starts with the problem statement. It structures the activities upto the level of the individual techniques. Section 5.5 concludes. The goal of this chapter is to present the model as a

structure for the development of ML applications. Specific technical support in activities for system design and technique selection is elaborated in the Chapters 6 and 7.

5.2. The process of ML Application in Literature

Over the years, learning techniques for standard tasks such as classification and clustering have been improved and refined (e.g. Mitchell, 1997). More complex tasks, such as cost sensitive classification (Turney, 1995), have been covered by newly developed techniques. As in many technical disciplines in early stages of development, the machine learning (ML) society progressed by technically improving existing ML techniques.

Consequently, practitioners are nowadays equipped with a multitude of learning techniques, many of them focussed on specific problem types, data set characteristics and application types. Several tool sets (e.g. SPSS Clementine[16]; SAS Enterprise Miner[17]; WEKA, Witten & Frank 2000) provide comprehensive sets of techniques with an interface to open up these techniques for exploration. These tool sets also incorporate non-ML inductive techniques, such as inductive statistics and neural networks, with a similar functionality as the available ML technique.

A number of studies have considered the process of ML application. In its most basic form, it supports a systematic selection of techniques. Weiss & Kulikowski (1991) present an approach for the classification task with the knowledge acquisition aim of extracting a model from data, an application we nowadays would consider data mining. The requirement for this approach is to provide a model to be used for classification. Problem analysis, data collection, operational use nor model maintenance are covered by their approach. The problem is seen as the problem of selecting a suitable technique for providing a specific knowledge model. Their technical toolbox contains four complexity levels of techniques. The levels are processed in order of ascending complexity until a technique is encountered that satisfies the quality requirements. Satisfaction is primarily assessed in terms of model accuracy. The application order of the various techniques is a combination of increasing expressive power and decreasing comprehensibility of the resulting model, which serves as a secondary criterion for satisfaction: the best model performs accurate enough, and is as expressive and comprehensible as required.

With the increasing interest in data mining, knowledge discovery in databases and the application of machine learning in complex software systems, an increasing number of design approaches have been proposed. Some of them have been discussed in the previous chapters. Table 5.1 generalizes these approaches in terms of the project phases that they cover, and overviews the support that these approaches provide per phase.

MLT stands for Machine Learning Toolbox, ESPRIT II project P2154. The toolbox contains about 40 inductive techniques for classification. Craw et al. (1992) and Kodratoff et al. (1994) provide an approach and a tool to support the application of these techniques. The approach, meant for implementation in the MLT Consultant tool, was designed to support users in selecting the proper technique for their problem. Support is offered in decision-tree shaped taxonomies, each taxonomy aiming at a specific design question. Taxonomies exist for items such as *ML application goals* (learning for similarity detection, acquiring knowledge or classifying instances), *Nature of available data* (Incremental, in Batch), *Nature of available background knowledge* (background knowledge is usable, background

[16] http://www.spss.com/clementine

[17] http://www.sas.com/technologies/analytics/datamining/miner/

Phase	MLT	WEKA	B&S	DTI	Fea	A&Z	CRISP
Application analysis			P	B			B
Feasibility analysis				B			
Application design				B			
Data collection					P	P	B
Preprocessing, Feature Selection		P			P	P	P
Technique selection	T	P	P	B	P	P	P
Running technique	T		P	B	P	P	P
Operational use			P	B			B

Table 5.1. The support provided by various approaches. T stands for technical support, P for phasing, and B for both technical and phasing support. The approaches are explained in the text.

knowledge must be used). Every decision structure is used to select a sub-set of potentially suitable techniques.

Garner et al. (1995) describe another approach that is closely connected to the WEKA workbench. Similar to MLT, WEKA contains a number of classification techniques to be used in a data mining process. In comparison with the former approach, the WEKA approach is more oriented towards the process phases, and gives less support in actually taking design and analysis decisions. Moreover, the WEKA approach emphasizes the importance of interaction between the ML analyst and the data providers, who are normally the domain experts. Using WEKA, especially when exploring the data space, influences the requirements of the data provider, and may generate new questions. The requirements actively change *in the course of the ML application process*. In order to manage the explorative redefinition of the project goals, WEKA foresees intensive interaction between the ML analyst and the data provider.

The approach of Brodley & Smyth (1997), listed as B&S in Table 5.1, is not limited to explorative data mining purposes. Their approach extends the scope of the process for applying ML techniques to the process of analyzing the problem environment, not only in terms of data, but also in terms of (what they call) domain specific factors such as application specific and human factors. In the description of their approach, the authors explicitly discuss the important aspects to be analyzed. This method is, as the ones discussed before, specific for classification tasks. Like the WEKA approach, B&S indicate the phases of the process, without providing elaborated design and analysis guidelines.

Such guidelines are included in the DTI approach for realizing neural network applications (DTI, 1994). Problem solving starts with the definition of the application. Neural networks are suitable for a broad range of tasks, amongst others classification, optimization, and prediction of continuous values. The scope of the DTI starts as early as the application analysis. Furthermore, the phases as well as the contents of the design steps are provided in more detail than most methods that were discussed earlier. The organization of the total approach is similar to classical *waterfall approaches* for software development, with clearly indicated phases and milestone products. Some of these phases are extremely detailed, others are more globally defined.

In the area of data mining and knowledge discovery three approaches have been proposed by Adriaans & Zantinge (1996; A&Z in Table 5.1), Fayyad et al (1996; Fea in Table 5.1), and CRISP-DM (Chapman et al., 2000; CRISP in Table 5.1), respectively. Adriaans & Zantinge and Fayyad et al. provide a fairly similar phasing of the data mining and KDD process. Both indicate the steps to take, and justify these steps. Informally, input and output results per

phase are given. Adriaans & Zantinge provide additional descriptions of techniques, and for some of the activities examples of how they can be realized.

A more comprehensive model is developed in the CRISP-DM project (Chapman et al., 2000; CRISP in Table 5.1). Compared to the former two approaches, it also covers the broader context of the application. The CRISP process model distinguishes 6 phases: *Business Understanding*, *Data Understanding*, *Data Preparation*, *Modeling*, *Evaluation* and *Deployment*. For each of the phases, the detailed contents are described in terms of the tasks within each phase, their goals and the contents of the project documentation. For some of the tasks, guidelines are provided in the form of heuristics (e.g. technique selection is supported with selection lists of techniques per problem type).

The available approaches can provide support on 3 different aspects of a project:

- **Technique orientation:** Process models can be oriented towards (a group of) techniques or tools (e.g. DTI, MLT, WEKA), or they can be technique independent (e.g. B&S, Fea, A&Z, CRISP-DM).
- **Application orientation**: Process models can be oriented towards a specific type of application, such as data mining (e.g. Fea, A&S, CRISP-DM), or they can support any application type (e.g. B&S, DTI). In the latter case, analysis and design of the application becomes a more dominant phase.
- **Management orientation**: Process models can focus on phasing and project management (Fea, A&Z), they can provide detailed technical support on how to make design and development decisions of the product life cycle (MLT), or they can combine the two (DTI, CRISP-DM).

5.3. Towards support of ML application design

The previous section described a number of process models for designing ML applications. As can be observed in Table 5.1, none of the available models covers the entire design and implementation process. Only three of the mentioned models provide some technical support in how to take better design decisions. The Chapters 2, 3 and 4 have given indications on the type of improvement that practitioners need. In Chapter 2, survey respondents indicate that they need methodological support: what to do and in which order to end up efficiently and effectively with a working application. Existing approaches, primarily dedicated to data mining, provide process steps and their dependencies. The three application types (explorative data analysis, one-time knowledge acquisition and adaptive system development) face ML experts with similar challenges and problems, as can be seen in Figure 1.1. Approaches that focus on a specific type of application overlook the commonalties. Chapter 3 and 4 illustrate that designing a working system also includes the decomposition of complex tasks into sub-tasks. To each sub-task a knowledge-based method or a induction-based method is assigned.

Another requirement for methodological support that is revealed in Chapter 2 is the need for instruments for technique selection. In many of the reported projects techniques are selected on the basis of improper arguments: custom, personal preference or availability often plays a role.

Consequently, the starting point for the development is the need for a process model that covers the important ML application phases, that explicitly pays attention to the integration of knowledge-based and induction-based system components, and that offers support in design and technique selection. On the other hand, given the availability of existing process models

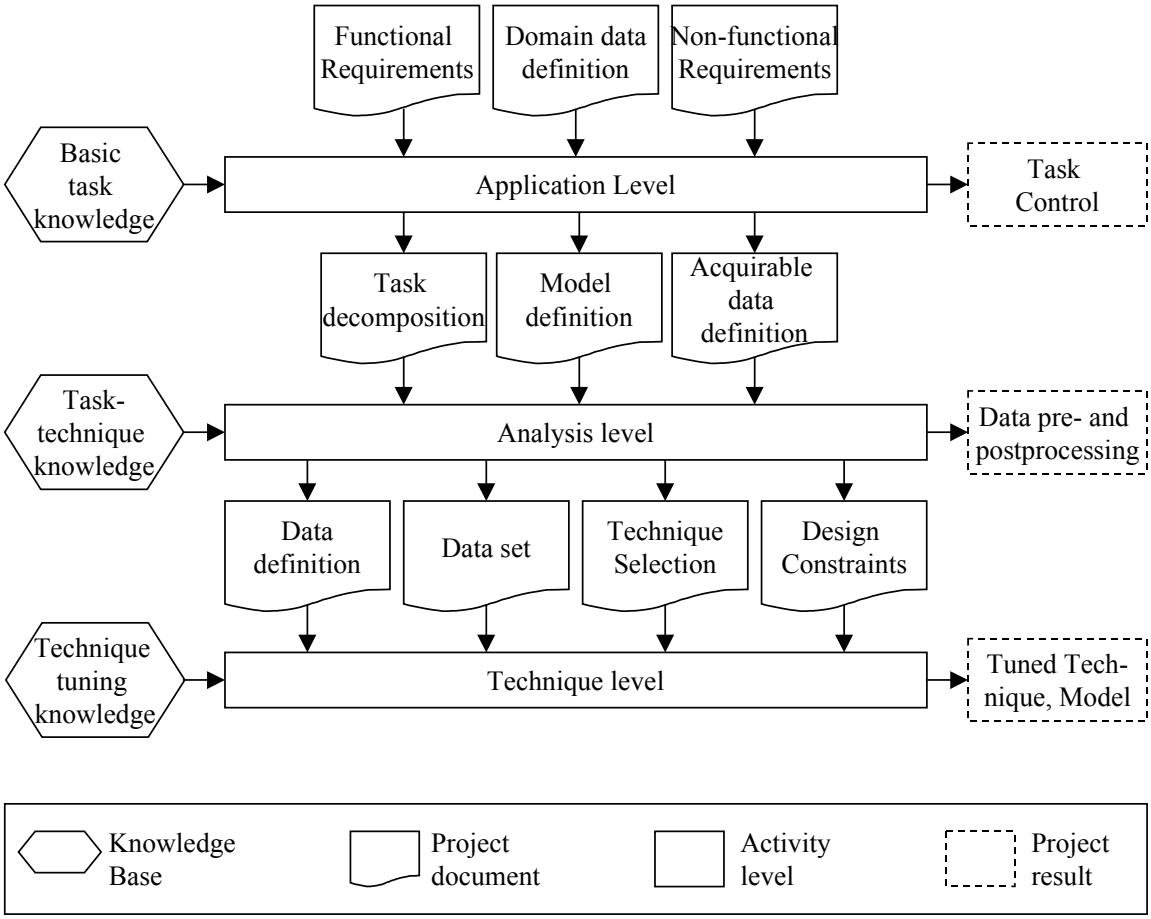


Figure 5.1 An overview of the MEDIA model

and the link with software development, the need for general-purpose phasing and project management support can be omitted, as it can be covered within existing approaches.

5.4. MEDIA

As a methodological contribution of designing induction-based applications we propose the MEDIA model, which provides an application independent activity structure. This model, that can be seen as a ML-design add-on to existing approaches, consists of the three activity levels that are introduced in Chapter 2: the application level, the analysis level and the technique level. These levels and the information flows that connect them are as relevant to an exploratory data mining or knowledge discovery project as they are for the development of a learning software module or an adaptive software system. The model covers the design and development of learning application. Based on specific functional and non-functional requirements that the application/employer imposes on a project and the results of other project activities, a precise list of actions is compiled; Welke et al. (1991) call such a flexible approach Methodology Engineering. The MEDIA model is meant to offer support for the development of applications of machine learning techniques. As such, it aspires to overcome the limitations mentioned in the previous sections.

In Figure 5.1 the major properties of the MEDIA model are shown. The actual design and analysis work is covered in three activity levels: the *application level*, the *analysis level* and the *technique level*. Each level covers a major aspect of ML application design, in ascending order of detail. In the center of each level, the *activity structure* of the model is depicted. On the left hand side, specific *knowledge bases* for each level are listed. The knowledge bases

contain specific design and analysis knowledge that is used to perform the design and analysis tasks. On the right hand side, the *results* of the activities are listed. The outcomes are in the form of models, control algorithms, parameter settings, and software components, which are eventually included in the final application.

The main idea behind this structure is the desire to separate the three stages of *application design*, *task analysis* and *technique operation*. The justification for the use of these levels is similar to the justification for the use of structured knowledge engineering methods such as in CommonKADS (Schreiber et al., 2000), which is primarily separating the *why*, the *what* and the *how*. The why is defined as the overall application functionality, its task decomposition and the control structure. The what is defined as a detailed study of the mapping between data, and the assignment of suitable techniques based on an understanding of the underlying data patterns, and the how is defined as the specification of the low-level technique settings that implement the required functionality.

5.4.1. The Activity Structure

The heart of the MEDIA model is formed by the activity structure as depicted centrally in Figure 5.1. The structure of the model is preliminary introduced in Chapter 2. The main activities performed in the MEDIA model are ordered in three levels. At the very top of the activity model the input into the ML design activities are depicted. The input consists of the *functional requirements*, the *non-functional requirements,* and the *domain data definition* of the embedding application. The functional requirements define the operational task that has to be realized in the learning application. In many cases, this operational task will be more complex than just a learning task. Often a ML project is started with the goal of employing certain knowledge, which is supposed to be implicitly present in the data. Making this knowledge explicit (e.g. in the form of a generated model) forms a sub-task in the overall task decomposition. In the case of the PTSS as described in Chapter 3, the functional requirement is to deliver a prescription for the keeping conditions of a batch of exotic fruits. The recipe takes information on a batch of fruits arriving from abroad as the input, and the resulting recipe should lead the fruit to be in a predefined quality at the end of the storage period. Moreover, the performance of the system in terms of the number of unacceptable fruit batches should be comparable to the performance of human experts.

Non-functional requirements define the constraints the application has to satisfy (e.g. response time, memory resources, and interpretability of the resulting model). For the PTSS, an important restriction is that it operates in interaction with a human expert. Fruit selection and quality measurement is a manual process. Especially the fruit selection criterion has to be comprehensible and sensible for an expert to accept it.

The domain data definition defines the concepts as defined in the embedding application. In the PTSS fruit planning domain, available data includes:

- **Batch data**, such as *origin*, *cultivar* etc.
- **Commercial data**, mainly the required *due date* of the product treatment and the *required quality* after the treatment. It is further assumed that a fixed final quality is delivered for all recipes. Consequently, required quality will not be used.
- **Product data**, being a number of measured values for individual product attributes such as *color*, *shape*, *firmness*, *weight* etc, describing per individual product in a batch various quality aspects at the start of a recipe.
- **Batch history**, such as storage temperature or the concentration of relevant gases such as ethylene, O_2 or CO_2

- **Treatment Recipe**, being setpoints for storage conditions, inlcuding storage temperature, concentrations ethylene, O_2 or CO_2.

Next in hierarchy is the *Application level* where the *why* question is answered. The application level deals with identification of knowledge and data sources, task decomposition and acquisition method assessment. These results in the following:

- *Task decomposition* - tasks are broken down into sub-tasks until a set of simple, formally described tasks is derived. The model acquisition tasks are also part of the task decomposition, if inductive techniques if inductive techniques are available (Engels, 1996). Additionally, the task decomposition describes all steps that are necessary in order to perform the operational task. For the recipe planning process of the PTSS, the task decomposition can be found in Figure 3.3.
- *Knowledge source definition* - describes the models that are introduced in the task decomposition. It connects domain data as input-output relation to sub-tasks. It also links the relations and functions of the task decomposition to the models to the extent that they aa relevant to sub-tasks that involve learning. More formal aspects, such as representation and model contents, are defined at lower levels.
- *Definition of acquirable data* - offers an overview of meta-data (or data characteristics), describing the data that is available for induction in as much detail as possible.

The definitions of knowledge sources and acquirable data are further elaborated in Chapter 6.

At the *Analysis level*, the *how* question is considered. For each task in the task decomposition, a selection of one or more techniques is established that can be expected to perform well on the task at hand. Components are only elaborated when they can be realized with machine learning techniques. For this reason the definition of domain data and the functional as well as the nonfunctional requirements are carefully analyzed and interpreted (see also Engels et al, 1997c). The interpretation combines heuristic and formalizable aspects of learning techniques and their function. In Chapter 7 we develop a tool that performs this selection task. Output at this level includes:

- *Data Definition* – that defines the sub-set of acquirable data items that are used for the inductive step,
- the accompanying *Data-set* containing the data for inducing models,
- the *Technique Selection*, containing a definition of the technique and the input-output design,
- and finally the *Design Constraints*. These are a direct translation of some of the non-functional requirements.

At the *Technique level*, the *what* question is answered. This last level delivers the actual results of the learning process, and generates a model and accompanying software, This level also involves technique design, where exploration of the optimal technique configuration, parameter settings and training set-up are determined.

5.4.2. Results of a Development Cycle

As discussed previously, activities are linked by shared input/output sets. Moreover, each level produces specific results such as parameter settings, software components and control algorithms. The output structures are found in the right hand section of Figure 5.1. At the application level, the main result is a control structure for the sub-tasks: how are the sub-tasks run to implement the required functionality? This task control takes the form of an algorithm

or flow chart, defining which tasks have to be performed in what order to obtain the best result. A control flow defines the order of execution and defines iterations and their stop conditions in the sub-tasks that comprise the task decomposition. Figure 3.2. for instance defines the control flow of the PTSS modules. Additionally a control algorithm could be added, if it were not the case that the user is a major control agent in this specific example.

At the analysis level, data pre- and post processing are defined, so that the selected techniques can actually be run. For instance, a projection function van be delivered that transforms the data-space to a space of lower dimensionality. Also, feature selection, data filters and error correction mechanisms are positioned at the analysis layer.

Finally the technique level may deliver, according to the specific needs of the application only a learned model, possibly complemented by a fully tuned technique (including the set of calibrated technique parameters, a model interpreter and eventually an adaptive learning module).

5.5. Tools for use within MEDIA

The MEDIA model offers a three level structure for designing induction-based applications. Within the model, design concepts and tools on the three development levels are needed that are dedicated to the application of machine learning. Apart from standard software development tools and software methodology, learning systems require additional tools for defining induction-based applications. Historically, there are abundant toolkits offering various machine learning techniques, some prominent examples include MLT (ESPRIT project P2154), Clementine[18] and WEKA (Witten & Frank, 2000). Furthermore, tools for user guidance in DM/KDD (Engels, 1999) and technique selection (e.g. MLT: Craw et al., 1992; Metal: Metal, 2004) have been proposed. In previous chapters the need for additional support on the following aspects has been identified:

1. How to recognize, in an early stage of problem solving, application potential for ML techniques, and how to provide the design concepts for the development process as adequately as possible?
2. How to integrate, from the design phase onwards, an ML solution within an embedding system (i.e. realizing a learning function in a large complex software system, or integrating an adaptive knowledge base in a manually operated process)?
3. How to select a good, if not the optimal, technique to solve a specific problem?
4. How to configure a technique for optimal performance?

All these subjects have been mentioned in at least one or two of the approaches discussed in section 5.2.. In most cases it was brought forward as a subject to pay attention to in a development phase, for instance as the expected content of a project document. However, the actual support in taking design decisions is insufficiently offered. How to identify ML application potential? Or how to select the best technique for a job?

However, within the MEDIA model, support tools are offered in taking design decisions. In Chapter 6 we develop an approach for supporting the first two problems listed above. Based on the overall functional requirements and an analysis of the available data, application potential for ML techniques is identified, and the total system functionality is defined in a task structure that makes effective use of both available data and available knowledge.

[18] http://www.spss.com/clementine

Next, in Chapter 7 we develop an approach to select, for a learning task with available data, the techniques that can expect a good quality model. The approach deploys technique specific knowledge on the relationship between data set characteristics and technique performance.

The tools for task decomposition and technique selection complete the tool sets for the Application level and the Analysis level of the MEDIA model. Problem 4, configuration of a machine learning technique, is not so much a design problem as it is a parameter-tuning problem. As such, it falls outside the scope of this work, although it is reckoned as a major obstacle in delivering an application. Fayyad et al. 1996 refer to this tuning process as the *search method*. For various techniques, there exist a wide range of technique specific methods (e.g. Error back propagation for multi-layer perceptrons, Rumelhart et al, 1986) and general purpose methods (e.g. genetic parameter search and simulated annealing; Chalmers, 1990; Hand et al., 2001) to find the best parameter setting.

5.6. Conclusions

The MEDIA model provides a methodological structure for designing machine learning applications. The design steps it defines are independent of the type of application, such as data mining/knowledge discovery, knowledge acquisition, or the construction of an adaptive system. The MEDIA model serves as a machine learning add-on to existing project management and system design approaches to ensure proper project progress and successful system implementation. To further complement the MEDIA approach at the technique level, knowledge on the configuration of specific techniques and on the tuning of technique specific parameters must be applied. The important contribution of the MEDIA model to the application of machine learning is that it fills the gap between the very low level design support provided by technique developers, and the phasing support provided by the numerous available approaches.

For practitioners, the MEDIA model helps to solve some major problems in efficiently designing systems with learning components. MEDIA helps, at the application level, to integrate a learning component in the embedding functionality. Moreover, it explicitly supports the identification of techniques with and without prospects for implementing the required functionality. In addition, MEDIA separates these two design activities from the actual technique configuration. By distinguishing these levels of design, MEDIA helps system designers to shift focus in the course of the project from the rationale behind the use of it (the *why*) via the selection of techniques (the *what*) towards the technicalities of running a learning technique (the *how*).

6. Planning the Acquisition of Knowledge by Combining Manual and Machine Learning Techniques

This chapter is an updated and extended version of: F. Verdenius & M.W. van Someren (1999), Top-Down Design and Construction of Knowledge-Based Systems with Manual and Inductive Techniques, in: B.R. Gaines, R. Kremer & M. Musen (eds), *Proceedings of the 12th KAW Workshop*, Banff, Canada

ABSTRACT

This chapter presents an approach to the planning of knowledge acquisition for knowledge-based systems. A knowledge-based system is an information system with a separate reasoning mechanism that processes specific domain knowledge in order to provide high-level problem solving performance. The approach combines task decomposition as commonly applied in software engineering and knowledge acquisition with the use of inductive techniques. The decomposition of a compound problem into sub-problems is guided by the expected *acquisition effort* and *resulting quality* of the acquired knowledge. The concepts *effort* and *quality* are commonly encountered as control parameters in project management. Based on the Cocomo software-engineering model for cost estimation, we propose models for cost and effort estimation of knowledge elicitation and machine learning. The method is illustrated with a *rational reconstruction* of the design process for the application for the planning of fruit treatment that was introduced in Chapter 3. A combined elicitation/machine learning solution is compared with pure elicitation and machine learning solutions.

6.1. Introduction

Knowledge acquisition involves the formalization of (human) knowledge that is required to perform a certain task. The implicit assumption is that if humans would perform this task it would require (expert) knowledge. The aim of knowledge acquisition is to provide the knowledge to a knowledge-based system that performs comparable to a human expert. Knowledge acquisition can be considered to consist of two sub-processes: the *structuring* of knowledge in a knowledge base in order to facilitate problem solving behavior, and the *elicitation* of knowledge from sources such as human experts, documents or exemplary data (De Boer, 1994). It is the knowledge base that is seen as the distinguishing component between an information system and a knowledge-based system.

An information system performs relatively simple information processing functions that are unambiguously defined. The total functionality is incorporated in the code. In contrast, the main predefined functionality of a knowledge-based system is the inference engine. The facts, rules and common sense patterns that are evaluated by the inference engine are collected in the knowledge base. It contains specific domain knowledge that is required by the inference engine to exhibit the high-level problem solving performance (cf. Guida & Tasso 1994).

Software engineering is concerned with the theories, methods and tools which are needed to develop software for computers (Sommerville, 1996). It is the discipline of providing effective software efficiently. Within software engineering a multitude of tools has been developed to specify the requirements of software, to design and implement software, to perform software maintenance, and to manage the process of software development.

As stated earlier, knowledge-based systems add a layer on top of information systems. Knowledge engineering is the process of constructing knowledge-based systems, including the selection of applications, organizational modeling and acquisition of knowledge models (Schreiber et al., 1993). When the function of an application has been defined, the knowledge bases have been specified and the knowledge has been acquired, the problem changes from a knowledge acquisition problem into an software engineering problem: How to build the system effectively and efficiently?

The required knowledge can be obtained in different ways. The classical way is to elicit knowledge from one or more sources, human experts or documents, and to formalize this knowledge in an operational language, e.g. using an expert system shell (e.g. De Boer, 1994; Guida & Tasso 1994). In model-based knowledge acquisition, as in CommonKADS, knowledge is modeled independent of the actual implementation, in dedicated knowledge modeling languages (Schreiber et al, 1993; Schreiber et al, 2000; Aben, 1995). In the CommonKADS view, engineering of knowledge-based systems comes on top of the software engineering.

Apart from eliciting knowledge from experts, knowledge can also be obtained from data by means of *machine learning*. Machine learning techniques generalize patterns in data sources that contain examples of the input and output of the problem solving process. A database of relevant facts is collected. For example a bank could collect data on mortgage payment behavior in order to relate the personal financial history of a client to the payment reliability. In a medical application one could relate test results to diagnoses. And in an agricultural application one could relate an expert assessment of external product features to the internal physiological state. Inductive methods extract a knowledge model to classify instances or to predict a numerical value.

Both knowledge elicitation and machine learning have their strengths and weaknesses when it comes to applicability, costs and quality of their results. Knowledge elicitation requires the availability of at least one human expert or documented knowledge source. Expert disagreement reduces the chances for successful elicitation of expertise, and increases the costs. And a clear, well-defined and structured expert task increases the quality of the acquired knowledge. Machine learning, on the other hand, is limited to a number of well defined task types. It requires a database design that contains the data that is necessary to extract the knowledge model for the required task. The size of the database has to correspond to the complexity of the underlying pattern in the data. Given the strength and weaknesses of the two approaches, for many practical problems a pure approach is not optimal. Here we briefly review the two approaches and then discuss an approach that combines knowledge elicitation and machine learning.

6.1.1. Knowledge Elicitation

Many design problems in the construction of knowledge-based systems are characterized by the availability of different types of knowledge sources. Examples of knowledge sources are human expertise, electronic and paper documents, publications, manuals, collections of observations, existing digital knowledge bases, and, increasingly, electronic sources such as the internet.

If a domain involves a complex relation between problem data and solutions, then direct elicitation knowledge to map problem data into solutions is not effective. It leads to questions to a domain expert that are too global and will therefore not result in useful knowledge. For example, if in a planning problem only the planning goals, initial system state and the format of a required plan are known, a knowledge engineer can only ask general questions like:

Given the planning goals G and initial state I, how do you find a plan P?

Such general questions involving a complex domain are hard to answer for a human expert. A planning expert would not formulate a planning rule that explicitly links the input planning goals and input information to a detailed plan. Instead, a planning expert would most likely come up with a number of intermediate results and a number of steps that link these intermediate results (e.g. a further classification of problem types, draft plan refinement). More specific questions can be asked if additional knowledge, for instance in the form of such intermediate conclusions, is available.

Research in knowledge acquisition has resulted in the formulation of a typology of predefined reasoning tasks (c.f. the task templates in CommonKADS, Schreiber et al 2000) and formalisms that support the formulation, reuse and deployment of reasoning knowledge (e.g. Terpstra et al, 1992). If a task template and a knowledge formalism are found to fit a particular knowledge acquisition problem, they act as a basis for the dialogue between knowledge engineer and expert. Task templates and knowledge formalisms serve two important functions. First, they provide a common language that can be used to phrase more specific questions to the expert. Second, they make it possible to decompose the knowledge acquisition problem into sub-problems, a form of divide-and-conquer.

One difficulty with this approach is the selection of appropriate task templates and knowledge formalisms. The range of templates and formalisms is likely to be large, and without tools, the task of selecting an appropriate concept is far from trivial. Specific tools have been developed to support the user in this approach (Schreiber et al, 2000). In this approach, the formulation of the template is leading in acquiring the knowledge; (economic) feasibility is of minor importance. Once a task template is selected, however, it may turn out to be technically infeasible or too expensive to acquire the knowledge for each component of the template.

6.1.2. Machine learning

Machine learning technology gives the prospect of (partially) automating the elicitation of knowledge and the construction of knowledge-based systems (Brodley, 1992). There are several ways to apply machine learning techniques for knowledge elicitation (Shapiro, 1987; Michie, 1995; Wiegerinck & Heskes, 2002). A direct approach is to collect a set of problem-solution pairs that are provided or approved by a human expert and to apply a machine learning technique to automatically construct a knowledge base. With a suitable reasoning engine, such a knowledge base can be used as a module in a knowledge-based system.

If a concept to be learned is complex it may be hard to induce it at one go. Complexity may be expressed as the number of required attributes involved, or in the functional definition of the mapping between input and output data. The size of the required data set depends on the complexity of the concept to learn. Existing heuristics, many undocumented but well-known among practical experts, link the size of the database to the number of input and output attributes. Typically, such heuristics are connected with specific task types, or with techniques. Examples of such heuristics are:

- For classification:
  - #instances ≥ 5*#attributes (Eilers et al., 2001)
  - #instances ≥ 25 * #attributes * #classes
- For neural networks:
  - #instances ≥ k * #connections, k ∈ [10, 100]

A large data set, consisting of many instances with many attributes, may be needed to acquire adequate knowledge for a complex concept. The exact size of a data set further depends on

problem space characteristics such as noise (measurement noise, classification noise), the (non-) probabilistic nature of the domain and the distribution of the data over the date space.

Acquiring data with the required quality and content may be problematic. In some industrial environments, operational data may be abundantly available and virtually free of costs. In research environments research data management systems are introduced to make data available for consultation and analysis (e.g. Koenderink et al, 2003; Faneyte & Top 2004). Frequently, obtaining data is complex and costly (e.g. when dedicated experiments have to generate the data).

6.1.3. Combining Knowledge Elicitation and Machine Learning

Several authors have presented approaches and techniques for combining inductive techniques and knowledge acquisition. We distinguish two possible approaches to combining machine learning and knowledge elicitation: *integrative* (or *strong*) and *complementary* (or *weak*). The two approaches address two key issues in acquiring knowledge: optimizing the quality of the acquired knowledge and decomposing a complex task in a number of smaller tasks.

The *integrative* approach combines knowledge elicitation and machine learning in one tool for eliciting the knowledge for one task. Alternately or simultaneously machine learning and knowledge acquisition are used to formulate, review and refine one single knowledge base. One of the early examples of this approach was developed by Shapiro (1987) and folowed by several others. Morik et al. (1993) introduce *balanced co-operation*: divide the knowledge acquisition tasks between human and computer to optimize the knowledge acquisition process. This was elaborated in the MOBAL system that allowed the system developer access to all data, generalizations and the included meta-rules as constraints on possible generalizations. It also supports automated refinement of knowledge that was explicitly entered by the user (see also Craw and Sleeman, 1990, Ginsburg, 1988; Aben and van Someren, 1990). Overviews are given amongst others by Michie (1995) and Webb et al, (1998).

The *complementary* approach addresses another key issue in designing knowledge systems: decomposing a complex acquisition problem into a number of smaller problems where knowledge models can be acquired more easily. For each of the sub-problems either knowledge elicitation or machine learning is used to provide the knowledge model. Although in principle any problem can be formulated as one single mapping of input data on output data, for many realistic problems a better solution can be found when it is decomposed into smaller sub-problems: the divide-and-conquer approach. Weak approaches assume acquisition problems that may be large in terms of the number of variables but that, at the lowest level of decomposition, can be provided with a knowledge model of satisfactory quality in terms of task performance. Potentially, every sub-problem may use a different elicitation method (Terpstra et al., 1993). When a system, at least in one subsystem, either contains knowledge obtained by machine learning, or maintains during run-time a knowledge base by means of machine learning, we call the system an *induction-based application*. To efficiently build effective induction-based applications requires both machine learning skills and knowledge acquisition expertise.

Complex learning tasks that involve large numbers of variables and complex, noisy patterns need many data. In many cases there exists prior knowledge about the domain that can be exploited to simplify the learning task. Consequently, there is a need for a structured and economic approach to the decomposition of the acquisition problem into singular tasks.

Decomposing the problem and separately acquiring the data for individual components reduces the required number of examples for successful learning (e.g. Shapiro, 1987).

This paper develops a method for planning the knowledge acquisition process and designing the structure of the resulting system. This method is designed to enable optimal use of knowledge elicitation and inductive learning for available sources of knowledge. It covers the task decomposition functionality at the application level of the MEDIA model of Chapter 5. The approach is based on the divide-and-conquer principle that underlies many design approaches, with special attention to acquisition economy and the quality of the resulting system. We integrate the elicitation-based approach (including the use of predefined methods and ontologies) with the inductive approach, and we show how optimal use can be made of available sources of knowledge using systematic decomposition of the learning task.

The remainder of this section is organized as follows: Section 6.2 specifies cost estimation methods for knowledge acquisition and inductive learning. We propose models for estimating costs of knowledge elicitation and inductive learning approaches, translating the principles that underlie the Cocomo model used in software engineering (Boehm, 1981) to knowledge elicitation and machine learning. Section 6.2 also discusses the estimation of the resulting quality of elicited and learned models. These trends are translated into a model for estimating the quality of knowledge elicitation. In Section 6.3 we introduce and approach for designing *induction-based applications* utilizing the models discussed in Section 6.2. The presented weak approach to designing induction-based applications is a central component in the Method for the Design of Inductive Applications (MEDIA) as introduced in Chapter 1 (see also Verdenius & Engels, 1997). Section 6.4 illustrates the application of the approach by detailing the design process of a fruit treatment planning system, and contrasting the resulting design with both a pure inductive and a pure elicitation-based solution. In Section 6.5 we discuss the implications and applicability of this method, and present some directions for further work.

6.2. Cost and quality estimation

In order to decide between knowledge elicitation, machine learning or further decomposition, we need to be able to estimate ex ante the *cost* and *quality* of the different options. Two important dimensions in project management are the effort (normally considered in the form of money, or project resource costs) that is invested and the quality of the final result (e.g. Bos & Harting, 1998). The two dimensions behave like communicating vessels. If the project costs become too high, they can be reduced by offering less quality. And if the quality is not at the required level, additional investments in effort may be required.

6.2.1. Estimating Costs of Machine Learning and Knowledge Elicitation

Cost estimation of a knowledge elicitation effort has not been a prominent research subject. Scott et al. (1991) indicate that "*... making time estimates for an expert-system project (...) is more an art than it is a science.*" Schreiber et al. (2000) indicate that planning experience is needed, and refers to cost models that are used in the software engineering. There exist rules of thumb that provide indications for project size (the number of entries in a knowledge base, with its strong dependence of the development language), but explicit guidelines to estimate elicitation costs are hard to find. There are only a few documented attempts that provide detailed estimate methods for the development costs of a knowledge acquisition activity during an early project phase. Lethbridge & Skuce (1994) introduce size measures that are closely related to the object oriented knowledge representation in their tool CODE4: The total number of CODE4 concepts and the number of main subjects. They state that these quantities can be estimated early in a development phase. They use these metrics, among many others,

to assess the success of their tool in informal use. A relation between their metrics and development costs however has not been formulated. Menzies (1999) made a cost-benefit analysis of using ontologies, extending Cocomo results. However, it is a discussion on large scale cost effects of applying ontologies, by generalizing the investments in the development of ontologies over various projects. The expected benefits come from the reduction of development time when gaining experience over time. Our concern however is to assess the costs of knowledge acquisition in a single project.

We consulted knowledge acquisition practitioners, both academic and industrial, to get an impression of how knowledge acquisition stages are actually planned in practice. They indicate a number of heuristics, the more prominent fitting in the classes *expert judgement*, *case-wise experience, price-to-win,* and *Parkinsons law,* as defined by Sommerville (1996). One variant of the expert judgement approach is to build a prototype with limited functionality, and to ask an expert to estimate the coverage of the total required functionality. The total cost estimate for elicitation is the extrapolation of the prototype to the total functionality (De Boer, personal communication, 2004). Casuistry can be combined with an expert estimate for planning (Van der Spek, personal communication, 2004). In this approach, a planner with knowledge engineering expertise collects information on the case to plan, and weighs this information with previous experience to come to a cost estimate. Weight allocation can be on the basis of implicit experience, but in principle it may also be implemented in a case based system. Typical information includes the number of experts to consult, whether or not experts reach consensus on solutions, and the applied elicitation techniques. Another factor is the type of result that is required. Possible types include a *knowledge base* for an automated system, but also several knowledge management instruments such as a *manual*, a *list of do's and don'ts*, or the *documentation of a working process*. In a commercial setting, *price-to-win* is an often-encountered approach (amongst others reported by Dunselman, personal communication, 2004). In this approach an estimated or negotiated budget serves as the planning constraint, and knowledge acquisition is accommodated to fit the budget. In order to control this, knowledge acquisition may be set up as in iterative process, where each iteration allows for a further refinement of the results, and iteration stops either when the budget is exhausted, or the quality of the result cannot be improved. All these approaches have in common that the planning strategies, and especially the heuristics that are deployed, remain implicit.

The situation is not much different for machine learning. Although machine learning techniques have been applied abundantly in software engineering to estimate software development costs (Porter & Selby, 1989, In 't Veld, 1992; Menzies, 2001; Menzies & Kiper, 2001, Boetticher, 2001), little effort has been spent on planning the required resources for a machine learning project. Moreover, systematic data on development costs and quality seem unavailable at this time. In our survey of ML applications in the Netherlands (chapter 2), the (perceived) low costs is an argument for only a few organisations to use ML techniques. Documented cost estimation models for ML application could not be identified.

It is our conviction that cost estimation models should be available for machine learning and knowledge elicitation are to become standard tools in system development projects. The fact that such models do not yet exist may be an indication of immaturity of the technology. In information technology, cost estimation models have been developed over time (Sommerville, 1996). We borrow and adapt one of their approaches to make it plausible that cost estimation is a sensible development. A cost estimation model for machine learning and one for knowledge elicitation, without a complete specification and calibration of such a model, are developed. The resulting models are input to the design approach.

6.2.2. Algorithmic Cost Modeling

In software engineering, algorithmic cost modelling has become popular, for example in the form of function point analysis and Cocomo (Boehm, 1981; Boehm et al, 1997). Algorithmic models are built on the basis of historic information on project costs. These costs are related to factors such as project size, staffing, complexity and time constraints. Cocomo has been the basis for cost estimation in several software engineering approaches. We use Cocomo as a basis to develop a qualitative knowledge acquisition (KA) costing approach. The general form of the Cocomo model is:

$$Effort = C \,.\, PM^{S} \,.\, M,$$

With *Effort* as the number of person months to be invested, *C* a complexity measure, *PM* is a size-related product metric, *S* is a measure for the relation between size and effort and *M* a complexity multiplier. *C* strongly depends on the kind of software product to deliver. In software engineering environments it typically varies in value from 2.4 to 3.6. Mostly, *PM* is expressed as the number of thousands of source code instructions (*KDSI*), it serves as an estimate for the size of the project; in the case of high level specification languages (e.g. Levy, 1987), the number of lines coded in this language may become the production factor. *M* can incorporate a number of relevant costing factors, that are grouped in *Product attributes*, *Computer attributes*, *Human attributes* and *Project attributes*.

Due to the complex, involved and repetitive nature of specifying and calibrating the dedicated $Cocomo_{KE}$, $Cocomo_{ML}$ and $Cocomo_{Decomposition}$ model versions for knowledge elicitation, machine learning and decomposition respectively, the original Cocomo model is used as an initial basis to explore each of the knowledge elicitation costs.

6.2.2.1. $Cocomo_{KE}$

A $Cocomo_{KE}$ model should predict the amount of *Effort* to invest in the elicitation of expertise from experts and documents. The first step is to define the meaning of *PM*. A simple measure to aim for, that is closely related to the software engineering conception of *PM*, is the number of lines of code in the knowledge model. Lethbridge and Skuce (1994) propose the number of concepts or the number of subjects as a knowledge acquisition size measure for a given formalism and development tool.

The factors that the consulted knowledge elicitation experts contribute are mapped onto the attribute categories as defined for software engineering. Categorizing relevant factors for knowledge elicitation according to the Cocomo attribute groups that are considered in software economics models (Boehm, 1981), we distinguish:

- **Product attributes**: type of expertise in the system (procedural, schematic, heuristics, formalised models, et cetera), type of result (manual, dos-and-dont list, top 5 of things to do, automated system, flow chart, et cetera), critical role of expertise (operational in process vs. knowledge on marketing, et cetera), knowledge base complexity (e.g., Lethbridge & Skuce, 1994), et cetera
- **Computer attributes**: testing tools, programming environment, programming language, et cetera
- **Human attributes**: the number of experts, inter-expert consensus, partial or total expertise, KA analyst capability, application experience, KA tool experience, et cetera
- **Project attributes**: used elicitation techniques (interviews, think aloud sessions, et cetera), testability, development risk, number of project collaborators, et cetera

6.2.2.2. $Cocomo_{ML}$

A $Cocomo_{ML}$ model should predict the amount of *Effort* to invest in the elicitation knowledge for the application of machine learning techniques. Again, the first step to define is the meaning of *PM*. It is tempting to define the size of the generated code as the *PM* parameter, but caution is needed here. In software engineering, as in knowledge engineering, the end product, lines of code, are literally handcrafted: the knowledge engineer produces the code by typing the result. The lines of code in ML solutions are not generated by the ML operator, but by the ML technique. Consequently, another *product size metric* for machine learning has to be defined, a metric that determines the size of a project in terms of workload. We did not identify any literature source on estimating project size in the current data mining community. At present, the PM factor is predominantly confined to the size of the data set and an expert's estimate on the complexity of the patterns to learn. The size of a data set can be expressed in data-points, e.g. in terms of #records*#attributes. It is noted that pattern complexity may be a misleading measure. Michie (1995) shows how a proper formulation of the problem (by means of constructing additional attributes) can dramatically reduce the size (as well as the learning complexity) of the problem solution, even though the size of the data set increases. Therefore extensive empirical research on the relation between problem characteristics and project workload is required for improving the PM metric. Although indispensable and necessary, such a project goes beyond the scope of this thesis.

Now, typical attribute groups become:

- **Product Attributes**: complexity estimate, data size, required reliability, et cetera
- **Computer Attributes**: learning time constraints, size limitations result, execution time constraints, et cetera
- **Human Attributes**: ML analyst capability, application experience, learning algorithm experience, et cetera
- **Project Attributes**: use of software tools , integration application - inductive application, et cetera

Further study and calibration are needed to make adapted Cocomo variants useful tools for estimating costs of knowledge elicitation and machine learning projects.

6.2.2.3. $Cocomo_{Decomposition}$

A $Cocomo_{Decomposition}$ model should predict the amount of *Effort* to invest in the elicitation knowledge when decomposing the functionality. When functionality is decomposed, the effort of the total system is the sum of the effort of the decomposed components. Consequently, a $Cocomo_{Decomposition}$ consists of adding the costs of knowledge elicitation for the constituing sub-tasks. Therefore a separate Cocomo model for decomposition is not required.

6.2.3. Quality estimation in machine learning and knowledge acquisition

We want to weigh the estimated costs against the estimated benefits in our decomposition approach.. however, in information system development practice, this way of weighing alternatives during the system development is not common. In the IT world, the potential benefits of a system with specified functionality are estimated in the business case. Based on these potential benefits, the budget for development and implementation is estimated with a budgeting model that links aspects such as size and complexity to the estimated development effort. And in the next step, a development plan is established, based on the (now fixed) specification of the functionality. The functionality to deliver may be re-negotiated in case of

budgeting problems, but to a large extent the functionality is fixed when planning the development. The quality is a binary attribute of the module: the component complies, or it complies not. In software engineering, the quality of the end result is normally expressed as a combination of the realisation of the functional requirements, combined with maintenance characteristics.

The (pre-fixed) functionality is a major difference between software engineering and knowledge acquisition. Knowledge systems split the system functionality in two. The inference structure, consisting of the task control and underlying inference mechanisms, can be compared to a traditional IT system. The acquisition of the knowledge base requires adifferent kind of assessment. In knowledge elicitation, the quality aim is not primarily to develop a model with predetermined functionality. An additional factor is the performance quality of the resulting knowledge component, that is to say does the model exhibit a predictive capability that is substantial? The best way to assess that is by answering the question to what amount the module performs with a comparable predictive accuracy as the human expert the model was obtained from.

Numerous quality attributes can be found in the literature for machine learning. Initially, the are interest is in the accuracy. For some straightforward knowledge elicitation tasks, such as classification and prediction, similar measures can be handled. Before actually running a ML technique or knowledge elicitation task, a priori quality estimates are needed to balance the development costs against the benefits. However, a priori quality estimation is a seldom explored terrain, even more so than in the case of cost estimation. It is hard to predict whether a fixed quality measure can be set at the start of knowledge elicitation or machine learning. Consequently, planning the effort to realize the pre-defined quality is difficult. Often, requirements are formulated comparative to expert performance, e.g. obtain 90% of the accuracy a domain expert achieves. Alternatively, real world performance targets are set, e.g. obtain an accuracy that leads to less then 10% fruit deterioration. .

Ex ante quality estimation tools are being developed for machine learning. One of the results of the MetaL project (Metal 2004) is a web site that takes a data set as input, analyses the data characteristics, and produces a ranking of algorithms as a result. MetaL uses empirical data on the performance of the learning algorithms on a multitude of data sets. The descriptors of a new data set are used to identify similar data sets, and the performance of the algorithms on the adjacent data sets are used to estimate the performance on the new, unseen set. There is experimental evidence that that this approach offers a usable prediction of ML performance (Soares & Brazdil, 2000). Moreover, the algorithms can be actually run on the data to assess the actual performance on the data. Another, related approach is landmarking (Pfahringer et al., 2000) that uses prediction quality of a number of simple ML techniques to predict the estimated quality of more complex techniques. Although these approaches were developed with the goal of ranking machine learning technique, in order to identify the most suitable technique, they can also be used as a prediction of the accuracy.

However, estimating the quality of a solution in terms of predicted behaviour has not yet been explored extensively for knowledge elicitation. Currently, as indicated by domain experts, quality estimation for knowledge elicitation is a realm of informal rules. Apparent differences of opinions among experts, or the fact that an expert needs more than 10 minutes to solve a problem are seen as indications that domain knowledge may not be elicitable. The complexity of an elicitation problem may be indicated by the number of problem features, by the number of possible solutions, by the time it takes a novice to become an expert, or by the existence of a general accepted handbook.

In the case of MetaL quality estimates are based on descriptors of the data set (Bensusan & Alexandros, 2001). We propose to do the same for knowledge elicitation: Collect systematic data on the performance of knowledge elicitation problems and elicitation approaches, and formulate a parametric model to relate problem characteristics to the quality of the delivered solution. The availability of solutions in the areas of software engineering and machine learning provide reason to be confident on the feasibility of such an approach. However, the required effort to derive such a model is considerable. As such, the derivation of such a model falls outside the scope of this work. For the moment, we assume the availability of a model that takes problem descriptors as input, and provides a numeric quality estimate on the same accuracy scale as is available for machine learning techniques.

6.3. Decomposition of Knowledge Elicitation Problems

This section describes the decomposition of knowledge elicitation problems. The choice between alternative decompositions is motivated by the possibility to elicit an adequate knowledge component against reasonable costs. Two main differences with standard approaches for the development of knowledge systems are that:

1. cost and accuracy estimates of three alternative acquisition options, direct machine learning, direct elicitation and problem decomposition, are considered and evaluated at the design level, without actually building the components, and that
2. the decomposition process is directed by the availability of knowledge and data sources and economic considerations rather than by a priori template knowledge models.

The machine learning and data mining community mostly considers the decomposition of data mining problems in terms of the phases in the *process of data mining* (Adriaans & Zantinge, 1996; Fayyad et al, 1996; Chapman et al., 2000) rather then in decomposing the *system functionality* (Engels, 1999). Typical phases include data selection, data cleaning and data conversion (see for example Garner et al, 1995; Fayyad et al., 1996; Brodley & Smyth, 1997; Engels et al., 1997; Chapman et al., 2000). This section considers decomposition of the required functionality in a number of functionalities of reduced size and complexity (cf. Engels, 1999, ch 6). The aim is to minimize the acquisition costs. It is, in that sense, more related to stepwise refinement as defined in software engineering (e.g. Wirth, 1976) and to task decomposition in knowledge engineering (e.g. Schreiber et al, 2000) than to the earlier mentioned data mining methodologies.

The main difference between the approach presented here and the approach presented by Engels (1999) is that in the latter work task types are defined top-down on the basis of knowledge acquisition sources, whereas in the approach presented here, a bottom-up survey of sources in the domain guides the decomposition.

Our approach offers three extensions compared to most other knowledge acquisition methods. First, it considers data as a potential source for acquiring knowledge. Human experts are normally considered the main source of knowledge in knowledge acquisition, and many tools and techniques focus on extracting knowledge from human experts (Schreiber et al, 2000). Second, our approach explicitly takes the costs and accuracy of acquisition into account. Decomposition of a task into sub-tasks can take place if the resulting task structure scores better on the cost-benefit scale. Finally, we explicitly consider machine learning as acquisition method when decomposing tasks[19]. Decomposition of a task into sub-tasks can

[19] Terpstra et al. (1993) offer an approach where machine learning is integrated in a knowledge acquisition context. However, the choice for machine learning is motivated by the task structure, and not by the availability of data sources to learn from.

take place if in the domain ontology data-types are available that can serve as input and output to sub-tasks. Moreover, the sum of input and output of the sub-tasks decomposition needs to have the same input- or output-structure as the task that is decomposed. When this is not possible, or the resulting decomposition is not preferred by the knowledge engineer, additional sub-tasks have to be defined by the user in order to allow decomposition.

The starting point of the decomposition process is the definition of the functional requirements: the input-output mapping of the problem to be solved. This is typically a (formal or verbal) definition of available and demanded data items and their semantics. The domain ontology provides the relevant concepts that are available to express problems and solutions. In CommonKADS (Schreiber et al. 2000) the domain ontology (or domain knowledge) is further defined as a combination of a domain schema and a knowledge base. The domain schema contains type definitions of the static information and knowledge structure of the application domain. The knowledge base contains the instances of the types. We interpret a domain schema to consist of:

Data types: An overview of the data sets that are available in the domain (record name, fields, field types, constraints, relations)
Knowledge types: Existing or obtainable mappings between data types.

Examples of data types in the fruit domain are:

- **Batch Id**: numerical, e.g. *ddmmyynnnn* with *ddmmyy* a date and *nnnn* a 4-digit serial number,
- **Weight**: numerical,
- **Cultivar**: a string value from a set of potential cultivars,
- **Country** of origin: a string value from a set of potential countries, and
- **Orchard**: *CCcccc* with *CC* the EAN country code, and *cccc* the EAN company identification.

An example of a knowledge type in the fruit domain would be the definition of an expert product selection:

$$\text{SampleSelected} = f_{\text{Classification}}(\text{expert selection criteria})\ (\text{product data}),$$

indicating that for some product, described in product data, the attribute SampleSelected is attached to product data in a classification function that takes expert selection criteria as knowledge model.

6.3.1. Acquisition planning

The next step is to plan the use of the available resources for the actual knowledge acquisition. The result of this step is a plan that specifies which resources are to be used to acquire the knowledge for the components of a target system. The structure in which these components are connected to form the final system is also produced as output.

A knowledge acquisition problem can be solved in two ways:

- by *direct elicitation* of the knowledge from a source, e.g. a human expert or a document, or
- by *learning* from observations.

If a problem is not solvable as a whole, it may very well be solvable by reduction: *decomposition into sub-problems* that are solved recursively. Moreover, decomposition may lead to a more cost-effective solution. Decomposition continues until the sub-problems can be mapped onto acquirable knowledge resources,that is to say that a set of resources is found, connectable in a data-flow structure, from which knowledge can be acquired to perform the task of the target system. When a task is directly mapped onto a source that can not further be decomposed it is referred to as a *primitive task*. Our notions of task and primitive task are

adjacent to the notions of task and inference as used in knowledge acquisition (e.g. CommonKADS, Schreiber et al. 2000).

Each of the options *direct elicitation*, *learning from observations* and *decomposition*, involves further choices. The main criterion for these choices is *acquisition economy*: the balance between the expected costs of implementing the option and the expected accuracy of the resulting knowledge (O'Hara and Shadbolt, 1996). In case of *direct elicitation* by manual acquisition methods, the accuracy depends on the quality of the sources (e.g. human experts), the knowledge engineers, and of the communication process. In case of *learning from observation* the accuracy of the result will depend on the availability of reliable data, the complexity of the actual relations, and on knowledge about the type of relation that is to be induced. If many reliable data are available, if the underlying relation is not very complex and if the type of function is known, then machine learning is likely to be successful. Otherwise there is a risk of constructing knowledge that is incorrect.

The idea of *decomposition into sub-problems* is based on the top-down approach of stepwise refinement (e.g. Wirth, 1976) and divide-and-conquer strategies (Michie, 1995). In decomposing complex tasks, one aims at a minimal coupling between components, and a maximal internal coherence. The main difference is that in software engineering the main principle that guides decomposition is minimizing the complexity of the resulting systems and thereby supporting activities like debugging, quality management, system maintenance and re-using the system. In the construction of knowledge-based systems, knowledge acquisition is a main factor to determine the total project costs and benefits. Additional to the software engineering arguments, decomposition in a knowledge acquisition context is useful if (cheap and accurate) sources of knowledge are available for sub-tasks of the overall knowledge acquisition task but not for the overall task. For example, there may be abundant data for one sub-problem and a communicative expert for another sub-problem but no sources for the problem as a whole. This is a reason to split the knowledge acquisition problem into sub-problems that are then acquired separately. Another situation where decomposition can be cheaper and give more accurate results than a single step inductive approach is when there is no prior knowledge to bias machine learning on the overall problem.

6.3.2. Acquisition Economy[20]

To decide if decomposition is a good idea, we compare the expected costs and benefits of elicitation, machine learning and decomposition. The benefit of an acquisition operation (elicitation based or machine learning based) is the accuracy of the resulting knowledge[21]. The costs of the acquisition process depend on the selected acquisition approach. In case of elicitation, these include time of the expert and of the knowledge engineer, equipment, etc. In case of an inductive approach, these include the costs of collecting and cleaning data and of

---

[20] We focus here on *acquisition economy*, not on *implementation economy* or on *operations economy*. To asses the total strategy, the costs of implementing and operating PSM's should be included; this comes close to the cost assessment that is applied in software engineering approaches (Sommerville, 1996); this falls outside the scope of our work. *Decomposition costs*, the costs associated to the design, only matter if there is a large cost difference in design effort between design alternatives. This has to be assessed, and is not yet included in this paper.

[21] Ultimately the benefit may be expressed in terms of financial benefits but these are often even more difficult to estimate than the accuracy and the acquisition costs.

applying a machine learning system to the result. If we decompose the acquisition problem, the costs and benefits are the sum of those associated with the sub-problems. So we get for elicitation/machine learning:

$$EG_{e/ml} = w * EA_{e/ml} - EC_{e/ml} - EC_{implementation} - EC_{operation} \quad (6.1)$$

and for decomposition:

$$EG_{compound} = \Pi_j (w_j * EA_j) - \Sigma_j EC_{e/ml\_of\_KB\_j} - \Sigma_j EC_{implementation\_j} - \Sigma_j EC_{operation\_j} \quad (6.2)$$

With:

- *1..j modules*
- *EG = expected gain in manmonth effort*
- *EC = expected costs in manmonth effort*
- *EA = expected accuracy in percentage*

Here the weight parameters w_j, with unit *manmonth/% accuracy*, indicate the importance of accuracy relative to costs of acquisition; they give, per module, a relative economic importance to accuracy. As such, they can be seen as a measure for *return on investment in accuracy*. The expected accuracy of a compound acquisition is the product of the accuracy of its components, which is a pessimistic estimate. In case of reuse or combination of knowledge bases, the actual acquisition costs will decrease. As we argued above, in some cases direct elicitation is almost impossible because the expert cannot answer very global questions. This means that the *costs* are high and the accuracy of the knowledge is low. In machine learning applications the costs of actually running a system are usually rather small compared to other costs, such as designing the target system, collecting data and tuning the machine learning tool. Thus this costs of running a system could be left out. The formulas specify, as part of the expected gains, the implementation costs. In the rest of the chapter, this subject is ignored; it is considered the area of software engineering.

```
KA(ka-problem, sources, w):
1) IF a source is available for goal
   THEN    use this source,
           estimate costs & accuracy
           compute gain
   ENDIF
2) REPEAT: use sources to decompose ka-problem into ka-sub-problem
3) estimate expected gain of elicitation/induction(ka-problem,
   sources, w):
   a) estimate costs(elicitation/induction, sources, w)
   b) estimate accuracy of resulting
      knowledge(elicitation/induction, sources, w)
   c) expected gain(elicitation/induction) = (w * estimated
      accuracy) - estimated costs
4) estimate acquisition costs and accuracy and compute gain
5) UNTIL no more decompositions
6) select the option (elicitation, induction or a decomposition)
   that has the highest expected gain.
7) IF the result of the last step is a compound ka-problem THEN
   FOR EACH ka-sub-problem DO KA (ka-sub-problem, sources, w)
```

Algorithm 6.1 Decomposition method for knowledge acquisition problems

6.3.3. The Decomposition Process

Decomposition is visualized in *data flow diagrams* (DFD; Sommerville, 1996). DFD's offer an intuitive notation to show how data is processed by a system. A DFD represents processes, data stores and data movements between processes. Examples later in this text apply a DFD-like representation, where intermediate data sources are omitted to improve the comprehensibility. A decomposition is constructed by

- inserting a source description (input output description; see Table 6.2) that is connected to one or more types of data in the current goal;
- adding or deleting a connection in the data-flow structure; and
- inserting a method (a sub-procedure) for a component in the data-flow structure.

The method for decomposing a knowledge acquisition problem is based on the idea that the reasons for decomposing a knowledge acquisition problem are applied in the order given above. The method is a form of *best-first search* that uses expected costs and benefits to evaluate candidate decompositions.

In case of further decomposition, the method is applied recursively to the sub-problems. The algorithm is listed as Algorithm 6.1. If costs and accuracies cannot be estimated the alternative is to perform a pilot study to assess the costs and expected accuracy as indicated by de Boer (personal communication, 2004). In the context of elicitation it amounts to performing elicitation on part of the task and evaluating the result. In the context of machine learning it amounts to comparative studies by cross validation. Main goal of such studies is to assess, in terms of costs and accuracy, the complexity of the learning task at hand. As a side-effect of such pilot studies, information on the best performing machine learning techniques may be collected for further use in the next phase.

6.4. Example: The Product Treatment Support System

We illustrate the method with the example on planning systematic treatments for fruit ripening (Verdenius, 1996) that is introduced in Chapter 2. The developed PTSS system involves both knowledge elicitation and machine learning components. And although the project has been realised before the existence of a systematic design approach, this section can be viewed as a post hoc *design rationale* of knowledge acquisition process for the PTSS. In this reconstruction, the focus is on finding the structure of the knowledge sources.

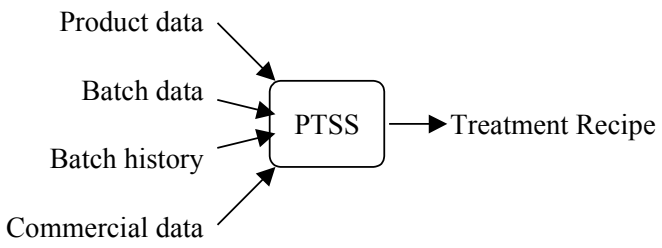


Figure 6.1. Description of the input and output that defines the learning problem

The initial acquisition goal for the PTSS is:

> *Construct a system that transforms input information on a batch of fruit that arrives from abroad into a recipe for storing and ripening the fruit, such that the final fruit quality matches the predefined requirements.*

6.4.1. Requirement Definition and Source Identification

Figure 6.1 shows the overall learning problem. Figure 6.2 illustrates part of the domain model. The outcome of the task, i.e. the solution to the planning problem, is a *treatment*

recipe. A recipe is a prescription of the values of control variables for a set of relevant *treatment conditions* in a specific time interval. A treatment recipe is a prescription of external conditions that are to be imposed onto the product in order to generate a specific quality transition. The time interval is subdivided in fixed-duration time-slices. Storage conditions may include attributes like temperature, relative humidity and ethylene concentration; the product type determines the actual relevance of conditions.

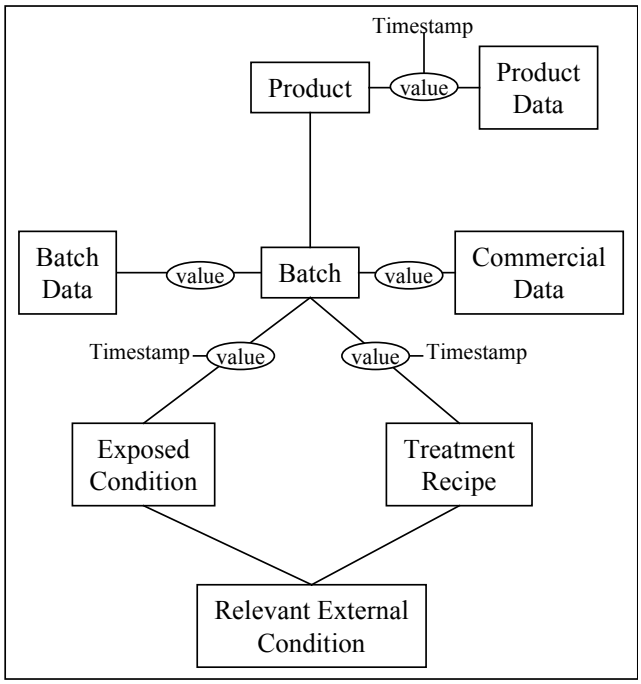


Figure 6.2. Part of the domain ontology en the form of an entity relation diagram

The first step is to identify available sources of knowledge for this task. This should be read as a schema that can be instantiated with specific knowledge. The following data about a batch of fruit are available:

- **Batch data**, such as *origin*, *cultivar* etc.
- **Commercial data**, mainly the required *due date* of the product treatment and the *required quality* after the treatment. It is further assumed that a fixed final quality is delivered for all recipes. Consequently, required quality will not be used.
- **Product data**, being a number of measured values for individual product attributes such as *color*, *shape*, *firmness*, *weight* etc, describing per individual product in a batch various quality aspects at the start of a recipe.
- **Exposed conditions,** such as storage temperature or the concentration of relevant gases such as ethylene, O2 or CO2

Some values are provided with a timestamp; examples are firmness and storage temperature. By taking a vector of values with successive timestamps, a historic development of a feature is obtained.

Table 6.2 lists knowledge sources that are available in this domain. These knowledge sources cover the application of machine learning and the elicitation of knowledge from experts and documents. In this stage of the design process, the resources are identified and the basic method (elicitation or machine learning) for acquiring knowledge from the resource is established, but no effort is made to extract the actual knowledge. Costs and accuracy are estimated according to the guidelines given in Section 6.2. Actual acquisition of the knowledge is postponed until a complete plan is available. Only when a complete plan is available a specific machine learning technique or elicitation approach will be determined.

Acquisition of knowledge for commercial applications should be guided by economic principles. To be able to do so, the expected costs and the expected accuracy of the knowledge that can be acquired from a resource should be estimated at the moment of acquisition planning and based on the best possible implementation of the acquisition process. That is, such an estimate should give technique independent estimate of effort and quality, based on the problem complexity. It is then up to the knowledge acquisition specialists to execute the knowledge acquisition process within the effort and quality constraints.

Some sources may have no costs, e.g. if the required knowledge already exists in an executable format. Moreover, note that for a (sub)problem and (sub)solution combination, there may be more than one way to acquire the knowledge. For example, it may be possible to directly acquire knowledge that relates *External Conditions* and *Planning Destination* to a *Detailed Recipe*.

In Table 6.2, the costs of using these resources and the expected accuracies were estimated using rules of thumb. More sophisticated methods are becoming available. Pilot runs of machine learning techniques may help to predict the performance for an inductive approach (cf. the landmarking approach discussed earlier). For knowledge elicitation, a similar approach may be developed, based on the concepts proposed in Section 6.2. For example, *Specify Requirements* involves finding a detailed recipe specification from product data, batch data and the required due date of the batch. The cost for this task is estimated from the availability of resources and the complexity of the task. The size of the space defined by the properties in *Specify Requirements* gives an indication of the number of data that must be acquired to obtain certain accuracy in case of an inductive approach. This in turn gives an estimate of the costs. In this case a cost of 0.7 and an accuracy of 0.95. The relation is likely to be complex and this suggests that many cases are needed. Costs and accuracy of an elicitation approach are estimated from the time that it takes to acquire the expertise for a task. If this is unknown then a rough estimate is made based on the complexity, similar to the inductive approach. The accuracy is estimated from a pilot experiment.

Estimated costs and accuracies are not always available. Obtaining these estimates may require a more detailed study, for instance in the form of initial experiments. This in itself may induce additional costs.

6.4.2. Acquisition Planning

Figure 6.1 shows the overall learning problem and Table 6.3 gives an overview of the sources and techniques that were actually used to acquire knowledge for the various components. Below we reconstruct the process that leads to this decomposition and the choice of acquisition methods.

The estimated costs of acquiring the complete system by elicitation are relatively low because it is based on a set of standard recipes that are already available. The costs for machine learning on the other hand are high because of the amount of data that is required to enable successful machine learning. There are about 15-40 input variables and between 12 and 52 output variables (depending on the due date of the batch and thus the duration of the storage). The relation is therefore likely to be very complex and it would take a large data set to find an initial model, if possible at all. We estimate costs and accuracies of single step acquisition (elicitation or machine learning). The estimates are presentedin Table 6.4. The last column gives the expected gain using the value 3 for weight values w_i as a reasonable value, when interpreted as the return on investment.

Source ID	Input	Output	Cost	Acc	Acquisition Method	Estimate method
Expert Select Product	$[\text{Product Data}]_n$, Selection Criteria	$[\text{Product Sample Data}]_m$	0.2	0.3	Knowledge elicitation	Cost & accuracy estimated by piloting
Design Recipe	Recipe Specification, Due Date, Planning Strategy, Design Model	Recipe	0.4	0.9	Knowledge elicitation	Costs estimated from problem complexity. Accuracy estimated
Postulate Recipe 1	Due Date, Standard Recipes	Recipe	0.2	0.2	Knowledge elicitation	Cost and accuracy follow from existing situation
Select Standard Recipe	Due Date, Standard Recipes	Standard Recipe	0.1	0.6	Knowledge elicitation	Cost and accuracy follow from existing situation
Adapt Standard Recipe	Standard Recipe, Adaptation model, Batch History, Cultivar, Batch Origin	Recipe	0.8	0.3	Knowledge elicitation	Costs and accuracy estimated, based on existing situation and expert indication on complexity
Estimate Quality 1	$[\text{Product Data}]_n$, Batch Origin, Cultivar, Batch History	Quality Assessment	0.8	0.85	Machine learning	Cost obtained from expert assessment. Accuracy calculated
Estimate Quality 2	$[\text{Product Data}]_m$ $m << n$	Quality Assessment Sample	0.4	0.92	Machine learning	Cost obtained from expert assessment. Accuracy calculated
Select Product	$[\text{Product Data}]_n$	$[\text{Product Sample Data}]_m$	0.3	0.98	Machine learning	Cost estimated. Accuracy obtained from experiments
Specify Requirements	Quality Assessment (Sample), Due Date, Batch Origin, Cultivar	Recipe Specification	0.4	0.95	Machine learning	Costs estimated from experiments. Accuracy estimated
Postulate Requirements	Product Data, Batch History, Batch Origin, Cultivar, Due Date	Recipe Specification	0.8	0.8	Machine learning	Cost and accuracy estimated
Postulate Recipe 2	Quality Assessment, Batch Origin, Cultivar, Batch History	Recipe	0.9	0.15	Machine learning	Costs estimated from problem complexity. Accuracy estimated from experiments
Postulate Recipe 3	Quality Estimate, Batch Origin, Cultivar, Batch History	Recipe	0.4	0.25	Machine learning	Costs estimated from problem complexity. Accuracy estimated from experiments

Table 6.2. Sources of knowledge for the example of fruit storage planning (Chapter 3). The source ID is a label to identify the knowledge source; labels indicate the functionality that the source supports. The *Determined by* and *Conclusions* indicate the types(s) of in- and output that are covered by the source. E.g. *Estimate Quality 1* is a source that transforms the vector of n descriptions of products into a statistical estimate of the quality of a batch. *Acquisition method* indicates how the content of the source is obtained. Statistical estimate and machine learning obtain knowledge from exemplary data, where expert, oracle, fixed strategy and random selection are considered 'elicitated' models.

Subtask	**Acquisition means**
Select products	Induction (Decision tree)
Estimate Quality 2	Standard Math (Average)
Specify recipe	Induction (Neural Networks)
Detail recipe	Elicitation (Constraint network)

Table 6.3 Acquisition methods and sources for components

Method	**Accuracy**	**Costs**	**Gain**
Postulate Recipe 1 (elicitation using standard recipes)	0.2	0.2	0.4
Postulate Recipe 2 (inducing a recipe planning model from data)	0.15	0.9	-0.45

Table 6.4 Estimated cost and accuracy of single step acquisition(w_I= 3)

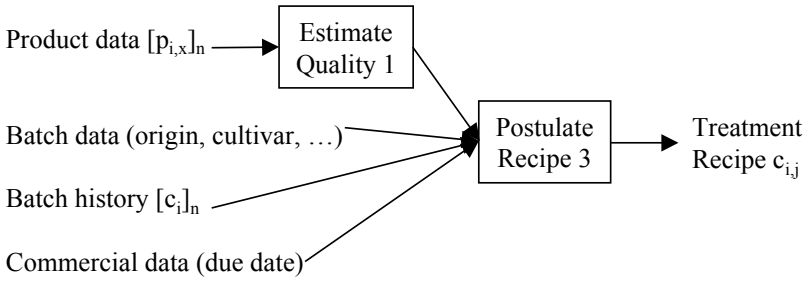


Figure 6.3 Decomposition 1

We now consider decomposition. The available sources plus the causal and temporal structures define a number of possible decompositions of the initial knowledge acquisition problem. There are many possibilities. Here we describe some of them with estimates of the expected accuracy and acquisition costs.

6.4.2.1. Decomposition 1

From the initial problem of Figure 6.1, the first decomposition step is to abstract from the quality data on individual products to the quality of the batch. This requires a number of measurements. Taking the average of a number of measurements requires a large product sample (see Verdenius 1996). In Figure 6.3, the resulting decomposition is depicted. The expected gain of this decomposition is: 3 * 0.21 - 1.2 = -0.56.

6.4.2.2. Decomposition 2

The next step in the decomposition aims at overcoming the weakest point in decomposition 1. Postulate Recipe 3 has a poor cost/accuracy ratio. It can be replaced by a two step approach, where recipe specification is followed by a recipe design. The resulting decomposition is shown in Figure 6.4. The expected gain of this option is: 3 * 0.73 – 1.8 = 0.58. Already a non-negative outcome, but still worse then the original problem formulation (over the ROI).

6.4.2.3. Final Decomposition

The final decomposition again is advocated by first identifying the weakest point in the best-so-far, and identifying a task combination with a better pay-off. Here, it appears that estimating the product quality can be optimized by first drawing a small sample from the total data set, and using these data to estimate the quality. Due to sample reduction the benefit increases. The resulting decomposition is shown in Figure 6.5. The expected gain of this is: 3 * 0.77 - 1.5 = 0.81

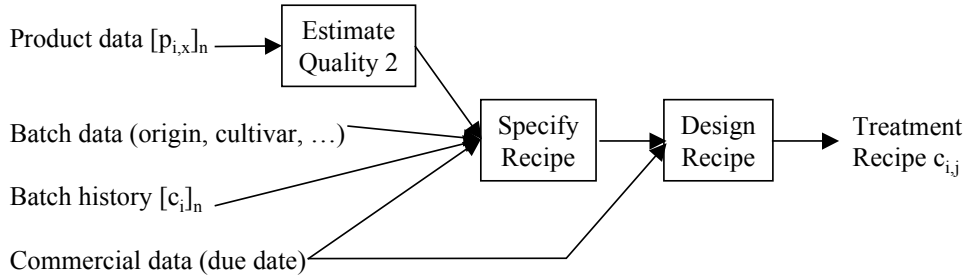


Figure 6.4 Decomposition 2

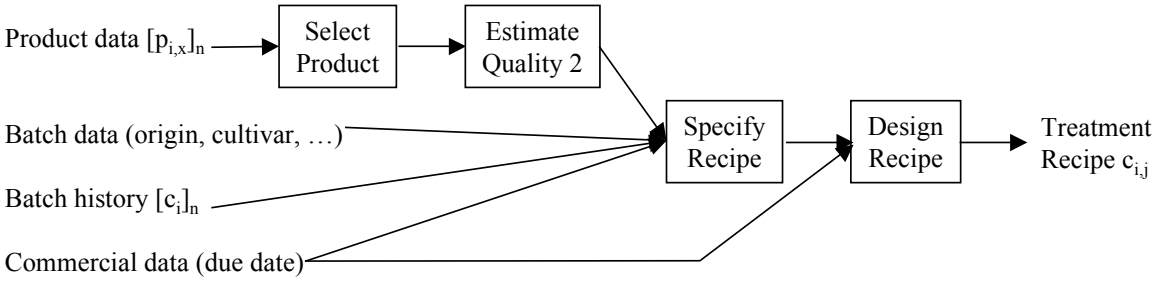


Figure 6.5 Final decomposition of the knowledge acquisition problem (intermediate data sources have been omitted)

6.4.3. Pure solutions

Expert technicians have a tendency to transform any problem to suit there specific techniques. As illustrated in Chapter 2, this is a problem in machine learning, as the expertise to rationally consider specific machine learning techniques against alternatives is lacking. This makes it reasonable to assume that ML-experts would come up with a learning solution for the entire problem. From the available knowledge sources in Table 6.2 it is easy to determine the best performing solution that exclusively uses ML-techniques. Figure 6.6 shows the resulting decomposition. The expected gain of this solution is 3*0.14-1.6=-1.19. This is mainly due to the cost-quality ratio of the *Postulate Recipe 2* source.

Alternatively one can imagine a solution with the exclusive use of knowledge elicitation. Figure 6.7 shows the best potential decomposition that exclusively deploys knowledge elicitation sources. The expected gain of this solution is 3*0.77-1.50 =0.30.

With these examples we illustrate that pure solutions may be attractive from the point of view of a human designer, but that in terms of operational quality, they are non-optimal.

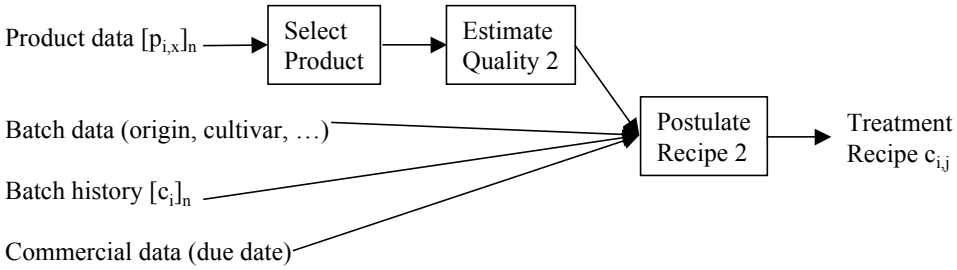


Figure 6.6. Pure inductive solution

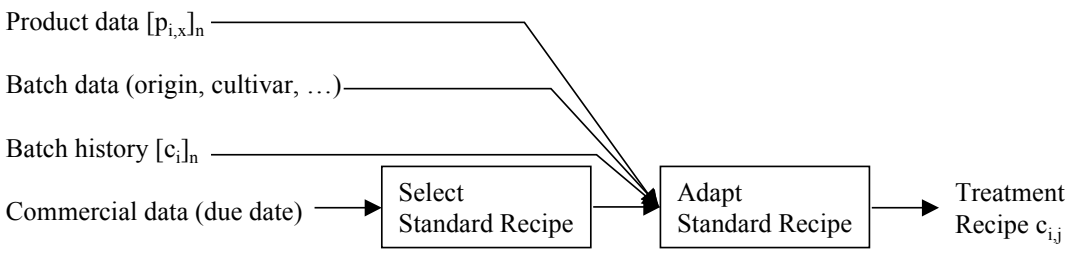


Figure 6.7 Pure elicitation solution

6.4.4. Data analysis, technique selection and application

Following the acquisition plan as detailed in the earlier sextions, it is now possible to focus on the actual knowledge acquisition for the components. The first component, *Select Products* implements a sampling procedure. For each product, a number of easy-to-assess data items are available. Based on these items, the product is classified as being either *near_batch_mean* or *far_from_batch_mean*. This is a classification task. Historic data on the relation between product descriptors and batch mean are available. On the other hand, looking at this relation is fairly uncommon for human experts. Consequently, elicitation of knowledge from human experts is not an option (low accuracy vs. high costs). Data analysis may discover that the underlying type of function is relatively comprehensible. And interpretability is a major (non-functional) requirement, if this reasoning step in order to select fruits has to be performed by human experts. In the actual planner, acquisition has been implemented by means of a decision rule learner. The rules are extracted, and handed over to a human expert to perform the actual selection on location.

The next component is the actual assessment of the batch quality. This is simply the averaging of the measurements. The main difference between the two available *Estimate Quality* sources can be found in the number of (expensive) measurements that is required in the case of unselected and selected estimation. The unselected estimation requires between 60 and 200 expensive measurements. The selected measurements require only 5-10. This does not dramatically affect the accuracy, but dramatically reduces the costs.

The two options of elicitation or machine learning must be evaluated for the acquisition of recipe specification. Human experts are not used to *Specify Requirements* on batch level; i.e. expertise is not available. Historic data are available for machine learning of the required knowledge. Twenty-one attributes may be taken as input, while the size of the output is limited (only 1 parameter was output in the fruit planning system). Based on a comparison between linear and non-linear models, a preference was developed for non-linear models. An error back propagation neural network (Rumelhart et al., 1986) was selected, and trained to classify the new, unseen cases. The best performing network was established in an experimental setup (see Chapter 3).

6.5. Conclusion

6.5.1. Discussion

Applying machine learning for knowledge acquisition is more than running an appropriate machine learning technique. In general, both knowledge and data will only be available for acquiring parts of the total required body of knowledge. In order to cover the total required functionality, knowledge-based and induction-based components should collaborate, and decisions must be taken on how to combine them to acquire the required knowledge. In this chapter we have presented a rational approach to combining machine learning techniques and elicitation of human expertise in a knowledge acquisition context. It offers a mixed approach to planning of knowledge acquisition. Based on the availability of elementary knowledge sources in the problem domain (bottom-up), the overall problem is decomposed in feasible components (top-down). The decomposition process balances acquisition costs and expected benefits, offering an explicit rationale to make the choice even when the estimates of costs and accuracies are not yet firm. This was illustrated for the PTSS system of Chapter 3.

The approach gives better solutions than if pure elicitation or pure machine learning would be deployed. The pure approaches ignore valuable sources of knowledge that lead to better, more cost-effective solutions.

The quality of a solution formulation currently incorporates development costs and predictive quality. In practical situations however, other factors may become relevant, e.g. implementation and operation costs, comprehensibility, acceptance by staff or consumers and compatibility with other solutions. In such cases, the expected gain criteria (1) and (2) of Sub-section 6.3.2 may become more complex, and incorporate various quality attributes. This would generate the need for a multi-criteria decision approach (Saaty, 1980).

6.5.2. Comparison with other methods

Many existing knowledge acquisition methods deploy problem decomposition (e.g. Marcus, 1988, Terpstra at al., 1993; Schreiber, 1993). However, many of these methods focus on *modeling languages* and rarely make the underlying principles explicit, especially in regards to how to apply (problem decomposition) rationally. These methods also do not cover the use of inductive techniques. Necertheless, some decomposition approaches are proposed within the ML domain. Engels' User Guidance approach (UGM; Engels, 1999) starts to reason from the functionality of machine learning techniques. The main difference between the UGM view on simple tasks and the CommonKADS view on knowledge acquisition of primitive inferences is that the former has a close binding with the techniques, whereas the latter has binding with the inference layer, that is technique independent.

In contrast, we reconstruct the rationale behind the knowledge acquisition methods and extend them towards the use of machine learning methods. We presented criteria and a method for decomposing knowledge acquisition problems into simpler sub-problems and illustrated this with a reconstruction of a real world application. This process can be applied to inductive methods, knowledge elicitation and other manual acquisition methods.

In modern approaches for knowledge acquisition, especially in CommonKADS, the starting point for divide-and-conquer approaches is identified from libraries of standard models. For example, suppose that the acquisition problem is to construct a system that can design storage recipes for fruits. The knowledge engineer may decide to adopt a model from a library of problem solving methods (McDermott, 1988; Breuker and VandeVelde, 1994). First, the problem is specified as before:

> ***Input****: Fruits Characteristics, Current Quality, Required Quality, and Recipe Duration*
>
> ***Output****: Storage Recipe, i.e. condition set-points for a series of time-slices*

The KADS library now offers a number of models (see Table 6.5).

Name	**Input**	**Output**
Design	Needs and desires	Design solution
Configuration	Components, required structure, constraints, requirements	Configuration
Planning	Initial state, goal state, world description, plan description, plan model	Plan
Assignment, scheduling	Components, resources	Assignment

Table 6.5. Candidate models for fruit storage planning task

It is not a priori obvious which of models can be appropriately employed here. *Recipe Duration* can be viewed as needs and desires, constraints, requirements, and plan descriptions. *Fruits Characteristics* and *Product Quality* do not have an immediate

counterpart in the terminology above. The *Storage Recipe* corresponds most closely to an assignment, although it can also be viewed as a plan, a design solution, or a configuration. Although *assignment and scheduling* sound like a good choice, the models for this type of task concern allocation of resources to tasks in a schedule. This does not correspond to our task. *Planning* is a better term. The inputs of the most general model for *planning* (Valente, 1994) are: *initial state*, *goal state*, *world description*, *plan description* and *plan model*. A plan is an ordered set of actions that starts in the initial state and ends with a state that satisfies the requirements of a goal state. The world knowledge describes general information about the world in which the actions will take place.

In our example, *Fruits Characteristics* and *Product Quality* can be viewed as "initial state". However, the storage recipe does not involve discrete states and therefore a planning process is problematic. Even when the process is somehow made discrete, there are many possibilities and the *goal state* provides little guidance for the evaluation of intermediate states. Another problem is that if we compare this to the available resources in Table 6.2, we see that the resulting model is not coherent. The *Fruits Characteristics* and *Current Quality* are not the description of the *initial state* parameter of the planning operators. The approach outlined in Breuker and VandeVelde (1994) does not tell us what to do in this situation. An obvious step is to apply the whole approach recursively to the task of finding the input of the planning operators from *Fruits Characteristics* and *Current Quality*. We shall not pursue this here. It is noted that the planning model cannot actually be applied because of the continuous character of the operators and the process, which is not mentioned in the description of the planning model as a prerequisite. Moreover, the analysis process is about the same as that of our approach. This is because the data-flow structure of the available knowledge is of much more importance at this stage than the structure of the data and the knowledge. Our approach postpones the choice between discrete models and continuous models until later and only then selects a modeling technique.

The situation would be more complicated if the available resources do not map well onto functional components in the solution to the acquisition problem. Suppose that for the fruit storage problem a human expert would have been the only resource. Direct elicitation without decomposition would then probably be impossible but there would be nothing on which decomposition could be based. This would make it necessary to either somehow elicit the data-flow structure or to alternate the construction of the data-flow structure with the acquisition of the components.

Compared with inductive engineering methods our methodology has a broader scope. MEDIA includes the identification of resources, then accounting economic factors, and the structuring of the acquisition problem. Machine learning technology plays a specific role in the overall method. A straightforward inductive approach to the fruit storage problem would probably have been more expensive and less successful. The reason is in the complexity of the relation between the "raw" data about a batch of fruits, its destination, the recipe and the data collection costs.

The presented approach requires information about available expertise and data sources. Dependent on the actual process and the complexity of the system, the information on available sources can be obtained step-by-step during the planning process or entirely beforehand. In the former case the information is always dedicated to the processes at hand, but the planning process is limited to a greedy strategy. If the information is obtained beforehand, an exhaustive search for the potentially best available global plan is possible, but the disadvantage is that not all information collected may be useful. Whether the information is obtained during or before the planning process, these two options represent a less

expensive planning process with the chance for sub-optimal results, compared to a more (time consuming and extensive) planning process that guarantees better and/or cheaper results.

The method outlined in this chapter can be extended to include the expected gain of having the resulting system. This inclusion would generate a more comprehensive model that includes both the costs of knowledge acquisition and the cost of maintaining and operating the resulting application. See Van Someren et al. (1997) for a model of machine learning methods that include costs of measurements and costs of errors, in the context of learning decision trees. These two models can be integrated into a single model, see for example DesJardins (1995) for a similar model for robot exploration.

6.5.3. Suggestions for further work

The main point for further research is to have a better understanding of the relation between knowledge resources as listed in Table 6.2, acquisition costs, and accuracy. This requires cost and accuracy metrics to be defined for both knowledge elicitation and machine learning methods. In software engineering, several of these measures have been developed (Sommerville, 1996). To the best of our knowledge comparable metrics are not yet available for knowledge elicitation. Chapter 7 introduces an approach to support selection of inductive techniques based on the estimated suitability of inductive techniques for a data set. For knowledge elicitation a similar approach can be envisaged. Recent work in machine learning has resulted in quality estimators for machine learning performance on specific tasks. Costing models for both inductive and knowledge elicitation components are in their infancy. Some work on predicting effort of software development projects has been reported, even with the use of machine learning approaches (e.g. Boetticher, 2001; Menzies, 2001). The translation and extension to machine learning and knowledge elicitation methods however still has to be made. This would require substantial amounts of project data.

Without the availability of empirical models for cost and quality forecasts, we have to rely on estimated accuracy and cost data as used in this chapter. In practice, these estimates are based on the experiences of the knowledge engineers and machine learning analysts. A first attempt to adjust the existing software engineering practice of Cocomo (Boehm, 1981; Boehm et al., 1997) has been presented in the form of qualitative models for ML and KA component development. Further empirical studies are required to calibrate available parameters.

7. Guarded Selection of Inductive Techniques

Material of this chapter has been published as F. Verdenius, Entropy Behaviour for Selection of Machine Learning Techniques, in: H. Blockeel & L. Dehaspe (eds.), *Proceedings of Benelearn 1999*, Maastricht, 113-120, and as F. Verdenius & M.W. van Someren, Detecting orthogonal class boundaries in entropy behaviour, M. Wiering (ed.), *Proceedings of Benelearn 2002*, Utrecht. Important difference between these publications and this chapter is the use of prototype matching as analysis technique instead of wavelets.

People can have the Model T in any color--so long as it's black.
Henry Ford (1863 - 1947)

Abstract

Machine learning techniques construct models of a particular model class, each with its own bias. One of the important issues in machine learning is technique selection: finding a technique that performs best on a given data set. For any new data set appropriate techniques have to be identified as there exists no universal learning technique that performs optimally for all data sets. Available approaches for technique selection apply heuristic expert rules, meta-learning from previous experiences, or default techniques. Meta-learning (Giraud & Keller, 2002) selects techniques based on data characteristics, by empirically relating data characteristics to the performance of machine learning techniques.

This paper introduces *guarded technique selection*, a principled approach to technique selection. Every technique is supervised by its own dedicated guard. A guard checks for compliance of patterns in the data with the requirements of its technique. If a guard ascertains compliance, the technique is indicated for application and the model can be applied.

We define a guard for techniques that perform best on data with orthogonal class boundaries. An orthogonal class boundary can be detected in the entropy behavior of attributes of the data set as a sharp *cusp*. The proposed guard detects these cusps by matching prototypical curves onto the entropy behavior of attributes. The fit per attribute is summed over all attributes of a data set to get a global *cuspiness* measure. Using generated data, we show that the cuspiness of entropy behavior can be used to assess the relative suitability of techniques for orthogonal decision nodes. The approach leads to improved selection of concept learners.

Our measure can enhance existing advisers for machine learning techniques, such as the MetaL toolbox. Moreover, it may help to solve existing questions in the machine learning domain. Testing the method on a number of UCI data sets teaches that, apart from concept type, data distribution is a crucial feature for the algorithm. This results in a check on data distribution as an extension of the algorithm. Instead of using a standard prototype definition, the prototype should be formulated to match the actual distribution of data points in the data set at hand. In the end the principle of guarded technique selection requires that for every class of machine learning techniques, the underlying principles are formulated in feasible guards.

7.1. Introduction

An important step in designing real world machine learning applications is technique selection. It is the main subject of the Analysis level of the MEDIA model introduced in Chapter 5. Schaffer (1994a) has shown that there can not exist a universal learning technique

that performs optimal for all data sets, and consequently for any new data set appropriate techniques have to be identified. Machine learning techniques construct models of data. The quality of a learned model depends on the appropriateness of the model class of the machine learning technique for representing the patterns in the data set (Brodley, 1992). For many years the credo of machine learning users appeared to mimic Henry Fords attitude towards the color of a T-Ford: "I can learn any pattern in the data, so long as I can make it fit the model class of my technique". A survey in the mid-nineties identified a limited number of strategies for model class selection (Verdenius & van Someren 1997). The most widely accepted approach was to rely on a limited number (mostly 1 or 2) of default techniques. The problem solving strategy was to adapt, transform or reformulate the problem (i.e. the data set and/or the classification task at hand) to squeeze the best possible performance out of the technique-data combination. Van Someren (2001) refers to this as *the Procrustean approach*. A minority of the identified applications used a comparative explorative strategy. In this approach, a number of available techniques is applied to the same problem, evaluated and the best performing technique is actually used in the application. Comparison of the performance is based on cross validation (Schaffer, 1994b).

A data set is a sample of the underlying distribution. The classification model which the machine learning technique has to learn is a partial estimate of the underlying class distribution that is normally unknown, and can not be used to support technique selection. Consequently, the desire to apply machine learning techniques leads to a deadlock situation. To select suitable techniques, one needs to know the class distribution, but to estimate the underlying class distribution one needs a suitable technique.

It is broadly reckoned that a machine learning technique performs best on data sets that meet specific characteristics of that technique. One of the conclusions that came out of the StatLog project (Michie et al., 1994) was that for specific (groups of) data sets specific techniques performed best. One example is the relative strong performance of nearest neighbor techniques on image data sets. The principles that underlie the nearest neighbor approach however have not been related to the apparent success of this approach to image analysis, and even within the image analysis data sets of StatLog there are a few examples where nearest neighbor performs poor.

This paper explores *guarded technique selection* as a method for technique selection based on the link between data characteristics and technique constraints. It focuses on the relation between the technique representation and the class distribution to learn. The assumption of this work is that differences in learning performance between machine learning algorithms on a single data set should be analyzed in terms of the assumptions that the algorithms make on the underlying class distributions in the data. This is amongst others expressed in the work of Langley (1996) when he distinguishes technique families for the induction of several types of concepts, such as *logical conjunctions*, *threshold concepts*, *competitive concepts*, *decision lists*, *inference networks* and *concept hierarchies*. These concept types primarily refer to the representation languages for machine learning techniques. In distinguishing them the relations to the underlying class distributions are obvious. The concept types are not mutually exclusive, the members of one technique family induce concepts from data sets based on the same mapping principles; all univariate decision trees, for example, recursively partition data sets on single attribute tests of the type $a_i < c_j$ (with a_i is an attribute, and c_j an attribute value for a_i). In the next section, we discuss existing approaches for technique selection, and identify some of their limitations. Section 7.3 introduces *guarded technique selection* as an approach that may overcome existing limitations in technique selection. Section 7.4 describes the effect of the type of class boundaries on entropy behavior, and how this can be used to assess the type of underlying concept. In addition prototype matching is introduced as the

instrument to analyze entropy behavior. Section 7.5 presents experimental evaluation of the approach. Section 7.6 draws conclusions, and discusses potential applications of the tool in the various machine learning approaches and technique selection.

7.2. The Context of Technique Selection

Automated technique and model selection has been for a long time a research subject for machine learning scientists. The goal is to remove human designers from the learning process, and to enable automatic technique selection and operation (e.g. Brodley, 1992). In search of automated technique selection, different lines of research have been described in the literature:

- **Heuristic expert rules.** The EU project Machine Learning Toolbox (MLT, ESPRIT II P2154; Kodratoff et al., 1994) aimed at delivering a suite of machine learning techniques. A suite of 48 techniques is developed within the MLT project. The toolbox is accompanied by an advisory tool, CONSULTANT (Craw et al. 1992), containing expert heuristics to support designers in selecting, calibrating and applying 10 of the techniques in the suite to a wide range of learning tasks. A typical rule to generate an advice for applying C4.5 (Quinlan, 1993) would be (Kodratoff et al., 1994):

  ```
  IF ML application goal = classify AND
     There are several classes in parallel
  THEN
     Use c4.5
  ```

  The heuristics formalize the reasoning strategy of a human expert of that day. They offer grip and structure to novices in analyzing new machine learning problems. The knowledge base of CONSULTANT is obtained by analyzing human expertise, and filling a rule based expert system with the resulting knowledge. That knowledge focuses on two aspects of machine learning application. On the one hand there is the qualitative relation between the task and descriptive data characteristics, e.g. ML application goals, data availability, data format, data quality (Kodratoff et al. 1994). On the other hand there are technique characteristics such as the way a technique uses background knowledge or post-learning user interaction (Kodratoff et al., 1994). Consequently, a typical CONSULTANT advice does not take into account the more fundamental data characteristics, neither does it exploits the relation between technique performance and the actual data.
- **Meta-learning**. Meta-learning uses experimental experience on the relation between data characteristics and technique performance to derive empirical strategies for technique selection. In meta-learning the performance of machine learning techniques on data sets is generalized into technique selection knowledge. Two European research projects, StatLog (Michie et al., 1994) and MetaL (MetaL, 2004) have contributed substantially to the development of meta-learning. The StatLog project surveys the performance of a broad spectrum of classification techniques from statistics, machine learning and neural networks on a variety of data sets. The aim is to develop knowledge on the relation between data set characteristics and technique performance. Meta-learning, using learning techniques to generalize from technique performance data to applicable heuristics, is one of the instruments to interpret the experimental results (Brazdil et al., 1994). The resulting heuristics enable users to select techniques. The MetaL project has further explored the meta-learning approach, bringing it to on-line support. The heuristics form the basis for a ranking tool (MetaL, 2004). The MetaL ranking tool predicts the prospective ranks of various commonly used machine learning techniques and their potential performance.

Data-sets are characterized by a large number of descriptors. Typical descriptors used in both the StatLog and the MetaL project include:

- *Descriptive measures*: e.g. the number of records in the data set, the number of attributes,
- *Statistical measures*: e.g. test statistic for uniformity of covariances, mean absolute correlation coefficients,
- *Information theoretic measures*: e.g. entropy of classes, joint entropy of class and attribute.

By running 22 techniques on 29 data sets, and relating the performance of these techniques to the data descriptors, StatLog aims to deliver technique selection knowledge. Although the resulting rules have some predictive value, there are at least two issues that affect its applicability. First, the rationale of the delivered rules is questionable. An example of a resulting rule given by Brazdil et al. (1994):

```
CART-Appl      ← N ≤ 6435, skew > 0.57
CART-Non-Appl  ← N > 6435
CART-Non-Appl  ← N ≤ 6435, skew ≤ 0.57
```

Here, N is the number of records in the data set, and skew is the mean skewness of the attributes in the data set. This rule is a generalization of the *22 specific* techniques on *29 specific* data sets. It is, however, not a universal rule for applying machine learning techniques to real world data sets. After all, the boundary of 6435 strongly depends on the (accidental) composition of the data set collection that is applied here. It is obvious that another composition of the data set collection would result in other rules. It is, however, not obvious what the composition of the collection of data sets should be in order to derive generally applicable heuristics. The population $\mathbb{D}$ of possible data sets is infinitely large. Variables such as the number of records n, the number of attributes k, the distribution of attribute types (discrete, continuous) and many more would span a huge space. First, Even when it would be possible to draw a sample from $\mathbb{D}$ that is representative for the data sets that can be expected in practice, N might appear not be a determinant in selecting techniques. And in itself, the idea of 'drawing a representative sample from $\mathbb{D}$' is already problematic, because one first has to define the characteristics of $\mathbb{D}$. Second, and more fundamental, the relation between the techniques applied in MetaL (the dependent attributes) and data descriptors (the independent attributes) used in MetaL is unclear. The set of descriptors that is used is large and contains very different types of measures. In other words, a data set with a large number of attributes sparsely covers the data space. The resulting model has the risk of being overfitted and of representing spurious patterns. The appearance of N in the mentioned rule illustrates this risk.

Various researchers have proposed other, more relevant data descriptors (e.g. Engels & Theusinger, 1998; Lindner & Studer, 1999). Although some of these descriptors partially meet the above objections, it remains a principal point that meta-learning approaches aim to generalize over an unknown space spanned by technique and data set that is only sparsely covered by the available technique and data set instances.

- **Landmarking**. Another way of using previous experience on technique performance is landmarking, as described by Pfahringer et al. (2000). In landmarking, the performance of a limited group of simple and efficient learners (the so-called landmarks) on the target data set is used to predict the performance of a broader suite of machine learning techniques on that data. This offers both drawbacks and advantages compared to meta-learning. The major advantage lies in the chance to select landmarks that exploit the same underlying principles that are used by the learning techniques one wants to select from. On the other

hand, when the suite of techniques covers various underlying principles, for instance by containing techniques that exploit specific types of class boundaries, nearest neighbor techniques, multilayer perceptrons or bayesian learners, one may need a larger number of landmarks. And this would require substantial pre-analysis in order to determine the necessary set of landmarks. Moreover, as in meta-learning, landmarking does not result in insight on how the universe of data sets maps onto the universe of learning techniques.

A common factor of these approaches is that they use a selection criterion that is based on general-purpose data descriptors. In MLT, the cornerstone is that the expertise of human analysts can be transferred into a knowledge base. StatLog and MetaL apply empirical knowledge on technique performance, based on descriptive data characteristics and a distance-based comparison of data sets, respectively. Historically, the approaches represent the main steps in the methodological thinking in the application of machine learning techniques. Initially, designers use one technique in the Procrustean way. Next, technique selection heuristics were formulated based on increasing experience. And finally large scale experimentation is used to extensively explore, complete and refine the heuristic approach. Each of the approaches makes a contribution to automatic model selection. There is however one last step to be taken. As suggested by Brodley (1992), the quality of a learned model depends on how appropriate the model class of the used learning technique is able to represent the patterns in the data set. If this statement is true, and many researchers consider it as such, a further goal for research on technique selection can be derived from it:

Ascertain that the model class of the used technique fits the patterns that are to be learned.

This requires for each group of machine learning technique(s):

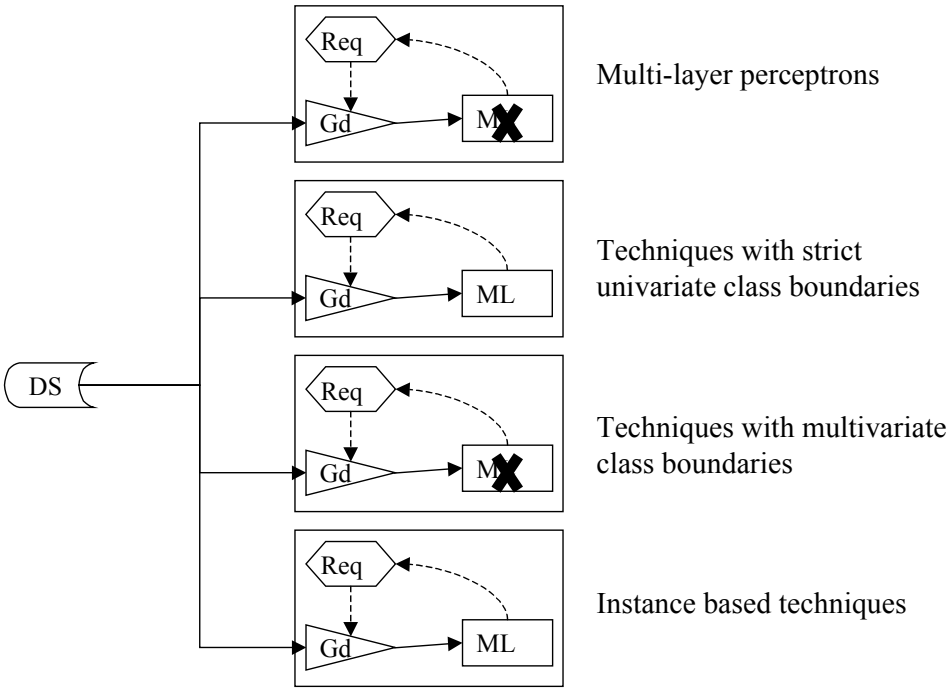


Figure 7.1. Guarded technique selection. A data set (DS) has to be analyzed. Each technique guard (Gd) assesses the applicability of 'their' technique (ML) for the data set by assessing the degree of compliance with the technique requirements (Req). Guards use a definition of the required underlying class distributions (hexagonal) that is dedicated to one technique.

- A description of the model class it delivers, in terms of data set requirements. In other words, one needs to know the patterns in a data set for which the technique is or is not applicable.
- An instrument to assess whether a data set complies with the requirements of that technique group. The instrument should take the data set as input and deliver the degree of compliance.

Preferably, there is a norm on compliance/non-compliance on this *degree of compliance*, to enable the selection of techniques for a data set on the basis of 'best compliance.' This would imply an absolute guard that predicts which technique gives the best results.

7.3. Guarded Technique Selection

Figure 7.1 gives an overview of guarded technique selection (GTS). A guard contains dedicated knowledge on the requirements that a specific class of learning technique(s) poses on data for it to be applicable. Every technique class has its own guard that checks the requirements that the technique class poses to data in order to be successfully applied. As a consequence, only those techniques are applied that one can expect to have an acceptable result.

Figure 7.2 schematically illustrates how a guard serves as an outpost for the ML technique. The requirements are extracted from a technique class and formulated as a check on data. In the ideal case, developers of a new technique provide these requirements as part of their technique. The guard is constructed in order to check for these requirements, resulting in a relative suitability measure for that technique. To illustrate the functioning of guarded technique selection, we develop in this paper one such guard for techniques that learn by applying univariate partitions on data. The guards will have to be developed for other groups of techniques.

Guarded technique selection is logically connected to our opening statement that the quality of a learned model depends on how appropriate the model class of the used machine learning technique is able to represent the patterns in the data set. After all, if this statement is true, the next step is to check, given a specific technique and the data set to analyze, whether the patterns in the data set fit the model class of that technique.

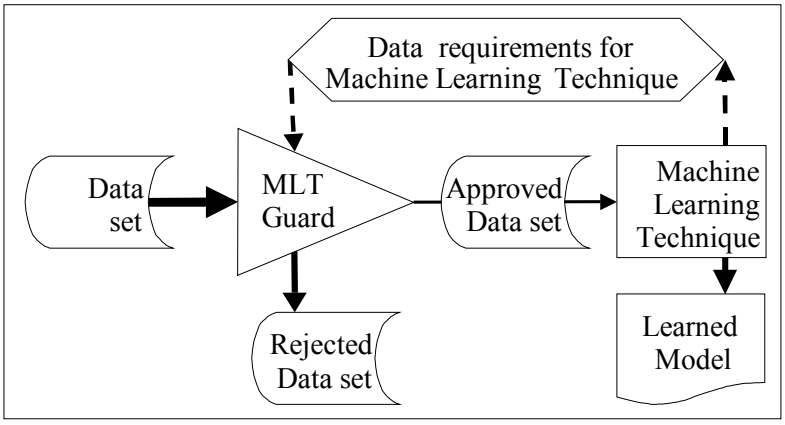
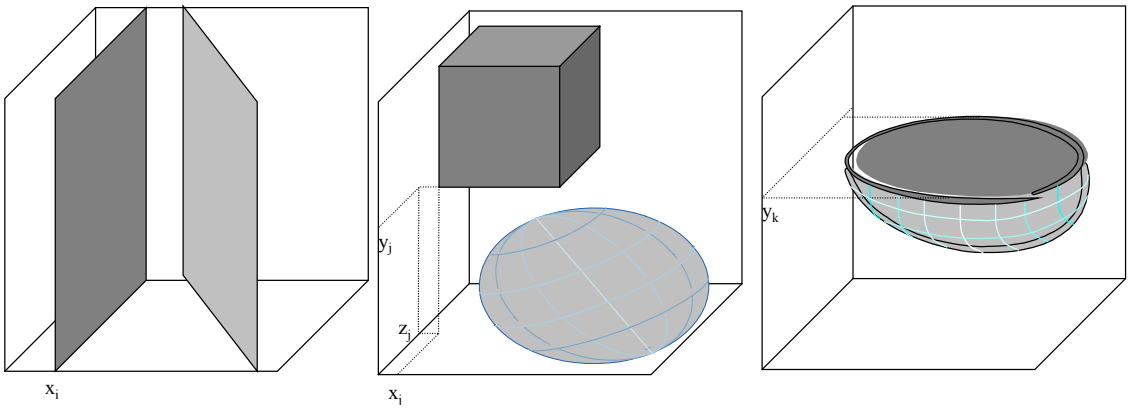


Figure 7.2. A schematic overview of a guard as an outpost for the ML technique and the accompanying data flows. The guard (triangle) expresses the degree of compliance of the data set with the requirements (hexagonal) for the technique that it guards. If the degree of compliance is high enough, the learning technique is applicable, and a model is derived from the data.



(a) (b) (c)

Figure 7.3. Orthogonal (dark) and non-orthogonal class boundaries (light) in R^3. Simple separating planes (a) 3-D sub-spaces (b) and complex subspaces (half elliptical sphere) (c).

Guarded technique selection takes the view that every technique class has its own guard. This raises questions on feasibility: Is it possible to define a guard for every technique? And doesn't this approach threaten the overall efficiency? The efficiency of learning may be influenced when guards become very computationally intensive. It is then requisite to keep guards as simple as possible. On the other hand, there is also a gain in efficiency when guards prevent a number of techniques to be applied. Nevertheless in return for a decrease in computational efficiency, there may be an increase in comprehensibility of the performance of a technique: a technique can be expected to perform well if its requirements are met. GTS can be compared with the existing statistical practice, where tests and techniques are appropriate only within a specific context (e.g. the PMC correlation presupposes a linear context; it can be obtained for non-linear data, but the value is limited). GTS tests combinations of data sets and machine learning techniques on appropriateness by testing whether requirements of specific techniques are met in the data set.

7.4. Entropy Behavior as Indicator for Orthogonal Class Boundaries

7.4.1. Orthogonal class boundaries

Our goal is to propose a guard for those techniques that perform best when orthogonal class boundaries are present in the data. An orthogonal boundary is described by a constant value for one attribute. In other words, an orthogonal boundary in $\Re^n$ is defined as $X_i=c$ for exactly one i, i.e. a boundary that stands orthogonal to one axis. An orthogonal subspace can be bound by a conjunction of orthogonal boundaries ($X_1 = c_1$ & $X_2 = c_2$ etc.). It is important to understand that the concept of *orthogonality* of a class boundary is something else than the *predictive capacity* of a class boundary. In Figure 7.3a, the dark gray plane is defined by $X=x_i$ and is orthogonal. The light gray plane is a linear combination of two attributes. Suppose that both planes serve as a class boundary (in two different data spaces). Then the orthogonality of the light gray plane is much lower than the orthogonality of the dark gray plane, but their predictive capacity for class membership is the same. Furthermore, class boundaries may be combinations of a number of bundaries. In Figure 7.3b, the dark gray subspace can be described as a logical combination of univariate class boundaries, for instance in the form of a decision tree (Mitchell, 1997); concequently, it can be described as a combination of orthogonal boundaries, and thereby is an orthogonal class boundary. And the flat section of the half sphere in Figure 7.3c can be described as a univariate boundary orthogonal $X_i=y_k$. We call techniques that perform best on orthogonal class boundaries in

data sets *orthogonal techniques*. Examples of orthogonal techniques are the univariate concept hierarchies (Langley, 1996) and rule post-pruning on decision trees (Mitchell, 1997). The guard we aim for should identify orthogonal class boundaries in a data set. The guard should detect all dark gray planes in Figure 7.3 as orthogonal, and the light gray boundaries (linear (a) or nonlinear (b&c) multivariate combinations of attributes) as non-orthogonal. Consequently, the six planes of the cube in Figure 7.3b represent 6 separate class boundaries, that are all orthogonal.

This is based on a classification of class boundaries (Figure 7.4). Orthogonal class boundaries are class boundaries that can be described as (combinations of) univariate descriptors. Many machine learning techniques learn concepts by combining orthogonal class boundaries. This includes techniques that learn univariate logical conjunctions and univariate decision trees. Successful application of techniques from this group requires the data set to have underlying class boundaries that are (close to) orthogonal to attribute axis.

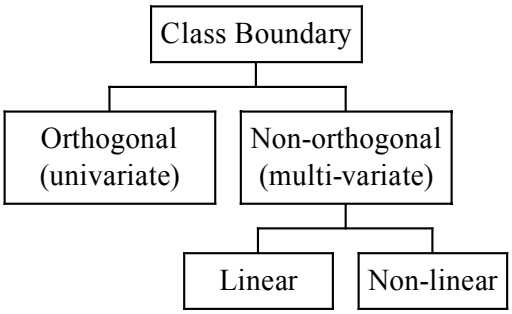


Figure 7.4. Taxonomy of class boundaries

We call all other class distributions *non-orthogonal* because they suggest a class boundary to be some combination of attributes. Non-orthogonal class boundaries can be further separated into linear and non-linear class boundaries. We call techniques that perform well on non-orthogonal class boundaries *non-orthogonal techniques*. In the rest of this paper, we will distinguish orthogonal class boundaries from linear non-orthogonal class boundaries. This work focuses on data sets with continuous valued numeric attributes. Other data types fall outside the scope of this work. For each attribute, many different values are assumed to be present; in practice, this means that we need larger data sets with more records. Later in this text we define a lowerbound for the number of values per attribute that is required.

Running Example: **Quételet's Body Mass Index**

Throughout the chapter the concepts of GTS are illustrated with a running example: the Quételet ratio for the body mass index. Physicians use the Body Mass Index (BMI) as a measure for overweight (Consumed, 2004). BMI is also known as Quételet index Q_p, named after the statistician Adolphe Quételet (1796-1874). The Quételet index Q_p of a person p is the ratio between the mass m_p and squared length l_p of that person:

$$Q_p = \frac{m_p}{{l_p}^2}, \qquad \textbf{(7.3)}$$

It is generally accepted to call people

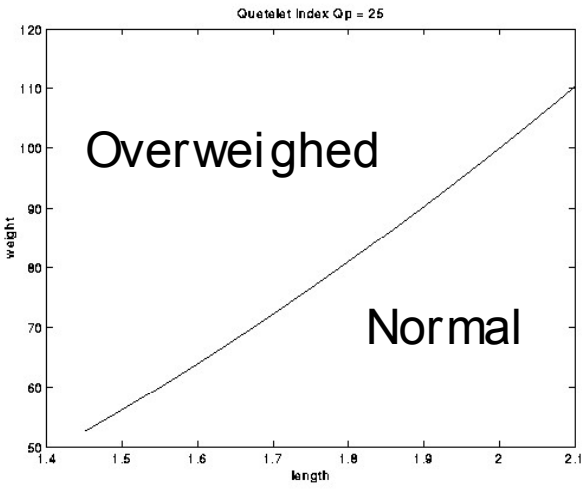


Figure 7.5. The Q_p=25 class boundary. Length-weight combinations above the solid line represent overweighted individuals.

"overweighted" when their $Q_p \geq 25$[22]. Figure 7.5 shows the 2-dimensional length-weight plane, with the separator at $Q_p = 25$. Suppose that, without knowing the BMI concept and the formal classification, we are interested in classifying people as overweighed, based on only a few attributes, and that we have 2 data sets, each containing the data of 1000 persons. Cases are uniformly distributed over each attribute. Both sets contain three attributes. In data set 1 this is a random variable with no relation to Q_p. In data set 2 the third attribute is the Q_p value. For each case, we have the judgement of a physician, *Normal* or *Overweighed*, available. Consequently, data set 1 has no orthogonal class boundaries, while the attributes lenth and weight define a non-orthogonal class boundary. Additional to the length-weight class boundary of data set 1, data set 2 also has an orthogonal class boundary for data set 2 at $Q_p=25$.

7.4.2. Definition of Entropy behavior

The entropy $H(X)$ of a discrete random attribute X is defined as

$$H(X) = -\sum_{x_i \in X} p(x_i) \log p(x_i) \qquad (7.4)$$

with the log to the base 2, and $p(x_i)$ being the probability of $X=x_i$:. The entropy over the class distribution of a data set is often used in machine learning techniques to estimate the predictive value of a "split" of an attribute into intervals, for instance in decision tree learning (Mitchell, 1997, ch. 3, e.g. C4.5: Quinlan, 1993) and rule learning (Clark & Niblett, 1989). The attribute-value combination that gives the highest reduction of entropy is considered most predictive (it gains most information), and is selected as decision node in a decision tree.

Suppose we have a data set D, consisting of pre-classified records $r = [a_1, a_n, c]$. Attribute values $a_i \in \Re$, and the class label c being one of k class labels: $c \in \{c_1, \ldots, c_k\}$. Now the class distribution of data set D with k elements serves as a random attribute. Let H_{tot} be the entropy over the class distribution of the total data set with k_D instances. When partitioned at $A = a_i$, the data set falls apart in two subsets $D_{A \leq ai}$ and $D_{A>ai}$. For these subsets, the partial entropies are $H(D_{A \leq ai})$ and $H(D_{A>ai})$. The distribution of information gain over the entire range of an attribute is analyzed to distinguish orthogonal class boundaries from non-orthogonal class boundaries. If the number of instances in the data sets is $k_{A \leq ai}$ and $k_{A>ai}$ respectively, the **entropy behavior** B_A at a potential partition point $A = a_i$ is defined as:

$$B_A(a_i) = H_{tot} - \left(\frac{k_{A \leq a_i}}{k_D} H(D_{A \leq a_i}) + \frac{k_{A>a_i}}{k_D} H(D_{A>a_i}) \right) \qquad (7.5)$$

Entropy behavior of an attribute, i.e. the course of information gain over an attribute axis, is a by-product of information-gain based decision tree learners. In the greedy process of finding the most suitable partition point, a decision tree learner calculates the information gain for all attributes and for all potential partition points. Entropy behavior of an attribute is the function that couples an attribute value to the information gain when partitioning the data set at that value.

7.4.3. Class boundaries and entropy behavior

[22] The formal classification over Q_p includes four classes: underweight: $Q_p < 20$, normal weight: $20 \leq Q_p < 25$, overweight: $25 \leq Q_p < 30$, obese: $Q_p \geq 30$. Sometimes the lowerbound is set to 18.5

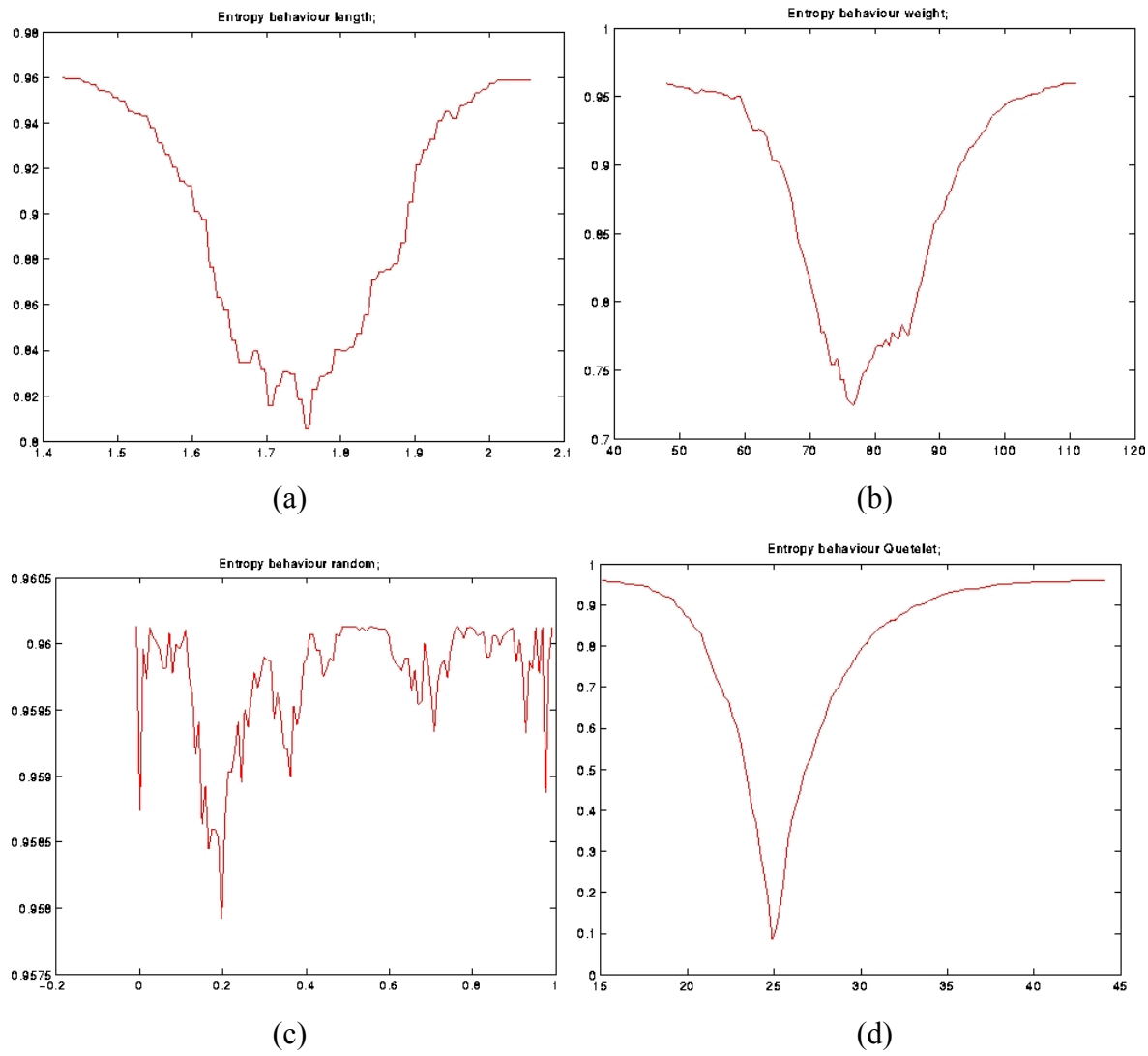


Figure 7.6. Entropy behaviors of length (a), weight (b), the random attribute (c) and Q_p (d) respectively. Note the differences in the scaling of the entropy value (vertical axis, representing the B_A).

The guard has to identify class boundaries of the form $A_i = a_c$. In other words, if a constant value of an attribute in a data set (at least over part of its range) coincides with a class boundary, the guard should signal that. Typical representations that are suited for such class boundaries include univariate decision trees and univariate decision rules.

Figure 7.7 shows the class boundaries, class density functions, entropy behaviors and decision trees for an orthogonal class boundary at $A_i = a_c$ and for a linear non-orthogonal class boundary over the interval $A_i = [a_{i,1}, a_{i,2}]$, with different combinations of $a_{i,1}$ and $a_{i,2}$. Figure 7.8a shows the expected entropy behavior for $a_c=0.5$ for three different noise levels. The analytical derivation of these functions is given in the appendix 1. If we assume a noise-free orthogonal class distribution with the class boundary at $a_c=0.5$ and a uniform data

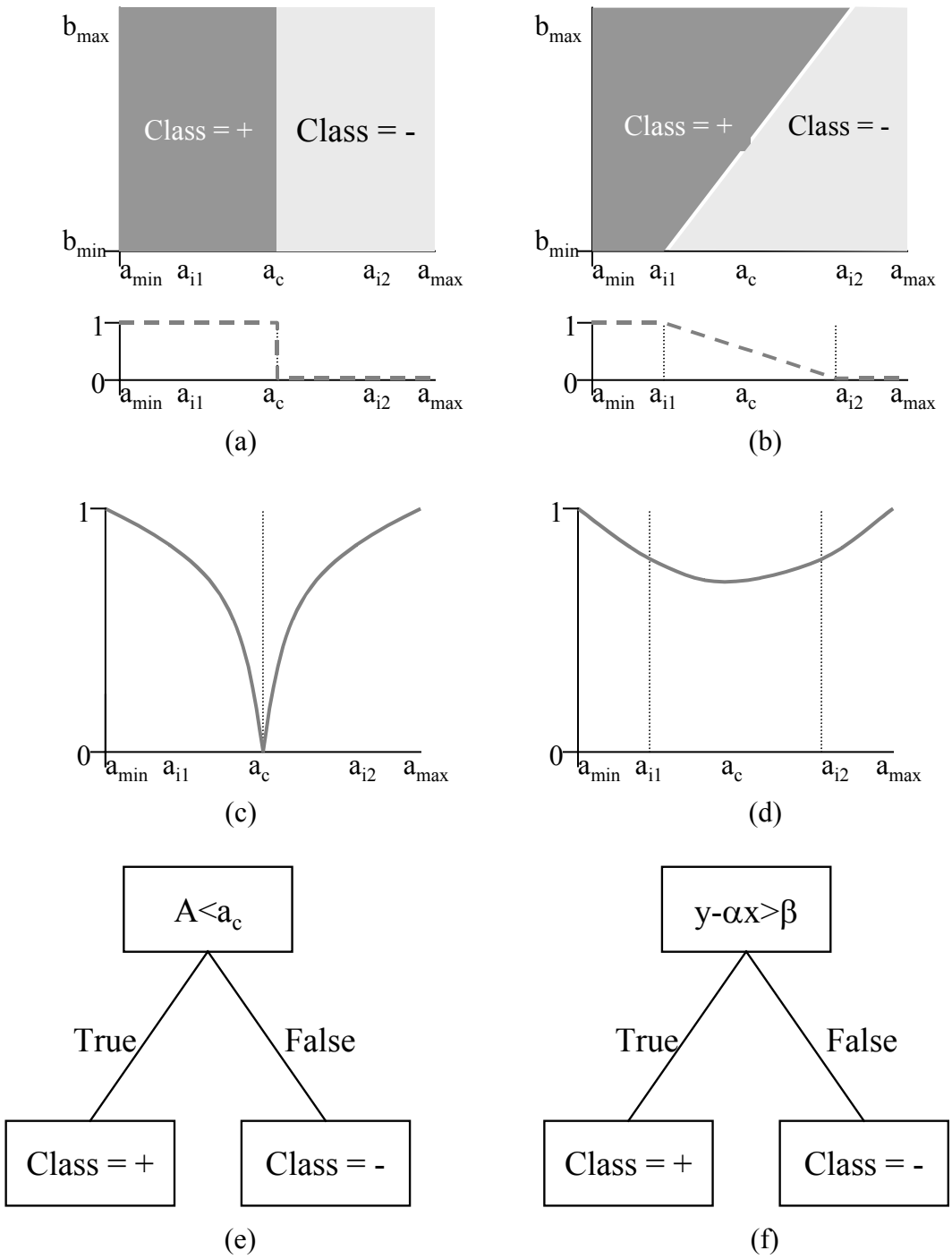
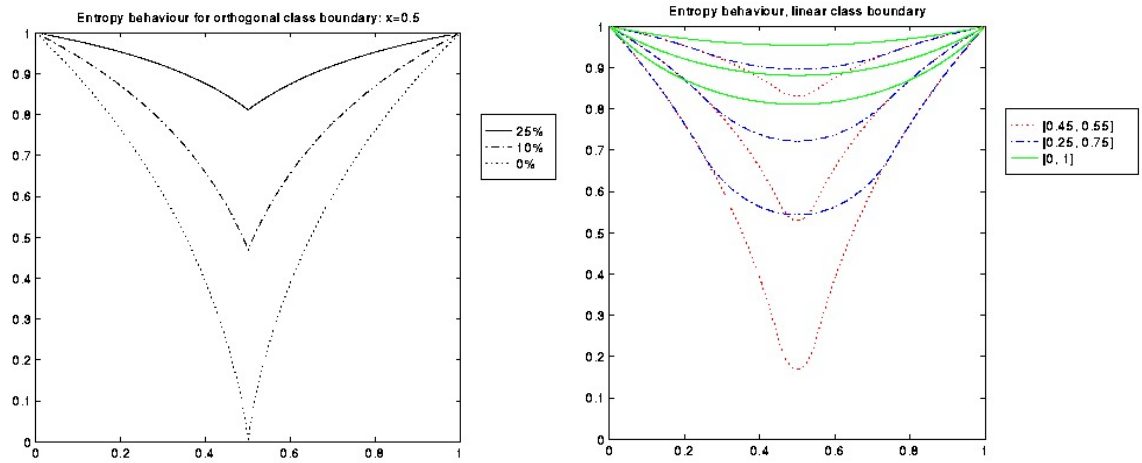


Figure 7.7. The relation between class boundary, entropy behavior and concept type for an orthogonal and a linear concept (a) the density of the class distribution for an orthogonal class boundary and (b) a non-orthogonal class boundary (c) the entropy behavior for the orthogonal class boundary (d) the entropy behavior of the linear class distribution (e) the decision tree for the orthogonal class boundary (f) the decision tree for the linear class boundary.
The orthogonal concept of (a) leads to a strict class boundary for the A attribute, the linear concept of (b) leads to a non-strict boundary.



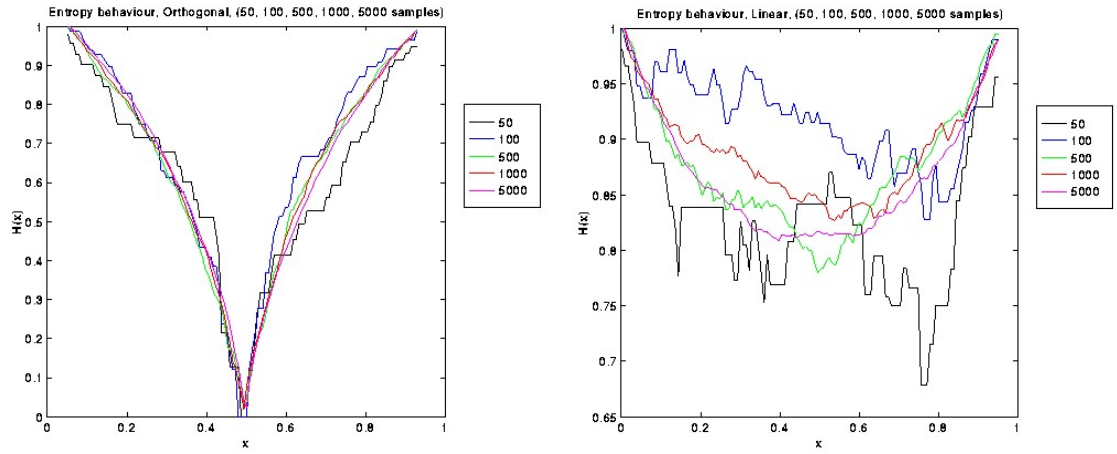
(a) (b)

Figure 7.8. Theoretical entropy behavior at noise levels 0%, 10% and 25% on the interval [0, 1] and a_c=0,5. (a) For the orthogonal class boundary of Figure 7.3a, and (b) for linear non-orthogonal class boundary of Figure 7.3b. The transition area [$a_{i,1}$, $a_{i,2}$] has been taken to be the intervals [0.45, 0.55] (red dotted lines), [0.25, 0.75] (blue dashed lines) and [0,1] (green line) respectively. The curves are based on the derivations in the appendix.

distribution, we have complete class information at that point, and consequently $H(A_I=a_c) = 0$. The minimum entropy value increases for increasing noise levels, but there is a sharp minimum at the class boundary for all noise levels. Figure 7.8b shows the entropy behavior for the linear class boundary of Figure 7.7b, over three different transition intervals $[a_{i,1}, a_{i,2}]$. There is never total class information for a linear class boundary, and consequently the minimum entropy value does not reach as low a value as in the orthogonal case. The actual minimum entropy value depends on the size of the transition interval. As in the case of an orthogonal class boundary, there is a minimum value at $a_{i,1}+½(a_{i,2}- a_{i,1})$, and again the actual minimum value depends of the noise level. There is, however, no sharp cusp at the minimum. Verdenius (1999) already note that other non-orthogonal class boundaries (for instance where the class boundary is described by a sigmoid function of the attributes, $a=1/(1+e^{(1+b)}$ in a two-dimensional data space) produce entropy behavior that is quite similar to that of the linear class distributions. In the rest of this paper, linear class boundaries are therefore taken as the standard to compare with orthogonal class boundaries.

Orthogonal class boundaries are characterized by convex shoulders and a steep cusp in the entropy behavior at the position of the orthogonal class boundary. The cusp is characterized by a discontinuity in the first derivative. Linear class boundaries (and other gradual class boundaries) result in an entropy behavior that is concave, and lacks a sharply marked minimum.

For new, previously unseen data with an unknown underlying class distribution, the analytical form of B_A is unknown. However, it can be approximated in case an attribute contains sufficient potential partition points $A=a_i$, that is, if the attribute counts a substantial number of different values. An approximation of the entropy behavior for an attribute A_i in a data set is obtained by dividing the value range for A_i in a large number of equidistant values a_i. For each a_i the corresponding $B_{Ai}(a_i)$ is calculated. In this paper we generate entropy behavior in 128 data points. In case of an uneven distribution over the axis, percentile points can be used as partition points instead of attribute values to create uniform behavior.



(a) (b)

Figure 7.9. Effect of data sample size on smoothness of entropy behavior (a) for an orthogonal class boundary (b) for a linear class boundary. In all cases entropy behavior is determined as a 128-point entropy sample. A small data sample size (black and blue curves, sample size $\in$ [50,100]) results in a bumpy entropy behavior, a large data sample size (red and pink curves, sample size $\in$ [1000,5000]) results in smooth curves that approximate the theoretical curves of Figure 7.8.

Entropy behavior $B_A(a_i)$ results in a curve with potentially many local extremes. At finer scales ($s \rightarrow 0$) these local extremes reflect sampling noise: random variations that contain no information. The smaller the data set, the more effect this sampling noise has on the entropy behavior. An illustration is given in Figure 7.9, where for orthogonal (a) and linear (b) class boundaries the entropy behavior is shown for sample sizes of 50, 100, 500, 1000 and 5000 records. A small sample size results in a bumpy entropy behavior, a large sample size results in much smoother curves, where the noise induced local extremes have shrinked into small wrinkles. By comparing the expected entropy behavior (Figure 7.8) with the apparent entropy behavior that is derived from a data sample (Figure 7.9), the effect of this sampling noise can be assessed.

Running Example: **Quételet's Body Mass Index**

The entropy behaviors of Figure 7.6 are based on 128 entropy values, with an equidistant subdivision of the attribute space. The entropy behavior for the length attribute, for instance, is derived by taking the minimum and maximum length in the data set (1.42m and 2.05m respectively) and partitioning that interval in 128 equidistant steps.

The entropy behaviors for the attributes *length*, *weight*, the *random* attribute and *Quételet* are given in Figures 7.6a, 7.6b, 7.6c and 7.6d respectively. Length and weight reach their minimum entropy values at approx 1.75m and 75 kg, respectively. The information gain at that point is approximately 16 and 24.5 % respectively. The random attribute of data set 1 shows a maximum entropy reduction of 0.2%, and the Quételet attribute of data set 2 gives a maximal information gain of approx 91%.

The entropy behaviors of length (Figure 7.6a) and weight (Figure 7.6b) show a more or less gradual course towards their respective minima. The entropy behavior of the random attribute (Figure 7.6c) shows random variation with no significant minimum. The Q_p attribute exhibits a cusp-shaped minimum at $Q_p=25$.

7.4.4. Prototype matching

We now know that orthogonal and non-orthogonal class boundaries are reflected in entropy behavior B_A. As illustrated in Figures 7.7, 7.8 and 7.9, there is an intuitive understanding of the relation between patterns of entropy behavior and the underlying class distribution. The task at hand now is to define indicators of orthogonal class boundaries and of non-orthogonal class boundaries in the entropy behavior of real world data sets. It is our hypothesis that indications for orthogonal and non-orthogonal class boundaries can be combined to allow an assessment of whether or not orthogonal techniques are appropriate for a particular data set.

We use the entropy behavior of orthogonal and non-orthogonal class distributions as *entropy prototypes*, and compare them with the actual entropy behavior of a data set. The degree of similarity between entropy behavior and entropy prototypes is determined per attribute to serve as indicators for orthogonal and non-orthogonal class boundaries, respectively. Next, the degrees of similarities for orthogonal and non-orthogonal similarity are summed, and correlated with the performance of orthogonal and non-orthogonal machine learning techniques respectively. In other words, class boundaries are classified on the basis of their orthogonality/non-orthogonality. Based on these norms, the data sets are classified as *appropriate or not-appropriate* for *orthogonal machine learning techniques*, without actually running specific machine learning techniques.

In pseudo code, the procedure of prototype matching goes as follows:

Given: Prototypes E_o and E_l for orthogonal and linear class boundaries
A data set with n attributes a_i

Do
For each attribute a_i
Calculate entropy behavior B_{ai}
For every scale s
Calculate local discrepancies $\delta_l(a_i,s)$ and $\delta_o(a_i,s)$ between B_{ai} and E_l and E_o respectively
Calculate relative weighted orthogonality/linearity per attribute per attribute, by summing $\delta_l(a_i,s)$ and $\delta_o(a_i,s)$ with weight factors per scale, and relating it to the information gain of the attribute.
OR is the ratio between the relative weighted orthogonality and relative weighted linearity, summed over all attributes
End

7.4.4.1. Prototype Matching Basics

We construct prototypes of entropy behavior for both orthogonal and non-orthogonal class boundaries and match them locally with the entropy behavior. As indicated earlier in Section 7.4.3, we take the entropy behavior of a linear class boundary to stand for other non-orthogonal class boundaries.

The entropy behavior is characterized in terms of discrepancy between prototypes and actual behavior. If the entropy behavior in an environment of a local minimum reflects similarities with the orthogonal entropy prototype, an orthogonal class boundary can be hypothesized. If the local entropy behavior shows similarities with the non-orthogonal entropy prototype, a non-orthogonal class boundary can be hypothesized. The entropy behavior for an attribute can reflect, at one attribute value, both characteristics of orthogonal and non-orthogonal behavior (non-exclusive). By deriving a characterization of entropy behavior as orthogonal or non-orthogonal for all attributes of a data set, the data set is characterized as orthogonal or non-orthogonal.

The actual prototype analysis is performed by analogy with wavelet analysis (Mallat, 1999). In an earlier stage, wavelet analysis has been tested as technology for cusp detection (Verdenius, 1999; Verdenius & van Someren, 2002), but a lack of resolution of wavelet analysis to distinguish between orthogonal and non-orthogonal class boundaries in entropy behavior made it necessary to develop protoype matching as an alternative tool. As in wavelet analysis, we take scale effects into account. Iteratively a prototype is dilated to finer and finer **scales** *s* of comparison. By dilating the prototype, local class boundaries (cf. Figure 7.3b) can be detected. Next, the dilated prototype is translated over all attribute values in the entropy behavior in order to cover the entire attribute value range, and scaled to the range of the sub-segment of the entropy behavior to be matched (see Figure 7.10). The translation guarantees that class boundaries are located independent of their location at the attribute axis. Then, the discrepancy with the local entropy behavior is determined (the yellow shaded area of Figure 7.10). The derivation of orthogonal and non-orthogonal prototypes that we used is presented in the appendix 1.

The **discrepancy** $\delta(a_i)$ between an entropy behavior $B_A(a_i)$ and an orthogonal or non – orthogongal **entropy prototype** E in the range [0, 1]is calculated as follows[23]. First E has to be dilated and translated onto the local entropy behavior (see Figure 7.10). Dilating E at a scale s results in E_s with signal length l_s. Fitting it to $B_A(a_i)$ requires two steps. First, E_s is translated over a **distance** a_i, to be centered around the entropy behavior at the partition point a_i. Next, the entropy behavior is scaled to fit the minimum $min_B=min(B_A([a_i-l_s/2,\ a_i+l_s/2])$ and maximum $max_B=max(B_A([a_i-l_s/2,\ a_i+l_s/2])$ of the entropy behavior B_A over the range $[a_i-l_s/2,\ a_i+l_s/2]$:

$$E_{a_i,s}(x)=\frac{E_s(x-a_i)}{(\max_B-\min_B)}+\min_B \qquad (7.6)$$

$$\delta(a_i,s)=\int_{a_{\min}}^{a_{\max}}\left|E_{a_i,s}(a)-B_A(a)\right|da\ , \qquad (7.7)$$

with a_{min} and a_{max} the minimum and maximum attribute value in the data set, respectively. A data set is a discrete sample of the continuous class distribution that leads to (Equation 7.3). Therefore, we apply a discrete form of E and B_A. This leads to a formulation of a discrete discrepancy:

$$\delta(a_i)=\frac{\sum_a\left|\left(E_{a_i,s}(x)-B_A(x)\right)\right|}{(a_{\max}-a_{\min})} \qquad (7.8)$$

7.4.4.2. Analyzing Entropy Minima

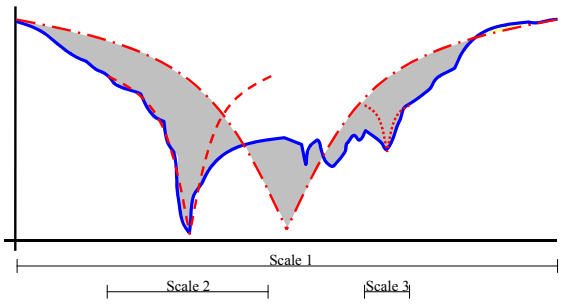
The discrepancy $\delta(a_i,s)$ between entropy behavior and prototype is the integrated difference between the entropy behavior and the entropy prototype, as illustrated in (Equations 7.5 & 7.6). By translating the prototype along the attribute axis A to any value a_i, a discrepancy profile for every (discrete) point of the entropy behavior is obtained. By dilating the entropy prototype with a scale factor s the local behavior is obtained. The result of the discrepancy

[23] E here indicates any entropy behavior prototype. If we specifically refer to a **linear prototype** or **orthogonal prototype**, this is indicated by a subscript : E_l and E_o refer to a linear and orthogonal prototype respectively. These subscripts will be used in other symbols as well.

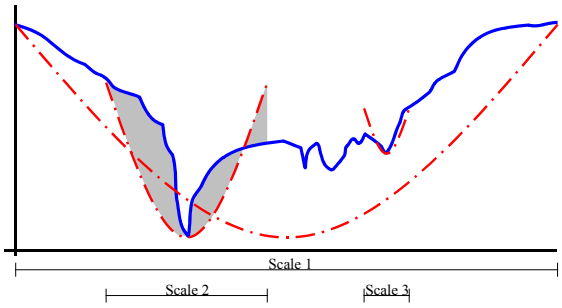
analysis is a two-dimensional **discrepancy matrix**, $f_l(A)$ or $f_o(A)$, that gives a measure of discrepancy for every attribute value a_i at any scale s with the linear or orthogonal prototype, respectively.

We now define the **orthogonality matrix** $M_{ol}(A) = 1$ *if* $f_o(A)<f_l(A)$, and 0 otherwise, so that it is 1 where the entropy behavior shows a dominant similarity with the orthogonal prototype (the index *ol* indicates the orthogonal similarity being smaller than the linear). The linearity matrix $M_{lo}(A)$ is defined analogously, so that it is 1 where the entropy behavior shows a dominant similarity with linear behavior (the index *lo* indicates the linear similarity being smaller than the orthogonal). By summing M_{ol} and M_{lo} per scale, we get the **orthogonality/linearity of an attribute per scale**, sM_{ol} and sM_{lo}, respectively:

$$sM_{ol}(s) = \sum_i M_{ol}\left(a_i, s\right),$$
$$sM_{lo}(s) = \sum_i M_{lo}(a_i, s) \qquad \textbf{(7.9)}$$



(a)



(b)

Figure 7.10. Matching prototypes (dashed curves) on entropy behavior (black curve) (a) 3 scaled prototypes for orthogonal class boundary (b) 3 scaled prototypes for non-orthogonal class boundary. The shaded area represents the mismatch, in (a) over the full scale, in (b) over a sub-scale

Orthogonality at coarse scales represents a global similarity of entropy behavior with the orthogonal prototype. Orthogonality at finer scale represents similarity of parts of the entropy behavior. Similarity at the finest scales represents noise wrinkles. Consequently, in the global assessment of class boundaries, M_{ol} at coarse scales should receive a higher weight than at fine scales. This leads to a **weighted orthogonality** $\boldsymbol{wsM_{ol}}$ and **weighted linearity** $\boldsymbol{wsM_{lo}}$:

$$wsM_{ol}(A_i) = \sum_j \frac{sM_{ol}(s_j)}{2^{s_j}},\ wsM_{lo}(A_i) = \sum_j \frac{sM_{lo}(s_j)}{2^{s_j}} \qquad \textbf{(7.10)}$$

The measures wsM_{ol} and wsM_{lo} apply to individual attributes A_i. For attributes with a high entropy reduction, the orthogonality represents a more relevant characteristic than for an attribute with only a minor entropy reduction (e.g. a random attribute, which only provides noise). In this way, the contribution of wsM are adjusted according to the predictive ability of the attributes. For all attributes A_i we define rH to be the **relevance of entropy reduction** $\boldsymbol{rH}$ of that attribute:

Attribute	sM_{ol}	sM_{lo}	rH	wsM_{ol}	wsM_{lo}
Length	1850.1	183.5	0.16	297.6	29.5
Weight	1909.3	130.8	0.25	468.0	32.1
Random	2024.0	10.0	0.00	4.7	0.0
			Summed:	770.3	61.6
			OR		12.5

Table 7.1. Orthogonality measures for Quételet data set with random attribute

Attribute	sM_{ol}	sM_{lo}	*rH*	wsM_{ol}	wsM_{lo}
(See Length & Weight attributes of Table 7.1)					
Quételet	2029.6	15.5	0.91	1848.4	14.1
			Summed:	2614.0	75.7
			OR		34.5

Table 7.2. Orthogonality measures for Quételet data set with Quételet attribute

$$rH(A_i) = \frac{\min_{A_i=a} H(D_{A_i=a})}{H_{tot}} \qquad (\mathbf{7.11})$$

Now the **relative weighted orthogonality** of an attribute A_i is defined as

$$rwsM_{ol}(A_i) = wsM_{ol}(A_i).rH(A_i) \qquad (\mathbf{7.12})$$

The relative linearity for A_i is defined analogously. By now summing the relative orthogonality and linearity over all attributes leads to the orthogonality and linearity of the data set. It is obvious that the derived measures are scaled in relation to the number of attributes, and the sampling size of the entropy behavior. To obtain a more absolute measure, we introduce, as the final step, the **orthogonality ratio OR**:

$$OR = \frac{\sum_{A_i} rwsM_{ol}(A_i)}{\sum_{A_i} rwsM_{lo}(A_i)} \qquad (\mathbf{7.13})$$

The orthogonality ratio *OR* is the parameter that we will use as the indication of orthogonality of a data set.

Running Example: Quételet's Body Mass Index

In Table 7.1 and 7.2, the values of the orthogonality measures (Equations 7.7, 7.9 and 7.10) are given for the two data sets. The entropy behavior of attribute *length* (see Figure 7.6a) has an orthogonality ratio of sM_{ol}=1850.1. Due to a relative entropy gain *rH* of 0.16 the relative orthogonality is 297.6; the linearity of the same entropy behavior is 183.5, with a relative linearity of 29.5. For the attribute *weight* (see Figure 7.6b), the same calculations lead to a relative orthogonality of 468.0 and a relative linearity of 32.1. For the random attribute (see Figure 7.6c), the relative entropy gain is 0.00 (0.002308, to be more precise), reducing the high absolute orthogonality of 2024.0 to a mere 4.7. For data set 1 this leads to a summed orthogonality of 770.3, a linearity of 61.6 and an *OR* of 12.5. For data set 2, with the (very relevant) attribute *Quételet* instead of the random variable, the *OR* is much higher. First, *Quételet* scores an orthogonality of 2029.6, which contributes 1848.4 to the relative orthogonality due to a relative entropy gain of 0.91. Consequently, the summed orthogonality

of data set 2 is 2614.0, and the ratio *OR* is 34.5. The following conclusions can be drawn from these results:

- The random attribute in data set 1 scores high on orthogonality sM_{ol}, but due to its low relevance of entropy reduction *rH*, it does not contribute to the summed orthogonality and *OR*.
- The orthogonal attribute *Quételet* contributes to the summed orthogonality *OR* because it is very orthogonal (sM_{ol})and very relevant (*rH*).
- The orthogonality ratio *OR* offers a measure for orthogonality that is independent of the number of attributes in the data set.

These results illustrate how the *OR* measure is influenced by the presence of orthogonal class boundaries in a data set. The logical next step, presented in the next section, is to develop a criterion on how OR relates to the type of class boundary, and to correlate this to technique performance.

7.5. Experimental Evaluation

With OR it is possible to express the orthogonality of a data set. In order to use OR as a guard for orthogonal techniques, we need a decision criterion on OR, i.e. in what OR-value range can we use orthogonal techniques and in what range non-orthogonal techniques? In this section, we experimentally evaluate the prototype matching method, and determine the threshold value(s) of OR for deciding between orthogonal and non-orthogonal techniques. This is done in three steps. First, we study the influence on the orthogonality ratio *OR* of the angle that a linear class boundary makes with attribute axes on a 2-dimensional data set (Section 7.5.1). The class boundary is either orthogonal, or a linear combination of the two attributes, with varying slope-intercept combinations. Moreover the noise level varies from 0 to 25%. Second, we test on more complex data how sensitive our approach is to class boundaries that occur in partial subspaces of a multivariate data set (Section 7.5.2). The data in this experiment is either a hierarchical combination of orthogonal class boundaries or a hierarchical combination of linear class boundaries. In this second experiment, we use class boundaries in 3 angles, and 3 noise levels per data set. For these two experiments, we use the performance of the Quest tool for learning univariate and multivariate decision trees (Loh & Shih, 1997) as validation. Moreover, we use the MetaL ranking tool and the performance of orthogonal and linear machine learning techniques in the MetaL toolbox (MetaL, 2004). Finally, in Section 7.5.3, we test the applicability of the approach beyond its assumptions on five 'real world' data sets to learn the influence of the data distribution on the sensitivity of the approach.

7.5.1. Two dimensional data

Goal: The first experiment studies two effects. First the correlation of OR with the orthogonality of the underlying class distribution is assessed. The hypothesis is that OR correlates with the slope class boundary: the more orthogonal the class boundary, the higher the value of OR. For the measure to be practically useful, the correlation with the noise level should be minimal. Second, the correlation with the performance of orthogonal and non-orthogonal techniques is assessed. Here it is the expectation that orthogonal techniques perform good for orthogonal class boundaries and poor for non-orthogonal class boundaries, while non-orthogonal techniques are expected to perform reasonably well overall. Based on the outcome a decision criterion on *OR* for appropriateness of (non-)orthogonal techniques is formulated.

Method: In the first experiment 110 data sets are generated. Every data set contains 2000 points from a 2-dimensional data space with attributes A_1 and A_2 from the continuous interval [0,1] (see Figure 7.11). Points are uniformly distributed over the data space. All cases belong to one of two classes $\{c_1, c_2\}$. The data space is split by a class boundary, dividing the space into two areas, I and II. The data sets vary in slope of the class boundary (the angle of the class boundary with the A_1–axis varies from 45° to 90° in 10 equal steps, giving a slope $\in$ {1, 1.19, 1.43, 1.73, 2.14, 2.75, 3.73, 5.67, 11.43, ∞}) and in noise level n (n runs from 0% to 25% in steps of 2.5%). Consequently, the probability that a record (a_1, a_2) is of class c_1 in areas I and II are $p(C=c_1 \backslash (a_1,a_2)$ *IN area I)* $= 1-n$ and $p(C=c_1 \backslash (a_1,a_2)$ *IN area II)* $= n$ respectively, and the probabilities for class c_2 are $p(C=c_2 \backslash (a_1,a_2)$ *IN area I)* $= n$ and $p(C=c_2 \backslash (a_1,a_2)$ *IN area II)* $= 1-n$. In other words, the noise level n defines the probability that a record is classified incorrectly. The noise level is uniformly applied on the entire data space. For every combination of slope and noise-level, 10 random data sets are drawn. The outcomes of the prototype analysis and Quest are averaged over these 10 sets.

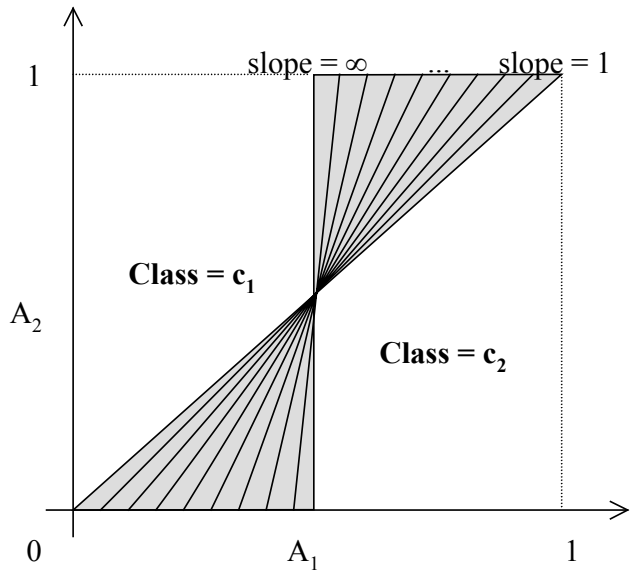
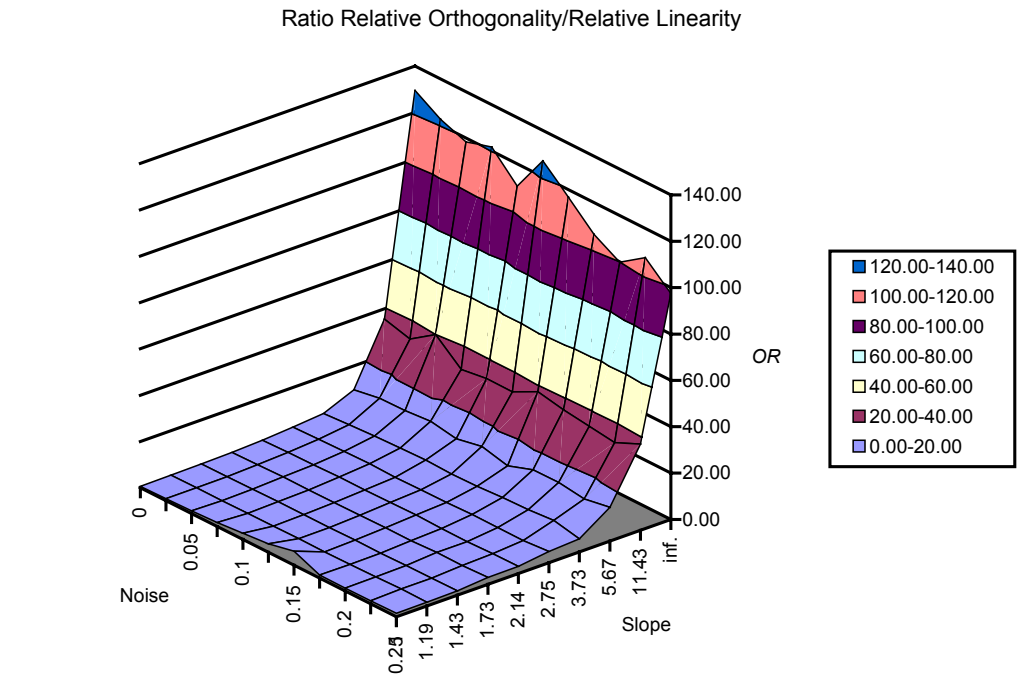


Figure 7.11. Class boundaries for artificial 2-dimensional data with slope $\in$ {1, 1.19, 1.43, 1.73, 2.14, 2.75, 3.73, 5.67, 11.43, ∞})

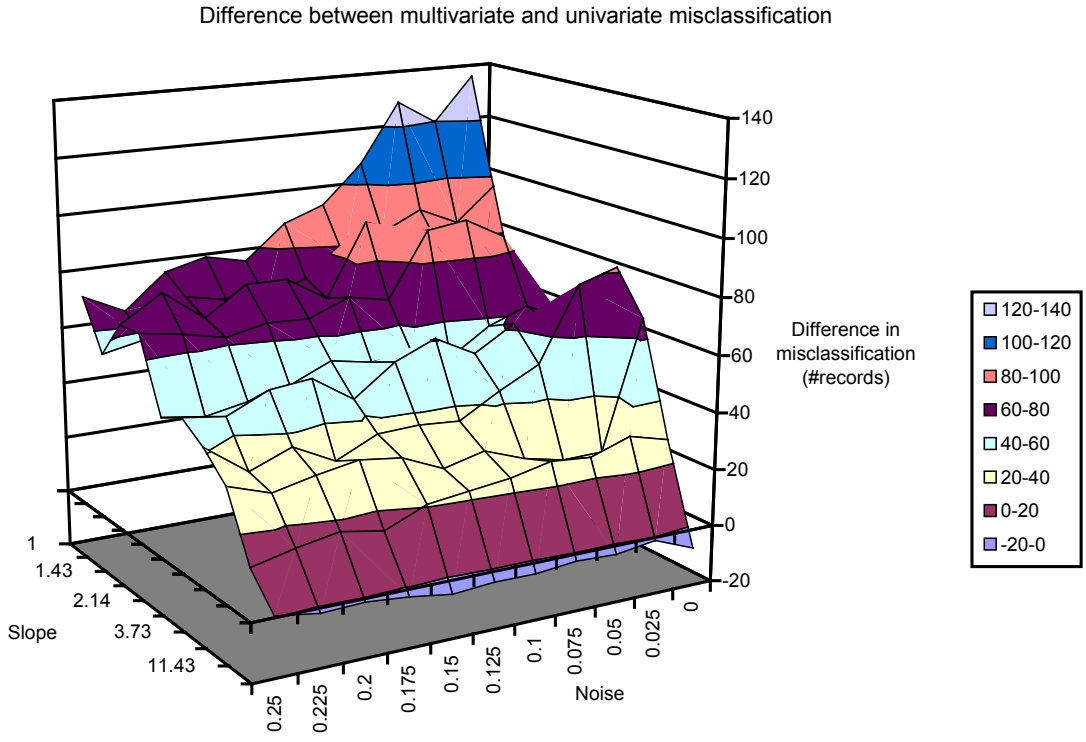
The entropy behavior is derived over both attributes of the data sets. The number of entropy values per attribute, f, is set to 128; in our experiments this value proved adequate and it ensures feasible computational complexity without posing too rigid constraints on the number of records in the data set. Every interval boundary $A=a_i$, i = {1, 2, ….128} is used to calculate $H(D_{A=ai})$.

For all data sets multivariate and univariate decision trees are generated using Quest (Loh & Shih, 1997). The performance of the decision trees is measured in the number of misclassified cases. The Quest parameters that are used can be found in the appendix 2. Each Quest-run is performed using 10-fold cross validation. For 10 individual data sets, with slope$\in$ {1, 1.19, 1.43, 1.73, 2.14, 2.75, 3.73, 5.67, 11.43, ∞} and 0% noise, we have obtained a MetaL ranking.

Results: Figure 7.12a shows the relation between the slope of the class boundary, the noise level in the data set, and the *OR*. The *OR* gradually increases from [0.5, 2] to [4-8] as the slope increases from 1 to 3.73. The *OR* values increase steeply to ranges of [95, 130] as the slope increases further. This trend occurs over all noise levels. Figure 7.12b gives the difference in misclassifications between univariate and multivariate trees. Overall multivariate trees score substantially less misclassifications than univariate. But when the slope increases (approx. > *3.73*), this difference vanishes.



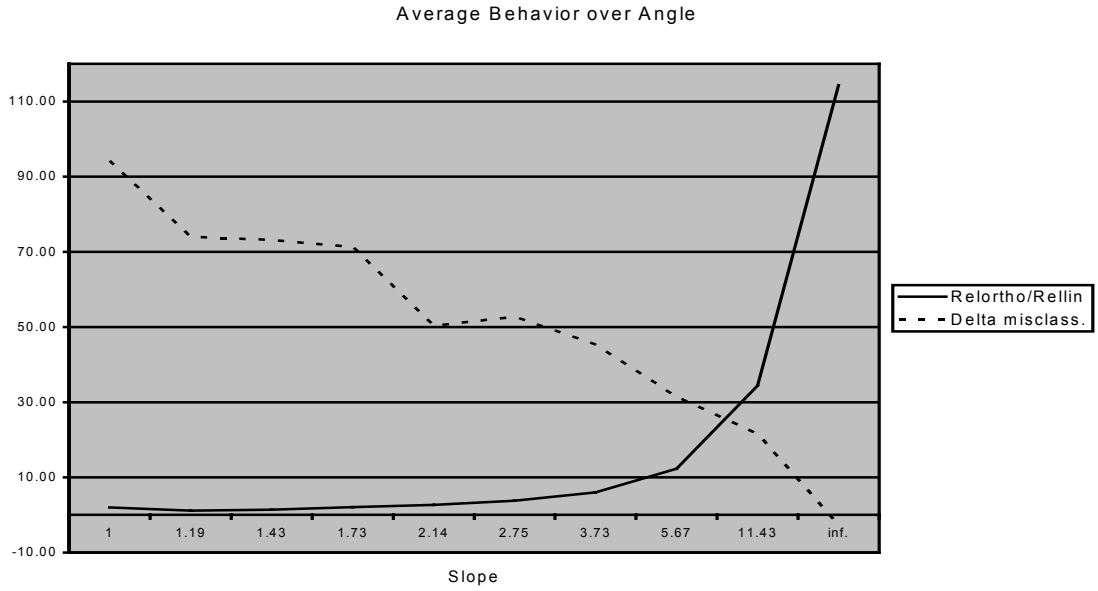
(a)



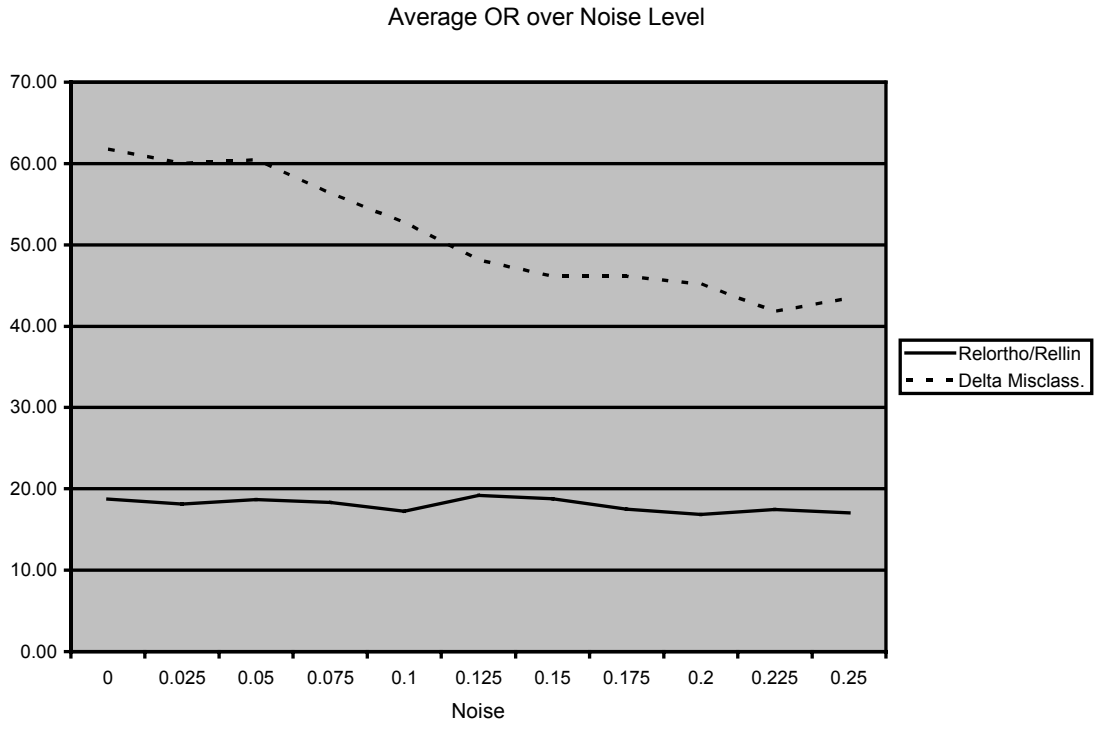
(b)

Figure 7.12. (a) The relation between noise and angle of the class boundary and the OR value and (b) the difference in misclassification of orthogonal and non-orthogonal techniques.

Note that for presentation purposes the axis orientation between the two figures has been changed.



(a)



(b)

Figure 7.13. The tendencies of Figure 7.12, but now averaged over Angle (a) and noise level (b).

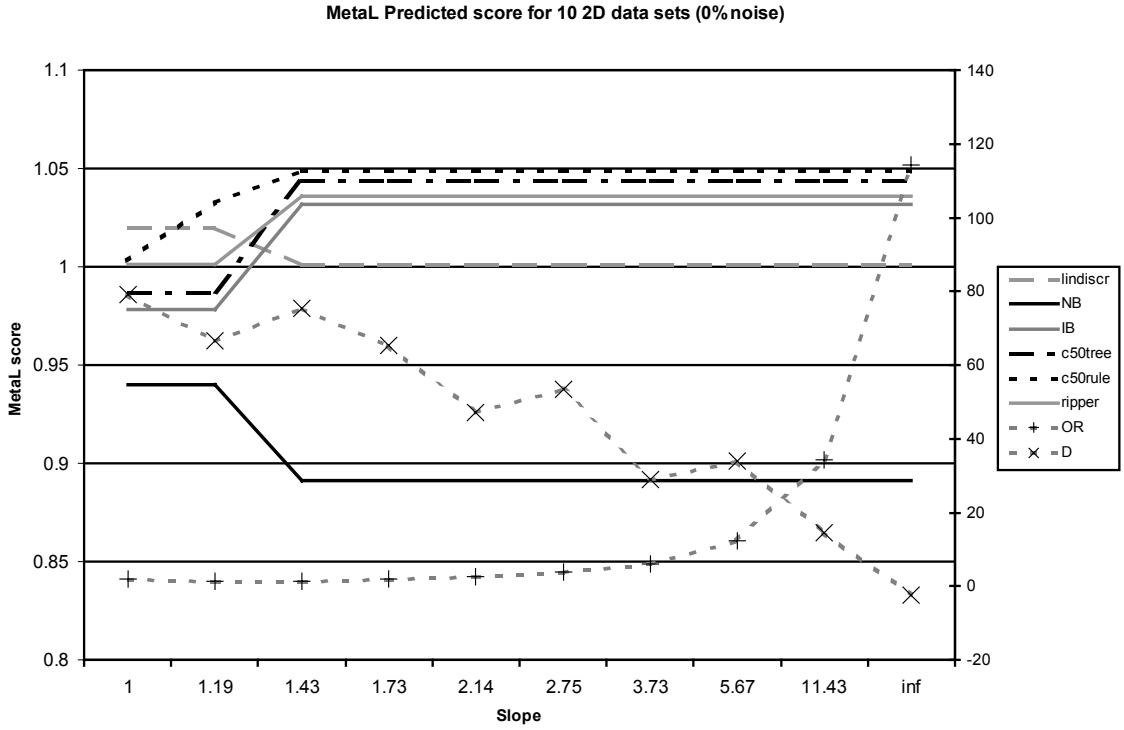


Figure 7.14. Comparison of MetaL ranking for three non-orthogonal machine learning techniques (lindiscr, NB and IB), three orthogonal machine learning techniques (C50tree, C50rules and Ripper) with *OR* and performance D. OR and D are expressed in the secundary scale on the right hand side of the chart. Results are based on one data sample per slope-value.

Figure 7.13 provides another view on both results. Figure 7.13a, gives the projection of Figure 7.12a and Figure 7.12b on the slope axis, averaging *OR* and relative misclassifications over all noise levels. An opposite trend for both curves can be observed in Figure 7.13a. Where the *OR* curve increases vastly, i.e. the slope rises to values > 3.73, the relative misclassifications gradually decreases. Figure 7.13b contains the similar projection on the noise axis, averaging over all slope values. The dependency between the orthogonality ratio and the difference in misclassifications between linear and orthogonal trees is confirmed, and the dependency between the noise level and the difference in misclassifications between linear and orthogonal trees proves less clear.

Figure 7.14 gives the MetaL rankings as function of *slope* for 3 orthogonal techniques (C50rules, C50tree; Rulequest, 2004; and Ripper; Cohen, 1996) and 3 non-orthogonal techniques (linear discriminant, naive Bayes and instance-based learning). For trend comparison the figure also contains the *OR* value over *slope* and the performance of univariate and multivariate classification with Quest. The latter two use their own scaling on the righthand side of the chart. The MetaL predicted score, based on descriptive statistics of the data set, shows little variation over *slope*. *OR*, as discussed before, correlates with *slope*. The actual performance difference of a univariate and a multivariate technique shows a correlation with *OR* of -0.66, whereas the (absolute value of the) correlation of *OR* with any of the MetaL predicted scores range from 0.19 to 0.22.

Conclusions: As assumed in the case of an orthogonal class boundary (*slope = inf*), univariate decision trees outperform the multivariate trees, while the multivariate decision trees perform better in the case that the class boundary would be a linear combination (*slope* $\rightarrow$ *1*). Figure 7.12 shows the *OR* plane as function of angle and noise level (a), and the plane of relative misclassifications: #multivariate misclassifications-#univariate misclassifications (b). The value of *OR* in Figure 7.12a raises as the angle increases. The increase of the slope of OR increases when the angle increases. Figure 7.12b shows the difference between multivariate and univariate classifier performance, expressed as difference in misclassification (multivariate – univariate). Here, we see a global downward tendency when the angle increases. The correlation coefficient of the two matrices of Figure 7.12 is -0,72 (t-test $P < 0.001$).

We also look at the correlation of *OR* with relative misclassifications, averaged over angle and noise level, respectively (Figure 7.13). There appears to be a high, and highly significant, correlation between angle and the average *OR* values for all noise levels (correlation of -0,79; t-test $P < 0.005$). The correlation between noise level and the average *OR* values for all angles is much lower, and less significant (correlation of 0,50; t-test $P < 0.4$). Figure 7.13a, confirms the tendencies of Figure 7.8. The correlation coefficient between OR and the performance is -0.79 (t-test $P<0.005$). In Figure 7.13b the *OR* shows to be hardly dependent on the noise level, where the differences in misclassifications decrease when the noise level increases. Both techniques seem equally hindered in noisy data sets. While in noise free environments, the multivariate classifier performs better than the univariate one. Here, the correlation coefficient is 0.50, (t-test $P< 0.4$).
From these results we extract the following decision criterion:

- When $OR > 15$, take the model to be orthogonal.
- When $OR \leq 4$, take the model to be non-orthogonal.
- When $4 < OR \leq 15$, apply both orthogonal and non-orthogonal models.

We put this heuristic to the test in the next experiment.

The MetaL scores are not grounded on the relation between data characteristics and technique performance, but are based on general descriptive characteristics of the data set. As a consequence, the MetaL score predictions over the various data sets in this experiment do not vary much; the statistical characteristics of the data sets are identical, but the underlying classification concepts differ. The prototype matching analysis of entropy behavior does exploit this relation, and thus reflects in the *OR* value the internal structure; in this case the orthogonality.

Running Example: Quételet's Body Mass Index

The *OR*-values of Tables 7.1 and 7.2 for the analyses of the Quételet data are interpreted as follows. Applying the criterion to set 1, with an *OR* value of 12.5, the conclusion is that both an orthogonal and a non-orthogonal technique could be appropriate. For set 2, with an *OR* value of 34.5, an orthogonal technique is indicated. When analyzing the two data sets with the MetaL ranking tool, the predicted ranking of the two data sets is identical. However, the actual results when applying linear and orthogonal techniques to the data indicate otherwise. Table 7.3 shows the learning results as obtained with the learning techniques that are included in MetaL. The table shows both the predicted score and the actual performance for the non-orthogonal techniques (Lindiscr, IB and NB) and the orthogonal techniques (C50rule, C50tree and Ripper). It is noted that the MetaL scores do not differ for both data sets, while the underlying concept has substantially changed. This shows in the *OR* scores in Tables 7.1 and 7.2. The OR score accurately predicts the best performing technique class when using the criterion as derived in the above section.

	Lindisc	IB	NB	C50rul	C50tree	Ripper
Set 1 Predicted	*1.02*	0.978	0.94	1.003	0.987	1.001
Performance	*1.02*	0.994	0.984	0.997	0.997	0.991
Set 2 Predicted	*1.02*	0.978	0.94	1.003	0.987	1.001
Performance	0.977	0.991	1.003	*1.008*	*1.008*	-

Table 7.3. MetaL ARR scores (Soares & Brazdil, 2000) for Quételet data sets. Predicted (dark rows) gives the prediction according to the MetaL tool, Performance (light rows)gives the actual results as obtained with the indicated technique applied through the MetaL workbench (MetaL 2004). In italics the best prediction and performance are indicated.

7.5.2. Multi-dimensional data

Goal: In the former experiment, we have confirmed the relation between *OR* and the slope of the underlying class boundary, and between *OR* and the performance of orthogonal and non-orthogonal techniques. This experiment aims to discover the relation between OR and the orthogonality of class boundaries in more complex concepts, especially if class boundaries are hierarchically organized over a larger number of attributes. Moreover, we wish to establish if the OR value is a better indicator for the orthogonality of class boundaries than MetaL ranking.

Method: In this experiment data sets with 5 attributes $[A_1..A_5] \in [0,1]$ are generated. Class labels are assigned according to a strict univariate concept:

```
Class C1:
  A1> 0.5 AND A2≤0.3, OR
  A1> 0.5 AND A3>0.7, OR
  A1≤ 0.5 AND A4≤0.3, OR
  A1≤ 0.5 AND A5>0.7
Class C2:
  Otherwise
```

or according to multivariate variants:

```
Class C1:
  A1> 0.5 AND A2 > α A2_+ β OR
  A1≤ 0.5 AND A4 ≤ α A5_+ β
Class C2:
   Otherwise
```

The α value is the slope of the class distribution, and determines the degree of (non)orthogonality of the class distribution. Values for α are varied to create multivariate class boundaries with angles of 90°, 75°, 67.5° and 45° (slope of ∞, 3.73, 2.41 and 1 respectively). The β stands for the intercept, and controls the proportion of the classes in the data set; β-values are set such that the chances for records class 0 and 1 are equal. Uniform class noise of 0, 10 and 25 % is applied, so that in total 12 parameter settings are used. Data-sets are generated with 2000 records, and the total experiment was repeated for 10 different samples. The reported results are the averages over these samples. For all parameter combinations in this experiment, we also obtained, for one data set per parameter setting, the MetaL rankings over linear and non-linear techniques.

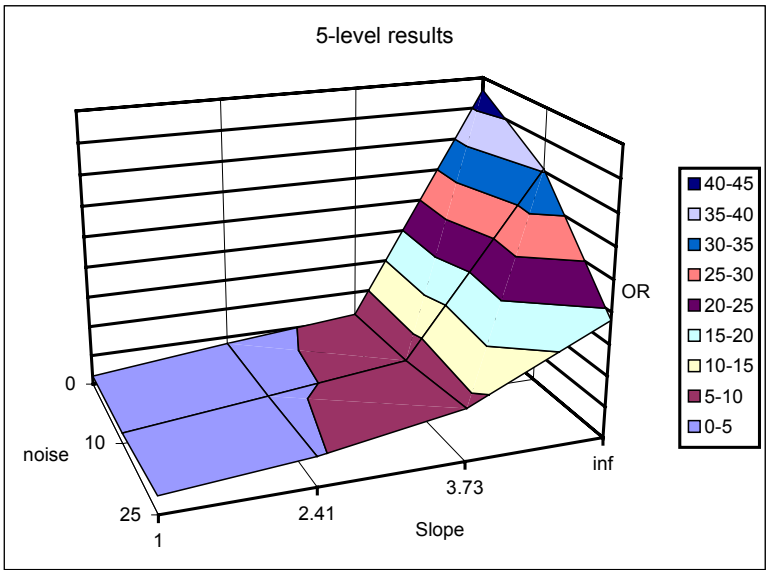


Figure 7.15. *OR* values for 5-variable data sets, aggregated over 10 runs. Linear class boundaries (*slope*<= 3.73) results in low *OR* values. When *slope*→∞, *OR* values increase disproportional. In noisy data sets, the effect is tempered, but still substantial.

Attr.	sd_{ol}	sd_{lo}	***rH***	wsd_{ol}	wsd_{lo}
A1	1978.0	53.3	0.001644	3.25	0.09
A2	1992.1	37.8	0.084298	167.93	3.18
A3	1903.4	128.8	0.05873	111.79	7.57
A4	1978.1	51.1	0.086066	170.25	4.40
A5	1970.6	59.5	0.082242	162.06	4.89
			Summed	615.29	20.13
			OR		30.57

Table 7.4. Analysis results of one of a 5-variable data set for orthogonal class boundaries with 0% noise.

Results: The class distribution over A_1 is constant and consequently, exhibits no significant entropy behavior. It does, however, show a high ratio between wsd_{ol} and wsd_{lo}. The other attributes show, for orthogonal class boundaries listed in Table 7.4, clear cusps in entropy behavior. Compared to the entropy behavior of Section 7.5.1, the attributes show a reduced relative entropy gain (see Figure 7.16) due to the hierarchical nature of the class boundaries (attribute A_2 is decisive for only half of the data set; for the other half it behaves as a random attribute). Figure 7.17 presents the A_2 attribute in case all attributes have a slope of 1, i.e. the class boundaries are multivariate. Table 7.4 gives the analysis of the prototype matching results for one orthogonal data set. It clearly indicates the orthogonal character of the class boundaries. Given the *OR* value, there is little doubt about the prevalent orthogonal character of the data set. The overall results (Table 7.5) show a clear difference in *OR* in the cases where orthogonal class boundaries occur in the data.

The MetaL rankings appear to be insensitive for the actual parameter combination. The score predictions for the non-orthogonal techniques (linear discriminant, naive Bayes and instance based learning) and the orthogonal techniques (C50tree, C50rules and Ripper) were 1.071, 0.997, 0.892, 0.943, 0.963, 0.979, respectively. In other words, the MetaL ranking proved insensitive to variations in concept types. Again, as can be concluded from Tables 7.4 and 7.5

and Figure 7.15, the *OR*-value distinguishes the parameter settings, and correlates with the orthogonality of the underlying class boundaries.

Conclusions: The main conclusion of this experiment is that the *OR*-value is a good indicator for the underlying class boundary. The MetaL toolbox, in contrast, ignores differences in class boundaries. Figure 7.15 justifies the conclusion that OR enables differentiating orthogonal and non-orthogonal class boundaries, and that the criterion formulated in the former experiment is confirmed.

Noise level	**Slope**			
	∞	**3.73**	**2.41**	**1**
0	42.96	5.97	3.86	1.42
10	35.38	6.62	4.09	1.69
25	18.76	8.41	4.76	2.86

Table 7.5. *OR r*esults on 5 variable data set, for various angles and noise levels as described in the text. The given *OR* values are average values out of 10 repeats.

7.5.3. Applicability beyond Assumptions

Goal: The experiments in Sections 7.5.1 and 7.5.2 concern data sets where the data samples are uniformly distributed over the data space. This is in line with the assumptions underlying the formulation of the prototypes (see Appendix 1) as being for uniformly distributed data. In this last experiment, the prototype matching method is tested on (slightly modified) UCI data sets and a data set from our own practice (see appendix 3 for a description of the data sets). These data sets are characterized by non-homogeneous data distributions. As such, these sets do not contain data with the exact characteristics for which the prototypes were actually defined. The goal of this experiment is to test the sensitivity of the prototype matching method against this uniformity assumption, and to relate its performance to the accuracy of MetaL on these data sets.

Method: For each data set the orthogonality ratio *OR* is calculated. The *OR* values are compared with the MetaL rankings and the performance of orthogonal and linear machine learning techniques in the MetaL toolbox (MetaL 2004).

Results: Table 7.6 shows the OR results. According to the heuristic developed earlier, *OR* is expected to give an indication of the applicability of orthogonal machine learning techniques for learning the classification in the data set. The data sets are ordered by increasing *OR* value. If *OR* is a robust predictor for orthogonality, one expects that the PHA data are non-orthogonal, Wdbc and Spam to be orthogonal, and GI2 and Abalone2 to be either of the two.

As in the earlier experiments, the MetaL tool is used to obtain a ranking of the 6 available learning algorithms (C50rules, C50tree, Ripper, linear discriminant analysis, instance-based learning and naive Bayes). Moreover, we let MetaL apply these 6 learning approaches to the data and assess the performance in terms of the adjusted ratio of ratios (ARR, Soares & Brazdil, 2000) of these techniques. Table 7.7 shows the MetaL results, with indications of the best-predicted technique (italic) and the actual best performing technique (bold). The left half of the table contains the MetaL prediction and performance for non-orthogonal techniques; the right half contains the MetaL prediction and performance for orthogonal techniques. If the MetaL ranking is a robust predictor for orthogonality, one expects the predicted-best and actual best technique to be positioned in the same color segment.

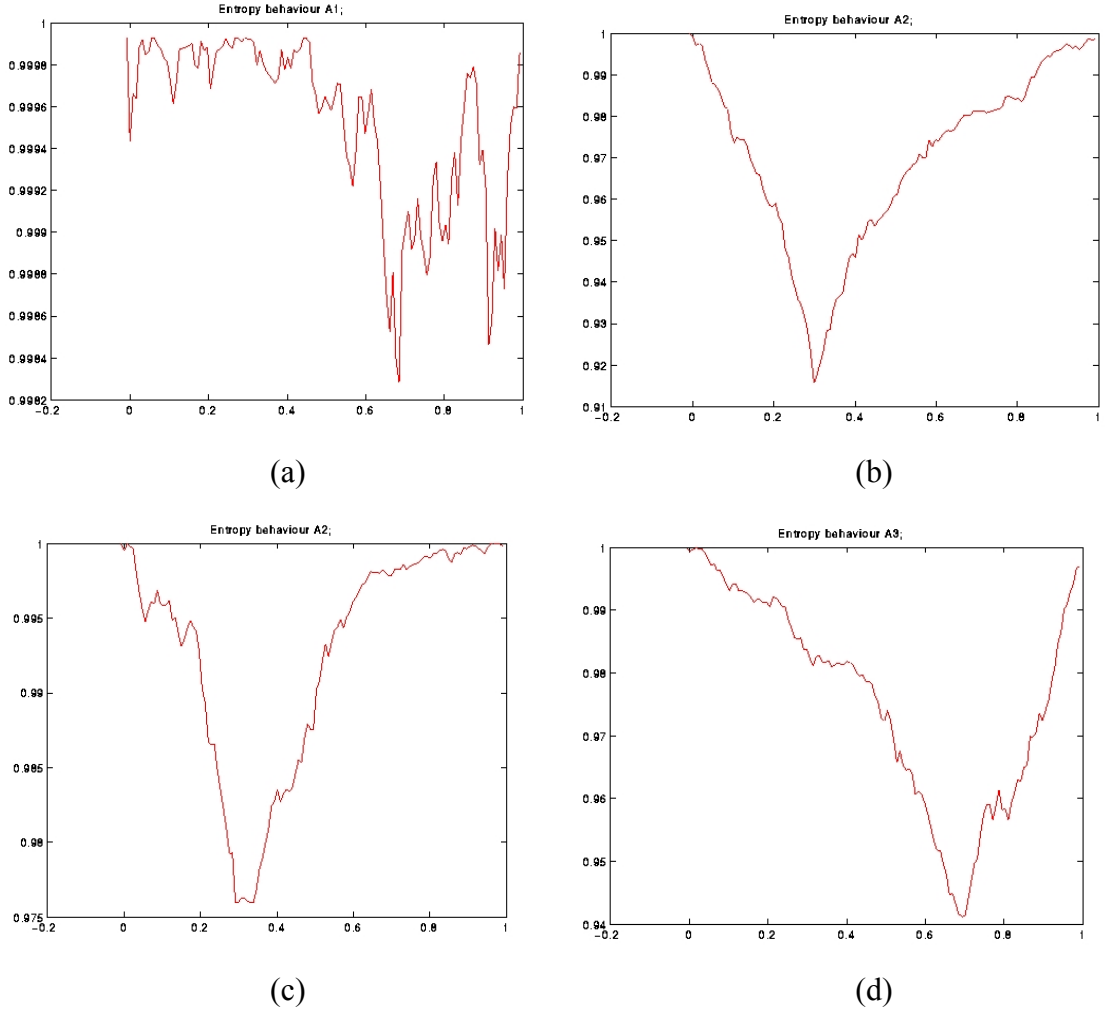


Figure 7.16. Entropy behavior of attributes in the experiment of Section 7.4.2 for the orthogonal class distribution. Attribute A_1 (a) shows random behavior. Attributes A_2 (b) and A_3 (d) show cuspy behavior, which is potentially blurred when classification noise is imposed (c). Note that the (vertical) entropy axis differs in scaling for the different charts.

Data-set	wsd_{ol}	wsd_{lo}	**OR**
PHA	5082.1	1504.7	3.4
GI2	3490.7	846.6	4.1
Abalone2	2065.2	243.0	8.5
WDBC	15256.4	1012.8	15.1
Spam	7273.3	411.2	17.7

Table 7.6. Prototype Matching Results on UCI and real world data sets.

Conclusions: Both MetaL and prototype matching score poor on these data sets. MetaL correctly predicts 2 out of 5 non-orthogonality predictions (PHA, Spam). Prototype matching correctly predicts 1 out of 5. However, the performance of MetaL is not significantly better than that of the OR score(s) given the 5 data sets and the small differences in predictive accuracy between the various techniques.

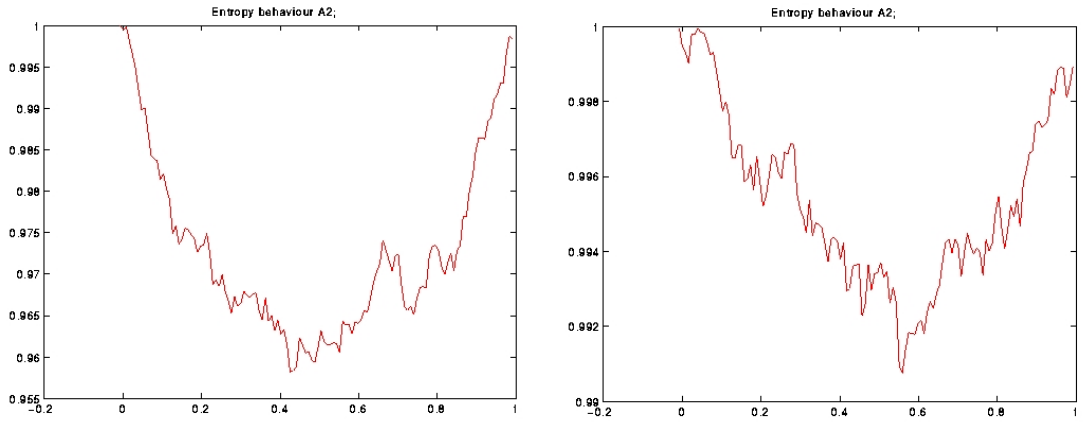


Figure 7.17. Entropy behavior for attribute A2 with slope=1, for 0 % noise (left) and 25% noise (right). Note that the (vertical) entropy axis differs in scaling for the different charts.

		Technique					
Data-set		**Lindiscr**	**NB**	**IB**	**C50rules**	**C50tree**	**Ripper**
PHA	Rank	1.00	0.89	1.03	*1.05*	1.04	1.04
	Performance	0.96	0.77	1.03	1.07	**1.07**	1.06
GI2	Rank	1.00	0.91	1.02	1.03	*1.03*	1.03
	Performance	0.99	0.97	**1.02**	1.02	1.02	0.98
Abalone2	Rank	*1.07*	1.00	0.89	0.96	0.94	0.98
	Performance	1.02	0.95	0.94	1.02	1.02	**1.02**
WDBC	Rank	1.00	0.89	1.03	*1.05*	1.04	1.04
	Performance	**1.01**	0.98	1.00	1.00	1.00	0.98
Spam	Rank	0.91	0.78	1.01	*1.07*	1.05	1.07
	Performance	0.98	0.89	0.98	**1.04**	1.03	1.02

Table 7.7. MetaL ARR scores for real-world data sets. The numbers indicate the predicted score of MetaL (dark rows) and the actual score based on the MetaL toolbox (light rows). The italic numbers indicate the highest scores. The three techniques in the left part of the table are non-orthogonal techniques, the three techniques to the right are orthogonal.

As the data sets are violating a major assumption underlying the validity of the prototype matching approach these scores for OR are not surprising. When analyzing these data sets, they appear to have a highly non-uniform data distribution. In terms of density of data points along the attribute axis, normal distributions dominate flanked by a number of extremely skewed distribution functions (e.g. in the Spam database, where attributes reflect highly skewed appearance frequencies of specific words in e-mails). Because the prototypes used in the current version are based on uniform data distributions, it is hard to draw conclusions.

This leads to the definition of a future extension of the method. Before applying the prototype matching approach, the type of data distribution of each attribute has to be assessed. The prototypes E (Equation 4) need to be extended with a qualifier $_d$ for the type of data distribution: $E_{t,d}$, with t stands for l (linear) or o (orthogonal) as specified in footnote 2. Consequently, the attributes are matched using the appropriate prototype, e.g. normal distribution, uniform distribution or other continuous distributions.

7.6. Conclusion and Discussion

Existing technique selection approaches such as meta-learning are based on fairly 'arbitrary' selections of data sets and techniques. As such, they give no other explanation than "it worked well on a number of similar data sets," thus ignoring in their evaluation, the principles of the techniques, and the way that these connect to patterns in a data set. Moreover, it is debatable whether the simple, statistical and information theoretic data set measures that are most commonly used in deriving the technique selection bear a fundamental relation with technique performance.

Technique dependent guards offer a more fundamental way of technique selection. The guards test per technique or technique class for a specific data set if the assumptions underlying the technique hold. Or, phrased differently, whether it can be expected that the data set complies with the specific bias of the technique.

For orthogonal techniques prototype matching is a guard that assesses the amount of orthogonal class boundaries in data sets, without actually running the technique. The applied prototypes outperform MetaL on data sets that comply with the assumption of uniform data distribution that underlies the used prototypes. Beyond these assumptions, both approaches perform equally poor.

Selecting the appropriate machine learning technique for a specific data set is a critical success factor in applying machine learning techniques. This chapter discusses guarded technique selection (GTS) as a principled approach to technique selection. In GTS, every technique (group) is 'protected' by a proprietary guard that fires when a data set meets the specific preconditions for that technique. The proof of concept is given for orthogonal machine learning techniques, i.e. techniques that build classifiers by recognizing (partial) orthogonal class boundaries. Orthogonal class boundaries declare themselves as cusps in the entropy behavior of attributes. Non-orthogonal class boundaries show a more gradual, entropy behavior. Analysis of entropy behavior resulted in entropy prototypes for orthogonal and non-orthogonal class boundaries. Prototype matching, making a quantifiable comparison between the respective similarities of these prototypes with the entropy behavior of a new data set, enables the choice between orthogonal or non-orthogonal machine learning techniques.

Within the assumption of uniform data distribution that underlie the tested prototypes, this approach proves more accurate then the existing approach of meta-learning, implemented in the MetaL tools. Meta-learning explores the empirical relation between data characteristics and technique performance. The set of data characteristics used by MetaL however is broad, and not specifically focused on those aspects of machine learning techniques that make the difference. As illustrated for the two-dimensional, the five-dimensional and the qQuételet data sets, MetaL misses crucial aspects of the underlying class boundaries in its analysis.

The importance of operating within the assumptions has been illustrated. When the characteristics of the data set do not meet the assumptions that underlie the prototypes, the performance of GTS drops to the level of the MetaL tools. Explicit testing of data characteristics per attribute, and calibration of prototypes for each attribute according to these characteristics, is an extension of the GTS approach that follows from this result. Relevant features that may influence prototype formulation are data distribution (assumed to be uniform) and noise distribution (ditto).

The prototypes used in this work are modeled on the basis of one single class boundary on the attribute axis. Multiple class boundaries are reflected as numerous cuspy minima in the entropy behavior, at a finer scale than would one global class boundary. Only matching the

prototype with the entire entropy behavior would overlook these local cusps. The calculation of a local match, by combining scaling and translating of the prototype signal, makes the analysis sensitive for these 'hidden cusps'. This approach is inspired by the wavelet analysis approaches, and guarantees that multiple class boundaries will be reflected in the OR-value.

In this chapter, we use the prototype matching approach as a cornerstone for guarded technique selection. The prototype matching principle may also be deployed in local selection of a learning method, data discretization and meta-learning. Technique selection as studied here presupposes the monolithic application of one machine learning technique to the entire problem space as covered by the data set. Alternatively, one can choose to locally optimize the model by hierarchically combining machine learning techniques (e.g. perceptron trees, Utgoff, 1988). GTS can be used to select techniques in sub-spaces of data sets, e.g. in hierarchical model structures. Van der Ham (2002) explores this line during earlier stages of this work (cf. Brodley, 1995b). Although this may lead to more accurate models, especially in problem domains with composite class distributions with different underlying types of class boundaries, it does not change the problem of technique selection. It merely transforms the problem of model selection to a smaller subspace of the total data set.

Discretization support makes use of the hierarchical and local analysis of date. Fayyad and Irani (1993) show that, if local minima in entropy behavior exist, these local minima can be used for discretization, and that iterative detection does not decrease the quality of the identified minima. With the prototype matching approach, local minima are identified in parallel, and local discretization points can be easily detected.

Finally, the OR measure may be included in the MetaL toolbox as a meaningful descriptor of data sets. To complete the prototype matching approach, future work has to aim for complementary prototypes for other data distributions and noise effects. To complete the GTS approach, guards for additional groups of techniques need to be developed.

In conclusion, this chapter introduces guarded technique selection based on relevant criteria, with a verifiable rationale. The orthogonality ratio, OR, is a fundamental property of a data set that matches one-to-one the assumptions that a class of machine learning technique makes about class distributions in data sets.

Appendix 1. Derivation of entropy prototypes

For a strict class boundary (Figure 7.18) with 2 classes, a uniform data distribution and a probability for one class of p if $a_i \leq a_c$ and $(1\text{-}p)$ if $a_i > a_c$, the entropy behavior as function of partition point a_i , $H_o(a_i)$ is expressed by:

$$H_o(a_i) = \frac{a_i - a_{\min}}{a_{\max} - a_{\min}} e_{low}(a_i) + \frac{a_{\max} - a_i}{a_{\max} - a_{\min}} e_{high}(a_i) \quad \textbf{(7.14)}$$

with e_{low} and e_{high} defined as:

$$e_{low}(a_i) = -\left(y_1(a_i)^2 \log(y_1(a_i)) + y_2(a_i)^2 \log(y_2(a_i))\right) \quad \textbf{(7.15)}$$

$$e_{high}(a_i) = -\left(y_3(a_i)^2 \log(y_3(a_i)) + y_4(a_i)^2 \log(y_4(a_i))\right) \quad \textbf{(7.16)}$$

where y_1, y_2, y_3 and y_4 are defined as:

$$y_1(a_i) = \frac{(\min(a_i, a_c) - a_{\min})p + (\max(a_i, a_c) - a_c)(1-p)}{a_i - a_{\min}} \quad \textbf{(7.17)}$$

$$y_2(a_i) = \frac{(\min(a_i, a_c) - a_{\min})(1-p) + (\max(a_i, a_c) - a_c)p}{a_i - a_{\min}} \quad \textbf{(7.18)}$$

$$y_3(a_i) = \frac{(a_c - \min(a_i, a_c))p + (a_{\max} - \max(a_i, a_c))(1-p)}{a_{\max} - a_i} \quad \textbf{(7.19)}$$

$$y_4(a_i) = \frac{(a_c - \min(a_i, a_c))(1-p) + (a_{\max} - \max(a_i, a_c) - a_c)(1-p)}{a_{\max} - a_i} \quad \textbf{(7.20)}$$

For a non-strict class boundary (Figure 7.18) with 2 classes, a uniform data distribution and a probability for one class of p if $a_i \leq a_1$ and $(1\text{-}p)$ if $a_i > a_2$, and a gradual decrease for $a_1 < a_i \leq a_2$,the entropy behavior as function of partition point a_i $H_l(a_i)$is expressed analogous to (9)-(11), but now with y_1, y_2, y_3 and y_4 defined as:

$$\begin{aligned} y_1(a_i) &= \frac{\min(a_i, a_{i1})}{(a_i - a_{\min})} p \\ &+ \frac{\min(a_{i2}, \max(a_i, a_{i1})) - a_{i1}}{(a_i - a_{\min})} * \left(\left(1 - \frac{\min(a_{i2}, a_i) - a_{i1}}{(a_{i2} - a_{i1})} \right) p + \frac{\min(a_{i2}, a_i) - a_{i1}}{2 * (a_{i2} - a_{i1})} \right) \\ &+ \frac{\max(a_i, a_{i2}) - a_{i2}}{(a_i - a_{\min})} (1-p) \end{aligned} \quad \textbf{(7.21)}$$

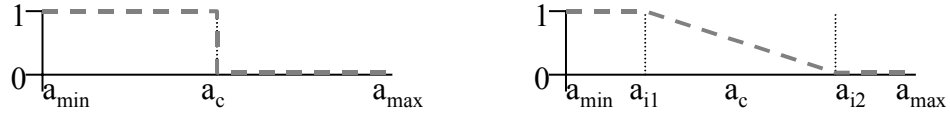


Figure 7.18. The density of the class distribution for (a) a strict class boundary that comes with an orthogonal concept and (b) a non-strict class boundary that belongs to a linear class boundary.

$$y_2(a_i) = \frac{\min(a_i, a_{i1})}{(a_i - a_{\min})}(1-p)$$
$$+ \frac{\min(a_{i2}, \max(a_i, a_{i1})) - a_{i1}}{(a_i - a_{\min})} * \left(\left(1 - \frac{\min(a_{i2}, a_i) - a_{i1}}{(a_{i2} - a_{i1})} \right)(1-p) + \frac{\min(a_{i2}, a_i) - a_{i1}}{2*(a_{i2} - a_{i1})} \right) \quad \textbf{(7.22)}$$
$$+ \frac{\max(a_i, a_{i2}) - a_{i2}}{(a_i - a_{\min})} p$$

$$y_3(a_i) = \frac{a_{i1} - \min(a_i, a_{i1})}{(a_{\max} - a_i)} p$$
$$+ \frac{a_{i2} - \min(a_{i2}, \max(a_i, a_{i1}))}{(a_{\max} - a_i)} * \left(\left(1 - \frac{a_{i2} - \max(a_{i1}, a_i)}{(a_{i2} - a_{i1})} \right)(1-p) + \frac{ai2 - \max(a_{i1}, a_i)}{2*(a_{i2} - a_{i1})} \right) \quad \textbf{(7.23)}$$
$$+ \frac{a_{\max} - \max(a_i, a_{i2})}{(a_{\max} - a_i)}(1-p)$$

$$y_4(a_i) = \frac{a_{i1} - \min(a_i, a_{i1})}{(a_{\max} - a_i)}(1-p)$$
$$+ \frac{a_{i2} - \min(a_{i2}, \max(a_i, a_{i1}))}{(a_{\max} - a_i)} * \left(\left(1 - \frac{a_{i2} - \max(a_{i1}, a_i)}{(a_{i2} - a_{i1})} \right)p + \frac{ai2 - \max(a_{i1}, a_i)}{2*(a_{i2} - a_{i1})} \right) \quad \textbf{(7.24)}$$
$$+ \frac{a_{\max} - \max(a_i, a_{i2})}{(a_{\max} - a_i)} p$$

Note that the equations 7.15-7.18 are special cases of 7.19-7.22, with $a_{i1} = a_{i2}$.

Appendix 2. Quest parameters in experiments of Section 7.4

The Quest tool (Loh & Shih, 1997) was used in a batch version for automatic comparison. Below the standard parameter values are represented for the experiments in Section 7.4. The Q-numbers refer to the Quest manual (Shih, 2002). If text and a number (in brackets) is given, the text is the choice text, and the number is the value in the batch set.

Q1-Q3 Depend on data set
Q4 Input Priors: Estimated from data (1)
Q5 Misclassification costs: Equal (1)
Q6 Minimal Node size: 100
Q7 Input splitting method: univariate (1) linear (2) *
Q8 Input variable selection method: unbiased statistical tests (1)
Q9 Alpha value: 0.05
Q10 Input method of split point selection discriminant analysis (1)
Q11 Input number of SEs for pruning: 1.00
Q12 Prune by: CV (1)
Q13 Number of CV-fold: 10
Q14 No test sample: (1)
Q15 Details for CV trees wanted: yes (2)
Q16 NO Pstricks LaTeX code TreeTeX LaTeX code or allCLEAR code wanted (1, 1, 1)
Q17 No indication for class label and terminal node id wanted (1)

Appendix 3. Description of data sets

In our experiments we use 4 slightly adapted versions of commonly known data sets from the UCI data repository: Abalone, Glass Identification, Spam and the Wisconsin Breast Cancer data set. Additionally, a data set from A&F is used, that relates to PHA production, a microbiological process.

Abalone2

Abalone2 is a 2-class continuous version of the UCI Abalone data set. In Abalone2, all non-continuous attributes were removed, and the task was changed from predicting the exact number o rings (29 classes in the original data set) to predicting whether an individual has less than or equal to 9 rings, or more.

- #records: 4177
- #attributes: 7 continuous attributes (Length, Diameter, Height, WholWght, ShckWght, ViscWght, ShllWght)
- #class 1: 2096 (number of rings <= 9)
- #class 2: 2081 (number of rings > 9)

GI2

Our GI2 data set is identical to the UCI Glass Identification data set, with one exception: the class label indicates whether the glass sample comes from window glass or not. In the original data set, the class to predict has

- #records: 214
- #attributes: 9 continuous attributes (RI, Na, Mg, Al, Si, K, Ca, Ba, Fe)
- #class 1: 163 (window glass)
- #class 2: 51 (window glass)

Spam

The standard UCI version.

- #records: 4601
- #attributes: 57
- #class 1: 1813 (spam)
- #class 2: 2788 (no spam)

WDBC

The standard UCI version of the Wisconsin Database on Breast Cancer is used, but records with missing values are removed.

- #records: 569
- #attributes: 30 (10 attributes radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, fractal dimension, as mean, standard error and "worst")
- #class 1: 212 (Malignant)
- #class 2: 357 (Benign)

PHA

The PHA data set contains process control data for a microbiological reactor. The process yield is the dependent attribute. The process is considered to result in a sigmoid yield curve, representing a production phase, a growth phase and a lag phase. The lag phase in th is specific process is commercially not interesting. Process operators should stop the process as soon as the lag phase is reached, if they would be able to detect the phase transition from

growth phase to lag phase. Process yield however is not an on-line measurement. The process is sampled, and the resulting samples are analyzed off-line in a laboratory. The class indicates whether the process is in its growth phase, predicted from on-line process variables. If a reliable prediction can be made of yield, based on the process parameters, process efficiency may increase. In that case, a software sensor can be used to indicate to operators that the lag phase has begon.

- #records: 8144
- #attributes: 18
- #class 1: 3801 (growth phase)
- #class 2: 4343 (other phase)

8. Conclusion

Methodological aspects of applying machine learning for practical problem solving area studied in this thesis. Machine learning is one area of AI research that has made, in the form of data mining applications, a more or less independent entrance into the market of industrial application. Apart from its application in data mining, machine learning techniques can also be deployed as instruments for acquiring models for knowledge based systems. In this role they offer the prospects for making systems fully adaptive, that is systems that are able to accommodate their knowledge models to changing environments.

A system development method can comprise an *activity model* and a *coherent set of development tools*. Existing methodological research in machine learning has focused on operating the technology. Given the ability to master the explorative operation of techniques the need arises to bind requirements from the business application environment to the design and operation parameters of the technology. The work in this thesis addresses these two critical factors by subdividing the related issues into four questions of interest as extensively detailed in Chapter 1, and offering for each suggestion, answers and/or tools as summarized below:

1. The need for support in the practical application of machine learning techniques have been identified on the basis of both a survey among practitioners and the analysis of two application projects.

2. A model has been formulated that both considers the integration of the machine learning module in the embedding system as well as the design issues for machine learning modules.

3. A decomposition approach to design a knowledge system that efficiently deploys available data and knowledge resources.

4. An approach has been developed to select machine learning techniques for a given problem based on the types of models that techniques construct from data.

The results of the survey in Chapter 2 reveal a gap between the methodological concepts for practical application that were provided by machine learning scientists and the methodological needs that were experienced in practice. This thesis has focused on the needs that are identified in this survey: efficient application design and technique selection support.

8.1. Support Need for Applying ML Techniques

The goal of the survey reported in Chapter 2 is to assess the status of current ML application and to learn from practitioners the research items that need further attention in order to facilitate ML application in practice. The main findings are that practitioners apply a highly explorative working style. When an opportunity for applying machine learning techniques is suspected, available techniques are tested in search of a satisfying solution. Although tools are available, respondents report the regular development of their own machine learning software. This is partly due to the technological focus of much of the available research.

An important finding of the survey is that methodological and technical shortcomings block a wide breakthrough of machine learning techniques in practical applications. Another conclusion of the survey is that the main methodological shortcoming is due to the fact that there is no design approach that values the deployment of machine learning techniques against alternative choices, such as knowledge systems. Consequently, there is poor justification for design decisions on the deployment of machine learning techniques. The main technological shortcomings are in the areas of technique selection and parameter tuning.

First, the dissemination of knowledge on available techniques is poor. Many of the developed techniques that are reported in the research community simply never reach practitioners. Although modern tools offer a multitude of available techniques, hardly any support is offered to select amongst these techniques, apart from brute force comparative exploration.

Furthermore, respondents report the lack of an established or standard application practice. The application of machine learning techniques is not a standard option when developing software or knowledge systems. And this lack of machine learning techniques as a standard option, as is revealed in the survey, is what is most changed over the last few years. The rise of DM/KDD, amongst others in the form of Business Intelligence solutions, is substantial. In spite of this development, according to fairly recent KDNuggets survey results reported in the Chapter 2 addendum, the offered options have not resulted in widely accepted solutions for the methodological and technique selection problems that are originally reported in the 1994 survey.

Chapter 3 and 4 discuss two illustrations from practical application that support the existence of *methodological support* and *technique selection* as two major research issues. In Chapter 3, a planning system is developed that combines machine learning techniques with expert knowledge components. Although the results of deploying the system are encouraging, methodologically the project experienced the problems as sketched above. Combining machine learning and expert knowledge components is an experimental process, with little guidance and a lot common sense. The same is relevant for the selection and configuration of ML techniques, which, amongst others, is apparent in the way the NN module is designed. The PTSS system that is described in Chapter 3 served as a test case for the design approach in Chapter 6.

Chapter 4 presents two separate systems in the context of waste water treatment. The issue that these systems illustrate is that for the design, both at a functional as well as on a technical level, it is crucial to consider the availability of background knowledge. In the case of the WQSM, explicability of the results is a crucial (non-functional) requirement. Moreover, a substantial body of background knowledge is available in the form of a database of components. In the SCS on the other hand, time is the critical success factor, and knowledge on proper decisions is scarce.

The integration of expert knowledge and induction can take two forms. The most common form, where knowledge system components collaborate with inductive components, is illustrated in Chapter 3. This approach is elaborated in Chapter 6. In Chapter 4, an alternative way of integration is suggested where explicit knowledge and learning from experience are combined in one module. That path has not yet been explored.

8.2. Designing Learning Applications

One issue that is raised in Chapter 2 is the need for efficient design support, that balances the use of machine learning components versus the use of a traditional knowledge component. Chapter 5 introduces the MEDIA model as a general structure that supports the effective design of machine learning applications. The leading principle of this approach is the separation between system functionality (the ‘mappings of input on output data’ that, when connected and controlled in the right way, produce the desired system behavior), the deployment of specific learning techniques for the functional modules, and the configuration of technique specific parameters. Consequently, the design process is cut into three levels: designing the functionality, selecting techniques, and configuring the techniques. At the application level the functional decomposition is designed based on an early identification of potential knowledge sources in the form of data, expertise, models, documentation or any

other form. Next, for the machine learning components, appropriate techniques are identified based on the characteristics of specific mapping, the match between the underlying assumptions of techniques, and the actual patterns in the data set. Finally, the optimal parameter settings have to be identified for individual techniques. This last step at the technique level applies knowledge that is very technique specific, and falls outside the scope of this work.

Chapter 6 provides an approach for designing the system functionality. In this approach designing functionality is presented as a planning problem. Based on an analysis of the domain, potential components are listed, with for each component the potential acquisition type (elicitation vs. learning), estimates for acquisition costs and estimates for the quality of the mapping the component can provide. The planning problem consists of finding a particular structure of components that is able to provide a sufficient output quality against acceptable costs. It offers a mixed top-down and bottom-up approach to planning of knowledge acquisition. The overall problem is decomposed into feasible components based on the availability of elementary knowledge sources in the problem domain. The driving force behind the decomposition balances acquisition costs and expected benefits.

8.3. Selecting learning techniques

The second most prominent need in ML application support identified in Chapter 2 is technique selection. In Chapter 7, Guarded Technique Selection (GTS) is presented as a new approach to technique selection. A guard is a software agent that checks whether a data set complies with the assumptions that underlie the technique. The compliance is expressed as a numeric value that indicates whether a technique is expected to come up with a model of reasonable predictive accuracy. The outcome of running guards of various techniques or technique groups to a data set is to obtain a selection of techniques for which the data set complies with the underlying assumptions.

As an example, a guard is developed for orthogonal machine learning techniques in Chapter 7. These techniques (recursively) partition a data set, where the partition-border is univariate. Data-sets with class boundaries that do not comply with orthogonal character, will result in either non-optimal predictive accuracy, or in a huge concept definition that might be fairly simple in terms of other techniques.

The entropy behavior of an attribute, i.e. the potential entropy gain as a function of the attribute value to partition on, appears to contain information on the nature of class boundaries. Orthogonal boundaries, even when composed of a combination of univariate boundaries, reflect in entropy behavior as sharp-edged, cuspy, minima. Other types of class boundaries reflect a more gradual, non-cuspy minima. Prototype matching, i.e. matching entropy behavior with hypothetical entropy behavior for orthogonal and non-orthogonal class boundaries, gives a good indication for the orthogonality of the patterns in a data set. The orthogonality, expressed in the OR ratio, can be used to assess whether orthogonal techniques or linear techniques perform best on a data set. In Chapter 7, the applied prototype is developed for uniform data distributions. Within this assumption, prototype matching proves superior to the MetaL approach in predicting the concept class that performs best. Beyond this assumption of uniform data distribution, the superiority of prototype matching vanishes. The availability of prototypes from the same type of data distribution appears necessary for a satisfying outcome.

Given our results we could argue that guarded technique selection leads to the definition of a new guard: assess the type of data distribution to determine the type of guard to be selected. However, there is a clear complexity reduction, from assessing the type of entropy behavior

from a noisy entropy behavior curve to assessing the type of data distribution in a data set. Consequently, selecting a prototype for a data distribution is much more feasible than selecting a technique for a data set.

8.4. Implications for the Application of Machine Learning Techniques

The MEDIA model, which combines knowledge acquisition, machine learning, and the technique selection tool as described within the dissertation, helps to improve the process of designing industrial applications of ML techniques. The MEDIA model structures the design process in increasing levels of technical detail. The decomposition of a complex problem in sub-tasks is reducible to explicit considerations based on rational criteria with the planning approach, especially when complemented with empirically founded costing models. Technique selection with GTS removes the arbitrariness of technique selection in many applications. Moreover, the motivation of design decisions can be unambiguously documented when the MEDIA model and the other presented tools are applied.

Somerville (1996) defines software engineering as being "*... concerned with methods, tools and techniques for developing and managing the process of creating and evolving software products.*". In parallel, knowledge engineering can be conceived as being concerned with theories, methods and techniques for analyzing and building knowledge systems (cf. Schreiber et al., 2000). In doing so, knowledge engineering uses concepts of software engineering, and extends it with specific analysis and design tools, in addition to techniques to analyze and build knowledge components.

Many of the models that have been proposed for applying ML techniques focus on technical aspects of the technology, and consider it as stand alone technology. Like the next layer of a Russian matruška puppet the MEDIA model, the design approach and the GTS extend knowledge engineering methods with the instruments to handle machine learning techniques.

How does the availability of these concepts influence the application practice? The development of the systems in Chapter 3 and 4 would have gained goal orientation if a MEDIA-like approach would have been available. The collection of data and knowledge sources, the decomposition of the functionality and the selection of techniques could have been more efficient. Whether or not the resulting systems would have been different is debatable. We show in Chapter 6 how the system design process for the PTSS in Chapter 3 could have been approached. Moreover, redesign, adaptation and maintenance of the systems, when required, would have become much more effective.

The justification for the ML technique selection, which in the current situation is based primarily on an assumed *it works best for this data set*, can be replaced by a directed strategy of asking: does the data set still comply with the criteria of the technique? If so, application of the same technique is justified. And if not, a technique that better fits the underlying concept type must be identified.

The work that is reported in this thesis raises a number of supplementary research questions. First, the connection between the MEDIA model and existing software and knowledge engineering approaches can be formalized. The interaction between the MEDIA application level and the embedding system has not been considered. Second, better cost and quality estimation methods are needed to support the planning method of Chapter 6. The empirically grounded methods are especially promising. However, a large research effort is needed in order to generate a reliable method, which means the input of data from a large number of practical projects. And third, the instrument of guards for technique selection needs further

extension. The guard for orthogonal techniques needs to be extended to cover the interesting data distributions, and guards for the most frequently used techniques need to be formulated. This, in return, requires that the fundamental properties of machine learning techniques need to be formulated in terms of data characteristics.

Finally, we emphasize the point that for a broader application of ML technology, it is not only the technical excellence that matters, but also the extent to which the technical capacity can be used by relatively general software and knowledge engineers to solve practical problems. The application potential for ML techniques is massive, especially in an era when data is increasingly routinely generated. Mobile communications, Internet, medical techniques, industrial monitoring systems, customer loyalty programs, tracking and tracing information of product flows, and bioinformatics, to name a few, generate terabytes of data. To effectively manage all these data sets, applicability of technology is as important as their technical competence. This is only possible if the technological development of machine learning is accompanied by the development of methodological concepts. This thesis has made some steps in this direction.

References

Aamodt, A., & Plaza, E. (1994). Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches, *AI Communications 7*, Nr. 1, 39-59

Aarts, R.J. (1992). Knowledge-based planning of mashing profiles, in: M. Nazmul Karim & G. Stephanopoulos (eds), *Modelling and Control of Biotechnical Processes 1992*, Oxford (UK): Pergamon Press, 387-390.

Aarts, R.J. (1995). Computer Planned Recipes for Effective Production Management, in: *Proc. Eur. Brew. Conv. 25th Congress*, Brussels, Oxford (UK): Oxford University Press, 695-703.

Aben, M., & Van Someren, M.W. van (1990). Heuristic Refinement of Logic Programs, in: L.C. Aiello (ed): *Proceedings ECAI--90*, London (UK), Pitman, 7-12.

Aben, M.W.M.M. (1995). *Formal Methods in Knowledge Engineering*, PhD thesis, University of Amsterdam

Adriaans, P. & Zantinge, D. (1996). *Data Mining*, Harlow (UK), Addison-Wesley

Aha, D, Kibler, D. & Albert, M.K.,(1991). Instance Based Learning Algorithms, *Machine Learning*, 6(1), 37-66

Aha, D. & Riddle, P.J. (eds) (1995). *Working notes for applying machine learning in practice*: a workshop at the twelfth international machine learning conference, technical report AIC-95-023, Washington DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence

Baldi, P. & Brunak, S. (2001), *Bioinformatics: the Machine Learning Approach* (2nd ed), Cambridge (MA): MIT Press

Bemelmans, T.M.A. (1994). *Bestuurlijke informatiesystemen en Automatisering* (6th ed, Management Information Systems and Automation, in Dutch), Deventer (NL), Kluwer Bedrijfswetenschappen

Bensusan, H., & Alexandros, K. (2001). Estimating the Predictive Accuracy of a Classifier, in: De Raedt, L. & Flach, P. (eds), *Proceedings of ECML 2001*, LNAI 2167, Berlin (GE): Springer, 25-36

Berthold, M. & Hand, D.J. (1999). *Intelligent Data Analysis, An Introduction*, berlin (GE): Springer

Boehm, B. (1981). *Software Engineering Economics*, Engelwood Cliffs (NJ): Prentice-Hall

Boehm, B.W., Abts, C., Clark, B., and Devnani-Chulani. S. (1997). *COCOMO II Model Definition Manual*. The University of Southern California.

Boerma, T.M. & Van Rijbroek, P.C.L. (1992). *3rd Descriptive Cultivar List For Ornamental Plants* (3e Beschrijvende Rassenlijst voor Siergewassen, in Dutch), Wageningen (NL): CPRO-DLO

Boetticher, G. (2001). "Using Machine Learning to Predict Project Effort: Empirical Case Studies in Data-Starved Domains", in: *Proceedings of Model Based Requirements Workshop*, San Diego, 17 - 24

Bos, J. & Harting, E. (eds) (1998). *Creating in Projects* (Projectmatig creëren, in Dutch), Schiedam (NL): Scriptum Management

Bratko, I., Cestnik, B. & Kononenko, I. (1996). Attribute-based Learning, *AI Communications 9* (1), 27-32

Brazdil, P., Gama, J. & Henery, B. (1994). Characterising the applicability of Classification Algorithms Using Meta-Level Learning, in: Bergadano, F. & Raedt, L. de (eds.), *Proceedings of ECML-94*, Berlin (GE): Springer-Verlag, 83-102

Breuker, J., & VandeVelde, W. (1994). *CommonKADS Library for Expertise Modelling,* Amsterdam (NL): IOS Press

Breunese, A.P.J., Top, J.L., Broenink, J.F. & Akkermans, J.M. (1998 Library of Reusable models: Theory and Application, *Simulation 71* (1), July 1998, 7-21

Brodley, C.E. (1992). Dynamic Automatic Model selection, COINS Technical Report 92-30, University of Massachusetts

Brodley, C.E. (1995a) Recursive bias selection for classifier construction. *Machine Learning,* 20, 63-94.

Brodley, C.E. (1995b). Automatic Selection of Split Criterion during Tree Growing Based on Node Location, in: Prieditis, A. & Russell, S. (eds), *Machine Learning: Proceedings of the Twelfth International Conference*, San Mateo (CA): Morgan Kaufman, 73-80

Brodley, C.E.& Smyth, P. (1995). *The Process of Applying Machine Learning Algorithms*, in: Aha, D.W. & Riddel, P.J. (1995), 7-13

Brodley, C.E., & Smyth, P. (1997). Applying Classification Algorithms in Practice, *Statistics and Computing 7*, 45-56

Broeze, J., Verdenius, F., Sloof, M., Rekswinkel, E. & van der Sluis, H. (1997). Advanced Modelling of Wastewater Treatment Systems, In: Kaylan, A.R. & Lehman, A. (eds) *Proceedings of the European Simulation Multi-conference ESM '97* Istanbul (Tu), 670-674

Chalmers, D.J., (1990), The Evolution of Learning, an Experiment in Genetic Connectionism, in: Touretsky, D.S., Elman, J.L., Sejnowski, T.J. & Hinton, G.E., *Proceedings of the 1990 Connectionist Summer School*, San Mateo (CA): Morgan Kaufmann, 81-90

Chang, S.K. (ed) (2002), *Handbook of Software Engineering and Knowledge Engineering*, Singapore: World-Scientific

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C. & Wirth, R. (2000). *CRISP-DM 1.0, Step-by-step data mining guide*, from the CRISP –DM Web site: www.crisp-dm.org

Clark, P. & T. Niblett (1989). The CN2 Induction Algorithm. *Machine Learning*, 3(4):261-283

Cohen, W.W. (1996). Learning Trees and Rules with Set-Valued Features, in: *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference* (AAAI 96, IAAI 96), Volume 1, Portland (OR): AAAI Press / The MIT Press, 709-716

Consumed (2004). www.consumed.nl, version May 2004, Lemma "Body-mass-index"(in Dutch)

Craw, S., & Sleeman, D. (1990) Automating the refinement of knowledge-based systems. in: Aiello, L. C. (ed.), *Proceedings ECAI--90*, London (UK): Pitman, 167-172

Craw, S., Sleeman, D., Graner, N., Rissakis, M. & Sharma, S. (1992). CONSULTANT: Providing Advice for the Machine Learning Toolbox, in: Bramer, M.A. & Milne, R.W., *Research and Development in Expert Systems IX*, London, 6-23

De Boer, T.W. (1994). *Modelleren van expertise in expertsystemen* (Modeling expertise in expert systems, in Dutch), PhD thesis, Groningen

DeHertogh, A., Aung, L. & Benschop, M. (1983). The Tulip: Botany, Usage, Growth, and Development, in: Janick, J. (ed), *The Tulip: Botany, Usage, Growth, and Development*, Westport (Conn.): AVI Publishing Company, 45-125

DesJardins, M. (1995) Goal-directed learning: a decision-theoretic model for deciding what to learn next. in: Leake, D. & Ram, A. (eds), *Goal-Driven Learning*, MIT Press, pp. 241-250

DTI (1994). *Neural Computing, Learning Solutions*, London (UK): Department of Trade and Industry

Eilers, P.H., Boer, J.M., Van Ommen, G.J., Van Houwelingen, H.C. (2001). Classification of microarray data with penalized logistic regression, in: *Proceedings of SPIE*, 4266: 187-198

Engels, R. (1996). Planning Tasks for Knowledge Discovery in Databases; Performing Task-oriented User-Guidance, in: Simoudis, E., Han, J. & Fayyad, U.M. (eds), *Proceedings of the 2nd Int. Conference on Knowledge Discovery and Data Mining*, AAAI Press, 170-175

Engels, R. (1999). *Component-Based User Guidance in Knowledge Discovery and Data Mining*, PhD thesis, University of Karlsruhe

Engels, R., Aha, D., & Verdenius, F. (eds) (1998). *Proceedings of the ICML'98 workshop "The Methodology of Applying Machine Learning: Problem Definition, Task Decomposition and Technique Selection*, AAAI Press, Technical Report WS-98-16

Engels, R., Evans, B., Herrmann, J. & Verdenius, F. (1997a), *Proceedings of the ICML'97 workshop "Machine Learning Applications in the real world: Methodological Aspects and Implications"*, Nashville (TN)

Engels, R., Evans, B., Herrmann, J. & Verdenius, F. (1997b). Preface, in: Engels et al., 1997a, 2-3

Engels, R., Lindner, G. & Studer, R. (1997c). A Guided Tour Through the Data Mining Jungle, in: Preigibon, D., Heckermann, D. & Mannila, H. (eds), *Proceedings of the 3nd Int. Conference on Knowledge Discovery and Data Mining*, AAAI Press, 170-175

Engels, R. & Studer, R. (1996), User Guidance for Clementine; Towards implementation of a User Guidance Module, Technical report AIFB, University of Karlsruhe

Engels, R., & Theusinger, C. (1998). Using a Data Metric for Offering Preprocessing Advice in Data-mining Applications, in: Prade, H. (ed), *Proceedings of the Thirteenth European Conference on Artificial Intelligence* (ECAI-98), 430-434

Evans, B. & Fischer, D. (1994). Overcoming Process Delays with Decision Tree Induction, *IEEE Expert*, February 1994, 60-66

Falkenhainer, B. & Forbus, K. (1991). Compositional Modelling: finding the right model for the job, *Artificial Intelligence* 51:95-143

Faneyte, D. & Top, J.L. (2004). Reseach Question decomposition: a way to organise research output, in: Nase, A., Van Grootel, G. (eds), *Proceedings of CRIS 2004*, Leuven: 193-194

Fayyad, U.M & Irani, K.B. (1993). Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning, in: Proceedings of IJCAI-93, San Mateo (CA): Morgan Kaufmann: 1022-1027.

Fayyad, U.M., Piatesky-Shapiro, G., & Smyth, P. (1996). From Data Mining to Knowledge Discovery: An Overview, in: Fayyad U.M., Piatetsko-Shapiro, G., Smyth, P. & Uthurusamy, R. (eds.). *Advances in Knowledge Discovery and Data Mining*, Menlo Park (Ca): AAAI-Press, 1-37

Fowler, M., & Scott, K. (2000), *UML distilled, a brief guide to the Standard Object Modeling Language* (4th ed.), Reading (Ma): Addison-Wesley.,

Garner, S.R., Cunningham, S.J., Homes, G., Nevill-Manning, C.G., & Witten, I. (1995), Applying a Machine Learning Workbench: Experience with Agricultural Databases, in: Aha & Riddle (1995): 14-21

Ginsberg, A. (1988). *Refinement of Expert System Knowledge Bases: A Metalinguistic FrameWork for Heuristic Analysis.* Pitman

Giordana, A. & Neri, F. (1996). Genetic Algorithms in Machine Learning, *AI Communications 9* (1), 21-26

Giordana, A., Saitta, L., Bergadano, F., Brancadori, F. & De Marchi, D. (1993). Enigma: A system that learns diagnostic knowledge, *IEEE Transactions on Knowledge and Data Engineering 5* (1), February, 15-28

Giraud-Carrier, C. & Keller, J. (2002). Meta-learning, in: Meij, J. (ed). *Dealing with the Data Flood: Mining Data, Text and Multimedia*, The Hague (NL): STW, 832-844

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Reading (Ma): Addison & Wesley

Guida, G. & Tasso, C. (1994). *Design and Development of knowledge-based systems*, Cichester (UK): John Wiley & Sons

Hand, D.J., Mannila, H. & Smyth, P. (2001). *Principles of Data Mining*, Cambridge (MA): MIT Press

Henze, M., Gujer, W., Mino, T., Matsuo, T., Wentzel, M.C. & Marais, G.V.R. (1995). Activated Sludge Model No.2., *IAWPRC Scientific and Technical Report No.3*, London (UK): IAWQ, ISSN 1025-0913

Holland, J.H., Holyoak, K.J., Nisbett, R.E. &. Thagard, P.R. (1986). *Induction, Processes of Inference, Learning and Discovery*, Cambridge (MA): MIT Press

Holmes, G., & Smith, T.C. (2001), Data Mining, in: Tijskens, L.M.M., Hertog, M.L.A.T.M. & Nicolaï, B.M., *Food Process Modelling*, Cambridge (UK): Woodhead Publising Ltd., 137-155

Hunter, L., (1993), Planning to learn about protein structure, in: Hunter, L., *Artificial Intelligence and Molecular Biology*, AAAI Press / The MIT Press, 259-288

In ‘t Veld, L.J., Koppelaar, H. & Pelikaan, R. (1992). Neurale Begrotingsmodellen (Neural Budgeting Models, in Dutch), *Informatie34* (6), 349-355

Kanters, E. & Wets, G. (1997). ‘A mine is a dark, uninviting and sometimes dangerous place’ (in Dutch), Interview with U.M. Fayyad, *Informatie 39* (9), 29-36

Kiang, M.Y. (2003), A Comparative Assessment of Classification Methods, *Decision Support Systems 35* (4), 441-454

Kibler, D. & Aha, D. (1987). Learning representative exemplars of concepts; An initial study, in: *Proceedings of the Fourth International Workshop on Machine Learning*, UC-Irvine, 24-29

Kodratoff, Y. & Moustakis, V. (1994). *Managing Machine Learning Application Development And Organizational Implementation*, Tutorial notes, ECAI 1994

Kodratoff, Y., Moustakis, V. & Graner, N. (1994). Can Machine Learning Solve My Problem, *Applied Artificial Intelligence 8* (1), 1-31

Koenderink, N.J.J.P., Hulzebos, J.L., Roller, S., Egan, B. & Top, J.L. (2003). *Antimicrobials Online: Concept And Application Of Multidisciplinary Knowledge Exchange In The Food Domain*, paper presented at AfoT-2003, Barcelona

Kohonen, T. (1984). *Self-organisation and Associative Memory*, Berlin (Ge): Springer-Verlag

Kohonen, T. (1995). *Self-Organizing Maps*, Berlin (Ge): Springer-Verlag

Kolodner, J. (1993). *Case Based Reasoning*, San Mateo (Ca): Morgan Kaufmann

Kroese, M., P.M. Wognum, A.M.C. van Rijn, S.M.M. Joosten en N.J.I. Mars (1994) Methods for the development of knowledge systems (Methoden voor het ontwikkelen van kennissystemen, In Dutch), *Informatie 36* (11), 696-706

Kumar, V. (1992). Algorithms for Constraint-Satisfaction Problems, A Survey, *AI Magazine*, Spring 1992, 32-44

Langley, P. (1993). *Workshop on Fielded Applications of Machine Learning*, Final report on ONR grant No N00014-93-1-0209

Langley, P. (1996). *Elements of Machine Learning*, San Fransisco (CA): Morgan Kaufmann

Langley, P., & Simon, H.A. (1994). Applications of Machine Learning and Rule Induction, *Communications of the ACM,* 38(11): 54-64

Larrigaudiere, C., Guillen, P. & Vendrell, M. (1995). Harvest maturity related changes in the content of endogenous phytohormones and quality parameters of melon, *Postharvest Biology and Technology 6* (1/2), 73-80

Lethbridge, T.C. & Skuce, D. (1994). Knowledge base metrics and informality: user studies with CODE4, in: *Proceedings of KAW 1994*, Banff, Alberta, Canada

Levy, L.S. (1987). *Taming the Tiger, software engineering and software economics*, New York (NJ): Springer-Verlag

Lim, T-S, Loh, W-Y. & Shih, Y-S. (2000). A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old, and New Classification Algorithms, *Machine Learning 40* (3): 203 - 228

Lindner, G. & Studer, R. (1999). Algorithm Selection Support for Classification, in: Decker, R. & Gaul, W. (eds), *Classification and Information Processing at the Turn of the Millennium*: 162-169

Loh, W.L. & Shih, Y-S (1997). Split Selection Methods for Classification Trees, *Statistica Sinica*, Vol. 7, 815-840

Lokhorst, C., Udink ten Cate, A.J. & Dijkhuizen, A.A. (eds.) (1996). *Proceedings of the 6th International Congress for Computer Technology in Agriculture (ICTTA '96)*, Wageningen (NL): Agro-Informaticareeks nr. 10

Lovelock, J.(1988). *The ages of Gaia*, Oxford (UK): Oxford University Press

Lundeberg, M., Goldkuhl, G. & Nilsson, A. (1985). *System Development According to ISAC, The ISAC methodology* (Systeemontwikkeling volgens ISAC, De ISAC Methodiek, in Dutch), 2nd edition, Alphen a/d Rijn (NL): Samson Uitgeverij

Mallat, S. (1999). *A wavelet tour of signal processing* (2nd edition), San Diego (CA): Academic Press

Marcus, S. (ed) (1988). *Automatic knowledge acquisition for expert systems.* Boston (MA): Kluwer

McDermott, J., (1988). Preliminary Steps Toward a Taxonomy of Problem Solving Methods, in: Marcus, S. (ed), *Automating Knowledge Acquisition for Expert Systems*; Dordrecht (NL): Kluwer Academic Publishers

Menzies, T. (1999). Cost-Benefits of Ontologies, www.menzies.com

Menzies, T. (2001). Practical Machine Learning for Software Engineering and Knowledge Engineering, in: Chang, S.K., *Handbook of Software Engineering and Knowledge Engineering*, vol 1, 981-02-4973-X; World-Scientific; Available from http://menzies.us/pdf/00ml.pdf

Menzies, T. & Kiper, J.D. (2001). Better reasoning about software engineering activities, in: *Proceedings of ASE 2001*, San Diego(CA), IEEE Computer Society: 391-394

Meseguer, P. (1989). Constraint Satisfaction Problems, An Overview, *AI Communications 2* (1), 3-17.

MetaL (2004). www.metal-kdd.org, version of February 20, 2004

Michalski, R.S. (1994). Inferential Theory of Learning: Developing Foundations for Multistrategy Learning, in: R.S. Michalski & G. Tecuci (eds): *Machine Learning IV; A Multistrategy Approach*, San Francisco (Ca): Morgan Kauffman, 3-61

Michie D. (1995). Problem decomposition and the learning of skills, in N. Lavrac & S. Wrobel (eds), *Machine Learning: ECML-95*, Notes in Artificial Inteligence 912, Berlin (Ge): Springer-Verlag, 17-31.

Michie, D., Spiegelhalter, D.J. & Taylor, C.C. (1994). *Machine Learning, Neural and Statistical Classification*, New York (NJ): Ellis Horwood

Minton, S. (ed), (1993), Machine Learning Methods for Planning, San Mateo (CA): Morgan Kaufmann

Mitchell, T.M., (1997). *Machine Learning*, New York (NJ): McGraw-Hill

Moe, R.& Wickstrom, A. (1973). The effect of storage temperature on shoot growth, flowering and carbohydrate metabolism in tulip bulbs. *Physiol. Plant. 28*, 81-87

Morik, K., Wrobel, S., Kietz, J.-U. & Emde, W. (1993) *Knowledge acquisition and machine learning*, London (UK): Academic Press.

Nijssen, G.M. & Halpin, T.A. (1989), *Conceptual schema and relational database design*, Englewood Cliffs (NJ): Prentice-Hall

Nolan, R.L. (1979). Managing the Crisis in Data Processing, *Harvard Business Review*, March-April, 115-126

O'Hara, K., & Shadbolt, N. (1996). The thin end of the wedge: Efficiency and the generalised directive model methodology. In Shadbolt, N.; O'Hara, K.; & Schreiber, G., eds., *Advances in Knowledge Acquisition*. Berlin (Ge): Springer-Verlag. 33--47.

Paull, R.E. (1993). Tropical fruit physiology and storage potential, in: B.R. Champ, E. Highley & G.I. Johnson (eds.), *Postharvest handling of tropical fruits*, 198-204

Peleg, K. (1989). Method and apparatus for automatically inspecting and classifying different objects, *US Patent 4884696*

Pfahringer, B., Bensusan, H. & Diraud-Carrier, C. (2000). Meta-learning by landmarking various learning algorithms, in: P.Langley (ed), *Proceedings of the Seventeenth International Conference of Machine Learning* (ICML 2000). Stanford (CA): 743-750

Porter, A.A. & Selby, R.W. (1990). Empirically-Guided Software Development Using Metric-Based Classification Trees, *IEEE Software archive 7* (2), University of California, Irvine, October 1989: 46 - 54

Provost, F. & Kohavi, R. (1998). On applied research in machine learning, *Machine learning 30*, n.2/3, p. 127-32

Quinlan, J.R. (1986). Induction of Decision Trees, *Machine Learning*, 1, 81-106

Quinlan, J.R. (1993). *C4.5, Programs for Machine Learning*, San Mateo (CA): Morgan Kaufmann

Rees, A. (1969). Effects of duration of cold treatment on subsequent flowering of tulips. *J. Hort. Sci. 44*, 27-36

Rudström, Å (1995). *Applications of Machine Learning*, Technical report No. 95-018, Department of computer and Systems Sciences, Stockholm Universitet ISSN 1101-8525

Rulequest (2004). www.rulequest.com, version of May, 2004

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. & Lorensen, W. (1991). *Object-oriented modelling and design*, Englewood Cliffs (NJ): Prentice-Hall,

Rumelhart, D.E., Hinton, G.E. & Williams, R.J. (1986). Learning Internal Representations by Error Propagation, in: Rumelhart, D.E. & McClelland, J.L., Parallel Distributed Processing 1, Cambridge (MA): MIT-Press, 318-362.

Saaty, T.L. (1980). *The Analytic Hierarchy Process, Planning, Priority Setting, Resource Allocation*, New York (NJ): McGraw-Hill

Saitta, L. & Neri, F. (1998), Learning in the "Real World", *Machine learning 30 (2-3)*: 133-163

Salzberg, S.L. (1999). On Comparing Classifiers: A Critique of Current Research and Methods, *Data Mining and Knowledge Discovery 1*, 1-12

Schaffer, C., (1993). Selecting a Classification Method by Cross-Validation. *Machine Learning 13*: 135-143

Schaffer, C. (1994a). A Conservation Law for Generalization Performance, in: *Proceedings of the Eleventh International Conference of Machine Learning*, 259-265

Schaffer, C. (1994b). Cross validation, Stacking and Bi-Level Stacking: Meta-Methods for Classification Learning, in: Cheeseman, P. & Oldford, R.W. (eds), *Selecting Models from Data: AI and Statistics IV*, Berlin (Ge): Springer-Verlag, 51-59

Schreiber, A.Th.; Wielinga, B.J.; & Breuker, J.A., (eds.) (1993). *KADS: A Principled Approach to Knowledge-Based System Development*, London (UK): Academic Press.

Schreiber, A.Th.; Akkermans, J.M., Anjewierden, A.A., de Hoog, R., VandeVelde, W. & Wielinga, B.J. (2000). *Engineering of knowledge; The CommonKADS: Methodology*, Cambridge (MA): MIT Press

Scott, A.C., Clayton, J.E. & Gibson, E.L. (1991). *A Practical Guide to Knowledge Acquisition*, Boston (MA): Addison-Wesley

Shapiro, A. (1987) *Structured induction in expert systems*, Wokingham: Addison Wesley.

Shearer, C. & Khabaza, T. (1995). Data Mining by Data Owners- Presenting Advanced Technology to Non-Technologists through the Clementine System, in: G.E. Lasker & X. Liu (eds), *Advances in Intelligent Data Analysis, Volume 1*, 167-171

Shih, Y-S., (2002). *QUEST User Manual*, http://www.stat.wisc.edu/p/stat/ftp/pub/loh/treeprogs/quest/questman.pdf, version of April 17, 2002

Simoudis, E., John, G., Kerber, R., Livezey, B. & Miller, P. (1995). Developing Customer Vulnerability Modles using Data Mining Techniques, in: G. Lasker & X. Liu (eds), *Advances in Intelligent Data Analysis, Volume 1*,, 181-185

Simpson, P.K. (1990). *Artificial Neural Systems, Foundations, Applications and Implementations*, New York (NJ): Pergamon Press

Sleeman, D. (1994). Towards a Technology and a Science of Machine Learning, *AI Communications 7* (1), 29-38

Sloof, M. & Simons, A.E. (1994). Towards a task model for the design of simulation models, in: *Proceedings of the Conference on Modelling and Simulation*, 721-725

Sloof, M., Tijskens, L.M.M. & Wilkinson, E.C. (1996). Concepts for Modelling Quality of Perishable Products, *Trends in Food Science & Technology 7*, 165-171

Soares, C. & Brazdil, P. (2000). Zoomed ranking: Selection of classification algorithms based on relevant performance information. In: Zighed, D.A., Komorowski, J., & Zytkow, J. (eds), *Proceedings of the Fourth European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD2000)*, Berlin (Ge): Springer-Verlag, 126-135

Sommerville, I., (1996). *Software Engineering*, 5th edition, Wokingham (UK): Addison and Wesley, UK

Streif, J. (1989). Ripening behavior of apples with different harvest time (Reifeverhalten von Äpfeln mit Unterschiedlichen Erntetermin, in German), in: *Proceedings of CA-Lagersymposium für Obstffrüchte*, Elbingnode, 72-80

Terpstra, P., van Heijst, G., Wielinga, B., & Shadbolt, N. (1993). Knowledge Acquisition Support Through Generalised Directive Models. In David, J.-M., Krivine, J.-P., & Simmons, R. (eds.), *Second Generation Expert Systems*. Berlin-Heidelberg (Ge): Springer-Verlag. 428-455.

Tijskens, L.M.M. & Polderdijk, J.J. (1996). A generic model for keeping quality of vegetable produce during storage and distribution, *Agricultural Systems 51* (4), 431-452

Timmermans, T., & Hulzebosch, A.A. (1996). Computer vision system for on-line sorting of pot plants using an artificial neural network classifier, *Computers and Electronics in Agriculture (15),* 41-55

Turner, W.S., R.P. Langerhorst, G.F. Hice, H.B. Eilers en A.A. Uijttenbroek (1990), *SDM System Development Methodology* (5_{th} printing), Rijswijk (NL): Cap Gemini Publishing

Turney P.D. (1995). Cost-Sensitive Classification: Empeirical Evaluation of a Hybrid Genetic Decision Tree Induction Algorithm, *JAIR 2*, 369-409

Utgoff, P.E. (1988). Perceptron Trees: A Case Study in Hybrid Concept Representations, in: *Proceedings of the Seventh National Conference of Artificial Intelligence*, 601-606

Valente, A. (1994) Planning, in: Breuker, J. & VandeVelde, W. (1994). *CommonKADS Library for Expertise Modelling,* Amsterdam (NL): IOS Press

Van den Broek, J.G.A. (1981). The evaluation of methods for system development (Het evalueren van methoden voor systeemontwikkeling: In Dutch), in: NGI (1983), *Methodieken voor systeemontwikkeling, NGI bundel 3A*, 14-23, overgenomen uit: Informatie 23 (2)

Van der Ham, F. (2002). *Induction of multivariate decision trees based on orthogonal class boundaries* (Inductie van multivariate beslisbomen gebaseerd op orthogonale klassegrenzen, in Dutch), MSc thesis, Amsterdam

Van Someren, M. (2001). Model class Selection and construction: beyond the Procrustean approach to Machine Learning Applications, in: Paliouras, G., Karkaletsis, V., & Spyropoulos, C.D. (eds), *Machine Learning and Its Applications*, Berlin (Ge): Springer-Verlag, 196-217

Van Someren, M.W., Torres, C. & Verdenius, F. (1997). A Systematic Description of Greedy Optimization Algorithms for Cost Sensitive Generalisation, in: Liu, X. & Cohen, P. (eds), *Proceedings of IDA-97*, Berlin (Ge): Springer-Verlag, 247-258.

Verdenius, F. (1991). A Method of Inductive Cost Optimization, in: Y. Kodratoff (ed), *Proceedings of the Fifth European Working Session on Learning, EWSL-91,* Berlin (Ge): Springer-Verlag, 179-191

Verdenius, F. (1995). Applications of Inductive Learning Techniques: State of the Art, in: Bioch, J.C., Tan, Y.-H. (eds.), *Proceedings of the 7th Dutch Conference on Artificial Intelligence*, Rotterdam (NL), 231-239

Verdenius, F., (1996). Managing Product Inherent Variance During Treatment, *Computers and Electronics in Agriculture 15*, 245-265

Verdenius, F. (1999). Assessing Suitability of Orthogonality Assuming Techniques by Analyzing Entropy Behavior of Numeric Data, in: H. Blockeel, L. Dehaspe, *Proceedings of the ninth Belgian-Dutch Conference on Machine Learning*, 113-120

Verdenius, F. & J. Broeze (1997). Resource Allocation in Wastewater Plants: The Super-Charger Scheduler, in: *Proceedings of ESIT '97*, Aachen (GE), 55-59

Verdenius, F. & J. Broeze (1999). Generalised And Instance Specific Modelling For Biological Systems, *Environmental Modelling & Software 14*, 1999, 339-348

Verdenius, F., & Engels, R. (1997). A Process Model for Developing Inductive Applications, in: Daelemans, W., Flach, P., & Van den Bosch, A., *Proceedings of Benelearn-97*, Tilburg University (NL), 119-127

Verdenius, F., & Hunter, L. (2001), The power and pitfalls of inductive modelling, in: Tijskens, L.M.M., Hertog, M.L.A.T.M. & Nicolaï, B.M., *Food Process Modelling*, Cambridge (UK): Woodhead Publising Ltd., 105-136

Verdenius, F., A.P.H. Seadt, K. Pleijsier & W.A. Vedder (1994). Towards Generic Tools for Managing Treatment and Storage of Fruits and Vegetables, in: Pala , M. (ed), *The Proceedings of the International Symposium on New Applications of Refrigeration to Fruit and Vegetable Processing*, Istanbul, June 8-10, 1994, IIR, 127-134

Verdenius, F. & J.L. Top (1998). Case Based Modelling of Dynamic Systems, in: L.M.M. Tijskens & M.L.A.M. Hertog, *Proceedings of the first international symposium MODEL-IT*, Wageningen 1998, Acta horticultura 476, pp. 279-288

Verdenius, F. & Van Someren, M.W. (1995). Applications of Inductive Learning Techniques: Methodological Issues, in: G.E. Lasker, X. Lui (eds.), *Advances in Intelligent Data Analysis, Volume 1*, Windsor Ontario, Ca, 206-210

Verdenius, F. & Van Someren, M.W. (1997). Applications of Inductive Learning Techniques: A Survey in the Netherlands, *AI Communications 10, nr. 1*, 3-20

Verdenius, F. & Van Someren, M.W. (2002). Detecting Orthogonal Class Boundaries in Entropy Behaviour, paper presented at the twelfth Belgian-Dutch Conference on Machine Learning

Visser, R.G (1993). *Forcing tulips on crates* (De trek van tulpen op kisten, in Dutch), Informatie en Kennis Centrum Akker en Tuinbouw, Lisse (NL), 105 pg.

Wang, C. & Worthington, I. (1979). A non-destructive method for measuring ripeness and detecting core breakdown in 'Bartlett' pears, *Journal of the America Society for Horticultural Science 104* (5), 629-631

Webb, G.L., Wells, J. & Z. Zheng (1998), An experimental Evaluation of Integrating Machine Learning with Knowledge Acquisition, *Machine Learning 35*, 5-21

Weiss, S.M. & Kulikowski, C.A. (1991). *Computer Systems that Learn from Data*, San Mateo (CA): Morgan Kaufmann

Welke, R.J., Kumar, K. & Van Dissel, H.G. (1991). Methodology Engineering (in Dutch), *Informatie 33* (5), 322-328

Wiegerinck, W. & Heskes, T. (2002). Belief Networks/Bayesian Networks, in: Meij, J. (ed). *Dealing with the Data Flood: Mining Data, Text and Multimedia*, The Hague (NL): STW, 660-660

Wirth, N., (1973). *Systematic Programming, An introduction*, Englewood Cliffs (NJ): Prentice-Hall

Wirth, N. (1976). *Algorithms + Data Structures = Programs*, Engelwood Cliffs (NJ): Prentice-Hall

Witten, I.H. & Frank, E. (2000). *Data Mining: practical machine learning tools and techniques with Java implementations*, San Francisco (Ca): Morgan Kaufmann

Summary

This thesis considers methodological support for industrial application of machine learning techniques. Machine learning (ML) research has delivered a large number of promising techniques for tasks such as classification and numerical prediction. Since the mid-eighties, application of ML techniques has gained much interest, both from academic researchers and from industrial workers. ML techniques that build models from exemplary data are considered *ready for real-world application,* at least from the technical point of view.

The models that result from ML application can be incorporated within knowledge based systems. In this mode, ML techniques perform knowledge acquisition and knowledge maintenance for specific inferences. Alternatively, to obtain the same knowledge models, human expertise is elicited by knowledge engineers. When the models are also automatically updated, enabling the system to function in a changing environment, the system becomes adaptive. Another mode of ML application is as data analysis tool. The goal then is to detect patterns in data. The resulting models are exploited for exploring regularities in a domain and, after validation of the models, also for human problem solving. This application has become known under the headings of Data Mining and Knowledge Discovery in Databases.

As the research focus to date has mainly focused on technological issues, the need arises to rationally integrate the resulting techniques in real world applications, and to select, given a specific task, the best technique to perform that task. This thesis starts with a discussion of properties of process models that support the application of machine learning techniques. Numerous process models are available that support the development of information- and knowledge systems in software engineering and knowledge engineering. Typical process models consist of an activity structure and tools that support analysis and design steps.

A number of ML design and analysis process models are proposed in literature. The existing process models focus on the mere application of learning techniques in an exploratory data mining context, sometimes related to one specific technique group, such as neural networks. All these approaches start from the assumption that the final application consists of learning techniques. In real-world situations however, functional (and financial) considerations dominate. Unfortunately, there is no process model available that considers the application of learning techniques in the design process of a complex embedded system, where the deployment of learning techniques has to be considered against alternatives.

In spite of a growing number of reported learning applications in literature, the current application practice is unknown at the beginning of this work. To identify bottlenecks in the application practice and to determine the application requirements of industrial practitioners, a survey of ML applications is organized in 1994. The survey addresses industrial companies, research companies and research institutes in the Netherlands.

The survey results show that ML techniques are often exploratory, and driven by technological availability. The application practice is ad hoc, and makes hardly any use of existing ML tools. Moreover, only a fraction of the available techniques is actually used in practice. Respondents indicate a need for pragmatic support in technique selection and efficient production of working solutions. To update our 1994 results with more recent data, the survey is complemented with recent results of a poll on the KDNuggets Internet site. These results suggest that, though the scale of application has changed (especially due to the success of data mining and knowledge discovery over the last decade), the main results of the 1994 survey still hold, except that tool usage has increased substantially.

Three main questions arise from the results of the survey:

- **Overall**: *Can a process model be formulated that both considers the design issues for machine learning modules as well as the integration of the machine learning module in the embedding system?*
- **Top Down**: *How to decompose a complex functional requirement so that the potential use of ML techniques is recognized and they are properly integrated in the ultimate design?*
- **Bottom Up**: *How to select, as a part of the design process, an appropriate ML technique for realizing the learning behavior as specified in the design?*

Two cases illustrate these issues. The Product Treatment Support System (PTSS) is a planning aid for warehouse and quality managers that store and treat agricultural produce over a long period of time. The system helps them to design batch specific treatment condition recipes (temperature, relative humidity, and gas conditions). In the system, statistical estimation, decision trees, neural networks and constraint satisfaction modules co-operate to match the performance of human experts in the domain.

In the wastewater domain two systems are discussed that represent different ways of incorporating knowledge and ML techniques. The Super Charger Scheduler combines a case-based reasoning approach with an indexing mechanism based on self organizing maps. The system, offering a new, previously non-existent functionality, could not exploit existing knowledge. Moreover, a fast response time and reliable output is more important than fully comprehensible results. The Water Quality Simulation Module on the other hand assembles complex simulation models from partial sub-models on the basis of a high-level model specification. Sub-models, with strict pre- and post-conditions, have to be handled properly. In the assembling process, knowledge on the actual configuration of the plant and on the functionality of the sub-models is used. For this domain, comprehensibility is a crucial factor for acceptance by domain experts. Moreover, the body of available knowledge for this system has been substantial.

The survey results and the experiences with these two systems illustrate the need for an explicit activity to support the mastering of the two signaled design problems: handling available data- and knowledge sources properly, and identifying, from the functional perspective, the opportunities for applying ML techniques. The Method for Designing Induction-based Applications (MEDIA) makes the required functionality foremost in designing machine learning applications.

In the MEDIA model the design process of complex systems is subdivided in three 'levels.' At the *application level,* the system functionality is decomposed. Potential inductive components are identified based on available knowledge and data sources. These inductive components are further designed and implemented in the next two levels. For the design of the knowledge intensive modules, MEDIA relies on existing tools and approaches that have been developed in the knowledge engineering framework, e.g. within CommonKADS. At the *Data Analysis level,* selection of appropriate techniques is supported. At the *Technique level,* tuning of the technique design and technique parameters optimizes the performance of the selected learning technique. Technique optimization guidelines are normally provided by the developer of specific techniques, and not in the MEDIA framework.

The task decomposition that is assumed in the application layer of the MEDIA model can be put into practice with the approach that is developed in Chapter 6. ML components are combined with knowledge-based components dependent on the available data and knowledge sources in the domain. The cost and the potential quality of the components guide the

decomposition of the high level task. At any step in the decomposition, the basic consideration is whether it is worth while to build the component as a learning module, as a knowledge-based module, or to decompose it further, making use of available sources. With the PTSS system, the method is illustrated to derive a solution that balances cost and quality. The resulting decomposition also has a better cost-quality ratio than a pure ML solution or a pure knowledge-based solution. The approach makes use of estimated cost- and quality figures for both knowledge-based and machine learning components. Although an outline of estimators is sketched as an extension of existing software engineering process models, further work is required in this area.

The second tool that is developed within the MEDIA framework supports technique selection, to be used at the data analysis level. One of the outcomes of the survey is that in practice techniques are not selected on 'technical' criteria, but on the basis of availability and other non-functional arguments. The general framework of *guarded technique selection* implies a fundamental shift in how technique selection is conceived. Guarded technique selection (GTS) assumes that techniques (or groups of functionally closely related techniques) are 'supervised' by dedicated software components, the *guards*. A guard checks whether a data set matches with the type of patterns that is suited for its master technique. The result is a *degree of suitability*. Collecting all degrees of suitability gives a selection of techniques that are 'most suitable' for constructing a model from the data set.

This concept is illustrated by constructing the guard for *orthogonal techniques*. Orthogonal techniques are those learning techniques that construct concept descriptions based on class boundaries orthogonal to one attribute axis. This group of techniques includes some of the more commonly applied ML techniques such as univariate decision tree learners. By analyzing the types of class boundaries, and matching these with typical patterns in the entropy behavior of data sets, it is found that opportunities to apply orthogonal techniques can be identified without exhaustive comparative analysis of the data set with all available ML techniques. Within the assumptions that underlie the used prototypes, the identification of orthogonal concept types functions substantially better than the meta-learning alternative of MetaL. Combined with other analytical instruments, prototype matching can help to rationalize the process of technique selection in the near future.

With the MEDIA model, the rational decomposition approach and the guarded technique selection, this thesis delivers a set of coherent tools to support the industrial application of machine learning techniques. The MEDIA model helps practitioners to structure the development process. It changes the commonly applied explorative approach to a more goal-oriented approach, and at the same time offers the opportunity to integrate, in the design phase, knowledge-based and learning components. Finally, this work leads to a more comprehensible selection of ML techniques for a task. The proposed guarded technique selection provides a sound theoretical foundation for technique selection where it formerly was based on more arbitrary considerations. Moreover the maintenance and re-engineering of learning components is facilitated by this approach.

Samenvatting

Dit proefschrift beschrijft onderzoek naar hoe de industriële toepassing van machinaal lerende technieken methodologisch kan worden ondersteund. Onderzoek naar machinaal lerende (vanaf nu: ML) technieken heeft geresulteerd in een groot aantal veelbelovende technieken voor taken zoals classificatie en numerieke voorspelling. Sinds het midden van de tachtiger jaren heeft de toepassing van ML technieken de belangstelling getrokken van zowel academici als praktijkmensen. ML technieken die modellen induceren uit voorbeeld data, de zogenaamde inductief lerende technieken, werden technisch als *klaar voor toepassing in de praktijk* beschouwd.

De modellen die bij toepassing van ML technieken worden verkregen kunnen worden toegepast in combinatie met kennis gebaseerde componenten. ML technieken worden dan als het ware gebruikt voor kennis acquisitie en onderhoud van kennis voor specifieke componenten. Een alternatieve manier om die kennis te verkrijgen is het eliciteren van expertkennis door kennis analisten. Als de ML technieken worden gebruikt om de modellen ook automatisch op basis van nieuwe ervaring te actualiseren, worden de systemen adaptief. Ze kunnen behouden dan hun geldigheid, bijvoorbeeld als de omgeving verandert.

Een andere manier om ML technieken in te zetten is als data analyse instrument. Het doel is dan om patronen in data te ontdekken. De resulterende modellen worden gebruikt om patronen te identificeren, en eventueel door mensen te laten gebruiken bij het nemen van beslissingen of het maken van keuzes. Dit staat bekend als Data Mining of Knowledge Discovery in Databases.

Nadat de onderzoeksaandacht in eerste instantie is uitgegaan naar het ontwikkelen van nieuwe technieken, is er nu ook behoefte de functionaliteit die ML technieken bieden te integreren in complexere praktijktoepassingen, en om gegeven een specifieke taak een geschikte ML techniek te kunnen selecteren. Daarbij is het van belang dat de ontwerpbeslissingen op rationele gronden worden genomen. Dit proefschrift begint met een discussie van procesmodellen die de toepassing van ML technieken ondersteunen. In software engineering en knowledge engineering zijn diverse procesmodellen bekend voor respectievelijk informatie en kennis-systemen. Dergelijke procesmodellen kunnen bestaan uit een activiteitenstructuur en gereedschappen om analyse en ontwerpstappen te kunnen maken.

In de literatuur zijn verschillende procesmodellen voorgesteld voor het ontwerpen en analyseren van ML problemen. De bestaande procesmodellen richten zich met name op de toepassing van ML technieken in een op exploratie gerichte data mining context, soms gekoppeld aan en specifieke techniek klasse, zoals neurale netwerken. Al deze procesmodellen starten vanuit de aanname dat de uiteindelijke oplossing exclusief bestaat uit lerende technieken. In de praktijk echter, tellen functionele (en financiële) overwegingen zwaar, en is het combineren van lerende technieken en expert systemen vaak interessant. Er is echter nog geen procesmodel dat ontwerpers ondersteunt in het beantwoorden van de vraag of een lerende oplossing onderdeel moet worden van een complexe toepassing, waarbij het gebruik van lerende technieken moet worden afgewogen tegen alternatieven.

In weerwil van het groeiende aantal gerapporteerde praktijktoepassingen, was bij het begin van deze studie weinig bekend over hoe ML technieken werden toegepast. Om zicht te krijgen op beperkingen bij het realiseren van praktijktoepassingen, en om de behoefte aan ondersteuning bij toepassers te bepalen, is in 1994 een enquête gehouden naar de stand van zaken bij toepassing van ML technieken. De doelgroep van de enquête waren industriële

bedrijven, onderzoeksbedrijven en kennisinstellingen die gebruik maakten van ML technieken.

Uit deze enquête blijkt dat ML technieken vaak exploratief werden gebruikt, en sterk gedreven door de beschikbaarheid van technieken. Het toepassingsproces werd ad hoc ingericht, en het gebruik van bestaande ML hulpmiddelen was gering. Bovendien werd slechts een klein deel van de beschikbare technieken ook daadwerkelijk in de praktijk gebruikt. Respondenten gaven aan behoefte te hebben aan pragmatische ondersteuning in techniek selectie en het efficiënt ontwerpen van werkende oplossingen. Om de resultaten uit 1994 te actualiseren zijn gegevens gebruikt uit recente internet polls van KDNuggets. Deze gegevens geven aan dat, hoewel de schaal van toepassing veranderd is (met name door het succes van data mining en knowledge discovery in het laatste decennium), de belangrijkste resultaten uit 1994 nog overeind staan. De enige verandering is dat het gebruik van ML tools substantieel lijkt te zijn toegenomen.

De belangrijkste onderzoeksvragen die uit de enquête naar boven komen zijn:

- **Algemeen:** *Is er een procesmodel te formuleren dat zowel de ontwerpvragen voor lerende modules in ogenschouw neemt als de integratie van lerende modules als subsysteem in een complexere toepassing?*
- **Top-down:** *Hoe kan een complexe functionaliteit zo worden gedecomponeerd dat de toepassingsmogelijkheden voor ML technieken worden herkend, en ML toepassingen worden ingebed in het totale systeemontwerp?*
- **Bottom up:** *Hoe kan, als deel van het ontwerpproces, een geschikte ML techniek worden geselecteerd voor het realiseren van het leergedrag dat in het ontwerp wordt gespecificeerd?*

Twee praktijkvoorbeelden illustreren deze punten. Het Product Treatment Support System (PTSS) is een planningshulp voor magazijn- en kwaliteitsmanagers die verantwoordelijk zijn voor de opslag en behandeling van agrarische producten over een langere tijdsperiode. Het systeem helpt bij het ontwerpen van een batchspecifiek recept voor behandelcondities zoals temperatuur, relatieve vochtigheid en gas condities. In het systeem werken statistische schatters, beslisbomen, neurale netwerken en constraint satisfaction modules samen om een kwaliteit te bereiken die vergelijkbaar is met een menselijke domeinexpert.

Op het gebied van afvalwaterverwerking worden twee systemen besproken die ieder een verschillende benadering geven voor het integreren van kennis-gebaseerde en ML technieken. De Super-Charger Scheduler combineert een case-based reasoning benadering met een indexeermechanisme gebaseerd op self-organizing maps. Het systeem, gebouwd voor een nieuwe, niet eerder bestaande, functionaliteit, kon geen gebruik maken van reeds bekende expertise. Omdat het bovendien een procestaak betreft, zijn snelle respons en betrouwbaarheid belangrijker dan de inzichtelijkheid van de geproduceerde planningen. Bij de Water Quality Simulation Module aan de andere kant, stelt op basis van een algemene modelspecificatie complexe modellen van het zuiveringsproces samen uit deelmodellen. Submodellen, met strikte pre- en postcondities moeten op de juiste manier worden ingepast in het totale model. Tijdens het bouwen van het model wordt kennis gebruikt over de actuele configuratie van het zuiveringsproces en over de functionaliteit van de sub-modules. Voor dit domein is inzichtelijkheid van het eindresultaat, en de weg daar naar toe essentieel voor acceptatie door domeinexperts. Bovendien is er in ruime mate kennis aanwezig over dit domein.

De enquête resultaten en de ervaring met deze twee systemen laten zien dat er behoefte is aan een activiteitenmodel om twee problemen de baas te worden: het slim combineren van bestaande kennis- en databronnen, en het identificeren, vanuit het oogpunt van

systeemfunctionaliteit, van mogelijkheden om ML technieken toe te passen. De MEDIA methode voor het ontwerpen van inductiegebaseerde technieken ontwerpt ML toepassingen waarvoor de gevraagde functionaliteit leidend is.

In het MEDIA model is het ontwerpproces voor complexe systemen opgedeeld in drie niveaus. Op het *Toepassingsniveau* wordt de totale systeemfunctionaliteit gedecomponeerd in sub-functies. Op basis van beschikbare kennis- en databronnen binnen het domein worden potentiële inductieve systeemcomponenten geïdentificeerd Deze inductieve systeemcomponenten worden verder ontworpen en geïmplementeerd op de volgende twee niveaus. Voor het ontwerpen van de kennisgebaseerde componenten zijn bestaande tools en benaderingen vanuit de knowledge engineering voorhanden, zoals CommonKADS. Het *Data Analyse Niveau* ondersteunt selectie van geschikte technieken voor een sub-taak. En op het *Techniek Niveau* worden de ontwerp en optimalisatie parameters van de gekozen techniek zo ingesteld dat de prestatie van de ML techniek zo goed mogelijk wordt. Aanwijzingen voor optimalisatie van de techniek worden idealiter door de ontwikkelaars van de techniek verstrekt.

De taak decompositie die in de toepassingslaag wordt uitgevoerd kan gebruik maken van de benadering die in hoofdstuk 6 wordt voorgesteld. Afhankelijk van de beschikbare data- en kennisbronnen in het domein worden ML componenten gecombineerd met kennisgebaseerde componenten. In het ontwerp worden de componenten zo bij elkaar gebracht dat de kosten en opbrengsten geoptimaliseerd worden. Tijdens elke stap in het decompositieproces is de afweging of het de moeite waard is om een component als ML module te bouwen, als kennisgebaseerde module, of om verder te gaan met her decomponeren. Het PTSS dient als voorbeeld om de aanpak te illustreren, waarbij naar de balans tussen kosten en opbrengsten wordt gezocht. De gevonden oplossing is van betere kwaliteit van pure ML of pure kennisgabaseerde oplossingen. De benadering maakt gebruik van geschatte kosten- en kwaliteitsgetallen voor zowel kennisgebaseerde als ML componenten. De contouren van dergelijke schatters zijn wel beschreven, maar daar moet nog meer werk aan gebeuren.

Het tweede instrument binnen het MEDIA raamwerk ondersteunt techniek selectie, en wordt op het analyse niveau ingezet. Een van de uitkomsten van de enquête was dat in de praktijk technieken niet werden geselecteerd op technisch inhoudelijke grond, maar om redenen van ‘beschikbaarheid’ en andere niet-functionele criteria. Het instrument van *guarded technique selection* betekent wel een fundamentele verandering in de wijze waarop techniek selectie wordt uitgevoerd. Guarded techniek selectie (GTS) gaat er van uit dat technieken (of groepen technieken die functioneel van de zelfde principes uitgaan) worden ‘bewaakt’ door specifieke software componenten, de *guards*. Een guard controleert of een data set die kenmerken bezit waar de bewaakte techniek geschikt voor is. Het resultaat van zo een inspectie is een *graad van geschiktheid*. Door al deze graden van geschiktheid gezamenlijk te beoordelen wordt het mogelijk die technieken die het ‘geschiktst’ zijn te selecteren voor het maken van een model uit die dataset.

Dit concept wordt geïllustreerd door een guard voor *orthogonale technieken* te ontwikkelen. Orthogonale technieken zijn die lerende technieken die concept beschrijvingen maken met klassegrenzen die orthogonaal staan op één attribuut as. Deze techniekgroep bevat enkele van de regelmatig toegepaste ML technieken, zoals univariate beslisbomen. Door de types klassegrenzen te analyseren, en te vergelijken met het entropie gedrag van datasets is ontdekt dat de mogelijkheden om orthogonale technieken succesvol toe te passen kan worden vastgesteld zonder uitputtende vergelijkende analyse van de data ste met alle beschikbare ML technieken. Onder de aannames die gemaakt zijn bij het opstellen van de prototypes is de identificatie van orthogonale concept types substantieel beter dan het beschikbare alternatief meta learning in MetaL. Gecombineerd met andere analytische instrumenten kan prototype

matching helpen om het proces van techniek selectie in de nabije toekomst op rationele gronden uit te voeren.

Met het MEDIA model, de rationele taak decompositie en de guarded techniek selectie levert dit proefschrift een verzameling coherente gereedschappen om de industriële toepassing van machine learning technieken te ondersteunen. Het MEDIA model helpt toepassers het ontwikkelproces te structureren. Het verandert de exploratieve benadering in een meer doelgerichte. Tegelijkertijd biedt het de kans om in de ontwerpfase kennisgebaseerde en lerende componenten te combineren. Bovendien leidt dit werk tot een inzichtelijker selectie van ML technieken voor een taak. De voorgestelde guarded techniek selectie geeft aan techniekselectie een gezonde theoretische basis, waar deze voorheen gebaseerd was op meer willekeurige argumenten. Ten slotte worden met de instrumenten uit het MEDIA raamwerk het onderhoud en herontwerp van inductiegebaseerde toepassingen beter ondersteund.

Volumes that appeared in the series SIKS dissertations:

1998

1998-1 Johan van den Akker (CWI) - *DEGAS - An Active, Temporal Database of Autonomous Objects*

1998-2 Floris Wiesman (UM) - *Information Retrieval by Graphically Browsing Meta-Information*

1998-3 Ans Steuten (TUD) - A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective

1998-4 Dennis Breuker (UM) - *Memory versus Search in Games*

1998-5 E.W. Oskamp (RUL) - *Computerondersteuning bij Straftoemeting*

1999

1999-1 Mark Sloof (VU) - *Physiology of Quality Change Modelling; Automated modelling of Quality Change of Agricultural Products*

1999-2 Rob Potharst (EUR) - *Classification using decision trees and neural nets*

1999-3 Don Beal (UM) - *The Nature of Minimax Search*

1999-4 Jacques Penders (UM) - *The practical Art of Moving Physical Objects*

1999-5 Aldo de Moor (KUB) - *Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems*

1999-6 Niek J.E. Wijngaards (VU) - *Re-design of compositional systems*

1999-7 David Spelt (UT) - *Verification support for object database design*

1999-8 Jacques H.J. Lenting (UM) - *Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation*

2000

2000-1 Frank Niessink (VU) - *Perspectives on Improving Software Maintenance*

2000-2 Koen Holtman (TUE) - *Prototyping of CMS Storage Management*

2000-3 Carolien M.T. Metselaar (UVA) - *Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief*

2000-4 Geert de Haan (VU) - *ETAG, A Formal Model of Competence Knowledge for User Interface Design*

2000-5 Ruud van der Pol (UM) - *Knowledge-based Query Formulation in Information Retrieval*

2000-6 Rogier van Eijk (UU) - *Programming Languages for Agent Communication*

2000-7 Niels Peek (UU) - *Decision-theoretic Planning of Clinical Patient Management*

2000-8 Veerle Coupé (EUR) - *Sensitivity Analyis of Decision-Theoretic Networks*

2000-9 Florian Waas (CWI) - *Principles of Probabilistic Query Optimization*

2000-10 Niels Nes (CWI) - *Image Database Management System Design Considerations, Algorithms and Architecture*

2000-11 Jonas Karlsson (CWI) - *Scalable Distributed Data Structures for Database Management*

2001

2001-1 Silja Renooij (UU) - *Qualitative Approaches to Quantifying Probabilistic Networks*

2001-2 Koen Hindriks (UU) - *Agent Programming Languages: Programming with Mental Models*

2001-3 Maarten van Someren (UvA) - *Learning as problem solving*

2001-4 Evgueni Smirnov (UM) - *Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets*

2001-5 Jacco van Ossenbruggen (VU) - *Processing Structured Hypermedia: A Matter of Style*

2001-6 Martijn van Welie (VU) - *Task-based User Interface Design*

2001-7 Bastiaan Schonhage (VU) - *Diva: Architectural Perspectives on Information Visualization*

2001-8 Pascal van Eck (VU) - *A Compositional Semantic Structure for Multi-Agent Systems Dynamics*

2001-9 Pieter Jan 't Hoen (RUL) - *Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes*

2001-10 Maarten Sierhuis (UvA) - *Modeling and Simulating Work Practice BRAHMS: a multiagent modeling and simulation language for work practice analysis and design*

2001-11 Tom M. van Engers (VUA) - *Knowledge Management: The Role of Mental Models in Business Systems Design*

2002

2002-01 Nico Lassing (VU) - *Architecture-Level Modifiability Analysis*

2002-02 Roelof van Zwol (UT) - *Modelling and searching web-based document collections*

2002-03 Henk Ernst Blok (UT) - *Database Optimization Aspects for Information Retrieval*

2002-04 Juan Roberto Castelo Valdueza (UU) - *The Discrete Acyclic Digraph Markov Model in Data Mining*

2002-05 Radu Serban (VU) - *The Private Cyberspace Modeling Electronic Environments inhabited by Privacy-concerned Agents*

2002-06 Laurens Mommers (UL) - *Applied legal epistemology; Building a knowledge-based ontology of the legal domain*

2002-07 Peter Boncz (CWI) - *Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications*

2002-08 Jaap Gordijn (VU) - *Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas*

2002-09 Willem-Jan van den Heuvel(KUB) - *Integrating Modern Business Applications with Objectified Legacy Systems*

2002-10 Brian Sheppard (UM) - *Towards Perfect Play of Scrabble*

2002-11 Wouter C.A. Wijngaards (VU) - *Agent Based Modelling of Dynamics: Biological and Organisational Applications*

2002-12 Albrecht Schmidt (Uva) - *Processing XML in Database Systems*

2002-13 Hongjing Wu (TUE) - *A Reference Architecture for Adaptive Hypermedia Applications*

2002-14 Wieke de Vries (UU) - *Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems*

2002-15 Rik Eshuis (UT) - *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*

2002-16 Pieter van Langen (VU) - *The Anatomy of Design: Foundations, Models and Applications*

2002-17 Stefan Manegold (UVA) - *Understanding, Modeling, and Improving Main-Memory Database Performance*

2003

2003-01 Heiner Stuckenschmidt (VU) - *Ontology-Based Information Sharing in Weakly Structured Environments*

2003-02 Jan Broersen (VU) - *Modal Action Logics for Reasoning About Reactive Systems*

2003-03 Martijn Schuemie (TUD) - *Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy*

2003-04 Milan Petkovic (UT) - *Content-Based Video Retrieval Supported by Database Technology*

2003-05 Jos Lehmann (UVA) - *Causation in Artificial Intelligence and Law - A modelling approach*

2003-06 Boris van Schooten (UT) - *Development and specification of virtual environments*

2003-07 Machiel Jansen (UvA) - *Formal Explorations of Knowledge Intensive Tasks*

2003-08 Yongping Ran (UM) - *Repair Based Scheduling*

2003-09 Rens Kortmann (UM) - *The resolution of visually guided behaviour*

2003-10 Andreas Lincke (UvT) - *Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture*

2003-11 Simon Keizer (UT) - *Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks*

2003-12 Roeland Ordelman (UT) - *Dutch speech recognition in multimedia information retrieval*

2003-13 Jeroen Donkers (UM) - *Nosce Hostem - Searching with Opponent Models*

2003-14 Stijn Hoppenbrouwers (KUN) - *Freezing Language: Conceptualisation Processes across ICT-Supported Organisations*

2003-15 Mathijs de Weerdt (TUD) - *Plan Merging in Multi-Agent Systems*

2003-16 Menzo Windhouwer (CWI) - *Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouses*

2003-17 David Jansen (UT) - *Extensions of Statecharts with Probability, Time, and Stochastic Timing*

2003-18 Levente Kocsis (UM) - *Learning Search Decisions*

2004

2004-01 Virginia Dignum (UU) - *A Model for Organizational Interaction: Based on Agents, Founded in Logic*

2004-02 Lai Xu (UvT) - *Monitoring Multi-party Contracts for E-business*

2004-03 Perry Groot (VU) - *A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving*

2004-04 Chris van Aart (UVA) - *Organizational Principles for Multi-Agent Architectures*

2004-05 Viara Popova (EUR) - *Knowledge discovery and monotonicity*

2004-06 Bart-Jan Hommes (TUD) - *The Evaluation of Business Process Modeling Techniques*

2004-07 Elise Boltjes (UM) - *Voorbeeldig onderwijs; voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes*

2004-08 Joop Verbeek(UM) - *Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise*

2004-09 Martin Caminada (VU) - *For the Sake of the Argument; explorations into argument-based reasoning*

2004-10 Suzanne Kabel (UVA) - *Knowledge-rich indexing of learning-objects*

2004-11 Michel Klein (VU) - *Change Management for Distributed Ontologies*

2004-12 The Duy Bui (UT) - *Creating emotions and facial expressions for embodied agents*

2004-13 Wojciech Jamroga (UT) - *Using Multiple Models of Reality: On Agents who Know how to Play*

2004-14 Paul Harrenstein (UU) - *Logic in Conflict. Logical Explorations in Strategic Equilibrium*

2004-15 Arno Knobbe (UU) - *Multi-Relational Data Mining*

2004-16 Federico Divina (VU) - *Hybrid Genetic Relational Search for Inductive Learning*

2004-17 Mark Winands (UM) - *Informed Search in Complex Games*

2004-18 Vania Bessa Machado (UvA) - *Supporting the Construction of Qualitative Knowledge Models*

2004-19 Thijs Westerveld (UT) - *Using generative probabilistic models for multimedia retrieval*