Discovering Prediction Rules in AHA! Courses

Cristóbal Romero[1], Sebastián Ventura[1], Paul de Bra[2], Carlos de Castro[1]

[1]University of Córdoba, Campus Universitario de Rabanales, 14071, Córdoba, España
{cromero,sventura,cdecastro}@uco.es
[2]Eindhoven University of Technology (TU/e), PO Box 513, Eindhoven, The Netherlands
debra@win.tue.nl

Abstract. In this paper we are going to show how to discover interesting prediction rules in student usage information, and use them to improve adaptive web courses. We have used AHA! to make a course that adapts both the presentation and the navigation depending on the level of knowledge that each particular student has. We have made several changes in AHA! to improve its power in the educational area. We use the logging information from AHA! to discover relations in the usage data (reading times, difficulty levels and test results). The most interesting relations are shown to the teacher so that he can carry out the appropriated modifications in the course to improve it.

1 Introduction

Web-based education is a popular research and application area, thanks to benefits such as independence from physical localization and platform. Especially Web-based Adaptive Educational Hypermedia Systems (Brusilovsky, 98) have shown advantages because they can personalize the course for each specific student. They build an individual user model and they use it to adapt the application to the user. For example, they adapt the content of a hypermedia page to the knowledge or objective of the user, or they suggest the most relevant links to follow. But the methodology used to create and use adaptive courses is "static": when a course (and its adaptation rules) is finished and published on the Internet it is never modified again. The teacher only accesses the students' evaluation information obtained from the course to analyze the students' progress. We propose a dynamic methodology, where the usage information is used by the teacher to find "trouble spots" in the course. The teacher can then make changes that aim to improve the course. (The data gathering and analysis of course has to be repeated to validate that the changes are indeed improvements.) Our approach is to use data mining (Zytkow & Klosgen, 01) in order to discover useful information to help in the decision making processes. Nowadays some researches are beginning to use different data mining techniques in web-learning activities (Zaïne, 2001). We propose an evolutionary algorithm for discovering prediction rules, and compare it to other data mining techniques. We use the students' usage information obtained from a Web-based Adaptive Course (on the topic of Linux). The discovered rules can be shown to the teacher in order to help him decide how the course could be

modified to obtain best student performance. Below we are going to describe our dynamic construction methodology, and then we describe the architecture of the AHA! courses and the modifications we have made. Next we describe the students' usage information obtained from AHA!. Then we introduce the evolutionary algorithms for rule discovery. And finally we describe the implementation of a visual mining tool.

2 Methodology

The dynamic construction methodology of Web-based Adaptive Courses that we propose (Romero et. al, 02) is recurrent, evolutionary and as the number of students who use the system increases, more (logging) information becomes available to the teacher to improve it. In our methodology we can distinguish four main steps:

1. Construction of the course. The teacher constructs the course using information on the domain model and the pedagogic model. He also chooses an interface style. Normally an authoring tool is used to perform this authoring task. The remaining information: a tutor model is usually given by the system itself and the student model is acquired at execution time. Once the teacher and the authoring tool finish the elaboration of the course, then all contents may be published on a web server.
2. Execution of the course. Students take the course using a web browser. In a transparent way the usage information is picked up and stored by the server in the log file of each student.
3. Prediction Rule Mining. After log files have been transferred to a database the teacher can apply the evolutionary algorithms to discover important relationships among the data picked up. We want to discover relations between knowledge levels, times and scores. To do this task, the author can use a generic data mining tool or he can use our specific visual tool (EPRules).
4. Improving the course. The teacher can use the discovered relationships to carry out the modifications that he believes more appropriate to improve the performance of the course. For example he can modify the course's original structure (joining concepts, changing concepts from level or chapter, etc.) and content (eliminating or improving bad questions, bad pages, etc.). To do it, he uses the authoring tool.

3 Architecture of modified AHA!

In order to test our methodology and our algorithms for discovering prediction rules we have developed an adaptive web-course to obtain the necessary usage information. We have chosen AHA! Version 1.0 (De Bra & Ruiter, 01) to build our course because it lets us turn the information into an adaptive application, it stores usage information in log files and the source code is freely available. We had to make some modifications in order to introduce knowledge levels and repetitive multiple-choice tests. We want to adapt the course to each particular user depending on his overall knowledge level. The architecture of the modified AHA! consists of a domain model, user model and adaptation engine.

3.1 Domain model

The domain model in the AHA! system (De Bra & Ruiter, 01) consists of concepts and relations between them. One concept is a component that represents an information abstract element and it has an identification, a set of attribute-value pairs and a set of links. One concept has one or several related web pages. Each web page is a XML file with conditionally included fragments and hypertext links.

The AHA! modified domain model that we are going to use is based on the typical structure of the educational material in which one course consists of several chapters that have several concepts and subconcepts. Another important characteristic that we have added to the AHA! domain model is the classification of concepts by difficulty levels (novice, medium and expert). In this way each chapter can have three different difficulty levels, each one with different concepts.

3.3 User model

The user model in AHA! (De Bra & Ruiter, 01) consists of a set of attribute-value pairs about knowledge, time and visit. In the case of knowledge, each attribute is associated to one concept of the domain model. For each user the system maintains a file in which the values of the attributes are stored. In AHA! the knowledge attribute values are integers between 0 to 100. They show the knowledge level that the student has about that concept and they change as the student visits the concepts. AHA! also has another attribute that indicates if the student has read the concept or not (values 0 or 1). Finally, for each page, AHA stores the time when the user visited the page and the time when the user left the page. (The reading time can be inferred from this.)

Our modified AHA! student model is practically the same with a difference in the way of calculating the knowledge attribute and the values that it can have. In our system we calculate the knowledge variable basing on the obtained score in the question associated to the concepts. In AHA! the value is only increased as the user reads the pages about the concepts. Also, in our system the variable can only have four values: 0 (no level of knowledge. No test done yet.), 10 (low level of knowledge), 50 (medium level of knowledge) and 100 (high level of knowledge).

3.2Adaptation engine

In AHA! (De Bra & Ruiter, 01) the information content adaptation of a hypermedia document and the link adaptation is carried out using a set of rules. These rules specify the relation structure between the domain model and user model, that is, the personalized presentation generated for each student. The adaptation rules are defined by the author of the course. Applying these rules we will obtain the adaptation that will be shown to the user.

Our modified AHA! adaptation engine uses rules for adapting the presentation and navigation to the specific knowledge level of each student. In AHA! all the content adaptation rules are located in the same XML domain files. In our system we have created a new type of file named chapter evaluation files. These files contain all the

necessary rules to adapt a specific chapter. So, in the XML domain files we have only content without rules, separating in this way the content from the adaptation. The adaptation we have implemented in our system is: if the level of knowledge of the student in a chapter is high then the content will be shown in a high level. If the student increases or decreases his level in the chapter, then the content level will be increased or decreased too. In Figure 1 we show the specific adaptation of a chapter in our system.

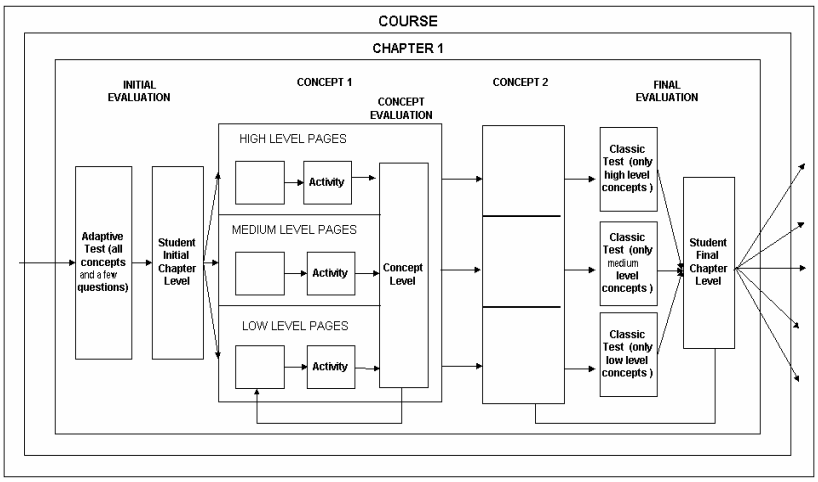

Fig. 1. Adaptation of one chapter in modified AHA!.

In Figure 1 we can see that in order to know the student knowledge level in each chapter, the student first has to pass an adaptive initial test that determines the initial level to be shown in the chapter. Next, the system obtains the progression through the activities and finally it obtains the final level through a classic final test. In order to finish a chapter the student has to reach the expert level, that is, if initially the student obtained a low level, he would have to first reach the medium level and then the expert level. When the student finishes a chapter in expert level, he can go to the next chapter and the process starts again (firstly doing an initial test, next the activities and then the final test).

5 Usage information

The usage information we are using for discovering prediction rules is the information picked up from the courses developed with the modified AHA! system. The AHA! system stores the usage information in two log files (access log and user model). We have added one more log file, the *test* file to store the scores of the questions (activities and tests). Before applying our prediction rule algorithm we move all the information stored in the log files to a database. This makes it easier to perform data mining over multiple students' log data and also makes this much more efficient. During this translation process we carry out some pre-processing of the data (attribute

selection, noise elimination, etc.). The three sets of log files (logs, model and test) are converted to three tables of a database:

–Times. This table is created from the log file and it contains information about the XML pages (content, question, etc.) and the time at which the student has accessed them.

–Levels. This table is created from the model file and it contains information about the knowledge level (high, medium, low) that the student has for each concept of the course.

–Success. This table is created from the test file and it contains information about the success or failure of the students on the questions (tests or activities).

6 Prediction Rule Mining

We have used an evolutionary algorithm to obtain prediction rules (Noda et. al, 99) from the usage data of a web-based adaptive course. The prediction rules show important dependence relations between data that can be used for decision making. The general form of a prediction rule is:

$$\text{IF } \text{Cond}_1 \text{ AND } \ldots \text{ Cond}_n \ldots \text{ AND } \text{Cond}_m \text{ THEN Pred} \quad (1\mathbf{1})$$

This kind of rule consists of two parts. The rule antecedent (the IF part) contains a conjunction of m conditions on values of predictor attributes, whereas the rule consequent (the THEN part) contains a prediction about the value of a goal attribute. We have chosen this type of knowledge representation because it is intuitively clear for most users.

On the other hand evolutionary algorithms are a paradigm based on the Darwin evolution process, where each individual codifies a solution and evolves to a better individual by means of genetic operators (mutation and crossover). In general the main motivation for using evolutionary algorithms for rule discovery is that they perform a global search and cope better with attribute interaction than greedy rule algorithms often used in data mining (Freitas, 02). Most data mining methods are based on the rule induction paradigm, where the algorithm usually performs a kind of local search (hill climbing). Also, the fitness function in evolutionary algorithms evaluates the individual as a whole, i.e. all the interactions among attributes are taken into account. In contrast, most rule induction methods select one attribute at a time and evaluate partially constructed candidate rules, rather than full candidate rules. The Genetic Process we have used consists of the following steps (Michalewicz, 96): the first step is Initialization; next Evaluation, Selection and Reproduction steps are repeated until the Finalization condition is fulfilled (see Figure 2).

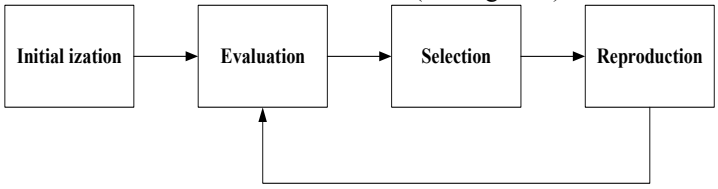

Fig. 2. Evolutionary Process.

–Initialization consists of generating a group of initial rules specified by the user. They are generated starting from the most frequently occurring values in the database. We use a Michigan approach in which each individual (chromosome) encodes a single rule. We use encoding values in which a rule is a linear string of conditions, where each condition is a variable-value pair. The size of the rules is dynamically depending of the number of elements in antecedent, and the last element always represents the consequent. The generic format of the rules we are going to discover in Backus Naur Form (BNF) is:

<rule>::= (<antecedent> <consequent>) (**2**)
<antecedent>::= <condition> | <antecedent>
<consequent>::= <condition>
<condition>::= <attribute> <operator> <value>
<attribute>::= <type>.<name> | <type>.<name>(<number>)
<type>::= TIME | SUCCES | LEVEL
<name>::= STRING
<number>::= INTEGER
<operator>::= ≤ | ≥ | =
<value>::= INTEGER | YES | NO | LOW | MEDIUM | HIGH

–Evaluation consists of calculating the fitness of the current rules and keeping the best ones (exceed a specified minimum value). Our evaluation function can use the measures directly chosen by the user instead of using only the typical confidence and support. We have implemented a lot of rule evaluation measures (Tan & Kumar, 00) that arise from statistics, machine learning and data mining areas (certainty factor, correlation, interest, Laplace, conviction, Gini, Pearson, etc.). The rules that exceed a minimum value specified for the chosen measure are directly stored in the final rule vector. This rule vector is shown to the user at the end of the evolutionary process.

–Selection chooses rules from the population to be parents to crossover or mutate. We use rank-based selection that first ranks the population and then every rule receives fitness from its ranking. The worst will have fitness 1, second worst 2, etc. and the best will have fitness N (where N is the number of rules in the population). Parents are selected according to their fitness. With this method all the rules have a chance to be selected.

–Reproduction consists of creating new rules, mutating and crossing current rules (rules obtained in the previous evolution step). Mutation consists of the creation of a new rule, starting from an older rule where we change a variable or value. We randomly mutate a variable or values in the consequent or antecedent. Crossover consists of making two new rules, starting from the crossing of two existing rules. In crossing the antecedent of a rule is joined to the consequent of the another rule in order to form a new rule and vice versa (the consequent of the first rule is joined to an antecedent of the second). So it is necessary to have two rules to do the crossover.

–Finalization is the number of steps or generations that will be applied to the genetic process. We could also have chosen to stop when a certain number of rules are added to the final rule vector.

7 Implementation and Results

We have developed a specific visual tool (EPRules: Educational Prediction Rules) in Java to facility the process of discovering interesting prediction rules (see Figure 3). This tool is oriented to the teacher so it is more easy to use, more comprehensible and more intuitive than other current generic data mining tools.

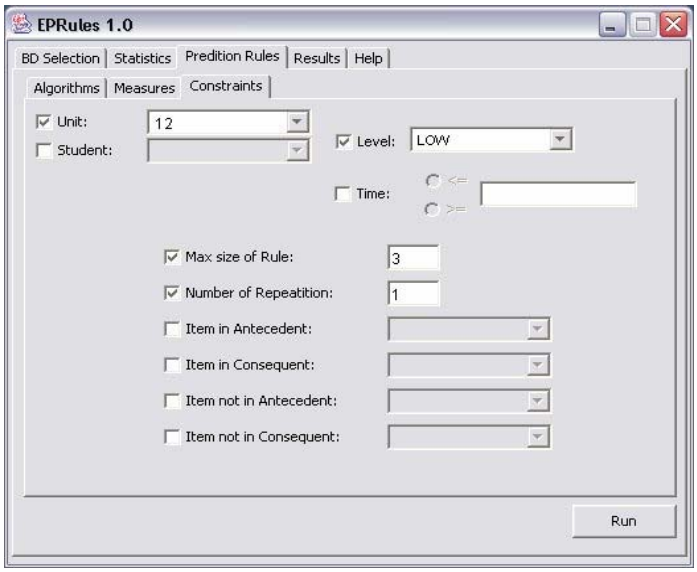

Fig. 3. Visual Tool for discovering prediction rules.

The main tasks of EPRules are:

–Selection and Preprocessing. You can select an existing database or the AHA! log files you want to preprocess and convert them to the database of usage information.

–Statistics. You can select the basic statistics (maximum, minimum, mean, mode, etc.) of the times, scores and levels obtained by students in the pages and questions of the course.

–Prediction rules. You can select the parameters of the evolutionary algorithm (population size, crossover and mutation probability, number of evolutions, etc.), the measures to use (support, confidence, interest, etc.) and constraints (unit, student, level, time, etc.) to apply in the prediction rules.

–Results. You can visualize the statistics (in numerical and graphical way) and the prediction rules (antecedent, consequent and measures values).

We have used the data picked-up from a Linux course developed with the modified AHA!. The course has been used by 50 students in Computer Science from the University of Cordoba (Spain).

To test our evolutionary algorithm we have compared it with an Apriori algorithm (Agrawal & Srikant, 94) that is a classical algorithm for discovering rules. The specific parameters used for the two algorithms are shown in Table 1.

Evolutionary Algorithm	Apriori Algorithm
Size of rule = 3	Size of rule = 3
Min Support = 0.8	Min Support = 0.8
Min Chose-Measure = 0.8	Min Confidence = 0.8
Size of population = 100	
Number of evolution = 50	
Crossover Probability = 0.8	
Mutation Probability = 0.2	

Table 1. Parameters of Evolutionary and Apriori algorithms.

We have compared the Apriori algorithm that only uses confidence and support with our evolutionary algorithm using one different measure each time (Figure 4).

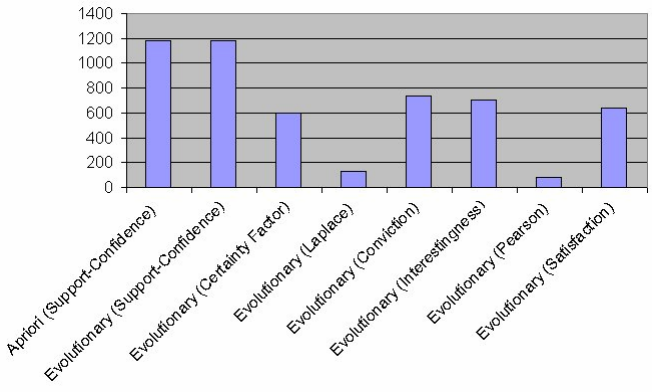

Fig. 4. Evolutionary versus Apriori algorithm.

We have compared the execution time and the number of discovered rules, and the Apriori algorithm was always faster and it found many more rules than our evolutionary algorithm. This actually is a good result. Evolutionary algorithms are always slower, but the discovery of a small number of rules is a benefit for the author. Due to the "completeness" nature of the Apriori algorithm it discovers all the possible rules and there are a lot of redundant and not-interesting rules. And if there are so many rules it is difficult for the teacher to understand and to identify what the really interesting rules are. This problem has been recognized before and was called "interestingness" (Piatesky-Shapiro & Matheus, 94) and to resolve it the use of a lot of different measures have been proposed. Our solution to this problem is that the teacher can choose the measure he is interested in. In this way all the presented discovered rules will be interesting for him. The objective is to show only a few interesting rules to the teacher. To achieve this we use the chosen measure and constraints. The teacher can specify some conditions or filters that the discovered rules have to fulfill. These conditions are about chapters (in what chapter he is interested), students (in which student or students he is interested in), levels (in which difficulty level he is interested), etc. The discovered rules show relations about three aspects (time, success and level) depending on the type of the condition of the rule consequence:

–Time. It shows conditions (antecedent items) that have an effect on a time condition (consequent item).
–Level. It shows conditions (antecedent items) that have an effect on a level condition (consequent item).
–Success. It shows conditions (antecedent items) that have an effect on a success condition (consequent item).

In Table 2 we show several interesting examples of discovered prediction rules with size 2 and with their support and interestingness measure.

Antecedent	Consequent	Sup	Int
TIME.testi3-0(3)>60	SUCCESS.testi3-0(3)=NO	0.8	0.93
LEVEL.historia1-0=HIGH	LEVEL.intro1-2=HIGH	0.85	0.95
SUCCESS.ftp12-2(1)=YES	SUCESS.telnet11-1(2)=YES	0.85	0.91
LEVEL.testf14-1=LOW	LEVEL.testf6-1=LOW	0.8	0.93

Table 2. Example of Discovered Prediction Rules.

First, we have to know the exact meaning of the nomenclature we have used to name the concepts so we can understand the rules correctly. The name of a concept consist of the page name (the string after the dot), the chapter number (after the page name), slash delimiter, the level number (0 low, 1 medium and 2 high) and the question number in brackets if it is a question, not a content page. The first rule shows that many students have problems (time and failure) with question 3 of the initial test in chapter 3, so the teacher should modify it. The second rule shows a relation between two concepts of the same chapter (1) but at a different level (0 and 2), so the teacher should put them together in the same level. The third rule shows a relation between two questions of different chapters, so the teacher should verify that they are really different questions. And finally the fourth rule shows a relation between the final levels of two chapters, so the teacher can add a direct link between these two chapters if there isn't one yet.

7. CONCLUSIONS

In this paper we have described a methodology to develop and improve web-based adaptive hypermedia courses using the AHA! system as their base. We have modified the AHA! functionality to specialize it in an educational environment. We have added a knowledge level (low, medium and high) so that we can adapt the presentation and navigation of the course to the level of each student in each chapter. But our objective was to be able to improve the performance of the developed courses by applying evolutionary algorithms to discover prediction rules among all the usage data. These rules are shown to the teacher so he can decide what changes to make to the course.

We have tested our evolutionary algorithms with the data obtained from the Linux course and we have shown that they can produce potentially useful results. We have compared it with a classic algorithm for discovering rules (namely Apriori) and we have shown that our evolutionary algorithm is a good alternative for extracting a small set of interesting rules, which is very important for decision-making. To achieve

the small number, we have used constraints and measures that the teacher selects because they are more important to him.

We are now developing a more sophisticated evolutionary algorithm with multiobjective optimization techniques (Coello, 99), to obtain more interesting rules. The idea is to use not only the support and other measure for finding the rules but also to use several different metrics at the same time. We are also developing a new evolutionary algorithm using syntax restricted genetic programming. We have shown that the rules can directly be represented like a free context grammar, so we can implement them easily. Using this type of representation the form of the rules is more flexible and modifiable.

REFERENCES

1.Agrawal R., Srikant R. (1994) Fast Algorithms for Mining Association Rules. Conf. on Very Large Databases. Santiago de Chile.

2.Brusilovsky, P. (1998) Adaptative Educational Systems on the World-Wide-Web: A Review of Available Technologies. Conf. on ITS. San Antonio.

3.Coello, C.A. (1999) An Updated Survey of Evolutionary Multiobjetive Optimization Techniques: State of the Art and Future Trends. IEEE pp. 3-13.

4.De Bra P., Ruiter J. (2001). AHA! Adaptive Hypermedia for All. Conf. on WebNet. Orlando, Florida.

5.Freitas, A. A. (2002) A survey of evolutionary algorithms for data minig and knowledge discovery. Advances in Evolutionary Computation. Springer-Verlag.

6.Michalewicz, Z. (1996) Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York

7.Noda E., Freitas A., Heitor S.L. (1999) Discovering Interesting Prediction Rules with a Genetic Algorithm. Conf. on Evolutionary Computation. Washington DC.

8.Piatesky-Shapiro G., Matheus J. (1994) The interestingness of deviations. Proceedings of the AAAI-94 Workshop on KDD. Seattle, Washington.

9.Romero, C., De Castro, C., Ventura, S. (2002). Using Genetic Algorithms for Data Mining in Web-based Educational Hypermedia System. Workshop on Adaptive Systems for Web-based Education. Malaga.

10.Tan, P., Kumar V. (2000) Interesting Measures for Association Patterns: A Perspective. TR00-036. Department of Computer Science. University of Minnesota.

11.Zaïne, O. R. (2001) Web Usage Mining for a Better Web-Based Learning Environment. Conf. on Advanced Technology for Education, Banff, Alberta.

12.Zytkow J., Klosgen W. (2001) Handbook of Data Mining and Knowledg Discovery. Oxford University Press.