Descriptive and Predictive Modelling Techniques for Educational Technology

Wilhelmiina Hämäläinen

Licentiate thesis August 10, 2006
Department of Computer Science
University of Joensuu

Descriptive and Predictive Modelling Techniques for Educational Technology

Wilhelmiina Hämäläinen

Department of Computer Science
P.O. Box 111, FI-80101 Joensuu, FINLAND.

Licentiate thesis

Abstract

Data-driven models are the basis of all adaptive systems. Adaption to the user requires that the models are driven from real user data. However, in educational technology real data is seldom used, and all general-purpose learning environments are predefined by the system designers.

In this thesis, we analyze how the existing knowledge discovery methods could be utilized in implementing adaptivity in learning environments. We begin by defining the domain-specific requirements and restrictions for data modelling. These properties constitute the basis for the analysis, and affect all phases of the modelling process from the selection of the modelling paradigm and data preprocessing to model validation.

Based on our analysis, we formulate general principles for modelling educational data accurately. The main principle is the interaction between descriptive and predictive modelling. Predictive modelling determines the goals for descriptive modelling, and the results of descriptive modelling guide the predictive modelling.

We evalute the appropriateness of existing dependency modelling, clustering and classification methods for educational technology, and give special instructions for their applications. Finally, we propose general principles for implementing adaptivity in learning environments.

Computing Reviews (1998) Categories and Subject Descriptors:

G.3 Probability and statistics
H.1 Models and principles

H.3 Information storage and retrieval
I.6 Simulation and modelling
I.2 Artificial intelligence
K.3 Computers and education

General Terms:

Knowledge discovery, educational technology

Additional Key Words and Phrases:

Data mining, machine learning, adaptive learning environments

Acknowledgments

First of all, I want to thank my supervisor, professor Erkki Sutinen for a challenging research topic, valuable comments and support. I am also grateful to professor Hannu Toivonen for his good advice and critical comments.

Several people have helped me in the empirical part of this research. Osku Kannusmäki has been very helpful to give me all *ViSCoS* data I needed. Teemu H. Laine has done a great work as my assistant in preprocessing the *ViSCoS* data. In addition, he has constructed and tested the preliminary multiple linear regression models under my supervision. Ismo Kärkkäinen has kindly offered me his k-means clustering tool and guided how to draw contours by *Matlab*. Ph.D. Kimmo Fredriksson has adviced me in using *Gnuplot*. My students, Fedor Nikitin and Yuriy Lakhtin have made preliminary experiments to model *ViSCoS* data by neural networks, and Mikko Vinni has experimented the enlargements of naive Bayes models.

Finally, I want to express my gratefulness to my beloved, Ph.D. Kimmo Fredriksson, and our cat Sofia for all support and understanding.

Contents

Chapter 1

Introduction

In all empirical science we need paradigms – good instructions and practices, how to succeed. Educational technology is still a new discipline and such practices are just developing. Especially paradigms for modelling educational data by knowledge discovery methods have been missing. Individual experiments have been made, but the majority of research has concentrated on other issues, and the full potential of these techniques has not been realized.

One obvious reason is that educational technology is an applied science and many other disciplines of computer science are involved. Most of the researchers in the domain have expertise in educational science, while few master machine learning, data mining, algorithmics, statistics, and other relevant disciplines. The other reason is domain-specific: the educational data sets are very small – the size of a class – and specialized techniques are needed to model the data accurately.

This thesis tries to meet this need and offer a wide overview of modelling paradigms for educational purposes. These paradigms concern both descriptive modelling – discovering new information in educational data – and predictive modelling – predicting learning outcomes. Special emphasis is put on selecting and applying appropriate modelling techniques for small educational data sets.

1.1 Motivation: modelling *ViSCoS* data

University of Joensuu has a distance learning program *ViSCoS* (*Virtual Studies of Computer Science*) [HSST01] which offers high-school students a possibility to study university-level computer science courses. The program has been very popular, but

the programming courses have proved to be very difficult and the drop-out and failing rates have been large. To handle this problem, we have begun researching ways to recognize the likely drop-out or failure candidates in time. This is valuable information for the course tutors, but also a critical element for developing adaptivity in the learning environment.

After discussing with *ViSCoS* teachers, we have defined the following goals for this research:

1. **Identify common features for drop-outs, failed and skilled students:**
   Given the final results of the course, we should try to identify common features shared by drop-outs, failed and skilled students. As a sub-problem, we should discover whether these students fall into any homogeneous groups.

2. **Detect critical tasks which affect course success:**
   Given the task points and final results, we should identify tasks that indicate drop-out, failure or excellent performance. It is possible that the tasks contain some "bottle-neck" tasks, which divide the students.

3. **Predict potential drop-outs, failed or skilled students:**
   Given the exercise task points in the beginning of course, we should predict who will likely drop out, fail or succeed excellently. The predictions should be made as early as possible, so that the tutoring could be adapted to student's needs. Predicting the drop-outs and failed students is the most critical, because they should be intervened immediately. However, specially skilled students would also benefit from new challenges.

4. **Identify the skills measured by tasks:**
   Given the course performance data, we should try to identify the skills measured by the exercise tasks. According to course designers, the tasks measure different skills, but such evaluations are always subjective. As a modest goal, we could at least try to find groupings of tasks.

5. **Identify student types:**
   Given the task points and grouping of tasks, we should try to discover, if the students fall into any clear groups. Supposing that the task groups measure different skills, we can further study the relations between skills and course results.

In the following chapters, we will process this problem further. We will define the modelling goals, select the appropriate modelling paradigms, analyze the available data, preprocess it and finally construct various models.

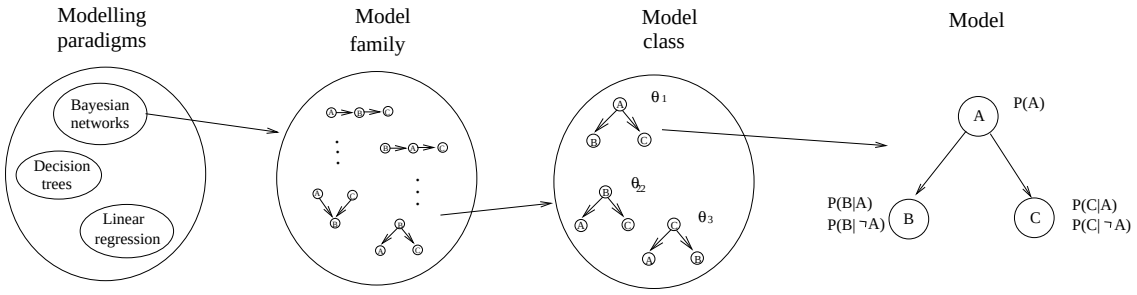

Figure 1.1: The hierarchy of modelling concepts with examples from Bayesian networks.

1.2 Research methods and contributions

The main problem of this research is to develop general principles for modelling educational data. This probelm can be divided into five subproblems (the concepts are defined in Table 1.1 and illustrated in Figure 1.1):

1. Selecting the modelling paradigm.
2. Defining the model family: the set of possible model structures is restricted by defining the variables and optionally some constraints on their relations.
3. Selecting the model structure: the optimal model structure is learnt from data or defined by an expert.
4. Selecting the model: the optimal model parameters are learnt from data.
5. Validation: the descriptive or predictive power of the model is verified.

The scientific paradigm in computer science contains general principles how to perform optimally in each of these subproblems. However, each domain has its own requirements and restrictions. The general principles should be applied and revised and new principles created according to domain-specific needs. In educational technology, such principles have been missing. This is exactly the goal of this research. First we should determine the specific requirements and restrictions of data modelling for educational purposes. Then we should select, apply and revise the existing modelling principles for our needs.

The principles proposed in this thesis have been derived from three components (Figure 1.2): general modelling principles, and the specific requirements and restrictictions in educational technology.

Table 1.1: The basic concepts concerning modelling.

Concept	**Definition**
Model	A model $\mathcal{M} = (S, \theta)$ is defined by a model structure S and assigned parameter values θ.
Model structure	Model structure S defines the variables and their relations in the model. E.g. in a Bayesian network model, the structure consist of a graph, whose nodes define variables and edges define the dependencies between variables.
Model parameters	Model parameters θ are assigned numerical values, which measure the variables and their relations in the model. E.g. in a Bayesian network model, the parameters define the prior probabilities of the root nodes and conditional probabilities associated to edges.
Model family	A model family is a set of models, whose model structure belongs to some fixed set T: $\{M_i \mid M_i = (S_i, \theta_i) \mid S_i \in T\}$. E.g. all Bayesian networks with variables A, B and C.
Model class	Model class is a set of models, which have the same structure S: $\{M_i \mid M_i = (S, \theta_i)\}$.
Modelling paradigm	Modelling paradigm is a set of definitions, assumptions, and methods for constructing and using certain kind of models. E.g. Bayesian networks, neural networks, decision trees, multiple linear regression. The paradigms can be divided into several sub-paradigms, like different types of neural networks.

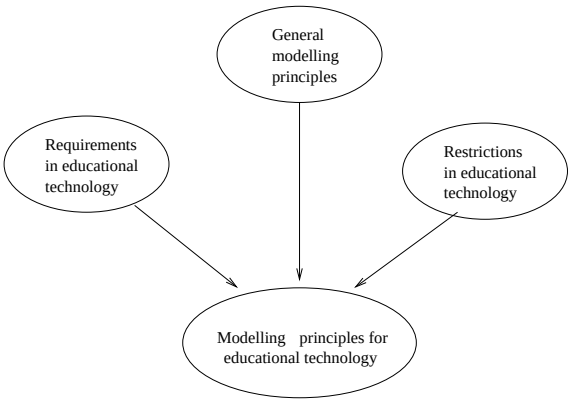

Figure 1.2: The main components of the research.

The main requirement in educational systems is that the model is *robust*, i.e. it is not sensitive to small variations in data. This is crucial, because the data changes often. The model should adapt to new students, new teachers, new exercise tasks or updated learning material, which all affect the data distribution. For the same reason, it is desirable that the model can be updated, when new data is gathered. A special requirement in educational technology is that the model is *transparent*, i.e. the student should understand, how the system models her/him (e.g. [OBBE84]).

The main restrictions cumulate from educational data. According to our analysis, the educational data sets are typically very small – only 100-300 rows. The attributes are mixed, containing both numeric and categorial values. In additon, there are relatively many *outliers*, exceptional data points, which do not fit the general model.

Based on these assumptions, we have selected the most appropriate modelling paradigms for educational applications. We have applied them to *ViSCoS* data and created new methods for our needs. We have collected and derived several good rules of thumb for modelling educational data. All these principles are combined into a general framework for implementing adaptivity in educational systems in a data-driven way.

The main contributions of this thesis are the following:

- Giving a general overview of possibilities and restrictions of applying knowledge discovery to educational domain.
- Constructing a general framework for implementing adaptivity in learning systems based on real data.

- Evaluation of alternative dependency modelling, clustering and classification techniques for educational purposes.
- Constructing accurate classifiers from small data sets of *ViSCoS* data for predicting course success during the course.

Most of the methods have been implemented by the author. If other tools are used, they are mentioned explicitely.

Some of the results have been represented in the following articles:

- Hämäläinen, W.: General paradigms for implementing adaptive learning systems [Häm05].
- Hämäläinen, W., Laine, T.H. and Sutinen, E., H. Data mining in personalizing distance education courses [HLS06].
- Hämäläinen, W., Suhonen, J., Sutinen, E. and Toivonen, H. Data mining in personalizing distance education courses [HSST04].
- Hämäläinen, W. and Vinni, M.: Comparison of machine learning methods for intelligent tutoring systems [HV06].

The third article was accepted, but not published in the conference proceedings due to technical problems. The author has been the main contributor in all papers.

1.3 Organization

The organization of this thesis is the following: In Chapter 2, previous work on applying knowledge discovery techniques in educational domain is presented. In Chapter 3, we discuss about the general modelling principles and give guidelines for modelling educational datasets. In Chapter 4, typical educational data is analyzed and guidelines for feature extraction, selection and other preprocessing techniques are given. In Chapter 5, we evaluate the most appropriate dependency modelling techniques for educational data. In Chapter 6, the existing clustering methods for educational purposes are analyzed and our own clustering method is introduced. In Chapter 7, we analyze the suitability of existing classification methods for educational data. The best methods are compared empirically with the *ViSCoS* data. In Chapter 8, a general framework for constructing an adaptive educational system from data is introduced. The final conclusions are drawn in Chapter 9. The basic concepts and notations are introduced in Appendix 9.

Chapter 2

Background and related research

In this chapter, we describe the main ideas of knowledge discovery and its applications in educational technology. We report previous experiments on applying knowledge discovery methods in the educational domain.

2.1 Knowledge discovery

Knowledge discovery in databases (*KDD*), or briefly knowledge discovery, is a subdiscipline of computer science, which aims at finding interesting regularities, patterns and concepts in data. It covers various techniques from traditionally separated disciplines of machine learning, data mining and statistics. However, knowledge discovery is not just a combination of data mining and machine learning techniques, but a larger iterative process, which consists of several phases (Figure 2.1).

Understanding the domain is the starting point for the whole *KDD* process. We should understand both the problem and data in a wider context: what are the needs and goals, what kind of and how much data is in principle available, what policies restrict the access, etc. The preprocessing phase is often very worksome. We have to select suitable data sources, integrate heterogeneous data, extract attributes from unstructured data, try to clean distorted data, etc. The educational data exists in many forms and often we have to do some manual work. Discovering patterns is the actual data mining or machine learning task. Given a good preprocessed data set it is quite straight-forward, but often we have to process in a trial and error manner, and return back to prepare better data sets. In the postprocessing phase the patterns are further selected or ordered and presented (preferably in a visual

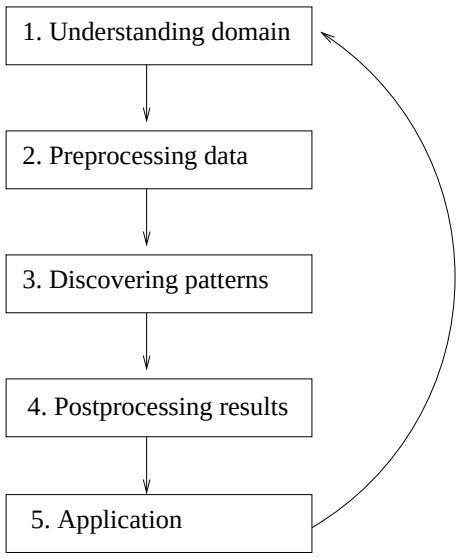

Figure 2.1: Knowledge discovery process.

form) to the user. Finally, the results are taken into use, and we can begin a new cycle with new data.

In the current literature, concepts "knowledge discovery", "data mining" and "machine learning" are often used interchangeably. Sometimes the whole *KDD* process is called data mining or machine learning, or machine learning is considered as a subdiscipline of data mining. To avoid confusion, we follow a systematic division to *descriptive* and *predictive modelling*, which matches well with the classical ideas of data mining and machine learning[1]. This approach has several advantages:

- The descriptive and predictive models can often be paired as illustrated in Table 2.1. Thus, descriptive models indicate the suitability of a given predictive model and can guide the search of models (i.e. they help in the selection of the modelling paradigm and the model structure).
- Descriptive and predictive modelling require different validation techniques. Thus the division gives guidelines, how to validate the results.
- Descriptive and predictive models have different underlying assumptions (bias) about good models. This is reflected especially by the score functions, which guide the search.

[1]It matches also the division to *unsupervised* and *supervised learning* used in machine learning literature.

Table 2.1: Examples of descriptive and predictive modelling paradigm pairs. The descriptive models reveal the suitability of the corresponding predictive model and guide the search.

Descriptive paradigm	**Predictive paradigm**
Correlation analysis	Linear regression
Associative rules	Probabilistic rules
Clustering	Classification
Episodes	Markov models

We will return to the interaction between descriptive and predictive modelling in Chapter 3. Now we will give a brief historical review of classical data mining and machine learning to illustrate the differences of these two approaches. The main differences are summarized in Table 2.2.

Table 2.2: Summary of differences between classical data mining and machine learning.

	Data mining	Machine learning
Assumptions:	Data is primary, it does not necessarily have any structure or only local patterns.	Data is secondary, produced by some underlying global model, which should be learnt.
Goals:	To find all interesting patterns. which describe the data set	To learn one or few carefully defined models, which can be used to predict future events.
Models:	Simple models or local patterns.	Relatively complex, global models.
Emphasis:	The whole *KDD* process.	Only the learning step.
Source of data:	Often collected as a side product of some other project.	Specially selected for the given learning task.
Amount of data:	Huge data sets, even millions of rows.	Only hundreds or thousands of examples in a training set.

The main difference of data mining and machine learning concerns the goals of modelling and the underlying ontological (and epistemic) assumptions. In machine learning the main goal is to learn a *model*, which can be used to predict future events. The underlying assumption is that there is a model, which has produced the data. The model is *global*, because it explains the whole data set, but more important is that it generalizes to new data. By contrast, in data mining, the main goal is to discover new interesting information which describes the current data set. Often, we do not even try to find global models, but only *local patterns*, which describe some parts of data. Now the data itself is primary, and we do not assume the existence of any models, before they are discovered. According to these emphases, the primary

task of machine learning is called *prediction task* or *model-first approach*, while the primary task of data mining is called *descriptive task* or *data-first approach*.

The other differences between machine learning and data mining are more practical. In machine learning, we can search more complex models, because the task is more focused. We want to learn only one or few carefully defined models, which predict the variable in question. In data mining, the task is less focused, because we want to discover all interesting patterns. The patterns are usually quite simple, because we have no a priori assumptions what kind of patterns to search, and we have to try all possible patterns. In addition, the data sets in traditional data mining are huge, and we could not search complex models efficiently. Typically, the origin of data sets is also different: in machine learning, the training set has been specially selected for the learning task, while in data mining, the data may be originally collected for some other purpose.

Both data mining and machine learning owe a great deal to statistics, and many new ideas have born in the interaction between the disciplines. Data mining and machine learning have naturally more emphasis on computational issues and they have been able to develop efficient search methods for complex models (machine learning) and exhaustive search in large data sets (data mining). The volume of data can be seen as one difference, although both data mining and machine learning methods can be applied to small data sets, as well, as we will demonstrate in this thesis. Statistics has also an important role in validation of results both in data mining and machine learning. This topic is further discussed in Chapter 3.

2.2 Knowledge discovery in educational technology

One of the main main goals of educational technology is to adapt teaching to individual learners. Every teacher knows that learning is most effective, when the teacher knows all her/his students and can differentiate the teaching to their individual needs. In large classes and distance learning this is not possible, and new technology is needed.

In the simplest form, a learning system gives the teachers more information about students. For example, in our *ViSCoS* project, the most acute need is to predict potential drop-outs and failing students as early as possible. The actual tutoring actions are left for the teacher's responsibility. In the other extreme are *intelligent tutoring systems* (*ITS*) (see e.g. [dB00, KV04, CCL03, Was97, Web96, SA98,

CHZY03]), where the whole tutoring is automated. This is achieved by two components: a *student model* and a *tutoring module*. The student model consists of a set of features describing the student's current state. Typically, it contains cognitive information (knowledge level, prerequisite knowledge, performance in tests, errors and misconceptions), user preferences (learning style, goals and habits), action history, and maybe some additional information about learner's attitudes, emotions and motivation. Some features can be directly observed, but most of them are derived from observations. Given the student model, the tutoring module selects the most suitable actions according to some rules. These actions involve generating tests and exercises, giving hints and explanations, suggesting learning topics, searching learning material and collaborative partners, etc.

The problem of current intelligent tutoring systems is that they are stable. The student model and actions are usually determined by fixed rules. Even if more sophisticated methods like Bayesian networks are used, the model is predefined by the system designers. According to our literature review, none of the existing intelligent tutoring systems learns the model from real student data. Thus, the adaptivity of such systems means that the students are adapted to some existing model or theory, instead of adapting the model to reality of students.

Implementing the whole intelligent tutoring system based on real data can be impossible. The available data sets are very small, typically the size of a class, and complex models cannot be learnt accurately from data. However, half-automated learning systems or elements of adaptive learning environments can be learnt from data. In simple, single-purpose systems we can collect large amounts of data, because the same task occurs again and again in different variations. This data can be used to learn a model, which predicts the student's success in the next task, for example the next error ([MM01]) or the next help-request ([BJSM03]). The student's general knowledge level can be approximated by her/his expected course score. In a coarse level (e.g. fail/success), the score can be predicted quite accurately already in the beginning of course ([KPP03]). The students' typical navigation paths in the learning material can be analyzed and used to suggest appropriate material for new students ([SB03]). The students' knowledge-level errors or misconceptions can be revealed by analyzing their solutions ([SNS00]). This is important information for the teachers, but also for automatic feedback generation.

2.3 Related research

Current adaptive educational systems are very seldom based on real data. However, there has been some research to this direction. To get a complete picture of the

current state of affairs, we have performed a meta-analysis on research literature addressing either predictive or descriptive modelling on educational data. For this review, we have systematically checked all proceedings of *Intelligent Tutoring Systems* and volumes of *International Journal of Artificial Intelligence in Education* since 1998. In addition, we have searched papers in *ACM*, *IEEE* and *Elsevier* digital libraries with appropriate keywords, and checked all relevant references. This massive search resulted over 100 candidate papers, which promised to apply machine learning, data mining or other intelligent techniques on educational data, but only 23 of them described actual experiments using real data. 15 of them address predictive modelling and the rest 8 descriptive modelling.

2.3.1 Predictive models learnt from educational data

In Table 2.3, we have summarized all experiments, which have learnt predictive models from educational data. In some cases, the data was artificially generated, and the applicability to real data remained uncertain.

Predicting academic score or success in studies

In the first five experiments the goal was to predict the academic score or success in studies.

In [KPP03], drop-out was predicted based on demographic data (sex, age, marital status, number of children, occupation, computer literacy, job associated with computers) and the course data in the first half of the course (scores in two writing assignments and participation to group meetings). Six classification methods (naive Bayes, decision tree, feed-forward neural network, support vector machine, 3-nearest neighbour and logistic regression) were compared with different attribute sets. All the methods achieved approximately the same accuracy. In the beginning of the course, given only demographic data, the accuracy was 63%, and in the middle of the course, given all attributes, it was 80%. Naive Bayes classifier achieved the best overall accuracy, when all results in all test cases were compared.

In [MBKKP03], the goal was to predict the course score, but now the drop-outs were excluded. The data was collected from the system log and contained attributes concerning each task solved (success rate, success at first try, number of attempts, time spent on the problem, giving up the problem, etc.) and other actions like participating in the communication mechanism and reading support material. Six classifiers (quadratic Bayesian classifier, 1-nearest neighbours, k-nearest neighbours, Parzen window, feed-forward neural network, and decision tree) and their combination were

Table 2.3: Summary of experiments which have learnt predictive models from educational data. Abbreviations for the most common methods are: DT= decision tree, BN = Bayesian network, NB= naive Bayes classifier, FFNN= feed-forward neural network, SVM= support vector machine, k-NN= k-nearest neighbour classifier, LR=logistic regression. In Validation field, the validation method (CV= cross-validation, TS=test set) and the best accuracy (classification rate) are reported. In some experiments, other validation methods and accuracy measures were used.

Reference	Predicted attribute	Other attributes	Data size	Methods	Validation
[KPP03]	Drop-out (binary)	11 Demographic, course activity	350	NB,DT, FFNN, SVM, 3-NN, LR	TS, 83%
[MBKKP03]	Course score (binary)	6 Solving history in previous tasks	227	quadratic Bayes, 1-NN, k-NN, Parzen window, FFNN, DT, combination	CV, 87%
[ZL03]	Course score (binary)	74 Demographic, preferences, habits	300	Boosting weak classifiers	TS, 69%
[BTR04]	Graduating in 6 years (binary)	56 Demographic, academic, attitudinal	>5000	FFNN, SVM	CV, 63%
[MLW$^+$00]	Need for remedial classes (binary)	Demographic, earlier school success	about 600?	Association rules classifier, NB	TS, precision 47%, recall 91%
[BW98]	Time spent to solve a problem (numeric)	25 Concerning problem type, student's abilities	1781	FFNN	CV, SSE 33% smaller than in random guess
[BJSM03]	Words, where the student asks help	20 Concerning word, student, learning history	550 000 words 53 students	DT, NB	TS? 75%
[MM01]	The next errors	Error types in previous answers (categorial)	3300	BN	TS, coefficient of determination $r^2 = 0.75$
[JJM$^+$05]	Mastering a skill (binary)	Previous answers, and whether a hint was shown (binary)	360 (simulated students)	HMM	Model parameters were compared to actual ones
[Vom04]	Mastering a skill (binary)	Previous answers	149	BN	CV, correctly classified skills
[DP05]	Test score (binary)	Answers to test questions (binary)	48+41 (2 courses)	BN variation	CV, compared to LR
[CGB03]	Appropriate material for a student	5 Concerning learning style, material type	400+800 simulated data	Adaptive Bayes	TS 200+400, 89%
[SB03]	Interesting material for a student	Last html pages visited, keywords in each page	30 students, 133 pages	Clustering, a variation of DT	Student feedback
[CF04]	Optimal emotional state for learning	Personality type (categorial)	137	Probabilistic rules, DT	Not reported
[SGMM03]	Error or misconception	Log data?	simulated data, size not reported	FFNN	Not reported

compared. When the score variable had only two values (pass/fail), the accuracy was very good. The combination of classifiers achieved 87% accuracy, and the best individual classifier, k-nearest neighbours, achieved 82% accuracy. The accuracy was further improved by optimizing the combined classifier by genetic algorithms, but now only training error was reported.

In [ZL03], the approach was very similar: to combine several weak classifiers by boosting to predict the final score. However, now the data was gathered by a questionnaire. In addition to demographic data, it contained attributes concerning the student's network situation, surfing habits, teaching material applications, online learning preferences and habits, overall opinion of the course, etc. Each weak classifier used only one of 74 attributes to predict the course score. The combination achieved only 69% accuracy. However, the boosting revealed the most influencing factors for the course success.

In [BTR04], the goal was to predict, whether the student graduates in six years. The data was gathered by a questionnaire, and contained 56 demographic, academic, and attitudinal attributes. Feed-forward neural networks and support vector machines were learnt from the data. Both methods achieved only 63% accuracy, despite of the large data set (over 5000 rows). In addition, feature selection by principal component analysis only decreased the accuracy.

In [MLW$^+$00], the goal was to select students, who would need remedial classes. The selection was made by comparing the predicted average course score in the next semester to some threshold. The data was described only vaguely (demographic data and school success in the previous year). The authors applied their own classification method, described in [LMWY03]. The idea was to search all association rules, which could be used for predicting the final score. In classification, all relevant rules were applied and a score was calculated based on rule confidences and frequencies. The method was compared to naive Bayes and decision tree classifiers by calculating precision and recall measures. The new method outperformed both naive Bayes and decision trees, although the precision was still low (47%). The problem in this research was that some students had actually participated remedial classes, but their success was estimated as they had been in the normal classes.

How the student succeeds in the next task?

The next four experiments concerned how to predict the student's success, errors or help request in the next task given data from the previous tasks. All experiments were based on data collected from a real educational system. The systems had been specialized to one simple task type, which was repeated in different variations. Thus, large data sets could be gathered for training.

In [BW98], neural networks were used to predict the time spent in solving an exercise task. Each task concerned basic arithmetic operations on fractions. The data set contained beliefs about student's ability to solve the task, the student's average score in required sub-skills, the problem type and its difficulty, and the student's gender. First, a group model was learnt from the whole data. In cross-validation, the model achieved 33% smaller *SSE* compared to a random guess. Next, the model was adapted to each individual student by setting appropriate weights. Now 50% smaller *SSE* was achieved, compared to a random guess, but it was not clear, whether these results were validated.

In [BJSM03] a system for teaching pupils to read English was used. The goal was to predict on which word the student asks help. The data set contained student information (gender, approximated reading test results of the day, help request behaviour), word information (length, frequency, location in the given sentence) and the student's reading history of that word (last occurrence, how many times read, average delay before reading, etc.). Only the poor students' data was selected, because they ask help most frequently. First, a group model was learnt from all data and then the model was adapted using individual students' data. Two classifiers, naive Bayes and decision trees were compared. The group models were presumably validated by a separate test set. (The authors tell that they could use only 25% of data for training because of technical problems.) Naive Bayes achieved 75% accuracy and decision trees 71% accuracy. When individual models were compared, only the training errors were reported. Now decision trees performed better, with 81% accuracy. When the information about graphemes (letters or letter groups) in words was taken into account, the accuracy increased to 83%.

In [MM01], a system for learning English punctuation and capitalisation was used. The task was to predict the errors (error types) in the next task, given the errors in the previous tasks. A correct answer should fulfill certain constraints, and each broken constraint corresponded to an error type. The system contained 25 constraints, which occurred in 45 problems. 3300 records log data was collected, when the system was in test use.

A bipartite Bayesian network was used to represent the relationships between constraints in the previous and the next tasks. Given this general form of a graph structure, both the model structure and parameters were learnt from data. This group model was adapted to the individual student, by updating the conditional probabilities after every task. The group model was validated by a separate test set. The coefficient of determination, which described the number of correctly predicted constraints as a function of the number of all relevant constraints, was $r^2 = 0.75$. The individual models could not be validated (there was continual training), but they had a positive effect on the students' scores.

Computerized adaptive testing

Some researchers have considered the problem of estimating the student's knowledge (mastering a topic), given a task solution (correct or incorrect). This is an important problem in *ITS*s, but also in computerized adaptive testing (*CAT*), where we should select the next question according to the student's current knowledge level. The problem in defining probability $P(Topic = mastered|Answer = correct)$ is that we cannot observe mastering directly. Even if the task measures only one skill, the student can guess a correct answer or make a lapse. In practice, the tasks usually measure several inter-related skills. The next experiments introduce data-driven solutions to this problem.

In [JJM+05], the goal was to predict, whether a student answers correctly to the next task, given the previous task solutions (correct or incorrect) and information, whether a hint was shown. It was assumed that each task measures only one skill. For each skill, a hidden Markov model was created. Each Markov model contained a sequence of hidden state variables (mastering a skill), connected to observation variables (answer and hint). All variables were binary. Artificial data of 360 students was generated, assuming some model parameters. The data was used for model construction and the parameters were compared. All conditional probabilities could be estimated correctly with one decimal precision. However, most of the probabilities were just 0 or 1, based on unrealistic assumptions (e.g. a skill could not be mastered, if no hints were shown).

In [Vom04], the same problem was solved by a Bayesian network. The data was gathered from 149 students, who solved questions on arithmetic operations on fractions. The tasks tested 19 skills and seven error types were discovered in students' solutions. The relevant skills and error types for each task were analyzed. Then a Bayesian network which described the dependencies between the skills was learnt from data. Some constraints like edges and hidden nodes were imposed according to the expert knowledge. Finally, the model parameters were learnt from the data for the best model structures. The models were compared by cross-validation. In each model, the states of skills (mastered or non-mastered) were determined after each task. The differences were very small, and nearly all models could predict correctly 90% of skills after nine questions.

In [GC04], the problem was solved in a more traditional way, based on *item response theory* (*IRT*) [Lor80]. The probability that a student with certain knowledge level answers the item correctly was described by an *item characteristic curve*. In [GC04], the item characteristic curve was estimated by a logistic function. The parameters were determined experimentally, but the parameters could be learnt from data, as well.

In [DP05], a new variation of belief networks was developed. It was assumed (unrealistically) that all dependencies were mutually independent, i.e. $P(Y|X_1, ..., X_k) = \prod_i P(Y|X_i)$, where X_is are Y's parent nodes. With this simplification the model structure could also be learnt from data, by determining the binary conditional dependences. The model was compared to a traditional *IRT* system, where the logistic regression parameters were also learnt from data. The evaluation was based on the number of questions needed to classify the examinees as mastering or non-mastering the topic. This was determined by comparing the estimated final score to a given threshold value. Real data from two courses was used to simulate adaptive questioning process. In the first data set, both methods could classify correctly over 90% of examinees after 5-10 questions, but in the second data set the new method performed much better. It could classify over 90% of examinees correctly after 20 items, while the traditional method required nearly 80 items.

Individually recommended material for students

The problem of adapting learning material or guiding the navigation in www material is speculated in several sources (e.g. [Bru00, Sch98]). Most systems use fixed rules, but the ideas of general recommendation systems can be easily applied to educational material. However, the underlying idea of recommending similar material for similar users is not necessarily beneficial in the educational domain. The students should be recommended material, which supports their learning, and not only material which they would select themselves. In the following two experiments the recommendations were learnt from real data.

In [CGB03], the goal was to recommend students the most appropriate material according to their learning style. The material was first preselected according to its difficulty and the student's knowledge level. The student's learning style was defined by a questionnaire, which measured three dimensions (visual/verbal, sensing/conceptual, global/sequential). The material was characterized by two attributes, learning activity (explanation, example, summary, etc.) and resource type (text, picture, animated picture, audio, etc.). A variation of naive Bayes, called *adaptive Bayes*, was used for predicting the most appropriate material, given learning the style and material attributes. The model was otherwise like a naive Bayes, but the conditional probabilities were updated after each new example. The idea was that all actually visited pages were interpreted as appropriate material. The model was compared to traditional naive Bayes and an adaptive Bayes with fading factors, where smaller weights were given to older examples. The test data consisted of two simulated data sets, where students' preferences changed 2-3 times. The best results (89% training error) were achieved with an adaptive Bayes using

fading factors.

In [SB03], the goal was to recommend interesting www pages for students. Now the recommendations were based on keywords that occurred in the pages. First, the last 10 pages visited by the student were clustered according to their keywords. The cluster with the most recent page was selected and a decision tree was learnt to classify new pages into this cluster. The authors developed a new version of the classical *ID3* decision tree algorithm, which created decision trees in three dimensions. In the new *SG-1* algorithm, several decision trees were created, when several attributes achieved the same information gain. The method was tested with real students, but the students followed the recommendations rarely and argued that the system suggested irrelevant links. The problem was further analyzed by clustering all 133 pages according to their keywords. The clustering revealed one large, heterogeneous cluster as a source for unfocused recommendations.

Other work

The last two experiments in the table are very vaguely reported, and not validated.

In [CF04], the goal was to determine the most optimal emotional state for learning, given the student's personality type. 137 students answered a questionnaire, which determined their personality type (extraversion, people with high lie scale, neuroticism, psychoticism), and selected their optimal emotional state from 16 alternatives. The authors reported that they used naive Bayes classifiers for rule construction, but in fact they just used the same parameter smoothing technique which is used in estimating parameters for Bayesian networks. In addition, they reported that they could learn the current emotional state from a colour series by decision trees with 58% accuracy, but no details were given.

In [SGMM03], the goal was to diagnosize the student's errors and misconceptions related to basic physics. The data was collected with a simulation tool, where students were asked to draw gravitation and contact forces, which affect the given objects. The teacher planned a set of simulated students and determined their knowledge levels in different categories. In addition, each student's behaviour was described by some fuzzy values. Then a feed-forward neural network was trained to classify the errors. 87% training accuracy was achieved, but the results were not validated.

In addition, there has been some general discussion, how to apply machine learning methods in educational technology. For example, [LS02] discusses in a very superficial level, how to apply machine learning in intelligent tutoring systems.

2.3.2 Descriptive models searched from educational data

Descriptive data modelling techniques (other than basic statistics) are more seldomly applied to educational data than predictive techniques. One reason may be that data mining is still quite a new discipline. In Table 2.4, we have summarized all experiments, which have searched descriptive models from educational data. Some of the methods could be used for predictive modelling as well, but the intention has been descriptive analysis. For example, decision trees are often used to discover decision rules. These rules are interpreted – and sometimes even called – as association rules. However, this is misleading, because the decision rules can be very rare (i.e. non-frequent) and decision trees reveal only some subset of rules affecting the class variable. The reason is that the decision tree algorithms find only a local optimum and sometimes the attributes are selected randomly. Thus, a full search of real association rules should be preferred.

Analyzing factors which affect academic success

The first three researches in the Table analyze the most influencing factors on academic success or failure (academic average score or drop-out).

In [SK99], the factors which affect average score and drop-out among distance learners were analyzed. The data was collected by a questionnaire. The questions concerned job load, social integration, motivation, study time, planned learning, and face-to-face activities. First, all correlations were analyzed, and a path model was constructed. According to this model, the most influencing factors on average score were study time, planned learning and face-to-face activities. In the second experiment, a logistic regression model for predicting drop-out (enrollment in the next semester) was constructed. Now social integration and face-to-face activities were the most significant factors.

In [SAGCBR$^+$04], a quite similar analysis was performed. The task was to analyze the factors which affect academic average mark, desertion and retention. The data set consisted of nearly 23 000 students, described by 16 attributes, including age, gender, faculty, pre-university test mark, and time length of stay at university. First, the dependences were determined by regression and contingency table analysis. Then the data was clustered, resulting 79 clusters. The clusters with the highest academic average score, the lowest academic average score, and the longest period of stay were further analyzed. Finally, decision rules for predicting good or poor academic achievement, desertion or retention were searched by C4.5 algorithm. The confidences of rules were determined and the strongest rules were interpreted.

Table 2.4: Summary of experiments which have searched descriptive models from educational data.

Reference	Goal	Attributes	Data size	Methods
[SK99]	Factors which affect score and drop-out	20, concerning job load, motivation, study time, activity	1994	Correlation, logistic regression
[SAGCBR+04]	Factors which affect academic success/failure	16, mostly demographic and academic	23000	Clustering, decision trees
[BTR04]	Factors which affect drop-out	6, demographic and attitudinal	100	Correlation, linear regression, discriminant analysis
[KB05]	Navigation patterns in log data	Action type, time-stamp, user, target	1170 sessions	Association rules, episods
[RVBdC03]	Relations between knowledge level in concepts and time spent on pages	Knowledge level, visited pages, time-stamps, success in tasks	50 students	Association rules by genetic algorithms
[MSS02]	Grouping students according to skills	Task points	102	Clustering
[VC03]	Discovering dependencies in questionnaire data	Student's knowledge in 15 topics, test score	436	Classification rules
[SNS00]	Analysis of knowledge-level errors in students' solutions	Erroneous Prolog programs	65+56	Clustering

In [BTR04], only the factors which affect drop-out in distance learning were analyzed. The data set consisted of 100 students, described by six attributes: age, gender, number of completed courses, employment status, source of financial assistance and "locus of control" (belief in personal responsibility for educational success). The data was analyzed by correlation analysis, linear regression and discriminant analysis. The analysis revealed that the locus of control could alone predict 80% of drop-outs and together with financial assistance 84% of drop-outs.

Mining navigation patterns in log data

In adaptive hypermedia systems the main problem is to define recommendation rules for guiding students' navigation in the learning material. One approach is to base these rules on students' actual navigation patterns. The following two experiments have addressed this problem.

In [KB05], the aim was to find recommendation rules for navigation in hypermedia learning material. The data consisted of 1170 sessions of log data. For each action, the action type, time-stamp, user id, and target object (concepts and fragments) were recorded. 90% of data was used for the pattern discovery and 10% for verifying the results. In the first experiment, temporal order was ignored, and association rules between visited concept were discovered. In the second experiment, the temporal order was taken into account, and segmential and traversal patterns were discovered. Segmential patterns (serial episodes) described typical orderings between actions, which were not necessarily adjacent. In traversal patterns it was required that the actions were adjacent. The discovered patterns were used to predict the actually visited concepts in the test data. Sequential patterns achieved the best accuracy, about 51%, but slightly better results were achieved when all pattern types were combined.

In [RVBdC03], a new way to discover interesting association rules in log data was proposed. The goal was to find relations between the student's knowledge level in a concept, time spent to read material associated to that concept, and the student's score in a related task. The data consisted of visited pages and timestamps, the student's current knowledge level and success in tasks. The association rules were discovered by a new method based on genetic algorithms, where an initial set of rules was transformed by genetic operations, using a selected fitness measure. The method was tested in the course data of 50 students, and compared to the traditional apriori algorithm. The apriori algorithm worked faster and found more rules, as expected. However, the authors speculated that their method could find more interesting rules, because the fitness measure and other parameters could be selected according to modelling purposes.

Analyzing students' competence in course topics

Some researchers have made preliminary experiments to analyze students' competence in course topics, based on exercise or questionnaire task points. The problem in such an analysis is that a task can require several skills, but not in equal amounts. This is especially typical for computer science, where majority of knowledge is cumulative and interconnected. Some solution ideas have been discussed in [WP05].

[MSS02] described the first experiment to cluster data gathered in the *ViSCoS* project. The data consisted the weekly exercise points in *Programming* courses, but it was gathered earlier than our current data set. The goal was to cluster the students according to their points in different skills. The exercise tasks were analyzed and weights for 14 concepts were defined by experts. Because the task points were recorded on weekly basis, it was assumed that the students had got equal points in all tasks during one week. For each student, a weighed sum of exercise points in each skill was calculated and normalized to sum 1. Then the students were clustered to five groups by a probabilistic clustering method. The resulting clusters were analyzed by comparing the mean values of attributes and total skills in each cluster. It turned out that the individual skill means diverged only in the cluster of the poorest students (low total skills). In all other clusters no difference was found, but the grouping was simply based on total skills.

In [VC03], the students' questionnaire answers were analyzed by "association rules". In fact, the rules were simple decision rules learnt by C4.5 algorithm, and only the rule confidences were reported. The data set consisted of 436 students, whose knowledge in 15 topics and final scores were given. The topic values were determined according to related question answers. However, the same topic could be covered in several questions, and thus several data rows, one per each possible answer combination, were generated for each student. The final score was simply the sum of questionnaire points, classified as good, average or poor. A decision tree was learnt to predict the student's final score, given her/his skills in topics, and decision rules were extracted from the tree. The training error was about 12%. However, both the topic attributes and final scores were derived from the same attributes, and the dependency was trivial.

Analyzing students' errors in program codes

[SNS00] described a method for identifying knowledge-level errors in students' erroneous Prolog programs. The motivation was that the method could be used to construct an error library and assist in student modelling.

The described method worked in two phases. First, the student's intention (a correct program, which the student has tried to implement) was identified. Then all erroneous programs with the same intention were hierarchically clustered to form an error hierarchy. The assumption was that the main subtrees in such a hierarchy corresponded to knowledge-level errors.

The method was tested in two simple programming tasks. In both tasks about 60 erroneous programs were achieved. The differences between programs were defined by the number of `remove, add, replace` and `switch` operations needed to transform a program to another. Clustering was implemented by two methods, either ignoring the error dependencies or using heuristic rules to model the causal relationships. The resulting cluster hierarchies were analyzed by experts, who had defined the knowledge-level errors in each program. The clustering with causal dependencies achieved clearly better results. In task 1, 84%, and in task 2, 70% of knowledge-level errors in students' programs were detected. In task 1, 95%, and in task 2, 75% of discovered errors corresponded to real knowledge-level errors.

Other work

In addition, we have found a couple of small analyses, which were only vaguely reported. In [CMRE00], nine tutoring sessions with a human tutor were analyzed. The goal was to identify the discussion protocols and their changes. Decision trees were used to find rules which described the protocol changes. In [MLF98], a human expert's problem solving paths were analyzed. A class regression tree was somehow used to identify the action sequences which led to success or failure. In [MBKP04], an idea for discovering interesting association rules from log data was proposed. A small experiment was made, but the results were not reported.

In [DSB04], it was discussed in a very general level, how educational systems could benefit from "data mining" (both predictive and descriptive methods). In [HBP00], general ideas of applying web mining in distance learning systems were discussed.

Chapter 3

Modelling

The main problem in modelling educational data is to discover or learn robust and accurate models from small data sets. The most important factor which affects the model accuracy is the model complexity. Too complex models do not generalize to other data sets, while too simple models cannot model the essential features in the data. Model complexity has an important role in robustness, too, but there are several other factors which affect robustness. Each modelling paradigm and model construction method has its own *inductive bias* – a set of conditions, which guarantee a robust model. These conditions should be checked when we select the modelling paradigm, model structure, and model parameters.

In the following, we will first define the basic modelling terminology. Then we analyze the main factors – model complexity, inductive bias and robustness – which affect the model selection. We formulate principles for selecting an appropriate modelling paradigm and a model in the educational domain. Finally, we summarize the main techniques for model validation.

3.1 Basic terminology

The main problem of modelling is to find a model, which describes the relation between target variable Y and a set of other variables, so called *explanatory variables*, X. In all predictive tasks and most of the descriptive tasks, the target variable values $t[Y] = y$ are given in the data. In machine learning, such tasks are called *supervised learning*, because the true target values "supervise" the model search. In *unsupervised learning*, the target values are unknown, and the goal is to learn values which optimize some criterion. For example, in clustering we should divide

all data points into clusters such that the clustering is optimal according to some criterion. In the following, we will concentrate on supervised modelling. We will return to clustering in Chapter 6

In data-driven modelling the models are learnt from a finite set of samples called a *training set*. Because the training set has only a limited size, we seldom find the true model, which would describe all possible data points defined in the relational schema. Thus, the learnt model is only an approximation.

Definition 1 (True and approximated model) *Let R be a set of attributes and r according to R a relation. Let $X = \{X_1, ..., X_k\} \subseteq R$ be a set of attributes in R and $Y \in R$ the target attribute. Then the* true model *is a function $F : Dom(X_1) \times ... \times Dom(X_k) \rightarrow Dom(Y)$ such that $t[Y] = F(t[X])$ for any tuple t defined in R. The approximated model is a function approximation $M : Dom(X_1) \times ... \times Dom(X_k) \rightarrow Dom(Y)$ such that $t[Y] \approx M(t[X])$ for all tuples $t \in r$.*

In machine learning literature (e.g. [Mit97][7]), true models are often called as *target functions*, and approximated models as *hypotheses*. In the following, we will call the approximated models simply models.

In descriptive modelling, the models are often only partial, and we cannot define them by ordinary functions. However, we can define partial models by restrictions of functions:

Definition 2 (Partial model) *Let R, $r \in R$, $X \subseteq R$ and $Y \in R$ be as before, and $r' \subsetneq r$. Model M is* partial*, if $M|_{r'} : Dom(X_1) \times ... \times Dom(X_k) \rightarrow Dom(Y)$ such that $t[Y] \approx M(t[X])$ for all tuples $t \in r'$.*

The accuracy of the model is defined by its true error. The same error measure is used to measure how well the model fits the training set. Some examples of common error measures will be discussed in Section 3.7. The following definition is derived from [Mit97][130-131]:

Definition 3 (Training error and true error) *Let $d(M(X), Y)$ be an error between the predicted value $M(X)$ and the real value of Y. Then the* training error *of model M in relation r, $|r| = n$, is*

$$error(M, r) = \frac{\sum_{t \in r} d(M(t[X]), t[Y])}{n}$$

and the true error *of model M is*

$$error_{true}(M) = \lim_{|r|\to\infty} error(M, r).$$

If the true error is small, the model is called *accurate*. The relative accuracy measures how well the model generalizes to new data.

If the error measure d is

$$d(M(X), Y) = \begin{cases} 1, & \text{when } M(X) \neq Y \\ 0, & \text{otherwise} \end{cases}$$

then the true error is the probability to make an error in prediction. Often we express the accuracy of the model as probability $1 - p$, where p is the error probability. Notice that the true error is unobservable, although we can try to estimate it. We will return to this topic in Section 3.7.

In model selection, the main goal is to minimize the true error. Because the true error is unobservable, a common choice is to use some heuristic *score function*. Typically the score function is based on training error and some additional terms, which guide the search. An important criterion is that the score function is robust, i.e. insensitive to small changes in the data.

Given the score function, we can define the optimal model:

Definition 4 (Optimal model) *Let M be a model, r a relation, and $score(M, r)$ a function, which evaluates the goodness of model M in relation r. We say that the model is* optimal*, if for all other models $M' \neq M$, $score(M', r) \leq score(M, r)$.*

In many problems searching a globally optimal model is intractable, and the learning algorithms produce only a locally optimal model.

3.2 Model complexity

The desired model complexity is an important choice in model construction. On one hand, we want to get as expressive and well-fitting model as possible, but on the other hand, the more specialized the model is, the more training data is needed to learn it accurately.

A special problem in educational domain is the small size of data. The smaller the training set is, the more the training error can deviate from the true error. If we select the model with the smallest training error, it probably *overfits* the data. The following definition is derived from [Mit97][67]:

Definition 5 (Overfitting) *Let M be a model and $r \in R$ a relation as before. We say that M* overfits *data, if there exists another model M' such that $error(M, r) < error(M', r)$ and $error_{true}(M, r) > error_{true}(M', r)$.*

In overfitting the model adapts to the training data so well that it models even noise in the data and does not generalize to any other data sets. Overfitting can happen even with noise-free data, if it does not represent the whole population. For example, if a prerequisite test is voluntary, only the most active students tend to do it, and we cannot generalize the results to other students.

The best way to avoid overfitting is to use a lot of data. A small training set is more probably biassed, i.e. it does not represent the real distribution. For example, the attributes can have strong correlations in a sample, even if they are actually uncorrelated. When large amounts of data are not available, the best we can do is to select simple models. In practice, the attribute set can be reduced by selecting and combining original attributes. This is also an important option when selecting the modelling paradigm. Descriptive modelling paradigms impose usually very simple patterns, but in predictive modelling, the paradigm may require too complex models. For example, decision trees require much more data to work than naive Bayes classifier. We will return to this topic in Chapter 7.

The other extreme, concerning model complexity, is to select too simple models. A simple model generalizes well, but it can be so general that it cannot catch any essential patterns in the data. We say that the model has *weak representational power*. Sometimes this phenomenon is called *underfitting*. It means that the model approximates poorly the true model or there does not exist any true model. In predictive modelling such a model causes poor prediction accuracy. In descriptive modelling it describes only trivial patterns, which we would know even without modelling. In descriptive modelling, this happens easily, because we perform an exhaustive search of all possible patterns. If we try enough many simple patters, at least some of them will fit the data, even if there are actually no patterns at all.

As a compromise, we should select models which are robust and have sufficient representational power. Such models have the smallest true error and they are accurate. In practice, we should select simple models, which still explain the data well. Very often, this happens through trial and error, in an iterative process of descriptive and predictive modelling.

3.3 Inductive bias

Inductive bias or *inductive principles* (see e.g. [Mit97, EC02]) mean a set of assumptions under which the model generalizes to new data. These assumptions concern the properties of training data (e.g. the size and the distribution of the training set), properties of the desired model (e.g. the model family or the model class, where it belongs) and successful learning (e.g. the learning style, score functions). Inductive bias occurs in all phases of the modelling process. The most general bias is expressed in the modelling paradigm, while others concern the learning style, method or algorithm.

Some of the biasses are explicitly expressed in the method and the user can manipulate them. For example, in probabilistic clustering, the user can decide the form of the distribution. However, most of the biasses are implicit and unchangeable, and the user should just be aware of them.

Mitchell [Mit97][64] divides the inductive bias into two types:

1. In *restriction bias* we restrict the search space (model family). This kind of bias is also called *representational* or *syntactic*, because it restricts what kind of models we can represent. The restriction can be inherent already in the modelling paradigm. For example, linear regression model supposes that the target function is linear, and rules out all other functions. Inside a modelling paradigm, we can restrict the search space by selecting the number of attributes (model size). The bias can also concern the learning style. The stricter the restrictions, the more efficiently we can learn the model – if the reduced search space contains the true model any more.

2. In *preference bias* we do not rule out any models, but instead put preferences on them. For example, we can prefer more special models to general ones, or favour the simplest models like in *Occam's Razor Principle*. Preference biasses are frequently used in heuristic optimization algorithms to navigate in the search space and find good models efficiently.

The restriction bias concerns the selection of the model structure by fixing the model family or the model class. On the other hand, preference bias concerns the selection of the model structure and the parameters by suggesting the preferrable model classes and guiding the search in them. Preference bias is usually more recommendable, but sometimes we cannot avoid restriction bias. The effects of bias are very domain-specific, and the same bias can be either beneficial or undesirable, depending on the domain. In fact, prior knowledge about the domain can work

as an efficient bias. This kind of *semantic bias* can be expressed in the form of either restriction or preference bias. This idea is occupied in our general principle of descriptive and predictive modelling, where we first learn domain knowledge through descriptive modelling.

Both of the previous bias types concern only model selection. In addition, we can separate a third kind of bias which concerns the properties of data. We call these bias as *data bias*. Data bias is usually expressed in the learning algorithm. The learning algorithm is guaranteed to work correctly, only if these assumptions are met. An example of such bias is an assumption that the training set represents the whole population and thus the distribution is also the same. If the future examples have very different distribution, the method does not work. A more restrictive data bias is the assumption of normal distribution, which in practice holds very seldom. If the training set is very large, the error is small, but for small data sets the error can be significant. In clustering, there is a vast number of different methods with different biasses. These will be discussed later in Chapter 6.

In Table 3.1, we list examples of inductive biasses. We have classified the biasses as restrictive bias, preference bias or data bias. Some of the example biasses are consequences of other, more general assumptions. For example, if we assume that the training error of the model is zero, it also implies that the data set does not contain outliers and the data is consistent.

3.4 Robustness

Robustness is a very important property of a model. Intuitively, it means that the model is stable and small variations in data do not have dramatic effects on results. For example, a robust model tolerates better outliers (exceptional data points), noise (small errors in attribute values) or missing attribute values in the training data. An unrobust model is sensitive to such variations, and does not generalize to future data as well.

Definition 6 (Robust model) *Let $r_1 \in R$ and $r_2 \in R$ be two training sets and M_1 and M_2 be two models learnt from r_1 and r_2 by the same learning algorithm. Let d_1 measure the difference between data sets and d_2 the difference between true errors. We say that model M_1 (M_2) is robust, if for all $\epsilon_1 > 0$, there is a small $\epsilon_2 > 0$ such that*

$$\textit{If } d_1(r_1, r_2) < \epsilon_1 \textit{ then } d_2(error_{true}(M_1), error_{true}(M_2)) < \epsilon_2.$$

Table 3.1: Examples of inductive bias. For each assumption, we give the type of the bias (R=restrictive, P=preference, D=data bias) and examples, where it is used.

Assumption	**Type**	**Examples**
Consistent learning: The true model is fully representable in the selected model family, and the model has zero training error.	R	Learning algorithms
Occam Razor principle: Simpler models are preferred to more complex ones	P	Learning algorithms
Greedy heuristic: Selecting always the locally optimal model leads to global otpimum.	P	Learning algorithms
Linear dependency: The dependency can be described by a linear function	R	Linear regression
Conditional independence: the explanatory variables are conditionally independent from each other.	R	Naive Bayes
Linear decision boundary: the decision boundary between class values is linear.	R	Perceptron, naive Bayes
Normality assumption: The data is normally distributed	D	Learning algorithms
Valid sampling: The training set represent the whole population and the distribution is the same.	D	Learning algorithms
Data consistence: there are no such rows $t_1, t_2 \in r$ that $t_1[X] = t_2[X]$ but $t_1[Y] \neq t_2[Y]$.	D	Decision trees
Occurrence of outliers: There are no serious outliers in data	D	Linear regression, clustering with SSE score function

The robustness of the model depends on the relative size of ϵ_2 compared to ϵ_1. Notice that the robustness depends on the learning algorithm, and ultimately, the modelling paradigm, which have their own inductive biasses. Each learning algorithm produces robust models, if the underlying assumptions hold. In practice, some learning algorithms are more tolerable to the violations of their inductive bias than others. Such algorithm are often called robust (e.g. [STC04][13]). Similarly, modelling paradigms, which make less restrictive assumptions or which tolerate better the violation of these assumptions are robust.

Some modelling paradigms and learning algorithms have very restrictive semantic bias. I.e. they require precise domain knowledge to work correctly. For example, in the back-propagation algorithm for learning feed-forward neural networks we should know the ideal number of hidden nodes, type of activation function and stopping criterion. Often these parameters are not known, and the resulting model can be very unstable.

Robustness depends on also the size of data set and the model complexity. Thus, it is also related to the overfitting problem: if the model is very sensitive to data, it overfits easily and describes even errors in the data set. On the other hand, a robust model can tolerate errors and does not overfit so easily. As a result, it generalizes better to new data.

3.5 Selecting the modelling paradigm

Selecting the modelling paradigm has a critical role in data modelling. An unsuitable modelling paradigm can produce false, unstable or trivial models, even if we use the best available learning algorithms. Often, we have to try several modelling paradigms, and learn several models, before we find the optimal model for the given data. The number of alternative modelling paradigms can be pruned by analyzing the following factors:

- Properties of data.
- The inductive bias in the modelling paradigm.
- The desired robustness and the representational power of the model.

The problem is to find such a modelling paradigm that the properties of the data match the inductive bias of the paradigm and the resulting models are accurate. We recall that in accuracy, we often have to make a compromise between the robustness and the representational power.

3.5.1 Data properties

The first question is the type and the size of the given data. Some modelling paradigms require only numeric or only categorial data, while others can combine both. Numeric data can always be discretized to categorial, but categorial data cannot be transformed to numeric. The only exception is the transformation of categorial attributes to binary values. However, this transformation increases the number of attributes significantly and the model becomes easily too complex.

In educational domain, the most critical factor of data is the size of the data relative to the model complexity. As a rule of thumb, it is often suggested that we should have at least 5-10 rows of data per each model parameter. The number of model parameters depends on the number of attributes and their domain sizes. Often, we can reduce the number of attributes and/or their domain sizes in the data preprocessing phase. We will return to this topic in Chapter 4. In addition, we should select a modelling paradigm, which produces simple models. In practice, we recommend to take the simplest modelling paradigms like linear regression or naive Bayes as a starting point, and analyze whether they can represent the essential features in the data. Only if the data requires higher representational power (e.g. non-linear dependencies, or non-linear class boundaries), more powerful modelling paradigms should be considered.

The representational power is part of data bias in the modelling paradigm. The other assumptions in data bias should be checked as well. For example, our analysis suggests that the educational data is seldom normally distributed. One reason is the large number of outliers – exceptional and unpredictable students. If we want to use a paradigm, which assumes normality, we should first check how large is the deviation from normality and how sensitive the paradigm is to the violation of this assumption.

When the goal is predictive modelling, we recommend to analyze the data properties first by descriptive modelling. According to our view (Figure 3.1), descriptive and predictive tasks are complementary phases of the same modelling process. This view is especially useful in adaptive learning environments, in which the model can be developed through several courses. The existing data is analyzed in the descriptive phase and a desirable modelling paradigm and model family are defined. An initial model is learnt, and applied to new data in the prediction phase. In the same time we can gather new features from users, because the descriptive modelling often reveals also what data we are missing. After the course, the new data is analyzed, and the old model is updated or a new better model is constructed. As a result, our domain knowledge increases and the predictions improve in each cycle.

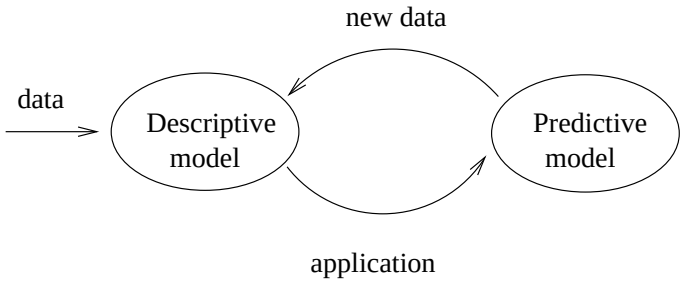

Figure 3.1: Iterative process of descriptive and predictive modelling. Descriptive modelling reveals the underlying patterns in the data and guides the selection of the most appropriate modelling paradigm and model family for the predictive modelling. When the predictive model is applied in practice, new data is gathered for new descriptive models.

3.5.2 Inductive bias in the modelling paradigm

Data bias is only a part of inductive bias in the modelling paradigm. Each modelling paradigm contains several assumptions, under which it works optimally. Unfortunately, these assumptions are often implicit, and thus difficult to check. In this thesis, we have analyzed the inductive bias in the main modelling paradigms and evaluated their general validity in the educational domain. The summaries of this analysis are presented in Sections 5.7.3, 7.7 and 6.6.

Usually the learning style is defined in the paradigm level. Different learning styles have their own bias. In the coarse level, the learning styles can be divided into *eager* and *lazy* (*instance-based*) learning (see e.g. [Mit97][244-245]). An eager learner constructs one global model during the training and the model cannot be changed afterwards. In pedagogical terms, an eager learner is like a student, who already considers how to apply the theory to new problems, before it is actually asked. On the other hand, a lazy learner tries to postpone learning as long as possible. The learning algorithm stores all training examples, and approximates the function value for a new instance locally, according to previous examples. Thus, a lazy leaner is like a student, who collects all given material, but does not try to construct any general principles. Given a new problem, s/he tries to apply the most similar old examples and interpolates the answer. The most common variation of lazy learning is *k-nearest neighbours* method. For every new data point the method calculates mean, median or some other function of k nearest neighbours' attribute values.

In eager learning, the inductive bias is the assumption that a global model can be learnt from the training example. In lazy learning, the inductive assumption is that

the new point is similar to the closest previous examples. Eager learner takes more time in the learning phase, but the application is very fast. In contrast, in lazy learning the training is fast, but application is more worksome. In addition, lazy learning requires a lot of training data to work. In educational applications, eager learning is preferable, because we do not have enough data. In addition, it is not critical if the learning phase takes time, but the actual predictions should me made in real time.

3.6 Selecting the model

Model selection consists of three steps: defining the model family, selecting the model class and selecting the model parameters. In some algorithms, the model class and model parameters are learnt in the same step.

The model family is usually defined by the human expert. By selecting the model family, the expert defines what can be expressed in the model. In the largest model families, only the available variables are defined, and all relations are learnt from data. This is always recommendable, when we search simple patterns like association rules. When we search more complex models, the resulting model family is often too large for exhaustive search, and the learning algorithm produces only a locally optimal model. More accurate models can be learnt, if we can restrict the search space (model family) by a semantic bias. In practice, the expert can define some restrictions or preferences on the possible model structures. The best way to decide the restrictions is to perform first descriptive analysis (e.g. dependency modelling) and then select the meaningful restrictions.

The model class and model parameters are usually learnt from data, although in educational applications, the model class is often predefined. This is justified, if the structure is so complex that it cannot be learnt accurately from the data. However, there is always a danger that the predefined model structure does not fit the data, because the expert's domain knowledge is incorrect or insufficient. For example, it is often hard to decide which exercise tasks are dependent. If strong dependencies are missing from a Bayesian network model, it cannot represent the real joint probability distribution. On the other hand, if the model contains several unnecessary dependencies, the model becomes too complex, and the parameters cannot be learnt accurately.

Once again, a better alternative is to define the model structure according to descriptive analysis. Descriptive models can already define the whole model structure for predictive models. For example, if the goal is to learn a naive Bayes model,

it is enough to discover all important dependencies between the class variable and explanatory variables and check that the explanatory variables are conditionally independent, given the class variable. If we discover strong dependencies between the explanatory variables, we can add them to the model.

The expert can also define preference bias in the form of a score function, which is used to evaluate the goodness of alternative models. For example, the preference for simple models can be expressed by *minimum description length* (*MDL*) score function, which favours models which can be described by shortest code. However, it should be noticed that the description length depends on the representation, and *MDL* does not give an absolute measure for simplicity. In addition, several heuristic overfitting avoidance methods have been developed, with varying success. In some domains they can achieve great improvements, while in others they can lead to even poorer performance [DHS00, Sch93].

3.7 Model validation

The aim of model validation is to give insurance that we have found a good model, or at least that we do not accept a poor model. The same techniques can also be used for comparing alternative models and selecting the best one. However, it is good to remember that the methods are just insurance policies, and they do not eliminate chance and variability in data. In the following, we will briefly introduce the most common validation techniques for descriptive and predictive modelling. For further reading we recommend e.g. [MA03, Kar01, Mit97, HMS02].

3.7.1 Statistical tests

In descriptive modelling we want to verify that the discovered patterns are meaningful and not only due to chance. This is a real danger especially with small data sets, because often we perform an exhaustive search for quite simple patterns. It is quite probable that some of the discoveries are spurious and could have occurred in totally random data, as well. It is always good to follow Smyth's [Smy01] instruction and imagine how the method would have worked with totally random data. This is exactly the goal of statistical significance tests.

The statistical *significance tests* follow the general schema of proof by antithesis. First we formulate a hypothesis H which describes our discovery and a *null hypothesis* $H_0 = \neg H$. In the null hypothesis it is assumed that the discovery is insignificant. Then we evaluate some test measure X from the data and calculate the probability

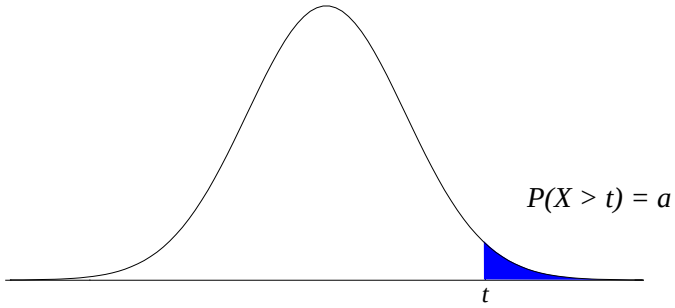

Figure 3.2: The idea of the significance test. The distributional form depends on the given test measure X. Value $P(X > t) = a$ is the significance level, typically 0.01, and t is the corresponding critical value, given the test distribution. If the value of the test measure, $X = x$, lies on the shaded area (i.e. $x > t$), the observation is statistically significant at the given level a.

$P(X \geq x) = p$ that the observed value $X = x$ could occur by chance[1]. If this probability is very small, we can reject the null hypotheses H_0 and accept the hypothesis H at the *level of significance* p. The idea of significance test is illustrated in Figure 3.2.

Typical levels of significance p are:

0.05	nearly significant
0.01	significant
0.001	very significant

The corresponding test measure values $t_{0.05}$, $t_{0.01}$, $t_{0.001}$ are called *critical levels*. For common distributions, the critical values are given in the statistical tables and it is enough to compare the test measure to the critical values of the given test distribution. The distribution parameter(s) k (or k_1, k_2) is called the *number of the degrees of freedom* and the corresponding critical value is notated by $t_p(n)$ (or $t_p(k_1, k_2)$).

Example 1 *Let us consider a data set of 100 students, who have participated* Discrete mathematics *course. Attributes gender (female, male)* $G \in \{F, M\}$ *and final results (failed, passed)* $FR \in \{0, 1\}$ *have uniform distribution:* $P(G = F) = P(G =$

[1]This so called *right-tail test* is used, when the test measure is larger than expected. If the measure is smaller than expected, the *left-tail test* $P(X < x)$ is used instead. In the *two-tailed test* we calculate probability $P(X < x_1 \vee X > x_2)$.

$M) = P(FR = 1) = P(FR = 0) = 0.5$. We observe that 34 male students (68% of all male students) have failed the course. Is this discovery significant?

Now the null hypothesis is that the attributes G and FR are independent. The test measure X is the frequency of failed male students. The data follows binomial distribution with parameter $p = 0.5$. The probability is $P(X \geq 34) = 0.028$, which is statistically not significant at significance level $p = 0.01$. However, if 35 male student had failed the course, the discovery would be significant $P(X \geq 35) = 0.01$.

If we test only a few patterns, the statistical significance tests can be used as such. However, in knowledge discovery, we often search all possible patterns, and the probability to find spurious patterns increases. For example, a correlation efficient which is significant at level 0.05, occurs once in every 20 efficients by chance. If we test 100 correlation coefficients, we are likely to find five spurious correlations. This has been historically one of the main reasons to criticize data mining. As Sullivan et al. [STW99] state it: "*Data snooping occurs when a given set of data is used more than once for the purposes of inference or model selection. When such data reuse occurs, there is always the possibility that any satisfactory results may simply be due to chance rather than to any merit inherent in the method yielding the results*".

As a solution, we should use the stricter bounds for significance, the more patterns we test. Some authors (e.g. [Gar05]) suggest to use significance level p/k, where k is the number of patterns to be tested. For example, if we test 20 patterns and want them to be significant at 0.01 level, then we should test them at 0.005 level. However, this rule is so strict that there is a risk that we do not recognize all significant patterns.

3.7.2 Testing prediction accuracy

In predictive modelling we have slightly different aims. We would like to ensure that the model has not overfitted and generalizes well. Small training error does not give any guarantees about good prediction accuracy in the future, because we can often achieve zero training error, if we just select sufficiently complex model. Instead, we should try to estimate the generalization error on unseen data. As a solution, different kind of testing schemas are used.

In the ideal case, we can reserve part of data as a validation or test set and use the rest for training. Now the validation set gives good outlines how well the model works with unseen data. However, if the original data set is very small, this only increases the risk of overfitting. In this case, *k-fold cross-validation* is a better solution. The idea is that we partition the original data set to k disjoint subsets of

size n/k. Then we reserve one subset for validation and learn the model with other $k-1$ subsets. The procedure is repeated k times with different validation sets and finally we calculate the mean of prediction errors. The same method can be applied for comparing different models.

The most popular error functions for measuring prediction error are the *sum of squared errors* (*SSE*) and *classification rates*. *SSE* is defined as

$$SSE = \sum_{i=1}^{n} (y_i - f(x_i))^2,$$

in which y_i is the real value and $f(x_i)$ the predicted value of data point x_i ($i = 1, ..., n$). If we want to take into account the data size n, *mean squared error* (*MSE*) is used instead:

$$MSE = \frac{SSE}{n}.$$

SSE can be used to measure prediction error of any numeric-valued target function f. In addition, it is often used as a score function in learning phase.

In classification, where the predicted values are typically categorial, *classification rates* are often used. Let us first consider the case of two classes, positive class c_1 and negative class c_2:

true positive $m(f(x) \in c_1 \wedge y \in c_1)$	false negative $m(f(x) \in c_2 \wedge y \in c_1)$	$\Sigma =$ $m(y \in c_1)$
false positive $m(f(x) \in c_1 \wedge y \in c_2)$	true negative $m(f(x) \in c_2 \wedge y \in c_2)$	$\Sigma =$ $m(y \in c_2)$
$\Sigma = m(f(x)) \in c_1)$	$\Sigma = m(f(x) \in c_2)$	$\Sigma = n$

Now the rate

$$\frac{\text{true positive} + \text{true negative}}{n}$$

tells the classification accuracy (proportion of correctly classified examples), and

$$\frac{\text{false positive} + \text{false negative}}{n}$$

tells the classification error. When we have more classes, we simply calculate true positive and false negative rates for all classes.

3.7.3 Testing in practice

The final test for the model is how well it works in practice. However, testing models with real students is problematic. Blind tests are not recommendable, because the system should be transparent – i.e. the student should know, how the system has classified her/him. This leads to a counterpart of *Heisenberg uncertainty principle* in education technology: the measuring itself affects the measurements. The better we can predict the student's current state, the more it affects her/his proceeding, and the outcomes are unpredictable. Thus, a good system can perform the worst in the real learning context! The situation does not differ much, if the influence is indirect, and only the course tutors know the predictions. However, the ultimate measure should be, how well the system improves learning – not how well it succeeds in prediction. In this sense a system which reduces drop-outs performs well, and a system which increases drop-out works poorly.

Chapter 4

Data

Data is the main subject of knowledge discovery. The selection of attributes, their types and domains have a strong influence on the model accuracy. In this thesis, we demonstrate that with careful data preprocessing we can model even small educational data sets accurately.

In the following, we will define the main concepts concerning data, analyze typical characteristics of educational data, and propose appropriate preprocessing techniques. First, we will describe the *ViSCoS* data, which is used as an example in the whole Chapter.

4.1 Motivation: *ViSCoS* data

In the *ViSCoS*-project, the first problem was to gather data from several sources and determine the best attributes for our modelling purposes, especially for detecting drop-outs and potential failures.

The available data sets contained information from two programming courses, *Programming 1* (*Prog.1*) and *Programming 2* (*Prog.2*) from academic years 2002-2003 and 2003-2004. In the academic year 2002-2003, 56 students had participated to *Prog.1* course, and 46 students to *Prog.2* course. In the academic year 2003-2004, the corresponding numbers were 69 and 42. The course content (learning material in Internet) had stayed the same during both academic years, but some exercise tasks had been changed and some had been given in a different order.

Prog.1 data sets contain students' exercise task points for 9 weeks and *Prog.2* data sets for 10 weeks. Only the sum of weekly exercise points have been recorded, and

the individual task points are not available. The exercise task points gave students 0..4 bonus points, which were added to the total points.

In both courses, the students had to participate a final exam. Most of the exam tasks were different in the consecutive academic years, but the topics and difficulty were approximately the same. Only the sum of exam points (max 30) and sum of total points (max 34) were recorded, but the individual task points were available in students' solution papers. The final grades for accepted students were recorded into a separate file. The final grades were expressed by values $\{1-, 1, 1+, 2-, 2, 2+, 3-, 3+\}$, where 1- is the lowest and 3 the highest grade. The student was accepted, if the total points were at least 15.

The students had agreed that their *Programming* course data can be used anonymously for research purposes. The researchers did not have access to students' personal data. The students' names and students numbers were removed in the beginning of the project, and replaced by anonymous student identifier. Before that, the student's gender was derived from the first name. Demographic data like age, school, and home town, as well as course data in the other courses could be valuable information, but we should get students' agreement before they could be used.

4.2 Basic concepts

In the following, we define the main concepts which characterize data.

4.2.1 Structured and unstructured data

The main division of data is to *structured* and *unstructured data* (see e.g. [Kan02][10-11]). In structured data, all data items can be represented as a set of attribute–value pairs, like *name="Kitty Catfish"*, *age=23*, *can_program=true*. Unstructured data like text, sounds and images do not have any attributes, but a data item is just one atomic entity, a string of characters.

In knowledge discovery, it is assumed that all data is structured. The complexity of structures depends on the representation. In this thesis, we assume that all data is represented in the relational schema, and the data structure consists of attribute-value pairs. In other data representations, the structures can be more complex (e.g. attribute hierarchies or aggregates of attributes).

In educational domain, the data is often structured, but sometimes we cannot avoid unstructured data. For example, if we want to analyze students' program codes or

learning diaries, we should first *extract* all interesting features. Feature extraction means a transformation, where raw data set D is transformed to a relation $r \in R$:

Definition 7 (Feature extraction) *Let $R = \{A_1, ..., A_k\}$ be a set of attributes, D a data set, and r a relation according to R.* Feature extraction *is a mapping $f_e : D \rightarrow r$, which maps each data item $d \in D$ into a tuple $t \in r$:*

$$t = f_e(d) = \{(A_1 = a_i), ..., (A_k = a_k)\},\ a_i \in Dom(A_i).$$

Feature extraction techniques for unstructured data go out of the scope of this study. Generally, the same techniques can be applied in educational domain than in other domains. For example, in *pattern recognition* (see e.g. [DHS00]) several sophisticated methods have been developed for extracting features from low-level unstructured data like images, sounds, and video. On the other hand, extracting features from text documents falls under scope of *information retrieval* (see e.g. [BYRN99]). The roles of pattern recognition and information retrieval for knowledge discovery are illustrated in Figure 4.1 [1]

4.2.2 Basic data types

Data types classify the attributes according to their domains. The basic division is to *numerical* (*quantitative*) and *categorial* (*qualitative*) data types (see e.g. [Kan02, JD88]). Numerical data has meaning as numbers and we can measure order, distance and equality of two numerical variables. In contrast, categorial data has no meaning as numbers, and generally we can measure only equality between two categorial variables. The general classification is presented in Figure 4.2.

Numeric data can be further classified as *discrete* or *continuous*. Discrete data can get only countably many values, and if the domain is pictured on the number line, it consists only of isolated points. Discrete numbers are not necessarily integers, but also decimal numbers with a predefined precision. For example, in the *ViSCoS* data, the total course points could get any values in the set

[1] We note that our view is simplified and emphasizes the role of knowledge discovery. Some researchers interpret pattern recognition as a total process, which includes knowledge discovery. On the other hand, pattern recognition methods can be used in information retrieval, if the goal is to retrieve multimedia documents. Some researchers include information retrieval into knowledge discovery.

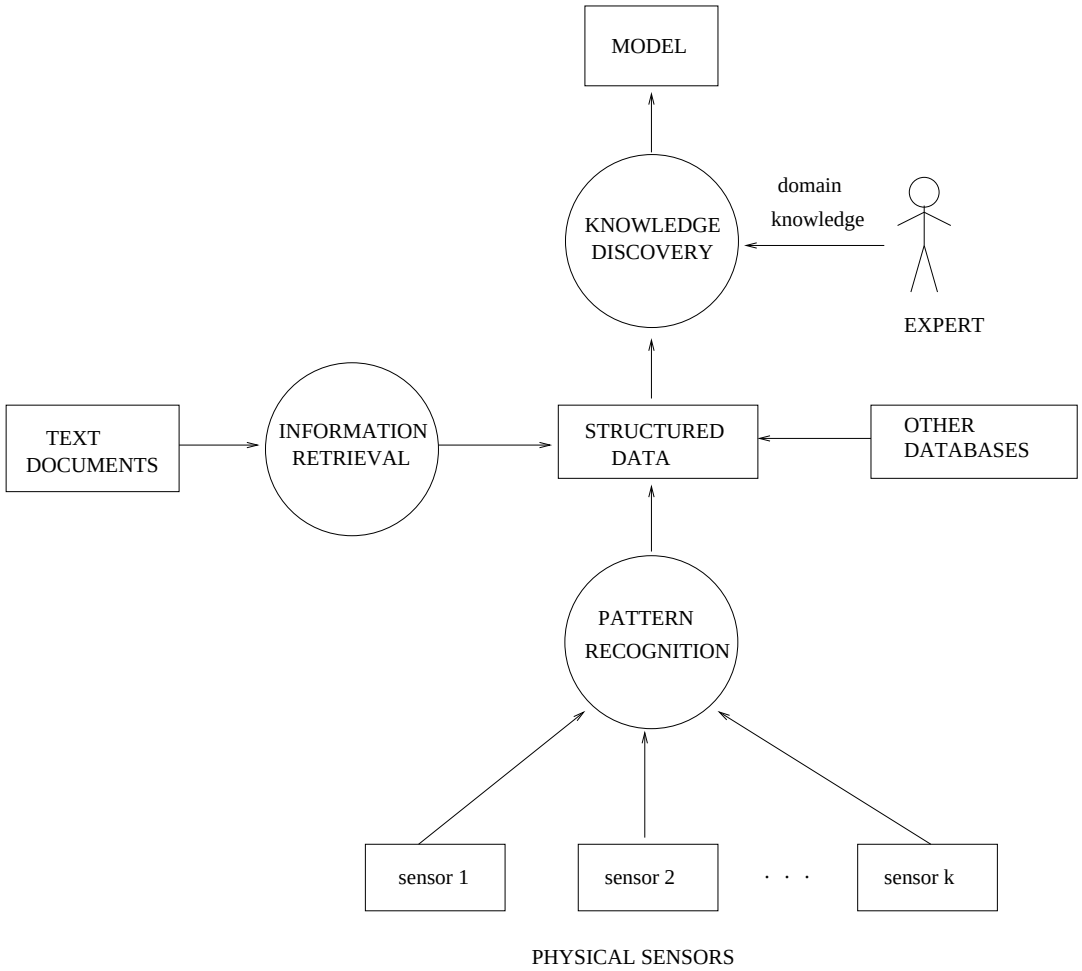

Figure 4.1: The roles of pattern recognition and information retrieval for knowledge discovery process.

$\{0.00, 0.25, 0.50, ..., 33.50, 33.75, 34.00\}$. All exercise task points were also given in 0.25 point precision. On the other hand, continuous data can in principle get any value in the given interval [2]

The categorial variables can be further classified as *nominal* or *ordinal*. The nominal values are only names for categories and have no other meaning than categorizing. If numerical symbols are used, they can always be replaced by other symbols (e.g. *gender: 1=female, 2=male*). The ordinal values have a natural ordering, but the differences have no meaning. For example, in *ViSCoS* data, the grades $\{1-, ..., 3\}$

[2]Note that in computer system the continuous values cannot be represented with exact precision, but they are always discretized to some precision.

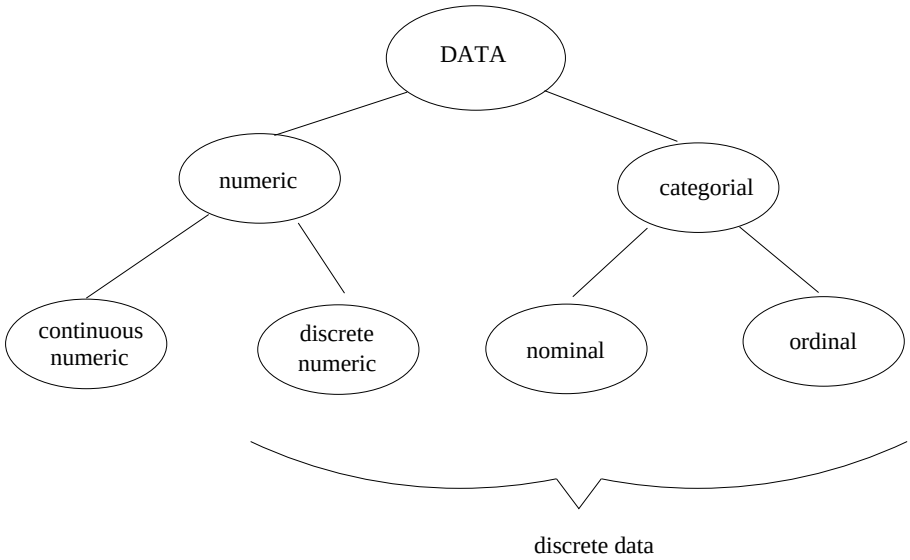

Figure 4.2: Classification of data types.

have an order, but the meaning of difference between two grades depends on the interpretation. Sometimes the difference between 3− and 2+ is interpreted as 0.5, sometimes as 0.33. Interpreting the difference between values *excellent* and *good* is even harder.

It should be remarked that categorial data is also discrete in the sense that categorial variables can get only countably many values. When we are analyzing categorial data, we will typically work with counts or percentages of objects or individuals which fall within certain categories.

According to our analysis, most of educational data is discrete. Numeric attributes measure typically points or scores. Physical devices, which are typical sources for continuous data, can gather data for educational systems, but handling such low-level data does not differ from other domains. The same techniques for extracting discrete features from physical measurements can be used in learning environments as well.

The most important characteristics of data types are the relations which hold for them. In Table 4.1, based on [Kan02][19-20], we have summarized whether the most common relations – order, distance and equality – hold for a given data type. These properties have a strong influence on what kind of modelling paradigms and score functions we can use. For example, in clustering we have to define a metric, which measures the distance or similarity between data points. For numeric data this is trivial, but categorial data is more problematic. This topic is further discussed in Chapter 6.

Table 4.1: Summary of basic data types and relations which hold for them. Signs + and − indicate whether the relation holds for the given data type or not.

Data type	Order	Distance	Equality
Numerical	+	+	+
discrete	+	+	+
continuous	+	+	+
Categorial	+/−	−	+
nominal	−	−	+
ordinal	+	−	+

The numeric variables can be further described according to their measuring scale. If the data is measured in the *interval scale*, there is no absolute zero point and the ratio relation does not hold. For example, if we measure temperature in Celsius temperature scale, +30 ℃ does not indicate twice as warm as +15 ℃. The ratio relation holds only, if the data is measured in the *ratio scale*. Quantities like age, exercise and exam points are measured in this type of scale. This is a very useful property, because now we can compare two students according to their exam points and draw conclusions like "Type A students have got twice as many points as type B students in the exam". However, if the attributes are in different scales, we usually have to transform them into the same scale before modelling.

4.2.3 Static and temporal data

The data can be classified as *static* or *temporal* (*dynamic*) according to its behaviour with respect to time [Kan02][20-21]. In static data, the attribute values do not change in time, while in temporal data, the same tuple can have different attribute values, when the time proceeds.

Example 2 *A hypermedia learning system keeps record on all actions executed by the user. The actions are recorded as 4-tuples* $< userid, actiontype, target, time >$. *The action type can be* login, logout, load material, load exercise *or* return exercise. *The target identifies the page or exercise to be loaded or the solution to be returned. For login and logout operations the target is empty.*

Each session from login *operation to* logout *operation is saved as a sequence of actions. For each user, the log contains several sessions, and in each session, the* action type *and* target *values vary in time.*

In the educational context, static data is more commonly available, and most modelling paradigms are also designed for the static data. However, there are potentially interesting sources of temporal data, which should be further studied. For example, if we collect log data described in the previous example, we can analyze the students' typical navigation patterns through the topics, and even predict the best order for topics. This kind of experiments have been reported in [CGB03, SB03, KB05, RVBdC03]. We will return to this topic in Chapter 8.

4.3 Properties of educational data

In the following, we will analyze typical educational data according to our literature review (Chapter 2) and our own observations. We conclude that the educational data is typically sparse, skewed and contains relatively many outliers.

4.3.1 Data size

The most critical property of the educational data is the data size. The data sets in the educational domain are typically much smaller than in other domains where knowledge discovery has been applied. The problem is that we cannot learn the model accurately, if we do not have sufficiently data relative to the number of model parameters. As a rule of thumb (see e.g. [JDM00, Dui00]), we have sufficiently data, if for each model parameter we have at least 5-10 rows data. The number of parameters depends on the number of attributes and their domain sizes. In the simplest modelling paradigms, like linear regression and naive Bayes using binary data, the number of parameters is $O(k)$, where k is the number of attributes. Thus, we can simply check that $n/k > 5$. In the more complex modelling paradigms, the number of parameters can be an exponential function of the number of attributes. For example, in a general Bayesian network, containing k v-valued attributes, the number of parameters is in the worst case $O(v^k)$, and we should have at least $5v^k$ rows data.

The relative amount of data is measured by the *density* of data. It tells the average number of data points in a cell of *attribute space* (*data space*).

Definition 8 (Attribute space) *Let $R = \{A_1, ..., A_k\}$ be a set of attributes, whose domains are $Dom(A_1), ..., Dom(A_k)$. We say that $S = Dom(A_1) \times ... \times Dom(A_k)$ is an* attribute space *spun by attributes $A_1, ..., A_k$. Attributes $A_1, ..., A_k$ are called* dimensions *of the space. The* volume *of the attribute space is $|S| = |Dom(A_1)|...|Dom(A_k)|$.*

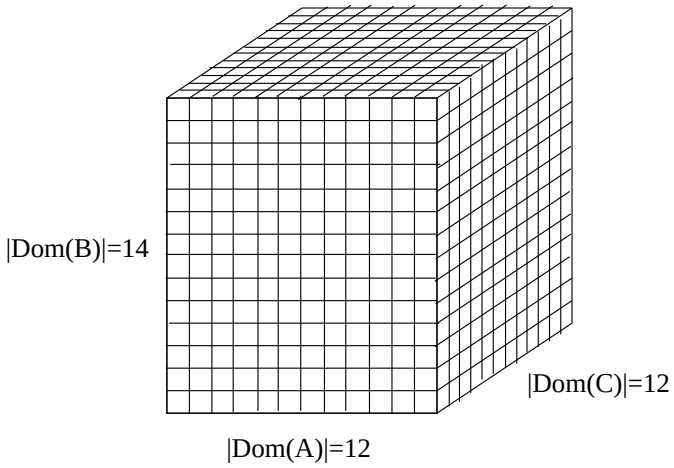

Figure 4.3: Attribute space spun by three attributes A, B and C. A tells the student's exercise points in basic programming skills, B the points in loops and arrays and C in applets.

In Figure 4.3, we give an example of 3-dimensional attribute space, whose volume is $12 \times 14 \times 12 = 2016$. Notice that if the data contains nominal attributes, the attribute space does not have a unique representation, because the order of nominal values is not fixed.

Definition 9 (Density of data) *Let $R = \{A_1, ..., A_k\}$ be a set of attributes, which spin an attribute space S. Let $|r| = n$ be the size of relation $r \in R$. Then the* average density of data *is*

$density(r) = \frac{n}{|S|}$.

If the average density is low, we say that the data is *sparse*. Otherwise it is *dense*. The exact thresholds for sparse and dense data depend on the context, but generally $density(r)$ should be at least 5 for predictive modelling.

In descriptive modelling, the problem is that we do not find any frequent patterns, if the data is very sparse. Once again, clustering is a special case. Sparsity is not a problem for clustering, but on the contrary, the clusters are harder to separate if the data is too dense. However, the distance metrics are usually very sensitive to high dimensions, and in practice we have to reduce the number of attributes.

According to our analysis, the educational data sets are typically too sparse for complex modelling paradigms. In Table 4.2, we have summarized the data sizes

and number of attributes in the previous experiments on applying knowledge discovery to educational data (Chapter 2). We have divided the experiments into three categories, according to the research scope. In the university level researches the data was collected from several courses and often during several years. The goal was to model the general academic success. In the course level researches the data was collected from one course and usually from one class. The goal was to model the course success. In the task level researches the data contained students' solutions to a set of tasks. The goal was to model the success in the next task.

Table 4.2: Summary of data sizes in the previous research on applying knowledge discovery to educational data. On each level, we report the minimum, maximum and average size (Avg.n) of data sets and the average number of attributes (Avg.k). The number of data sets in each category is given in parenthesis.

Scope	[Min,Max]	Avg.n	Avg.k
University level (5)	$[100, 22969]$	6113	25
Course level (9)	$[41, 350]$	181	26
Task level (4)	$[149, 55000]$	15058	22

The most interesting category is the course level researches, because most educational systems are designed for one course. In such systems, the data size is restricted by the size of the class. Student data from several classes can be pooled only, if the whole course content has remained unchanged. In the previous research, the average data size was $n = 181$ rows, but the average number of attributes was relatively high, $k = 26$. The upper bound for the average data density is $n/k = 7$. The density is so low that only simple models could be learnt/discovered accurately.

In the other two categories we have sufficiently data. Especially, in the university level, the data sets are generous relative to the number of attributes ($n/k = 225$). The task level researches are more difficult to evaluate, because the same attributes can be learnt for several tasks or concepts. For example, in [BJSM03] nearly 20 attributes were learnt for all words in the system, but the number of words was not reported. However, the size of the data set (555 818 read words) was so large that even tens of thousands of words could be estimated accurately. In all task level studies the data was collected from a system, which had specialized in one simple task type. The same skills were tested in all tasks, and thus a small student group (in [BJSM03] only 53 students) could produce sufficiently data. Unfortunately, this kind of systems are restricted to practicing elementary skills and they are hardly applicable in the university level.

Temporal data sets like system logs are often large and give impression that we have sufficiently data. However, now the number of attributes k is also large and

the number of model parameters is exponential $O(2^k)$. The following examples illustrate the problem.

Example 3 *Let us consider a learning environment, which models the student's skills in m concepts $C_1, ..., C_m$. Each concept is binary-valued and describes whether the student masters the skill according to her/his task solutions. At each time step, the student can be any of 2^m states described by the concept value combinations. If we want to predict the student's success in the next task, we should evaluate in the worst case 2^{2m} transition probabilities between the states.*

In practice, we can simplify the model, because an individual task does not measure all skills. Still, we would need $O(2^n)$ rows of data.

Example 4 *Let us consider a simple hypermedia learning system, which contains 20 html pages. The system log records only the page visits $< user, page >$ in the time order. The data sets contains 2000 sessions made by 100 students. A average session length is 19 page visits.*

We would like to discover all sequences of 5 pages which occur frequently. Now the number of all possible five page sequences (5-permutations) is $20^5 = 3200000$. Each session defines approximately 15 five page sequences. Now each possible five page sequence has $1 - \left(\frac{3199999}{3200000}\right)^1 5 = 0.0000047$ probability to occur in the given session. The pattern frequency follows a binomial distribution with parameters $n = 1000$ and $p = 0.0000047$. Probability that a sequence occurs in one session is 0.0047, but probability that the same sequence occurs in two sessions is only 0.0000011. It is likely that we do not find any frequent patterns!

In practice, we should reduce the number of possible actions by grouping the html pages according to their contents. Then we could search temporal patterns between the page groups. In addition, we could search action sequences, where actions occur in the same order but are not necessarily adjacent. Both tricks reduce the number of possible patterns and patterns become more frequent.

4.3.2 Data distribution

The density of data is usually not uniform, but varies in different parts of the attribute space. If the deviation from the uniform density is relatively large, we say that the data is *skewed* [BMFH97][5]. Note that only ordinal – either numeric or ordinal categorial – data can be skewed.

Definition 10 (Skewed data) *Let S, be an attribute space spun by a set of ordinal attributes R, $|R| > 1$. Let $r \in R$ be a relation and density(r) its average density. Let us denote the density of r in a subspace $T \subseteq S$ by density$_T(r)$. If S contains a subspace $T \subseteq S$ such that for some user-defined thresholds $\theta > 0$, $\gamma > 0$*

$$\frac{|T|}{|S|} > \theta, \text{ and } \frac{(density(r) - density_T(r))^2}{density(S)} > \gamma,$$

we say that r is skewed.

The skewness depends on two domain-dependent factors, γ and θ, which state that the deviation from the overall density in subspace T is high enough and the subspace T is large enough.

Note that the above given definition refers to "skew in frequency". In statistics, skewness often refers to a deviation from a symmetric density distribution ("skew in value"). Skew in value implies skew in frequency, but the opposite is not necessarily true [BMFH97][5].

Skewness is not necessarily a negative phenomenon, but vice versa. We cannot find any patterns in a totally uniform data. All models assume that the data is somehow skewed and the variations in the data density reflect the underlying model. Skewness becomes problem only when it does not fit the distributional form assumed in the modelling paradigm.

Educational data is often skewed, because the attributes are typically mutually dependent.

Example 5 *Let us consider the data in Figure 4.4. Attribute A tells the student's exercise points in basic programming skills and B the points in loops and arrays. A and B are clearly dependent. Students, who have achieved only a little of points in A, have not performed any better in B category. This can be seen as a sparse area in the left top corner. On the other hand, most students have achieved a lot of points in both categories. This can be seen as a dense area in the right top corner.*

Especially dense areas in the attribute space are not a problem for modelling, but they can reveal interesting tendencies in the data. Sparse areas are more problematic, because the model parameters cannot be estimated accurately in those areas. In the extreme, some areas are nearly empty, and we have no guarantees how the model behaves in those areas. Such areas are sometimes called *gaps* in the data.

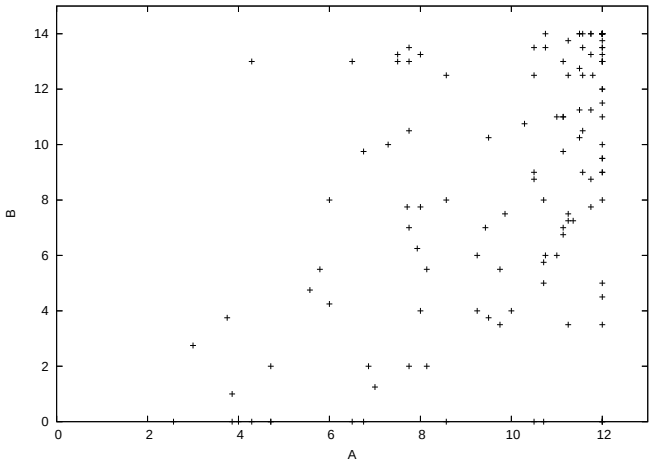

Figure 4.4: An example of skewed data. Attribute A tells the student's exercise points in basic programming skills and B in loops and arrays.

Another typical phenomenon for educational data is relatively large number of exceptional points, *outliers*. According to a classical definition, outlier is "*an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism*" [Haw80]. Other definitions refer to "abnormal behaviour", a point which does not fit the model, or a deviation from the probability distribution. The definitions depend on the context and in the the following we give a very general definition:

Definition 11 (Outlier) *Let $P(D|M)$ be the probability of data D and $P(o|M)$ the probability of point $o \in D$ given model M. Let $\gamma < 1$ be a user defined constant. If $P(o|M) < \gamma P(D|M)$, we say that o is an* outlier.

The problem in the definition and detection of outliers is the assumption of a model, which the data should fit. In reality, there may be no model at all or the discovered model can be spurious. Especially, if the sample is small and not random (e.g. students who have answered a prerequisite query), the reality can be just the opposite: the discovered pattern represents abnormal behaviour and the few outliers are normal in the whole population.

In Figure 4.5, we illustrate the difficulty of defining outliers. In the model, we assume that the course total points $TP1$ are linearly dependent on exercise points in B category. Now the number of outliers depends on how large deviation from the regression line is accepted.

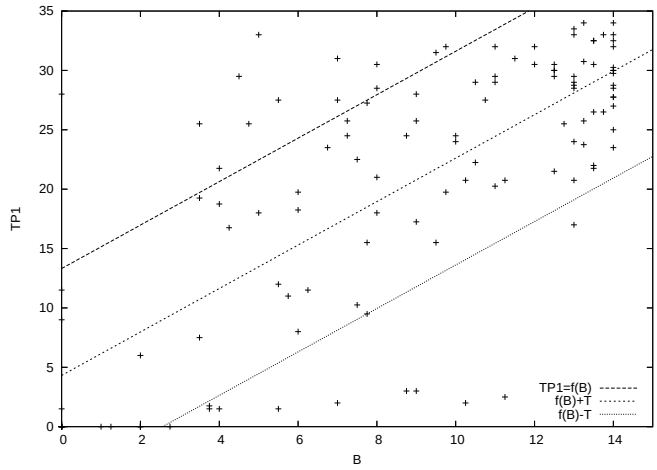

Figure 4.5: Examples of outliers, when we assume a linear regression model $TP1 = f(B)$. The number of outliers depends on how large deviation T from the regression line $f(B)$ is accepted.

4.4 Preprocessing data

The tasks of preprocessing phase can be divided into three categories: 1. *feature extraction*, 2. *feature selection* and 3. *processing distorted data*.

In feature extraction, the goal is to produce new better attributes from the original ones. In the previous, we have already defined the feature extraction from the unstructured data. Similarly, we can transform a structured data set (i.e. a relation) into a new relation, defined by new attributes. In feature selection, the goal is to determine the best subset of attributes for modelling purposes. In literature, feature extraction and feature selection are often used interchangeably [JDM00]. Some methods like *principal component analysis* combine both feature extraction and selection. They produce new features from the old ones, but at the same time they suggest the most relevant features to be selected. The third preprocessing task concerns how to process distorted data containing different *anomalies* like missing values, errors, and exceptional data values.

The main goal of feature extraction and selection is to find a good set of relevant features. The only problem is how to define relevance or utility. Very often the utility of features is discovered after modelling, and we should test all possible subsets of features, transformed in different ways, which is generally intractable. The dilemma is that for the optimal feature set we would already need the guidance of models – like a clustering or a classifier – but without a good set of features we

cannot learn such a model. As Duda & al. [DHS00][581] state it: "*The division of pattern recognition into feature extraction followed by classification is theoretically artificial. A completely optimal feature extractor can never be anything but an optimal classifier.*" As a solution, several heuristic methods have been developed, and the modeller should select the best ones according to the domain knowledge. This is a matter of art, but in the following, we try to give some general instructions for the educational domain.

The suggested techniques are based on our observations and the analysis of existing literature. Data preprocessing is unfortunately quite a neglected topic in the knowledge discovery literature and we have tried to fill this shortage in this thesis. Normalizations and some other preprocessing techniques as well as outlier detection are briefly described in [Kan02][22-28,33-36]. Feature extraction and selection are mostly discussed in pattern recognition literature (e.g. [JDM00, JZ97, DHS00]).

4.4.1 Feature extraction

Very often the educational data set consists of different types of attributes – both numerical and categorial, ordinal and nominal – in varying scales and measuring intervals. The goal of feature extraction is to remove these differences so that the attributes could be used in the same model. Another goal is to reduce the data volume by reducing the sizes of attribute domains, and thus produce simpler models. In addition, the minor differences in the attribute values are smoothed, which reduces the influence of noise in the data.

In the following we list common feature extraction methods. The methods have been divided into four categories, according to how much they change the original attributes. Scaling methods do not change the attributes or their domain types, but only the attribute values. In domain reduction techniques we create a new attribute from an old one by reducing the attribute domain. In the last two methods we create a totally new set of attributes by combining the original ones either based on domain knowledge or attribute transformations.

Scaling

Scaling is the most frequently used feature extraction technique for numeric data. The simplest form is linear scaling, in which new attribute value x' is a linear function of original value x: $x' = \alpha x + \beta$. For example, in *decimal scaling* we just shift the decimal point: $x' = 10^i x$. If the attribute is not in ratio scale (i.e. it does no have an absolute zero point), the constant β is also needed.

If the goal is to scale all attributes into the same scale (the same domain), the method is called *normalization*. In *min-max normalization* the attribute values are normalized to interval $[0, 1]$ by equation

$$x' = \frac{x - min(X)}{max(X) - min(X)},$$

where $min(X) \neq max(X)$.

Another popular method is known as *standard deviation normalization* or simply *standardization*. It is especially useful, if the variables should also have the same dispersion (standard deviation). The new attribute values are calculated by subtracting the mean and dividing by the standard deviation:

$$x' = \frac{x - mean(X)}{stdev(X)},$$

where $stdev(X) > 0$. As a result, all variables will have zero mean and unit standard deviation. This transformation is often recommended, when Euclidean metric is used (e.g. in clustering or linear regression) and the data contains several outliers.

Domain reduction

In the educational domain data reduction techniques are often needed, because the data sets are small and sparse. With more dense data the model parameters can be evaluated more accurately. In addition, reduction enables simpler models, and the danger of overfitting decreases. However, there is always a danger of overgeneralization. Important information can be lost and even new inconsistencies can appear [Xia03].

The size of the attribute domain can be reduced efficiently by aggregating the attribute values into groups, which are given new labels. For numeric data this is called *aggregation* and for categorial data *generalization*. *Discretization* is an important special case of data reduction, which transforms numeric (continuous) values into a smaller set of discrete (either numeric or categorial) values. Discretization and data reduction can be implemented in several ways, but most often the original domain is simply divided into equal-sized intervals. If the equal-sized intervals have very different frequencies, a better approach is to select intervals which contain equally many tuples. This method produces more robust models, because all model parameters are estimated from equally many data points. However, the resulting model can be more difficult to interpret.

Ordinal data can be generalized similarly, by defining value intervals. For example, the course grades $1-, ..., 3$ can be replaced by new values *satisfactory* $= [1-, 2-]$, *average* $= [2-, 2+]$), and *excellent* $= [3-, 3]$. Nominal values can be combined by logical operations like disjunctions. More sophisticated techniques like [Xia03] use χ^2 -test and entropy to determine which attribute values or intervals should be combined.

Sometimes it is desirable to reduce all data to binary-valued 0/1 data. For example, the efficient algorithms for searching association rules assume that the data is binary. If the attribute domain contains only a couple of values, the data can be transformed to binary by creating new attributes (attribute $A = \{a_1, ..., a_n\}$ is replaced by n binary attributes). However, if the domain is larger (e.g. numeric), this is no more feasible and discretization is needed.

Combining attributes by domain knowledge

In educational data sets the feature selection is often based on the domain knowledge. The domain experts have either intuitive or theoretical knowledge about the main factors which affect the learning process. The goal is to calculate these attribute values from the data. The decisions are always problem-dependent. In the following, we give an illustrative example.

Example 6 *In the* ViSCoS *data, the main part of data consisted of weekly exercise task points.* Prog.1 *consisted of 9 exercise weeks and* Prog.2 *10 exercise weeks. The number of attributes was too high relative to data sizes and, in addition, the compositions of weekly exercises were not the same in two academic years. Before we could pool the data sets from both years, we should first abstract away the differences.*

As a solution, we analyzed the topics covered in the exercise tasks. In Prog.1 *the topics fell into three categories: 1) basic programming structures, 2) loops and arrays, and 3) applets. In* Prog.2*, we could also recognize three categories: 1) object-oriented programming, 2) graphical applications, and 3) error handling. The new attributes A, B, C, D, E, F are defined in Table 4.3.*

The new attribute values were defined by summing the exercise points in each category and normalizing them to the same scale. Normalization was required, because the maximum points varied in the consecutive years. Because the minimum point values were 0 in all categories, we could perform min-max normalization by simply multiplying with $\frac{\max_{new}}{\max_{old}}$, where $\max_{new}$ was the maximum of maximum points in two years.

Table 4.3: Exercise task categories.

Cat.	Description
A	Basic programming structures (*Prog.1*, weeks 1-3)
B	Loops and arrays (*Prog.1*, weeks 4-6)
C	Applets (*Prog.1*, weeks 7-9)
D	Object-oriented programming (*Prog.2*, weeks 1-3)
E	Graphical applications (*Prog.2*, weeks 4-8)
F	Error handling (*Prog.2*, weeks 9-10)

The resulting data set consisted of 8 numeric attributes, which were also discretized to binary-valued data. The decision of binary data was based on trial-and-error. We tried also discretization to three values, but the the resulting data set was too sparse to reveal any statistically significant patterns.

The exercise points $A, ..., F$ *were discretized by dividing the attribute domains into two equal size intervals. I.e. for all* $X = A, ..., F$*, the binary attributes* X' *were derived by rule:*

$X' = little$*, if* $X < \frac{\max(X)}{2}$*, and* $X' = lot$*, otherwise.*

This discretization was selected for its simplicity and quite likely it is not optimal. In the point distributions (Figure 4.6 and Figure 4.7) we observe that the data is very skewed. In the future research we will test discretizing the data by searching optimal segmentations.

The total points were discretized in another manner, because the goal was to predict the final results (pass/fail). The new attributes $FR = FR1, FR2$ *were derived from total points* $TP = TP1, TP2$ *by rule:*

$FR = fail$*, if* $TP < 15$*, and* $FR = pass$*, otherwise.*

In addition, we created binary attribute $TP1$*, which measured whether a student who had passed the* Prog.1 *course has achieved a little or a lot of total points in* Prog.1*. This attribute could be useful for modelling* Prog.2 *data, because only the students who had passed* Prog.1 *could participate* Prog.2*. A similar attribute was defined also for* $TP2$*, although it was not needed for modelling* Programming *courses. The new attributes* $TP' = TP1', TP2'$ *were derived from* $TP = TP1, TP2$ *by rule*

$TP' = little$*, if* $TP < 23$*, and* $TP' = lot$*, otherwise.*

The resulting attributes and their numeric and binary-valued domains are summarized in Table 4.4. Notice that the binary data can be represented as 0/1 data, and

in the following we will often use abbreviations $\neg X$ and X for $X = little\ (fail)$ and $X = lot\ (pass)$.

Table 4.4: Selected attributes, their numeric domain (*NDom*), binary-valued qualitative domain (*QDom*), and description.

Attr.	NDom.	QDom.	Descr.
A	$\{0, .., 12\}$	{little, lot}	Exercise points in A.
B	$\{0, .., 14\}$	{little, lot}	Exercise points in B.
C	$\{0, .., 12\}$	{little, lot}	Exercise points in C
D	$\{0, .., 8\}$	{little, lot}	Exercise points in D
E	$\{0, .., 19\}$	{little, lot}	Exercise points in E
F	$\{0, .., 10\}$	{little, lot}	Exercise points in F
$TP1$	$\{0, .., 34\}$	{little, lot}	total points of *Prog.1*
$TP2$	$\{0, .., 34\}$	{little, lot}	total points of *Prog.2*
$FR1$	$\{0, 1\}$	{fail, pass}	final result of *Prog.1*
$FR2$	$\{0, 1\}$	{fail, pass}	final result of *Prog.2*

Linear transformations: *PCA* and *ICA*

Attribute transformations can be used for extracting and selecting features from numeric data. In the classical approaches the new attributes are produced as linear combinations of the original ones. Usually the main goal is dimension reduction, i.e. to produce a new smaller set of attributes, which would still represent the data well. The underlying idea is that the number of real dimensions of the data can be smaller than the apparent number of dimensions, and in fact the features can be represented in a lower dimensional space. The educational data is often sparse and skewed, and it is quite likely that it could be represented by fewer but more descriptive attributes.

The most popular transformation technique is *principal component analysis* (*PCA*) (e.g. [HMS02][74-84]). The goal of *PCA* is to produce a new set of attributes (*principal components*), which catch the main sources of variance. It is assumed that such attributes describe the data best. The new attributes are uncorrelated and *PCA* suits for removing harmful correlations from the data. That is why *PCA* is often performed before linear regression or clustering analysis. If *PCA* is used for dimension reduction, we select only those components which have the highest variance.

PCA has also some restrictions and disadvantages [Agg01, AGGR98, TPP$^+$02]. The main problem is that the new attributes are not necessarily the same as the

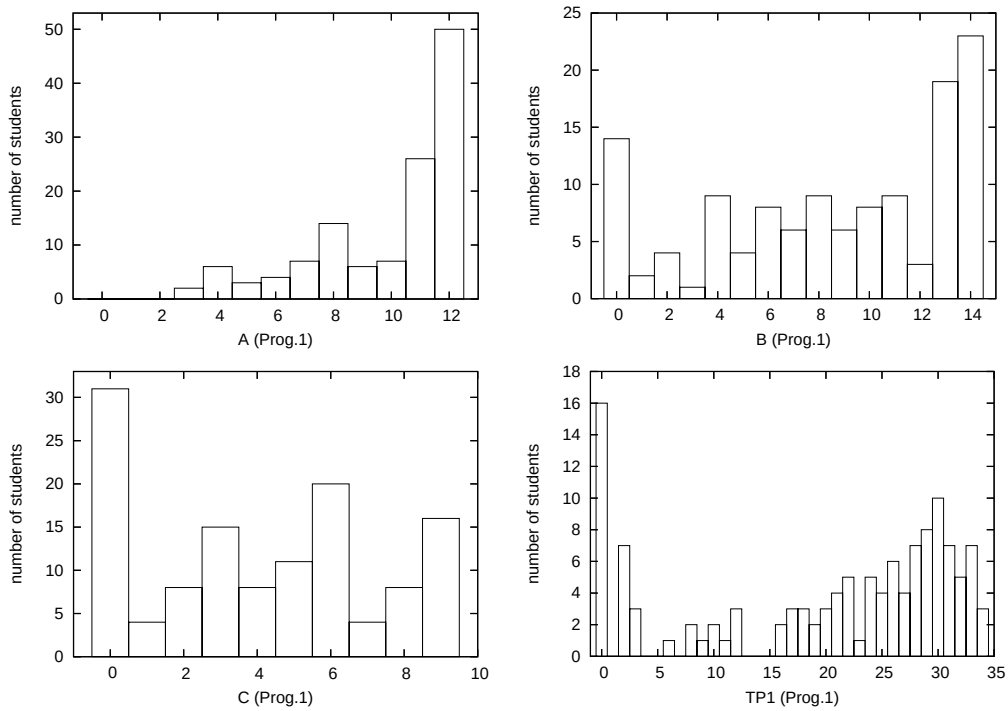

Figure 4.6: Point distributions in *Prog.1* course. To illustrate the distributions better we have represented the attributes in different scales.

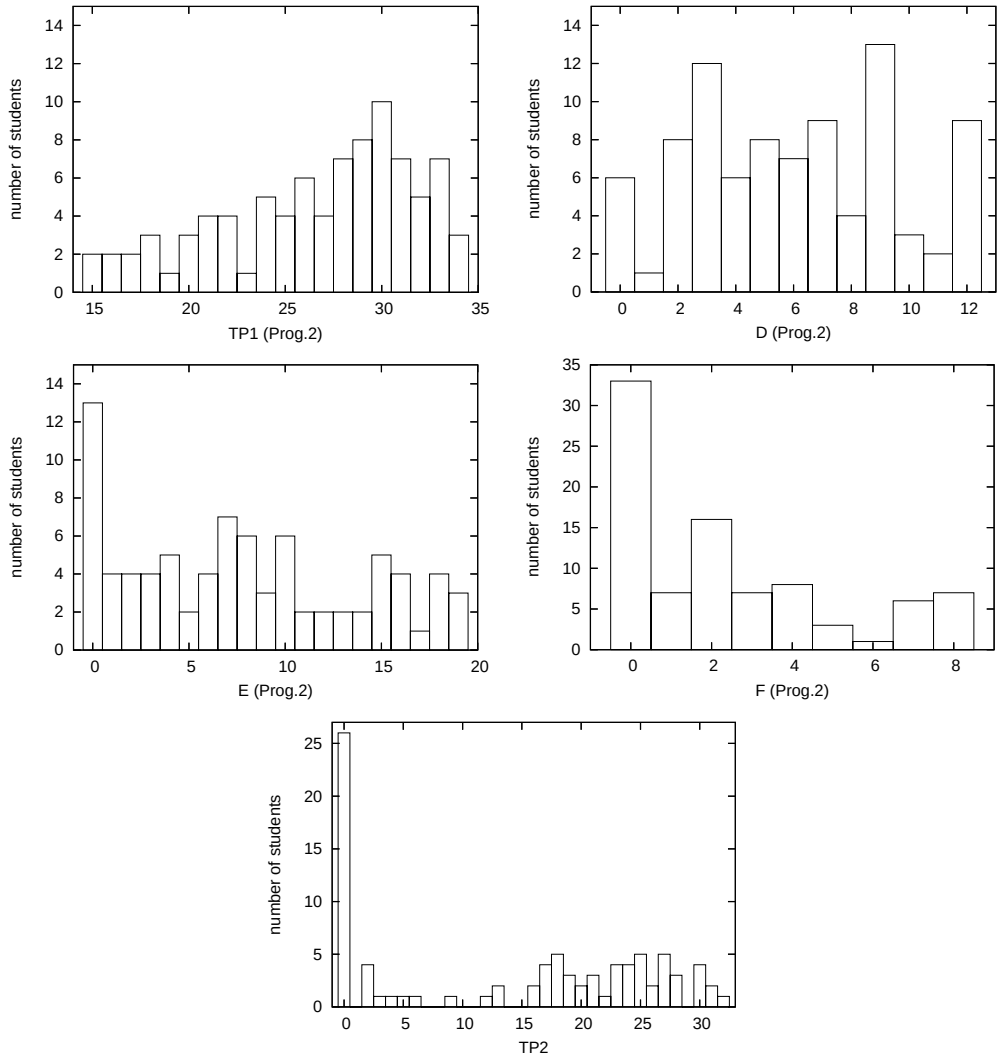

Figure 4.7: Point distributions in *Prog.2* course. To illustrate the distributions better we have represented the attributes in different scales.

most discriminatory features. *PCA* can produce really poor results, if the data consists of several "noise attributes", which contain large but meaningless variance. A pathological example is an attribute, which has a uniform distribution and which does not correlate with any other attribute. *PCA* considers this attribute as the most important, while it is actually the most meaningless one.

Another problem is that *PCA* assumes normal distribution. The sparser and more skewed the data set is, the more it deviates from normality. In addition, the data size should be large enough. Different rules of thumb have been proposed for the minimal size of data (see. e.g. [OC02]), but there is no consensus. Sometimes it is required that the data size is at least 100 or 200 rows, while other rules require that there are at least 5 or 15 rows of data per each original attribute. These rules indicate that *PCA* is appropriate for the educational data sets only in special cases, but the potential benefits are so valuable that the topic should be further researched.

Independent component analysis (*ICA*) (e.g. [Com94, AAG03]) is another well-known transformation technique. It can be considered as a generalization of *PCA*, because now the resulting new attributes (*independent components*) are not merely uncorrelated but statistically as independent as possible. The statistical independence is measured by mutual information between the new attributes. If the data is normally distributed, *ICA* produces exactly the same attributes as *PCA*. In practice, *ICA* produces more discriminatory features and thus it is preferred in classification tasks. In addition, it can be used for removing statistical dependencies between explanatory variables, which is desirable in naive Bayes classifier.

ICA does not assume normally distributed data, which makes it more flexible. The effects of data size have not been studied, but it is quite likely that *ICA* requires large data sets to work correctly [AAG03].

Finally, we note a special problem concerning applications of *PCA* and *ICA* in the educational domain. In the educational domain the models should be easy to interpret, but *PCA* and *ICA* produce often incomprehensive models, because the new attributes do not have their natural meaning anymore.

4.4.2 Feature selection

In the educational domain the feature selection is especially important. The motive is the same as in feature extraction: we should reduce the data volume to increase its density. In feature extraction the volume was reduced by domain reduction, but now we should select the best subset of attributes, $X \subseteq R$, for modelling purposes. The problem is to recognize irrelevant attributes from relevant ones.

In descriptive modelling we usually perform an exhaustive search over all possible patterns, and then select the best ones according to some goodness criterion. There is not necessarily any need for feature selection, because all attribute sets will be tested. The only exception is clustering, where irrelevant attributes can corrupt the distance measures and real clusters are not discovered. In high dimensions (already with 10-15 attributes) the contrast between distances to the nearest and furthest data points often blurs [BGRS99] and clusters become hard to separate. This occurs especially with the Euclidean distance metric L_2 and higher-order metrics L_p, $p > 2$ [HK98, AHK01]. As a solution, we can use Manhattan metric, L_1, or fractional metrics L_p, $0 < p < 1$. This makes clusters better separable, but it does not take into account the meaningful directions. Often the clusters can be found more easily in a new subspace of transformed attributes.

In predictive modelling feature selection is more crucial, because the models are more complex and we cannot try all possible models. In addition, the prediction accuracy can suffer for irrelevant attributes, because the model becomes unnecessary complex and overfits the data. If the data set is large, the irrelevant features are less harmful for the prediction accuracy, but working in high dimensions is inefficient.

The goodness of attributes depends on the modelling goals, but generally we try to identify the most *discriminatory* or the most *descriptive* or *expressive* attributes. Discriminatory features refer to the attributes which classify the data well, while descriptive or expressive features produce models which describe essential patterns in the data. In practice, the optimal feature set is selected according to some criterion J, which depends on the modelling purpose. For example, in classification a common criterion is the classification accuracy $J = 1 - error(M, r)$, where M is the model learnt with selected attributes X from a data set r.

The only way to guarantee an optimal attribute subset is to try all dimensionalities $l < k$, where $|R| = k$, and all $\binom{k}{i}$ l-attribute combinations, and select the one which maximizes the criterion J. The search can be pruned by branch-and-bound technique, but still exhaustive search cannot be avoided [CC78]. In practice the problem becomes intractable even with moderate k, and local optimization techniques are needed (see e.g. [JZ97]). The simplest technique is to test all attributes separately and select the set of individually best attributes. However, if the attributes are strongly dependent, the technique can select irrelevant attributes. Another approach is to merge or divide attribute sets iteratively in a greedy manner, but the resulting attribute set is only locally optimal.

Once again, the problem can be solved by a combination of descriptive and predictive modelling. In descriptive modelling we search all interesting patterns, and then select the best attribute combinations for predictive modelling.

In some cases we can recognize the irrelevant attributes more easily. Derived attributes, which have been calculated from other ones, contain no new information and can be always pruned. For example, in the *ViSCoS* data we had several attributes which were derived from exercise points (total exercise points, their proportion from the maximum exercise points, and bonus points). If the derived attributes are not known by the domain experts, they can be recognized by comparing distributions: if the distributions of two attributes are the same, then the other one can be dropped. If the distributions are very similar, the attributes can be combined simply by taking a normalized sum of attribute values.

4.4.3 Processing distorted data

Processing distorted data is often called "data cleaning". The goal is to remove or reduce the influence of different anomalies like missing or erroneous values, duplicates, inconsistencies (contradictory data), noise, and outliers.

In the educational domain we can suppose that the data is quite clean. The attribute values are often entered manually, but with special care, and students usually check them, too. However, when we combine data from several sources there are always some tuples which occur in one record but not in the other, resulting *missing attributes*. The true attribute value is also missing, when an attribute value is clearly *erroneous*, e.g. the student has got more than maximum points (which the student does not complain). In both cases there are two basic solutions: we can either delete the whole tuple or replace the missing value with some constant ("unknown"), or mean value in the whole population or in some subclass. In an extreme case, we can learn a classifier and predict the most probable value. If we are learning a classifier and the class value is missing, the tuple is usually deleted. However, if we have only a little data, we cannot delete too many tuples. On the other hand, replaced values are never the same as correct values. Substitution with mean values decreases artificially the variation and it can considerably change the model (for example correlation and linear regression). If all the missing values are in the same attribute, it may be better to drop the whole attribute. For example Agrawal [AGGR98] recommends this solution for clustering with several missing attribute values. Finally, it is should be remarked that several missing attribute values can also indicate some interesting pattern in the data. For example, it is typical that only the most active students participate in voluntary activities.

Duplicates are an easier problem. Usually the tuples are identified by student numbers and duplicates are easy to recognize. In addition, duplicates are not necessarily problem in some models. For example, in clustering they have usually no effect.

Sometimes different data sources are *inconsistent*, i.e. they contain tuples which concern the same data point but some attribute values are different. The inconsistencies may be due to different formats, updates, or errors. Usually it is best to use domain knowledge or ask an expert (teacher) to correct the values. For example, the *ViSCoS* data contained some students whose final points and grades were not consistent. Probably the students had participated in several exams, and the points were not updated to all records. Because these inconsistencies could not be solved, the only solution was to delete them.

Noise is a general term used to refer to different kinds of random errors in data. Noise is usually present in physical measurements, where other signals from the device or environment are mixed with the measured ones. In the educational domain the data is seldom noisy in this sense, but we can imagine other sources for noise. For example, if the exercises are evaluated by different teachers, their subjective tendencies can add bias or "noise" to the points. The main solutions for noisy data are "data smoothing" and robust models. Data smoothing can be performed by scaling, discretization and other domain reduction techniques, which lift the abstraction level and smooth out the minor differences. On the other hand, robust models do not overfit to represent noise, but only the main structures in the data.

We have already noted that the educational data contains often relatively many outliers, which decrease the model accuracy. The problem is two-fold: first we should detect the outliers and then decide how to handle them. The problem is difficult, because the same techniques – e.g. linear regression and clustering – which can reveal outliers, are themselves especially sensitive to outliers. The reason is that linear regression and most clustering algorithms use *SSE* score (i.e. they minimize the sum of squared errors), which is very sensitive to outliers. Generally, all measures which use mean and variance suffer for outliers. Median is a more robust measure than mean, if there are only a few outliers [Hig00].

One-dimensional outliers, which differ significantly from other data points only in one attribute, are relatively easy to detect by *univariate tests*. The outlier is defined as a function of mean and variance, e.g. a data point x is an outlier in dimension A, if $|x[A] - mean(A)| > \gamma stdev(A)$ for some coefficient $\gamma > 1$. The only restriction is that we should assume some data distribution, before we can set an appropriate coefficient γ.

Visual checking (scatter plots) is often a useful technique, but it suits only for 2-dimensional data. For higher dimensional data all attribute distributions can be checked separately, but the outliers can also "hide" in higher dimensions. This kind of outlier has no extreme values in any one dimension, but is very distant from other points in the whole attribute space.

For multidimensional outlier detection a common approach is to define some distance or similarity function, and search for very distant or dissimilar data points. In distance-based techniques we define some distance function d, a threshold γ and proportion p. If $d(x, y) > \gamma$ for at least p % of data points y, x is considered as an outlier. The obvious problem is how to define appropriate parameter values γ and p. In addition, this methods suits only for numeric data, for which we can define a distance function. For other data types we can define a similarity or dissimilarity function, and define outliers as a minimal set of data points, whose removal minimizes (maximizes) dissimilarity (similarity) function.

In the educational domain the outliers are of special interest, because they usually reveal interesting information. Both exceptionally poor and excellent students need a special attention. Thus, the outliers should not be removed from the data, if they are not clear errors. If this is not clear, it is recommended (e.g. [Sto96]) to model the data both with suspected outliers and without them. In clustering a natural solution is to separate the outliers into their own cluster. In predictive modelling we can define the outliers as data points which do not fit the model, and search an optimal model for the majority of data. Now the majority of data can be predicted accurately, but for outliers we should invent special prediction rules or leave them unpredicted.

Usually better than deleting outliers is to soften their influence by data smoothing and use of robust models. The large differences in attribute values can be smoothed in the preprocessing phase by different transformations like logarithms, square roots, and *PCA*, but these transformations also blur the natural interpretation of attributes. Domain reduction is an efficient way to deal with outliers. For example, if we replace numerical values by ordinal categorial values, the large attribute values do not matter any more. However, radical reductions can lead to over-generalization. Once again, the best solution is to use robust models, which are not sensitive to outliers. We will return to this topic, when we compare clustering and classification methods in Chapters 6 and 7.

Chapter 5

Modelling dependencies between attributes

The main goal of predictive modelling is to predict a target attribute Y from a set of other attributes $X = \{X_1, ..., X_k\} \subseteq R$. Variables X are called *explanatory*, because they explain Y. The existence of such model requires that Y depends on X. Thus, the first step of the modelling process is descriptive analysis of dependencies between Y and X. The task is two-fold: First, we should select an attribute set X which best explains Y. Then we should analyze the type of dependency. Given this information, we can select an appropriate predictive modelling paradigm and define restrictions for the model structure.

In the following we define the main types of dependencies for categorial and numeric data. We introduce three techniques (correlation analysis, correlation ratios, and multiple linear regression) for modelling dependencies in numeric data and four techniques (χ^2 independence test, mutual information, association rules, and Bayesian networks) for categorial data. In both cases, we begin by analyzing pair-wise dependencies between two attributes, before we analyze dependencies between multiple attributes $X_1, ..., X_k$ and target attribute Y. This approach has two benefits: First, we can avoid testing all 2^k dependencies between subsets of $\{X_1, ..., X_k\}$ and Y, if Y turns out to be independent of some X_i. Second, this analysis can reveal important information about suitable model structures. For example, in some modelling paradigms, like multiple linear regression and naive Bayes model, the explanatory variables should be independent from each other. Finally, we analyze the suitability of described modelling techniques for the educational domain.

5.1 Dependencies

Dependencies between attributes fall into two main categories. Dependencies between discrete attributes are measured by *satistical dependency*, based on the observed probability distribution. The dependencies between numeric attributes are measured by *functional dependency*, based on individual data points.[1]

Statistical dependency between variables X and Y is usually defined through statistical independency. The reason is that we can always observe absolute independency, but it is much more difficult to determine when the dependency is significant.

Definition 12 (Statistical independency) *Let R be a set of discrete attributes, and $X \in R$ and $Y \in R$ be attributes defined in R. Attributes X and Y are* statistically independent *in relation $r \in R$, if $P(X,Y) = P(X)P(Y)$, where probability P is the relative frequency in r.*

Statistical independency can be generalized to multiple attributes: Attributes $X_1, ..., X_k$ are mutually independent, if $P(X_1, ..., X_k) = P(X_1)...p(X_k)$.

If X and Y are not independent, they are more or less dependent. The strength of the statistical dependency between X and Y is defined by comparing $P(X,Y)$ and $P(X)P(Y)$. Sometimes the dependency between X and Y can be explained by a third attribute Z. Once again, we can define unambigiously only the conditional independency:

Definition 13 (Conditional independency) *Let R be a set of discrete attributes, $X \in R$, $Y \in R$ attributes and $Z \subseteq R$ a set of attributes defined in R. X and Y are* conditionally independent *given Z, if $P(Y|X,Z) = P(Y|Z)$ and $P(X|Y,Z) = P(X|Z)$.*

The (unconditional) statistical independency can be interpreted as a special case of conditional independency, when $Z = \emptyset$. I.e. X and Y are statistically independent, if $P(X|Y) = P(X)$ and $P(Y|X) = P(Y)$.

Functional dependency is a special kind of dependency, where the dependency relation is defined by some function:

[1] Some authors use term "statistical dependency" to refer to any kind of dependency observed by statistical means. In this thesis we follow a more common interpretation.

Definition 14 (Functional dependency) *Let $X = \{X_1, ..., X_k\} \subseteq R$ be a set of attributes, r a relation according to R, and $Y \in R$ an attribute defined in relational schema R. Y is* functionally dependent *on X in relation r, if there is a function $f : Dom(X_1) \times ... \times Dom(X_k) \rightarrow Dom(Y)$ such that for all $t \in r$ $t[Y] = f(t[X])$.*

In relational databases the attributes are always discrete, and the function is not continuous. In addition, we are seldom interested in the actual dependency function. In such a case we can simply define that Y is functionally dependent on X, if for all $t \in r$ $t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$. However, when the attributes are numeric, we can approximate a continuous function, which fits our data points. The quality of such an approximation depends on how representative our data set is. If the data is sparse or contains several gaps, we cannot approximate the function accurately.

5.2 Correlation analysis

In the following we recall the most common measure for correlation, Pearson correlation coefficient. We discuss about restrictions and extensions of the common correlation analysis. Finally, we analyze the *ViSCoS* data by Pearson correlation and correlation ratios to reveal linear and non-linear dependencies.

5.2.1 Pearson correlation coefficient

Correlation analysis (see e.g. [Gar05, Sta04])is the simplest descriptive technique, which reveals linear dependencies between two variables[2]. It is recommendable to perform correlation analysis for all numeric data before any other modelling, because undesirable correlations can corrupt the whole model. The same holds for other dependencies, but correlations are easiest to compute and they already reveal if the variables are not independent. In addition, if the data is normally distributed, the lack of correlation means also statistical independence[3], and no other tests are required. According to our experiments, the assumption of normality does not hold for educational data (students' task points), but still the dependencies are quite linear.

The most well-known measure for correlation is *Pearson product-moment correlation coefficient*, which measures the degree of linear dependency between numeric variables.

[2]Sometimes term "correlation" is used to refer to any kind of functional dependency, but we follow the most common interpretation of correlation as a linear relationship.

[3]This can be easily seen, when we assign $r = 0$ in the 2-dimensional normal density function.

Definition 15 (Pearson correlation coefficient) *Let X and Y be numeric variables and $stdev(X)$ and $stdev(Y)$ be finite and nonzero. Then the* correlation coefficient $r = corr(X, Y)$ *between X and Y is*

$$corr(X,Y) = \frac{cov(X,Y)}{stdev(X)stdev(Y)} = \frac{\sum_{i=1}^{n}(x_i - mean(X))(y_i - mean(Y))}{\sqrt{\sum_{i=1}^{n}(x_i - mean(X))^2 \sum_{i=1}^{n}(y_i - mean(Y))^2}}.$$

Very often, the square of correlation coefficient, r^2, is reported instead of r. It has a natural interpretation as a proportion of variance in X explained by Y (or vice versa).

The values of correlation coefficient fall between $[-1, 1]$. The sign defines the direction of the relationship. We say that the correlation is either *negative* or *positive* depending on the sign of the correlation coefficient. The negative correlation between X and Y means that Y decreases while X increases and the positive correlation means that both variables increase simultaneously. The absolute value of the correlation coefficient measures the strength of the relationship. Value $corr(X, Y) = 0$ indicates that X and Y are linearly independent, and $|corr(X, Y)| = 1$ indicates a perfect linear relationship. However, in practice, the correlation coefficient achieves very seldom values $+1$ or -1. According to [DD03], this can happen only if the attribute distributions diverge at most in the location and/or the scale.

An important question is when the correlation is significant. General guidelines are sometimes given for defining weak, moderate, or strong correlation but they are rather arbitrary, because the significance depends on the context. One important criterion is the size of the data set: the smaller the data set is, the stronger the correlation should be to be significant. In statistics, the significance of the correlation efficient is usually determined by t-test. The correlation is significant at level p, if

$$t = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}} \leq t_{n-2,p},$$

where $t_{n-2,p}$ is the critical value with $n-2$ degrees of freedom. In Table 5.1, we have calculated the thresholds for significant ($p = 0.01$) and very significant ($p = 0.001$) correlation, when the data size is 80,...,120. If we test only a couple of correlation coefficients, we can use lower bounds, but the more correlations we test, the more probably we will encounter spurious correlations. Thus, it is recommended (e.g. [Gar05]) to use stricter bounds, the more correlations we test. For example, if we test 50 correlations and want to find all correlations which are significant at level 0.05, then each individual correlation should be significant at level 0.001.

Table 5.1: The critical values of correlation coefficient r in significant ($p = 0.01$) and very significant ($p = 0.001$) correlation according to t-test, given data size n.

n	$p = 0.01$	$p = 0.001$
80	0.26	0.34
90	0.24	0.32
100	0.24	0.31
110	0.22	0.29
120	0.21	0.28

5.2.2 Restrictions and extensions

The correlation coefficient can be distorted or misleading for various reasons. The main assumption is that the relationship can be described by a linear curve. Pearson correlation coefficient does not tell anything about the strength of non-linear relationships. The correlation coefficient can be zero, even if there is functional dependency between variables. The following example is imaginary, but similar situations can occur in practice.

Example 7 *Let's consider the students' total points and satisfaction in* Theory of computation *course. The total points* $TP \in \{0, ..., 100\}$ *are approximately normally distributed with* $mean(TP) = 50$. *The students have evaluated their satisfaction* SAT *with the course in scale* $[0, ..., 100]$ *%. The satisfaction is a function of total points*

$$f(SAT) = \frac{-(TP - 50)^2}{25} + 100.$$

I.e. the students with average total points have been very satisfied with the course, but the students with extremely low or high total points have been very unsatisfied. Now the correlation coefficient is $r = 0.0$, *even if the dependency is fully functional.*

Another important note is that the correlation is not the same as causation. Correlated variables may or may not have causal relationship. In addition, the connection between variables may be indirect, due to a third unknown variable.

The main restriction of correlation analysis is that the variables should be numeric and measured in the interval scale. The only exception is dichotomous categorial variables like 0=*female*, 1=*male*. However, it can be hard to evaluate the significance of correlation between different data types or even between two numeric

variables measuring different quantities. In this case other measures like *Spearman's* ρ or *Kendall's* τ are recommended (e.g. [Gar05, Sta04]). Both of them apply for measuring correlation between two ordinal variables or an ordinal and a numeric variable.

Data preprocessing can have remarkable influence on the correlation coefficients. For example, domain reduction often restricts the variance and the correlation coefficients become attenuated. On the other hand, scaling is not needed, because the Pearson correlation performs an implicit standard deviation scaling. One problem is that the Pearson correlation is very sensitive to noise and outliers. This can be problematic in the educational data, where outliers are typical. In addition, the results can be misleading, if the relationship varies depending on the values. This phenomenon is known as the *lack of homoscedasticity*.

Finally, the significance tests assume that the variables are at least approximately normally distributed. In the educational data this assumption holds rarely, but, fortunately, the violation of this condition is not critical, if the data sets are large enough. According to several researchers (e.g. [Sta04]), the normality assumption can be ignored, if the data size is more than 100.

Correlation ratio [Kel35] is an effective measure, which can detect nearly any functional dependency between variables. Correlation ratio η can be interpreted as a generalization of the Pearson correlation coefficient for measuring non-linear relationships. It measures the proportion of the variability of Y for fixed values of X and the variability of Y across the whole data set. For numeric data the most common measure for the statistical variability is standard deviation and for categorial data entropy. Assuming the standard deviation measure, we define:

Definition 16 (Correlation ratio) *Let X be a discrete numeric attribute and Y any numeric attribute. Let x be a value of X, n_x the number of data points having value $X = x$, $Var_x(Y)$ the variance of Y, when $X = x$, and $Var(Y)$ the variance of Y in the whole data set. Then the square of the correlation ratio between X and Y, $\eta^2(X, Y)$, is given by*

$$\eta^2(X, Y) = \frac{nVar(Y) - \Sigma_x n_x Var_x(Y)}{nVar(Y)}.$$

Now η^2 has a natural interpretation as a proportion of Y's variance explained by individual variances in discrete values of X. Contrary to correlation, this relationship is not symmetric, and generally $\eta(X, Y) \neq \eta(Y, X)$. If the relationship is purely linear, then $\eta^2 = r^2$. Otherwise the difference $\eta - |r|$ tells to which extent the relation

is non-linear. This is especially useful, when we should decide whether the data can be modelled by linear models or whether we need a more powerful model.

Finally, we should note that the correlation ratio is sensitive to the precision used to represent decimal numbers. This phenomenon was first observed in our experiments with the *ViSCoS* data, but soon we realized also the reasons: If the precision is very high, there is only one Y value per each X value, and $Var_x(Y)$ is zero for all x. The resulting $\eta^2(X, Y)$ is one, which indicates non-linear dependency, unless $r^2 = 1$. However, when we discretize X with appropriate precision, the non-linearity decreases.

5.2.3 Pair-wise functional dependencies in *ViSCoS* data

In the *ViSCoS* project we also started with the correlation analysis. In addition, we calculated the correlation ratios between all attribute pairs in the *Prog.1* and *Prog.2* data sets. The goal was to discover all pair-wise dependencies in the data and evaluate how linear they are. The results are represented in Table 5.2. The thresholds for significant correlations have been defined according to t-test, using Table 5.1.

We observe that the exercise points in B category (loops and arrays) correlate strongly with total points $TP1$ in *Prog.1* and moderately with total points $TP2$ in *Prog.2*. The latter correlation can be partially reduced to the correlation between B and E, which suggests that skills in loops and arrays are important prerequisites for graphical applications. The exercise points in category E have a very strong correlation with $TP2$ in *Prog.2* course. These observations suggest that the student's performance in the middle of the course has a strong impact on the final results.

The strong correlation between $TP1$ and $TP2$ can either tell about the general learning tendencies, or prove the importance of managing basic programming skills before proceeding to more difficult topics.

Gender (G) was excluded from the correlation table, because it did not have any significant correlations with other attributes. For example, the correlation coefficients between *gender* and *TP1*/*TP2* were only approximately 0.16.

When we analyze the correlation ratios, we observe that most of dependencies are quite linear. The most non-linear dependencies hold between attribute pairs, where the first attribute belongs to *Prog.1* and second one to *Prog.2*. Especially, attribute A has only weak correlation with *Prog.2* attributes, but the non-linear dependencies are very strong. This suggests that managing the basic programming structures (A) is important prerequisite for all other skills. Dependency between C and $TP2$ is

also strongly non-linear. This suggest that we could use it as a background variable for predicting the *Prog.2* final results, but not in a linear model.

When we compare correlation ratios $\eta(X, Y)$ and $\eta(Y, X)$, we observe that most non-linear dependencies follow a temporal order. *Prog.2* attributes are more dependent on *Prog.1* attributes than vice versa. This is uniform with our assumption that *Prog.1* skills are prerequisites for *Prog.2*. It should be further researched if this condition holds generally, and correlation ratios identify prerequisite dependencies between the course topics.

5.3 Linear regression

The natural counterpart of correlation in predictive modelling is *linear regression* (see e.g. [MA03][ch. 11-12],[HMS02][368-390][Sta04]). The idea of linear regression is to find the best linear model (function), which can relate the dependent variable Y to various explanatory variables $X_1, ..., X_k$. The explanatory variables are called *independent variables*, because they should be linearly independent from each other. In the simplest case, Y depends linearly on just one independent variable X. When there are more variables, the model is called *multiple linear regression*. Multiple linear regression is also a useful descriptive technique, because it generalizes the correlation coefficient for multiple variables, and reveals the relative importances of individual explanatory variables.

In the following, we will first give the basic definitions. Then we will discuss about the restrictions (inductive bias) of linear regression. Finally, we will demonstrate multiple linear regression by the *ViSCoS* data.

5.3.1 Multiple linear regression model

The main idea of multiple linear regression is to model target variable Y as a linear function of explanatory variables $X_1, ..., X_k$. In real data sets this function holds seldom, because the data set often contains inconsistent data points t_1, t_2 for which $t_1[X] = t_2[X]$, but $t_1[Y] \neq t_2[Y]$. However, if the linear dependency holds approximately, we can predict the expected value of Y at point $X_1 = x_1, ..., X_k = x_k$, $\hat{y}$, quite accurately.

Definition 17 (Multiple linear regression) *Let $X = \{X_1, ..., X_k\}$ and Y be numerical variables, where Y is linearly dependent on X_is and X_is are linearly*

Table 5.2: Correlation coefficients $r = corr(X, Y)$, correlation ratios $\eta(X, Y)$, $\eta(Y, X)$, and the differences $\eta - |r|$ for all variable pairs (X, Y) in the *ViSCoS* data. The strongest correlations (significant at level $p = 0.001$) and non-linearities ($\eta - |r| > 0.450$) have been emphasized. The first six rows have been calculated from the *Prog.1* data and the rest from the *Prog.2* data.

(X, Y)	$r = corr(X, Y)$	$\eta(X, Y)$	$\eta(X, Y) - \|r\|$	$\eta(Y, X)$	$\eta(Y, X) - \|r\|$
(A,B)	**0.528**	0.836	0.308	0.735	0.207
(A,C)	0.354	0.829	**0.475**	0.644	0.290
(A,TP1)	**0.602**	0.850	0.248	0.852	0.250
(B,C)	**0.667**	0.847	0.180	0.852	0.185
(B,TP1)	**0.750**	0.857	0.107	0.917	0.167
(C,TP1)	**0.683**	0.819	0.136	0.828	0.144
(A,D)	0.164	0.856	**0.692**	0.700	**0.536**
(A,E)	0.240	0.783	**0.542**	0.754	**0.514**
(A,F)	0.194	0.805	**0.611**	0.508	0.314
(A,TP2)	0.236	0.798	**0.563**	0.700	**0.465**
(B,D)	**0.418**	0.733	0.315	0.691	0.272
(B,E)	**0.630**	0.832	0.202	0.812	0.182
(B,F)	0.378	0.782	0.403	0.622	0.243
(B,TP2)	**0.462**	0.755	0.293	0.750	0.288
(C,D)	**0.527**	0.832	0.305	0.815	0.288
(C,E)	**0.442**	0.830	0.388	0.859	0.417
(C,F)	0.262	0.702	0.440	0.480	0.218
(C,TP2)	0.304	0.772	**0.468**	0.840	**0.535**
(TP1,D)	**0.412**	0.771	0.358	0.792	0.380
(TP1,E)	**0.453**	0.797	0.344	0.752	0.299
(TP1,F)	0.335	0.748	0.414	0.563	0.228
(TP1,TP2)	**0.562**	0.805	0.243	0.742	0.181
(D,E)	**0.591**	0.805	0.214	0.859	0.268
(D,F)	**0.421**	0.706	0.286	0.676	0.255
(D,TP2)	**0.597**	0.801	0.205	0.879	0.282
(E,F)	**0.729**	0.897	0.168	0.848	0.120
(E,TP2)	**0.732**	0.905	0.173	0.924	0.192
(F,TP2)	**0.610**	0.862	0.252	0.925	0.315

independent from each other. Then for all $\overline{x} = (x_1, ..., x_k) \in Dom(X)$ *the expected value of* Y *given* $X = \overline{x}$, $\hat{y}$, *is defined by linear equation*

$$\hat{y} = \alpha + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k,$$

in which α *and* $\beta_1, ..., \beta_k$ *are real-valued regression coefficients.*

An important note is that the multiple linear regression model does not reveal the linear depency, but merely defines the form of the linear dependency, if such exists. It is easy to fit a linear equation to any data, but the validity of such a model depends on whether the relationship is actually linear.

The regression coefficients are usually determined by the *least squares method* [MA03][448-457]. The idea is to solve the equations

$$\begin{array}{ll} y_1 = \alpha + \beta_1 x_{11} + \beta_2 x_{12} + & ... + \beta_k x_{1k} + \epsilon_1 \\ y_2 = \alpha + \beta_1 x_{21} + \beta_2 x_{22} + & ... + \beta_k x_{2k} + \epsilon_2 \\ & ... \\ & ... \\ & ... \\ y_n = \alpha + \beta_1 x_{n1} + \beta_2 x_{n2} + & ... + \beta_k x_{nk} + \epsilon_n, \end{array}$$

given data rows $(x_{i1}, ..., x_{ik}, y_i) \in r$, such that the sum of squared errors, $\sum_i^n \epsilon_i$, is minimal.

Now the linear tendency can be estimated by the square of *multiple correlation coefficient* [MA03][475]

$$r^2 = \frac{Var(Y) - MSE}{Var(Y)},$$

where $MSE = \frac{SSE}{n}$ is the mean squared sum of errors. r^2 tells how much of Y's variance is explained by the linear model. The nearer r^2 is to 1, the more linear the dependency is. Often this is reported as percents r^2 *100%, which is called the *coefficient of determination.*

To analyze the validity of a linear regression model, we should analyze the errors or *reciduals* ϵ_i. Since the data usually contains several data points which share their X values, we should analyze variables E_i,

$$Dom(E_i) = \{e_j \mid e_j = y_j - \alpha + \beta_1 x_{i1} + \beta_2 x_{i2} + ...\beta_k x_{ik}\}$$

for all $\overline{x_i} \in Dom(X)$.

The linear model is appropriate, if E_is are mutually independent, normally distributed, have zero mean, and the same variance. These properties are violated, if the data does not have linear tendency or contains outliers. The easiest way to check these properties is to plot the residuals and check that the residuals scatter symmetrically around zero and are in the same scale. If we were careful, we should also check the normality by statistical tests.

Finally, we should check how well the given model fits the data. The most common way to validate a linear regression model is the *F-test*. The idea is to measure how large proportion of the prediction error (*SSE*) can be explained by the linear model. The larger the F-value, the better the linear model describes the data. A more informative measure is the significance of the observed F-value. It tells the probability that the observed linear model could have occured by chance, and the smaller the probability the more certain we can be that the observerd pattern is real.

In addition, several other significance tests can be applied to linear regression. If the model is used for describing relationships, it is best to validate the significance of coefficients separately. We can also calculate a *confidence interval*, which contains the true line with given p % probability, or a *prediction interval*, which contains p % of data points. If we want to estimate the real prediction accuracy, then it is recommended to calculate the standard error in the predicted Y value for a test set or using cross-validation.

5.3.2 Restrictions and extensions

Multiple linear regression is a good candidate for modelling numeric educational data, because it does not require large data sets. However, it is also easy to misuse, if we are not aware of the inductive bias – the underlying assumptions concerning the data.

The main assumption is that the data has a linear tendency. In practice, small deviations from linearity are not serious, but strong non-linearity can lead to errouneous models. Especially, if the data contains large "gaps", we have no guarantees that the data would follow linear tendency in those areas. If the gaps are small, we can usually interpolate the curve between sample points, but extrapolating values outside the sample is always risky. This concerns especially the educational data, which is often sparse and skewed.

If the relationship between the target variable and explanatory variables is clearly non-linear, we can try general regression models. In such models we can can allow higher order terms and interactions between components X_i. The problem is that

the resulting model is often too complex and overfits to small data sets like those in the educational domain. In addition, it is very hard to decide the appropriate form of the non-linear function, if the data has a larger dimensionality than two.

The second assumption is that the data does not contain outliers. Linear regression model – as well as all models based on SSE score function – is especially sensitive to outliers (see e.g. [Hub81][162]). The outliers are not necessarily a problem, if they happen to lie on the linear curve, but very far from other points. In this case it is a matter of interpretation whether they are really outliers or whether there is just a gap in the data. The real outliers can corrupt the whole model, because the large values (deviations to any direction) dominate the SSE score function. Sometimes it is suggested to delete all outliers or at least all zero values, which are often special cases. However, a better approach is to smooth their influence. For example, in *robust* or *biweight regression* [Hub64] the data points have different weights in SSE, according to their deviation from the current predictions. The new deviations are calculated and the model is updated, until it does not improve any more. As an alternative, we suggest to use other, less sensitive distance functions like L_1 metric for calculating SSE. These alternatives are discussed in Chapter 6.

On the other hand, linear regression model can be used for detecting outliers. We have designed a simple technique for this purpose. For each data point $(\overline{x_i}, y_i)$ we test

$$OL = \frac{{e_i}^2 - MSE}{MSE} \geq min_o,$$

where min_o is a user-defined threshold. Because the outliers are always data dependent, we cannot give any absolute values for min_o. In practice, the threshold should be defined by trial-and-error. For example, in the *ViSCoS* data, we have achieved good results with threshold $min_o = 3$ in both *Prog.1* and *Prog.2* data sets.

The third assumption is that the independent variables are linearly independent. In the educational data we often have the problem that the independent variables are correlated. This phenomenon is called *collinearity* (see e.g. [Far02][117-120]). It is an especially serious problem when a linear regression model is used for descriptive modelling, but also the prediction accuracy can suffer for collinearity. As a result of collinearity, the estimated regression coefficients β_i become imprecise and unreasonable. The model cannot differentiate the influence of correlated variables on dependent variable, but their magnitudes and signs are rather arbitrary. In the extreme case, the coefficients are so unstable that they cannot be estimated at all.

Relatively small correlations are not necessary harmful for prediction purposes. The smaller the data set and the more noise in Y values, the bigger influence collinearity

has. In addition, the influence on the prediction accuracy is stronger at points which are far from the training set. The problem is to identify when collinearity is serious. The simplest way to detect collinearity is correlation analysis, but it is not very informative. As a rule of thumb, it is sometimes (e.g. [Xyc06]) recommended to consider collinearity harmful, if for any two independent variables X_1, X_2 and dependent variable Y holds $corr(X_1, X_2) > 0.9$ or $corr(X_1, X_2) > corr(X_1, Y)$ or $corr(X_1, X_2) > corr(X_2, Y)$.

Principal component analysis is an efficient way to detect and also remedy collinearity. Low eigenvalues (lengths of principal components) and big differences in eigenvalues usually indicate harmful collinearity [Far02, Xyc06]. Principal component analysis can give advice, which variables are spurious and should be removed. Correlations can also be removed by replacing some or all of the attributes by their principal components, which are orthogonal and thus uncorrelated. This is also the underlying idea of *ridge regression* [LS72].

5.3.3 Multiple linear regression models for *ViSCoS* data

In the correlation analysis we found several linear dependencies between the course variables. These dependencies suggested that total points $TP1$ and $TP2$ can be predicted by linear regression from other variables. To test this assumption, we constructed several multiple linear regression models from the *ViSCoS* data by *Gnumeric* program. The best models are described in Table 5.3. We tried also to predict $TP2$ using A, B and C instead of $TP1$, but the results were much poorer, equivalent to using just *Prog.2* variables for the prediction.

The quality of models is analyzed by three statistical measures: the square of multiple correlation coefficient, r^2, the significance of F-value, and the standard error. According to r^2 values, the best model is $TP1 = f(A, B, C)$. However, all models but $TP2 = f(TP1, D)$ received reasonably high r^2 values, which indicates that linear models fit quite well our data. The significance of F-value gave similar results, as expected, and confirmed that all models are statistically significant, i.e. all of them have very small probability to have occured by chance.

The standard error in predicted Y describes the average error in the predictions. The average errors are 6-8 points, which means that we cannot predict the exact points accurately even in the training set. However, when the goal is to predict only the final results ($FR = 1$, if $TP \geq 15$, and $FR = 0$, otherwise), we can achieve much better accuracy. To test the classification accuracy on new data points, we performed 10-fold cross-validation and calculated the general classification rate. In *Prog.1* course, we could predict the finals results with nearly 90% accuracy, but

Table 5.3: Multiple Linear regression models for predicting the total points in the *Prog.1* and *Prog.2* courses. The regression coefficients are represented only with two decimal precision. The first two models are learnt from the *Prog.1* data and last three from the *Prog.2* data. For each model we report the square of multiple correlation coefficient (r^2), the significance of F-value (F sign.), and the standard error (sterr). The last column (crate) tells the classification rate in 10-fold cross-validation, when the model was used to predict final results.

Model	r^2	F sign.	sterr	crate
$TP1 = 1.29A + 1.46B - 5.19$	0.62	2.1 e-26	7.18	0.865
$TP1 = 1.28A + 0.93B + 1.08C - 5.26$	0.68	7.6 e-30	6.62	0.896
$TP2 = 0.82TP1 + 1.43D - 16.02$	0.48	1.8 e-8	8.39	0.807
$TP2 = 0.55TP1 + 0.64D + 0.89E - 11.67$	0.63	1.8 e-14	7.13	0.795
$TP2 = 0.54TP1 + 0.65D + 0.68E + 0.72F - 11.53$	0.64	2.9 e-15	7.05	0.795

in *Prog.2*, we achieved only 80% accuracy. An interesting phenomenon was that knowing F attribute did not increase the classification accuracy. This is explained by the strong collinearity between E and F attributes ($r^2 = 0.729$). This collinearity is observable in the other validation measures, too. It indicates that the regression coefficients of model $TP2 = f(TP1, D, E, F)$ do not suit for descriptive analysis.

Finally, we searched outliers in both *Prog.1* and *Prog.2* data sets with our own method. The measure for outliers was

$$OL = \frac{(t[TP] - t[PRED])^2 - MSE}{MSE} \geq min_o,$$

where $t[PRED]$ is the predicted total points. In *Prog.1* we used linear regression model $TP1 = f(A, B, C)$ and in *Prog.2* model $TP2 = f(TP1, D, E, F)$. With $min_o = 3.0$, we found in *Prog.1* seven outliers, and in *Prog.2* four outliers. The discovered outliers are reported in Table 5.4. We observe that the discovered outliers with high OL values have clear interpretation: seven of them are students, who have achieved a lot of task points but still failed, and two of them are students, who have gained only a little of task points, but still performed very well. Only three of the discovered students, with smallest OL values, did not have any clear interpretetion. This suggests that the used min_o value was too small. With smaller min_o values we did not find any clearly interpretable outliers.

This analysis revealed the most problematic group of students in the *Programming* courses. We recall that the main goal of our research was to predict students, who

Table 5.4: The discovered outliers in the *Prog.1* and *Prog.2* data. For each outlier, we report the attribute values, OL measure, and interpretation. In interpretation, we have divided the outliers into three types: "Surprising fail" means that the student has achieved a lot of task points but still failed the course; "Surprising success" means that the student has achieved only a little of task points, but still succeeded; "Not a clear outlier" means that the point has no clear interpretation.

Data point	OL	Interpretation
$A = 11.6, B = 9.0, C = 3.0, TP1 = 3.0$	6.80	Surprising fail.
$A = 12.0, B = 4.5, C = 0.0, TP1 = 29.5$	4.44	Surprising success.
$A = 9.4, B = 7.0, C = 3.0, TP1 = 31.0$	3.90	Not a clear outlier.
$A = 10.5, B = 8.8, C = 1.4, TP1 = 3.0$	4.19	Surprising fail.
$A = 12.0, B = 0.0, C = 0.0, TP1 = 28.0$	6.52	Surprising success.
$A = 9.5, B = 10.3, C = 0.0, TP1 = 2.0$	3.92	Not a clear outlier.
$A = 11.8, B = 11.3, C = 0.0, TP1 = 2.5$	6.44	Surprising fail.
$TP1 = 29.5, D = 9.0, E = 17.8, F = 7.0, TP2 = 4.0$	10.65	Surprising fail.
$TP1 = 32.0, D = 4.5, E = 5.5, F = 4.0, TP2 = 1.5$	3.11	Surprising fail.
$TP1 = 28.5, D = 12.0, E = 8.0, F = 0.0, TP2 = 2.0$	3.87	Surprising fail.
$TP1 = 25.8, D = 5.0, E = 7.5, F = 1.0, TP2 = 25.3$	3.05	Not a clear outlier.

either fail or drop out. Unfortunately, most of the discovered outliers fall into this category. They cannot be predicted by the multiple linear regression model, and quite likely the other modelling techniques do not perform any better. Some of the dicovered outliers are so pathological cases (a lot of points in all task categories, but failed course) that they cannot be explained by the existing attributes.

5.4 Discovering pair-wise statistical dependencies

The previous dependency detection techniques apply only for numeric data. Now we introduce techniques for discovering dependencies in categorial data. The techniques themselves can be generalized to numeric data, but the analysis is much simpler, if we first transform the numeric data to categorial. In the same time the data becomes denser and we can find patterns which would not appear in the sparser data.

In the following we will first introduce two alternative techniques, χ^2-independence test and mutual information, for discovering statistical dependences between two attributes. Then we introduce two general techniques which model statistical dependencies between target variable Y and a set of variables $X_1, ..., X_k$. The first

Table 5.5: A 2×2 contingency table containing frequencies of X and Y attribute values in a data set of size n.

	$Y = 1$	$Y = 0$	Σ
$X = 1$	$m(X = 1, Y = 1)$	$m(X = 1, Y = 0)$	$m(X = 1)$
$X = 0$	$m(X = 0, Y = 1)$	$m(X = 0, Y = 0)$	$m(X = 0)$
Σ	$m(Y = 1)$	$m(Y = 0)$	n

technique, association rules, detects only *partial dependencies*, i.e. dependencies between some attribute values, while the second technique, Bayesian networks, describes the whole dependency structure among attributes.

5.4.1 χ^2-independence test and mutual information

In statistics, the significant dependencies between attributes are detected by χ^2-*independence test* (see e.g. [MA03][627-633]). For binary-valued attributes the test is the following:

Definition 18 (χ^2-independence test) *Let R be a set of attributes, r a relation according to R, and $X \in R$ and $Y \in R$ two binary-valued attributes in R. Let $m(X = i, Y = j)$ ($i = 0, 1$, $j = 0, 1$) be the observed frequencies in relation r. X and Y are* statistically independent with probability $1 - p$, *if*

$$\chi^2 = \sum_{i=0}^{1} \sum_{j=0}^{1} \frac{(m(X = i, Y = j) - m(X = i)m(Y = j))^2}{m(X = i)m(Y = j)} < {\chi_p}^2(1), \tag{5.1}$$

where $\chi_p^2(1)$ is the critical χ^2 value at level p and 1 degree of freedom.

The idea of χ^2-independence test is to compare the observed frequencies of $X = i$, $Y = j$, $i, j = 0, 1$ (Table 5.5) to the expected frequencies under independence assumption. The test measure follows χ^2-distribution, and the critical values for the selected significance level can be checked from statistic tables. For example, at level $p = 0.01$ $\chi^2_{0.01}(1) = 6.63$. If condition 5.1 does not hold, then X and Y are considered dependent at significance level p.

The test can be easily generalized for non-binary attributes. The only difference is the number degrees of freedom, which is $(r - 1)(c - 1)$, if $|dom(X)| = r$ and $|dom(Y)| = c$. If the frequencies are very small, the test cannot be used and we

should use *Fisher's exact test* [Bra68], instead. As a rule of thumb, it is suggested (e.g. [MA03]) that all of the expected frequencies should be at least 5, before the χ^2-independence test can be used. When we test just two binary attributes, this condition usually holds for educational data sets, but the validity of the general independence test should always be checked.

Mutual information (see e.g. [DHS00][632]) is an information-theoretic measure for detecting dependencies. It measures how much information two variables X and Y share. If X and Y are independent, then X contains no information about Y and vice versa, and thus their mutual information is zero. In the other extreme, when X and Y are identical, all the information is shared. Otherwise mutual information measures the distance between the joint distribution and the product of the marginal distributions of X and Y.

Definition 19 (Mutual information) *Let R be a set of attributes and $X \in R$ and $Y \in R$ be two categorial attributes in R. The* mutual information *of X and Y is*

$$I(X,Y) = \Sigma_x \Sigma_y P(X=x, Y=y) \log \frac{P(X=x, Y=y)}{P(X=x)P(Y=y)}. \tag{5.2}$$

Equation 5.2 can be expressed equivalently as

$$I(X,Y) = H(X) + H(Y) - H(X,Y),$$

where $H(Z)$ is the *entropy* in Z:

$$H(Z) = -\sum_{z_i \in Dom(Z)} P(Z=z_i) \log P(Z=z_i).$$

The entropy measures uncertainty of Z and is usually counted in bits (2-based logarithm). If X and Y share all their information, the mutual information is the same as X's (or Y's) entropy: $I(X,Y) = H(X)$.

5.4.2 Pair-wise statistical dependencies in the *ViSCoS* data

In this analysis we computed the χ^2-independence measures and the mutual information measures for all attribute pairs in the *Prog.1* and *Prog.2* data. The results are presented in Tables 5.6 and 5.7. In the χ^2-independence test we used critical value $\chi^2_{0.01}(1) = 6.63$. In mutual independence test no critical values are available, but I values tell only the relative strength of dependencies. We used threshold $I \geq 0.100$ for strong mutual dependencies.

In both data sets, (*Prog.1* and *Prog.2*), we can find several significant dependencies. In *Prog.1* data, only A and C (basic programming structures and applets) can be considered statistically independent. They are also the most independent according to the mutual information measure. All the other attribute pairs have strong dependencies. Especially, B and $FR1$ are strongly dependent according to both measures, which was already observed in the correlation analysis.

Table 5.6: The dependencies in *Prog. 1* data measured by χ^2 independence test and mutual information I. The strongest dependences ($\chi^2 \geq 6.63$, $I \geq 0.100$) are emphasized.

Attr.	χ^2	I
(A,B)	**20.28**	0.**106**
(A,C)	1.18	0.005
(A,FR1)	**19.10**	**0.085**
(B,C)	**16.21**	**0.106**
(B,FR1)	**47.08**	**0.278**
(C,FR1)	**18.23**	**0.136**

In Table 5.7 we observe interesting relations between *Prog.1* and *Prog.2* variables. A is statistically independent from all *Prog.2* variables and B has only a light dependency with E. On the other hand, C is statistically dependent with all *Prog.2* variables. This indicates the importance of managing applets in *Prog.2* course which was confirmed by the course teachers. The total points in *Prog.1*, $TP1$, have strong dependencies with all but F variable in the *Prog.2* data. This suggests that *Prog.1* skills do not help in the last tasks of *Prog.2*. The tasks in the F category concerned mostly error handling, but also various extra topics.

All *Prog.2* variables are statistically dependent on each other. The strongest dependencies are between D and $FR2$, E and $FR2$, and E and F according to both measures. The last dependency between E and F (graphical applications and error handling) remained unexplained. However, it is so strong that it should be taken into account in the models, which assume independence between explanatory variables.

Table 5.7: The dependencies in *Prog. 2* data measured by χ^2-independence test and mutual information I. The strongest dependences ($\chi^2 \geq 6.63$, $I \geq 0.100$) are emphasized.

Attr.	χ^2	I
(A,D)	0.45	0.004
(A,E)	2.25	0.027
(A,F)	0.80	0.011
(A,FR2)	0.70	0.006
(B,D)	0.96	0.008
(B,E)	**8.32**	0.082
(B,F)	4.28	0.058
(B,FR2)	3.02	0.025
(C,D)	**19.99**	**0.171**
(C,E)	**10.60**	0.088
(C,F)	**8.46**	0.070
(C,FR2)	**12.10**	**0.103**
(TP1,D)	**10.26**	0.090
(TP1,E)	**9.72**	0.089
(TP1,F)	4.56	0.047
(TP1,FR2)	**27.23**	**0.243**
(D,E)	**16.31**	**0.138**
(D,F)	**8.19**	0.070
(D,FR2)	**23.02**	**0.200**
(E,F)	**20.38**	**0.178**
(E,FR2)	**32.01**	**0.296**
(F,FR2)	**13.06**	**0.130**

5.5 Discovering partial dependencies by association rules

The previous techniques modelled statistical dependencies between two attributes. Now we will introduce association rules for searching dependencies between a target attribute and a set of attributes. The discovered dependencies are only partial, which means that they concern only certain attribute values. This kind of dependencies cannot be discovered by global modelling techniques, which require that the dependency holds for all attribute values. For example, we can find association rules which hold only for the failed students or male students.

5.5.1 Association rules

Association rules (see e.g. [MT02]) are a special case of probabilistic rules of form *"90% of students who take* Data Mining *course take also* Educational Technology *course"*. If only five students of 500 had taken the *Data Mining course*, the previous rule would not have much weight. But if 400 students had performed *Data Mining course*, it would be remarkable that 360 of them had also studied *Educational technology*. These two aspects, the commonness and the strength of the rule are defined by *frequency* and *confidence*.

Definition 20 (Association rule) *Let R be a set of categorial attributes and r a relation according to R. Let $X = \{X_1, ..., X_k\} \subseteq R$ and $Y \in R$, $Y \notin X$, be attributes defined in R. Let us denote the value combination $(X_1 = x_1), ..., (X_k = x_k)$, $x_i \in Dom(X_i)$, by $X = \overline{x}$. Then the confidence, cf, of rule $(X = \overline{x}) \Rightarrow (Y = y)$ is*

$$cf(X = \overline{x} \Rightarrow Y = y) = \frac{P(X = \overline{x}, Y = y)}{P(X = \overline{x})} = P(Y = y | X = \overline{x})$$

and the frequency, fr, of the rule is

$$fr(X = \overline{x} \Rightarrow Y = y) = P(X = \overline{x}, Y = y).$$

Given the user-defined thresholds $min_{cf}, min_{fr} \in [0, 1]$, rule $(X = \overline{x}) \Rightarrow (Y = y)$ is an association rule in r, if

(i) $cf(X = \overline{x} \Rightarrow Y = y) \geq min_{cf}$, and

(ii) $fr(X = \overline{x} \Rightarrow Y = y) \geq min_{fr}$.

The first condition requires that an association rule should be *confident* (strong enough) and the second condition requires that it should be *frequent* (common enough).

The simplest way to discover all association rules is to determine all frequent value combinations $(X_1 = x_i), .., (X_k = x_k)$, and then test for all $X_i \in X$, whether $(X_1 = x_1), ..., (X_{i-1} = x_{i-1}), (X_{i+1} = x_{i+1}), ..., (X_k = x_k) \Rightarrow (X_i = x_i)$ is confident. The search space can be pruned by observing that the frequency is monotonic property, i.e. if $X = \overline{x}$ is frequent, then also all its subsets are frequent [Toi96][15-16]. However, finding all association rules is still a time-consuming process. If all attributes $X_i \in R$, $i = 1, ..., k$, have v values, then there can be $(1+v)^k$ frequent value combinations, and from each value combination we can construct k different rules. So, for example, if $k = 8$ and $v = 3$, we can find in the worst case 65536 frequent

value combinations and we have to check 524288 rules! This is still computable, but when the number of attributes and the domain sizes increase, the problem becomes intractable. That is why most applications concentrate on finding rules without negations in binary-valued data. In the educational domain the negations are also important. For example, in the *ViSCoS* project we are interested in dependencies which describe failed or dropped students. Fortunately, the data sets are typically small, which means that in practice we also have to select attributes and reduce the attribute domains.

An important question in association rules is how to define *interesting* rules. Interestingness is a very subjective property, and there is no consensus how to define it. Intuitively, an interesting or important rule should have a high frequency and high confidence, but it is difficult to decide how they should be weighed.

Some researchers in educational technology ([SAGCBR+04, VC03]) report only the rule confidence. The problem in this approach is that rule $X \Rightarrow Y$ has always high confidence, if Y has high frequency. In an extreme case, when X and Y are statistically independent, $cf(X \Rightarrow Y) = fr(Y)$. For example, if 450 from 500 students have taken *Educational technology* course and 5 of the remaining 50 students have taken *Signal processing* course, then the confidence of rule "*Students who study Signal processing, study also Educational technology*" is 0.90, even if there is no dependence. That is why we cannot ignore the frequency.

Table 5.8: Measures for ranking association rules, given their frequency and confidence. For simplicity, all measures have been defined for binary-valued attributes. Abbreviation $X = \overline{x}$ corresponds to value assignment $(X_1 = x_1), ..., (X_k = x_k)$.

Measure	Definition
I-measure	$I = fr(X \Rightarrow Y) \log \frac{cf(X \Rightarrow Y)}{fr(X)fr(Y)}$
J-measure	$J = fr(X \Rightarrow Y) \log \frac{cf(X \Rightarrow Y)}{fr(X)fr(Y)} + fr(X \Rightarrow \neg Y) \log \frac{cf(X \Rightarrow \neg Y)}{fr(X)fr(\neg Y)}$
χ^2-measure	$\chi^2 = \sum_{\forall \overline{x} \in Dom(X)} \sum_{y=0}^{1} \chi^2(X = \overline{x}, Y = y)$
Partial χ^2	$\chi^2(X = \overline{x}, Y = y) = \frac{(m(X=\overline{x},Y=y) - nfr(X_1=x_1)...fr(X_k=x_k)P(Y=y))^2}{nfr(X_1=x_1)...fr(X_k=x_k)P(Y=y)}$

In Table 5.8 we list four measures which take into account both frequency and confidence. For simplicity, we have defined all measures for binary-valued data, but they can be generalized for any categorial data. I-measure computes simply the mutual information of rule $X \Rightarrow Y$. J-measure ([SG92]) is the sum of the mutual information of rules $X \Rightarrow Y$ and $X \Rightarrow \neg Y$. J-measure was originally designed for classification purposes, but is sometimes used for ranking association rules (e.g. [MT02]). It measures frequency and confidence both in $X \Rightarrow Y$ and its negation,

$X \Rightarrow \neg Y$. For classification purposes this is adequate, but if the goal is to find all interesting partial dependencies, we can miss some interesting rules, if their negations are rare or unconfident.

The third measure is a generalization of the χ^2-measure for attribute sets. The χ^2-measure of a rule can be compared to the critical value at the desired level of significance. An alternative suggested in [BMS97] is to rank the rules according to their χ^2 values. The restriction of the χ^2-measure is that the expected frequencies of attribute combinations should be large enough. As a general rule of thumb, it is recommended (e.g. [MA03]) that each expected frequency should be > 1 and at least 80% of expected values should be > 5. This condition is often broken, when the antecedent of a rule, X, contains more than one attribute. In addition, the χ^2-test measures the statistical dependency between attributes $X_i \in X$ and Y, and interesting association rules can be missed, if the dependency holds only between some attribute values.

As a solution, we propose *partial* χ^2-measure, which calculates the χ^2 value only for the values which occur in the rule. Now the expected frequencies are always high, because the association rules have already been selected according to their frequency. The partial χ^2 values cannot be compared to the critical χ^2 values as such, because the contribution of other value combinations is missing. Instead, the partial χ^2 value can be compared to $\frac{\chi^2_p}{2^k}$, where $|X| = k$ and p is the desired level of significance. This value is the average χ^2 value of each attribute combination in a significant rule.

5.5.2 Association rules in the *ViSCoS* data

For the discovery of association rules we used the binary-valued versions of the *ViSCoS* data. In Table 5.10 we have listed all frequent rules for predicting the final results of *Prog.1* and *Prog.2* courses. The minimum frequency thresholds were defined according to partial χ^2 measure to catch only those rules which are statistically significant at level 0.01 (i.e. there is 1% probability that the association rule could have occured by chance). The corresponding min_{fr} values are given in Table 5.9. For the minimum confidence threshold we used value $min_{cf} = 0.7$.

The most interesting rule is $TP1 = lot \Rightarrow FR2 = 1$, which means that we can predict the *Prog.2* final results quite well, when we only know the *Prog.1* total points. The contrary rule $TP1 = little \Rightarrow FR2 = 0$ is even stronger, with confidence $cf = 0.91$, which means that nearly all students who have got only a little of points in *Prog.1*, have failed in *Prog.2*. However, this rule is very rare (only $fr = 0.227$), and it cannot be considered as statistically significant in the χ^2 sense. The same

Table 5.9: min_{fr} values for association rules in the *Prog.1* ($n = 125$) and *Prog.2* ($n = 88$) data sets. The thresholds are calculated according to χ^2 values at 0.01 significance level. $k = |X, Y|$ tells the number of attributes in rule $X \Rightarrow Y$.

k	$n = 125$	$n = 88$
2	0.35	0.37
3	0.20	0.22
4	0.12	0.13
5	0.07	0.08
6	0.04	0.05

problem concerns rule $C = lot \Rightarrow FR2 = 1$ ($fr = 0.330$, $cf = 0.784$). Rule $B = lot \Rightarrow FR2 = 1$ ($fr = 0.511$, $cf = 0.608$) is relatively frequent, but the confidence is relatively low.

Most of the rules concern students who have passed the course. The reason is that the number of failed students was much smaller, and the rules concerning them are not frequent. In the *Prog.1* data we did not find any rules concerning failed students. In *Prog.2* we found four rules. Attribute F occurs in all of them, which indicates that F attribute is important for predicting the final results in *Prog.2*.

The association rules can already be used for prediction. E.g. we can predict that students who have done a lot of A and B tasks will pass the *Prog.1* course with 92% probability. However, the frequent rules cover only some subsets of students, and for general prediction purposes we would need a global model.

5.6 Bayesian models

Association rules describe only a partial probabilistic model and thus they do not suit for general prediction purposes. Bayesian networks describe the joint probability distribution of all variables. Thus, we can predict the probability of any attribute we are interested in. In additon, they describe a dependency structure of all variables in one model.

In the following we will describe the main idea of Bayesian networks and construct a couple of example models from the *ViSCoS* data. We will return to Bayesian networks in Chapter 7, when they are used for classification.

Table 5.10: The strongest frequent rules, and their frequencies and confidences for predicting final results for *Prog.1* and *Prog.2*. The first five rules have been calculated from the *Prog.1* data and the rest from *Prog.2* data. The min_{fr} values were selected according to partial χ^2-test to guarantee significance at level 0.01, and min_{cf} was 0.7.

Rule	fr	cf
$A = lot, B = lot, C = lot \Rightarrow FR1 = 1$	0.270	1.000
$A = lot, B = lot \Rightarrow FR1 = 1$	0.582	0.922
$A = lot \Rightarrow FR1 = 1$	0.697	0.773
$B = lot \Rightarrow FR1 = 1$	0.598	0.924
$A = lot \Rightarrow B = lot$	0.631	0.713
$TP1 = lot, D = lot, E = lot, F = lot \Rightarrow FR2 = 1$	0.121	0.846
$TP1 = little, E = little, F = little \Rightarrow FR2 = 0$	0.198	0.947
$TP1 = lot, D = lot, E = lot \Rightarrow FR2 = 1$	0.275	0.926
$TP1 = lot, D = lot \Rightarrow FR2 = 1$	0.363	0.846
$TP1 = lot, E = lot \Rightarrow FR2 = 1$	0.363	0.943
$A = lot, B = lot, F = lot \Rightarrow FR2 = 1$	0.473	0.956
$B = lot, F = lot \Rightarrow FR2 = 1$	0.473	0.956
$C = littleD = little \Rightarrow E = little$	0.407	0.974
$D = little, E = little, F = little \Rightarrow FR2 = 0$	0.308	0.824
$D = little, F = little \Rightarrow FR2 = 0$	0.363	0.744
$E = little, F = little \Rightarrow FR2 = 0$	0.404	0.720
$TP1 = lot \Rightarrow FR2 = 1$	0.539	0.727

5.6.1 Bayesian network model

In *Bayesian networks* (see e.g. [Pea88] the statistical dependencies are represented visually as a graph structure. The idea is that we take into account all information about conditional independences and represent a minimal dependency structure of variables. Formally we define:

Definition 21 (Bayesian network model) *A Bayesian network model is a pair (G, θ), in which G is the structure of the network and θ is a set of parameters associated with it. G is a directed acyclic graph, in which the vertices correspond to variables in the problem field and the edges represent the dependencies between variables. If we denote parents of X by $F(X)$ and children of X by $D(X)$, then for every variable X in the network holds*

$$i)\ P(X|Y, F(X)) = P(X|F(X)) \text{ for all } Y \notin D(X), \text{ and}$$

$$ii)\ P(X|F'(X)) \neq P(X|F(X)) \text{ for all } F'(X) \subseteq F(X).$$

The set of parameters θ consists of conditional distributions of form $P(X|F(X))$, where $F(X) = \emptyset$ for all root nodes X.

The first condition states that every variable is independent from all variables which are not its children, given its parents. The second condition states that the set of parents is minimal.

The Bayesian network structure can be defined by an expert, but if the model is used for descriptive purposes, we should learn it from data. If the goal is prediction, we can utilize the expert knowledge by restricting the search space (model family). If expert knowledge about existing dependencies and independencies is uncertain or unavailable, we can collect more domain knowledge by descriptive modelling, like association rules. For example, the dependency analysis of the *ViSCoS* data revealed that $TP1$ has the strongest depenendency with $FR2$ from all *Prog.1* attributes. Thus, we can exclude the other *Prog.1* variables from *Prog.2* models.

The most common score function for Bayesian networks is *minimum description length* (*MDL*), which in the same time maximizes the log likelihood of the model, given data, and prefers simple model structures. Learning a globally optimal Bayesian network structure is generally an *NP*-hard problem [Coo90], and in practice, the learning algorithms find only a local optimum.

Often the same joint probability distribution can be represented by several alternative networks. If all arrows in a Bayesian network describe causal relationships, the model is called a *causal network*. For a human interpreter, this kind of representation is more intuitive, but when the model becomes more complex, the non-causal model may be simpler to construct. Especially, if we have k attributes $B_1, ..., B_k$ such that A and B_i are dependent for each i and B_is are mutually independent, it is much easier to estimate probabilities $P(B_i|A)$ separately than the distribution $P(A|B_1, ..., B_k)$. For example, if A and B_i are binary-valued, we have to estimate only $2k$ conditional probabilities and k prior probabilities $P(B_i)$, while the latter model requires 2^k conditional probabilities and one prior probability $P(A)$. In addition, often the data does not contain all value combinations $B_1, ..., B_k$ and we cannot estimate all probabilities from the data. In this case, the simpler and more robust non-causal model is preferrable.

In the extreme, the whole model can be represented as a two-layer network with one root node and a set of leaf nodes. This model is called *naive Bayes model* (see e.g. [DP97]), because it makes *Naive Bayes assumption* that all leaf variables $B_1, ..., B_k$ are conditionally independent, given the root variable A. In practice, this assumption holds seldom, but the model still works well. Especially in classification this kind of Naive Bayes classifiers are popular, and we will return to this topic in Chapter 7.

5.6.2 Bayesian models for the *ViSCoS* data

Figure 5.1 represents two Bayesian networks, $B1$ and $B2$, for modelling *Prog.1* and *Prog.2* course data. The associated probabilities are given in Tables 5.11 and 5.12. Both models have been learnt from data by *minimum message length* (*MML*) principle using *CAMML* software [WNO]. Before learning we have imposed some restrictions on the graph structures, based on the temporal order of variables. In $B1$ it was required that node A precedes B and B precedes C. In $B2$ all *Prog.1* variables $(B, C, TP1)$ had to precede all *Prog.2* variables $(D, E, F, FR2)$. In addition, the temporal order of variables D, E and F was given.

In model $B1$ we observe that A, B and C are not directly dependent on each other, but through node $FR2$. This is new information, which we could not obtain by analyzing binary dependencies. Another observation is that the model is nearly equivalent to naive Bayes model $NB1$ (Figure 5.2). The only difference is the direction of relation $A - FR1$, which we can always change in Bayesian models. Both models produce exactly the same predictions for all data points. They would produce different predictions only, if we did not estimate the prior probabilities $P(A)$ and $P(FR1)$ from the data but using domain knowledge. For example, if

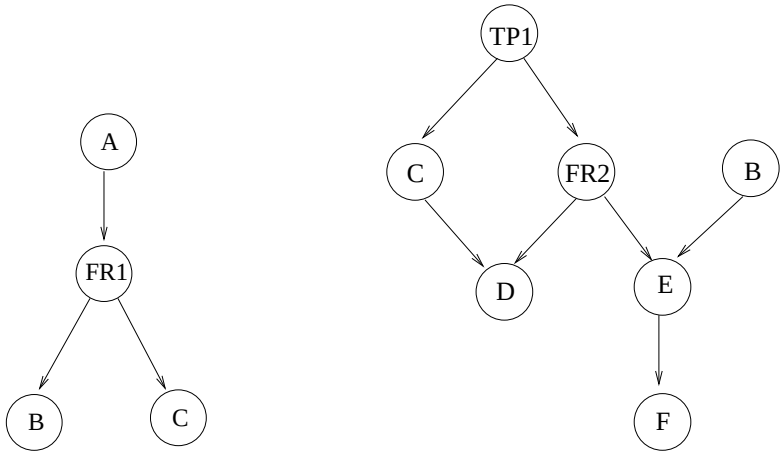

Figure 5.1: Bayesian networks $B1$ (left) and $B2$ (right) for modelling *Prog.1* and *Prog.2* course data. The model structures have been learnt by *MML* principle, after imposing some temporal order restrictions on variables.

Table 5.11: Probabilities for Bayesian network $B1$ modelling *Prog.1* course data.

$P(A)$	0.87698
$P(FR1\|A)$	0.77027
$P(FR1\|\neg A)$	0.28125
$P(B\|FR1)$	0.81667
$P(B\|\neg FR1)$	0.17568
$P(C\|FR1)$	0.04054
$P(C\|\neg FR1)$	0.41667

the teachers know that the current student group is better than in previous years (but the course is otherwise the same), they can give higher prior probability for $P(FR1)$ in $NB1$ or $P(A)$ in $B1$. However, in $B1$ the prior estimate has no influence on predictions, after we know A. In $NB1$ model the domain knowledge is taken into account in all predictions, because we update $P(FR1)$ three times according to evidence on A, B and C. This is clearly an advantage, when we have some domain knowledge and want to use it in predictions.

$B2$ model reveals also interesting information. The learning algorithm has dropped variable A as irrelevant. In addition, the model contains the same dependencies between C and D, B and E, E and F and $TP1$ and $FR2$, which we have already found by correlation analysis, but explains them better. When *Prog.2* course begins, we already know B, C and $TP1$, and the relation $TP1 - C$ has no influence.

Table 5.12: Probabilities for Bayesian network $B2$ modelling *Prog.2* course data.

$P(TP1)$	0.74719
$P(B)$	0.83708
$P(C\|TP1)$	0.54478
$P(C\|\neg TP1)$	0.06522
$P(FR2\|TP1)$	0.72388
$P(FR2\|\neg TP1)$	0.10870
$P(E\|FR2, B)$	0.75000
$P(E\|FR2, \neg B)$	0.08333
$P(E\|\neg FR2, B)$	0.08333
$P(E\|\neg FR2, \neg B)$	0.15000
$P(D\|FR2, C)$	0.85000
$P(D\|FR2, \neg C)$	0.47727
$P(D\|\neg FR2, C)$	0.38889
$P(D\|\neg FR2, \neg C)$	0.14516
$P(F\|E)$	0.43421
$P(F\|\neg E)$	0.04808

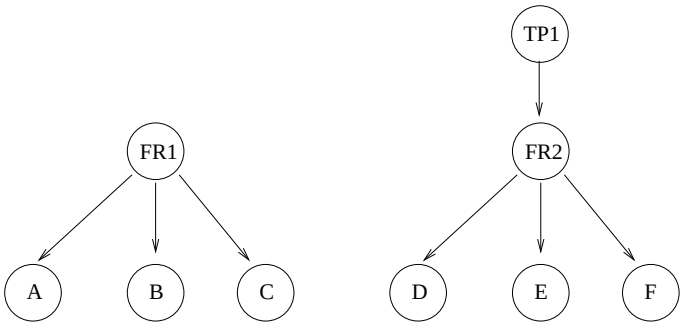

Figure 5.2: Naive Bayes models $NB1$ and $NB2$ for modelling *Prog.1* and *Prog.2* course data. In $NB2$ we have used $TP1$ as a background variable, which determines the prior probability of $FR2$.

However, B and C are taken into account, when we update $FR2$ in the light of D and E. Compared to naive Bayes model $NB2$ (Figure 5.2), the model contains more information, but it has a more complex structure, which increases the risk of overfitting and poor generalization.

5.7 Comparison and discussion

In the previous we have described several techniques for modelling dependencies between attributes. The main techniques for numeric attributes are (multiple) correlation analysis and linear regression, and for categorial attributes association rules and Bayesian networks. Correlation analysis and association rules are descriptive techniques and linear regression and Bayesian networks are primarly predictive techniques.

Now we will give a deeper analysis of the relations between these four paradigms and give some guidelines for selecting the most appropriate modelling technique for a given problem in the educational domain. The final selection depends on two things: the purpose of modelling (what kinds of relationships we want to model) and the properties of data.

5.7.1 Modelled relations

In the basic form all four techniques model binary relationships between two variables. Multiple linear regression, association rules and Bayesian networks define also relationships between one variable Y and a set of other variables $X_1, ..., X_k$. The Pearson correlation coefficient is defined only for two variables, but we can calculate the multiple correlation coefficient between one variable and a set of variables indirectly, from a multiple linear regression model.

Correlation differs from the others also in another aspect: it describes a symmetric relation between two variables, while all the others describe non-symmetric relations. However, we can often conclude the opposite model $Y \rightarrow X$ from $X \rightarrow Y$. In simple linear regression we can define the inverse function $x = f^{-1}(y) = \frac{y-\alpha}{\beta}$ from $y = f(x) = \alpha + \beta x$, if coefficient $\beta \neq 0$. In Bayesian networks, we can always change the direction of an arrow, calculate new probabilities, and the model remains the same. However, we cannot change conjunctive rule $X_1, ..., X_k \rightarrow Y$ to a set of binary rules $Y \rightarrow X_i$, unless X_is are conditionally independent given Y. In association rules the direction is critical. Both $X \Rightarrow Y$ and $Y \Rightarrow X$ have the same frequency $P(X, Y)$, but the confidences depend on different factors, in the first rule on $P(X)$, and in the second on $P(Y)$. It is still quite possible that both rules are confident. Sometimes we can find confident rules which seem to be contradictory. For example, let us consider the following rules:

"*If the student's major is computer science, then s/he has not passed course x.*"
"*If the student has passed course x, then her/his major is computer science.*"

In fact, these rules just tell that very few students have passed course x, but most of them are computer science students. There is no reason why both $P(\neg X|Y)$ and $P(Y|X)$ could not be high simultaneously.

We have already observed that association rules and probabilistic models have higher representational power than correlation and linear regression. Correlation and linear regression can describe only linear dependencies, while conditional probabilities reveal general dependencies. This means that we can find rule $X \Rightarrow Y$ with 1.0 confidence, even if correlation $corr(X, Y) = 0$. Thus, correlation analysis can reveal only dependencies but not independences. Only when the joint distribution $P(X, Y)$ is normally distributed, the zero correlation imposes independence.

The general regression models can in principle model any kinds of dependencies. The problem is to find the form of the function needed. If we knew the joint density function $f(x, y)$, then we could derive the regression function $y = f(x)$ [MA03][176]. In practice, these density functions are seldom known, and we have to approximate the regression function from the data. On the other hand, we could also derive the probability function from a regression model. *Logistic regression* (e.g. [HL89]) defines the probabilities $P(Y = y)$, when X_is are known. When Y is a binary-valued variable, the function is

$$P(Y = 1) = \frac{1}{1 + e^{b_0 + b_1 x_1 + b_2 x_2 + ... + b_n x_n}}.$$

This is in fact a non-linear transformation of the multiple linear regression model:

$$\log \frac{P(Y = 1)}{P(Y = 0)} = b_0 + b_1 x_1 + ..b_n x_n.$$

This model is sometimes used in computerized adaptive testing (e.g. [DP05]).

5.7.2 Properties of data

An important aspect for selecting the modelling paradigm is the type of the data we have. Pearson correlation and linear regression have been defined only for numeric (either discrete or continuous) data. Other correlation measures like Spearman's ρ or Kendall's τ can be used for ordinal categorial data. On the other hand, association rules are defined only for categorial data. If the original data is numeric, we have to discretize it to appropriate value intervals (see e.g. [RS01, SA96]). The Bayesian models are originally defined for any discrete data, but they can be generalized for continuous data (see e.g. [Neo04][ch 4]). However, learning a model with density functions is a much more difficult task.

There are also other properties of data which we should consider. In the educational data it is typical that all variables are dependent on each other, the data contains several outliers, and there are gaps in the data. Let us first consider the problem of mutually dependent X_i variables. In correlation analysis this is not any problem, because all binary relationships are studied separately. However, in linear regression collinearity – linear dependencies between X_is – can be a serious problem. It ruins the descriptive power of the model and can affect also the predictions. In the general Bayesian models all statistical dependencies are implicitely described in the model, and the only disadvantage of several dependencies is that the model becomes more complex. This means that we need more data to define all parameters accurately and the reasoning becomes more complex. A more serious problem arises in naive Bayes model, where we expect that all variables X_i are conditionally independent, given Y. The more X_is are dependent, the more erroneous results the model yields. For association rules these dependencies constitute no problem.

Outliers are especially problematic, when we measure distances between numeric data points. Outliers can corrupt the correlations and produce either too weak or too strong correlation coefficients. Linear regression is especially sensitive to outliers and relatively small number of outliers can ruin the whole model. The resulting model is inaccurate both for descriptive and predictive purposes. Bayesian models and association rules are more immune to outliers. This is partly due to use of categorial data, but the more important reason is the way of measuring dependencies. An outlier, $t = [x_1, ..., x_k, y]$, affects only one parameter, $P(Y = y|X = x_i, ..., X_k = x_k)$, and not the whole model. In addition, the influence is relatively small – proportional to the number of other data points having the same values $X_1 = x_1, ..., X_k = x_k$. Very often, none of the "normal" points share the same values, and the outlier has no effect on predicting other, "normally behaving" data points. The most serious case is when we should determine a parameter from a few normal values and an outlier. In association rules these rare patterns would not be represented at all, and outliers have no influence on results. On the other hand, searching exceptionally rare rules can reveal outliers.

Gaps in data are not necessarily any problem, if the data is representative and we can expect that none of the future points will fall into gaps. However, in the educational domain we cannot be so optimistic. More often, the gaps are a sign of unknown areas, where we cannot make any conclusions. In addition, when gaps are occupied, they tend to be occupied by outliers, and the problem becomes only worse. In correlation analysis and linear regression the problem is that we cannot even expect the relationship to be linear in those areas. In linear regression we can interpolate across gaps, but it is risky and the validation measures become unreliable. In Bayesian models we cannot interpolate probabilities for missing attribute values. The common approach is to assign equal distribution for unknown values, but it

can be seriously misleading.

5.7.3 Selecting the most appropriate method

We have summarized the properties of the four methods in Table 5.13. In addition to modelling purpose and data properties we have considered the efficiency of model construction. Correlation analysis and linear regression are very fast techniques, while the other two are much slower. Searching all possible association rules in a large data set can be intractable, but fortunately the educational data sets are quite small. Learning the Bayesian network structure is an *NP*-hard problem, but usually we can restrict the search space by domain knowledge, and the parameters can be estimated fast.

Table 5.13: Comparison of correlation analysis, linear regression, association rules and Bayesian networks.

	Corr.	Lin.reg.	Ass.	Bayes
Purpose	descr.	pred.	descr.	pred.
Type of dependency	linear	linear	statistical	statistical
Data type	numeric	numeric	categorial	any
Dependencies between explanatory variables	no effect	serious	no effect	harmful in naive Bayes
Influence of outliers	sensitive	very sensitive	no influence	quite robust
Model construction	fast	fast	slow	parameters fast, structure slow

Chapter 6

Clustering

In this and the next chapter we will consider two ways to group data, namely *clustering* and *classification*. The idea in both is similar, and sometimes clustering is called as *unsupervised classification*, i.e. classification when the class variable is not known. Clustering is a representative of descriptive modelling, in which we try to find groupings which occur naturally in the data. On the other hand, classification is predictive modelling, in which we give a pre-defined grouping and try to predict the group of a new data point.

While clustering has importance for its own sake, it can also guide the classifier construction. For example, in the *ViSCoS* project we would like to discover whether students or exercise tasks constitute any groups. However, the ultimate goal is to learn a classifier which would predict the course outcomes based on other attributes. Thus, a natural goal is to search a clustering of students to two clusters based on exercise data and check whether the clusters correlate with the course outcomes. If we cluster the data in different attribute spaces, the best clustering reveals the most relevant feature combinations for the prediction task. Thus, clustering can be used for feature extraction and selection.

In the educational domain the data imposes several restrictions and criteria for a good clustering method. Educational data is often skewed and contains several outliers. Thus, we cannot suppose any specific cluster form or distribution. The outliers should not blur the real clusters, but instead it is desirable that the method could reveal the outliers as a side-product. Deciding the number of clusters is also difficult, because we do not know beforehand how many clusters should be reserved for the outliers. Clear, well-separated clusters are always ideal, but in practice we expect that the student groups will be overlapping. In this case, we would also like to measure the relative strength of cluster membership, $Member(p, c)$, for data

point p and cluster c.

In the following, we will first define the clustering problem and the related distance or similarity measures. Then we introduce the main approaches for clustering. All methods are illustrated by clusters discovered in 2-dimensional *ViSCoS* data. We analyze the suitability of the existing clustering methods for the educational domain and give guidelines how to cluster educational data. Finally, we report our experiment on clustering the whole *ViSCoS* data.

6.1 Clustering problem

The main problem of clustering is how to define a cluster. The intuition is that we would like to get dense, well-separated groups, in which the data points are similar to each other and different from the data points in the other groups. If the data is metric, the goal is to find dense regions in the attribute space, which are separated by sparser areas. However, "similarity" or "closeness" are very subjective measures and the exact definition depends on the selected features and the modelling purpose. For example, let us consider how we could group a cat, an owl, and a hen. If we are interested in the biological relations, a natural grouping is to put the owl and the hen into the class of birds and the cat into the class of mammals. On the other hand, the cat and the hen are domestic animals, but we could also group the cat and the owl together as predators.

In the following, we give a general definition of the clustering problem:

Definition 22 (Clustering problem) *Let $D = \{p_1, ..., p_n\}$ be a set of data and $C = \{c_1, ..., c_k\}$ a set of clusters. Let $sim : S \times S \rightarrow \mathbb{R}^+ \cup \{0\}$ be a similarity measure such that $sim(p_1, p_2)$ measures the similarity between p_1 and p_2 for all $p_1, p_2 \in S$. Let $member : S \times C \rightarrow [0, 1]$ be a function such that $member(p, c)$ measures the strength of membership of point p in cluster c.*

The clustering problem is to define the set of clusters, $C = \{c_1, ..., c_k\}$, such that

i) for all $p \in D$ there is a cluster $c \in C$ such that $member(p, c)$ is maximal across C,

and for all $c_i \in C$ holds

ii) $sim(p_1, p_2)$ is high for all $p_1 \in c_i$ and $p_2 \in c_i$, and
iii) $sim(p_1, p_2)$ is low for all $p_1 \in c_i$ and $p_2 \notin c_i$.

If $member(p, c) = 1$ for exactly one $c \in C$ and $member(p, c') = 0$ for all $c' \neq c$, the clustering is called strict *or* partitive. *Otherwise, it is called* soft *or* non-partitive.

Condition i) defines for each data point the best cluster or clusters, condition ii) measures the *inter-cluster similarity* and condition iii) the *intra-cluster dissimilarity*.

The most important decision is how to define a cluster and select suitable measure *sim* for similarity. When the measure has been selected, we just have to maximize the inter-cluster similarity and/or maximize the intra-cluster dissimilarity in the data. Unfortunately, the optimal solution is *NP*-hard [GJ79] and intractable even for small sets of educational data[1]. Thus, in practice, we have to be satisfied with an approximate solution. Several heuristic clustering algorithms have been developed to encounter this problem. The algorithms themselves contain certain inductive bias as discussed in Chapter 3, but the main source of bias is the optimized metric [EC02]. Both biasses should be taken into account when selecting the most appropriate clustering method for the educational data.

The first problem is to decide what kind of clusters we prefer. Often, we have to select between compact and well-separated clusters. If the goal is to find compact and homogeneous clusters, then the choice is to maximize intra-cluster similarity. On the other hand, if want to find well-separated clusters, then the goal is to minimize inter-cluster similarity. In practice, the most often used score function is squared sum of errors, *SSE*, which minimizes the sum of squared distances from the cluster centroids. As a result, it produces both homogeneous and well separated clusters. However, homogeneous clusters are not necessarily compact, and if we preferred compactness, we should minimize the sum of all pairwise distances between the data points in a cluster.

In addition, the representation of clusters affects on what kinds of clusters we can find. Two common choices are to represent a cluster by its centroid (central point) or by its boundary points. Centroid representation works well, when clusters are compact or isotropic, but it cannot fully represent elongated or non-isotropic clusters. On the other hand, boundary representation works well for non-isotropic clusters.

In the following, we will first describe different measures for similarity and analyze the underlying inductive biasses. After that we will introduce the main clustering methods, which can be divided into three categories[2]:

1. *Hierarchical methods* construct a hierarchy of (typically) nested clusters.
2. *Partitioning methods* try to find an optimal partitioning into a specified number of clusters.

[1] According to our experiments, computing the optimal solution with the branch-and-bound method becomes unfeasible already when we have ≥ 30 rows of 2-dimensional data and the number of clusters is ≥ 3.

[2] The taxonomy of clustering methods follows [HMS02]. Other taxonomies and comparisons of clustering methods can be found e.g. in [JMF99] and [Ber02].

3. *Probabilistic model-based clustering* tries to find the underlying probabilistic model, which has produced the data.

The first two approaches are partitive and the resulting clusters are separate, while probabilistic clustering is non-partitive and produces overlapping clusters[3]. In probabilistic methods, the cluster membership is measured by the posterior probability of a cluster given the data point: $member(p, c) = P(c|p)$. Another difference is that the first two approaches learn a global model, while probabilistic clustering can be interpreted as a collection of local models – or equivalently, as one very complex model. On the other hand, hierarchical clustering differs from the latter two, because it creates a hierarchy of clusters and we can select the appropriate number of clusters afterwards. In the latter two approaches, the number of clusters, k, is fixed, and we have extra difficulty to define optimal k. In addition, outliers will also be clustered, but we do not know beforehand how many clusters we should reserve for the outliers. In practice, the best solution is to try with several values of k and select the best clustering.

6.2 Measures for distance and similarity

In clustering, we need some function *sim* to measure the similarity between two data points. Selection of such measure is critical, because it defines what we consider as a cluster. For numeric attribute spaces, the distance between data points is a natural choice, but for categorial attributes we need other kinds of similarity measures. When the similarity measure *sim* has been defined, the inter-cluster similarity between clusters c_i and c_j, $D(c_i, c_j)$, can be calculated from similarities $sim(p_1, p_2)$, where $p_1 \in c_i$ and $p_2 \in c_j$.

We will first consider the distance measures d for numeric data. It is required that the distance function should be a *metric* (e.g. [HMS02][32]):

Definition 23 (Metric) *Let S be an attribute space. Distance function $d : S \to \mathcal{R}^+ \cup \{0\}$ is a* metric*, if the following conditions hold for all $x, y, z \in S$:*

1. $d(x, y) = 0 \Leftrightarrow x = y$ *(reflexivity)*
2. $d(x, y) = d(y, x)$ *(symmetry)*

[3] *Fuzzy clustering* methods (see e.g. [JMF99]) are another example of non-partitive methods. In fuzzy clustering, the *member* function gets fuzzy values.

3. $d(x, y) \leq d(x, z) + d(z, y)$ *(triangular inequality)*

If we can find such a metric, the corresponding attribute space S is called a *metric space*. With numeric data, the most common selection is L_p metric (*Minkowski distance*):

$$L_p(x, y) = (\sum_{i=1}^{k}(x_i - y_i)^p)^{1/p},$$

where $p \in \mathbb{R}^+$, $x = (x_1, ..., x_k)$ and $y = (y_1, ..., y_k)$. When $p = 2$, this corresponds to Euclidean distance, but also other values of p can be used. For example, in high dimensions, *Manhattan distance* L_1 produces better results [HAK00, AHK01]. In fact, it has been observed (e.g. [AGGR98]) that the smaller p is, the more robust and the less sensitive to outliers the method is. The reason is that with small p the distance distribution becomes more homogeneous and clusters can be better separated.

Generally, L_p metrics work well, if the clusters are compact and well-separated, but they fail, if the attributes are in different scales. This is often the case with educational data, where attributes can measure very different things like age, exercise points, or average studying time per week. As a solution, the data should be standardized to one norm or some attributes should be weighed. If the attributes are strongly correlated, we should first remove correlations by *PCA* or use *Mahalanobis metric*, which takes the dependencies into account [JMF99].

The main problem with categorial data is the lack of metrics. For ordinal data we can define the order and equality relations, but for nominal data only the equality relation holds. As a solution, several alternative similarity measures have been proposed (see e.g. [HMS02][31-38], [Kan02][120-125]):

- In *overlap metric* the idea is to calculate the number of common attribute values which two data points share: $OM(x, y) = |\{A_i \mid \pi_{A_i}(x) = \pi_{A_i}(y)\}|$, when we have k attributes $A_1, ..., A_k$. Overlap metric is actually not a metric (triangular inequiality does not hold), but its complement $k - OM(x, y)$ is a metric.

- In *mutual neighbourhood distance* (*MND*) we consider the neighbourhood points and rank them according to closeness:

  $$MND(x, y) = NN(x, y) + NN(y, x),$$

  in which $NN(x, y) = i$, if x is the ith nearest point to y.

- String values are a special case of categorial data and several distance measures have been developed for measuring distance or similarity between two strings. *Hamming distance* is a special case of overlap metric, where we calculate the number of character positions, where the strings differ. In *minimum edit distance* we define the minimum number of edit operations (e.g. insertion, deletion, substitution) needed to transform one string to another.

Similarity measures for categorial and mixed data have been described in [WM97, CLKL04, IY94].

6.3 Hierarchical clustering

In hierarchical clustering (see e.g. [JD88][58-88]) we search a hierarchy of clusters, called a *dendrogram*. Dendrogram is a binary tree structure, where each node t contains a cluster, and the subtree of node t describes a clustering of t's cluster into subclusters.

Definition 24 (Dendrogram) *Let $S = \{p_1, ..., p_n\}$ be a data set. Let node t be a record which contains three fields: content(t) is the data content of t, and rchild(t) and lchild(t) refer to t's left and right child nodes.*

A dendrogam T *is a set of nodes $T = \{t_1, ..., t_l\}$ such that*

1. *for the root node $t \in T$, content(t) = S,*
2. *for all leaf nodes $t_1, ..., t_n \in T$, content(t_i) = $\{p_i\}$, and*
3. *for each non-leaf node $t \in T$ having lchild(t) = t_1 and rchild(t) = t_2 holds content(t) = content(t_1) $\cup$ content(t_2).*

The dendrogram can be constructed until the desired number of clusters is left or we can select the best clustering afterwards, after analyzing the dendrogram.

In *agglomerative methods* we begin from clusters of single data points and merge subclusters until only one cluster is left. *Divisive methods* work in the opposite way and divide superclusters, until all points are in their own clusters. The clusters to be merged or split are defined by some score function D, which measures similarity of clusters. The agglomerative methods are easier to implement and more popular. The basic algorithm for agglomerative clustering is given in Alg. 1.

Alg. 1 HierarchicalClustering(D, n)

Input: Data set $D = \{p_1, ..., p_n\}$, $n = |D|$
Output: Dendrogram T

```
1   begin
    Initialization:
2     C = ∅, T = ∅
3     for all p_i ∈ S
4       begin
5         C = C ∪ {{p_i}}
6         Create t_i such that content(t_i) = {p_i} and
          lchild(t_i) = rchild(t_i) = Null
7         T = T ∪ {t_i}
8       end
9     k = n
10    while (k > 1)
11      begin
        Search step:
12        Select c_1 ∈ C and c_2 ∈ C such that D(c_1, c_2) is minimal
          Merge step:
13        begin
14          Create a node t
15          lchild(t) = t_1 and rchild(t) = t_2 where
            content(t_1) = c_1 and content(t_2) = c_2
16          c_1 = c_1 ∪ c_2
17          C = C \ c_2
18          content(t) = c_1
19          T = T ∪ {t}
20          k = k − 1
21        end
22      end
23    output T
24  end
```

Hierarchical methods have several advantages: they are simple to implement, the resulting dendrogram contains information about hierarchy of clusters, and we do not have to know the number of clusters beforehand. In addition, several similarity measures can be used, and the methods can be applied to categorial data, too. However, it is not clear, which level of the dendrogram is the final clustering, and

when we could stop the search. For large datasets the hierarchical methods are not efficient (with complexity of at least $O(n^2 \log n)$ [JMF99]) and storing the $n \times n$ distance matrix takes space. In addition, they do not scale well for large-dimensional data. The main disadvantage is the static nature of greedy combinations. In the basic form the data points are not moved to other clusters, once they have been assigned to one cluster[4]. In addition, most hierarchical methods are dependent on data order and produce different clusterings with different data orders.

The results of hierarchical clustering depends strongly on the measure of similarity used. This inter-cluster distance is often called a *linkage metric*. It is calculated from distances between some or all points in two clusters. The choice of linkage metric defines the shape, size and density of clusters we can find. Most metrics produce hyperspherical (isotropic) clusters. In Table 6.1, we have listed the most common inter-cluster measures and types of clusters they tend to produce.

Table 6.1: Common measures for inter-cluster distance $D(c_i, c_j)$, given the distance between points $d(p_1, p_2)$. Cluster type describes what kinds of clusters the measure tends to produce.

Metric	$D(c_1, c_2)$	**Cluster type**
Single-link	$\min_{p_1 \in c_1, p_2 \in c_2}\{d(p_1, p_2)\}$	elongated, straggly, also concentric clusters
Complete-link	$\max_{p_1 \in c_1, p_2 \in c_2}\{d(p_1, p_2)\}$	small, compact clusters; cannot separate concentric clusters
Average-link	$\frac{\sum_{p_1 \in c_1, p_2 \in c_2} d(p_1, p_2)}{\lvert c_1 \rvert \lvert c_2 \rvert}$	quite compact, dense clusters
Minimum variance	$SSE(c_1 \cup c_2)$	compact, quite well-separated, isotropic clusters; cannot find elongated clusters or clusters of very different size
Distance of centroids	$d(centroid(c_1), centroid(c_2))$	approximates SSE, if clusters are of equal size

Single-link method is flexible in the sense that it can find non-isotropic clusters (with unsymmetrical shapes) and the clusters can be concentric. On the other hand, it has tendency to produce elongated and straggly clusters. This is called the "chaining effect": the measure combines clusters through other external points and the clusters become chain-like. As a result, it works best for well-separated, non-spherical clusters. One advantage of the single-link metric compared to other linkage metrics is that it is independent from data order and the resulting clustering is always unique. Single-link metric is also quite efficient to compute.

[4]This restriction is solved in the more sophisticated implementations. For example, when hierarchical clustering is implemented by neural networks, the points can be moved from a cluster to another during clustering [KL05].

Complete-link metric works usually better, but it is slower to calculate and does not suit for large data sets. It produces small and compact clusters, but it cannot separate concentric clusters. The main disadvantage of complete-link metric is the dependency on the data order.

Example 8 *Let as consider the following situation: X and Y are two numeric variables, which spin the attribute space S, and $p_1 = (x, 2y), p_2 = (x, 4y), p_3 = (x, 6y), p_4 = (x, 9y)$ are the only data points in S. The goal is to cluster the data points into two clusters by complete-link clustering. The distance between data points is measured by squared Euclidean distance $d(x, y) = (L_2(x, y))^2$ and the distance between clusters is $D(c_1, c_2) = \max_{q_1 \in c_1, q_2 \in c_2}\{d(q_1, q_2)\}$.*

Let $d(p_1, p_2) = d(p_2, p_3) = 4y^2$ be the minimal distance between data points in S and $d(p_3, p_4) = 9y^2$ the second minimal distance. In the first step we can combine either p_1 and p_2 or p_2 and p_3. If we combine first p_1 and p_2 into cluster c_1, then we should combine p_3 and p_4 next, because $D(c_1, \{p_3\}) = d(p_1, p_3) = 16y^2$. The resulting clusters are $c_1 = \{p_1, p_2\}$ and $c_2 = \{p_3, p_4\}$. However, if we first combine p_2 and p_3 to cluster c_1, then we should combine p_1 to c_1 next, because $D(c_1, \{p_1\}) = d(p_3, p_1) = 16y^2$, while $D(c_1, \{p_4\}) = d(p_2, p_4) = 25y^2$. Now the resulting clusters are $c_1 = \{p_1, p_2, p_3\}$ and $c_2 = \{p_4\}$.

Average-link metric produces clusters which are between single-link and complete-link in their compactness. It produces dense clusters, letting larger clusters to be sparser than the smaller ones. In practice, it has produced good results, but it is inefficient for really large data sets. In addition, it also suffers for the dependency on the data order.

Minimum variance metric is famous and it is used e.g. in the classical *Ward's method* [War63]. It minimizes the variance in the clusters through SSE score function. The resulting clusters are always isotropic and quite compact, but it is not possible to find elongated clusters. SSE score function is computationally efficient, because we can calculate $SSE(c_1 \cup c_2)$ from $SSE(c_1)$ and $SSE(c_2)$, given the centroids and sizes of c_1 and c_2. In practice, minimum variance metric works very well, although it is also sensitive to the data order. In addition, SSE function is very sensitive to outliers.

Distance of centroids is sometimes used to approximate the minimum variance metric. The metric is really efficient to calculate, but results can be quite poor. In fact, we have shown that distance of centroids works well (in minimum variance sense) only, when the alternative clusters to be combined are of equal size. In addition, we have observed that the metric is less sensitive to the data order than minimum variance.

In educational data, the clusters are not necessarily well-separated and they can have various shapes. Still, we would like to find compact and well-separated (not concentric) clusters if possible. Outliers are typical, and they should not distort the clustering, but on the other hand, it would be very useful to detect them. The computation time is not a limiting factor, unlike in most other domains. According to these criteria, complete-link and average-link measures seem to be best candidates, although they can produce different results with different data orders. As a solution, we should identify all ambiguous selections and try all alternatives to find the best clustering.

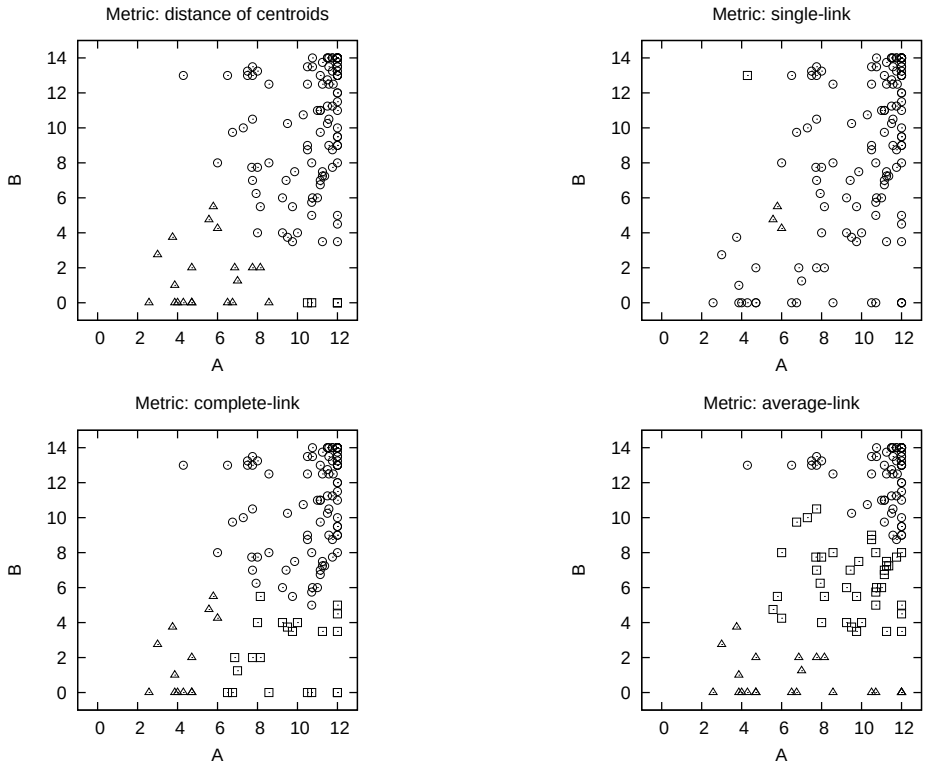

Figure 6.1: Results of hierarchical clustering to three clusters with different linkage metrics. The data consists of A and B exercise points in the *ViSCoS* data.

Figures 6.1 and 6.2 present the results of clustering A and B attributes in *Prog.1* data to three and two clusters with four linkage metrics: distance of centroids, single-link, complete link, and average-link metrics. The actual goal was to find two clear clusters, but three clusters were tried to check, whether algorithms can detect outliers. However, single-link metric was the only one which recognized the most obvious outlier. point $(4.29, 13.00)$. Otherwise the result was quite insensible.

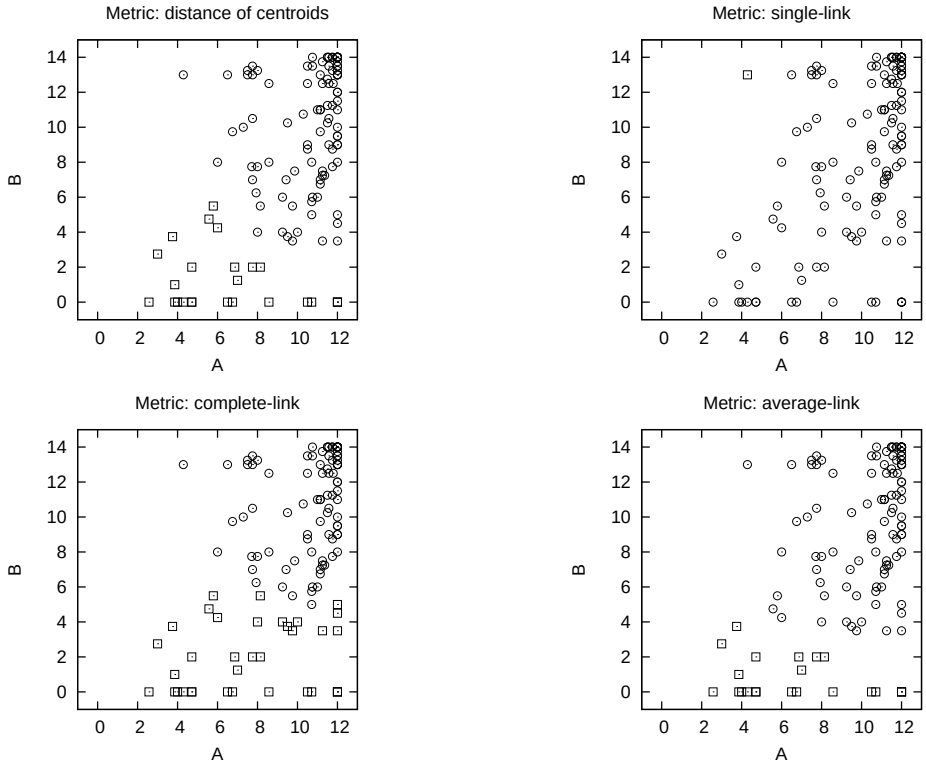

Figure 6.2: Results of hierarchical clustering to two clusters with different linkage metrics. The data consists of A and B exercise points in the *ViSCoS* data.

Distance of centroids found two larger clusters and a small group of points which had large A values but zero B value. In the figure, we can see just two points on A-axes between 10 and 11, but actually there are five points. In the corresponding dendrogram we observed that the most obvious outlier, $(4.29, 13.00)$, was in its own cluster, when the number of clusters was 9, but was then fused to neighbouring cluster. Complete-link and average-link metrics were not able to detect any outliers even when number of clusters was 10.

Clustering to two clusters produced better results. Once again, single-link metric separated just the most obvious outlier from other points. This was expected, because the clusters were not clear. However, all the other metrics produced more or less similar results with one big cluster in the top-right. Especially, the results by distance of centroids and average-link metric are very similar to each other. Visually checking, both of them are quite reasonable. To test the predictive power of these clusterings, we calculated the proprotion of students, who had actually passed the course in both clusters (Table 6.2). The results were compared to 2-means

clustering. According to this test, average-link metric performed best, because the cluster of failed students was also homogenous. However, all clustering methods produced surprisingly good results, which means that we can at least in principle predict the course outcomes very well, when only A and B attributes are known.

Table 6.2: Comparison of three clustering methods according to the final results in the course.

Method	c_1	c_2
Centroids	88% failed	86% passed
Average-link	96% failed	85% passed
2-means	85% failed	86% passed

6.4 Partitioning methods

Partitioning methods (see e.g. [JD88][]89-133) produce directly the partitioning into given k clusters. These methods are also called *relocation methods*, because they proceed by assigning data points from a cluster to another, until the score function converges. The clusters are represented by their centroids, which are typically either mean or median values of data points in clusters. Concept "medoid" is used, if we select the median among data points. These kinds of methods are also called *representative-based clustering*. The basic algorithm is given in Alg. 2.

The most common partitioning clustering method is *k-means clustering*. k-means defines centroids as cluster means. Now the inductive principle is to minimize intra-cluster variance or the sum of squared errors [EC02]

$$SSE = \sum_{j=1}^{k} \sum_{p_i \in c_j} d^2(p_i, centroid(c_j)).$$

k-means works well, if the real clusters are hyperspherical, compact and well-separated. The method is easy to implement and quite efficient even with large data sets, because we have to update only the means. However, the method is very sensitive to irrelevant attributes. Especially in high dimensions the irrelevant attributes dominate the distance measure and the method produces erroneous clusters. As a solution, irrelevant attributes should be deleted or have less weight in the distance calculation.

Alg. 2 PartitioningClustering(D, n, k)

Input: Data set $D = \{p_1, ..., p_n\}$, $n = |D|$, number of clusters k
Output: Centroids $c_1, ..., c_k$

```
1   begin
2     Select randomly k data points p_1, ..., p_k ∈ D
3     for i = 1 to k
4       c_i = p_i
5     while (not converged)
6       begin
7         for i = 1 to k
8           C_i = {c_i}
9         for all p_i ∈ D
10          begin
11            Search c_j such that d(p_i, c_j) is minimal
12            C_j = C_j ∪ {p_i}
13          end
14        Update centroids c_i
15      end
16  end
```

Another restriction is the use of distance metric d. The commonly used Euclidean distance L_2 is sensitive to outliers and does not work well in high dimensions. In such situations L_p-metrics with $p < 2$ are more recommendable [HAK00, AHK01]. However, all distance metrics require numeric data, and it is very difficult to apply k-means for categorial or mixed data. The search method is also problematic, because it is sensitive to the selection of the initialization parameters (the number of clusters, initial centroids, and the convergence condition). It is very likely that the method produces only a locally optimal clustering, unless we run it several times with different initializations. With large data sets, k-means can converge very slowly, depending on the termination criterion. Finally, k-means can produce unbalanced clusters, including even empty ones.

k-medians clustering solves some of these restrictions. It is otherwise similar than k-means, but now we define centroids as medians instead of means. The inductive principle is to minimize the absolute error of clustering [EC02]. This method works with all ordered data types and it is less sensitive to outliers. However, calculating medians is quite unfeasible. As a solution, in k-medoid it is required that the median point should be one of the data points. This can be implemented efficiently in $O(n^2)$

time [EC02].

Figure 6.3 represents the clustering results for A and B exercise points in *Prog.1* data by k-means algorithm, when $k = 3$ and $k = 2$. The cluster centroids are given in Tables 6.3 and 6.4. In 3-means clustering, the total SSE was 3102.7 and in 2-means clustering 3357.3. The results of 3-means clustering are quite similar to those of complete-link algorithm, and the method was not able to detect any ouliers. However, 2-means clustering produced nearly the same clustering as distance of centroids, except of one point, $(8.0, 4.0)$. This supports our assumption that the distance of centroids approximates SSE quite well. Table 6.2 shows that 2-means performed slightly poorer than the centroid-method, when the clusters were compared according to their predictive power.

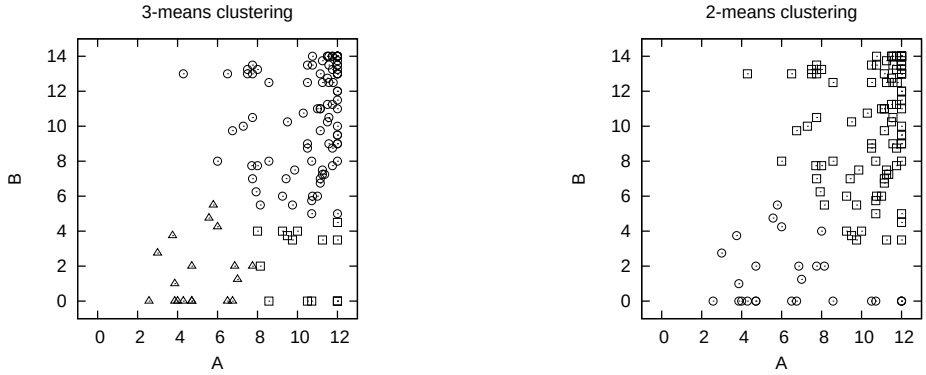

Figure 6.3: Results of 3-means and 2-means clustering. The data consists of A and B exercise points in the *ViSCoS* data.

Table 6.3: The clustering of A and B exercise points in the *ViSCoS* data by 3-means algorithm.

Cluster	$mean(A)$	$mean(B)$
c_1	10.77800	2.716667
c_2	10.60378	6.261111
c_3	5.419500	1.562500

6.5 Probabilistic model-based clustering

Probabilistic model-based clustering (see e.g. [FR00, BR93, SS04, FR98]) is an exam-

Table 6.4: The clustering of A and B exercise points the in *ViSCoS* data by 2-means algorithm.

Cluster	$mean(A)$	$mean(B)$
c_1	6.968966	1.603448
c_2	10.64896e	6.135417

ple of eager learning. It does not only produce a probabilistic clustering, but learns the model, which best describes the data. Now we can also predict clusters for new data points and update the model. The underlying idea is that the data comes from a mixture of probabilistic models (multivariate distributions). Each cluster has a prior probability and its own probability distribution. The whole model is typically represented as a multivariate finite mixture model:

Definition 25 (Multivariate mixture model) *Let S be a numeric attribute space and k the number of clusters. Let $f_j(p, \theta_j)$ be the density function of the jth cluster with parameters θ_j and π_j be the prior probability of the jth cluster. Then the* multivariate mixture model *is defined by function $f : S \to [0, 1]$ such that for all $p \in S$*

$$f(p) = \sum_{j=1}^{k} \pi_j f_j(p, \theta_j).$$

Now $f_j(p, \theta_j)$ defines the probability that data point p belongs to cluster c_j, and $f(p)$ describes the posterior probability of data point p given the whole model. If $f(p)$ is very low, the point does not fit the model, and we can interprete it as an outlier.

The density function f_j describes the data distribution in cluster c_j. In principle, we can define a different type of density function for each cluster. This is useful, when the data is very skewed. However, in practice, it is very difficult to define the appropriate distributional form without any prior knowledge. That is why it is usually assumed that the density in all clusters has the same distributional form, and only the distribution parameters are different. A common choice is to assume normal distribution.

Example 9 *Let S be an attribute space spun by two numeric attributes X and Y. Then the density function f_j in cluster c_j is defined by*

$$f_j(x,y) = \frac{1}{2\pi\sigma_X\sigma_Y\sqrt{1-r^2}} e^{-(\frac{1}{1-r^2}((\frac{x-\mu_X}{\sigma_X})^2 - 2r(\frac{x-\mu_X}{\sigma_X})(\frac{y-\mu_Y}{\sigma_Y}) + (\frac{y-\mu_Y}{\sigma_Y})^2))},$$

where $\mu_X = mean(X)$, $\mu_Y = mean(Y)$, $\sigma_X = stdev(X) > 0$, $\sigma_Y = stdev(Y) > 0$, *and* $r = corr(X,Y)$.

If X *and* Y *are uncorrelated* ($r = 0$), *the density function* f_j *reduces to*

$$f_j(x,y) = \frac{1}{2\pi\sigma_X\sigma_Y} e^{-(\frac{x-\mu_X}{\sigma_X})^2 - (\frac{y-\mu_Y}{\sigma_Y})^2}.$$

The whole clustering process consists of four steps:

1. Determine the number of clusters k.
2. Choose the density functions f_j for all clusters c_j.
3. Determine the cluster probabilities π_j and parameters θ_f from data.
4. Assign each point p to the most probable cluster, i.e. select such c_j that $member(p, c_j) = P(c_j|p) = \frac{\pi_j f_j(p)}{f(p)}$ is maximal.

Probabilistic clustering has several advantages. The clustering is not strict, but for each data point p, we define a probability distribution $P(c_1|p), ..., P(c_k|p)$. Now we can represent overlapping clusters, where the same data point belongs to several clusters with different probabilities. In two-dimensional case, the densities have a nice visual representation as contours, and outliers are easily recognized. The method is very flexible and can describe even complex structures. For example, every cluster can have different size and density, even different type of distribution. In addition, it has been observed that several other clustering methods are special cases of probabilistic clustering.

Example 10 *Let* X *and* Y *be two numeric variables, which follow normal distribution with parameters* $X \sim (\mu_X, \sigma^2)$, *and* $Y \sim (\mu_Y, \sigma^2)$. *If* X *and* Y *are mutually independent, then the density function is*

$$f_j(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x-\mu_X)^2 - (y-\mu_Y)^2}{2\sigma^2}}$$

for all clusters $j = 1, ..., k$.

Each data point (x, y) is joined to cluster c_j, for which $f_j(x, y)$ is maximal. Now $f_j(x, y)$ is maximal, when $(x - \mu_X)^2 + (y - \mu_Y)^2$ is minimal, which is the same criterion as in k-means clustering.

However, the flexibility of probabilistic clustering has also a drawback: the resulting clustering depends strongly on the initial parameters – especially the form of the distribution and the number of clusters – which can be hard to define without any prior knowledge. These two parameters depend on each other and, typically, the simpler probabilistic models we assume, the more clusters we need, and vice versa. The outliers can be described well as improbable data points, but if we assume a multinormal distribution, the method is not robust under outliers, which have a strong effect on variance. The method can be applied to categorial data, but mixed numeric and categorial data can be problematic to handle.

The best number of clusters is usually selected according to some score function like *Minimum Description Length* (*MDL*) or *Bayesian Information Criterion* (*BIC*) [FR00]. Both of them maximize the log-likelihood of data with some penalty terms for large k. Techniques based on cross-validation have also been applied successfully [Smy00]. According to [SR00], both *BIC* and cross-validation methods converge to the global optimum, when the sample size grows, but *BIC* is faster to compute.

Probabilistic clustering method is very sensitive to initial parameter settings and often the data is first clustered with another method. For example, initialization by hierarchical clustering has produced good results and at the same time the ideal number of clusters could be determined [DR98, FR98]. Another alternative is to implement the entire probabilistic clustering in a hierarchical way [MR84, BR93].

Usually the parameters are selected by *Maximum likelihood* (*ML*) principle, to maximize data (log-)likelihood given the model. This can be approximated by *Expectation Maximization* (*EM*) method, but other score functions and optimization methods can be used as well (see e.g. [RG97]). Probabilistic clustering by *EM* method is described in Alg. 3

Unfortunately, the *EM* method is quite time-demanding. For example, if we suppose multivariate normal distribution, then each step takes $O(kv^2n)$ time, in which v is the number of attributes [HMS02]. Convergence can also take time, and the method is not feasible for large data sets. In practice, probabilistic clustering with the *EM* algorithm has produced varying results. Fraley and Raftery [FR00] have reported good results with multivariate normal distribution, when the attributes were extracted from eigenvalue decompositions. On the other hand, the *EM* algorithm can produce poor results, when some clusters contain only a few data points or when the attributes are strongly correlated. In addition, it should be remembered

Alg. 3 EMProbabilisticClustering(D, n, k)

Input: Data set $D = \{p_1, ..., p_n\}$, $n = |D|$, number of clusters k
Output: Cluster probabilities $\pi = (\pi_1, ..., \pi_k)$,
cluster parameters $\theta = (\theta_1, ..., \theta_k)$

```
1   begin
2     for all j = 1 to k
3         Initialize π_j and θ_j
4     while (not converged)
5       begin
6         E-step: for all p_i ∈ D and for all j = 1 to k
7           z_ij = π_j f_j(p_i,θ_j) / Σ_{j=1}^{k} π_j f_j(p_i,θ_j)
8         M-step: for all j = 1 to k
9           Update π_j and θ_j such that
            Σ_{i=1}^{n} Σ_{j=1}^{k} z_ij log(π_j f_j(p_i, θ_j)) is maximal
10      end
11      output π, θ
12  end
```

that the *EM*-algorithm can get stuck at a local optimum and, as a solution, several tricks have been suggested to improve the results.

Table 6.5 represents the probabilistic clustering of A and B exercise points to six clusters. The clustering was implemented by *Weka*-tool [Wek], which suggested (according to cross-validation test) that the most likely number of clusters is six. Unfortunately, it is not possible to set another number of clusters, so that we could have compared the results with previous examples. In addition, *Weka* assumes that the cluster distribution is normal and the attribute dependencies are not taken into account. In the educational data these are often unrealistic assumptions, because the data is seldom normally distributed and the attributes are often mutually depdendent. In the future research, we will implement probabilistic clustering which allows other distributional forms and considers the dependencies.

Figure 6.4 visualizes the clusters in two ways. The first figure (on the left) presents only the cluster centroids and boundaries defined by $stdev(A)$ and $stdev(B)$. In the second figure (on the right), we have drawn the density contours by *Matlab*. The latter figure reveals that there is a dense area in the top-right corner. In fact, clusters c_1, c_3, and c_4 compose one cluster. However, the cluster is so irregular that it was not possible to recognize it assuming the normal distribution. Cluster c_2

in the top is also dense but small. All the other clusters are so sparse that it is questionable whether they compose any clusters at all.

Table 6.5: Result of probabilistic clustering of A and B exercise points to six clusters. $P(C)$ tells the prior probability of cluster C, and the size tells how many data points belong to the cluster.

Cluster	$mean(A)$	$stdev(A)$	$mean(B)$	$stdev(B)$	$P(C)$	Size
c_1	12.00	2.565	13.64	0.435	0.119	15 (12%)
c_2	7.24	1.306	13.09	0.258	0.055	8 (6%)
c_3	11.43	0.476	10.911	2.388	0.385	50 (40%)
c_4	11.41	0.656	2.832	2.802	0.08	9 (7%)
c_5	5.05	1.693	0.92	1.229	0.12	15 (12%)
c_6	8.53	1.643	6.30	2.649	0.24	28 (22%)

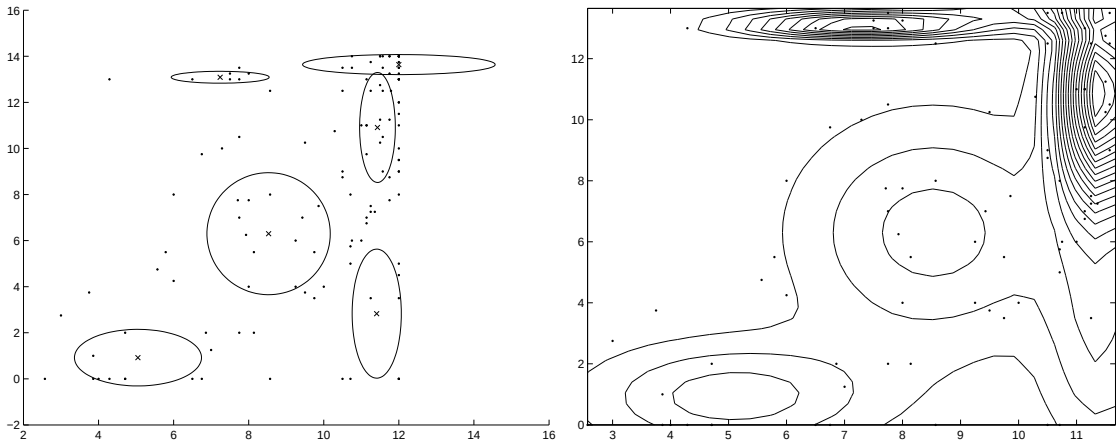

Figure 6.4: Probabilistic clustering of A and B attributes by the *EM* algorithm. The left figure represents the six clusters, discovered by *Weka*. The ellipsoids represent the standard deviations in A and B. The righ figure represents the corresponding density contours calculated by *Matlab*.

6.6 Comparison of clustering methods

Selecting the most appropriate clustering method for a given problem is always difficult. In practice, it is often recommendable to try several methods. We can also combine different methods. For example, we can first use some hierarchical method

Table 6.6: Comparison of clustering methods. + indicates that the method supports the property, and − that it does not. (+/−) indicates that the general method lacks the property, but its extensions can overcome the problem.

	Hier.	Relocation Partitioning		
		Prob.	k-medoids	k-means
Good clustering quality	−	+	+	+
Independent from data order	−	+	+	+
Robust to initial parameters	+	−	−	−
Resistant to outliers	+	+/−	+	−
Flexible cluster shape	+/−	+	−	−
Number of clusters unfixed	+	−	−	−
Easy interpretation	+	+	(+/−)	(+/−)
Works with categorial data	+	+	+/−	−
Computational efficiency	−	−	+	+/−

to find the number of clusters and approximate centroids, and then improve the results by k-means or probabilistic clustering.

In Table 6.6 we have compared the main approaches according to the most important criteria for clustering educational data sets. Some of the criteria are desirable in other domains as well, while others have a special importance in clustering educational data. The analysis is based on several sources, e.g. [HMS02, JDM00, JD88, JMF99, EC02, Ber02, FR00, SS04, FR98, KL05, Fox96].

Clustering quality is very difficult or even impossible to define. The resulting clusters depend on the inductive bias – how the cluster is defined – and methods with different bias cannot be compared. Often compact and well-separated clusters are preferred, but only if such clusters really exist in the data. The problem is that all clustering methods find some clusters, even if the data itself does not have any clustering. Thus, the first question should be whether the data has clustering tendency – i.e. if it really contains $k > 1$ clusters.

One way to compare clustering methods with the same bias (e.g. *SSE*) is to test how good local optimum it converges to. In this sense probabilistic *EM* is better than k-means, which is better than Ward's method, even if all of them use the same metric with certain assumptions.

The next two properties, *independence from data order* and *robustness to initial parameters* have also influence on the clustering quality. Most of the hierarchical methods are sensitive to the data order, while the other methods are sensitive to the selection of the initial parameters. As a result, they can produce different clusterings

with different data orders or initial parameters. In probabilistic clustering, it is especially difficult to decide the form of the distribution. Small, real word data sets hardly contain normally distributed clusters, as is often assumed.

Resistance to outliers is an important property for the educational data, which typically contains exceptional data points. It is desirable that the method recognizes the outliers and leaves them to their own clusters. However, outliers depend on the definition, like clusters, and there is no absolute measure to detect them. For example, a point can be an outlier locally, in the context of the closest clusters. For a dense cluster even a loose point can be an outlier, while a sparse cluster can contain even remote points. Probabilistic clustering can model such outliers, because each cluster has its own distribution. However, normal distribution is sensitive to outliers, and the overall clustering can be corrupted. k-means is especially sensitive to outliers, if we use the common Euclidean metric. k-medoids is more robust, because it uses L_1-metric. Hierarchical methods are often resistant to outliers, but they cannot detect local outliers.

Flexible cluster shape is a desirable property for all real world applications. We cannot expect that the data would fall into nice hyperspherical clusters. However, most clustering methods restrict the shape of clusters we can find. Partly this is due to cluster representation. If we represent the cluster by its centroid (k-means), all the other points should lie in a hypersphere around it. If we know also the standard deviations in clusters, the shape can be hyperellipsoidal. Finally, if we take into account the attribute correlations (probabilistic clustering using covariance matrix and Mahalanobis metric), then the orientation can also vary. Hierarchical linkage methods, which approximate SSE, produce more or less hyperspherical shapes. The single-link method does not have this restriction, and the clusters can be arbitrary shaped, but it suffers for the chaining effect.

Unfixed number of clusters is also a desirable property, because we seldom know the ideal number of clusters beforehand. Especially, if the data contains several outliers, we do not know how many clusters should be reserved for them. In hierarchical methods, the number of clusters is not fixed, and we can select the appropriate number by analyzing the dendrogram. In all other methods we have to fix the number of clusters beforehand.

All the methods have *easy interpretation* given two-dimensional data. Dendrograms of hierarchical methods and contours of probabilistic methods are the most informative, while k-means and k-medians give just a strict clustering, even if the actual clusters were overlapping. Dendrograms can be presented in higher dimensions, too, but they become too large and complicated for visual representation, when the size of the data increases.

For the educational data, it is often required that the method *works* also *with categorial data*. This means that we should be able to define and combine different similarity measures in clustering. In hierarchical methods this is easy, because most linkage metrics do not require a distance metric. For example, in single-link, complete-link, and average-link methods it is enough that we can define the similarity between two data points. Distance of centroids method applies to ordinal data, where we can define the cluster centroids. k-means works only with numeric data, but we can apply the general idea of partitioning clustering for other similarity measures. k-medoids works with any ordinal data. Probabilistic methods can be defined for categorial data, too, but mixing numeric and categorial data is very difficult.

Computational efficiency has minor importance in the educational domain, where data sets are typically small and low-dimensional. However, finding the globally optimal clustering is a heavy process even for the educational data sets. In the table the efficiency is measured by the time complexity on the number of rows n in small dimensional data. However, many clustering methods scale poorly to higher dimensions, in the sense of their efficiency and clustering quality.

6.7 Clustering the numeric *ViSCoS* data

In this experiment we clustered the whole *Prog.1* and *Prog.2* data sets, except the total points which were saved for testing the predictive power of clusterings. We selected probabilistic clustering method, because it can detect also outliers. The clustering was implemented by *Weka* tool using the *EM* algorithm. *Weka* assumes that the distribution in each cluster is normal and the dependencies between attributes are not considered, which decreases the clustering quality. In addition, the user cannot decide the number of clusters, but it is selected in *Weka* by cross-validation.

The resulting clusters for the *Prog.1* and *Prog.2* data sets are given in Tables 6.7 and 6.8. In *Prog.1*, the number of clusters was five, and the log likelihood of clustering was -6.88129. In *Prog.2*, the number of clusters was two, and the log likelihood of clustering was -17.15895. For each cluster c_i, we report the cluster probability π_i, the mean vector μ_i, standard deviation vector σ_i, and the probability to pass the course $P(FR = 1|C = c_i)$.

In *Prog.1* clustering, clusters c_1 and c_5 have the clearest interpretation. In c_1 all students have achieved only a little of points in all categories A, B, C and the probability to pass the course is only 0.053. In c_5 all students have achieved a lot of

exercise points in all categories and the probability to pass the course is 1.00. Cluster c_5 is also the most homogenous cluster, measured by the standard deviation. In addition, it covers about 31% of students. In clusters c_2 and c_4 the probability to pass the course is also high (> 0.81) and the mean exercise points are relatively high in all categories A, B, C. Cluster c_3 has the lowest predictive power, but it is also the smallest cluster. The students in c_3 have achieved a lot of points in A category, but ony a little of points in B and C. We assume that the cluster contains several outliers, the most unpredictable students.

Table 6.7: Probabilistic clustering of attributes A, B, C in the *Prog.1* data. π_i gives the cluster probability, μ_i the mean vector, σ_i the standard deviation vector, and $P(FR1 = 1|C = c_i)$ the probability to pass *Prog.1* course in the given cluster c_i.

c_1	π_1=0.165 μ_1=(6.63,1.54, 0.00) σ_1=(2.75,1.88,3.50) $P(FR1 = 1\|C = c_1)$=0.053
c_2	π_2=0.170 μ_2=(7.20,9.16,5.34) σ_2=(1.43,3.85,2.99) $P(FR1 = 1\|C = c_2)$=0.818
c_3	π_3=0.090 μ_3=(10.33,3.43,1.61) σ_3=(1.40,2.41,1.52) $P(FR1 = 1\|C = c_3)$=0.385
c_4	π_4=0.260 μ_4=(11.07,8.29,3.47) σ_4=(0.87,2.17,2.14) $P(FR1 = 1\|C = c_4)$=0.812
c_5	π_5=0.315 μ_5=(11.70,13.10,7.28) σ_5=(0.41,1.06,2.73) $P(FR1 = 1\|C = c_5)$=1.000

In *Prog.2* clustering, all students were divided into two nearly equal size clusters. Cluster c_1 is a clear cluster of succesfull students. The mean points in attributes $A, B, C, TP1, D$, and E are high, and the probability to pass the course is 0.933. Cluster c_2 is less clear. The students in c_2 have succeeded relatively well in *Prog.1* course, but in *Prog.2* their mean points are relatively low. The predictive power of the cluster is very low, $P(FR2 = 1) = 0.379$.

Table 6.8: Probabilistic clustering of attributes $A, B, C, TP1, D, E, F$ in *Prog.2* data. π_i gives the cluster probability, μ_i the mean vector, σ_i the standard deviation vector, and $P(FR2 = 1)$ the probability to pass *Prog.2* course in the given cluster c_i.

c_1	π_1=0.495 μ_1=(11.13,12.95,7.47,29.11,8.13,13.37,3.87) σ_1=(1.60,1.24,2.67,3.14,2.94,4.80,2.68) $P(FR2 = 1\|C = c_1)$=0.933
c_2	π_2=0.505 μ_2=(10.25,8.44,3.81,23.63,3.67,3.54,0.87) σ_2=(1.90,3.03,3.03,5.56,2.43,3.34,1.37) $P(FR2 = 1\|C = c_2)$=0.379

Next, we analyzed outliers in the *Prog.1* and *Prog.2* data sets, given the previous clusterings. We calculated the probability of each data point p by defining the density function $f(p)$ for the given clustering. The results are given in Tables 6.9 and 6.10. In *Prog.1* all data points had relatively high density and the cut-off value was $f(p) < 0.00002$. In *Prog.2* the points had lower densities and the cut-off value was $f(p) < 1e - 30$. The cut-off values were defined by trial and error, to cover at most 10% of data points such that the deviation to the next non-outlier was maximal.

In Table 6.9 we observe that none of the discovered outliers is a real outlier. The only common thing between the points was that none of them belonged to cluster c_3, which we assumed to contain most of outliers. However, we recall that all points in *Prog.1* clustering were relatively probable, which indicates that the clustering had overfitted.

On the other hand, in Table 6.10, 8/9 of the discovered outliers are clear outliers. They are students, who have performed well in the course, but still failed. The most interesting discovery is that the total points, $TP2$, were not used in the clustering, but still the clustering could reveal the students, whose final results are unpredictable. This is an important discovery, because it indicates that we can recognize the unpredictable students already during the course.

When we compare the previous outliers to those dicovered by the linear regression method (Table 5.4), we observe that only two data points are common. In the *Prog.1* data the common point is $(9.5, 10.3, 0.0, 2.0)$, which has no clear interpretation as an outlier, and in *Prog.2* point $(4.3, 13.0, 12.0, 28.5, 12.0, 8.0, 0.0)$, which describes a surprising fail. This is quite understandable, because linear regression defines

Table 6.9: Discovered outliers in the *Prog.1* data. The data point has been defined as an outlier, if $f(p) < 0.00002$. The points are interpreted as "Surprising success", "Surprising failure", or "Not a clear outlier".

Data point p	$f(p)$	Interpretation
(4.3,13.0,12.0,28.5)	6.75e-08	Not a clear outlier
(10.5,13.5,11.0,26.5)	1.33e-06	Not a clear outlier
(10.3,10.8,9.5,27.5)	2.37e-06	Not a clear outlier
(12.0,5.0,8.0,33.0)	3.73e-06	Not a clear outlier
(10.5,0.0,5.0,9.0)	1.60e-05	Not a clear outlier
(9.5,10.3,0.0,2.0)	1.78e-05	Not a clear outlier
(12.0,12.0,0.0,32.0)	1.96e-05	Not a clear outlier

Table 6.10: Discovered outliers in the *Prog.2* data. The data point has been defined as an outlier, if $f(p) < 1e-30$. The points are interpreted as "Surprising success", "Surprising failure", or "Not a clear outlier".

Data point p	$f(p)$	Interpretation
(12.0,14.0,12.0,30.0,12.0,19.5,7.0)	3.09e-49	Surprising failure
(10.5,13.5,11.0,26.5,12.0,19.5,2.0)	1.14e-43	Surprising failure
(4.3,13.0,12.0,28.5,12.0,8.0,0.0)	2.26e-39	Surprising failure
(12.0,14.0,6.0,34.0,12.0,15.5,2.0)	4.10e-39	Surprising failure
(12.0,14.0,9.0,32.0,11.8,17.5,6.0)	5.22e-38	Surprising failure
(12.0,10.0,7.0,24.5,12.0,8.8,5.0)	1.17e-35	Surprising failure
(12.0,13.3,8.0,34.0,12.0,7.0,2.0)	2.30e-34	Surprising failure
(12.0,13.5,8.5,22.0,12.0,4.0,1.5)	1.90e-32	Not a clear outlier
(12.0,13.8,12.0,33.0,11.5,15.8,8.0)	6.72e-32	Surprising failure

outliers as exceptions from the global linear model, while probabilistic clustering defines outliers as exceptions from local clusters. In the future research, we will study this problem further and try to develop suitable outlier detection methods for the educational data sets.

6.8 Clustering the categorial *ViSCoS* data

In the other experiment we clustered categorial (binary-valued) *ViSCoS* data. The goal of clustering was to find two separate and dense clusters, which would cover as

many data points as possible. Our expectation was that the failed students would fall into one cluster and the successful students into the other one.

For this purpose, we have designed a new heuristic algorithm called *BinClust*. The underlying inductive bias consists of the assumptions that the data is skewed and the clusters are separate, dense and large. In addition, the quality of clustering depends on the initial discretization to binary data. In discretization we have already grouped the data points in each dimension. Each binary valued data point represents several numerical data points, and it can be interpreted as a subcluster. When binary data is clustered, these subclusters are combined into larger clusters. Because the discretized data set is smaller than the original one, it is possible to search the globally optimal clustering of binary data.

In our algorithm, the distance between two data points is defined by Hamming distance for bit vectors (i.e. the sum of disagreeing bits): $d(p_1, p_2) = \sum_{i=1}^{k} xor(p_1[i], p_2[i])$. In addition, we need a measure for the distance between data point p and cluster C. This is defined as an average distance of p from all data points in C: $D(p, C) = \frac{\sum_{q \in C} d(p,q)}{|C|}$. The *BinClust* algorithm is given in Alg. 4.

We observe that the algorithm is similar to the existing hierarchical clustering algorithms. In the initialization phase, we require that the distance between data point p_i and cluster core s_j is at most 1 like in centroid-methods. In addition, we observe that all data points in one cluster can differ from each other by at most 2 bits like in complete-link methods. In the enlargement phase, the data points are added to clusters according to average-link method. The difference is that we want to get only two but large clusters, and thus we select the largest cluster candidates from the first phase. In addition, our algorithm is not order-sensitive (like centroid, complete-link and average-link methods), because we calculate the distance from the current cluster core, and not from the whole cluster under construction.

It should be remarked that in the initialization phase, it is possible that we find several equally maximal but overlapping sets. In this case we suggest that all alternatives are considered separately. After enlargement, we can select the clustering which covers more data points. The underlying assumption is that the data is so skewed that in practice we have only few alternatives. In the worst case, when the data has a uniform distribution, we should check all 2^k alternatives, where k is the number of attributes.

The clusterings of the *Prog.1* and *Prog.2* data sets are given in Tables 6.11 and 6.12. For each cluster, we give the cluster size, mean vector, and proportion of accepted students. In *Prog.1* we used attributes A, B, C and in *Prog.2* $B, C, TP1, D, E, F$. We dropped attribute A, because it was independent from all *Prog.2* attributes (D, E,F and $FR2$).

Table 6.11: Results of clustering the binary-valued *Prog.1* data by *BinClust* algorithm. Mean vector gives the mean values for A, B and C attributes. $P(FR1 = 1)$ gives the relative frequency (probability) of passing the course in the given group.

Group	Size	Mean	$P(FR1 = 1)$
c_1	44	[0.45,0.00,0.00]	16/44=0.364
c_2	57	[1.00,1.00,0.82]	53/57=0.930
Others	24	[0.29,0.50,0.88]	21/24=0.875

Table 6.12: Results of clustering the binary-valued *Prog.2* data by *BinClust* algorithm. Mean vector gives the mean values for $B, C, TP1, D, E$ and F attributes. $P(FR2 = 1)$ gives the relative frequency (probability) of passing the course in the given group.

Group	Size	Mean	$P(FR2 = 1)$
c_1	33	[0.73,0.00,0.39,0.06,0.06,0.00]	5/33=0.152
c_2	36	[0.94,0.83,1.00,0.92,0.81,0.42]	33/36=0.917
Others	19	[0.84,0.37,0.90,0.37,0.32,0.16]	12/19=0.632

In *Prog.1* the clusters covered only 54% of students. Cluster c_1 consisted of clearly weak students, but still 36% of them had passed the course. Cluster c_2 was more homogenous. The students in c_2 had achieved a lot of points in all categories, and 93% of them had passed the course. In the group of unclustered students, the passing probability was also relative high, 88%. The overall clustering quality was quite poor, which suggests that the *Prog.1* data is heterogenous and more clusters are needed. We recall that five clusters were needed to cluster the numeric *Prog.1* data. However, it is also possible that the discretization has failed to catch the important differences in the numeric data.

On the other hand, clustering the *Prog.2* data was very successful. We discovered two large and homogenous clusters, which covered 78% of data. Cluster c_1 described clearly weak students, who had got a little of points in all attributes except B. In this group 85% of students had failed the course. On the other hand, cluster c_2 described clearly successful students, who had got a lot of points in all categories except in F, which was hard for all students. 92% of students in c_2 had passed the course. The group of unclustered students was more heterogenous and the predictive power was poor ($P(FR2 = 1) = 0.63$).

Alg. 4 BinClust(D, n, F, k)

Input: Data set $D = \{p_1, ..., p_n\}$, $n = |D|$, frequencies $F = \{|p_1|, ..., |p_n|\}$, number of seeds k
Output: Clusters c_1, c_2

```
1   begin
2     Initialization:
3     Sort D into ascending order
4     for i = 1 to k
5       begin
6         Select s_i = p_i
7         D = D \ p_i
8       end
9       k = k + 1
10    for p_i ∈ D
11      for j = 1 to k
12            if (D(s_j, p_i) == 1))
13              begin
14                s_k = s_j ∪ p_i
15                k = k + 1
16              end
17    Select c_1 = s_i, c_2 = s_j such that s_i ∪ s_2 = ∅ and
      |s_i| and |s_j| are maximal
18    for all p_i ∈ D
19      if (p_i ∈ c_1 or p_i ∈ c_2)
20        D = D \ p_i
21    Enlargement:
22    for all p_i ∈ D
23      if (D(c_2, p_i) − D(c_1, p_i) ≥ min_d)
24        begin
25          c_1 = c_1 ∪ p_i
26          D = D \ p_i
27        end
28      else if (D(c_1, p_i) − D(c_2, p_i) ≥ min_d)
29        begin
30          c_2 = c_2 ∪ p_i
31          D = D \ p_i
32        end
33    output c_1, c_2
34  end
```

Chapter 7

Classification

Classification is the main problem of adaptive learning environments. Before we can select any adaption actions like selecting tasks, learning material, or feedback for the given learner in her/his current situation, we should first classify the student's current situation. Usually, all information is given in the student model, SM, whose properties determine the appropriate action. The actions are usually selected by rules *"If SM is of type X, then select action Y"*. The same pattern occurs in human tutoring, as well. For example, in the *ViSCoS* project, the rule is *"If the student will likely fail the course, then intervene"*. The form of intervention is up to course tutors, but predicting who will likely fail is often difficult for the teachers. In this case, we can base the classifications on the existing course data.

In the following, we will define the classification problem and analyze the appropriateness of the main classification methods for the educational domain. We give several guidelines for selection of the classification method for the given problem. Finally, we will report the empirical comparison of the best classifiers for the *ViSCoS* data.

7.1 Classification problem

The general classification problem is the following:

Definition 26 (Classification problem) *Let $R = \{A_1, ..., A_k, C\}$ be a set of attributes, where C is a discrete valued class attribute. Let the values of C be $Dom(C) = \{c_1, ..., c_l\}$. Let S be an attribute space spun by R.*

The classification problem *is to find a mapping* $f : Dom(A_1) \times ... \times Dom(A_k) \rightarrow Dom(C)$ *such that for all data points* $p \in S$, $f(p[A_1, ..., A_k]) = p[C]$.

In practice, we can seldom discover function f which would define $f(p)$ correctly for all current and future data points p. Especially, in the educational domain the data sets often contain inconsistent data points p_1 and p_2 for which $p_1[A_1, ..., A_k] = p_2[A_1, ..., A_k]$ but $p_1[C] \neq p_2[C]$. For example, the binary *Prog.1* data set contained 14 inconsistent data points (11% of the data), and the projection of the binary *Prog.2* data set to attributes $TP1, D, E, F, FR2$ contained 12 inconsistent data points (14% of the data). One solution is to define for all data points p a probability distribution $P(C = c_i|p)$ for all classes c_i, and then select the most probable class or classes. These kinds of classifiers are called *probabilistic classifiers*.

Definition 27 (Discriminative and probabilistic classification) *Let* R, C, *and* S *be as before, and* M *a classifier.*

If M *defines a mapping* $M : Dom(A_1) \times ... \times Dom(A_k) \rightarrow Dom(C)$ *such that for all* $p \in S$ $M(p[A_1], ..., p[A_k]) \in C$, *we say that* M *performs* discriminative classification.

If M *defines a mapping* $M : Dom(A_1) \times ... \times Dom(A_k) \rightarrow [0, 1]^l$ *such that for all* $p \in S$ $M(p[A_1, ..., A_k] = [P(C = c_1|p), ..., P(C = c_l|p)]$, *we say that* M *performs* probabilistic classification.

The first decision is to define the representative power required for the model. In classification this depends on the form of *decision boundaries*, i.e. the class borders (see e.g. [DHS00][216-219]). In the simplest case, the decision borders are linear combinations of attributes A_i. In 2-dimensional data this corresponds to a straight line, in 3-dimension a plane, and in higher dimensions a hyperplane. In this case, we say that the classes are *linearly separable*. However, the boundaries can be also non-linear, involving higher order terms or ratios. A more complex classifier is required to represent such decision boundaries, but in the same time the model becomes more risky for overfitting, especially when the data set is small. That is why it is often better to assume a simpler model, even if the actual boundaries are only approximately linear.

In the *ViSCoS*-project our main goal is to the predict potential dropouts and failures as early as possible. In addition, it would be nice to recognize excellent students, who might desire more challenges, but currently we will concentrate on simple binary class variable $FR = \{pass, fail\}$. Obviously, probabilistic classification suits better for our purposes. Currently even the training data is not consistent and we cannot expect that the course outcomes are deterministically predictable, when human

factors are involved. In addition, the probabilities contain more information about the chances to pass or fail the course than the pure class values.

In linear regression analysis we observed that the course outcomes can be predicted quite well assuming linear dependency. This indicates that the class boundaries are quite linear. Multiple linear regression models can already be used as classifiers, if we interpret the results as class values: $TP < 15$ corresponds to $FR = 0$, and $TP \geq 15$ corresponds to $FR = 1$. This is a very simple model, but in practice it is one of the best models for our small data sets. The only problem is that the linear regression cannot handle categorial data and, very often, we have to combine both numeric and categorial data. Naive Bayes model is a good candidate to compete with the linear regression, because it is also a simple model, which recognizes linear boundaries, but it can handle categorial data. In addition, we will analyze more complex models like decision trees, general Bayesian networks, neural networks, and nearest neighbour classifiers.

7.2 Tree models

The idea of *tree models* (see e.g. [HMS02][343-347]) is to represent a set of decision rules in a tree form. On each level of the tree the attribute space is partitioned to smaller subspaces, until the predicted variable – typically the class – is known. The most common tree model is a *decision tree* (see e.g. [Mit97][52-80]), which is used for classification. Similarly, *regression trees* can be used for predicting a continuous variable. In the following, we will concentrate only on the decision trees.

7.2.1 Decision trees

A decision tree has a very simple structure. Each inner node contains an attribute test and leaf nodes tell the class. When we have binary data, the tree is also binary and each test simply checks the truth value of the given attribute. Thus, each path in the tree is equivalent to logical conjunction $T_1 \wedge T_2 \wedge ... \wedge T_k$, where each attribute test T_i is of form $A_i = 1$. If we have non-binary categorial variables, we can either use higher order trees (with more than two children per node) or use logical disjunction in the test: $A_i = a_1 \vee ... A_i = a_l$. For numerical attributes, a threshold test $A_i < a$ is used instead. In principle, we could discretize the data first, but the resulting intervals are not necessarily optimal for classification purposes.

The earliest decision trees were constructed by human experts, but nowadays they are commonly learnt from data. The basic idea in all learning algorithms is to

partition the attribute space until some termination criterion is reached in each leaf. Usually, the criterion is that all points in the leaf belong to one class. However, typically data contains inconsistencies, i.e. there are data points which share all other attribute values but not the class value. This may be due to several reasons: the data may contain noise (errors), the attributes do not contain all the necessary information for deterministic classification, or the domain is truly nondeterministic. In this case we usually select the class, into which the most of data points in the node belong. This principle is known as *majority vote* principle (see e.g. [SL91]). An alternative is to report the class probabilities according to relative frequencies in the node. The class probability is in fact the same as confidence (conditional probability) of the corresponding decision rule.

The attribute-value tests are selected in the order of their *discriminatory power*, i.e. the features which divide most of data points to correct classes are set at the top. This is in fact an application of *Occam's Razor Principle*, which assumes that shorter trees are better than high [Mit97][65-66]. In practice, some score function is needed. For example, we can minimize the entropy or the probability of an error in each branch. An overview of available score functions is given in [SL91].

The problem of this basic approach is that it produces easily too complex models, which have seriously overfitted. There are two alternative solutions to this problem: *early stopping* and *post-pruning* [Mit97][68-69]. In early stopping approach we stop the tree construction before the model has become too complex. The problem of this approach is to decide the optimal point to stop (greedy) growing. In practice, the pruning approach works better. In pruning approach (e.g. in *C4.5* algorithm [Qui93]) we let the tree to overfit and prune it afterwards. Usually, pruning is performed by merging leaf nodes one by one or replacing a subtree by a leaf. An alternative is to consider the corresponding set of decision rules and remove all redundant (irrelevant) conditions from the rules. This approach has some advantages, but it is not recommended for noisy and non-deterministic domains [DHS00] like our educational data sets. In all approaches we need some criterion to decide the appropriate complexity. This criterion can be based on the cross-validation, χ^2-test, or some other measures like minimum description length (*MDL*) [Mit97][69].

Decision trees have many advantages: they are simple and easy to understand, they can handle mixed variables (i.e. both numeric and categorial variables), they can classify new examples quickly, and they are flexible. Enlargements of decision trees can easily handle small noise and missing attribute values. Decision trees have very high representational power, because they can approximate highly non-linear decision boundaries, even if the boundaries are everywhere piecewise parallel to attribute axes. However, it should be remembered that the resulting model can be seriously overfitted, especially if we have a small training set.

However, decision trees have also serious disadvantages. The main restriction is the assumption that all data points in the domain can be classified deterministically into exactly one class. As a result all inconsistencies are interpreted as errors. Thus decision trees do not suit for intrinsically nondeterministic domains. This restricts their applicability for modelling educational data, which contains often several outliers. Class probabilities have sometimes been suggested as a solution, but the resulting system is very unstable, because each leaf node has its own probability distribution [HMS02][346]. Thus even a minor change in one of the input variables can change probabilities totally, when the data point is assigned to another leaf node.

Another problem is that decision trees are very sensitive to overfitting, especially in small data sets. Small variations in the training data can produce totally different decision trees. In the educational applications the future data very seldom follows the same distribution as the training set and we would need more robust models. For example, Domingos and Pazzani [DP97] recommend to use naive Bayes instead of decision trees for small data sets (less than thousands of rows), even if the attributes were not independent, as naive Bayes assumes.

Often overfitting can be avoided if we learn a collection of decision trees and average their predictions. This approach is generally called *model averaging* or *ensemble learning* (see e.g. [VM02]). In ensemble learning we can combine several models with different structures, and even from different modelling paradigms. The most well-known ensemble learning method is *boosting* [Sch02]. In practice, these methods can improve classification accuracy remarkably and their applicability to educational domain should be further studied.

Finally, we should remark that learning a globally optimal decision tree is an *NP*-complete problem [HR76]. That is why the common decision tree algorithms employ some heuristics and can sometimes produce suboptimal results. One reason for the suboptimal partitions is that each node is split on just one variable [HMS02][347]. In practice, the class variable can change most rapidly with a combination of input variables. As a solution, we can split the input space along linear combinations of the input variables, but this of course complicates the building process. Pruning can also be very time-consuming, when the training set is large. Fortunately, the educational data sets are often small and we do not have to worry about complexity issues.

7.2.2 Decision trees for the *ViSCoS* data

In this example we demonstrate some restrictions of decision trees. We have constructed decision trees $T1$ and $T2$ for predicting final results $FR1$ and $FR2$ in *Prog.1* and *Prog.2* courses. The decision trees are represented in Figure 7.1. The trees have been learnt from the categorial *ViSCoS* data using *Weka* tool [Wek]. In *Weka* decision trees are learnt by the *id3* algorithm [Qui86], which does not prune the trees, and thus we have pruned the decision trees manually afterwards. The same decision trees are represented as classification rules in Tables 7.1 and 7.2. In addition, we have calculated the confidence and frequency for each rule.

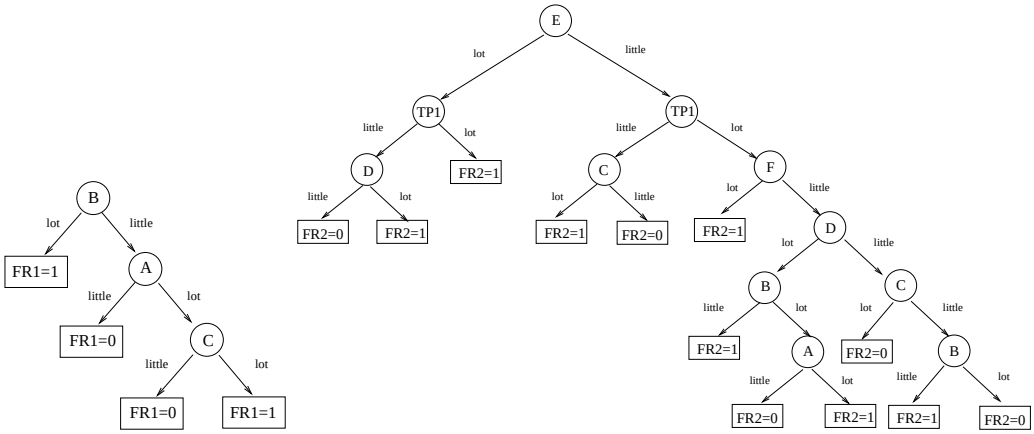

Figure 7.1: Decision trees $T1$ and $T2$ for classifying students in *Prog.1* and *Prog.2* courses. The models have been constructed by *id3* algorithm from categorial *ViSCoS* data.

Both decision trees have quite reasonable training errors. In *Prog.1* course, the classification error is 16% and in *Prog.2* 11%. However, the training error does not tell very much about the real accuracy. Further analysis of frequencies and confidences reveals that the models – especially $T2$ – are not very reliable. In $T1$ only the first two rules are strong and common enough. The latter two rules are useless, because one of them is too weak and the other one is extremely rare. Model $T2$ is so strongly overfitted that it hardly works outside the training set. The only useful rules are $E = lot \wedge TP1 = lot \Rightarrow FR2 = 1$ and $E = little \wedge TP1 = little \wedge C = little \Rightarrow FR2 = 0$. All the other rules are based on just a couple of instances. This is the reason for the counter-intuitive rules in the bottom of the right subtree. In addition, the class labels are decided by the majority vote principle, which gives arbitrary results when both classes are equally common in a leaf node.

It is quite obvious that decision trees do not suit for our modelling task. The course

Table 7.1: *Prog.1* classification rules, their frequencies, and confidences.

Rule	Freq.	Conf.
$B = lot \Rightarrow FR1 = 1$	73/125=0.58	0.92
$B = little \wedge A = little \Rightarrow FR1 = 0$	12/125=0.10	0.86
$B = little \wedge A = lot \wedge C = little \Rightarrow FR1 = 0$	18/125=0.14	0.60
$B = little \wedge A = lot \wedge C = lot \Rightarrow FR1 = 1$	2/125=0.02	1.00

Table 7.2: *Prog.2* classification rules, their frequencies, and confidences.

Rule	Freq.	Conf.
$E = lot \wedge TP1 = lot \Rightarrow FR2 = 1$	33/88=0.38	0.97
$E = lot \wedge TP1 = little \wedge D = lot \Rightarrow FR2 = 1$	1/88=0.01	1.00
$E = lot \wedge TP1 = little \wedge D = little \Rightarrow FR2 = 0$	2/88=0.02	1.00
$E = little \wedge TP1 = little \wedge C = lot \Rightarrow FR2 = 1$	1/88=0.01	1.00
$E = little \wedge TP1 = little \wedge C = little \Rightarrow FR2 = 0$	18/88=0.20	1.00
$E = little \wedge TP1 = lot \wedge F = lot \Rightarrow FR2 = 1$	2/88=0.02	1.00
$E = little \wedge TP1 = lot \wedge F = little \wedge D = lot$ $\wedge B = little \Rightarrow FR2 = 1$	3/88=0.03	0.75
$E = little \wedge TP1 = lot \wedge F = little \wedge D = lot$ $\wedge B = lot \wedge A = little \Rightarrow FR2 = 0$	1/88=0.01	1.00
$E = little \wedge TP1 = lot \wedge F = little \wedge D = lot$ $\wedge B = lot \wedge A = lot \Rightarrow FR2 = 1$	4/88=0.05	0.67
$E = little \wedge TP1 = lot \wedge F = little \wedge D = little$ $\wedge C = lot \Rightarrow FR2 = 0$	4/88=0.05	0.80
$E = little \wedge TP1 = lot \wedge F = little \wedge D = little$ $\wedge C = little \wedge B = little \Rightarrow FR2 = 1$	2/88=0.02	0.50
$E = little \wedge TP1 = lot \wedge F = little \wedge D = little$ $\wedge C = little \wedge B = lot \Rightarrow FR2 = 0$	6/88=0.07	0.67

outcomes are not deterministically determined by the given attributes. In addition, it is very implausible that we could ever model the domain deterministically, whatever attributes we would gain. Of course more discriminatory attributes would improve the accuracy, but at the same time we should prune some of the current attributes to avoid overfitting.

7.3 Bayesian classifiers

We have already introduced Bayesian networks in Chapter 5. In this section, we first analyze different variations of Bayesian networks for classifying educational data. Then we report our experiment on classifying the *ViSCoS* data by naive Bayes classifiers.

7.3.1 General Bayesian models vs. naive Bayes models

The general idea of *Bayesian classifiers* (see e.g. [Mit97][ch 6]) is the following: Given a set of explanatory attributes $X = X_1, .., X_k$, which can have values $\overline{x} \in Dom(X) = Dom(X_1) \times ... \times Dom(X_k)$, we estimate the class-conditional distributions $P(X = \overline{x}|C = c)$ for all $\overline{x} \in Dom(X)$ and all class values c from data. Then, given a new data point $p \in Dom(X)$, we update the class probabilities $P(C = c|p)$ by the Bayes rule.

In practice, the problem is the large number of probabilities we have to estimate. For example, if $|Dom(X_i)| = v$ for all $i = 1, ..., k$ and all X_is are mutually dependent, we have to define $O(v^k)$ probabilities. This means that we also need a large training set to estimate the required joint probability accurately.

Another problem which decreases the classification accuracy of Bayesian networks is the use of *MDL* score function for model selection [FGG97]. *MDL* measures the error in the model over all variables, but it does not necessarily minimize the error in the class variable. This problem occurs especially, when the model contains several attributes and the accuracy of estimate $P(A_1, ..., A_k)$ begins to dominate the score.[1]

The *naive Bayes model* solves both problems. The model complexity is restricted by a strong independence assumption: we assume that all attributes $X_1, ..., X_k$ are conditionally independent, given the class variable C, i.e. $P(X|C) = \prod_{i=1}^{k} P(X_i|C)$.

[1]Let the data be $D = \{p_1, ..., p_n\}$, where $p_i = (\overline{x_i}, c_i)$ and c_i is the class value of point p_i. The minimal description length of model M given data D is

$$MDL(M|D) = \frac{\log n}{2}|M| - LL(M|D),$$

where $LL(M|D)$ is the log likelihood of model given data. Since

$$LL(M|D) = \sum_{i=1}^{n} \log P(c_i|\overline{x_i}) + \sum_{i=1}^{n} \log P(\overline{x_i}),$$

the second term in $LL(M|D)$ begins to dominate over the first one, when the number of attributes increases. However, only the first term evaluates the model as a classifier. [FGG97]

This *Naive Bayes assumption* can be represented as a two-layer Bayesian network (Figure 7.2), with the class variable C as the root node and all the other variables $X_1, ..., X_k$ as leaf nodes. Now we have to estimate only $O(kv)$ probabilities per class. The use of *MDL* score function in the model selection is also avoided, because the model structure is fixed, once we have decided the explanatory variables X_i.

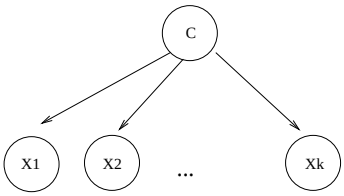

Figure 7.2: A naive Bayes model with class variable C and input variables $X_1, ..., X_k$.

In reality the Naive Bayes assumption holds very seldom, but in practice the naive Bayes classifiers have proved to work well. In fact, Domingos and Pazzani [DP97] have shown that Naive Bayes assumption is only a sufficient but not a necessary condition for the optimality of the naive Bayes classifier. In addition, if we are only interested in the ranked order of the classes, it does not matter if the estimated probabilities are biassed.

As a consequence of Naive Bayes assumption, the representational power of the naive Bayes model is lower than that of decision trees. If the model uses nominal data, it can recognize only linear decision boundaries. However, it cannot recognize even some linear boundaries, for example *m-of-n decision rules*, which require that at least m of n Boolean attributes are true [DP97]. If numeric data is used, then the model can represent even quite complex non-linear boundaries.

Otherwise, the naive Bayes model has many advantages: it is very simple, efficient, robust to noise, and easy to interpret. It is especially suitable for small data sets, because it combines small complexity with a flexible probabilistic model. The basic model suits only for discrete data and the numeric data should be discretized. Alternatively, we could learn a continuous model by estimating densities instead of distributions. However, this latter approach assumes some general form of distribution, typically normal distribution, which is often unrealistic. In addition, discretization simplifies the model and the resulting model is more robust to overfitting.

When X_i attributes are clearly dependent, we would expect that a general Bayesian network classifier would perform better. However, a more complex model is also more sensitive to overfitting, and the actual performance can be even poorer. The problem is that we cannot estimate all parameters accurately in a more complex model; in an extreme case, some attribute combinations do not occur at all in

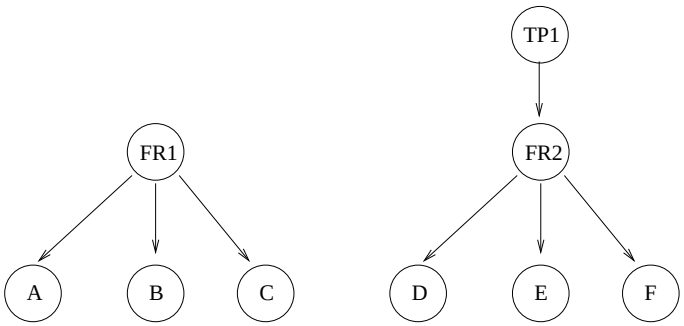

Figure 7.3: Naive Bayes models for predicting final results of *Prog.1* course, given the task points in A, B and C (left), and final results of *Prog.2* course, given the task points in D, E and F (right). $TP1$ (total points of *Prog.1*) has been used as a background variable, which defines the prior probability distribution of $FR2$.

the training set and we have the *zero count problem*. As a solution, Friedman et al. [FGG97] suggest two enlargements of naive Bayes model. In the *tree-augmented naive Bayes model* the X_i attributes can depend on at most one another A_i attribute in addition to the class variable. In the *Bayesian multinet* [GH96] a different, simple-structured Bayesian network is conctructed for each class c_i. According to [FGG97], the classification accuracy especially in small data sets can be further improved by estimating the parameters using Laplace smoothing. In our initial experiments [HV06] the enlargements performed only sligthly better that the naive Bayes model and the Laplace smoothing had no effect on the accuracy.

7.3.2 Naive Bayes models for the *ViSCoS* data

In this experiment we constructed two naive Bayes models, *NB1* and *NB2*, for both *Prog.1* and *Prog.2* courses. The model structures are presented in Figure 7.3 and the parameters are given in Table 7.3.

Nodes $FR1$ and $FR2$ tell the probability of passing/failing the course and the leaf nodes are used to update the probabilities, when exercise task points are gathered. Actually, the exercise task points in different categories depend on each other, but this is not taken into account.

In $NB1$ the prior probability of passing the course is simply assigned to 0.500. In $NB2$ we use $TP1$ as a background variable, and define the prior probability of passing the course given that the student has got a lot/little of total points in *Prog.1*. This proved to be the most frequent rule for predicting $FR2$ according to *Prog.1*

performance. The conditional probabilities have been simply calculated from the data with rule $P(Y|X) = \frac{P(X,Y)}{P(X)}$. The complement probabilities can be obtained by rule $P(\neg Y|X) = 1 - P(Y|X)$.

Table 7.3: Probabilities for naive Bayes models $NB1$ and $NB2$.

$P(FR1)$	0.500
$P(A\|FR1)$	0.955
$P(A\|\neg FR1)$	0.694
$P(B\|FR1)$	0.820
$P(B\|\neg FR1)$	0.167
$P(C\|FR1)$	0.416
$P(C\|\neg FR1)$	0.028

$P(FR2\|TP1)$	0.727
$P(FR2\|\neg TP1)$	0.091
$P(D\|FR2)$	0.700
$P(D\|\neg FR2)$	0.184
$P(E\|FR2)$	0.680
$P(E\|\neg FR2)$	0.079
$P(F\|FR2)$	0.340
$P(F\|\neg FR2)$	0.026

When the course proceeds, the prior probabilities are updated in the light of new evidence (exercise points in $X = A, B, C, D, E, F$) by the Bayes rule:

$$P(FR|X) = \frac{P(FR) \times P(X|FR)}{P(X)}.$$

The classification rates after each new piece of evidence are presented in Table 7.4. The classification rates have been calculated by *zero-one loss rule*, i.e. the student's status is classified as *passing*, if the passing probability is ≥ 0.5, and *failing*, otherwise. The actual probabilities contain of course more information, but they are no more comparable with the deterministic classifiers. The accuracies are determined by 10-fold cross-validation.

In the beginning of *Prog.1* course nearly all students had done a lot of exercises and thus they were predicted to pass the course. However, when the exercise points in B category were added, the predictions became already quite accurate, both for passing and failing. C points did not improve the results much, because only few students had done a lot of those tasks.

In *Prog.2* course, we can predict the outcomes already before the course has begun, based on *Prog.1* outcomes. These predictions are already surprisingly good – better than *Prog.1* predictions, when A points were known. An important observation is that we can predict the failed students very well, when just $TP1$ and D points are known. In fact, these predictions do not improve any more, when E and F are added to the model. With F points we recognize a strange phenomenon: the classification accuracy slightly decreases, even if we have got more evidence! This is totally

Table 7.4: Classification accuracies of naive Bayes models. The accuracies have been defined by 10-fold cross-validation.

Model	true pos.	true neg.
$A \Rightarrow FR1$	0.96	0.31
$A, B \Rightarrow FR1$	0.80	0.81
$A, B, C \Rightarrow FR1$	0.83	0.81
$TP1 \Rightarrow FR2$	0.96	0.53
$TP1, D \Rightarrow FR2$	0.76	0.61
$TP1, D, E \Rightarrow FR2$	0.82	0.87
$TP1, D, E, F \Rightarrow FR2$	0.80	0.87

correct, because F and E are highly correlated and thus the Naive Bayes assumption is clearly violated. This is the only case, when we have met the restrictions of naive Bayes model, and in practice these last predictions (when the course is over) are not so important. However, this demonstrates that the naive Bayes model should be used with care, when the attributes are correlated.

7.4 Neural networks

Artificial neural networks are commonly used in pattern recognition. They have achieved good results in solving complex problems with noisy data. In addition, they have higher representational power than other classifiers, and in principle any other system can be implemented by a neural network. Thus, it is natural to question whether they could solve all problems in the educational domain, too. Our answer is conditional "No", for the reasons explained in the following.

Feed-forward neural networks (*FFNN*) or *Feed-forward multilayer perceptrons* (see e.g. [DHS00][ch 6]) are the most widely used type of neural networks. The *FFNN* architecture consists of layers of nodes: one for input nodes, one for output nodes, and at least one layer of *hidden nodes*. On each hidden layer the nodes are connected to the previous and next layer nodes and the edges are associated with individual weights. The most general model contains only one hidden layer (Figure 7.4). This is usually sufficient, because in principle *any function can be represented by a three-layer network*, given sufficiently many hidden nodes [HN89, Cyb89]. This implies that we can also represent any kind of (non-linear) decision boundaries. However, in practice learning a highly non-linear network is very difficult or even impossible. For linearly separable classes it is sufficient to use a *FFNN* with no hidden layers. This kind of neural network is known as a *perceptron* [DHS00][ch 5].

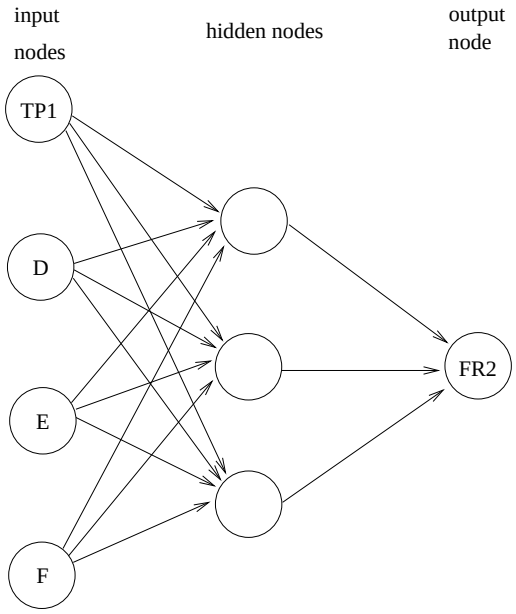

Figure 7.4: An example of *FFNN* structure of input nodes, hidden nodes, and an output node on successive layers. The model was constructed for predicting the final results in *Prog.2* course, given attributes $TP1, D, E$ and F.

The learning algorithm is an essential part of the neural network model. In *FFNN* the *back-propagation algorithm* [DHS00][288-296], based on *gradient descent search* is used. Since this algorithm is an essential part of the model, we will describe it briefly.

In classification we construct a network of k input nodes, one for each attribute $X_1, ..., X_k$, and l output nodes for l class values [2], $C = c_1, ..., c_l$. The output nodes can be interpreted as binary variables $C = c_i$, having values $[0, 1]$. These values as such tell the confidence on each class, but with appropriate modifications they approximate posterior probabilities $P(C|X)$ [DHS00][304-305]. The number of hidden nodes is selected and the weights are initialized. After that the training samples are fed to the first layer nodes, one by one. Each node computes a weighed sum of input values and performs a non-linear transformation, typically *sigmoid function* $f(x) = 1/(1 + e^{-x})$. The output values are further fed to the next layer, which performs the same operations, until the output nodes are reached. Then the output values are compared to the actual class value and the errors are computed. The errors are propagated backwards through the network and all weights are updated. This procedure is repeated, often several times for the same training examples, until

[2] For a binary class variable just one output node is sufficient.

some termination criterion is met.

This algorithm leaves several parameters open and the user should select the number of hidden layers, number of hidden nodes on each layer, initial weights, and the termination criterion. Especially the selection of the architecture (network topology) and the termination criterion are critical, because neural networks are very sensitive to overfitting. Unfortunately, there are no foolproof instructions and often the correct parameters are defined by trial and error. However, there are some general rules of thumb, which restrict the number of trials needed.

Duda et al. [DHS00][317] suggest to use a three-layer network as a default and add layers only for serious reasons. The number of hidden nodes depends on the problem: if the classes are linearly separable or otherwise well-separated, less nodes are needed, while overlapping classes require more nodes. However, we should also take into account the model complexity. The number of nodes determines the number of weights and thus the model complexity. As a rule of thumb, it is often recommended (e.g. [DHS00]) that the number of weights should be roughly $n/10$ for a training set of size n. If we have k input variables and one class node, then the number of hidden nodes should be approximately $\frac{n}{10(k+1)}$. For example, to model the *Prog.2* data (using four input variables) we could use at most two hidden nodes.

The initial weights are usually selected randomly. It is recommended (e.g. [Mit97][104,111]) to assign small positive values, which define a linear model. When the weights increase, the model becomes more and more non-linear. In this way, we search the simple models first, which can prevent overfitting. Another important decision is when to stop learning. If we stop it too early, the model cannot classify the training set accurately, but if we let it specialize too much, it does not generalize any more. Use of separate test set is a popular strategy [Mit97][111], but with small data sets it is not possible, and cross-validation should be used instead.

Feed-forward neural networks have several attractive features. They can easily learn non-linear boundaries and in principle represent any kind of classifiers. If the original variables are not discriminatory, *FFNN* transforms them implicitly. In addition, *FFNN*s are robust to noise and can be updated with new data.

However, neural networks have several disadvantages from our point of view. The main restriction is that they would need a lot of data – much more than the typical educational data sets. They are very sensitive to overfitting and the problem is even more critical with small training sets. The data should be numeric and categorial data must be somehow quantized, before it can be used. However, this increases the model complexity and the results are sensitive to the quantization method used.

The neural network model is a black box and neither teachers nor the students can understand the explanations for the outcomes. This is in serious contradiction

with the requirement of transparency. In addition, neural networks are unstable and achieve good results only in good hands – the user should have expertise to define all open parameters. This is partly the reason for the variable performance in comparative tests, as Duin [Dui00] observes: "*What effectively is evaluated is not the neural networks as such, but the skills of the data analyst in using the given neural network toolbox*". Finally, we should remember that finding an optimal *FFNN* is an *NP*-complete problem [BR88] and back-propagation algorithm is just an optimization method, which can get stuck at a local optimum. Still, the training can be time consuming, especially if we want to circumvent overfitting.

Example 11 *The* FFNN *described in Figure 7.4 was implemented by* Matlab *for predicting the final results in* Prog.2 *course, given attributes* $TP1, D, E, F$. *The model was tested with different initial parameters. The activation function was sigmoid. The model proved to be very unstable: the classification accuracy measured by cross-validation varied between 65% and 85% in consecutive executions with the same parameter assignment. Other model topologies for the* Prog.2 *data as well as* Prog.1 *models produced similar results. We conclude that the data is either unsuitable to be modelled by neural networks or – refering to Duin [Dui00] – the data analyst's skills were not sufficient.*

7.5 k-Nearest neighbour classifiers

k-Nearest neighbor classifiers (see e.g. [HMS02][347-352], [Mit97][231-236]) represent a totally different approach to the classification problem. They do not build any explicit global model, but approximate it only locally and implicitly. The main idea is that we classify a new object by examining the k most similar data points and making the class decision according to their class values. Usually we select the most common class among the neighbours, but alternatively we could compute the class probabilities in the neighbourhood.

Two practical issues concern how to select value k and a metric which defines the "closeness". Selecting appropriate k can be difficult and often we have to try several values. If k is fixed, then the size of the neighbourhood varies. In sparse areas the nearest neighbours are more remote than in dense areas. However, defining different ks for different areas is even more difficult. If k is very small, then the neighbourhood is also small and the classification is based on just a few data points. As a result the classifier is unstable, because these few neighbours can vary a lot. On the other hand, if k is very large, then the most likely class in the neighbourhood can deviate much from the real class. For small dimensional data sets a suitable k is usually

between 5 and 10. One solution is to weigh the neighbours by their distances. In this case, the neighbourhood can cover all data points so far and all neighbourhoods are equally large. The only disadvantage is that the computation becomes slower.

Defining the distance metric is another problem, which we have already discussed in Chapter 6. The problem in classification is that the metrics usually take into account all attributes, even if some attributes were irrelevant. This is a big problem, because the actually most similar neighbours become remote and the "wrong neighbours" corrupt the classification. The problem becomes more serious, when the number of dimensions and thus also the number of irrelevant attributes grows. As a solution, it has been suggested (e.g. [HAK00])to give relevance weights for attributes, but the relevant attributes can also vary from class to class. In practice, appropriate feature selection can produce better results.

The nearest neighbourhood classifiers have several advantages: they are easy to implement, no optimization or training is required, the classification accuracy can be very good in some problems, and they are quite robust to noise and missing values. Especially weighed distance smooths the noise in attribute values and missing values can be circumvented by restricting to the available attributes. Nearest neighbour classifiers have very high representational power, because they work with any kind of decision boundaries, given sufficiently data.

However, nearest neighbour classifiers have also serious disadvantages and the applicability in the educational domain is restricted. First of all, they require large data sets to work accurately. In addition, they are very sensitive to the curse of dimensionality. The problem is that in high dimensions the input space becomes sparse and the nearest neighbors are often far. Thus it is hard to recognize real neighbours from other points and the predictions become inaccurate.

The lack of an explicit model can be either an advantage or a disadvantage. If the model is very complex, it is often easier to approximate it only locally. In addition, there is no training and a set of data is immediately ready for use. We can always add and delete data rows without any modifications. However, this kind of "lazy methods" are slower in classification than model-based approaches. If the data set is large, we need some index to find the nearest neighbours efficiently. It is also noteworthy that an explicit model is useful for human evaluators and designers of the system.

7.6 Comparison of classification methods

Selecting the most appropriate classification method for the given task is a difficult problem and no general answer can be given. In this section, we evaluate the previously introduced classification methods according to the general requirements of educational systems. In addition to decision trees, naive Bayes classifiers, feed-forward neural networks and nearest neighbour classifiers, we have included multiple linear regression, which can also be used as an classifier for simple problems. The comparison is represented in Table 7.5. The analysis is based on several sources, e.g. [DP97, Mit97, DHS00, HMS02, JDM00, Dui00, Qui94].

The first criterion concerns the *form of decision boundaries*. Decision trees, $FFNN$s and nearest neighbour classifiers can represent highly non-linear boundaries. Naive Bayes model using nominal data can represent only a subset of linear boundaries, but with numeric data it can represent quite complex non-linear boundaries. Multiple linear regression is restricted to only linear boundaries, but it tolerates small deviations from the linearity. It should be noticed that strong representational power is not desirable, if we have only little data and a simpler, linear model would suffice. The problem is that complex, non-linear models are also more sensitive to overfitting.

The second criterion, *accuracy on small data sets* is crucial for the educational domain. The classifier should be learnt from a very small training set (around 100 rows of data) and still generalize well. This favours simple models, especially naive Bayes classifiers and their variations. Multiple linear regression can also work accurately, if the data has at least approximately linear tendency. On the other hand, decision trees, *FFNN*s, and nearest neighbour classifiers require much larger data sets (at least a thousand of rows of data, depending on the number of attributes) to work accurately.

The third criterion concerns whether the paradigm can handle *incomplete data*, i.e. noise and missing values. Educational data is usually clean, but missing values occur frequently e.g. in the questionnaire data. Naive Bayes, *FFNN*s, and nearest neighbour models are especially robust to noise in the data. Naive Bayes, nearest neighbour models, and some enlargements of decision trees can handle also missing values quite well. However, decision trees are generally very sensitive to small changes like noise in the data. Multiple linear regression cannot handle missing attribute values at all and serious noise (outliers) can corrupt the whole model.

Natural interpretation of the model is also an important criterion, since all educational models should be transparent. All the other paradigms except neural networks offer more or less understandable models. Especially tree structures (decision

Table 7.5: Comparison of different classification methods. Sign + means that the model supports the property, − that it does not. The abbreviations are $FFNN$= feed-forward neural network and LR= multiple linear regression.

	Decision trees	Naive Bayes	FFNN	Nearest neighbour	LR
Non-linear boundaries	+	(+)	+	+	−
Accurate on small training sets	−	+	−	−	+
Works with incomplete data	−	+	+	+	−
Works with mixed variables	+	+	−	+	−
Natural interpretation	+	+	−	−	+
Efficient reasoning	+	+	+	−	+
Efficient learning	+/−	+	−	No learning	+
Efficient updating	−	+	+	−	+

trees and naive Bayes) have a comprehensive visual representation.

The last criteria concern the *computational efficiency* of classification, learning, and updating the model. The most important is efficient classification, because the system should adapt to the learner's current situation immediately. For example, if the system offers individual exercises for learners, it should detect when more easier or challenging tasks are desired. Nearest neighbour classifier is the only one which lacks this property. The efficiency of learning the model is not so critical, because in our paradigm it is done only once after the course. It should be noticed that the nearest neighbour methods do not build an explicit model and the learning criterion is irrelevant. In some paradigms the models can be efficiently updated, given new data. This is an attractive feature for the *ViSCoS* project, because we can add the newest drop-outs into the data set, when the course results are still unknown.

An important observation is that there is always a wrestling between efficiency and accuracy: with suitable independence assumptions and approximation techniques we can speed up the model learning, but at the cost of accuracy. Efficiency and accuracy also depend on the specific data set: some methods work very well with low-dimensional data, but are intractable or poor with high-dimensional data. This means that we can suggest only some guidelines concerning the efficiency and accuracy of the given method.

7.7 Empirical comparison of classifiers for the *ViSCoS* data

In this section we compare two of the most promising candidates for classifying the *ViSCoS* data, multiple linear regression and naive Bayes models. The empirical comparison has been reported in [HLS06]. In the same paper we demonstrate, how results of descriptive analysis can be utilized in constructing classifiers. A similar approach has been reported in [LHM98], where decision rules were inferred from association rules.

Because generalization ability is more important than a small training error, we have performed 10-fold cross-validation to estimate the classification accuracy on new data points. The classification accuracy is estimated by true positive (TP) and true negative (TN) rates. The average TP and TN rates for all linear regression (LR) and naive Bayes (NB) models are presented in Table 7.6. To evaluate the general accuracy, we have calculated the *relative operation characteristic* (ROC) score, $ROC = \frac{TP}{1-TN}$ (Table 7.7).

An important option is that we should be able to classify the course outcomes as early as possible. Thus, the most important model structures are $A \rightarrow FR1$, $A, B \Rightarrow FR1$, $TP1 \rightarrow FR2$, $TP1, D \Rightarrow FR2$, and $TP1, D, E \Rightarrow FR2$. In the end of the course (when all task points are known) the predictions are of course more accurate, but it does not benefit so much any more.

Table 7.6: Comparison of prediction accuracy of LR and NB models. The prediction accuracy is expressed by the true positive TP and true negative TN rates. All models have been evaluated by 10-fold cross-validation and the classification rates have been averaged.

Model structure	LR rates		NB rates	
	TP	TN	TP	TN
$A \Rightarrow FR1$	0.83	0.47	0.96	0.31
$A, B \Rightarrow FR1$	0.91	0.72	0.80	0.81
$A, B, C \Rightarrow FR1$	0.93	0.81	0.83	0.81
$TP1 \Rightarrow FR2$	0.70	0.68	0.96	0.53
$TP1, D \Rightarrow FR2$	0.78	0.84	0.76	0.61
$TP1, D, E \Rightarrow FR2$	0.76	0.89	0.82	0.87
$TP1, D, E, F \Rightarrow FR2$	0.70	0.92	0.80	0.87

The most interesting and encouraging result is that both modelling paradigms were able to predict the course performance for more than 80% of students, when the

Table 7.7: *ROC* scores for *LR* and *NB* models.

Model structure	LR	NB
$A \Rightarrow FR1$	1.566	1.391
$A, B \Rightarrow FR1$	3.250	4.211
$A, B, C \Rightarrow FR1$	4.890	4.368
$TP1 \Rightarrow FR2$	2.188	2.043
$TP1, D \Rightarrow FR2$	4.875	1.949
$TP1, D, E \Rightarrow FR2$	6.909	6.308
$TP1, D, E, F \Rightarrow FR2$	8.750	6.154

course was still on. This is especially surprising in the *Prog.1* models, where no previous information is available and the predictions are totally based on exercise points. In the *Prog.2* models the *Prog.1* performance is already known. LR and NB models $TP1, D, E \Rightarrow FR2$ give especially valuable information, because drop-out and failing are bigger problems in *Prog.2* course and these models are able to predict the outcomes when only three weeks of the course has passed.

An important observation is that both modelling paradigms suffer from strong correlations between B and C and E and F. Especially, adding the F attribute to the previous model decreases the prediction accuracy in both LR and NB models.

When we compare the *ROC* scores, we observe that LR models have a better classification accuracy in all models but $A, B \rightarrow FR1$. Linear regression outperforms naive Bayes especially in models $TP1, D \rightarrow FR2$ and $TP1, D, E, F \rightarrow FR2$. This is mainly due to NB's over-simplified model structure – the attributes contained only binary-valued information (whether a student had got a little or a lot of points in a given category). In addition, the NB models can suffer for poor discretization of numeric data. The *Prog.1* data set contains 14 inconsistent data points and the *Prog.2* data set 12 inconsistent data points. This means that the upper bound for classification accuracy in $A, B, C \rightarrow FR1$ is 0.89 and for $TP1, D, E, F \rightarrow FR2$ 0.86 and no model can achieve better accuracy.

In the future we will study better discretization methods and enlargements of naive Bayes models. Naive Bayes models are attractive, because they are more general than the multiple linear regression and we expect them to adapt better to a new course setting with different tasks and maximum points. In addition, they produce probabilities, which are more informative than deterministic predictions.

According to our initial tests with those students who had filled the course prerequisite questionnaire, we expect even better classification accuracy in the future. Currently only 60% of students had filled the query, but we could find strong de-

pendencies between the previous programming skills and the course performance in both courses. Another important factor that should be queried is the student's knowledge in mathematics, which is known [Pag03, Dev03] to have a strong correlation with the programming skills.

Chapter 8

Implementing adaptivity in learning environments

In the previous chapters we have discussed methods for different modelling tasks. We have applied the techniques to the *ViSCoS* data, which contains only exercise points and final results. Now it is time to combine all these techniques and give general principles for implementing adaptivity in learning environments. The main task is to detect the learner's current state (specific requirements and preferences) and select the most appropriate action. The data set is much larger and diverse than in the previous applications: for a fully adaptive system we have to gather more data about the students, their previous knowledge, and preferences. The system log, which contains the whole action history, is also a valuable source of information. In addition, we assume that the system will be equipped with different kind of sensors, which offer a flexible user interface but also more information for adaption.

In the following we will first introduce the general framework of context-aware computing. Then we introduce three general principles for implementing adaptivity in practice. The first two principles describe action selection in two situations. In the first case we infer first the user's situation and then select the most appropriate action, while in the second case we infer the action directly by social filtering. The third principle describes how temporal information can be utilized in adaption.

8.1 Adaption as context-aware computing

To restrict the future development of adaptive learning environments as little as possible, we have adopted a wide (and visionary) view of *context-aware computing*

(*ubiquitous computing*, *ambient intelligence*) (see e.g. [DA99, CK00]), in which the whole context – the user, her/his actual situation and all relevant information – is used for determining the most appropriate action. In the traditional learning systems this information is gathered directly by the application, but in addition several sensors can used.

The common personal computer can be equipped with microphone, camera, light-pen, data-glove, etc. which recognize the user and observe her/his interaction with the system. There are already applications which analyze voice and camera images to infer the user's current task, mood, or intention [SSP98, TSV03, SYW01].

Mobile devices are especially useful, because they are carried by the user. We can get user's location and near-by people by GPS (global positioning system). This is a useful property for example in mobile learning. Indoors, the location can be recognized by a network of ultrasonic or radio beacons. The change of orientation (movements) can be recognized by inertial sensors (acceleration and rotation), motion sensors (change of motion) or camera. IR (infra-red) sensor reveals proximity of humans or other warmth sources. Light level, temperature, pressure and CO gas can be measured by simple sensors. In special education we can even utilize biological sensors, which measure pulse, blood pressure, and body temperature. For example, dyslexia is affected by stress and emotional factors and it often causes secondary symptoms like increase of body temperature.

Contexts are usually divided into *primary* and *secondary* contexts. The primary or low-level context means the environmental characteristics which can be observed directly by sensors: location, time, nearby objects, network bandwidth, orientation, light level, sound, temperature, etc. They can either measure the *physical* parameters in the environment or *logical* information gathered from the host (e.g. current time, GSM cell, selected action) and the sensors are called physical or logical, correspondingly [SAT+99]. The secondary or high-level context means a more abstract context, which is derived from the primary context: the user's social situation, current activity, mental state, etc.

However, this division is quite artificial. The features which are secondary contexts for some applications, can be further processed and combined to offer more high-level contexts for the other applications. Thus, we have adopted a hierarchical view (Figure 8.1), in which we have a continuum of contexts in different abstraction levels. In small systems there may be no need to separate sensor processing and context inference from the actual application, but generally it is more efficient to separate these tasks. If the same contexts are used in several applications, the sensor processing and extracting low-level contexts can be managed in one place. This also reduces the data overload, when preprocessing and compressing the data has been done on the low level.

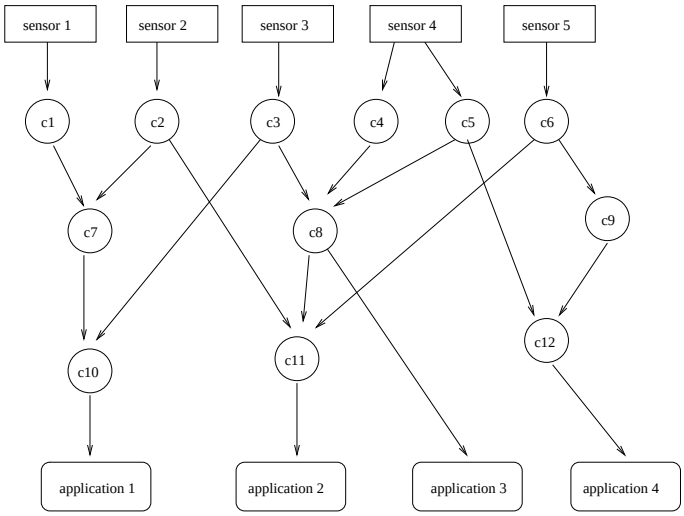

Figure 8.1: A hierarchy of contexts between sensors and applications.

In adaptive learning systems the hierarchical view is especially appropriate. The special nature of educational applications is manifested only in the highest levels, where we should define the high-level contexts and select the most suitable actions. Most of data is originated from logical sensors and thus already higher-level data, which does not require preprocessing. Typically we have small but clean data sets containing discrete valued data. The physical sensors offer also valuable information, but their processing does not differ from other applications and we can use general context servers, which preprocess the sensor data and extract the low-level contexts.

8.2 Action selection

The main problem is the following: We are given a set of contexts, $C = \{c_1, ..., c_k\}$ and a set of actions $A = \{a_1, ..., a_m\}$. The task is to determine the most appropriate action, a_j, in the given situation described by data point x. In the ideal case the current situation x corresponds to one of the predefined contexts c_i and it is enough to select the optimal action, given c_i. However, in practice the situation may contain features from several contexts, but it does not fit any of them precisely.

There are basically two approaches: either we first infer the current context and then select the action in it or we select the most appropriate action directly. The first approach is preferable in most educational applications, because the context contains also important information and selecting the "best" action is not so straightforward.

The latter approach, known as *social filtering* or *collaborative filtering*, suits for situations, where we do not have any previous data and we can trust the users' ability to select the best actions.

8.2.1 Inferring the context and selecting the action separately

In the first approach we assume that the set of possible contexts, $C = \{c_1, ..., c_k\}$, is known. If it is unknown, we can cluster the data and identify the main clusters as contexts. The user's current context is determined by classification, and the action is selected according to a given set of rules. Both classification and rule selection can be performed either in a deterministic or probabilistic way. Deterministic approach is used in traditional intelligent tutoring systems, although the classifiers are seldom learnt from data. In some systems the actions are selected by probabilistic rules, but the rules are predefined by system designers.

When the classifiers and action selection rules are learnt from the data, probabilistic models are more realistic. In this approach the classifier produces a probability distribution of contexts, $P(C = c_i|x), ..., P(C = c_k|x)$, given data point x which describes the current situation. The actions are selected by probabilistic rules of form "*If context is* c_i*, then select action* a_j *with probability* p". These can be interpreted as conditional probabilities $P(A = a_j|C = c_i) = p$, which give the probability of an action to be most appropriate or desired in context c_i.

Example 12 *Let us have three possible contexts* $C = \{a, b, c\}$ *and three possible actions* $A = \{action1, action2, action3\}$*. For example, a, b, and c correspond to different reasons for an incorrect task solution, like a="Student is frustrated", b="The task is too difficult", c="The student is missing essential knowledge". The actions could be e.g.* $action1$*="Show and explain the correct solution",* $action2$*="Generate a new, easier task", and* $action3$*= "Give support material for reading".*

The current situation (data point x) has been classified probabilistically with values $P(C = a|x) = 0.5$*,* $P(C = b|x) = 0.3$*,* $P(C = c|x) = 0.2$*. I.e. context a is the most probable one and would have been selected in the discriminative approach. The associated actions and their probabilities are*

$a \rightarrow action1$ (0.6), $a \rightarrow action2$ (0.4)
$b \rightarrow action3$ (0.6), $b \rightarrow action2$ (0.4)
$c \rightarrow action3$ (0.6), $c \rightarrow action2$ (0.4).

The deterministic policy would select $action1$*, which is the most appropriate in the most probable context (i.e. if the student is frustrated, the correct solution is*

shown). However, the other contexts do not support this action. If we calculate the probabilities for all actions we get:

$P(A = action1) = 0.5 \times 0.6 = 0.3$
$P(A = action2) = 0.5 \times 0.4 + 0.3 \times 0.4 + 0.2 \times 0.4 = 0.4$
$P(A = action3) = 0.5 \times 0.2 + 0.3 \times 0.6 + 0.2 \times 0.6 = 0.3.$

*So, in fact action*2 *(generate a new, easier task) is most probably the desired one, and action*1 *and action*3 *are equally probable.*

The best classification method is problem-dependent, but according to our analysis (Table 7.5), simple Bayesian networks are most suitable for educational data. Bayesian networks suit well for classifying high-level contexts, because the data is typically discrete and the attribute domains are small. Another attractive feature in Bayesian networks is that we can combine the probabilistic action selection rules into the same model with the classifier.

There are several ways to learn the probabilistic rules for action selection. Typically the personalized systems require a learning phase, where the user has to teach the system. Often, the user is asked to fill an explicit query to give initial data for the system, but most users prefer to teach the system on-line, when it is actually used. That is why we will now concentrate on such *continuous* or *on-line learning* (see e.g. [MCF+94]).

In the beginning, the user has to do everything manually and the system is only observing and recording varying contexts and actions selected. When the learning proceeds, the system begins to make suggestions. The learning continues, but becomes the more invisible the more data the system has collected. Finally, the system becomes so adapted that the user has to intervene only in exceptional situations.

The main problem of this approach is that it cannot adapt to new situations. Explicit teaching is worksome and continuous learning takes time. In addition, the system cannot suggest new actions, which may be more appropriate than the previously selected ones. Especially in educational applications the user may not select her/himself the best actions for learning.

One solution is to set default values for actions. These default values can be defined by system designers or learnt from previously collected action history. The best way is to collect first data from test use and construct a descriptive model, which reveals if there are typical selections in some situations (contexts). The simplest way is to search association rules between high-order contexts and actions. This also reveals the frequency of contexts (i.e. the most common and the rarest contexts), in addition to most typical actions. In educational applications we may also want

to favour actions which produce good learning outcomes. For this purpose we can select only those students' data who have performed well compared to their starting point.

8.2.2 Selecting the action directly

Social information filtering (see e.g. [SM95]) offers a way to select the most appropriate action directly. In this approach we utilize the information about other users' actions and select the action which has been appropriate for other users in the similar situations.

Social filtering methods are nowadays very popular in recommendation systems, for example proposing personally selected learning material [CLC05, Lee01, PGMK02]. The idea is to recommend new items to the user based on other, similar users' preferences. Typically, the similar users are determined by comparing user profiles. An initial profile may be constructed according to an explicit query or it can be learnt from the user's previous selections. The user profiles can be compared in several ways, for example by computing the *mean square difference* or *Pearson correlation* of profiles. It should be noticed that we are partially clustering the users by defining the *nearest neighbours*.

A similar method is used in *HITS* algorithm [Kle99] for searching most relevant Internet pages. In *HITS* the pages are assigned *hub* and *authority* values, according to how good pages (authorities) they refer and how good pages (hubs) they are referred by. The principle idea is that good hubs refer to good authorities and good authorities are referred by good hubs. There are several ways to select the initial set of pages, V. A common way is to use another (content-based) search engine results and add all pages referred by them and some or all of the pages referring to them. The links construct a graph structure with edges $E = \{(p, q) \mid p, q \in V,\ p \text{ refers to } q\}$.

In the beginning we initialize hub values $x(p)$ and authority values $y(p)$ for each page $p \in V$ such that $\sum_{p \in V} x(p)^2 = \sum_{p \in V} y(p)^2 = 1$. This normalization is repeated after each update cycle. The hub and authority values are updated by equations

$$\begin{aligned} x(p) &= \textstyle\sum_{(p,q) \in E} y(q) \quad \text{(sum of authorities pointed by } p\text{)} \\ y(p) &= \textstyle\sum_{(q,p) \in E} x(q) \quad \text{(sum of hubs pointing to } p\text{)}. \end{aligned}$$

I.e. the hub values are updated according to the authority values and the authority values according to the hub values, until the system converges and the best authorities are output.

The *HITS* method can be easily applied to other recommendation systems, too. For example, Chen et al. [CLC05] have introduced a system, which recommends the learner material, which corresponds to her/his knowledge level.

Example 13 *Let us consider a system, in which students collect and recommend good learning material for each other. We can define good material such that is recommended by competent readers and competent readers such who recommend good material. When we should recommend new material for a student, we construct an initial graph from the material liked by the student her/himself, all other students who have liked the same material, and other material liked by them. Material is associated an authority value, according to the students who recommend it, and students are associated hub values, according to which material they recommend. Then we can run the HITS algorithm and output the best material with the highest authority values.*

The social filtering methods are especially useful, when we should define the most appropriate action in a new situation. The idea is to find similar contexts (similar users and/or similar situations) and select the most popular action in them. The profile consists of lower-level contexts $C_1, ..., C_n$ (one of them possibly indicating the user) with their assigned values. The similarity function can be simply the mean squared difference or we can give more impact on some elements, e.g. favour the profiles of the same user.

It should be noticed that these implicitly defined high-level contexts are very narrow: we assume that each context contains only one action. It is very hard to select an action, if a procedure of several actions is performed in one context. In such cases we should first compose the atomic actions into larger procedures.

In the application of the *HITS* algorithm we cluster the contexts only according to their actions. First, we construct a graph of current context, all actions selected in it, all other contexts in which the same actions are used, and all their actions. This set may be too large and we can prune it according to lower-level contexts. The action values are initialized by the number of contexts in which they were selected and the contexts are initialized by the number of actions used in them. The values are normalized and updated by the normal manner, until the system converges. Finally, the action or actions with the highest values are selected.

This system is easy to implement, it works in new situations, and quite probably it pleases the user. But in educational environments we have one special problem: what pleases the student, it not always the same as the best action. For example, if a student has tendency for laziness, the system may recommend her/him to sleep in the lecture, like the other similar students have done. Thus we should define the

goodness of an action very carefully. This requires existing data about students' actions and learning performance. For a lazy student, the best actions would be such which have activated other similar students and led to good results. Thus, we should give weight according to students' success.

8.3 Utilizing temporal information in adaption

In learning environments we often have log data, which offers extra information for context inference. Depending on the application and individual user, the context changes can follow some temporal patterns. For example, after a lecture the student goes for a coffee break and wants to switch the system off. After loading a new task, s/he wants to read the related lecture slides, before solving the task.

The simplest approach is to use the data about the user's previous contexts in addition to currently observed data for predicting the current context. This can be achieved by *hidden Markov models* (*HMM*) (see e.g. [DHS00][128-139]). In the following we give instructions how to apply hidden Markov models in adaptation, how to learn them from educational log data, and represent all variables in the form of dynamic Bayesian networks.

8.3.1 Hidden Markov model

Definition 28 (1st order Hidden Markov model) *Let* $C = \{C_1, ..., C_k\}$ *be a set of binary-valued contexts and* $O = \{O_1, ..., O_l\}$ *a set of observed variables. The* 1st order Hidden Markov model *is a four-tuple* $(C, O, \theta_1, \theta_2)$*, where* θ_1 *describes the transition probabilities* $P(C_i|C_j)$*,* $i, j = 1, ..., k$*, and* θ_2 *describes the observation probabilities* $P(O_i|C_j)$*,* $i = 1, ..., l$*,* $j = 1, ..., k$*.*

In hidden Markov models it is assumed that the contexts cannot be observed directly, but we have to infer them from observations. That is why the variables $C_1, .., C_k$ are called *hidden*. This is especially feasible assumption in learning environments, where the contexts often describe unobservable variables like the student's skills or motivation. The log data contains only observable variable values $O_1(t), ..., O_l(t)$ at each time step t.

When we learn a hidden Markov model, we assume that at each time step the user is in one context, $C(t)$, and the value of variables O at time step t, $O(t)$, depends on only the current context $C(t)$. In addition, we assume that the parameters θ_1 and θ_2 are constant for all time steps. Given only the number of contexts, k, the

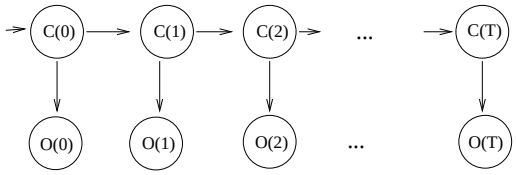

Figure 8.2: A Markov chain of contexts $C(0), ..., C(T)$ with associated observations $O(0), ..., O(T)$.

task is to learn parameters θ_1 and θ_2 from the data. This is a difficult problem and in practice we restrict the model space by additional assumptions. In the 1st order *HMM* we assume the *Markov property* that the current state $C(t)$ depends on only the previous state $C(t-1)$:

$$P(C(t)|C(0), ..., C(t-1)) = P(C(t)|C(t-1)).$$

In the 1st order *HMM* the data can be represented as a *Markov chain* (Figure 8.2). In the kth order *HMM* we generalize the Markov property and assume that the current state depends on the previous k states, i.e.

$$P(C(t)|C(0), ..., C(t-1)) = P(C(t)|C(t-k), ..., C(t-1)).$$

Hidden Markov models can be generalized by allowing dependencies between observation variables (lower level contexts), e.g. $O(t)$ depends on k previous observations, $O(t-1), ..., O(t-k)$, in addition to the current context $C(t)$. When the goal is to select the most appropriate action, it is useful to add action variables A and dependencies between them into the Markov model like proposed in Figure 8.3. This model takes into account actions which typically proceed each other. It is especially useful, when a context is associated by a procedure of actions, instead of a single action. Once again it is possible that the actions depend on several previous actions and a higher order model is needed.

Example 14 *Let us consider a simple learning environment, in which students can solve tasks, read learning material, write learning diary, and communicate with other students. Thus, we have four high-level contexts and each of them can be accompanied by several actions. When we have analyzed the log data, we have found the following patterns: Students check the material very often, when they are doing something else. Reading and communication with other students are typically related, but the order is not fixed. When the student is writing a learning diary, s/he very seldom communicates with other students. According to this information*

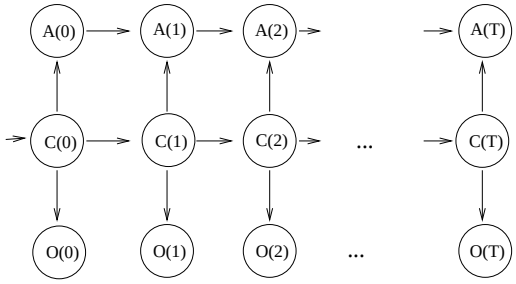

Figure 8.3: A Markov chain of contexts $C(0), ..., C(T)$ with associated observations $O(0), ..., O(T)$ and actions $A(0), ..., A(T)$. Now the actions depend on previous actions, in addition to current context $C(t)$.

and their related probabilities, we have constructed a HMM *model in Figure 8.4. Notice that we have not defined explicit time steps, but only the relative order of contexts. A new action constitutes a new time step, but the length of time steps can vary freely. The graph structure is complete, because the student can always enter any context.*

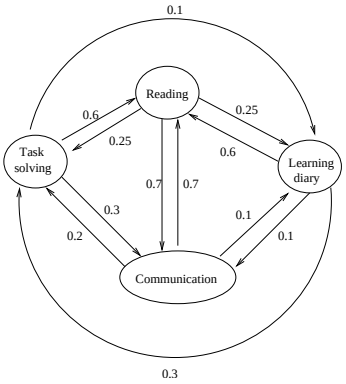

Figure 8.4: A 1st order hidden Markov model describing the change of contexts in a learning system.

8.3.2 Learning hidden Markov models from temporal data

Learning a hidden Markov model accurately is a difficult problem, because the model can be very complex. Because the educational data sets are typically small and sparse, we recommend to use as simple Markov models as possible. In addition,

we can restrict the search space by analyzing the temporal patterns first by descriptive means. It is quite likely that only some contexts, observations, or actions are statistically dependent and we do not have to model all possible relations. We suggest to analyze the temporal data first by *episodes* (see e.g. [MT02]). In the following we define the episode for any kind of *events*, which can be either observed variable values or actions recorded into the log. In practice, the number of possible variable value combinations or actions is often too large and we have to group them according to their type.

Definition 29 (Episode) *Let $A_1, ..., A_l$ be a set of event types, and (A, t) an event, which describes the event type at time step t. Given an event sequence, $s = ((A_1, t_1), ..., (A_n, t_n))$, a sequence of action types, $E_i = (A_{i_1}, ..., A_{i_k})$, and a window width, win, we say that* episode E_i *occurs in s, if s contains a subsequence w, $|w| \leq win$ and for all $A_{i_j} \in E_j$ there is an event $(A_{i_j}, t) \in w$ for some t.*

If all $A_{i_j} \in E_i$ occur in w in the same order than in E_i, the episode is called serial. *Otherwise it is* parallel.

The frequency of episode E_i in s, given win, is

$$fr(E_i, s, win) = \frac{|\{w \mid E_i \text{ occurs in } w, |w| \leq win\}|}{t_n - t_1 + win - 1}.$$

If $fr(E_i, s, win) \geq min_{fr}$ for some user-defined $min_{fr} > 0$, we say that episode E_i is frequent.

Now frequent serial episodes describe a sequence of contexts which typically succeed each other (e.g. context c_i succeeds context c_j with probability p). If the contexts succeed immediately, we can use 1st order *HMM*s, otherwise higher order *HMM*s are needed. Parallel episodes describe a set of contexts which occur closely, but the order is not fixed. In this case, the *HMM* should contain transitions to both directions (from c_i to c_j and from c_j to c_i) and, once again, a higher order *HMM*s may be needed. If a context does not have any typical successor, all other contexts are equally probable.

8.3.3 Dynamic Bayesian networks

The idea of hidden Markov models can be enlarged by *dynamic Bayesian networks* (*DBNs*) [Kja92]. The only difference is that the observations are organized as a Bayesian network. *DBN*s offer a nice solution to embed the dynamic nature of the

system into classification, by combining a hidden Markov model with a Bayesian classifier.

A dynamic Bayesian model consist of a Markov chain of hidden state variables (1st order *HMM*) and a series of Bayesian networks, each of them associated with one hidden state. Any kind of Bayesian networks can be used for classification, but general Bayesian networks are often too complex relative to the amount of data available. Thus, we propose to use naive Bayes classifiers or other simple structured Bayesian models. In *dynamic naive Bayes model* [HKLP99] both root variable X_0 and leaf variables $X_1, ..., X_n$ of the naive Bayesian network can depend on hidden state variable C. The model structure is represented in Figure 8.5.

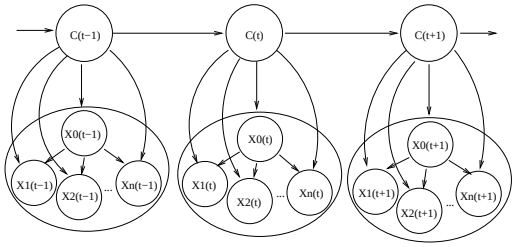

Figure 8.5: A dynamic naive Bayesian model.

In context-aware applications we can catch the unknown and unmeasurable highest level contexts like the student's actual skills, intention, or motivation by hidden variables $C(t)$. The root variable, $X_0(t)$, corresponds to the highest level context which can be observed, e.g. the user's current action, and leaf variables $X_1(t), ..., X_n(t)$ correspond to lower level contexts. This model is especially attractive for context-aware applications, because it separates predefined high-level contexts ($X_0(t)$) and real but unobservable contexts behind them ($C(t)$).

Example 15 *Let us consider a personal digital assistant designed for university students. In addition to normal calendar and communication services it contains facilities for recording lectures, writing notes, drawing pictures, and storing learning material. In the learning mode the highest level observable context is the user's action (reading, writing, drawing, talking, or listening), and the low-level contexts are abstractions of the sensor measurements (e.g. position of hands, focus of eyes, nearby people, sound volume). We assume that the user's intentions would define the highest level context, but they are hard or impossible to measure and classify. For example, when the exam is coming the student wants to keep a complete record about the last lecture, but in a group work session s/he is passive and writes down minimal notes. The activity and intentions may depend on several factors like course, topic, lecturer, other students, weekday, time of day, etc.*

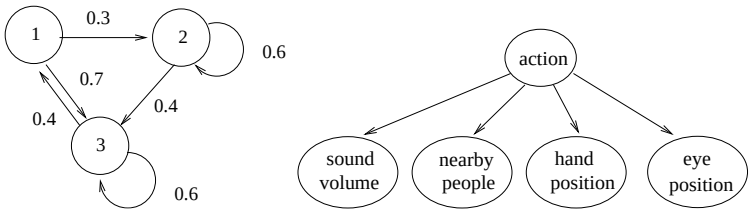

Figure 8.6: A hidden Markov model and a naive Bayesian model for the university student's digital assistant.

The system can be modelled by a dynamic Bayesian network which consists of a three state hidden Markov model and a naive Bayes model (Figure 8.6). This model can already be used for prediction, but we can try to analyze what the hidden states describe. In state S_1 the student mostly listens, her/his eyes are sometimes closed, and there are other people near by. This may be a passive lecture situation. In state S_2 s/he writes and draws actively, sometimes reads, but there are no other people. This seems like active self-studying, e.g. solving exercises. In state S_3 s/he talks and listens, sometimes writes, and there are other people near by. This situation suggests either an active group meeting or free time with friends. Thus, it seems that the student is very passive in lectures, but active in self-studying and group works (and/or social life with friends). Now we can also try to interpret the transitions of the hidden Markov model. It looks like our student goes usually to lectures and have breaks or group sessions between them. Sometimes she goes to study herself after the lecture, but never after the group session and/or social activities.

In the same way we could add any other probabilistic classification methods to the hidden Markov model. In this case, the *HMM* gives only the prior probabilities for the contexts and otherwise the classification is done normally. The same can be done, if we use *probabilistic clustering* in social filtering approaches. The only difference is that now we have to learn the *HMM* for actions instead of contexts. *HMM* gives the prior probability of an action, given the previous action(s), and the probabilities are updated according to the actions used in the similar situations. All three paradigms can be combined, if we use classification for context inference and social filtering for action selection, with *HMM* dynamics added to both of them.

Chapter 9

Conclusions

In this thesis we have given a wide overview of modelling educational data. The main goal has been to define general principles for designing adaptive learning environments. The core of such systems consists of predictive models, which determine the situation and select the most appropriate action. For full adaptivity, such models should be learnt from real data. However, the current systems use mostly predefined ad hoc models. Sometimes the model parameters are learnt from data, but in all general purpose systems the model structure is predefined. This is a serious restriction to adaptivity; if we put it very strongly, the learners are fit to the model, instead of fitting the model to the learners.

As a solution, we have proposed a dual principle of descriptive and predictive modelling, in which descriptive data analysis guides the construction of predictive models. The idea is that descriptive modelling produces accurate domain knowledge, which is used as a semantic bias in predictive modelling process. When the predictive models are applied in practice, new data is gathered for descriptive modelling and models evolve in an iterative manner. This principle is the main thread in this thesis.

The main results of this research are the following:

- Systematic analysis of educational data sets revealed that educational data sets are typically small, sparse, and skewed; they consist of mixed data types, and contain relatively many outliers.
- The main criteria for selecting the modelling paradigm and a model is to produce accurate and robust models, which have comprehensive interpretation. To achieve this goal, we have to analyze the inductive bias in the modelling

paradigm, learning style, and learning algorithm and select methods which match the properties in data.

- The analysis of clustering methods revealed that none of the existing methods is optimal for educational data, but probabilistic clustering is the most useful. In some cases we can get good results by discretizing the data to binary, and clustering the binary-valued data.
- The analysis of classification methods proposed simple classifiers, multiple linear regression for numeric data and naive Bayes classifier and its extensions for categorial or mixed data.
- The empirical experiments on the *ViSCoS* data demonstrated that accurate classifiers can be learnt from small and inconsistent data sets. This presupposes careful data preprocessing and model selection based on dependency analysis. We were able to predict the course final results (pass/fail) with 80% accuracy in the middle of course. In addition, we could detect outliers, whose final results were unpredictable, given only exercise points in *Prog.2* data.

The research has also revealed several important topics for further research:

- The effect of data preprocessing techniques on clustering quality and classification accuracy should be further researched. Especially, we should study whether principal component analysis and independent component analysis suit for educational data sets.
- The optimal discretization of numeric data is crucial for models which use categorial or binary data. In the future, we will test discretization by segmentation (1-dimensional clustering).
- Support vector machines are known to be good classifiers for small and sparse data sets. Their applicability to educational data should be researched. In addition, we should research how to represent the model in a comprehensive way to teachers and students.
- Variations of probabilistic clustering should be further researched. Especially, we should develop a method for analyzing the appropriate distributional form for clusters.
- New clustering methods should be developed for detecting maximal homogeneous clusters in heterogeneous data. One option is to develop our new *BinClust* method further. Outlier detection techniques should also be further studied.

- In this thesis we have assumed that all data is represented by attribute-value pairs. In the educational domain we have at least one important source of data, namely concept maps, which have a more complex structure. New techniques are needed for analyzing and clustering such graph-based data.
- In the *ViSCoS* project we have already collected new data in prerequisite questionnaires. This data should be analyzed and utilized in modelling. New temporal data is also available for analysis in the form of system log.

Appendix: Basic concepts and notations

Data representation

In this thesis we assume that the data is represented as relations. The basic concepts and notations are defined by relational algebra (Table 9.1).

Table 9.1: Basic concepts and notations concerning data representation.

Notation	**Meaning**
A_i	An attribute (variable).
$Dom(A_i)$	Domain of attribute A_i; i.e. the set of possible atomic values. (The atomicity is defined in the given data model.)
$R(A_1, ..., A_k)$	A relation schema is a finite set of attributes $R = \{A_1, ..., A_k\}$.
$\|R\| = k$	Order of relational schema, i.e. number of attributes in R.
$t = \{(A_1, t(A_1)), ..., (A_k, t(A_k))\}$	A tuple (row) is a set of attribute–value pairs. Formally, t is a mapping $t : R \rightarrow \bigcup_{i=1}^{k} Dom(A_i)$ for which $t(A_i) \in Dom(A_i)\ \forall i = 1, ..., k$.
$t = (a_1, ..., a_k)$	Tuple $t = \{(A_1, a_1), ..., (A_k, a_k)\}$, when the order of attributes is fixed.
r (given R)	A relation according to R (an instance of R) is a finite set of tuples according to R.
$\|r\|$	Size of relation r, number of tuples in r.
$t[X] = (t(A_1), t(A_2), ..., t(A_k))$ $\forall A_i \in X \subseteq R$	Projection of tuple t onto attribute set X, i.e. values of attributes X in tuple t.
$\pi_X(r) = \{t[X] \mid t \in r\}$	Projection of relation r onto attribute set X.
$E(t) \in \{0, 1\}$	Condition E is a logical expression of attributes, attribute values, comparison operations $=, <, \leq, >, \geq, \neq$ and logical operations $\wedge$, $\vee$ and $\neg$. $E(t) = 1$, if conditon E holds for t, and 0, otherwise. E.g. $E_1 = (Gender = female) \vee (20 \leq Age \leq 30)$.
$\sigma_E(r) = \{t \in r \mid E(t)\}$	Selection of rows from relation r given condition E. The results is a subset of tuples in r for which condition E holds.

In addition, it is often useful to suppose that all tuples have an identifier or a key attribute A_{id} such that for all tuples $t_1, t_2 \in r$, $t_1 = t_2$ when $t_1[A_{id}] = t_2[A_{id}]$. If the row order is fixed, the row number can simply be used as an identifier.

Frequencies and probabilities

The basic concepts and notations concerning frequencies and probabilities are summarized in Table 9.2. Notice that in data mining literature concept "frequency"

usually refers to relative frequency.

Table 9.2: Basic concept and notations concerning frequencies and probabilities.

Notation	**Meaning**
$m(F) = \|\sigma_F(r)\|$	The (absolute) frequency of feature F is the number of tuples, for which feature F holds. E.g. m(credits in math ≥ 10) tells the number of students, who have collected at least 10 credit units in math.
$fr(F) = \frac{m(F)}{\|r\|}$	Relative frequency of feature F tells the ratio of absolute frequency of F and size of r.
$P(X)$	Probability of X. In the frequentist interpretation $P(X) \approx fr(X)$.
$P(X\|Y)\frac{P(X,Y)}{P(Y)}$	Conditional probability of X given Y. In the frequentist interpretation $P(X\|Y) \approx \frac{fr(X,Y)}{fr(Y)}$.

We have adopted the frequentist interpretation of probabilities, where the probability of feature F is approximated by the relative frequency of F in the given sample. Although this is only an approximation, the relative frequency of F converges to real probability $P(F)$, when the sample size n grows indefinitely. This result is based on *Bernoulli's Theorem*, which states that the probability that the observed distribution deviates from the real probability approaches zero, when the sample size grows indefinitely:

$\forall \epsilon > 0 : \lim_{|r|\to\infty} P(|\frac{m(F)}{|r|} - p| \geq \epsilon) = 0.$

Satistical measures

The definitions and notations of basic statistical measures used in this tesis are summarized in Table 9.3. Mean, standard deviation, variance, and covariance are defined for discrete numeric variables and median for discrete numeric or ordinal variables.

Table 9.3: Statistical measures and their notations used in this thesis. Mean, standard deviation, variance, and covariance are defined for discrete numeric variables X and Y, but median is defined also for ordinal (categorial) variables.

Notation	**Meaning**
$mean(X)$	The mean of X, $mean(X) = \frac{\sum_{i=1}^{n} x_i}{n}$.
$stdev(X)$	The standard deviation of X, $stdev(X) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - mean(X))^2}{n}}$
$var(X)$	The variance of X, $var(X) = stdev^2(X)$
$cov(X,Y)$	Covariance of X and Y, $cov(X,Y) = \frac{\sum_{i=1}^{n}(x_i - mean(X))(y_i - mean(Y))}{n}$
$median(X)$	Median of X, $median(X) = \begin{cases} x_{n/2+1}, & \text{when } n \text{ is odd} \\ x_{n/2}, & \text{when } n \text{ is even.} \end{cases}$

Bibliography

[AAG03] U. Amato, A. Antoniadis, and G. Gregoire. Independent component discriminant analysis. *International Mathematical Journal*, 3(7):735–753, 2003.

[Agg01] C. Aggarwal. On the effects of dimensionality reduction on high dimensional similarity search. In *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS'01)*, pages 256–266, New York, NY, 2001. ACM Press.

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, pages 94–105, New York, NY, USA, 1998. ACM Press.

[AHK01] C.C. Aggarwal, A. Hinneburg, and D.A. Kleim. On the surprising behavior of distance metrics in high dimensional space. In *Proceedings of the 8th International Conference on Database Theory (ICDT 2001)*, volume 1973 of *Lecture Notes in Computer Science*, pages 420–434. Springer-Verlag, 2001.

[Ber02] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002. Available on `http://citeseer.ist.psu.edu/berkhin02survey.html`, retrieved 1.1. 2006.

[BGRS99] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. When is "nearest neighbor" meaningful? In *Proceedings of the 7th International Conference on Database Theory (ICDT'99)*, volume 1540 of *Lecture Notes in Computer Science*, pages 217–235. Springer-Verlag, 1999.

[BJSM03] J.E. Beck, P. Jia, J. Sison, and J. Mostow. Predicting student help-request behavior in an intelligent tutor for reading. In *Proceedings of the 9th International Conference on User Modeling*, volume 2702 of *Lecture Notes in Computer Science*, pages 303–312. Springer-Verlag, 2003.

[BMFH97] D. Barbara, W. Du Mouchel, C. Faloutsos, and P.J. Haas. The New Jersey data reduction report. *Bulletin of the Technical Committee on Data Engineering. Special Issue on Data Reduction Techniques*, 20(4), December 1997.

[BMS97] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In J. Peckham, editor, *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 265–276. ACM Press, 1997.

[BR88] A. Blum and R.L. Rivest. Training 3-node neural network is NP-complete. In *Proceedings of the 1988 Workshop on Computational Learning Theory (COLT)*, pages 9–18, MA, USA, 1988. MIT.

[BR93] J.D. Banfield and A.E. Raftery. Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 49:803–821, 1993.

[Bra68] J.V. Bradley. *Distribution Free Statistical Tests*. Prentice-Hall, Englewood Cliffs, N.J., 1968.

[Bru00] P. Brusilovsky. Adaptive hypermedia: From intelligent tutoring systems to web-based education. In *Proceedings of the 5th International Conference on Intelligent Tutoring Systems*, volume 1839 of *Lecture Notes in Computer Science*, pages 1–7. Springer-Verlag, 2000.

[BTR04] K. Barker, T. Trafalis, and T.R. Rhoads. Learning from student data. In *Proceedings of the 2004 IEEE Systems and Information Engineering Design Symposium*, pages 79–86, Charlottesville, VA, 2004. University of Virginia.

[BW98] J.E. Beck and B.P. Woolf. *Using a learning agent with a student model*, volume 1452 of *Lecture Notes in Computer Science*, pages 6–15. Springer-Verlag, 1998.

[BYRN99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, Reading, US, 1999.

[CC78] T.M. Cover and J.M. Van Campenhout. On the possible orderings in the measurement selection problem. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-7:657–661, 10 1978.

[CCL03] C.-Y. Chou, T.-W. Chan, and C.-J. Lin. Redefining the learning companion: the past, present, and future of educational agents. *Computers & Education*, 40(3):225–269, 2003.

[CF04] S. Chaffar and C. Frasson. Inducing optimal emotional state for learning in intelligent tutoring systems. In *Proceedings of the 5th International Conference on Intelligent Tutoring Systems*, volume 3220 of *Lecture Notes in Computer Science*, pages 45–54. Springer-Verlag, 2004.

[CGB03] G. Castillo, J. Gama, and A.M. Breda. Adaptive Bayes for a student modeling prediction task based on learning styles. In *Proceedings of the 9th International Conference on User Modeling*, volume 2702 of *Lecture Notes in Computer Science*, pages 328–332. Springer-Verlag, 2003.

[CHZY03] B. Cheung, L. Hui, J. Zhang, and S.M. Yiu. SmartTutor: an intelligent tutoring system in web-based adult education. *The Journal of Systems and Software*, 68(1):11–25, 2003.

[CK00] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College, Computer Science, Hanover, NH, November 2000.

[CLC05] C.-M. Chen, H.-M. Lee, and Y.-H. Chen. Personalized e-learning system using item response theory. *Computers & Education*, 44:237–255, 2005.

[CLKL04] V. Cheng, C.H. Li, J.T. Kwok, and C.-K. Li. Dissimilarity learning for nominal data. *Pattern Recognition*, 37(7):1471–1477, 2004.

[CMRE00] B.-I. Cho, J.A. Michael, A.A. Rovick, and M.W. Evens. An analysis of multiple tutoring protocols. In *Proceedings of the 5th International Conference on Intelligent Tutoring Systems*, volume 1839 of *Lecture Notes in Computer Science*, pages 212–221. Springer-Verlag, 2000.

[Com94] P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36:287–314, 1994.

[Coo90] G.F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, 33, 1990.

[Cyb89] G. Cybenko. Approximation by superpositions of a sigmoid function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[DA99] A.K. Dey and G.D. Abowd. Towards a better understanding of context and context-awareness. GVU Technical Report GIT-GVU-99-22, College of Computing, Georgia Institute of Technology, 1999. Available on ftp://ftp.cc.gatech.edu/pub/gvu/tr/1999/99-22.pdf, retrieved 1.1. 2006.

[dB00] B. du Boulay. Can we learn from ITSs? In *Proceedings of the 5th International Conference on Intelligent Tutoring Systems (ITS 2000)*, volume 1839 of *Lecture Notes in Computer Science*, pages 9–17, London, UK, 2000. Springer-Verlag.

[DD03] M. Denuit and J. Dhaene. Simple characterizations of comonotonicity and countermonotonicity by extremal correlations. *Belgian Actuarial Bulletin*, 3:22–27, 2003.

[Dev03] K. Devlin. Why universities require computer science students to take math? *Communications of the ACM*, 46(9):37–39, Sep 2003.

[DHS00] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification.* Wiley-Interscience Publication, New Yor, 2nd edition, 2000.

[DP97] P. Domingos and M. Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29:103–130, 1997.

[DP05] M.C. Desmarais and X. Pu. A Bayesian student model without hidden nodes and its comparison with item response theory. *International Journal of Artificial Intelligence in Education*, 15:291–323, 2005.

[DR98] A. Dasgupta and A.E. Raftery. Detecting features in spatial point processes with clutter via model-based clustering. *Journal of the American Statistical Association*, 93(441):294–302, 1998.

[DSB04] N. Delavari, M.R.A. Shirazi, and M.R. Beikzadeh. A new model for using data mining technology in higher educational systems. In *Proceedings of the 5th International Conference on Information Technology Based Higher Education and Training: (ITHET'04)*, pages 319– 324. IEEE, 2004.

[Dui00] R. Duin. Learned from neural networks. In *Proceedings of the 6th Annual Conference of the Advanced School for Computing and Imaging (ASCI-2000)*, pages 9–13. Advanced School for Computing and Imaging (ASCI), 2000.

[EC02] V. Estvill-Castro. Why so many clustering algorithms? A position paper. *SIGKKD Explorations*, 4(1):65–75, 2002.

[Far02] J.J. Faraway. Practical regression and Anova using R. `http://www.stat.lsa.umich.edu/~faraway/book/`, 2004 (web-version 2002). Retrieved 1.1. 2006.

[FGG97] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

[Fox96] E. Fox. Details of clustering algorithms (lecture notes). `http://maya.cs.depaul.edu/~classes/ds575/clustering/CL-alg-details.html`, 1995-1996. Retrieved 1.1. 2006.

[FR98] C. Fraley and A.E. Raftery. How many clusters? Which clustering method? Answers via model-based cluster analysis. *The Computer Journal*, 41(8):578–588, 1998.

[FR00] C. Fraley and A. Raftery. Model-based clustering, discriminant analysis, and density estimation. Technical Report 380, University of Washington, Department of Statistics, October 2000.

[Gar05] G.D. Garson. Pa 765 statnotes: An online textbook (course material for quantitative research in public administration). `http://www2.chass.ncsu.edu/garson/pa765/statnote.htm`, 2005. Retrieved 1.1. 2006.

[GC04] E. Guzman and R. Conejo. *A model for student knowledge diagnosis through adaptive testing*, pages 12–21. Number 3220 in Lecture Notes in Computer Science. Springer-Verlag, 2004.

[GH96] D. Geiger and D. Heckerman. Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence*, 82:45–74, 1996.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[HAK00] A. Hinneburg, C.C. Aggarwal, and D.A. Kleim. What is the nearest neighbor in high dimensional spaces? In *Proceedings of 26th International Conference on Very Large Data Bases (VLDB 2000)*, pages 506–515. Morgan Kaufmann, 2000.

[Häm05] W. Hämäläinen. General paradigms for adaptive learning systems. In *Proceedings of the IADIS Multi Conference on Computer Science and Information Systems (MCCSIS 2005)*, pages 476–483, 2005. Available on `http://www.cs.joensuu.fi/~whamalai/articles/paradigm.pdf`. Retrieved 1.1. 2006.

[Haw80] D.M. Hawkins. *Identification of outlier*. Monographs on Applied Probability & Statistics. Chapman and Hall, 1980.

[HBP00] S.H. Ha, S.M. Bae, and S.C. Park. Web mining for distance education. In *Proceedings of IEEE International Conference on Management of Innovation and Technology (ICMIT)*, volume 2, pages 715–219, 2000.

[Hig00] R. High. Dealing with 'outliers': how to maintain your data's integrity. *Computing News*, spring 2000. Available on `http://cc.uoregon.edu/cnews/spring2000/outliers.html`. Retrieved 1.1. 2006.

[HK98] A. Hinneburg and D. Keim. An efficient approach to clustering large multimedia databases with noise. In *Proceedings of the 4th ACM SIGKDD*, pages 58–65, 1998.

[HKLP99] W. Hämäläinen, P. Kuokkanen, K.-P. Laakso, and S. Perttu. Naive dynamic belief networks. In H. Tirri and K.-P. Laako, editors, *Research Seminar on Bayesian Networks*, pages 45–54. 1999. Available on `http://www.cs.helsinki.fi/u/kpalaaks/bayes-1999/`, retrieved 1.1. 2006.

[HL89] D.W. Hosmer and S. Lemeshow. *Applied Logistic Regression*. John Wiley & Sons, Chinchester, UK, 1989.

[HLS06] W. Hämäläinen, T.H. Laine, and E. Sutinen. Data mining in personalizing distance education courses. In C. Romero and S. Ventura, editors, *Data Mining in E-learning*, pages 157–171. WitPress, Southampton, UK, 2006.

[HMS02] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. MIT Press, Cambridge, Massachussetts, USA, 2002.

[HN89] R. Hecht-Nielsen. Theory of the backpropagation neural network. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 1, pages 593–605. IEEE, 1989.

[HR76] L. Hyafil and R.L. Rivest. Constructing optimal binary decision trees is NP-complete. *Information Processing Letters*, 5(1), 1976.

[HSST01] A. Haataja, J. Suhonen, E. Sutinen, and S. Torvinen. High school students learning computer science over the web. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning (IMEJ)*, 3(2), October 2001. Available on `http://imej.wfu.edu/articles/2001/2/index.asp`, retrieved 1.1. 2006.

[HSST04] W. Hämäläine, J. Suhonen, E. Sutinen, and H. Toivonen. Data mining in personalizing distance education courses. In *Proceedings of the 21st ICDE World Conference on Open Learning and Distance Education*, 2004.

[Hub64] P.J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35:73–101, 1964.

[Hub81] P.J. Huber. *Robust Statistics*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York, 1981.

[HV06] W. Hämäläinen and M. Vinni. Comparison of machine learning methods for intelligent tutoring systems. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*, volume 4053 of *Lecture Notes in Computer Science*, pages 525–534. Springer-Verlag, 2006.

[IY94] M. Ichino and H. Yaguchi. Generalized Minkowski metrics for mixed feature-type data analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4):698–708, 1994.

[JD88] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.

[JDM00] A.K. Jain, P.W. Duin, and J. Mao. Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[JJM+05] A. Jonsson, J. Johns, H. Mehranian, I. Arroyo, B. Woolf, A.G. Barto, D. Fisher, and S. Mahadevan. Evaluating the feasibility of learning student models from data. In *Papers from the 2005 AAAI*

Workshop on Educational Data Mining, pages 1–6, Menlo Park, CA, 2005. AAAI Press.

[JMF99] A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[JZ97] A. Jain and D. Zongker. Feature selection: evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997.

[Kan02] M. Kantardzic. *Data Mining: Concepts, Models, Methods, and Algorithms*. John Wiley & Sons, IEEE Press, Piscataway, NJ, 2002.

[Kar01] H. Karttunen. *Datan käsittely (Data management)*. CSC – Tieteellinen laskenta Oy, 2001.

[KB05] A. Kristofic and M. Bielikova. Improving adaptation in web-based educational hypermedia by means of knowledge discovery. In *Proceedings of the 16th ACM Conference on Hypertext and Hypermedia (Hypertext 2005)*, pages 184–192, 2005.

[Kel35] T.L. Kelley. An unbiased correlation ratio measure. *Proceedings of the National Academy of Sciences*, 21(9):554–559, 1935.

[Kja92] U. Kjaerulff. A computational scheme for reasoning in dynamic probabilistic networks. In *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence (UAI'92)*, pages 121–129. Morgan Kaufman, 1992.

[KL05] L. Khan and F. Luo. Hierarchical clustering for complex data. *International Journal on Artificial Intelligence Tools*, 14(5), 2005.

[Kle99] J.M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[KPP03] S.B. Kotsiantis, C.J. Pierrakeas, and P.E. Pintelas. Preventing student dropout in distance learning using machine learning techniques. In *Proceedings of 7th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES-2003)*, volume 2774 of *Lecture Notes in Computer Science*, pages 267–274. Springer-Verlag, 2003.

[KV04] K. Kabassi and M. Virvou. Personalised adult e-training on computer use based on multiple attribute decision making. *Interaction with Computers*, 16(1):115–132, 2004.

[Lee01] M.-G. Lee. Profiling students' adaption styles in web-based learning. *Computers & Education*, 36:121–132, 2001.

[LHM98] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings of Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)*, pages 80–86. AAAI Press, 1998.

[LMWY03] B. Liu, Y. Ma, C.K. Wong, and P.S. Yu. Scoring the data using association rules. *Applied Intelligence*, 18(2):119–135, 2003.

[Lor80] F.M. Lord. *Applications of Item Response Theory to Practical Testing Problems*. Lawrence Erlbaum Associates, 1980.

[LS72] D.V. Lindley and A.F.M. Smith. Bayesian estimates for the linear model (with discussion). *Journal of the Royal Statistical Society*, 34:1–44, 1972.

[LS02] R.S. Legaspi and R. Sison. A machine learning framework for an expert tutor construction. In *Proceedings of the International Conference Of Computers In Education (ICCE 2002)*, pages 670–674. IEEE, 2002.

[MA03] J.S. Milton and J.C. Arnold. *Introduction to Probability and Statistics: Principles and Applications for Engineering and the Computing Sciences*. McGraw-Hill, New York, 4th edition, 2003.

[MBKKP03] B. Minaei-Bidgoli, D.A. Kashy, G. Kortemeyer, and W. Punch. Predicting student performance: an application of data mining methods with an educational web-based system. In *Proceedings of 33rd Frontiers in Education Conference*, pages T2A–13–T2A18, 2003.

[MBKP04] B. Minaei-Bidgoli, G. Kortemeyer, and W.F. Punch. Association rule analysis for an online education system. In *Proceedings of the 2004 IEEE International Conference on Information Reuse and Integration (IRI 2004)*, pages 504–509, 2004.

[MCF+94] T.M. Mitchell, R. Caruana, D. Freitag, J. McDermott, and D. Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):81–91, 1994.

[Mit97] T.M. Mitchell. *Machine Learning*. McGraw-Hill Companies, New York, NY, USA, 1997.

[MLF98] T. Mengelle, C. De Lean, and C. Frasson. Teaching and learning with intelligent agents : Actors. In *Proceedings of the 4th International Conference on Intelligent Tutoring Systems*, volume 1452 of *Lecture notes in Computer Science*, pages 284–293. Springer-Verlag, 1998.

[MLW+00] Y. Ma, B. Liu, C.K. Wong, P.S. Yu, and S.M. Lee. Targeting the right students using data mining. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00)*, pages 457–464, New York, NY, USA, 2000. ACM Press.

[MM01] M. Mayo and A. Mitrovic. Opitimizing ITS behaviour with bayesian networks and decision theory. *International Journal of Artificial Intelligence in Education*, 12:124–153, 2001.

[MR84] F. Murtagh and A.E. Raftery. Fitting straight lines to point patterns. *Pattern Recognition*, 17:479–483, 1984.

[MSS02] N. Myller, J. Suhonen, and E. Sutinen. Using data mining for improving web-based course design. In *Proceedings of the International Conference on Computers in Education (ICCE 2002)*, pages 959–963. IEEE, 2002.

[MT02] H. Mannila and H. Toivonen. Knowledge discovery in databases: The search for frequent patterns (lecture material). `http://www.cs.helsinki.fi/u/htoivone/teaching/timuS02/b.pdf`, 1998 (minor modifications 2002). Retrieved 1.1. 2006.

[Neo04] R.E. Neopolitan. *Learning Bayesian Networks*. Prenctice Hall, Upper Saddle River, NJ, USA, 2004.

[OBBE84] T. O'Shea, R. Bornat, B. Boulay, and M. Eisenstad. Tools for creating intelligent computer tutors. In *Proceedings of the International NATO Symposium on Artificial and Human Intelligence*, pages 181–199, New York, NY, USA, 1984. Elsevier North-Holland, Inc.

[OC02] J. Osborne and A.B. Costello. Sample size and subject to item ratio in principal component analysis. *Practical Assessment, Research & Evaluation*, 9(11), 2002.

[Pag03] R.L. Page. Software is discrete mathematics. *ACM SIGPLAN Notices*, 38(9):79–86, September 2003.

[Pea88] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufman Publishers, San Mateo, California, 1988.

[PGMK02] K. Papanikolaou, M. Grgoriadou, G. Magoulas, and H. Kornilakis. Towards new forms of knowledge communication: the adaptive dimensions of a web-based learning environment. *Computers & Education*, 39:333–360, 2002.

[Qui86] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[Qui93] J.R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann, 1993.

[Qui94] J.R. Quinlan. Comparing connectionist and symbolic learning methods. In *Proceedings of a Workshop on Computational Learning Theory and Natural Learning Systems (vol. 1): Constraints and Prospects*, pages 445–456, Cambridge, MA, USA, 1994. MIT Press.

[RG97] S. Richardson and P.J. Green. On Bayesian analysis of mixtures with an unknown number of components. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 59:731–758, 1997.

[RS01] R. Rastogi and K. Shim. Mining optimized support rules for numeric attributes. *Information Systems*, 26(6):425–444, 2001.

[RVBdC03] C. Romero, S. Ventura, P. De Bra, and C. de Castro. Discovering prediction rules in AHA! courses. In *Proceedings of the 9th International Conference on User Modeling (UM 2003)*, volume 2702 of *Lecture Notes in Computer Science*, pages 25–34. Springer-Verlag, 2003.

[SA96] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In H.V. Jagadish and I.S. Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 1–12, Montreal, Quebec, Canada, 4-6 1996.

[SA98] J. Siemer and M.C. Angelides. A comprehensive method for the evaluation of complete intelligent tutoring systems. *Decision Support Systems*, 22(1):85–102, 1998.

[SAGCBR+04] A. Salazar-Afanador, J. Gosalbez-Castillo, I. Bosch-Roig, R. Miralles-Ricos, and L. Vergara-Dominguez. A case study of knowledge discovery on academic achievement, student desertion and student retention. In *Proceedings of the 2nd International Conference on Information Technology: Research and Education (ITRE 2004)*, pages 150–154. IEEE, 2004.

[SAT+99] A. Schmidt, K.A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven, and W. Van de Velde. Advanced interaction in context. In *Proceedings of the First International Symposium on Handheld and Ubiquitous Computing (HUC'99)*, volume 1707 of *Lecture Notes in Computer Science*, pages 89–102. Springer-Verlag, 1999.

[SB03] A.S.G. Smith and A. Blandford. ML Tutor: An application of machine learning algorithms for an adaptive web-based information system. *International Journal of Artificial Intelligence in Education*, 13:235–261, 2003.

[Sch93] C. Schaffer. Overfitting avoidance as bias. *Machine Learning*, 10:153–178, 1993.

[Sch98] E.W. Schwarz. Self-organized goal-oriented tutoring in adaptive hypermedia environments. In *Proceedings of the 4th International Conference on Intelligent Tutoring Systems*, volume 1452 of *Lecture Notes in Computer Science*, pages 294–303. Springer-Verlag, 1998.

[Sch02] R.E. Schapire. The boosting approach to machine learning: an overview. In *Proceedings of the MSRI Workshop on Nonlinear Estimation and Classification*, volume 171 of *Lecture Notes in Statistics*, pages 149–172. Springer-Verlag, 2002.

[SG92] P. Smyth and R.M. Goodman. An information theoretic approach to rule induction from databases. *IEEE Transactions on Knowledge and Data Engineering*, 4(4):301–316, 1992.

[SGMM03] R. Stathacopoulou, M. Grigoriadou, G.D. Magoulas, and D. Mitropoulos. A neuro-fuzzy approach in student modeling. In *Proceedings of the 9th International Conference on User Modeling*, volume 2702 of *Lecture Notes in Computer Science*, pages 337–341. Springer-Verlag, 2003.

[SK99] N. Shin and J. Kim. An exploration of learner progress and dropout in Korea National Open University. *Distance Education – An International Journal*, 20(1):81–97, 1999.

[SL91] S.R. Safavian and D. Landgrebe. A survey of decision tree classifier methods. *IEEE Transactions on Systems, Man and Cybernetics*, 21(3):660–674, May 1991.

[SM95] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating 'word of mouth'. In *Proceedings of ACM Conference on Human Factors in Computing Systems (CHI'95)*, volume 1, pages 210–217, 1995.

[Smy00] P. Smyth. Model selection for probabilistic clustering using cross-validated likelihood. *Statistics and Computing*, 10(1):63–72, 2000.

[Smy01] P. Smyth. Data mining at the interface of computer science and statistics. In R.L. Grossman, C. Kamath, P. Kegelmeyer, V. Kumar, and R. Namburu, editors, *Data Mining for Scientific and Engineering Applications*, volume 2 of *Massive Computing*, chapter 3, pages 35–61. Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[SNS00] R. Sison, M. Numao, and M. Shimura. Multistrategy discovery and detection of novice programmer errors. *Machine Learning*, 38(1-2):157–180, 2000.

[SR00] D. Stanford and A.E. Raftery. Principal curve clustering with noise. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:601–609, 2000.

[SS04] D.W. Scott and S.R. Sain. Multi-dimensional density estimation. In C.R. Rao and E.J. Wegman, editors, *Handbook of Statistics—Vol 23: Data Mining and Computational Statistics*. Elsevier, Amsterdam, 2004.

[SSP98] T. Starner, B. Schiele, and A. Pentland. Visual contextual awareness in wearable computing. In *Proceedings of the Second IEEE International Symposium on Wearable Computers (ISWC'98)*, pages 50–57, 1998.

[Sta04] Electronic statistics textbook (www version). `http://www.statsoft.com/textbook/stathome.html`, 2004. Retrieved 1.1. 2006.

[STC04] J. Shawe-Taylor and N. Cristianini. *Kernel Methods For Pattern Analysis*. Cambridge University Press, 2004.

[Sto96] D.W. Stockburger. Introductory statistics: Concepts, models, and applications (on-line textbook). `http://www.psychstat.missouristate.edu/introbook/sbk00m.htm`, 1996. Retrieved 1.1. 2006.

[STW99] R. Sullivan, A. Timmermann, and H. White. Data snooping, technical trading rule performance, and the bootstrap. *Journal of Finance*, 54:1647–1692, 1999.

[SYW01] R. Stiefelhagen, J. Yang, and A. Waibel. Estimating focus of attention based on gaze and sound. In *Proceedings of the 2001 Workshop on Perceptive User Interfaces (PUI '01)*, pages 1–9. ACM Press, 2001. Available on `http://isl.ira.uka.de/publications/PUI2001_rainer.pdf`. Retrieved 1.1. 2006.

[Toi96] H. Toivonen. *Discovery of Frequent Patterns in Large Data Collections*. PhD thesis, Department of Computer Science, University of Helsinki, 1996.

[TPP+02] A. Tsymbal, S. Puuronen, M. Pechenizkiy, M. Baumgarten, and L. Patterson. Eigenvector-based feature extraction for classification. In *Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference*, pages 354–358. AAAI Press, 2002.

[TSV03] J. Toivanen, T. Seppänen, and E. Väyrynen. Automatic recognition of emotion in spoken Finnish: preliminary results and applications. In *Proceedings of the International AAI Workshop on Prosodic Interfaces*, pages 85–89, 2003.

[VC03] S. Valenti and A. Cucchiarelli. Preliminary results from a machine learning based approach to the assessment of student learning. In *Proceedings of the 3rd IEEE International Conference on Advanced Learning Technologies (ICALT 2003)*, pages 426–427, 2003.

[VM02] G. Valentini and F. Masulli. *Ensembles of learning machines*, volume 2486 of *Lecture Notes in Computer Science*, pages 3–22. Springer-Verlag, 2002. Invited Review.

[Vom04] J. Vomlel. Bayesian networks in educational testing. *International Journal of Uncertainty, Fuzziness and Knowledge Based Systems*, 12(Supplementary Issue 1):83–100, 2004.

[War63] J. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58:236–244, 1963.

[Was97] B. Wasson. Advanced educational technologies: the learning environment. *Computers in Human Behavior*, 13(4):571–594, 1997.

[Web96] G. Weber. Episodic learner modeling. *Cognitive Science*, 20(2):195–236, 1996.

[Wek] Weka3. Data mining software in java. `http://www.cs.waikato.ac.nz/ml/weka/`. Retrieved 1.1. 2006.

[WM97] D.R. Wilson and T.R. Martinez. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, 6:1–34, 1997.

[WNO] C. Wallace, J. Neil, and R. O'Donnell. Camml software. `http://www.datamining.monash.edu.au/software/camml/`. Retrieved 1.1. 2006.

[WP05] T. Winters and T. Payne. What do students know? An outcomes-based assessment system. In *Proceedings of the 2005 International Workshop on Computing Education Research (ICER'05)*, pages 165–172, New York, NY, USA, 2005. ACM Press.

[Xia03] H. Xiaohua. DB-HReduction: a data preprocessing algorithm for data mining applications. *Applied Mathematics Letters*, 16:889–895, 2003.

[Xyc06] Xycoon. Linear regression techniques. In *Statistics - Econometrics - Forecasting (Online Econometrics Textbook)*, chapter II. Office for Research Development and Education, 2000-2006. Available on `http://www.xycoon.com/`. Retrieved 1.1. 2006.

[ZL03] W. Zang and F. Lin. Investigation of web-based teaching and learning by boosting algorithms. In *Proceedings of IEEE International Conference on Information Technology: Research and Education (ITRE 2003)*, pages 445–449, 2003.