Data Mining in Personalizing Distance Education Courses

Wilhelmiina Hämäläinen*whamalai@cs.helsinki.fi
Jarkko Suhonen *jarkko.suhonen@cs.joensuu.fi
Erkki Sutinen*erkki.sutinen@cs.joensuu.fi
Hannu Toivonen*hannu.toivonen@cs.helsinki.fi

Abstract

One of the biggest problems in distance education courses is how to deal with heterogeneous student population with various needs, skills and backgrounds. In a face-to-face learning situation the teacher can handle individual students more efficiently as s/he can observe the needs and skills of learners rather quickly. However, in the distance education setting the teacher is not available. Currently many of the distance courses are based on static learning materials and tools, which do not take into account the diversity of students. Adaptive hypermedia has been seen as a solution to individually richer learning environments. Adaptive hypermedia methods are often based on modelling the student’s current understanding of the learning context. However, this kind of approaches have several drawbacks; modelling a single student is a hard task and models work only on well-defined rather small domains. As a solution we shall introduce a new hybrid approach which combines both data mining and machine learning techniques. The focus is that instead of modelling the learner, we learn the model for the learning process.

1Department of Computer Science, University of Joensuu, Finland
2Department of Computer Science, University of Helsinki, Finland
1 Introduction

Future learning environments are thought to be based on taking care of individual
differences and needs of the students. Adaptive hypermedia methods are often
based on modelling the students’ current understanding of the learning context
(Brusilovsky & Maybury 2002). This modelling is then utilized to adapt the learning
environment according to an assumption of student’s current knowledge. Students
in distance education courses are often heterogeneous with different backgrounds,
skills, motivation and learning styles. This has also been discovered in the Virtual
 Approbatur program at the University of Joensuu (Haataja, Suhonen, Sutinen &
Torvinen 2001). In Virtual Approbatur high school students can take the first year
university level computer science studies via the internet. The program is almost
entirely distant, with few face-to-face learning situations. Students receive individual
feedback through weekly assignments, which all are assessed by instructors at the
university. Hence, every student receives individual feedback on every assignment
during the course. It has been discovered that the individual feedback has been one
of the key elements on successful implementation of the program.

However, the problem with individual assessment is the large amount of instructors
needed to assess the assignments. Hence, we need robust methods to automate or
semi-automate the support for an individual student. Furthermore, we need meth-
ods to expand the information collected from the assessment of students learning
processes. On the other hand we have lot of information from a single student;
her/his marks on courses, individual marks and assessment information from the
assignments, and background information. In order to fully utilize the information
collected from the Virtual Approbatur program, we have started to develop various
data mining and machine learning methods.

There are some examples of using data mining in educational technology, but it
seems that these kind of methods are not common (Silva & Vieira 2002, Chen, Lin,
Tzeng, Yeh & T.-P.Wang 2001). For example in the National University of Singapore
data mining application have been used for classifying and selecting those students
who need extra classes in a given subject. With the help of data mining they are
able to select the targeted students much more precisely than by traditional methods
(Ma, Liu, Yu & Wong 2000). In general one might argue that so far the possibilities
of probabilistic learning models and data mining techniques have not been fully
utilized in educational technology.

Some probabilistic learning models based on Bayesian networks or Markov chains
have been introduced (e.g. (Murray 1998, Milan 2002)), but unfortunately they
are awkward and assume a lot of unrealistic restrictions. Those models rely on the
experts’ estimates about the structure of the learning process (the dependencies)
and the probabilities of different events. In the worst case each student’s learning
process is supposed to progress according to the same path, and even the transition
probabilities between each learning phase are same. Millan et al. (Milan 2002) in-
troduce a more realistic progress structure, which considers the hierarchy of course
topics, but still the probabilities are guessed by an expert. However, such probabilities cannot be estimated without any statistical data or at least such estimates are not justified.

However, data mining and machine learning techniques give efficient tools for learning the models of real life events. The distinction between data mining and machine learning is sometimes very delicate, but usually we call data mining such methods which aim at the discovery of useful information in large databases. The discovered knowledge can be rules describing properties of the data, frequently occurring patterns, clusterings of the objects in the database, etc., but we do not know beforehand what kind of information we will find. (Further reading, e.g. (Toivonen 1996).) On the other hand in machine learning we know exactly what kind of information we are looking for: we want to learn a model of given type from a training set. The model can be a Bayesian network, a decision tree, a neural network, etc. and we already may have some previous information about the model (given by an expert or learnt earlier). For example an expert has given a structure of causal dependencies describing some phenomenon and we want to learn the probabilities (weights of dependencies) associated to those dependencies.

In this paper we shall introduce a new hybrid model, which combines both data mining and machine learning techniques in constructing a Bayesian network to describe the students’ learning process. The main goal is to classify students so that we can give them differentiated guiding according to their skills and other characteristics. Classification of the students can also be used for predicting the future success in the course or diagnosing the causes of dropping out. This kind of classification can be performed by fixing the task categorization and clustering the students. The other goal is to classify the tasks according to the skills or knowledge, which they measure. It can be done in the same way by fixing the student classification and clustering the tasks.

In the current phase the structure of the Bayesian network is given and we learn only parameter values. However our model structure is so general that it does not impose any unrealistic restrictions. And, what is most important, we have not fixed the number of possible variable values, but they can also be learnt from the real data. In the future our aim is to learn also the structure of the model (the most plausible dependencies describing the process).

So far we have identified following use cases for these methods:

1. **Classification of students according to their individual characteristics.** This enables us to build intelligent agents that would instruct students during their studies. Classification should be open, so that the student is aware of the classification and the rules.

2. **Gathering the general overview of the student population and their success.** At the distance education setting the instructor can modify the learning settings according to the overview and progress of the students. For example the difficulty of future assignments could depend on the result of prediction.

3. **Information collection from several courses.** Usually the information
collected from students is used only inside a course. Our goal is to expand the information gathering to affect several courses. Interpreted information could be utilized in several learning tool developed to support the students.

4. **Giving (the teacher) an overview of the course structure and how different topics are balanced.** Our method helps also the teacher to understand the structure of the course implied by the exercises. The teacher can thus check that all important topics are covered by exercises and clarify to oneself what kind of dependencies hold between the topics of the course. Other teachers can also easily get a view, what is handled in that course.

In the following chapters we shall first give a background review on subject and introduce our concrete problem setting. Then we shall introduce a Bayesian network model, which we want to learn from our data, and describe, how it can be used in assisting both our students and teachers. After that we shall represent our new idea, how to learn such Bayesian model with the help of datamining techniques, especially clustering. Finally we shall discuss about our future plans and open possibilities.

### 2 Background

#### 2.1 Virtual Approbatur Studies

Virtual Approbatur is an on going distance education project at the Department of Computer Science, University of Joensu. The objective of the project is to offer high school students an opportunity to take 15 credit units of the first year Computer Science studies in one and a half years via the Internet (Haataja et al. 2001). Virtual Approbatur program offers diverse sources for collecting interesting data.

The main challenge in the Virtual Approbatur program is to teach programming. Introductory programming courses cover about half of the studies. The programming part of the studies consists of three courses, all given in the Java programming language. Programming courses consists of the following units:

- **P1:** Basics of Programming;
- **P2:** Introduction to Object-Oriented Programming; and
- **P3:** Laboratory Project of Introductory Programming.

The first programming course P1 teaches basics concepts of programming using the procedural programming paradigm. The P2 course focuses on object-oriented programming, giving students basics of objects, classes, and event handling. The last course P3 gives the students an opportunity to practice acquired skills in a programming project. The students can choose the topic of project quite freely; most end up with designing a tiny game program. Other courses in the program introduce design and architecture of computers, operating systems, research fields of Computer Science, and ethics of Information and Communication Technology. As the programming part of the studies has proved to be the most crucial for the success
or failure, we have also concentrated our data mining method development primarily to concern programming courses (Meisalo, Suhtonen, Sutinen, & Torvinen 2002). The first group of students started their studies in Fall 2000, and the group finished their studies in December 2001. Altogether we have had over 400 students participating in 19 different courses during the time period from Fall 2000 to December 2002. Table 1 illustrates the number of students starting the Virtual Approbatur studies in last few years.

Table 1: Number of Students in Virtual Approbatur

<table>
<thead>
<tr>
<th>Year</th>
<th>Students</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>80</td>
</tr>
<tr>
<td>2001</td>
<td>150</td>
</tr>
<tr>
<td>2002</td>
<td>140</td>
</tr>
</tbody>
</table>

2.2 Role of Exercises

In the Virtual Approbatur courses different exercises play very important role. Especially in teaching the programming the role is very crucial, because programming is basically a skill-based activity. Some people even suggest that the programming is a form of art. Hence, it is a well established method to use exercises as a basis for teaching programming. By doing meaningful exercises students are thought to gradually improve their programming skills.

The objective of the exercises is to provide students with multifaceted and interesting problems. By doing the assignments the student can practice the theoretical concepts in a very practical way. In fact these exercises direct the students behavior remarkably during the courses, as they can both spark the students enthusiasm and hinder her/his progress in worst cases.

A typical programming course can include 40-50 exercises. As we look at the number of students during the last few years in Table 1 we easily notice that the information rate from the exercise data is quite large. Furthermore, it gets larger every semester. During the course there are typically 10 exercises sessions. Each exercise session includes 3-5 different exercises. In order to pass the course each student must complete 1/3 of the assignments. On-line tutors at the university assess the exercises and give marks, so the students get feedback from every submitted assignment. One exercise is worth 1-3 points, so some of the assignments have higher impact to student’s overall score.

3 Data

In our Virtual Approbatur program we have the following data available:
• students \( st_1, ..., st_n \)

• tasks \( t_1, ..., t_l \)

• exercise sessions \( e_1, ..., e_v \) (\( v \) sessions)

• information, if the task \( j \) belongs to the session \( j \) (relation \( \text{InSession}[i][j] \))

• total points the student \( i \) has got in session \( j \) \( SPoints[i][j] \)

• points the student \( i \) has got from an individual task \( j \) \( TPoints[i][j] \)

• Exam tasks \( kt_1, ..., kt_r \) (\( r \) tasks)

• points the student \( i \) has got from the exam task \( j \) \( EPoints[i][j] \)

• final result of the course (degree, or failed=0, or interrupted=-1) for the student \( i \) \( FResult[i] \in \{-1, 0, 1-, 1+, 2-, 2, 2+, 3-, 3\} \)

In addition we can use expert evaluations:

• the preliminary categorization of the skills the tasks measure \( k_1, ..., k_m \), and

• the weight of the category \( j \) in the task \( i \) \( Weights[i][j] \)

Summing up all points of one exercise session loses a lot of important information, and the data gathering should support saving all individual points. On the other hand, the information, how the tasks are divided into exercise sessions, can give extra information about the categories measured by tasks. There can also be some weekly (or sessionly) appearing variation in the individual student’s success (e.g. all tasks of one session are missing for some student).

In the future we can also collect other information, like the student’s background knowledge (we need some suitable classification); information, how many times the student has tried this course before; and information, if the student accepted the extra exercises offered and how s/he succeeded in them. In addition we can estimate the difficulty of the tasks, e.g. by comparing their average points.

4 Basic Model

Our basic model is a Bayeasian network, which represents the dependencies between student types, task categories and task points (Figure 1). The variables of our model are following:

• Individual students \( S \) and tasks \( T \) are so called background variables, which affect on the prior probabilities of categories and student types. We represent them in the figure only for the sake of clarity, even if they don’t have any
probability values themselves (a group of students is given and the student types and their probabilities depend on that given group, or, in special case, an individual student.)

- Category variable $C$ can get values $c_1, ..., c_k$, in which $c_i$ is some category (e.g. loop structure, conditional statement, arrays, etc.), and $k$ is number of categories. The prior probability $P(C = c_i) = P(C = c_i | T = t_j)$ gives the preliminary probability that the task $t_j$ measures the skills of the category $c_i$. If no specific task is given, the prior probability $P(C = c_i)$ tells the weight of that category in all tasks so far.

![Bayesian network diagram](image)

Figure 1: A Bayesian network, which describes the dependencies between students, exercise tasks and the final results in some course. The student variable $S$ and the task variable $T$ are background variables, which are fixed and represented here only for the sake of clarity. Other variables are: $C$=category, $ST$= student type, $EP$=exam points, $TP$=task points, and $FR$=final result.

- Student type variable $ST$ can get values $st_1, ..., st_m$, in which $st_i$ is some student type (e.g. $A, B, C, ...$ or after interpretation we may give them describing names, like "algorithmic", "topcoder", "bitfixer", etc.), and $m$ is number of types. If no specific individual is given, the prior probability $P(ST = st_i)$ gives the total distribution of the given student group into some types. If some specific students $s_j$ is given, we can fix the prior probabilities $P(ST = st_i | S = s_j)$ according to our evaluation of her/his type. (In the last chapter we will discuss more, how to do that without relying on any omniscient expert’s beliefs.)
Variables $EP$ and $TP$ represent the probability distribution of point values (e.g. in Virtual Approbatur an exercise task can give 0,1,2, or 3 points) in exam or exercises.

The conditional probability $P(EP = ep_i|C = c_j, ST = st_k)$ gives the distribution of exam point values $ep_i$, given the category $c_j$ and the student type $st_k$. So to get the total probability distribution $P(EP = ep_i)$ ($i = 1, ..., maxpoints$) we just sum up the productions $P(C) \times P(ST) \times P(EP|C, ST)$ for all possible category and student type combinations:

$$P(EP = ep_i) = \sum_{j=1}^{k} \sum_{l=1}^{m} P(C = c_j) \times P(ST = st_l) \times P(EP = ep_i|C = c_j, ST = st_l).$$

This can be used in several ways, depending on, how we have fixed the task variable $T$ and the student variable $S$. We can predict the most probable point value in exam tasks in general or in some specific task, for all students in general or for some specific student.

If some task $T = t_p$ is fixed we know its category weights (i.e. probabilities $P(C = c_j|T = t_p)$), and the total probability distribution $P(EP)$ gives the point distribution in that exam task for all student types. If also the student $S = s_r$ is fixed, we know her/his type distribution $P(ST = st_l|S = s_r)$ (e.g. 90% algorithmic, 10% topcoder), and the total probability $P(EP)$ gives her/his total point distribution in that exam task (e.g. $P(EP = 5) = 0.9$, $P(EP = 6) = 0.1$ tells that s/he gets 5 points with 90% probability and 6 points with 10% probability in that exam task). The predictions for the exercise task point values are similar.

We can also use the real point information for diagnostic reasoning: if some student $s_r$ has got $ep_i$ points in exercise task $t_p$, what is her/his student type distribution? Or, if we know, how the categories were represented in out exercise tasks so far (distribution $P(C = c_k)$), and how many points the students got all together in different categories (total probability $P(TP = tp_j)$), we can diagnose the distribution of students into different types $P(ST = st_i)$ by Bayes rule. For example the latter is

$$P(ST = st_i|TP = tp_j) = \frac{\sum_{k=1}^{m} P(C = c_k) \times P(ST = st_i) \times P(TP = tp_j|C = c_k, ST = st_i)}{P(TP = tp_j)}.$$

In addition our model contains the variable $FR$, which tells the final result of the course. Its value tells either the degree or other result (failed, interrupted) of the course (e.g. in our Virtual Approbatur project we give values -1=interrupted, 0=failed, and 1-, 1, 1+, 2-, 2, 2+, 3-, 3 are degrees in our Finnish university system). If all the exercise and exam task points are known, the result is functionally determined, but we can also predict its value during the course, if we know the conditional probabilities $P(FR|EP, TP)$ in each phase of the course. So for example, if we are given some student $s_i$ and
her/his exercise task points so far and we have learnt the conditional probability $P(FR|EP, TP)$ (given the exercise tasks) as well as evaluated student’s type $ST$, we can predict her/his success in the course.

So far we have not represented any new insight (in the computer scientist’s point of view), how to apply Bayesian model for education technology purposes. The new insight is, how to learn those conditional probabilities, which measure the dependencies between the model variables.

If we just knew the division of tasks into categories and students into student types, learning the conditional dependences would be straightforward. (Because $P(A|B) = \frac{P(A,B)}{P(B)}$, we would just have to calculate the frequencies.) In traditional applications of learning the Bayesian networks such discrete data values are given in the training set, but our case is different. Even if we could give some expert evaluations for the weights of categories in each task, we cannot classify the students into any objective types. Thus the problem is, how to classify the students and the tasks in a reasonable way. In this point the datamining methods, especially clustering techniques, are of great help.

5 Clustering problem

We have studied several alternative methods to cluster the tasks into categories and the students into student types. Here we represent two main ideas.

In the first method we fix the task categorization either to expert evaluations or use uniform distribution (each category has the same prior probability to be represented in the task). So the prior probabilities $P(C = c_k|T = t_j)$ are given (or $P(C = c_k)$, if we consider the whole set of tasks).

Then we use the existing database about students’ points in previous courses and compute for each student the performance vector $PV$. $PV[i][k]$ gives the performance of the student $s_i$ in category $c_k$ and it is defined as

$$PV[i][k] = \sum_{j=1}^{l} Points[i][j] \times Weights[j][k],$$

in which $l$ is maximum number of tasks. I.e. we sum up over all tasks the student’s points in the task $t_j$ multiplied by the weight that the category $c_k$ is presented in that task.

Finally we cluster the performance vectors to get the clusters and the probabilities that a given student belongs to some cluster (student type).

After getting new evidence (performance data in a new course), we can update the model by Bayes rule and get better categorization for the tasks and students.

The second method can be used to get a new categorization for the tasks, when we have once learnt the conditional probabilities with the previous classification. First we fix the student classification into student types by using uniform distribution or learning the types by the Hubs and Authorities algorithm (Kleinberg 1998). So
we know the prior probabilities \( P(ST = st_i) \). Then we compute for each task a performance vector \( PV2 \), which tells the performance of each student type in each task. \( PV2[j][i] \) gives the points all the students of type \( st_i \) have got in task \( t_j \), and is defined as

\[
PV2[j][i] = \Sigma_{k=1}^n Points[k][j] \times StudentType[k][i],
\]

in which \( n \) is number of students.

Now we can cluster the performance vectors \( PV2 \) and get a new categorization for tasks. Notice that the number of categories can also change, because we use dynamic clustering, in which the number of clusters is not given.

The Hubs and authorities (HITS) algorithm (Kleinberg 1998) we mentioned above is an interesting method to classify the students in coarse level. The basic assumption in HITS algorithm is that all students, who have done similar tasks, are similar, and all tasks, which have been done by similar students, are similar. The algorithm itself is very simple: first we select some initial set of similar students (e.g. those, who have got similar final results). Then we add all tasks they have done and all other students, who have done those tasks into a graph. The weights of students and tasks are updated recursively, until the system converges. Those students, who have the greatest weights in the end, are the most similar with the initial set of students.

We have also tested the HITS algorithm for the whole problem, i.e. clustering both students and tasks into homogenous groups. It is too early to give any interpretations for our results, but at least this method proved to be a good comparing point for our main methods.

6 Discussion

In principle our hybrid method of using clustering techniques in learning Bayesian networks could be generalized, so that we would learn not only the parameter values but also the structure of the network. Our aim is to get as independent as possible from any "omniscient" expert evaluations, which are by they nature doomed to be incomplete. If we could learn the category hierarchy (i.e. how the course is divided into subtopics) automatically, our program would be directly adapted in new courses without any assistance of an expert. This would mean a really adaptive system, in which the learner could study the topics any order s/he wants and have visual feeback, where s/he is going, and what would be the ideal next step. Also the teacher would get valuable information about the course structure implied by the exercises and exams.

In learning the model structure (dependency graph) the goal is to find such structure \( M \), which maximizes the probability of the training data \( D \) given the structure \( P(D|M) \). In general this is an NP-hard problem (Chickering, Geiger & Heckerman 1994), although the time requirement can be decreased by different heuristics, and the database is anyway relatively small. We can also utilize the expert knowledge in
learning the model structure, and give greater prior probabilities for those structures, which the expert supports, instead of using a uniform distribution.

We have also some minor development ideas: to add the information of the difficulty of the tasks (e.g. by comparing average points got from tasks) and the information, which can be concluded from the exercise sessions (the possible effect can be found by some variation of our hybrid techniques). Intuitively the exercise sessions can have at least two kinds of effects: the students may have weekly variation in their diligence, and on the other hand the tasks belonging to the same exercise session probably measure the same skills (so this information can help in clustering the tasks).

So far we have done only some experiments and used a slow brute force clustering algorithm, but in the future our plan is to implement an effective and usable student/teacher assistant software for general use.

References


Toivonen, H. (1996), Discovery of Frequent Patterns in Large Data Collections, PhD thesis, Department of Computer Science, University of Helsinki.